FEATURE RANKING FOR STRUCTURED OUTPUT PREDICTION

Matej Petković

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Matej Petković

FEATURE RANKING FOR STRUCTURED OUTPUT PREDICTION

Doctoral Dissertation

RANGIRANJE ZNAČILK ZA NAPOVEDOVANJE STRUKTURIRANIH VREDNOSTI

Doktorska disertacija

Supervisor: Prof. Dr. Sašo Džeroski

Co-Supervisor: Dr. Dragi Kocev

Ljubljana, Slovenia, September 2020

To my parents
Mojim staršem

Ko sem faks si iskal, sem gozdarstvo izbral
(D. Domajnko, Gozdar)

Acknowledgments

I thank my supervisor Sašo Džeroski for giving me the opportunity to work here, for giving support and general directions for research, and for thorough corrections of our papers.

I thank Marko Bohanec for also giving me the opportunity to work here.

I thank my co-supervisor Dragi Kocev for his help with writing the papers, for general lessons, for all the time has spent with me, and for other opportunities he gave me.

I thank Panče Panov for his help at the very beginning when the help is needed most.

I thank the masters of the grid – Barbara Krašovec, Janez Srakar and Jan Jona Javoršek – for their close-to-zero response times, no matter where they were at that time or when the issues occurred, for the environments that Barbara has created, and for adjusting the grid parameters if we were in a hurry. I also apologise for bringing a cluster or two down once or twice.

I thank Mili Bauer for explaining many things at the beginning, for solving bureaucratic problems in an elegant way, and for sweets.

I thank Nikola Simidjievski for various thoughts and lessons.

I thank Jurica Levatić for his love for space, which has caused many great things to happen.

I thank Tome Eftimov for the opportunities to work and write together.

I thank Kristian Kersting for accepting me into his group. That were three really pleasant months in Darmstadt.

I thank Martin Breskvar, Tomaž Stepišnik, Jure Brence, Blaž Škrlj, Ilin Tolovski, Ana Kostovska, Jasmin Bogatinovski and Laura Hughes for fun, ideas, raised question and jokes – either at lunch or in the office.

I thank Maša Matijašević for spotting all the mistyped words, redundant commas, and other mistakes.

Finally, I thank the evaluation board – Nada Lavrač, Hendrik Blockeel and Kristian Kersting – for taking their time, and for corrections that improved the thesis.

Merci to all of you.

Abstract

In this thesis, we develop feature ranking methods for a variety of learning settings, bridging the gap between the ever more complex data on the one hand, and the lack of feature ranking methods that would explain the models learned on these data on the other hand. The developed feature ranking methods address complex machine learning tasks from supervised, semi-supervised and unsupervised learning: supervised and semi-supervised structured output prediction (SOP), including multi-target regression (MTR), multi-label classification (MLC), and hierarchical multi-label classification (HMLC). We also extend feature ranking methods to relational learning, where the data representation is even richer and more complex. The feature rankings produced by our methods offer state-of-the-art performance in all the learning contexts, as showcased by extensive empirical studies presented in this thesis.

The developed feature ranking methods handle various learning contexts in an elegant and unified way. More precisely, we adapt two groups of feature ranking methods. The first group of feature ranking methods is tree-ensemble-based and contains the Symbolic, Genie3 and Random Forest scores. The scores are computed from different ensembles of predictive clustering trees (PCTs), i.e., random forests, bagging, and ensembles of extremely randomized PCTs. The second group of methods are distance-based methods that follow the Relief approach to feature ranking.

In the thesis, we first give the necessary background and present the main ideas behind the developed feature ranking methods. Then, extensive empirical studies are presented, evaluating the developed methods for each learning setting. The main empirical findings are that the proposed feature ranking methods yield meaningful rankings and outperform the existing methodology. Especially when the number of features is large, ensemble-based methods outperform the distance-based ones. Both groups of methods scale-up well, since they are subquadratic in the number of features and easily parallelizable.

The thesis also presents a practically relevant case study that uses the developed ensemble-based feature ranking algorithms for MTR to explain the models that predict the thermal power consumption of Mars Express spacecraft of the European Space Agency (ESA). We finish the thesis with a comprehensive discussion of its contributions and an outline of many possible directions for further work.

Povzetek

V disertaciji razvijemo metode za rangiranje značilk, namenjene različnim učnim scenarijem. S tem premostimo razkorak med vse kompleksnejšimi podatki in mankom metod za rangiranje značilk, ki bi pojasnile modele, naučene na takih podatkih. Razvite metode naslavljajo kompleksne naloge strojnega učenja s področja nadzorovanega, polnadzorovanega in nenadzorovanega učenja: večtarčno regresijo, večoznačno klasifikacijo in hierarhično večoznačno klasifikacijo. Metode za rangiranje značilk razširimo tudi na področje relacijskega učenja, kjer je predstavitev podatkov še bogatejša in kompleksnejša. V vseh naštetih učnih scenarijih je rangiranje značilk, dobljeno z razvitimi metodami, med najboljšimi v primerjavi z drugimi sodobnimi metodami, kar pokažemo z obsežnimi empiričnimi študijami.

Razvite metode za rangiranja značilk obvladujejo različne učne scenarije elegantno in poenoteno. Razvijemo dve skupini metod. V prvi, ki vsebuje metode, temelječe na ansamblih dreves, so kazalci imen Symbolic, Genie3 in Naključni gozd. Te kazalce izračunamo iz različnih ansamblov dreves za napovedno razvrščanje (DNR-jev): naključnih gozdov, vrečenja in ansamblov skrajno naključnih DNR-jev. V drugi skupini so metode, temelječe na računanju razdalj, ki sledijo pristopu algoritma Relief za rangiranje značilk.

V disertaciji najprej predstavimo potrebno ozadje in podamo zamisli, ki pripeljejo do razvitih metod za rangiranje značilk. Temu sledi predstavitev obsežnih empiričnih študij, ki ovrednotijo razvite metode v vseh učnih scenarijih. Iz teh študij sledi, da predlagane metode za rangiranje značilk proizvedejo smiselna rangiranja in nadvladajo obstoječe metode. Rangiranja, izračunana iz ansamblov DNR-jev, so boljša kot rangiranja, ki temeljijo na računanju razdalj, še posebej pri podatkovjih z visokim številom značilk. Obe skupini metod sta primerni tudi za velika podatkovja, saj nista niti kvadratni, kar se tiče števila značilk, hkrati pa je njune člane tudi lahko paralelizirati.

V disertaciji pokažemo tudi, da so razviti algoritmi za rangiranje značilk v kontekstu večtarčne regresije uporabni tudi v praksi, saj z njimi pojasnimo modele za napovedovanje toplotne energije na satelitu Mars Express Evropske vesoljske agencije. Disertacijo zaključimo z izčrpno debato o doprinosih disertacije ter začrtanimi smernicami za nadaljnje delo.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Laziness is the mother of invention (said Agatha Christie). Therefore, it is no wonder that people are making computers and robots smarter and smarter, so that the machines can do complex tasks for which some level of intelligence is necessary. These include virtual personal assistance (Hoy, 2018), playing Go at a super-human level (Silver et al., 2016), making medical diagnoses (de Bruijne, 2016), and much more.

All these are applications of *machine learning* (Mitchell, 1997), i.e., making machines learn from experience that is typically given in the form of some data. Learning is successful when a computer makes the right decisions in situations that it has not seen during learning. The outcomes of learning depend on the subfield of machine learning that our task or application belongs to. This thesis is mostly concerned with tasks related to predictive modeling, where the goal is to learn a (predictive) model that - given some descriptors (features[1]) of the data - predicts the value of a target variable.

With the advance of computers and data collection, we have entered the era of big data. A typical data set can be large in both the number of examples and the number of features. Having many examples is desirable since more examples, i.e., training, can lead to better models, whereas a large number of features has some undesirable consequences:

1. The algorithms used for constructing models may be too slow or may use too much memory.

2. Among the thousands of features, there are typically just a few that actually matter when making predictions, therefore we are wasting our time and loosing insight by taking into account all of them.

3. The learned models may be too complicated for a human to understand.

All of the problems above can be avoided by computing a *feature ranking*. A feature ranking algorithm assigns an importance (or relevance) score to every feature, typically following some well-motivated heuristic definition of importance, e.g., a feature is important if its values determine the values of the target variable[2]. The output of a feature ranking algorithm are the importance scores, which define the order (ranking) of the features.

Keeping only some top-ranked features (the features with the highest importance) in the data might make model induction much more efficient time- and memory-wise. This special case of using a feature ranking algorithm to reduce the number of features is referred to as feature selection, and completely solves the first two problems mentioned above. It might also solve the third one (as fewer features mean less complex models), but feature selection

[1]In this thesis, the word feature is used as a synonym for descriptor and descriptive attribute. Other meanings are possible, as explained by Fürnkranz et al. (2014).

[2]This is the motivation behind the info-gain score (Cover & Thomas, 2006).

is not the only way to obtain understandability/explainability. For example, the predictions of random forests (Breiman, 2001) or neural networks can be easily explained directly by feature ranking in terms of permutation-based feature importance scores (Breiman, 2001), even if the models are learned with the full set of features.

The most common tasks of predictive modeling are classification (where the target variable can take finitely many categorical values) and regression (where the domain of the target variable are the real numbers). A plethora of feature ranking methods exists that can be used in these two settings (Guyon & Elisseeff, 2003; Stańczyk & Jain, 2015). However, data are becoming more and more complex not only in terms of their size, but also in terms of their types.

The values of target variables are not limited to categorical values or numbers, but can also be vectors or (partially ordered) sets of values. For example, the tasks of multi-target regression (where the target values are vectors of numeric variables), multi-label classification (where the target values are sets of categorical values), and hierarchical multi-label classification (where the target values are partially ordered sets of categorical values) appear naturally. Consider, for example, models that take images as inputs and need to answer questions such as *How many cats and dogs are in the picture?* (the answer is a vector of two numeric values), *Which animals appear in the picture?* (the answer is a set of categorical values) or *Which species, genera and families of animals appear in the picture?* (the answer is a partially ordered set of categorical values).

The above predictive modeling tasks belong to the area of structured output prediction (SOP) (Bakır et al., 2007). There is a growing body of predictive modeling approaches that can tackle such problems, e.g., predictive clustering trees (Blockeel, 1998). However, this is not the case with feature ranking in the context of SOP, even though the characteristics of the descriptive part of multi-target or multi-label (and in general SOP) data sets, e.g., the number of features, are no different than those of classification and regression data sets.

The complexity of the data goes even further. Besides the complexity of the output in predictive modeling, the complexity of data can vary in two additional dimensions. These include the complexity of the input part of the data, on the one hand, and the availability of output target values in training data.

First, let us discuss the input part of the data. In the cases above, one table suffices to represent a data set. There, every row corresponds to an example and every column corresponds to either a feature or a target. Thus, a single table holds descriptions of objects of interest, e.g., patients (if we predict some disease), pictures (if we predict the presence of animals in pictures), etc. However, sometimes, this is not enough. For example, if we are trying to predict the category/genre of a movie, the features describing a movie (release date, title, etc.) might not be very informative and it is beneficial to know also something about people that see the movie (e.g., their age, gender, etc.). In that case, we need more than one table, e.g., a table for people, and some relations among those tables (e.g., *hasSeenMovie(person, movie)*, etc.). The number of features that can be generated from such *relational data* is typically very high, further emphasizing the need for feature ranking and selection.

Second, classification, regression, etc. are (if not specified otherwise) *supervised learning tasks*, where the value of the target variable is available for each example. Sometimes, labeling all the examples with their target values is too expensive, e.g., when testing chemical compounds (DiMasi et al., 2003), or would demand too much time since it has to be done manually, e.g., determining whether a tweet is written in a positive or negative tone (Kralj Novak et al., 2015). When the target values are missing for some, or even most of the training examples, the predictive modeling tasks at hand are referred to as semi-supervised learning tasks. The unlabeled data might still be of use when constructing a predictive model or computing a feature ranking.

In the extreme case, when no target values are present, i.e., when there is no target variable, we speak about *unsupervised learning*. In this setting, the most common machine learning task is clustering, where the aim is to find well-defined subgroups of data, called clusters. In the feature ranking for unsupervised learning, we cannot say how relevant a feature is for the target, however, we can determine how relevant the feature is for the produced clustering, or how redundant the feature is in the context of the others.

1.1 Motivation

The motivation for this thesis is the gap between the ever more present high-dimensional data sets with additional degrees of complexity on the one hand, and the lack of feature ranking methods applicable to such data sets on the other hand. Under additional degrees of complexity, we mean

- different types of structured outputs, i.e., target variables, in supervised learning, giving rise to tasks such as multi-target regression, multi-label classification, and hierarchical multi-label classification,
- the partial lack of example labels in semi-supervised learning for classification and regression on one hand, and different tasks of SOP, such as multi-target regression, multi-label classification, and hierarchical multi-label classification on the other hand,
- the complete lack of labels/targets in unsupervised learning, and
- the presence of multiple interconnected relations in relational learning.

For all (semi-)supervised learning tasks, the motivation for computing a feature ranking is the same as described in detail in the introductory parts of this section. For the cases of classification and regression, we want to determine the influence of the features on the target, i.e., their relevance for predicting the target. The motivation is similar for unsupervised learning, where we want to determine the influence of the features on the clustering of the data.

The above three dimensions of complexity, i.e., type of input (relational vs. propositional/tabular), type of output (classification/regression, different types of structured output values) and degree of supervision (unsupervised, semi-supervised and fully supervised) can be considered almost independently (except that unsupervised learning means no output values). Their combinations give rise to a wide variety of machine learning tasks, such as fully supervised classification in a relational setting and semi-supervised multi-target regression in a propositional setting. For the majority of these, especially supervised SOP and semi-supervised learning, the corresponding feature ranking tasks had not been addressed at all at the time the research for this thesis started.

Feature ranking methods did exist for classification and regression, and had started emerging for multi-label classification. There were also a number of methods for unsupervised feature ranking. However, the different existing methods were based on heterogeneous ideas and did not approach the task of feature ranking in a unified manner.

1.2 Goals and Hypotheses

The overall goal of this thesis is to approach the feature ranking tasks corresponding to the variety of machine learning tasks outlined above in a coherent manner, based on the same unifying principles. The approaches we propose are based on two approaches to feature ranking for classification and regression that are general enough to be extended to the various machine learning tasks. These are the feature ranking methods embedded in ensembles of classification and regression trees (Breiman, 2001), and methods for distance-based feature ranking from the Relief family of algorithms (Robnik-Šikonja & Kononenko, 2003). When designing the novel methods, we had the following goals in mind:

- Extending and developing the existing ensemble- and distance-based feature ranking methods to the tasks of multi-target regression (MTR), multi-label classification (MLC), and hierarchical multi-label classification (HMLC),
- Further extending those method and developing feature ranking methods for the semi-supervised learning (SSL) counterparts of the above supervised SOP tasks, as well as for unsupervised learning,
- Extensively evaluating the newly proposed methods by using standard benchmark data sets and comparing the new methods to the current state-of-the-art,
- Applying the proposed methods to real-world applications/case studies,
- Implementing the methods in a unified framework, which will simplify their use.

Closely related to the above goals, we have formulated and investigated (trying to confirm or disprove) the following hypotheses:

H1: It is possible to extend ensemble- and distance-based feature ranking approaches to the unsupervised feature ranking task, to the tasks of supervised SOP (i.e., MTR, MLC, and HMLC), and to their semi-supervised versions.

H2: It is possible to extend ensemble-based approaches to classification and the corresponding feature ranking methods to handle relational data.

H3: The proposed ensemble- and distance-based approaches yield relevant and state-of-the-art feature rankings for MTR.

H4: The proposed ensemble- and distance-based approaches yield relevant and state-of-the-art feature rankings for MLC.

H5: The proposed ensemble- and distance-based approaches yield relevant and state-of-the-art feature rankings for HMLC.

H6: For unsupervised, MTR, MLC and HMLC problems, the ensemble-based feature ranking approaches on average outperform the distance-based approaches to feature ranking when the number of features is extremely high.

H7: The proposed ensemble- and distance-based feature ranking approaches yield relevant and state-of-the-art feature rankings for feature-ranking problems in SSL for classification.

H8: The proposed ensemble- and distance-based feature ranking approaches yield relevant and state-of-the-art feature rankings for feature-ranking problems in SSL for different types of SOP.

H9: The proposed ensemble-based and distance-based approaches yield relevant and state-of-the-art feature rankings for unsupervised feature-ranking problems.

H10: The proposed ensemble-based approaches to relational classification yield state-of-the-art performance as well as relevant feature rankings.

1.3 Methodology

First, we design and implement the planned extensions of ensemble-based and distance-based feature ranking approaches, from the classification and regression setting to the variety of new learning settings. Second, using a plethora of benchmark data sets, we extensively evaluate the newly developed feature ranking methods, using appropriate metrics for assessing their performance. Third, we compare the performances of different feature ranking methods by applying appropriate statistical tests. Finally, in collaboration with domain experts, we address a practically relevant case study of predicting thermal power consumption of the different subsystems of a spacecraft, where we need both predictive modeling and feature ranking in the context of multi-target regression.

1.3.1 Development of Novel Feature Ranking Methods

As mentioned above, we have developed two groups of methods. The first group of methods are ensemble-based feature ranking scores that are computed from ensembles of predictive clustering trees. We define and implement the Symbolic (Petković, Kocev, et al., 2020), Genie3 (Huynh-Thu et al., 2010), and Random Forests (Breiman, 2001) scores, computing them from Bagging (Breiman, 1996; Kocev et al., 2013), Random Forest (Breiman, 2001; Kocev et al., 2013) and Extremely Randomized (Geurts et al., 2006; Kocev et al., 2020) tree ensembles. We provide the appropriate definition of the score for each structured-output prediction task (MTR, MLC, HMLC), and each degree of supervision (supervised, semi-supervised, unsupervised) separately.

The second group of methods are distance-based scores, mostly based on the Relief family of algorithms (Kira & Rendell, 1992; Robnik-Šikonja & Kononenko, 2003). Here, we need to find an appropriate adaptation of the distance definition in the target space (or get rid of it in the unsupervised case), for each of the tasks. Especially for MLC, we have many different options for defining the distance in the target space (Petković, Kocev, et al., 2018), which correspond to the many different performance measures used in MLC.

All the above feature ranking methods are implemented as algorithms incorporated in the Clus software[3] and made publicly available.

1.3.2 Evaluating the Quality of Feature Rankings

To evaluate the performance of a feature ranking approach on unseen data for a given data set, we proceed as follows. We use the train-test split of the data if it is provided, and 10-fold cross validation otherwise. In any case, we compute a feature ranking from the training part of the data, and use both training data and this ranking in a k-nearest neighbors (kNN) predictor evaluating its performance on the test part of the data.

More specifically, we evaluate a feature ranking via the performance of the k-nearest neighbors (kNN) predictor that uses the feature importance estimates to define feature weights in its distance function (Cunningham & Delany, 2007; Wettschereck, 1994). In addition to the simplicity of the kNN model, it allows for a time-efficient and unifying evaluation over all the different learning tasks, except for the unsupervised case. Finally, using this approach, a feature ranking can be evaluated directly with no need for feature-selection-like evaluation approaches (Slavkov, 2012).

[3]http://source.ijs.si/ktclus/clus-public

1.3.3 Comparing the Performance of Feature Ranking Methods

The proposed feature ranking methods are compared through statistical tests of the differences in the quality of feature rankings, performed in a well-established and computationally cheap frequentist approach, as suggested by Demšar (2006). Since the performance of the kNN models that use the computed feature importances in their distance definition are not directly comparable across different data sets, all evaluated feature rankings are first assigned their rank with respect to their performance on each data set: the best performing is assigned rank 1, the second-best is assigned rank 2 etc. These ranks then serve as the input of Friedman's statistical test, which determines whether the differences among the average ranks of the feature rankings (across different data sets) are statistically significant or not. If they are, a *post hoc* test can be used to determine where these differences occur.

In the case of one-versus-all comparison, an appropriate choice for a *post hoc* test is the Bonferroni-Dunn test (Demšar, 2006). This is used when a proposed feature ranking algorithm is compared to many baselines. In the case of an all-versus-all comparison, the Nemenyi *post hoc* test (Nemenyi, 1963) is used. Both tests control for the family-wise false-discovery rate.

These tests are also appropriate since their results can be easily presented with an average rank diagram, where we plot the algorithms to a real line so that their coordinate equals their average rank. The algorithms that are not statistically significantly different in terms of performance are additionally connected by a line. The down-side of this approach is that the used tests may not be the strongest. Another possible approach is to simply use the paired Wilcoxon's test and control for the family-wise false discovery rate with the Benjamini-Hochberg procedure (Benjamini & Hochberg, 1995): We apply the latter approach in some of our experiments.

Note that recently, a new Bayesian statistical comparison method has been proposed (Corani et al., 2017). However, it can be used only for comparing two algorithms, depends on many hyper-parameters, is computationally intensive, and thus not that suitable for practical use than the approaches described above.

1.3.4 A Case Study of Predicting Thermal Power Consumption of the Mars Express Spacecraft

In addition to showing that the newly developed feature ranking methods achieve state-of-the-art performance on standard benchmark data sets, we proved their usefulness in a practically relevant case study that was executed through a collaboration with domain experts from the European Space Agency (ESA). The task at hand was to predict the thermal power consumption of the 33 heaters of the ESA's Mars Express spacecraft (Chicarro et al., 2004). To solve this task, we developed accurate MTR models (including random forests of predictive clustering trees) and obtained the Genie3 feature rankings embedded in these models. The rankings explained the learned models and gave the operators of the spacecraft additional insights into the thermal dynamics of the spacecraft under different operating conditions (Petković, Boumghar, et al., 2019; Petković, Lucas, et al., 2019).

1.4 Contributions

The scientific contributions of the dissertation are summarized as follows:

- Two novel groups of feature ranking algorithms, a group of ensemble-based and a group of distance-based algorithms, for different kinds of supervised SOP tasks: MTR, MLC and HMLC.

- Two novel groups of feature ranking algorithms, a group of ensemble-based and a group of distance-based algorithms, for semi-supervised learning in the context of different SOP tasks: MTR, MLC and HMLC.
- Two novel groups of feature ranking algorithms, a group of ensemble-based and a group of distance-based algorithms, for unsupervised learning.
- A novel ensemble-based method for feature ranking in relational learning/classification.
- Extensive empirical evaluation of the developed methods across benchmark problems from various domains.
- A case study using the developed MTR ensemble-based algorithms both for prediction and feature ranking on a practical problem from the domain of spacecraft operations modeling.

The developed algorithms, their evaluations and the case study have been published as conference and journal publications that are listed in the Bibliography section at the end of this thesis.

1.5 Organization of the Thesis

The remainder of the thesis is organised as follows. Chapter 2 provides the necessary background information. This includes material on the different learning settings considered, covering unsupervised, supervised and semi-supervised learning (for SOP), as well as on feature ranking. The two most related approaches that we build upon are discussed: Tree ensembles for SOP and distance-based feature ranking. Chapter 3 briefly presents the key ideas behind the developed methods for feature ranking in the different learning settings: supervised structured output prediction, unsupervised learning and semi-supervised learning, including classification, regression and structured output prediction. The chapter concludes with the key ideas behind the developed methods for feature ranking for relational classification.

The contributions of the thesis are described in detail in Chapters 4 to 9. The implementations of the proposed methods and their extensive experimental evaluation are presented, as well as the case study. Each chapter includes one or two papers, either published or under review, where the contributions were originally presented, preceded by a summary with a short statement of the contributions. Chapter 4 treats feature ranking for multi-target regression, while Chapter 5 deals with feature ranking for multi-label classification and hierarchical multi-label classification. Chapter 6 covers feature ranking for semi-supervised learning and Chapter 7 covers feature ranking for unsupervised learning. Chapter 8 presents the case study of predicting thermal power consumption for the Mars Express spacecraft. Chapter 9 introduces feature ranking in relational learning. Finally, Chapter 10 concludes the thesis. It also discusses some directions for further work.

Chapter 2

Background

In this chapter, we describe the considered data representations (Section 2.1) and background machine learning tasks of predictive modeling and unsupervised learning (Section 2.2). We then proceed to formal introduction to feature ranking (Section 2.3), and – in our case, another closely related topic – predictive clustering (Section 2.4).

2.1 Data Representation

Before giving an introduction to the developed methods and tasks that they solve, we have to describe the corresponding data representations. In this thesis, we consider two of them: tabular/propositional data and relational data. Even though tabular data can be seen as a special case of relational data, the tabular data is presented separately, since it is considerably simpler.

2.1.1 Tabular Data

The most commonly used data format is (still) the tabular format where a data set is given as a single table, e.g., Table 2.1. There, the header gives the names of the features (in the example table, these are x_i, $1 \leq i \leq n$), and the name of the target attribute $\boldsymbol{y}$ if the target attribute is present.

Table 2.1: Each example $\boldsymbol{x}$ is described by a vector of numeric and/or nominal feature values $\boldsymbol{x}$. Some data sets contain also the target column $\boldsymbol{y}$ whose type (numeric, nominal, vector of numeric, set of nominal, etc.) depends on the task at hand.

x_1	x_2	$\cdots$	x_n	$\boldsymbol{y}$
3.142	`true`	$\cdots$	21	y_1
		$\cdots$		
2.718	`false`	$\cdots$	12	y_m

The rest of the rows describe examples, e.g., the first example is a pair $(\boldsymbol{x}, y_1)$ that consists of the descriptive part $\boldsymbol{x} = (3.142, \texttt{true}, \dots, 21)$, i.e., the values of the features x_i for this example, and the target part y_1, i.e., the value of the target attribute.

All the features are either numeric, i.e., their domain is a subset of $\mathbb{R}$, or nominal (categorical), i.e., they can take finitely many predefined values. For example, feature x_1 in the example table is numeric, whereas feature x_2 is nominal and can take the values `true` and `false`. The considered types of the target variable $\boldsymbol{y}$ – if it is present – are given later in Sections 2.2.2 and 2.2.3. The case when the target variable is not present in the data is described in Section 2.2.1.

2.1.2 Relational Data

The relational data format is an extension of the tabular format to the case where data is given in more than one table, and the tables are connected via some relations. Figure 2.1 gives a specific example of such a data set. This is the IMDB dataset, available at http://grouplens.org/datasets/hetrec-2011/. The IMDB dataset consists of five tables, each of them describing objects of a particular type (e.g., movies, ratings, users, etc.). The tables are interconnected with relations which are represented by lines. A single end of a line corresponds to one-to- (every rating belongs to a single movie) or -to-one (every rating is authored by a single user) relations. A multiple-end of a line corresponds to many-to- (a user might author many ratings) or -to-many (many ratings belong to a movie) relations. For this particular data set, the goal is to predict the value of the category column in the target table Movies (shown in bold).

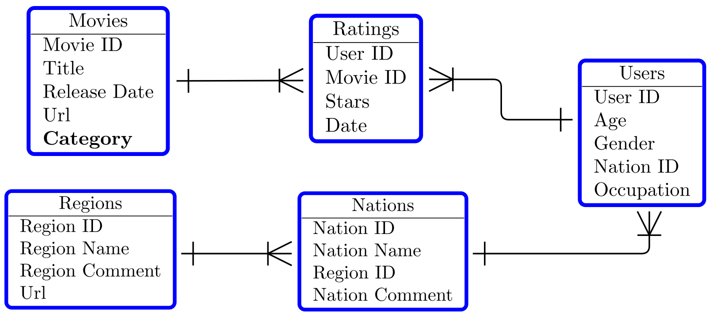


Figure 2.1: An example schema of the IMDB relational data set.

Now, every table in the data set describes objects of a particular type. For example, the Movies table describes movies (their features are title, release date, and url), the Ratings table describes ratings (their features are stars and date), etc. As in the tabular case, some columns in the tables are considered features (e.g., those mentioned in the previous paragraph), and there is one column in one of the tables (called the target table) that corresponds to the target variable. In the specific case above, this is the Category column (genre) in the Movies table.

Additionally, some columns simply provide identification (ID) of an object. For example, the Movie ID column identifies a movie, whereas a pair (User ID, Movie ID) identifies a rating, since every user can rate a given movie only once. ID columns appear in all tables and implicitly define the relations among the tables, such as `liesIn(region, nation)` or `ratedBy(user, movie)`.

As in the tabular case, the target column need not be present, and can be – if present – of several possible types. Thus, all of the tasks described in Section 2.2 for single-table data can also be defined for relational data.

This was a brief introduction to relational data. It only provided the essentials. Additional details, including how to convert such a representation into an equivalent format that is useful for relational learning algorithms, are given in Section 3.3.

2.2 The Basic Learning Tasks

The main topic of the thesis is the task of feature ranking. However, this task is tightly coupled with the tasks predictive modeling and clustering, which also provide basic motivation for performing feature ranking. Thus, before proceeding to a more formal description of feature ranking, we describe the considered tasks of predictive modeling (classification, regression, multi-label classification, hierarchical multi-label classification), and clustering.

In predictive modeling, one tries to learn the unknown mapping $\boldsymbol{y} : \boldsymbol{x} \mapsto \boldsymbol{y}(\boldsymbol{x})$, given some data (as described in Section 2.1). An approximation $\hat{\boldsymbol{y}}$ of the true mapping $\boldsymbol{y}$ is considered good if it gives accurate predictions $\hat{\boldsymbol{y}}(\boldsymbol{x})$ for the previously unseen examples $\boldsymbol{x}$.

In clustering, there is no target variable and the goal is to find a partition of the data set into groups, such that

- the examples in a group are similar to each other, for every group;
- the examples in any two different groups are not similar.

In the next subsections, we describe the aforementioned tasks. We start with clustering and then proceed to prediction modeling tasks. For all the tasks, the connection to feature ranking is given in Section 2.3.

2.2.1 Unsupervised Learning (Clustering)

In unsupervised learning, the target variable $\boldsymbol{y}$ is not present in the data, so the data contain only the descriptive part of examples. The most common task of unsupervised learning is *clustering* where the goal is to find well-defined relevant groups (clusters) of data. Various approaches exist for creating clusters of data. Some of them solve a problem in several stages, e.g., in the hierarchical clustering algorithms (Blockeel, 1998; Nielsen, 2016) where at the end, one decides where to cut the hierarchy of nested clusters to obtain the final ones. The other algorithms solve the problem directly, maybe the most well-known among them is the k-means algorithm (Arthur & Vassilvitskii, 2007). An example clustering of data into four well-defined subgroups is shown in Figure 2.2.

Clustering is the only unsupervised task that is treated in our work, however, it is not the only one in general. For example, in (association) rule learning (Fürnkranz et al., 2014), the goal can be to find at least some interesting subgroups in the data: the groups do not need to be disjunctive, and their union need not be the whole data set. Interestingness of a subgroup can be quantified, for example, by quantifying the difference between the distribution of the examples in the whole data set, and the distribution of the examples in the subgroup.

2.2.2 Supervised Classification and Regression

The two basic tasks of predictive modeling are classification and regression. In **classification** tasks, we answer questions such as *Which bird is in the picture: A parrot, an owl or a pigeon?* (see Figure 2.3), i.e., the target is a nominal variable with a finite domain $\mathcal{Y}$ of size $|\mathcal{Y}| \geq 2$. The elements of the domain are typically referred to as classes. In the special case of $|\mathcal{Y}| = 2$, e.g., $\mathcal{Y} = \{\texttt{true}, \texttt{false}\}$, the task at hand is binary classification. Otherwise, the task at hand is multi-class classification, e.g., $\mathcal{Y} = \{\texttt{parrot}, \texttt{owl}, \texttt{pigeon}\}$. In any case, every example in a classification data set belongs to a single class, and our goal is to predict the correct one. In the rest of the thesis, we simply refer to both tasks as classification.

A great number of methods exist that solve such tasks (Kotsiantis, 2007). However, it is worth noticing that some methods cannot handle multi-class problems that easily, e.g.,

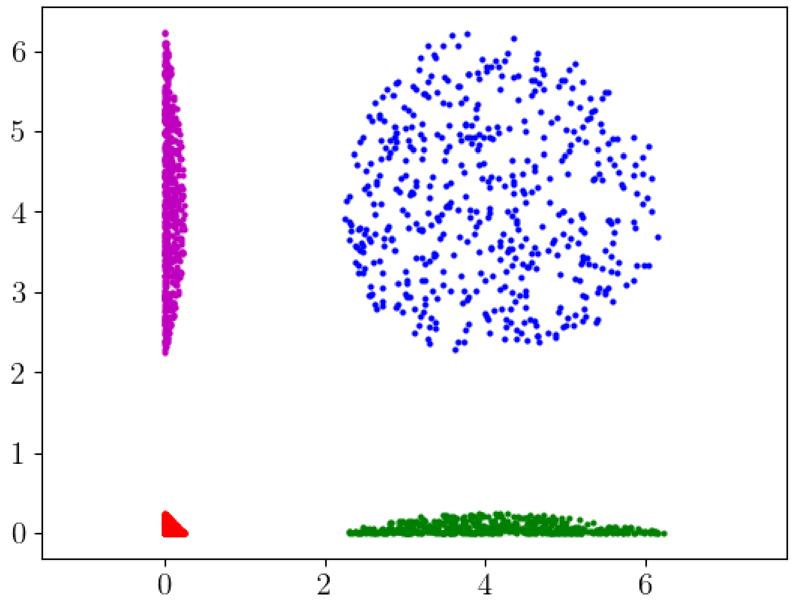


Figure 2.2: An example clustering of a two-dimensional data set. Each cluster is given its own color.

Figure 2.3: An instance from an example classification data set where the goal is to predict which bird (parrot, owl, pigeon) is in the picture.

Figure 2.4: An instance from an example regression data set where the goal is to estimate how many birds are in the picture.

support vector machines (Cristianini & Shawe-Taylor, 2010). The same holds for some evaluation measures. For example, accuracy can be used in all situations, while the F_1 measure (defined in Chapter 6) can be calculated by using macro-averaging of the true/false positives and true/false negatives for all the classes.

The other basic task of predictive modeling is **regression**. Here, we answer questions such as *How many birds are in the picture?* (see Figure 2.4), i.e., the domain of the target variable is a subset of $\mathbb{R}$, and since there are infinitely many possible target values, typically, one cannot expect that the approximation $\hat{\boldsymbol{y}}$ of the true mapping $\boldsymbol{y}$ would give a perfectly accurate prediction for any $\boldsymbol{x}$.

Again, many methods exist that can tackle regression problems (Ryan, 2008). In addition, the task of binary classification can be solved via regression in the following way. The two classes $\mathtt{a}$ and $\mathtt{b}$ are first mapped to 0 and 1, respectively. Then, a regression model is applied and we obtain an approximate mapping $\hat{\boldsymbol{y}}$. After a threshold $\vartheta \in [0, 1]$ is chosen, we predict an example $\boldsymbol{x}$ to belong to class $\mathtt{b}$ if and only if $\hat{\boldsymbol{y}}(\boldsymbol{x}) \geq \vartheta$. A similar procedure of thresholding can be applied for multi-class classification but here, one should resort to multi-target regression which is one of the tasks in structured-output prediction described in the next section.

2.2.3 Supervised Structured Output Prediction

A common name for the numeric and nominal types are primitive data types (International Organization for Standardization, 2007). More interesting or, at least, richer are those that are constructed out of the primitives by adding a vector, a set, or a partially ordered set structure to them[1]. The constructed data type is thus of the type structured (International Organization for Standardization, 2007) and the task at hand is called structured output prediction (Bakır et al., 2007; Kocev, 2011). By considering that the output is a structured datatype, we can answer the following questions (see Figure 2.5):

[1]The axioms of Zermelo-Fraenkel set theory (Hrbacek & Jech, 1999) assure that it suffices to use only sets to represent numbers and the aforementioned structures. However, it is much more user-friendly, for example, if zero (respectively one) is represented as a number 0 (resp. 1) rather than the empty set $\emptyset$ (resp. the set that contains the empty set $\{\emptyset\}$).

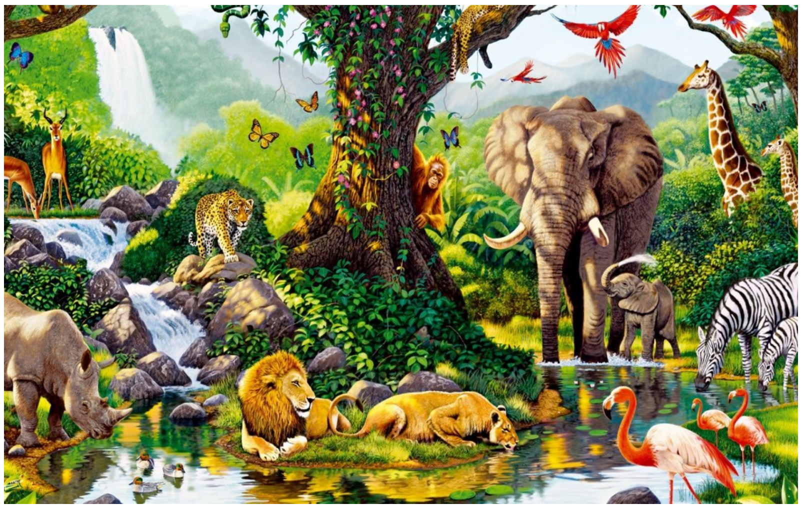


Figure 2.5: An instance from an example structured-output data set where we want to know i) *how many birds, how many mammals, and how many insects are in the picture*, or ii) *Which animal species are in the picture*, or iii) *Which species, genera and families are in the picture.* These questions respectively correspond to the tasks of multi-target regression, multi-label classification, and hierarchical multi-label classification.

1. How many birds, how many mammals, and how many insects are in the picture?
2. Which animal species are in the picture?
3. Which species, genera and families are in the picture?

These questions correspond to the different structured output tasks discussed in the forthcoming subsections.

2.2.3.1 Multi-target Regression

The first question (*How many birds, how many mammals, and how many insects are in the picture?*) can be answered by representing it as a multi-target regression (MTR) task where one predicts the values of a numeric vector, i.e., the target domain $\mathcal{Y}$ is a subset of $\mathbb{R}^t$, where $t \geq 2$ is the number of target components. The condition $t \geq 2$ can be relaxed to $t \geq 1$, so that the (single-target) regression (STR) task becomes a special case of MTR.

As mentioned previously, MTR is one of the means of solving multi-class classification problems where the i-th class is first mapped to the vector $\boldsymbol{e}_i$, where $\boldsymbol{e}_{ij} = 1$ if $i = j$ and $\boldsymbol{e}_{ij} = 0$ otherwise. After an MTR approximation $\hat{\boldsymbol{y}}$ of the true mapping is learned, the final prediction is the i_0-th class where $i_0 = \text{argmax}_i\, \hat{\boldsymbol{y}}_i(\boldsymbol{x})$.

A possible way to approach a MTR problem is problem transformation, which transforms one MTR problem into several STR ones and builds one predictive model for each component of the target vector separately. For this, any of the regression methods can be used. However, by doing so, possible interactions among the components might be overlooked.

Another way to approach a MTR problem and overcome the issue above is algorithm adaptation, i.e., adapting STR methods to handle vectors of numeric values. This has been done, for example, with regression trees (Blockeel, 1998; Kocev, 2011). By building a single model, one can additionally benefit regarding time-complexity and interpretability of models (Kocev & Džeroski, 2013).

2.2.3.2 Multi-label Classification

The second question (*Which animal species are in the picture?*) is one of the multi-label classification (MLC) tasks where one predicts a subset of a finite predefined set of labels $\mathscr{L}$. Thus, the target domain $\mathcal{Y}$ is now a power set $\mathcal{P}(\mathscr{L})$.

Going from standard classification to MLC corresponds to relaxing the mutual exclusivity of the classes in classification, since in this task, an example can be labeled with more than one or zero labels. The labels which label a given example, are referred to as relevant labels for this example.

Predictive modeling approaches to MLC can be grouped similarly as those for MTR. For example, one can always approach a MLC problem by learning a predictive model for every label separately, thus transforming a MLC problem into $|\mathscr{L}|$ standard classification problems (Tsoumakas & Vlahavas, 2007). A similar, yet even more time-consuming approach is to build $L(L-1)/2$ predictive models, one for each pair of labels $\ell_1 \neq \ell_2$ (Elkafrawy et al., 2015). Another option is to treat the subsets of labels as class values since they are all elements of the power set, thus transforming a multi-label problem into a multi-class classification problem with possibly extremely high number of classes (Tsoumakas & Vlahavas, 2007).

Another group of methods are the so-called method adaptation approaches where an existing method is adapted to directly address MLC problems. This has been done, for example, in the case of predictive clustering trees (Madjarov et al., 2012) and support vector machines (Elisseeff & Weston, 2001).

2.2.3.3 Hierarchical Multi-label Classification

The third question (*Which species, genera and families are in the picture?*) is one of the hierarchical multi-label classification (HMLC) tasks. HMLC is a further generalisation of MLC where the label set $\mathscr{L}$ is partially ordered with some partial ordering $\preccurlyeq$. This ordering results in a *hierarchical constraint*: if $\ell_1 \preccurlyeq \ell_2$ for two labels $\ell_{1,2} \in \mathscr{L}$, then a HMLC model must satisfy the condition $\ell_1 \in \hat{\boldsymbol{y}}(\boldsymbol{x}) \Rightarrow \ell_2 \in \hat{\boldsymbol{y}}(\boldsymbol{x})$. This means that if a less general (more specific) label ℓ_1 is predicted to be relevant for a given example $\boldsymbol{x}$, then also a more general (less specific) label ℓ_2 has to be predicted relevant for the same example.

Continuing with the example question that motivated HMLC tasks, label set $\mathscr{L}$ may consist of all taxonomic ranks of living beings, and the underlying ordering would be their taxonomic ordering as we know it, e.g., `African elephant` $\preccurlyeq$ `elephants`, `elephants` $\preccurlyeq$ `mammals`, `mammals` $\preccurlyeq$ `animals`, etc.

A possible approach to HMLC is to ignore the hierarchy in the learning phase and use any of the MLC approaches discussed above. However, at the prediction stage, the set of relevant labels must potentially be extended so that the hierarchy constraint is satisfied. However, method adaptation techniques where an existing method is adapted to a HMLC problem may be more appropriate. This can be done with predictive clustering trees (PCTs), which were shown to outperform their basic versions that follow the binary relevance approach (Vens et al., 2008) or with kernel-based approaches (Rousu et al., 2006).

2.2.4 Semi-supervised Learning

Data for any of predictive modeling tasks – either one of the primitive ones or one of the structured output ones – may contain some unlabeled examples, i.e., examples $\boldsymbol{x}$ for which the value of the target attribute is unknown.

If this is not the case, i.e., all target values are known, we refer to the learning setting as to *supervised* learning. In a sense, this case is the easiest from the learning perspective, since, typically, learning algorithms may handle missing values of features, but can only rarely cope with the missing values of target variable.

If there are some missing values of the target variable, we refer to the learning setting as to *semi-supervised* learning since it is somewhere in between supervised and unsupervised learning. One way of solving a semi-supervised learning problem is – considering the above limitations of the learning algorithms – to ignore the data with the unknown target value and learn only from the data with known target values. Note that all of the tasks discussed in the supervised setting are also present in the semi-supervised setting, e.g., MTR datasets containing examples with missing values for the target variables can be approached with methods for semi-supervised MTR.

However, under the *clustering hypothesis*, i.e., *Clusters of data examples (as computed in the descriptive space) well resemble the distribution of target values*, one can make use also of unlabeled data and boost the performance of the predictive models (Levatić, 2017; Zhu et al., 2009).

The unlabeled data can be used in more than one way. First, one can iteratively label them, so that on every iteration, a predictive model is built on the labeled data, and the target values of the unlabeled data are predicted. After assessing the certainty with which the predictions are made, only the examples for which the algorithm is the most certain about their labels keep their labels. This procedure is referred to as self-training (Zhu et al., 2009).

Another way to take into account the unlabelled data is using predictive clustering trees where the heuristic that estimates the quality of the splits is computed not only from the distribution of the target variable, as is the case for standard decision trees (Breiman et al., 1984), but rather takes into account also the distribution of the features (Levatić, 2017).

2.3 Feature Ranking

As mentioned in the Introduction, feature ranking is tightly coupled with either predictive modeling or clustering, since feature ranking either makes these tasks easier or explains their results. Now that these background tasks are covered, we can give the exact definition of feature ranking, and partially repeat the motivation for computing it.

To keep the notation easier, we limit ourselves to tabular data, and give the exact formulation for relational data later in Section 3.3. Also, since the definition and motivation are different for predictive modeling tasks and clustering, we consider two separate cases.

Predictive Modeling. The motivation for feature ranking in the context of predictive modeling is three-fold. First, since the number of features x_i in a typical data set is increasing over the years, some data may be prohibitively big for some learning algorithms. Therefore, correctly using only some small number of relevant (important) features may result in considerable speed-ups, making model learning much more efficient, or even possible, while not decreasing the quality of the model.

Second, the volume of space increases exponentially with the dimension[2]. This can make the data very sparse and the learning algorithms cannot induce a good model. Again, choosing only relevant features for building a model decreases the sparsity and can actually improve the predictive performance of the learned model.

Third, feature ranking can explain the predictions of the model. This is important for two reasons: i) If the model describes the behavior of some machine, e.g., the heaters of a

[2]Strictly speaking, we can speak about the volume of the subspace spanned by the numeric features. However, in the subspace that is spanned by nominal features, we face a similar phenomenon, namely the combinatorial explosion of the number of points in this space.

satellite (see Chapter 8), the machine's operator can better understand the behavior of the machine; ii) If a domain expert, e.g., a medical doctor, understands how the model makes the predictions, the expert will trust the model, which is important when delicate decisions have to be made. Solutions for the above problems may all be achieved by computing a feature ranking.

Feature ranking for predictive modeling. The input of a feature ranking algorithm is data that consists of examples $(\boldsymbol{x}, \boldsymbol{y})$. The output of a feature ranking algorithm is a list of scores $\mathit{importance}(x_i) \in \mathbb{R}$, for every feature x_i in the data set, where the scores resemble the influence of features x_i on the target variable $\boldsymbol{y}$. A feature ranking algorithm thus performs feature importance estimation.

If we order the features decreasingly with respect to the computed scores, we obtain their ranking. Then, keeping only some top-ranked features (the features with the highest scores) solves the first two problems. This amounts to using feature ranking algorithm as a feature selection approach[3]. Analyzing all the scores solves the third problem.

Selecting the optimal subset of features has been shown to be NP-hard (Welch, 1982), thus finding an optimal feature ranking should be at least as hard. This is why there is no general definition of importance. Rather, every algorithm defines its own heuristic for computing the scores $\mathit{importance}(x_i)$.

Clustering. The motivation for feature ranking in the context of clustering is similar to the motivation above. For example, reducing the number of features might considerably speed up a clustering algorithm, or increase the quality of clustering. Similarly, clustering might be better understood if we see the importance scores. However, there is an additional motivating example that more or less subsumes the already mentioned, and serves as a definition of feature ranking for unsupervised learning, which we adapt from (Doquet & Sebag, 2020): in unsupervised learning we cannot say how relevant a feature is for predicting a target variable, but we can estimate how redundant the feature is in the context of the others.

Feature ranking for unsupervised learning. The input of feature ranking algorithm is data that consists of examples $\boldsymbol{x}$. The output of feature ranking algorithm is a list of scores $\mathit{importance}(x_i) \in \mathbb{R}$, for every feature x_i in the data set. Features that are more relevant to the unsupervised learning task at hand (e.g., clustering) receive higher scores, while features that are less relevant or redundant, depending on the ranking approach, receive lower scores.

As in the predictive modeling case, we can sort the features with respect to their scores, and keep in the data only the most relevant/least redundant ones.

2.3.1 Feature Ranking and Feature Selection

The basic learning tasks, such as classification, regression and clustering, are very well-established, therefore it comes as no surprise that a plethora of feature ranking (and selection) methods exist that are applicable in these cases. The feature ranking methods are divided into three big groups (Guyon & Elisseeff, 2003): *filters*, *wrappers* and *embedded methods*. In this part, we give a brief overview of these groups. The concrete examples of the algorithms from this group are given in the next subsection.

In general, *filters* are the fastest, because they do not interact with any particular classifier (or regressor), which is typically a bottleneck of computing the feature ranking. Rather, feature importance scores are some heuristic values defined within a given filter. As we shall see in the concrete examples below, the heuristics are often provably suboptimal,

[3]Feature selection can be seen as a special case of feature ranking where the only possible values of feature scores are 1 (feature is selected) and 0 (feature is not selected).

but they mostly come with low or even optimal computational cost of $\mathcal{O}(mn)$, where m and n are the number of examples and features in a data set. This is the optimal cost since each algorithm needs this much time to read the whole data set.

Embedded feature ranking methods are all part of some classification, regression or clustering algorithm. Thus, most of the computational resources are spent on learning, and after the model (or clustering) is learned, the feature importance scores are more or less seamlessly computed from the model (or clustering).

Wrapper approaches are used especially when the feature ranking is computed to perform feature selection, or when the learning algorithm (e.g., support vector machines) is already chosen. The reason for this is that every wrapper consists of two parts: one is a learning algorithm by which a subset of features is evaluated, and the other is a heuristic for exploring the space of all feature subsets. A heuristic is necessary since selecting the optimal feature subset is a NP-hard problem (Amaldi & Kann, 1998). Thus, one starts the procedure with some set $\mathcal{C}$ of candidate features (e.g., $\mathcal{C}$ is empty or a full set of features) and then, guided by predictive performance of the models (or quality of the clusterings) obtained from the part of the data set that corresponds to features from $\mathcal{C}$, one iteratively adds and/or removes features from the set $\mathcal{C}$. Usually $\mathcal{C}$ is modified in a more (e.g., forward selection and backward elimination (Kittler, 1978)) or less (e.g., beam search (Siedlecky & Sklansky, 1988)) greedy way to save time. Moreover, greediness of the search procedure can also be an advantage, since it is reported (Guyon & Elisseeff, 2003) that it is a guard against overfitting.

2.3.2 Classification, Regression and Clustering Tasks

In this section, we provide a brief overview of the feature ranking methods for the basic learning tasks, starting with classification, continuing with regression and ending with clustering.

2.3.2.1 Classification

An example of a simple and computationally fast filtering approach for a classification data set is the χ^2 statistic. However, it is applicable only to nominal features. If we are given a nominal feature x_i, we use the χ^2-statistic to test the null hypothesis H_0 that the distribution of the target is independent of the value of x_i. The greater the value of the statistic, less probable it is that H_0 holds, hence higher values of χ^2 mean higher relevance of x_i. If a feature is numeric, then it has to be discretized, using for example binning, as shown in Figure 2.6.

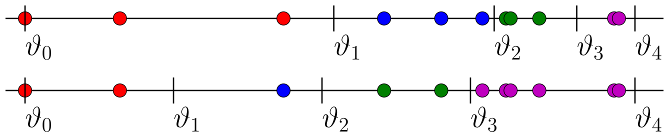


Figure 2.6: Equal-frequency (top) and equal-width (bottom) binning of eleven examples. In both cases, the range of a feature was divided into 4 subintervals (bins). Colors of the feature values correspond to a value of the newly constructed nominal feature.

Especially when computing feature ranking to do a feature selection, the problem might be the *redundancy* of some features. For example, if features x_1 and x_2 are highly correlated with each other (or even copies of each other), and have a large influence on the target, knowing the value of one of them gives us the same information about the target as knowing both values. Hence, sometimes it is desirable for the heuristic to take into account also the redundancy, but not always:

- If we rank the features in order to select the top-ranked ones, only one of the features x_1 and x_2 should be among the top-ranked features, since it suffices to keep only one of them.
- If we rank the features to see how much each of them influences the target, both x_1 and x_2 should be ranked at the top, since they have approximately the same influence.

In the first case, the *symmetrical uncertainty*-based filter (Witten et al., 2011) may give better ranks than the χ^2 method, since it also takes into account the redundancy. Symmetrical uncertainty between random variables z_1 and z_2 is defined as

$$U(z_1, z_2) = 2\frac{H(z_1) + H(z_2) - H(z_1, z_2)}{H(z_1, z_2),}$$

where H is the Shannon entropy (Shannon, 1948). The value of U measures how much information do z_1 and z_2 share about each other. The heuristic for ranking is defined as

$$\mathit{importance}(x_i) = \frac{U(x_i, y)}{\sqrt{\sum_{j \neq i} U(x_i, x_j)}} \tag{2.1}$$

However, its complexity is $\mathcal{O}(mn^2)$, which may be prohibitively large. There are many other information-theoretic-based filters, *info gain* being one of the most frequently used ones (Mitchell, 1997), since it is part of the ID3 algorithm (Quinlan, 1986). The measure again uses Shannon entropy H, and is defined as

$$\mathrm{IG}(x_i) = H(y) - H(y|x_i). \tag{2.2}$$

Another prominent member of the filter group is ReliefF (Robnik-Šikonja & Kononenko, 2003), which is an extension of the Relief algorithm (Kira & Rendell, 1992) to an arbitrary classification task. Since we extend the Relief family to the tasks of structured output prediction, unsupervised and semi-supervised learning, ReliefF will be described later in Section 2.3.5.

As for the embedded approaches, one has to first find an appropriate classifier, from which a ranking is then computed. One option are the support vector machines (Guyon et al., 2002) with linear kernels, where the final model is of form $\hat{\boldsymbol{y}}(\boldsymbol{x}) = \mathrm{sign}(\langle \boldsymbol{x}, \boldsymbol{n} \rangle - b)$, where $\boldsymbol{n}$ is the normal of the hyperplane that was fitted to separate positive and negative examples. Since the value of $\hat{\boldsymbol{y}}$ changes most quickly in the directions on the normal, the feature importance score is defined as $\mathit{importance}(x_i) = |\boldsymbol{n}_i|$.

Another possibility, for example, are feature rankings computed from (ensembles of) decision trees (Quinlan, 1986), or predictive clustering trees (Kocev, 2011). However, since we extend these approaches to structured output prediction and unsupervised learning, they will be described later in Section 2.3.4.

An example of a wrapper is the SVM-RFE algorithm (Guyon et al., 2002): recursive feature elimination, where we compute the ranking in the following way. We start with the set $\mathcal{C}$ of all features. Then, at each step, we learn a support vector machine model using linear kernels, i.e., fit the normal $\boldsymbol{n}$ and bias b to the data, to obtain the predictor $\hat{\boldsymbol{y}}(\boldsymbol{x}) = \mathrm{sign}(\langle \boldsymbol{x}, \boldsymbol{n} \rangle - b)$. Features x_i, for which $\boldsymbol{n}_i \approx 0$, have no or very limited influence on the value $\hat{\boldsymbol{y}}(\boldsymbol{x})$. Hence, on each step, we discard the feature x_i with the smallest $|\boldsymbol{n}_i|$.

At step j, the discarded feature is assigned the score of j, and the ranking is obtained after $n-1$ steps.

Another option in wrapper approaches is recursive feature addition, where one starts with an empty set $\mathcal{C}$ of candidates. Then, on every step, a classifier is learned that uses the features from the set $\mathcal{C} \cup \{x_i\}$, for every $x_i \notin \mathcal{C}$. Then, the feature x_i that improves the model's quality the most is added to $\mathcal{C}$. Of course, this procedure is computationally quite intensive since it learns $\mathcal{O}(n^2)$ models.

2.3.2.2 Regression

One of the simplest and the fastest filters for regression data sets is the *Pearson correlation coefficient* (Pearson, 1895). Here, the importance of a feature x_i is defined as

$$importance(x_i) = \left| \frac{\mathrm{Cov}\,(x_i, y)}{\sigma_x \sigma_y} \right|, \tag{2.3}$$

where Cov is the covariance operator, and σ_x and σ_y denote standard deviations of the feature and target, respectively. A high value of *importance*(x_i) means that the feature is highly correlated with the target, i.e., relevant.

Both Pearson correlation coefficient and χ^2 (described above) are optimal regarding the computational complexity, but they also both suffer from myopia, i.e., they are not able to capture any interactions between features, which can lead to highly suboptimal rankings: If binary target y is defined as $y = x_1$ XOR x_2 (while the other features are just random noise), these two measures perceive *all* features as (equally) random.

This is overcome by another member of the filter group, RReliefF (Robnik-Šikonja & Kononenko, 2003). Since we extend the Relief family to the other learning tasks, it will be described later in Section 2.3.5.

As for the embedded approaches, support vector machines with linear kernels can be used not only in the context of classification, but also in the context of regression. However, one should use the support vector machines adapted to regression instead.

Similar holds for other embedded approaches to feature ranking, namely the ensemble-based feature rankings computed from (ensembles of) decision trees (Quinlan, 1986) or predictive clustering trees (Kocev, 2011). As noted in the classification subsection, they will be described later in Section 2.3.4.

As for the wrapper methods, one should only replace a classifier by a regressor and use any of the approaches mentioned in the classification case (e.g., recursive feature elimination/ addition).

2.3.2.3 Clustering

The unsupervised filters are typically conceptually somewhat more complicated than their supervised counterparts, and their final scores *importance*(x_i) cannot be represented by a simple formula, such as Equation (2.2). For example, the Laplace score (He et al., 2005) is based on graph theory and the creation of the Laplace matrix of a graph, whose vertices are the examples and whose edges correspond to the level of similarity among the examples: a feature x_i is relevant if its values respect the graph structure, i.e., the higher the weight on the edge between two nodes, the more similar are the values of x_i for these two nodes.

Another filter-based approach is SPEC (spectral clustering) (Zhao & Liu, 2007) – a generalization of many existing feature ranking scores, including Relief and Laplacian score. The actual unsupervised score that the authors propose is thus a result of particular instantiations, such as choosing an appropriate similarity measure, e.g., radial basis kernels.

Similarity matrix (again using radial-basis kernel) is also one of the building blocks of the MCFS filter (Cai et al., 2010). After computing it, the algorithm proceeds to solving a generalized eigenvalue problem, and ensures the sparsity of the scores via L_1-regularization in the objective function.

Another variation of the above approaches is NDFS (Li et al., 2012) where $L_{2,1}$-regularization is applied. The best features are those which are the most closely related to the labels of the clusters constructed within the algorithm.

Existing unsupervised embedded methods for feature ranking include the Agnos family of feature ranking algorithms (Doquet & Sebag, 2020). For example, AgnoS-S, the best performing algorithm in the family, is based on an auto-encoder with one hidden layer with the number of neurons never greater (and typically considerably smaller) than n. Additionally, slack variables a_i, $1 \leq i \leq n$ are included into the auto-encoder (every feature value is in the input layer multiplied by the corresponding a_i), and into the loss function (in the form of L_1 penalty $\sum_i |a_i|$). After the auto-encoder is trained, one simply reads the importance scores $\mathit{importance}(x_i) = |a_i|$.

As for the wrapper approaches, an analogous procedure to the recursive feature elimination/addition in the context of classification and regression can be applied in the unsupervised scenario (instead of learning predictive models, one would cluster the data), however, we could not find any unsupervised wrappers in the literature.

2.3.3 Trees and Tree Ensembles

In this section, we first explain the mechanisms with which single trees and ensembles thereof are built. In the next section, we will then be able to define feature ranking scores embedded in the trees (and ensembles).

2.3.3.1 A Single Tree

Decision trees (Breiman et al., 1984; Quinlan, 1986) are mostly grown in a top-down fashion. In this part, we limit ourselves to classification and regression. A general framework that also covers structured output prediction and clustering is described later in Section 2.4. The pseudocode of tree-induction is given in Algorithm 2.1.

Algorithm 2.1: Tree(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$
2: if $t^* = \mathit{none}$ then
3:     return $\mathit{Leaf}(\mathit{prototype}(E))$
4: else
5:     for each $E_i \in \mathcal{P}^*$ do
6:         $\mathit{tree}_i = \text{Tree}(E_i)$
7:     return $\mathit{Node}(t^*, \bigcup_i \{\mathit{tree}_i\})$
```

Algorithm 2.2: BestTest(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = (\mathit{none}, 0, \emptyset)$
2: for each test $t$ do
3:     $\mathcal{P}$ = partition induced by $t$ on $E$
4:     $h = |E|\mathit{impu}(E) - \sum_{E_i \in \mathcal{P}} |E_i|\mathit{impu}(E_i)$
5:     if $h > h^*$ then
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $(t^*, h^*, \mathcal{P}^*)$
```

Starting with $E = \mathscr{D}_{\text{TRAIN}}$, i.e., the whole data set, the algorithm first greedily finds (line 1) a splitting criterion (also known as test) that splits the data into subgroups, so that the heuristic score h is maximized. The tests are typically of the form $x_i \in A$ for some subset A of the domain $\mathcal{X}_i$ of the feature x_i, as shown in Figure 2.7. Thus, there are only two possible outcomes of the test and thus, the data is partitioned into two parts. The heuristic h that measures the quality of a test is the decrease of the impurity impu of the target variable. The impurity function can be in theory any measure of how non-constant the target value is. Thus, heuristic (greedily) guides the algorithm towards the shallow trees that partition the data so that the target variable is constant on the every induced

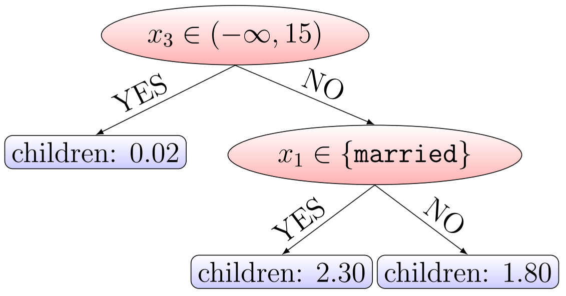


Figure 2.7: An example decision tree for the regression problem of predicting the number of children. The feature x_1 is the relationship status of a person, whereas the feature x_3 is the age of a person. Since x_3 is numeric, the test in the root of the tree is of form $x_3 \in A = (-\infty, \vartheta)$, where the threshold ϑ is set to 15. Since x_1 is nominal with the domain $\mathcal{X}_1 = \{\texttt{single}, \texttt{in-relation}, \dots, \texttt{married}\}$, the test in the other internal node of the tree is of form $x_1 \in A$ where A is a proper non-empty subset of $\mathcal{X}_1$. In this particular case, $A = \{\texttt{married}\}$.

subset of the data. The heuristic score is needed since finding the optimal tree is known to be NP-complete (Hyafil & Rivest, 1976).

The best among the tests is found by iterating over all the possible tests (Algorithm 2.2). For a given test, the corresponding heuristic score (decrease of the impurity after applying the test) is computed in line 4. If the test decreases the impurity (Algorithm 2.1, line 3), the tree-induction algorithm creates a leaf node and computes a leaf node with a prototype value (prediction) that is used in the prediction stage.

Otherwise, an internal node $\mathscr{N}$ with the chosen test is constructed, and the tree-induction algorithm is recursively called on the subsets E_i in the partition $\mathcal{P}$ of the data, defined by the test. The tests are typically of the form $x_i \in A$ for some subset A of the domain $\mathcal{X}_i$ of the feature x_i, as shown in Figure 2.7. Thus, there are only two possible outcomes of the test and thus, the data is partitioned into two parts. The trees grown from these subsets become child nodes of the node $\mathscr{N}$ (Algorithm 2.1, line 7).

To fully define the trees, we need to specify the actual impurity and prototype functions.

For the regression, impurity of the target is measured as the variance of the target, i.e., $\mathit{impu}(E) = \mathit{Var}(E, \boldsymbol{y}) = \frac{1}{|E|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in E}(\boldsymbol{y} - \bar{\boldsymbol{y}})^2$, where $\bar{\boldsymbol{y}} = \frac{1}{|E|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in E}\boldsymbol{y}$ is the average (over the subset E) of the target variable[4].

As the prototype function in regression problems, one typically uses the mean value $\bar{\boldsymbol{y}}$ but others, such as median, are also possible[5].

For the classification, there is more than one frequently used impurity function. As mentioned before, ID3 algorithm uses info gain, as defined in Equation (2.2). Predictive clustering trees, on the other hand, use Gini index $\mathit{impu}(E) = \mathit{Gini}(E, \boldsymbol{y}) = 1 - \sum_y p_E^2(y)$, where the sum goes over the possible values y of the target variable, and $p_E(y)$ is the relative frequency of the value y in the subset E. When adapting the impurity measures to the semi-supervised problems in Section 2.4.4, we show that Gini index is the most appropriate nominal analogue of variance.

Trees, as predictive models, are mostly valued for their interpretability. However, they might be undesirably unstable, which may hurt their predictive performance (Breiman, 1996). This is overcome by the ensemble techniques described in the next section.

[4]Biased estimator of variance is used to prevent the potential divisions by zero when $|E| = 1$.

[5]The prototype function should depend on the evaluation criterion. For example, mean minimizes the mean squared error, whereas the median minimizes the mean absolute error.

2.3.3.2 Tree Ensembles

An ensemble is a set of base predictive models. Its prediction for an example $\boldsymbol{x}$ is made by combining the predictions of the base models. For the classification, this is typically done by taking the majority vote of the baseline models. For the regression tasks, a mean over the base-model prediction is computed.

The motivation for introducing an ensemble is that it can be viewed as more stable versions of its base models, since – if the members are diverse models (Hansen & Salamon, 1990) – the ensemble predictions have lower variance and are therefore more accurate. Thus, since the trees are unstable, they are very suitable base models.

To introduce some diversity into the tree induction, one has to modify the completely deterministic tree induction Algorithms 2.1 and 2.2. There are several ways to do so and we use three of them.

Bagging Instead of being learned from the whole training set $\mathscr{D}_{\text{TRAIN}}$, each tree in the bagging ensemble (Breiman, 1996) is built from a different bootstrap replicate $\mathcal{B}$ of $\mathscr{D}_{\text{TRAIN}}$, called bag[6]. Thus, the bagging procedure manipulates only the training data but not the tree induction itself.

Random Forests This ensemble method (Breiman, 2001) goes a step further and, in addition to bootstrapping, modifies the selection of the best test. More precisely, instead of evaluating all possible tests in Algorithm 2.2, only the tests that a random subset of features yield are evaluated. The number of the retained features n' is a parameter to the algorithm. Its typical values are $\lceil \log_2 n \rceil$ or $\lceil \sqrt{n} \rceil$, etc. Thus, If $n' = n$ is the special case when random forest procedure boils down to bagging.

Extra Trees Ensembles of extremely randomized trees (Geurts et al., 2006) also consider only n' features in each node, but do not evaluate all the corresponding tests. Rather, only one test per feature is chosen randomly, thus only n' tests are evaluated, and the algorithm chooses the best one among these n' tests. The rationale behind this more extreme randomization is that this increases the diversity, and makes the ensemble a stronger learner. Note that, however, Extra-Tree ensembles originally do not use bootstrapping, but for some data sets (e.g., those with many binary features where the random choice of a particular test is useless), our experiments show that it is beneficial to use it, see, for example, Chapters 6 and 7.

Computing an ensemble instead of a single tree improved the predictive performance of the model. However, we lose some interpretability since the model becomes much more complex. However, there are feature ranking scores embedded into tree ensembles, which can again explain the ensemble. The definitions of these scores are given in the next section.

2.3.4 Feature Ranking with Tree Ensembles

Now, we have everything prepared to describe feature ranking scores that are embedded in tree ensembles for classification and regression. For its simplicity, we start with the Symbolic score[7], continue with the Genie3 (Huynh-Thu et al., 2010) score and finish with the Random Forest score (Breiman, 2001). Here, we assume that ensemble $\mathcal{E}$ has already been grown.

[6]A bootstrap replicate of a data set of size m is obtained by choosing an example from the data m-times uniformly at random, with repetition. The choices are independent.

[7]The score is implemented in xgboost package https://xgboost.readthedocs.io/en/latest/

Symbolic The motivation for the Symbolic score is that a feature x_i is relevant for a given tree $\mathcal{T}$ if the feature influences the predictions of the tree. Thus, the Symbolic score simply counts the occurrences of a feature in the internal nodes of a tree, and is defined as

$$importance_{\text{SYMB}}(x_i) = \frac{1}{|\mathcal{E}|}\sum_{\mathcal{T}\in\mathcal{E}} |\mathcal{T}(x_i)|, \tag{2.4}$$

where the sum goes over the trees $\mathcal{T}$ in the ensemble, and $\mathcal{T}(x_i)$ is the set of internal nodes in a tree $\mathcal{T}$ where x_i is present in the test. The set of the examples that reach a given node $\mathscr{N}$ is denoted by $E(\mathscr{N})$.

Genie3 The Genie3 score can be seen as a more sophisticated version of the Symbolic score, where, additionally, the quality of the split is taken into account. Here, instead of only counting the number of examples, the heuristic value of the test $h^*(\mathscr{N})$ in a node $\mathscr{N}$ is taken as a basic reward. It is important to note that the value $h^*(\mathscr{N})$ is proportional to the number of examples (see line 4 of Algorithm 2.2). The exact definition of the Genie3 score is

$$importance_{\text{GENIE3}}(x_i) = \frac{1}{|\mathcal{E}|}\sum_{\mathcal{T}\in\mathcal{E}}\sum_{\mathscr{N}\in\mathcal{T}(x_i)} h^*(\mathscr{N}). \tag{2.5}$$

Random Forest score The Random Forest score was designed to explain the predictions of the random forest ensemble (hence the name), but – as we shall see – it can be used with any predictive model. However, trees are especially appropriate because the predictions (needed for the computation of the score) can be obtained fast, provided the trees are balanced.

The Random Forest score measures to what extent noising (i.e., permuting) the feature values decreases the predictive performance of a tree. This is done in two steps. First, recall that a tree in a random forest (and the other aforementioned ensembles) is learned on a bootstrap replicate (called bag) of the training data set $\mathscr{D}_{\text{TRAIN}}$. Thus, approximately one third of the training data set are the so-called out-of-bag examples[8]. We denote them with $\text{OOB}_{\mathcal{T}}$.

Since these examples were not seen during the induction of the tree $\mathcal{T}$, one can use them to assess the predictive power of the tree. Using some error measure *Err* (where lower values correspond to better predictions), we denote the value of this error measure for the set $\text{OOB}_{\mathcal{T}}$ as $Err(\text{OOB}_{\mathcal{T}})$. We compare this value to the error $Err(\text{OOB}_{\mathcal{T}}{}^i)$ where the set $\text{OOB}_{\mathcal{T}}{}^i$ is obtained from $\text{OOB}_{\mathcal{T}}$ by randomly permuting the values of the feature x_i. The greater the (relative) difference between the error values, the more important the feature. The exact definition of the Random Forest score is thus

$$importance_{\text{RF}}(x_i) = \frac{1}{|\mathcal{E}|}\sum_{\mathcal{T}\in\mathcal{E}} \frac{Err(\text{OOB}^i_{\mathcal{T}}) - Err(\text{OOB}_{\mathcal{T}})}{Err(\text{OOB}_{\mathcal{T}})}. \tag{2.6}$$

A typical error measure in regression problems is the quadratic loss. As for the classification problems, one can set *Err* to accuracy. Since for accuracy, more is better, we first compute the scores following Equation (2.6), and then multiply them by the factor of -1.

[8]The probability of an example not being selected to the bag is $(1 - 1/m)^m$, m being the number of examples in $\mathscr{D}_{\text{TRAIN}}$. Thus, the expected proportion of the examples that are not part of the bag quickly converges to $m/e \approx 0.37m$ when $m \to \infty$.

Equation (2.6) shows that any model can be used for permutation-style feature ranking. Moreover, all the three scores can be actually seen as ensemble versions of the tree scores, i.e., the averages of the latter, since the outer operator is always averaging. As a consequence, the larger the ensemble, the more stable the scores.

In this section, we described the first group of feature ranking scores that we base our methodology on. In the next section, the remaining group of algorithms is described.

2.3.5 Distance-based Feature Ranking with Relief

The algorithms in the Relief group are all filters. Their main advantage over the other filters is that they are not myopic and at the same time, they do not need to test for the possible interactions of (pairs of) features explicitly which considerably increases the computational complexity of the algorithm. For example, they can (in linear time with respect to the number of features) detect relevant features in XOR relations (Robnik-Šikonja & Kononenko, 2003).

The rationale behind the Relief group of algorithms is that, given two examples that are close to each other (in the descriptive space), a feature x_i is relevant if the differences between the target values of the two examples are notable if and only if the differences in the values of x_i between these two examples are notable.

Closeness of the examples or differences of the attribute values are expressed as the distances in the appropriate subspaces of a data set. More precisely, if $\mathcal{X} = \bigtimes_{i=1}^{n} \mathcal{X}_i$ is the descriptive space and $\boldsymbol{x}^1, \boldsymbol{x}^2 \in \mathcal{X}$ are the descriptive parts of two examples, then the distances between these two examples in the feature spaces $\mathcal{X}_i$ are defined as

$$d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) = \begin{cases} \mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2] & : \mathcal{X}_i \nsubseteq \mathbb{R} \\ \frac{|\boldsymbol{x}_i^1 - \boldsymbol{x}_i^2|}{\max\limits_{\boldsymbol{x}} \boldsymbol{x}_i - \min\limits_{\boldsymbol{x}} \boldsymbol{x}_i} & : \mathcal{X}_i \subseteq \mathbb{R} \end{cases} \tag{2.7}$$

where $\mathbf{1} : \varphi \mapsto \mathbf{1}[\varphi]$ denotes the indicator function that returns 1 if the condition φ is satisfied and 0 otherwise, and $\boldsymbol{x}_i^1$ and $\boldsymbol{x}_i^2$ denote the values of the feature x_i for the two examples. Furthermore, the distance in the descriptive space $\mathcal{X}$ is then defined as

$$d_{\mathcal{X}}(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{1}{n} \sum_{i=1}^{n} d_i(\boldsymbol{x}^1, \boldsymbol{x}^2). \tag{2.8}$$

The definition of the distance $d_{\mathcal{Y}}$ in the target space depends on the task. For classification, we follow the nominal part of Equation (2.7), whereas for regression, the numeric part of the same equation applies.

Different members of the Relief family cover different tasks. For example, Relief (Kira & Rendell, 1992) can handle only binary classification whereas ReliefF (Robnik-Šikonja & Kononenko, 2003) can handle any classification tasks. Finally, RReliefF (Robnik-Šikonja & Kononenko, 2003) handles regression tasks. To some extent, the importance scores, as returned by all of them, equal the expected value of the expression

$$P_1 - P_2 = P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \mid \boldsymbol{y}^1 \neq \boldsymbol{y}^2) - P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \mid \boldsymbol{y}^1 = \boldsymbol{y}^2), \tag{2.9}$$

where example $\boldsymbol{r} = (\boldsymbol{x}^1, \boldsymbol{y}^1) \in \mathscr{D}_{\text{TRAIN}}$ is randomly chosen, and $\boldsymbol{n} = (\boldsymbol{x}^2, \boldsymbol{y}^2)$ is one of its nearest neighbors. Above, $\boldsymbol{y}^1, \boldsymbol{y}^2 \in \mathcal{Y}$ are the target parts of the corresponding examples. The conditional probabilities above are first expressed with unconditional ones (using Bayes formula $P(A|B)P(B) = P(A \wedge B)$), and Equation (2.9) then reads as

$$P_1 - P_2 = \frac{P_{\textit{diffFeat, diffTarget}}}{P_{\textit{diffTarget}}} - \frac{P_{\textit{diffFeat}} - P_{\textit{diffFeat, diffTarget}}}{1 - P_{\textit{diffTarget}}}, \tag{2.10}$$

where the labels *diffFeat* and *diffTarget* correspond, respectively, to the events that two instances have a different value of a feature, and a different value of the target.

The corresponding probabilities are in the algorithms modelled as distances in the appropriate spaces. For example, $P_{\text{diffFeat}} = P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2) \approx d_i(\boldsymbol{x}^1, \boldsymbol{x}^2)$. The exact formulation of the ReliefF and RReliefF algorithms is given in Algorithms 2.3 and 2.4, respectively.

.

Algorithm 2.3: ReliefF($\mathscr{D}_{\text{TRAIN}}$, I, K)

```
1: $\boldsymbol{w}$ = zero list of length $n$
2: for $\iota = 1, 2, \dots, I$ do
3:     $\boldsymbol{r} = (\boldsymbol{x}^\iota, \boldsymbol{y}^\iota)$ = random example from $\mathscr{D}_{\text{TRAIN}}$
4:     $H_1, \dots, H_K$ = $K$ nearest hits for $\boldsymbol{r}$
5:     for all classes $c \neq \boldsymbol{y}^\iota$ do
6:         $M_{c,1}, \dots, M_{c,K} \leftarrow K$ nearest misses for $\boldsymbol{r}$ from class $c$
7:     for $i = 1, 2, \dots, n$ do
8:         $\boldsymbol{w}[i]$ += $\left(\sum_{c \neq \boldsymbol{y}^\iota} \frac{P(c)}{1 - P(\boldsymbol{y}^\iota)} \sum_{l=1}^{K} d_i(M_{c,l}, \boldsymbol{r}) / (IK)\right) - \sum_{l=1}^{k} d_i(H_l, \boldsymbol{r}) / (IK)$
9: return $\boldsymbol{w}$
```

Algorithm 2.4: RReliefF($\mathscr{D}_{\text{TRAIN}}$, I, K)

```
1: $\boldsymbol{w}$ = zero list of length $n$
2: $\boldsymbol{P}_{\textit{diffFeat, diffTarget}}$, $\boldsymbol{P}_{\textit{diffFeat}}$ = zero lists of length $F$
3: $P_{\textit{diffTarget}} = 0.0$
4: for $\iota = 1, 2, \dots, I$ do
5:     $\boldsymbol{r}$ = random example from $\mathscr{D}_{\text{TRAIN}}$
6:     $\boldsymbol{n}_1, \boldsymbol{n}_2, \dots, \boldsymbol{n}_K$ = $K$ nearest neighbors of $\boldsymbol{r}$
7:     for $k = 1, 2, \dots, K$ do
8:         $P_{\textit{diffTarget}}$ += $d_{\mathcal{Y}}(\boldsymbol{r}, \boldsymbol{n}_k)$
9:         for $i = 1, 2, \dots, n$ do
10:            $\boldsymbol{P}_{\textit{diffFeat}}[i]$ += $d_i(\boldsymbol{r}, \boldsymbol{n}_k)$
11:            $\boldsymbol{P}_{\textit{diffFeat, diffTarget}}[i]$ += $d_i(\boldsymbol{r}, \boldsymbol{n}_k) d_{\mathcal{Y}}(\boldsymbol{r}, \boldsymbol{n}_k)$
12: for $i = 1, 2, \dots, F$ do
13:     $w_i = \frac{\boldsymbol{P}_{\textit{diffFeat, diffTarget}}[i]}{P_{\textit{diffTarget}}} - \frac{\boldsymbol{P}_{\textit{diffFeat}}[i] - \boldsymbol{P}_{\textit{diffFeat, diffTarget}}[i]}{1 - P_{\textit{diffTarget}}}$
14: return $\boldsymbol{w}$
```

They both take the numbers of iterations I and neighbors K as their arguments. The main difference between the algorithms originates from the fact that in classification, the event *examples have different class values* does not happen with probability 1. This is why we can separate the neighbors to hits (the neighbors that belong to the same class as the randomly chosen example $\boldsymbol{r}$), and misses (the neighbors that belong to the different class as the example $\boldsymbol{r}$), which are computed in lines 4 and 6 of Algorithm 2.3.

The reward for a feature in ReliefF is then proportional to the distances in the space $\mathcal{X}_i$ between and its K misses. Similarly, the penalty for the feature is proportional to the feature distances between $\boldsymbol{r}$ and its hits.

In the regression case, belonging to different class is not defined in a strict sense. However, in RReliefF, it is quantified via the distance $d_{\mathcal{Y}}$, i.e., the larger the distance, the more different are the "class" values. Otherwise, the RReliefF algorithm follows the same motivation as ReliefF, but now, the algorithm makes full use of Equation (2.10).

For example, the probability $P_{\text{diffFeat, diffTarget}}$ is modelled as the product of distances $d_i d_{\mathcal{Y}}$. Similarly, $P_{\text{diffFeat}} \approx d_i$ and $P_{\text{diffTarget}} \approx d_{\mathcal{Y}}$.

The generalization of the notion of belonging to different classes and the fact that RReliefF operates only with distances allows for generalizations that we have developed and are described in the next chapters.

Note that we did not show the pseudocode for Relief since the feature importances, as returned by ReliefF, are the same (when applied to a binary classification problem).

2.4 Predictive Clustering and Structured Output Prediction

The predictive clustering framework (Blockeel, 1998) unifies the tree induction processes for supervised learning (i.e., predictive modeling), unsupervised learning (more precisely, clustering), and lately also semi-supervised learning (Levatić, 2017). The trees in this framework are referred to as *predictive clustering trees* (PCTs). Moreover, predictive clustering generalizes the algorithms for inducing classification and regression trees (as presented in Section 2.3.3), so that they are applicable also to the tasks of structured-output prediction, such as multi-target regression, multi-label classification, and hierarchical multi-label classification. In the following subsections, we explain the main ingredients of learning PCTs and PCT ensembles.

2.4.1 Learning PCTs and PCT Ensembles

So far, it was sufficient to divide the columns in a data set into features and targets. A common name for these two types of columns are attributes, and so far, we could have used the name *descriptive attributes* instead of features and *target attributes* instead of targets. However, the first step towards generalization of the tree induction algorithm is defining three groups of attributes in a data set:

- *descriptive attributes* $\mathcal{A}_D$: These are the attributes that can appear as part of the test in the internal nodes of a tree.
- *target attributes* $\mathcal{A}_T$: These are the attributes whose values are output by the function *prototype* in the leaves of a tree.
- *clustering attributes* $\mathcal{A}_C$: These are the attributes that are included in the computation of heuristic values.

For example, in standard classification or regression, the groups of target and clustering attributes coincide, and consist of precisely one attribute (which we have been referring to as $\boldsymbol{y}$). The rest of the attributes are descriptive. In clustering, on the other hand, all three groups of attributes coincide and contain all the attributes present in a data set, as we explain in detail in Section 2.4.4. In the following sections, we also present PCT induction for the tasks of structured output predictions. To do so, it suffices to define three components of the tree induction algorithm, namely, the three groups of attributes, the impurity measure *impu* that is used in heuristic function h, and the function *prototype*.

2.4.2 MTR with PCTs

In Section 2.2.3.1, we defined multi-target regression (MTR) tasks as predictive modeling tasks where the target variable $\boldsymbol{y}$ is a vector of numeric values. However, when defining PCT induction for MTR tasks, it is better to think about target $\boldsymbol{y}$ as a group of numeric attributes, since this simplifies matters a lot.

More precisely, in MTR, groups $\mathcal{A}_T$ and $\mathcal{A}_C$ coincide and contain only numeric attributes. The rest of the attributes belong to group $\mathcal{A}_D$.

In the internal nodes, impurity of a data subset $E \subseteq \mathscr{D}_{\text{TRAIN}}$ is now defined as

$$impu(E) = \frac{1}{|\mathcal{A}_C|} \sum_{y \in \mathcal{A}_C} \frac{Var(E, y)}{Var(\mathscr{D}_{\text{TRAIN}}, y)},$$

i.e., as a normalized average of variances of the clustering attributes. The normalization factors take care of possible different scales of the clustering attributes. Note that when $|\mathcal{A}_C| = 1$, this definition boils down to the definition of (standard) regression impurity, as defined in Section 2.3.3.1.

The function *prototype* (used in the leaves of the tree) returns a vector of average values for the attributes in $\mathcal{A}_T$. The average in every leaf is computed from the training examples that belong to this leaf. The prediction of an ensemble of PCTs is the average of predictions of the trees.

2.4.3 MLC and HMLC with PCTs

Moving to the generalizations of classification, we now describe the tasks of multi-label classification (MLC) and its hierarchical version (HMLC).

As for the three attribute groups, we again have $\mathcal{A}_C = \mathcal{A}_T$, and the rest of the attributes are in $\mathcal{A}_D$, but the attribute types of the clustering (and target) attributes are now different, as compared to MTR.

As defined in Section 2.2.3.2, the target values in MLC tasks are subsets of the set of labels $\mathscr{L}$. However, if we denote $\mathscr{L} = \{\ell_1, \dots, \ell_L\}$, every subset S can be represented as its incidence vector $\boldsymbol{s} \in \{0,1\}^L$, where $\boldsymbol{s}_j = 1 \Leftrightarrow \ell_j \in S$. We can thus act as we have L binary targets, which can be regarded as either numeric or nominal. If we treat the binary targets as nominal, we can use the Gini index for each label in the impurity function:

$$impu(E) = \frac{1}{|\mathscr{L}|} \sum_{\ell \in \mathscr{L}} \frac{\text{Gini}(E, \ell)}{\text{Gini}(\mathscr{D}_{\text{TRAIN}}, \ell)},$$

However, numeric representation speeds up the tree induction since this reduces a MLC problem to a MTR one, regarding the definition of the impurity function *impu*[9]. This also means that the groups of $\mathcal{A}_C$ and $\mathcal{A}_T$ consist of L numeric attributes whose values in the data are constrained to zeros and ones. If we treat the binary targets as numeric, we can use the same impurity function as for MTR.

As for the function *prototype*, one again follows the MTR case, i.e., computes the averages of the target attributes. Then, additionally, those numeric values are thresholded, so that the final predictions equal

$$prototype(E) = \{\ell_j \mid \bar{\boldsymbol{y}}_j \geq \vartheta\}, \tag{2.11}$$

where the threshold ϑ is typically set to $\vartheta = 1/2$. The prediction of an ensemble of PCTs is obtained by majority voting, for every label separately (or by thresholding the per-tree averaged MTR predictions of PCTs in the ensemble).

As for the HMLC case where $\mathscr{L}$ is partially ordered (which results in a hierarchy of the labels), the impurity function has to take into account the fact that the labels that are higher in the hierarchy are more important. This is done by assigning every label a weight. To do so, the partial ordering is first represented as a directed acyclic graph. The graph has the labels as nodes. An edge between two nodes ℓ_1 and ℓ_2 exists if and only if

[9]In general, one would define the impurity function as $impu(E) = \min_{\boldsymbol{z} \in \mathcal{P}(\mathscr{L})} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in E} Loss(\boldsymbol{y}, \boldsymbol{z})$ which may not have a closed form. Going to MTR setting relaxes the condition $\boldsymbol{z} \in \mathcal{P}(\mathscr{L})$ and makes PCT induction feasible.

$\ell_1 \preccurlyeq \ell_2$, and $\neg(\ell_1 \preccurlyeq \ell \preccurlyeq \ell_2)$, for all $\ell \in \mathscr{L} \setminus \{\ell_1, \ell_2\}$. If ℓ_1 and ℓ_2 are connected ($\ell_1 \preccurlyeq \ell_2$), then ℓ_2 is a parent of ℓ_1. Finally, the weights $w(\ell)$ are recursively defined as

$$w(\ell) = \begin{cases} 1 & ; \; \mathit{parents}(\ell) = \emptyset \\ w_0 \cdot \mathit{mean}\left(\{w(\ell_p) | \ell_p \in \mathit{parents}(\ell)\}\right) & ; \; \mathit{parents}(\ell) \neq \emptyset \end{cases}, \tag{2.12}$$

where $w_0 \in (0, 1]$ is a user-defined parameter. The impurity function is now defined as

$$\mathit{impu}(E) = \frac{1}{|\mathcal{A}_C|} \sum_{\ell \in \mathcal{A}_C} w(\ell) \cdot \frac{\mathit{Var}(E, \ell)}{\mathit{Var}(\mathscr{D}_{\text{TRAIN}}, \ell)},$$

where $|\mathcal{A}_C| = |\mathscr{L}| = L$, and we conveniently denoted a clustering attribute as ℓ. Note that choosing $w_0 = 1$ boils down to ignoring the hierarchy structure and solving a HMLC problem as a MLC one.

For the prototype function, we use the one defined in Equation (2.11). Note that such predictions automatically satisfy the hierarchical constraint which is also one of the advantages of PCTs over, for example, support vector machines for HMLC (Barutcuoglu et al., 2006).

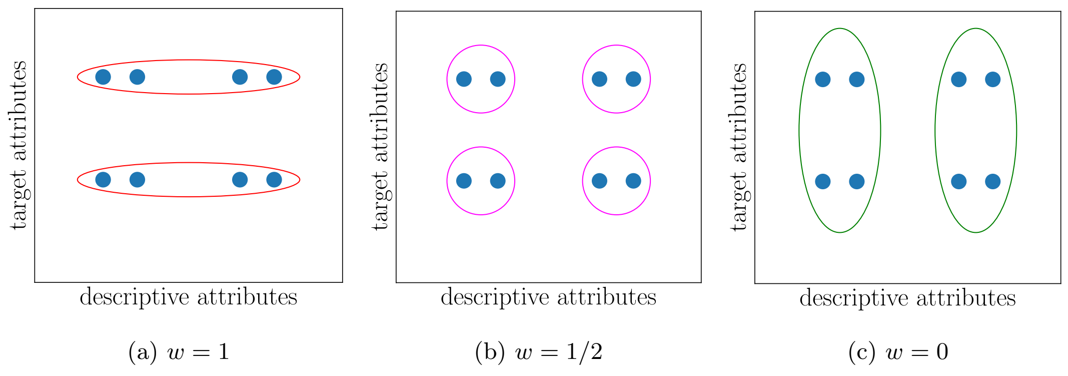


Figure 2.8: Comparison of the clusters in data when the w parameter in the impurity function of Equation (2.13) ranges from $w = 1$ (supervised), over $w = 1/2$ (semi-supervised), to $w = 0$ (unsupervised).

Finally, we make an additional note about compatibility of both impurities that can be used for binary attributes, e.g., in MLC. They can be regarded as numeric or nominal attributes, for which impurities are defined as their variances and Gini indices, respectively.

First, note that the variance of a Bernoulli variable X equals $p(1-p)$ where $p = P(X = 1)$. Moreover, Gini impurity of X equals $2p(1-p)$. Therefore, the induced trees (and feature ranking) are independent of our treatment of binary target variables in MLC.

The same reasoning can be used when a nominal feature x_i is 1-hot encoded, i.e., using the Gini index for a nominal feature x_i yields the same value as using the variance for the $|\mathcal{X}_i|$ numeric features obtained by 1-hot encoding the feature x_i. We can prove this as follows. Recall that Gini index of variable x_i is defined as $\text{Gini}(x_i) = 1 - \sum_{v \in \mathcal{X}_i} p_v^2$ where p_v is the relative frequency of value $v \in \mathcal{X}_i$. Next, note that the $|\mathcal{X}_i|$ features $x_{i,v}$ that the 1-hot encoding results in are Bernoulli variables (with the variance of $p_v(1-p_v)$). Thus, we can deduce that

$$\text{Gini}(x_i) = 1 - \sum_v p_v^2 = \sum_v p_v(1-p_v) = \sum_v \mathit{Var}(x_{i,v}).$$

2.4.4 Semi-supervised and Unsupervised Learning with PCTs

Now that all the predictive modeling tasks have been covered, we can proceed to the semi-supervised learning, and, in the case of PCTs, its special case unsupervised learning. In this section, the function *impu* and *prototype* belong to any (supervised) task, e.g., regression or HMLC mentioned so far.

To extend the PCT induction to the semi-supervised case where many values $\boldsymbol{y}(\boldsymbol{x})$ are missing, one extends the group of clustering attributes $\mathcal{A}_C$, so that the heuristic can make use of unlabelled examples. More precisely, for a semi-supervised version of a task, $\mathcal{A}_C = \mathcal{A}_D \cup \mathcal{A}_T$, where the groups of target and descriptive attributes $\mathcal{A}_D$ and $\mathcal{A}_T$ are defined as in the supervised case, and the impurity function is defined as

$$\mathit{impu}(E) = w \cdot \frac{1}{|\mathcal{A}_T|} \sum_{y_j \in \mathcal{A}_T} \alpha_j \cdot \mathit{impu}(E, y_j) + (1-w) \cdot \frac{1}{|\mathcal{A}_D|} \sum_{x_i \in \mathcal{A}_D} \beta_i \cdot \mathit{impu}(E, x_i), \quad (2.13)$$

where the constants α_j are the normalization constants that make the values $\mathit{impu}(E, y_j)$ comparable. For example, in MTR tasks, $\alpha_j = 1/\mathit{Var}(E, y_j)$. Similar goes for the constants β_i. Values $\mathit{impu}(E, y_j)$ and $\mathit{impu}(E, x_i)$ are computed only from the examples that have a known value of y_j and x_i.

The user-defined parameter $w \in [0, 1]$ defines the influence of the target attributes to the heuristic score. Three different cases are considered:

- Setting the parameter to $w = 1$ corresponds to solving semi-supervised problems in a supervised fashion, i.e., ignoring the unlabeled data. This case is shown in Figure 2.8a. Here, the width of the clusters in the descriptive space, i.e., the impurity of descriptive attributes, does not matter.

- Setting it to $w \in (0, 1)$ corresponds to taking into account both descriptive and target attributes, as shown in Figure 2.8b. Now, both descriptive and target attributes matter when computing the impurity.

- The second extreme case, i.e., $w = 0$ is shown in Figure 2.8c. Here, only the descriptive part of data plays a role when computing the impurity and the target attributes are ignored. This situation is in turn equivalent to the case where there is no target attribute present, i.e., to the unsupervised learning. Indeed, the second sum in the impurity definition (2.13) corresponds to the definition of the impurity function for clustering tasks.

Chapter 3

Extending Feature Ranking to Different Learning Tasks: Basic Notions

In this chapter, we present the basic ideas behind our extensions of feature ranking algorithms to a variety of learning tasks considered in this thesis. These include (supervised) structured output prediction, semi-supervised learning (including SOP) and unsupervised learning, as well as relational classification. We start with feature ranking for SOP based on ensembles of PCTs (Section 3.1), continue with the distance-based feature ranking (Section 3.2), and finish with feature ranking in a relational setting, based on ensembles of relational classification trees (Section 3.3).

3.1 Ensemble-based Feature Ranking for SOP and Unsupervised Learning

Our ensemble-based feature ranking scores for different learning tasks are extensions of the feature ranking scores for classification and regression presented in Section 2.3.4, just like predictive clustering trees are an extension of the standard decision trees for classification and regression for different learning tasks. Equipped with the powerful predictive clustering methodology, we show how one can elegantly extend the Symbolic, Genie3 and Random Forest feature ranking scores defined in Equations (2.4), (2.5) and (2.6) to structured output prediction, unsupervised learning and semi-supervised learning.

3.1.1 Feature Ranking for SOP

We first describe how the three extended scores are defined. We describe what is different for the considered tasks of structured output prediction. This is done for each score separately.

3.1.1.1 Symbolic Score

Arguably, the Symbolic score defined in Equation (2.4) is the most generic of the used scores since – after an ensemble of PCTs is built – it is obtained by counting the number of internal nodes where a given feature appears in the test. However, note that the splits that are closer to the root of the tree influence more examples than the splits deeper in the tree. Thus, we weight every occurrence of a feature in a tree node by the number of examples that reach this node, and define the Symbolic score as

$$importance_{\text{SYMB}}(x_i) = \frac{1}{|\mathcal{E}|}\sum_{\mathcal{T}\in\mathcal{E}}\sum_{\mathscr{N}\in\mathcal{T}(x_i)}\frac{|E(\mathscr{N})|}{|\mathscr{D}_{\text{TRAIN}}|}, \tag{3.1}$$

where the outer sum goes over the trees $\mathcal{T}$ in the ensemble $\mathcal{E}$, and the internal sum goes over the set of internal nodes $\mathcal{T}(x_i)$ where x_i is present in the test. The set of examples that reach a given node $\mathscr{N}$ is denoted by $E(\mathscr{N})$.

Evidently, the Symbolic score is ready to be used in all tasks of structured output prediction (MTR, MLC, HMLC), and no special adaptation of Equation (3.1) is needed.

Note that in our early work (for example, in feature ranking for MTR presented in Chapter 4), we used a slightly different definition, where instead of weighting the occurrences of a feature by the exact number of examples, we weighted them with the term $w^{depth(\mathscr{N})}$ as an approximation of $|E(\mathscr{N})|$, where $w \in (0, 1]$ is a user-defined parameter.

3.1.1.2 Genie3 Score

The integral part of the Genie3 score definition (see Equation 2.5) is computing the quality $h^*(\mathscr{N})$ of the test in an internal node $\mathscr{N}$, for every internal $\mathscr{N}$ in the trees of the ensemble. Given that one knows how to grow an ensemble of PCTs for a specific structured output prediction task means that the heuristic function h for choosing the tests is defined. Thus one can also compute the Genie3 score (and feature ranking) by using the same heuristic function h. Note that function h differs for different types of structured outputs. It follows the template $h(E) = |E|impu(E) - \sum_{E_i\in\mathcal{P}} |E_i|impu(E_i)$, as specified in line 4 of Algorithm 2.2. Eventually, it is the function *impu* that is different for different types of SOP. It is defined as follows:

- MTR:

$$impu(E) = \frac{1}{|\mathcal{A}_C|}\sum_{y\in\mathcal{A}_C}\frac{Var(E,y)}{Var(\mathscr{D},y)}$$

- MLC:

$$\begin{aligned} impu(E) &= \frac{1}{|\mathcal{A}_C|}\sum_{\ell\in\mathcal{A}_C}\frac{\text{Gini}(E,\ell)}{\text{Gini}(\mathscr{D},\ell)} \\ &= \frac{1}{|\mathcal{A}_C|}\sum_{\ell\in\mathcal{A}_C}\frac{Var(E,\ell)}{Var(\mathscr{D},\ell)}, \end{aligned}$$

- HMLC:

$$\begin{aligned} impu(E) &= \frac{1}{|\mathcal{A}_C|}\sum_{\ell\in\mathcal{A}_C} w(\ell)\cdot\frac{\text{Gini}(E,\ell)}{\text{Gini}(\mathscr{D},\ell)} \\ &= \frac{1}{|\mathcal{A}_C|}\sum_{\ell\in\mathcal{A}_C} w(\ell)\cdot\frac{Var(E,\ell)}{Var(\mathscr{D},\ell)}, \end{aligned}$$

as discussed in Section 2.4. Recall from there that $\mathcal{A}_C$ is the set of clustering attributes (i.e., the set of labels for MLC and HMLC), and that $w(\ell)$ is the weight of the label ℓ.

3.1.1.3 Random Forest Score

Whereas the Symbolic and Genie3 scores are parameter-less (given that the parameters of the PCT learning algorithm have been set), computing the Random Forest score demands specifying the error measure *Err* (see Equation 2.6). Of course, this error measure depends on the SOP task at hand.

For MTR, one typically uses the average relative mean squared error, defined as

$$\overline{\mathrm{RMSE}}(\hat{\boldsymbol{y}}) = \frac{1}{|\mathcal{A}_T|} \sum_{y_j \in \mathcal{A}_T} \frac{1}{Var(y_j)} \cdot \frac{1}{|\mathscr{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathscr{D}} (\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j(\boldsymbol{x}))^2$$

where $\hat{\boldsymbol{y}}$ is a MTR model, $\boldsymbol{y}_j$ and $\hat{\boldsymbol{y}}_j$ are the j-th components of the corresponding vectors, $\mathcal{A}_T$ denotes the set of the target attributes, and $\mathscr{D}$ is the set at hand. The per-target squared errors are thus normalized by the variance of the targets. Instead of $\overline{\mathrm{RMSE}}$, one could also use its absolute error analogue where the $(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j(\boldsymbol{x}))^2$ terms are replaced by the terms $|\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j(\boldsymbol{x})|$, and the per-target errors are normalized by $\sqrt{Var(y_j)}$. Other MTR error measures are also possible, but $\overline{\mathrm{RMSE}}$ resembles the tree heuristic (variance reduction) most closely.

As for MLC, there is a variety of error measures, as discussed in Chapter 5. Following a similar reasoning as in the MTR case (that the chosen measure *Err* should resemble the tree building heuristic), the most appropriate option for *Err* is the Hamming loss[1], defined as

$$HammingLoss(\hat{\boldsymbol{y}}) = \frac{1}{|\mathscr{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathscr{D}} \frac{|\boldsymbol{y} \Delta \hat{\boldsymbol{y}}(\boldsymbol{x})|}{|\mathscr{L}|}$$

where $\mathscr{L}$ is the label set and $A \Delta B$ denotes the symmetric difference of the sets A and B.

As for HMLC, the situation is similar to MLC, since here, the weighted versions of the MLC error measures apply. Thus, the default choice for the measure *Err* is the weighted Hamming loss, defined as

$$HammingLoss(\hat{\boldsymbol{y}}) = \frac{1}{|\mathscr{D}||\mathscr{L}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathscr{D}} \sum_{\ell \in \mathscr{L}} w(\ell) \cdot \mathrm{XOR}(\ell \in \boldsymbol{y},\ \ell \in \hat{\boldsymbol{y}}(\boldsymbol{x})) \tag{3.2}$$

where the label weights $w(\ell)$ are defined as in Equation (2.12) and the function XOR maps the conditions $\ell \in \boldsymbol{y}$ and $\ell \in \hat{\boldsymbol{y}}$ to 1 if the label ℓ belongs to exactly one of the sets $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$, and to 0 otherwise.

3.1.2 Unsupervised Feature Ranking

In the unsupervised scenario, we can check that using the definitions of the feature ranking scores from the supervised scenario still makes sense, and take into account a different interpretation of the ranking. This is easy to see for Symbolic ranking, since the attributes that appear in the tests in a tree influence the clustering, and those that appear closer to the root influence the clustering more. Similarly, the Genie3 score now measures how well a given test separates a data subset into two clusters, since the heuristic now measures the spread (impurity) of clusters of data.

As for the Random forest ranking, the measure *Err* is now an adaptation of $\overline{\mathrm{RMSE}}$, defined in the previous section. The adaptation is performed in two steps:

1. Instead of the target space, we use the descriptive space to calculate the error[2].

[1]This might not be obvious, but can be proved with some algebraic manipulation after expanding the term $|\boldsymbol{y} \Delta \hat{\boldsymbol{y}}(\boldsymbol{x})|$.

[2]Recall that when PCTs are used for clustering, all three groups of variables, i.e., descriptive, target and clustering variables coincide, and contain all the variables.

2. Generalizing the definition of the *Err* to also apply to nominal features. While for MTR, MLC and HMLC, all the targets for one task are of the same type, when we use PCTs for clustering, we typically have a mix of numeric and nominal variables.

For numeric features, one can simply replace the quadratic terms $(\boldsymbol{y}_j - \hat{\boldsymbol{y}}_j(\boldsymbol{x}))^2$ by the terms $(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)^2$, where the predicted value $\hat{\boldsymbol{x}}_i$ is the i-th component of the centroid of a cluster that corresponds to the leaf into which the example $\boldsymbol{x}$ belongs. For nominal features, these terms are replaced by $\mathbf{1}[\boldsymbol{x}_i \neq \hat{\boldsymbol{x}}_i]$. In both cases, we denote these error terms as $e(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)$.

Finally, if we recall the normalisation weights β_i from the semi-supervised learning impurity function (Equation (2.13)), defined as

$$\beta_i = \begin{cases} \frac{1}{Var(x_i)} & ;\ \mathcal{X}_i \subseteq \mathbb{R} \\ \frac{1}{\mathrm{Gini}(x_i)} & ;\ \mathcal{X}_i \not\subseteq \mathbb{R} \end{cases},$$

then we can finally define the measure *Err* as

$$Err = \frac{1}{|\mathcal{A}_D|} \sum_{x_i \in \mathcal{A}_D} \beta_i \frac{1}{|\mathscr{D}|} \sum_{\boldsymbol{x} \in \mathscr{D}} e(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i), \tag{3.3}$$

where $\mathcal{A}_D$ denotes the set of all descriptive attributes.

3.1.3 Semi-supervised Feature Ranking

In the previous two subsections we have covered our extended feature ranking approaches for the cases of supervised and unsupervised learning. Now, we do so for the most general setting, namely, semi-supervised learning.

Again, the Symbolic and Genie3 scores do not need any modifications even though the tree induction heuristic h is now changed. The Symbolic score is independent of h. For the Genie3 score, no changes are needed once we have changed h.

What remains is to define the Random forest score for semi-supervised feature ranking. In the previous subsection, we have already defined part of the score (Equation (3.3)) that corresponds to the unsupervised part of the SSL impurity from Equation (2.13). Following this definition and the one given above, we define the error measure *Err* for semi-supervised Random Forest score as

$$Err = w \cdot \frac{1}{|\mathcal{A}_T|} \sum_{y_j \in \mathcal{A}_T} \alpha_j \cdot \frac{1}{|\mathscr{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathscr{D}} e(\boldsymbol{y}_j, \hat{\boldsymbol{y}}_j(\boldsymbol{x})) + (1-w) \cdot \frac{1}{|\mathcal{A}_D|} \sum_{x_i \in \mathcal{A}_D} \beta_i \cdot \frac{1}{|\mathscr{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathscr{D}} e(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i),$$

where the normalization constants α_j are defined analogously to β_i, and depend on the predictive modeling task. Parameter w determines the amount of supervision: When $w = 1$, we have fully supervised learning, and when $w = 0$, we have unsupervised learning. The values $e(\boldsymbol{y}_j, \hat{\boldsymbol{y}}_j(\boldsymbol{x}))$ are defined in light of the SOP task at hand. For example, for SSL HMLC problems, we have (following Equation (3.2))

$$e(\boldsymbol{y}_\ell, \hat{\boldsymbol{y}}_\ell(\boldsymbol{x})) = w(\ell) \cdot \mathrm{XOR}(\ell \in \boldsymbol{y},\ \ell \in \hat{\boldsymbol{y}}(\boldsymbol{x}))$$

and we have $\mathcal{A}_T = \mathscr{L}$, and $\alpha_j = \alpha_\ell$. Normalisation constants α_ℓ are computed using similar error terms, but with $\hat{\boldsymbol{y}}(\boldsymbol{x})$ replaced by the predictions of the default model learned on the whole data set.

3.2 Distance-based Feature Ranking

In this section, we present the extensions of the Relief family of algorithms to the same tasks as in the previous section, i.e., to structured output prediction, unsupervised and semi-supervised learning (including SSL for SOP). As opposed to ensemble-based feature rankings, the Relief family is not directly connected to the predictive clustering framework. However, we keep the notation for the three groups of attributes (descriptive, clustering and target), since this makes explanation much easier.

Algorithm 3.1: RReliefF($\mathscr{D}_{\text{TRAIN}}$, I, K) with highlights

```
 1: $\boldsymbol{w}$ = zero list of length $n$
 2: $\boldsymbol{P}_{\textit{diffFeat, diffTarget}}$, $\boldsymbol{P}_{\textit{diffFeat}}$ = zero lists of length $n$
 3: $P_{\textit{diffTarget}} = 0.0$
 4: for $\iota = 1, 2, \dots, I$ do
 5:     $\boldsymbol{r}$ = random example from $\mathscr{D}_{\text{TRAIN}}$
 6:     $\boldsymbol{n}_1, \boldsymbol{n}_2, \dots, \boldsymbol{n}_K$ = $K$ nearest neighbors of $\boldsymbol{r}$
 7:     for $k = 1, 2, \dots, K$ do
 8:         $P_{\textit{diffTarget}} \mathrel{+}= d_{\mathcal{Y}}(\boldsymbol{r}, \boldsymbol{n}_k)$
 9:         for $i = 1, 2, \dots, n$ do
10:             $\boldsymbol{P}_{\textit{diffFeat}}[i] \mathrel{+}= d_i(\boldsymbol{r}, \boldsymbol{n}_k)$
11:             $\boldsymbol{P}_{\textit{diffFeat, diffTarget}}[i] \mathrel{+}= d_i(\boldsymbol{r}, \boldsymbol{n}_k) d_{\mathcal{Y}}(\boldsymbol{r}, \boldsymbol{n}_k)$
12: for $i = 1, 2, \dots, F$ do
13:     $w_i = \frac{\boldsymbol{P}_{\textit{diffFeat, diffTarget}}[i]}{P_{\textit{diffTarget}}} - \frac{\boldsymbol{P}_{\textit{diffFeat}}[i] - \boldsymbol{P}_{\textit{diffFeat, diffTarget}}[i]}{1 - P_{\textit{diffTarget}}}$
14: return $\boldsymbol{w}$
```

The key observation for extending the Relief family is that RReliefF (given in Algorithms 2.4 and 3.1) operates only with distances along individual descriptive attributes (features) and along the single numeric target. RReliefF can be thus generalized to SOP if one appropriately defines the target distance $d_{\mathcal{Y}}$ on the target space $\mathcal{Y}$. The critical lines in the pseudocode that enable the generalization and need to be changed are lines 8 and 11, where $d_{\mathcal{Y}}$ appears. Note that the distance d_i (lines 10 and 11) does not need to be changed.

With some further observations and modifications, one can extend the obtained algorithm to the unsupervised and the semi-supervised scenarios. Note that classification versions, namely Relief and ReliefF, are not appropriate for generalization, since their integral part is to compute the neighbors of an example that belong to the same and to different classes, a notion that only works in the classification context.

3.2.1 Feature Ranking for SOP

In this section, we present the instantiations of the distance $d_{\mathcal{Y}}$ for the different tasks of structured output prediction.

3.2.1.1 Feature Ranking for MTR

We start with MTR. Here, we define $d_{\mathcal{Y}}$ in an analogous way to $d_{\mathcal{X}}$, specified in Equation (2.8), namely

$$d_{\mathcal{Y}}(\boldsymbol{y}^1, \boldsymbol{y}^2) = \frac{1}{|\mathcal{A}_T|} \sum_{y_j \in \mathcal{A}_T} d_j(\boldsymbol{y}^1, \boldsymbol{y}^2),$$

where the numeric part of the distance d_j (given in Equation (2.7)) applies.

3.2.1.2 Feature Ranking for MLC

In MLC, we denote $\mathscr{L} = \{\ell_1, \dots, \ell_L\}$ and represent a set of labels $S \subseteq \mathscr{L}$ as its incidence vector $\boldsymbol{s}$ (as explained in Section 2.4.3). We can then follow the above definition for MTR and define the distance

$$d(S^1, S^2) = \frac{1}{L}\sum_{j=1}^{L} d_j(\boldsymbol{s}^1, \boldsymbol{s}^2)$$

This distance definition has been used previously in an MLC extension of RReliefF (Reyes et al., 2015). However, we note that $d(S^1, S^2)$, as defined above, may not be the most appropriate for MLC problems. Namely, it can be decomposed to the distances $d_j(\boldsymbol{s}^1, \boldsymbol{s}^2)$ and thus ignores the potential interactions among the labels.

Moreover, we note that this distance actually equals

$$d_{Hamming}(S^1, S^2) = \frac{|S^1 \Delta\, S^2|}{L}, \tag{3.4}$$

which mimics Hamming loss – one of the MLC evaluation measures. As this is one of the many MLC evaluation measures, we propose to use three other distances, namely

$$\begin{aligned} d_{Accuracy}(S^1, S^2) &= 1 - |S^1 \cap S^2| \,/\, |S^1 \cup S^2| \\ d_{F1}(S^1, S^2) &= 1 - 2|S^1 \cap S^2| \,/\, (|S^1| + |S^2|) \\ d_{SubsetAcc}(S^1, S^2) &= \mathbf{1}\left[S^1 \neq S^2\right]. \end{aligned}$$

They are all based on the corresponding MLC evaluation measures (MLC accuracy, MLC F_1 measure and subset accuracy), and cannot be decomposed into single-label distances. In Chapter 5, we show – among other things – that they outperform the distance $d_{Hamming}$.

3.2.1.3 Feature Ranking for HMLC

For HMLC, one can use the weighted versions of all the MLC distance instantiations, by defining $w = \sum_{j=1}^{L} w(\ell_j)$ and

$$\begin{aligned} |S| &= \frac{1}{w}\sum_{j=1}^{L} w(\ell_j) \cdot \mathbf{1}[\ell_j \in S], \\ |S^1 \cup S^2| &= \frac{1}{w}\sum_{j=1}^{L} w(\ell_j) \cdot \max\{\mathbf{1}[\ell_j \in S^1], \mathbf{1}[\ell_j \in S^2]\}, \\ |S^1 \cap S^2| &= \frac{1}{w}\sum_{j=1}^{L} w(\ell_j) \cdot \min\{\mathbf{1}[\ell_j \in S^1], \mathbf{1}[\ell_j \in S^2]\}. \end{aligned}$$

This generalizes the first three MLC distances given above, whereas the distance $d_{SubsetAcc}$ remains the same.

3.2.2 Unsupervised Feature Ranking

The predictive clustering terminology is useful when defining the unsupervised version of Relief, since – knowing that $d_{\mathcal{Y}}$ operates in the target space and $d_{\mathcal{X}}$ in the descriptive space – we can precisely define the algorithm by stating that, in unsupervised learning, $\mathcal{A}_T = \mathcal{A}_D$. As a consequence, $d_{\mathcal{X}} = d_{\mathcal{Y}}$.

3.2.3 Semi-supervised Feature Ranking

The semi-supervised version of Relief is considerably more complex (since it can cover all of the tasks mentioned so far) and needs more than one modification. Its pseudocode is given in Algorithm 3.2. In the remainder of the section, we describe the necessary modifications.

Algorithm 3.2: SSL-Relief($\mathscr{D}_{\text{TRAIN}}$, I, K, $[w_0, w_1]$)

```
1: importance = zero list of length $n$
2: $\boldsymbol{P}_{\text{diffAttr, diffCluster}}$, $\boldsymbol{P}_{\text{diffAttr}}$ = zero lists of length $n$
3: $P_{\text{diffCluster}} = 0.0$
4: $\boldsymbol{w} = \mathit{computeInstanceInfluence}(\mathscr{D}_{\text{TRAIN}}, w_0, w_1)$
5: $s = 0$                         # sum of weights of the pairs, used in normalization
6: for $\iota = 1, 2, \dots, m$ do
7:     $\boldsymbol{r}$ = random example from $\mathscr{D}_{\text{TRAIN}}$
8:     $\boldsymbol{n}_1, \boldsymbol{n}_2, \dots, \boldsymbol{n}_K = K$ nearest neighbors of $\boldsymbol{r}$
9:     for $k = 1, 2, \dots, K$ do
10:        $w = \boldsymbol{w}[\boldsymbol{r}] \cdot \boldsymbol{w}[\boldsymbol{n}_k]$
11:        $s \mathrel{+}= w$
12:        if $\boldsymbol{r}$ and $\boldsymbol{n}_k$ are labeled then
13:            $d_{\text{cluster}} = d_{\mathcal{Y}}$
14:        else
15:            $d_{\text{cluster}} = d_{\mathcal{X}}$
16:        $P_{\text{diffCluster}} \mathrel{+}= w\, d_{\text{cluster}}(\boldsymbol{r}, \boldsymbol{n}_k)$
17:        for $i = 1, 2, \dots, n$ do
18:            $\boldsymbol{P}_{\text{diffAttr}}[i] \mathrel{+}= w\, d_i(\boldsymbol{r}, \boldsymbol{n}_k)$
19:            $\boldsymbol{P}_{\text{diffAttr, diffCluster}}[i] \mathrel{+}= w\, d_i(\boldsymbol{r}, \boldsymbol{n}_k)\, d_{\text{cluster}}(\boldsymbol{r}, \boldsymbol{n}_k)$
20: for $i = 1, 2, \dots, D$ do
21:     importance[$i$] = $\frac{\boldsymbol{P}_{\text{diffAttr, diffCluster}}[i]}{P_{\text{diffCluster}}} - \frac{\boldsymbol{P}_{\text{diffAttr}}[i] - \boldsymbol{P}_{\text{diffAttr, diffCluster}}[i]}{s - P_{\text{diffCluster}}}$
22: return importance
```

SSL-Relief takes as input the standard parameters ($\mathscr{D}_{\text{TRAIN}}$, the number of iterations I, and the number of Relief neighbors K), as well as an interval $[w_0, w_1] \subseteq [0, 1]$, from which the influence levels of $\boldsymbol{r}$-$\boldsymbol{n}$ pairs are computed (line 4): First, for every $(\boldsymbol{x}, \boldsymbol{y}) \in \mathscr{D}_{\text{TRAIN}}$, we find the distance $d_{\boldsymbol{x}}$ to its nearest labeled neighbor. If $d = 0$, i.e., the value $\boldsymbol{y}$ is known, the influence w of this example is set to 1. Otherwise, the influence of the example $(\boldsymbol{x}, \boldsymbol{y})$ is defined by a linear function $d_{\boldsymbol{x}} \mapsto w(d_{\boldsymbol{x}})$ that goes through the points $(\max_{(\boldsymbol{x}', ?)} d_{\boldsymbol{x}'},\ w_0)$ and $(\min_{(\boldsymbol{x}', ?)} d_{\boldsymbol{x}'},\ w_1)$ where ? denotes the missing target values. These influence values are analogues to the w parameter in the definition of the impurity function for learning PCTs in SSL defined in Equation (2.13).

When iterating over the neighbors, the influence of the neighbouring pair is defined as the product of influence values of the examples in the pair. The clustering distance is set to either $d_{\mathcal{Y}}$ (when both examples are labeled) or $d_{\mathcal{X}}$ (otherwise). After that, the weights, this time denoted by **importance**, are updated as in the (un)supervised case.

The algorithm is designed to have the supervised and unsupervised versions described above as special cases. Thus, if no target values are missing, one obtains the supervised version of Relief for the task at hand. Similarly, if there is no target attribute or no example has a known target value, the algorithm boils down to the unsupervised version of Relief.

3.3 Feature Ranking for Relational Classification

In Section 2.1.2, we have already given some basic information about relational data. Now, we first briefly present how to convert that data format to a format usable by a relational tree learning algorithm. Then, we briefly present how a relational tree is learned and also give the definition of feature ranking in this context.

3.3.1 Data Conversion

Algorithms love a uniform representation of data. For example, if we are classifying pictures into cats and dogs, the model typically assumes that all the pictures have the same dimensions. This also holds for algorithms whose input are relational data, e.g., the IMDB data whose schematic representation has first been given in Figure 2.1 and is shown again in Figure 3.1 to make reading easier.

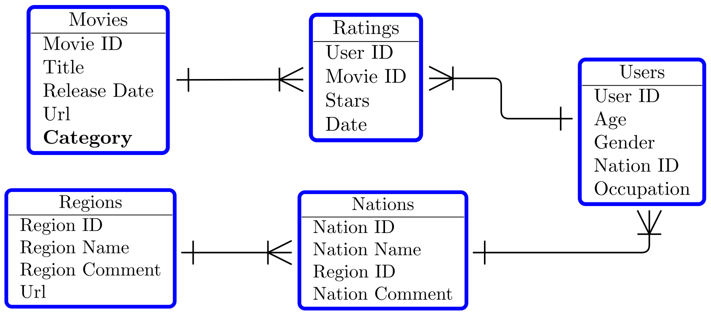


Figure 3.1: A copy of the schema of the IMDB data set, as presented in Figure 2.1.

As mentioned in Section 2.1.2, columns in tables of a relational data set are of two types, i.e., some columns play the role of the features or target variable (e.g., release date and category of a movie), whereas the ID columns implicitly define the relations among the tables.

To simplify the implementation of our relational trees learning algorithm, we *normalize* the data in the following way. First, we can assume that every table in the data set is of the same form as Table 3.1. In the table, object type corresponds to the type of the

Table 3.1: The structure of a generic table from a relational data set.

object type							
object ID	foreign ID$_1$	$\cdots$	foreign ID$_J$	x_1 (type$_1$)	$\cdots$	x_n (type$_n$)	$\boldsymbol{y}$
e_1	f_{11}	$\cdots$	$f_{1,J}$	x_{11}	$\cdots$	$x_{1,n}$	$\boldsymbol{y}_1$
$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$
e_m	$f_{m,1}$	$\cdots$	$f_{m,J}$	$x_{m,1}$	$\cdots$	$x_{m,n}$	$\boldsymbol{y}_m$

objects described in the table (e.g., *Rating*). The values in the first column (object ID) give the IDs of the objects (for example, *Rating ID*). The J foreign IDs, $J \geq 0$, reference the objects from the other tables that are related and somehow further describe the objects in this table (e.g., if the table describes ratings, the foreign ID$_1$ may be *User ID*, and the foreign ID$_2$ may be *Movie ID*). The columns x_i denote the features that directly describe the objects in the table (e.g., stars and date). Every feature has its own type. The last column is the target column. Note that the target column is actually present in at most one table in the data set (e.g., column category in the Movies table). As for the example ID column, it may happen that it is not explicitly present, since the foreign IDs completely determine it (for example, Rating ID is not necessary since the User ID and Movie ID completely determine the rating ID).

For every table, some descriptive relations are created, i.e., one for each feature x_i. If the target column is present in the table, an additional (target) relation is created. Here, we only present how to create a relation from a feature column since the creation of the target relation is analogous. The creation of the relations depends on whether the object ID column is present in the table or not. However, in both cases, the created relations are represented extensionally as the set of the tuples that are in this relation.

If the object ID column is present in the table, the creation of relations proceeds as follows. For every feature x_i, a binary relation is created, e.g., $r_{x_i} = \{(e_k, x_{k,i}) \mid 1 \leq k \leq m\}$, where we keep the notation from Table 3.1. Since foreign ID columns also behave as features, a relation r_{f_j} is created also for every foreign ID column, in an analogous way.

If the object ID column is not present in the table, we assume without loss of generality that first J' foreign IDs, $0 < J' \leq J$, define the object ID. Then, a $(J'+1)$-ary relation is created for every feature x_i, and also for every foreign ID f_j, such that $j > J'$. For example, $r_{x_i} = \{(f_{k,1}, \dots, f_{k,J'}, x_{k,i}) \mid 1 \leq k \leq m\}$.

Some concrete examples of the created relations are:

- $r_{\text{age}} = \{(\texttt{Ana}, 21), (\texttt{Bob}, 33), \dots\}$ (created from the feature age in the Users table),
- $r_{\text{gender}} = \{(\texttt{Ana}, \texttt{F}), (\texttt{Bob}, \texttt{M}), \dots\}$ (created from the feature gender in the Users table),
- $r_{\text{stars}} = \{(\texttt{Ana}, \texttt{titanic}, 8), (\texttt{Ana}, \texttt{psycho}, 4), (\texttt{Bob}, \texttt{psycho}, 9), \dots\}$ (created from the feature stars in the Ratings table – note that Rating ID is not explicitly present, but rather defined by User ID and Movie ID).

Thus, the input to our relational trees is a set of descriptive relations (together with the known types of their components), and a target relation. Instead of growing a single tree, one can build an ensemble of relational trees. Again, growing an ensemble results in better predictive performance and in worse interpretability of the predictive model.

This motivates us to extend the definition of feature ranking scores to the relational case and embed the Symbolic and Genie3 scores into the ensembles. The definition of relational feature ranking, and a brief description of the relational versions of the two scores, is given in the next subsection.

3.3.2 Relational Feature Ranking Scores

The definition of feature ranking in relational context is similar to the definitions of feature ranking for tabular data in the context of supervised and unsupervised learning. However, since we do not deal with relational data sets in the unsupervised learning context in this thesis, we only give the definition for the case where a target relation (and a target attribute) is present.

Feature ranking for relational data. The input of a feature ranking algorithm is a data set that consists of a set $\mathcal{R}_d$ of descriptive relations and a target relation $r_{\boldsymbol{y}}$. The output of the feature ranking algorithm is a list of scores $\mathit{importance}(r_d) \in \mathbb{R}$, for every descriptive relation r_d in the data set, where the scores reflect the influence of relations r_d on the target relation $r_{\boldsymbol{y}}$.

However, to extend the ensemble-based feature ranking scores to relational data, one has to first specify the possible splits in the trees, and then adapt the scores. In the remainder of the section, we give a brief introduction to both steps.

3.3.2.1 Relational Trees

The algorithm that we develop for growing a relational tree (for classification) follows the algorithm for growing a PCT, as given in Algorithms 2.1 and 2.2. Since the target relation behaves precisely as a standard target variable, one can use the same heuristic for growing a tree, and only specify candidate tests that are evaluated in internal nodes of the tree.

We give an introduction to the candidate tests (splits) in relational trees by following the schema of the IMDB data set in Figure 3.1. The basic set of splits contains conditions, such as

- *For a given movie, is there a 10-star rating of the movie?*, and
- *For a given movie, is there a user that rated this movie and comes from Spain?*,

as presented in the FOIL algorithm (Quinlan & Cameron-Jones, 1993). These conditions correspond to the existentially quantified conjunctions of descriptive relations, e.g.,

- $\exists \mathit{User} : \; r_{\text{stars}}(\text{titanic}, \mathit{User}, 10)$
- $\exists \mathit{User} \; \exists \mathit{Stars} \; \exists \mathit{Nation} : \; r_{\text{stars}}(\text{titanic}, \mathit{User}, \mathit{Stars}) \; \wedge \; r_{\text{nation ID}}(\mathit{User}, \mathit{Nation}) \; \wedge \; r_{\text{nation name}}(\mathit{Nation}, \text{spain})$

where the given movie was set to Titanic. Thus, in general, we form a conjunction of relations in which some components are grounded (have their value set to some constant value, e.g., spain) and some are ungrounded (variables, e.g., *Nation*).

In our work, we go beyond the existential quantification and use aggregates to define a richer set of splits, which is to some extent similar to the extension of TILDE (Blockeel & Raedt, 1998) by Vens (2007). By doing so, we can pose conditions, such as

- *For a given movie, is its average number of stars larger than ϑ* (for some threshold ϑ?), and
- *For a given movie, does the majority of the users that rated this movie belong to a nation from the set N?* (for some set of nations N).

These conditions are formalized as

- $\text{mean}_{(\mathit{User},\mathit{Stars})} \, r_{\text{stars}}(\text{titanic}, \mathit{User}, \mathit{Stars}) > 8.5$
- $$\begin{aligned} \text{mode}_{(\mathit{User},\mathit{Stars})} \, \text{mode}_{\mathit{Nation}} \, \text{mode}_{\mathit{NationName}} \; & r_{\text{stars}}(\text{titanic}, \mathit{User}, \mathit{Stars}) \\ & \wedge r_{\text{nation ID}}(\mathit{User}, \mathit{Nation}) \\ & \wedge r_{\text{nation name}}(\mathit{Nation}, \mathit{NationName}) \\ \in \{\text{seychelles}, \text{slovenia}, \text{spain}\} & \end{aligned}$$

where the given movie was set to Titanic, the threshold was set to $\vartheta = 8.5$ and the set of nations was set to $N = \{\text{seychelles}, \text{slovenia}, \text{spain}\}$. Note that aggregating over the variables *Nation* and *NationName* is necessary only for technical reasons (and corresponds to extracting x from a singletone $\{x\}$) since the corresponding relations are -to-one relations. Note also that other aggregates than *mean* and *mode* are possible, namely, *count*, *min* and *max*. Moreover, using only *count* and $\vartheta = 0$ is equivalent to existential quantification, thus these tests indeed extend the ones presented above. An example of a constructed relational tree is shown in Figure 3.2.

The features that contain more relations are more complex and possibly more expressive, but, on the other hand, the number of candidate splits is exponential in the number of relations in the conjunctions. Thus, we allow that in every child node (should it be an

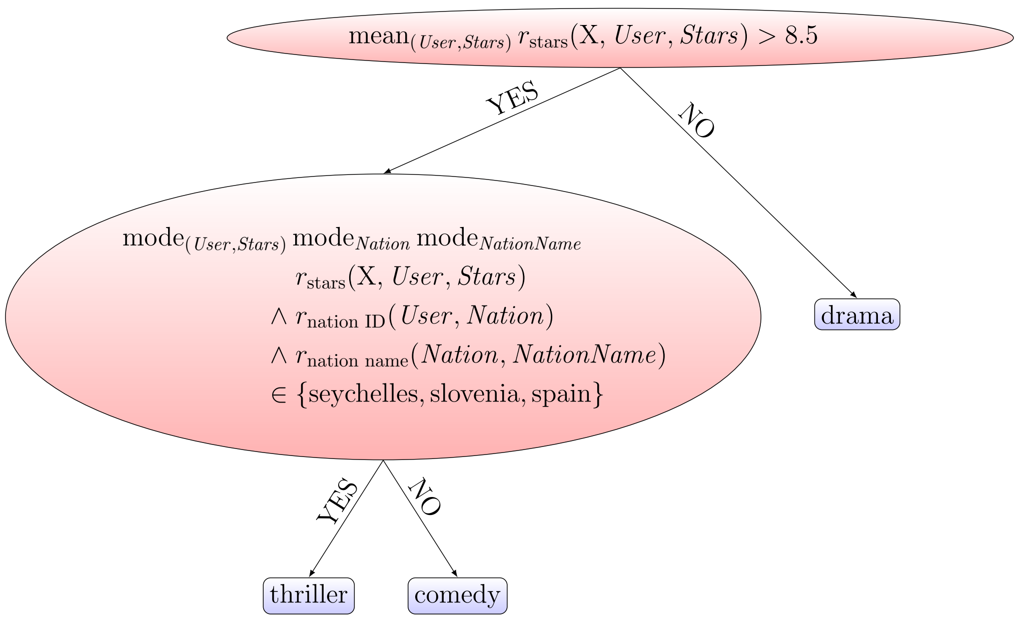

$\text{mean}_{(\mathit{User},\mathit{Stars})}\, r_{\text{stars}}(\text{X}, \mathit{User}, \mathit{Stars}) > 8.5$
YES
NO
$\text{mode}_{(\mathit{User},\mathit{Stars})}\, \text{mode}_{\mathit{Nation}}\, \text{mode}_{\mathit{NationName}}$
$r_{\text{stars}}(\text{X}, \mathit{User}, \mathit{Stars})$
$\wedge\, r_{\text{nation ID}}(\mathit{User}, \mathit{Nation})$
$\wedge\, r_{\text{nation name}}(\mathit{Nation}, \mathit{NationName})$
$\in \{\text{seychelles}, \text{slovenia}, \text{spain}\}$
drama
YES
NO
thriller
comedy

Figure 3.2: An example relational tree for assigning categories (genres) to movies.

internal node), the conjunction of relations in the split is a continuation of the conjunction in the parent node of this node. When generating the possible splits, a look-ahead parameter ℓ is introduced into the algorithm: In each child node, at most ℓ relations are added to the parent's conjunction.

That way, a tree can contain more and more complex features when growing. An example of a relational tree is given in Figure 3.2. A movie X enters the tree and is sent down the tree, according to the tests in the internal nodes. When the movie reaches a leaf, the corresponding genre is predicted. Note that the conjunction of the relations in the YES-child of the root node was constructed by continuing the conjunction (of a single relation) in the root node. Thus, we can deduce that the ℓ parameter was set to at least 2, since the split in the YES-child of the root node contains three relations, whereas the root node contains only one.

3.3.2.2 Genie3 and Symbolic Score for Relational Learning

One can regard the left-hand side of a condition in a node $\mathscr{N}$ as a (newly constructed) feature $f(\mathscr{N})$, whose building blocks are descriptive relations from the data set. Thus, once the possible splits are defined, we can extend the ensemble-based feature ranking scores to the relational setting, and estimate the relevance of the descriptive relations. However, for extending the Random forest score, one would need to define permutations of the values of the relations. Since it is not clear how to define this for not -to-one relations or non-binary relations, we limit ourselves to the Genie3 and Symbolic scores.

Since the motivation for both scores remains the same and we know how to reward an occurrence of a feature $f(\mathscr{N})$ in a node, the remaining step is to transfer the reward of the feature in the split to the descriptive relations themselves. To do so, first note that a constructed feature can be useless if even a single descriptive relation is left out. Thus, all the descriptive relations that appear in the constructed feature should be equally rewarded. Second, if the conjunction of relations in a child node is a continuation of the conjunction in the parent node, some relations have already been rewarded in the parent node. Thus, in the current node, only *the new part* of the conjunction is rewarded.

Thus, the relational Symbolic and Genie3 score are defined as

$$\begin{aligned}
importance_{\mathrm{SYMB}}(r_d) &= \frac{1}{|\mathcal{E}|}\sum_{\mathcal{T}\in\mathcal{E}}\sum_{\mathscr{N}\in\mathcal{T}(r_d)} P[r_d|f(parent(\mathscr{N}))]\cdot\frac{|E(\mathscr{N})|}{|\mathscr{D}_{\mathrm{TRAIN}}|}\\
importance_{\mathrm{GENIE3}}(r_d) &= \frac{1}{|\mathcal{E}|}\sum_{\mathcal{T}\in\mathcal{E}}\sum_{\mathscr{N}\in\mathcal{T}(r_d)} P[r_d|f(parent(\mathscr{N}))]\cdot h^*(\mathscr{N})
\end{aligned}$$

where $P[r_d|f(parent(\mathscr{N}))]$ denotes the proportion of appearances of relation r_d in the part of the feature $f(\mathscr{N})$ that is not already present in the feature in the node $parent(\mathscr{N})$, $\mathcal{T}$ denotes a tree in an ensemble $\mathcal{E}$, and $\mathcal{T}(r_d)$ the set of internal nodes $\mathscr{N}$ where the descriptive relation r_d appears in the test.

Let us give a concrete example of computation of the Symbolic score, using the tree in Figure 3.2. Here, we assume that this is the only tree in the ensemble $\mathcal{E}$, and that half of the training examples that reach a given internal node in the tree are sent to each of the node's children.

First, we compute the value of the Symbolic score for the relation r_{stars}. It appears both in the root node, where its reward equals $1 \cdot 1$, and in its YES-child, where its reward equals $0 \cdot 1/2$, since r_{stars} is not in the new part of the YES-child's test. Thus, $importance_{\mathrm{SYMB}}(r_{\text{stars}}) = 1$.

The remaining two relations ($r_{\text{nation ID}}$ and $r_{\text{nation name}}$) with a positive Symbolic score both appear only in the YES-child of the root, where each of them is rewarded $1/2 \cdot 1/2$. Thus, $importance_{\mathrm{SYMB}}(r_{\text{nation ID}}) = importance_{\mathrm{SYMB}}(r_{\text{nation name}}) = 1/4$.

Chapter 4

Feature Ranking for Multi-Target Regression

In this Chapter, we present in detail the first set of contributions of the thesis: those concerning the supervised learning setting and the structured output prediction task of multi-target regression (MTR). Recall from the introductory sections that the task of MTR is to simultaneously predict the values of several numeric targets (Section 2.2.3). Also recall that we presented our feature ranking methods for MTR in Sections 3.1.1 and 3.2.1. Our contributions in this context include:

1. An extension of ensemble-based feature ranking scores from the context of STR to the context of MTR.

2. An extension of distance-based feature ranking scores from the context of STR to the context of MTR.

3. An extensive experimental evaluation of the newly proposed feature importance scores for MTR on a collection of benchmark datasets assessing the performance of the scores individually and in cross-comparison.

Chronologically, we first proposed the extension of ensemble-based feature ranking methods from the STR to the MTR setting in a paper presented at the DS-2017 conference (Petković et al., 2017). As outlined in Chapter 3, the proposed feature ranking methods consist of pairs of ensemble generation methods (bagging, random forests and extra trees) and scores (Symbolic, Genie3, and Random Forest scores). We evaluated the performance of the methods on a range of benchmark datasets, finding good parametrizations for the methods in the process. The main findings are that all of the eight proposed methods (pairs of ensemble methods and scores) yield relevant feature rankings and that the best performing method is the one that uses random forests for learning the ensemble and Genie3 for calculating the feature importance scores. Moreover, this method is also computationally efficient: random forests are among the most efficient ensemble learning methods and computing the Genie3 score only adds a small computational cost of a single traversal of each tree in the ensemble.

We then extended our work on feature ranking for MTR in several directions and presented this extended version in a paper published in the Machine Learning journal (Petković, Kocev, et al., 2020), which is included in this Chapter in its entirety. In this paper, (1) we propose an extension of RReliefF for MTR and a MTR-via-STR version of it and provide a theoretical and empirical analysis of its computational complexity; (2) we provide a theoretical and empirical analysis of the computational complexity of ensemble-based feature ranking for MTR; (3) we analyze the performance and efficiency of the MTR-via-STR versions of both ensemble-based and RReliefF-based rankings; and finally (4) we evaluate the performance of all of the above feature ranking methods for MTR, analyzing the influence of the method's parameters on their performance.

The comprehensive experimental evaluation reveals the following. First, the ensemble-based ranking methods yield relevant feature rankings when the scores are computed using the most appropriate tree ensemble. The evaluation identified two best ensemble-based ranking methods (in terms of performance and efficiency): these are the Genie3 and Symbolic scores calculated from random forests. Second, the evaluation shows that Relief distance-based rankings can yield relevant rankings when the parameters of Relief are optimized, i.e., the numbers of iterations I and neighbors K in the algorithm are set to appropriate values, e.g., I is set to the size of the training set and K is set to 15. When we compared the top ensemble-based and Relief-based ranking methods, there were no statistically significant differences among them. Further analysis revealed that this is due to the strong variability in the properties of the data sets. Nevertheless, we can accurately determine which of the two groups (ensemble-based or Relief-based) of methods will perform better on a given data set.

Third, the analysis shows that the native MTR versions of ensemble-based ranking methods are statistically significantly faster than their MTR-via-STR counterparts (where a ranking is built for every target separately, and the final ranking is obtained by averaging the scores from the per-target rankings), but not worse in terms of quality. In the case of Relief, there are no differences. All in all, we recommend using the Genie3 and the Symbolic scores coupled with the random forest ensemble construction method, since they both yield relevant rankings and are fast enough.

This Chapter addresses the following hypotheses defined in the introduction:

H1: It is possible to extend ensemble- and distance-based feature ranking approaches to the unsupervised feature ranking task, to the tasks of supervised SOP (i.e., MTR, MLC, and HMLC), and to their semi-supervised versions.

H3: The proposed ensemble- and distance-based approaches yield relevant and state-of-the-art feature rankings for MTR.

H6: For unsupervised, MTR, MLC and HMLC problems, the ensemble-based feature ranking approaches on average outperform the distance-based approaches to feature ranking when the number of features is extremely high.

These three hypotheses are confirmed with the design and implementation of the MTR feature ranking approaches presented in this Chapter and the experimental study comparing their performance. Hypothesis H3 is completely confirmed by the results presented in this Chapter. For hypotheses H1 and H6, the parts pertaining to MTR are confirmed here, while their remaining parts are addressed in the forthcoming chapters.

The paper included in this Chapter is:

- Petković, M., Kocev, D. and Džeroski, S. Feature ranking for multi-target regression. *Machine Learning.* 109, 1179–1204 (2020).

The contribution of Matej Petković to this paper is as follows. MP contributed to the design of the ensemble-based and distance-based feature ranking methods for MTR and implemented these methods in computer code. He also participated in designing the experiments, carried out the experiments, and processed their results. He drafted the paper and revised it following the feedback from the co-authors and reviewers.

Machine Learning (2020) 109:1179–1204
https://doi.org/10.1007/s10994-019-05829-8

Feature ranking for multi-target regression

Matej Petković[1,2] · Dragi Kocev[1,2] · Sašo Džeroski[1,2]

Received: 1 July 2018 / Revised: 12 June 2019 / Accepted: 6 July 2019 / Published online: 27 August 2019

Abstract
In this work, we address the task of feature ranking for multi-target regression (MTR). The task of MTR concerns problems with multiple continuous dependent/target variables, where the goal is to learn a model for predicting all of them simultaneously. This task is receiving an increasing attention from the research community, but performing feature ranking in the context of MTR has not been studied thus far. Here, we study two groups of feature ranking scores for MTR: scores (Symbolic, Genie3 and Random Forest score) based on ensembles (bagging, random forests, extra trees) of predictive clustering trees, and a score derived as an extension of the RReliefF method. We also propose a generic data-transformation approach to MTR feature ranking and thus have two versions of each score. For both groups of feature ranking scores, we analyze their theoretical computational complexity. For the extension of the RReliefF method, we additionally derive some theoretical properties of the scores. Next, we extensively evaluate the scores on 24 benchmark MTR datasets, in terms of the quality of the ranking and the computational complexity of producing it. The results identify the parameters that influence the quality of the rankings, reveal that both groups of methods produce relevant feature rankings, and show that the Symbolic and Genie3 score, coupled with random forest ensembles, yield the best rankings.

Keywords Feature ranking · Multi target regression · Tree based methods · Relief

1 Introduction

Single target regression (STR) is the predictive modeling task of learning a model able to predict the values of a single numeric target variable. STR can be generalized to multi-target

Editors: Takuya Kida, Takeaki Uno, Tetsuji Kuboyama, Akihiro Yamamoto.

Matej Petković
matej.petkovic@ijs.si

Dragi Kocev
dragi.kocev@ijs.si

Sašo Džeroski
saso.dzeroski@ijs.si

[1] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

[2] Jožef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia

regression (MTR), where the goal is to learn a model that predicts several (at least two) target variables simultaneously. While STR is a well established research topic, MTR is only recently attracting interest in the research community (Kocev et al. 2013; Spyromitros-Xioufis et al. 2016; Borchani et al. 2015). MTR is a structured output prediction task with applications in a wide range of real life problems.

Prominent examples for MTR come from ecology and include predicting the abundance of different species sharing the same habitat (Džeroski et al. 2000), predicting forest properties (Kocev et al. 2009), chemometrics to infer concentrations of several analytes from multivariate calibration using multivariate spectral data (Burnham et al. 1999), real-time prediction of multiple gas tank levels of the Linz Donawitz converter gas system (Han et al. 2012), simultaneous estimation of different biophysical parameters from remote sensing images (Tuia et al. 2011), channel estimation through the prediction of several received signals (Sanchez-Fernandez et al. 2004) etc. Many other applications can be found in Sect. 4.2, where the data used in our experiments are described.

A possible way to approach a MTR problem is problem transformation, which transforms one MTR problem into several STR problems and builds one predictive model for each target separately. Another way to approach the problem is by algorithm adaptation, i.e., to adapt STR methods to handle several targets simultaneously. For example, regression trees can be generalized so that the heuristic function used to select splits considers the multiple targets and the leaves make predictions for all targets. By building a single model, we benefit regarding time-complexity, and also exploit the potential relations between the multiple targets: this results in more interpretable and compact models, as demonstrated when predicting communities of different species (Kocev and Džeroski 2013). For an overview of MTR methods, we refer the reader to Borchani et al. (2015).

The algorithm adaptation approaches, e.g., predictive clustering trees (PCTs) (Blockeel 1998), typically use search heuristics for MTR which aggregate those for different STR problems. As described later in Sect. 2.1, the most commonly used aggregation is the simple average. However, if we happen to have some background knowledge about the MTR problem, PCTs allow for incorporating this knowledge into the tree induction process via different weights for different problems.

Another important task in machine learning - which is the main topic of this work - is feature ranking, which is typically seen as a data preprocessing step. By using some scoring function, the scores $importance(x_i)$ of descriptive attributes (features) x_i are estimated and an ordering (ranking) of the features is made, based on their estimated importances. There are two main reasons for doing this. First, we may want to reduce the dimensionality of the input space, so that only the features that contain the most information about the target(s) are kept in the dataset. By doing this, we decrease the amount of memory/time needed to build a predictive model, while the performance of the model is not degraded. Particularly, when the dimensionality of the input data is extremely high, the performance of the model can also increase with the dimensionality reduction, since most of the original features are expected to be noisy. Second, dimensionality reduction typically results in models that are easier to understand, which is useful when a machine learning expert works in collaboration with a domain expert. Predictive models, such as decision trees, are easier to interpret when a small number of relevant features are used to learn them.

There is a plethora of feature ranking methods for the machine learning tasks of single target regression and classification (Stańczyk and Jain 2015). However, in the case of MTR, the task of feature ranking has not been studied to a great extent. To the best of our knowledge, our earlier work (Petković et al. 2017) is among the first in that direction.

In the field of statistics, few feature ranking methods can be found. In contrast to the proposed feature ranking methods that can handle both numeric and nominal features, the main drawback of these methods is that they allow only for numeric features, since they typically assume a (generalized) linear model that describes the dependence of the targets $\boldsymbol{y}$ on the features $\boldsymbol{x}$. One such method is forward selection (Brobbey 2015). It starts with a constant mapping $\boldsymbol{y} = c \in \mathbb{R}$, and repeatedly adds the most significant feature that improves the model. The sooner a feature is included in the model, the greater its importance is.

In this work, we propose two groups of feature ranking scores. The first group is based on ensembles of PCTs, which are a generalization of decision trees able to handle various types of structured output prediction tasks, including MTR. The proposed scores exploit different properties of the ensemble learning mechanism to estimate feature importances. The second group contains the score obtained by generalizing RReliefF (Robnik-šikonja and Kononenko 2003) towards the task of MTR. The key observation on which this score is based is that the distance function on the target space used in RReliefF can be generalized to more complex spaces than $\mathbb{R}$. We refer to these scores as MTR scores.

Another approach to feature ranking for MTR is to perform data transformation in the same way as in binary relevance for multi-label classification (Tsoumakas and Katakis 2007), where a separate classifier is learned for each label. We employ the same strategy: we calculate separate importance scores for each target variable, and then aggregate the feature importance scores into a single score. We thus define an analogue for every MTR score and refer to these scores as $\overline{\text{STR}}$ scores. The $\overline{\text{STR}}$ scores, e.g., those based on RReliefF, have already been shown to have state-of-the-art performance in the case of STR problems. Hence, they can serve as an additional baseline in the evaluation.

The ensemble-based feature ranking methods belong to the class of embedded methods, where the feature importance estimation is embedded in the decision model. In contrast, the RReliefF-based feature ranking methods belong to the group of filter methods (Guyon and Elisseeff 2003). The main difference between the two groups is that filter methods do not need to construct any decision/predictive model in the process of feature importance estimation. A third group of related methods are wrapper methods, which typically solve the problem of feature selection directly, i.e., try to find the most useful set of features (for some predictive model) without constructing a feature ranking.

An initial investigation of the proposed feature ranking methods for MTR based on ensemble methods has been presented by Petković et al. (2017). We extend that work along several major dimensions:

1. We propose an extension of RReliefF for MTR and a $\overline{\text{STR}}$ version of it, giving
    (a) Theoretical and empirical computational complexity analysis;
    (b) Discussion of the theoretical properties of the scores;
    (c) Parametrization of the methods and empirical performance evaluation;
2. We present a theoretical and an empirical computational complexity analysis for the MTR rankings derived by the ensemble-based scores;
3. We analyze the performance of $\overline{\text{STR}}$ rankings based on ensemble scores, giving
    (a) Theoretical and empirical computational complexity analysis;
    (b) Empirical performance evaluation;
    (c) Comparison of the MTR and $\overline{\text{STR}}$ rankings based on ensemble scores in terms of time complexity and performance;

Table 1 Notation in the paper

Symbol	Meaning
x_i, y_j	i-th feature and j-th target
D, T	Numbers of features and targets
$\mathcal{X}_i, \mathcal{Y}_j$	The domains of the i-th feature and the j-th target
$\mathcal{X} \subseteq \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$	Descriptive domain
$\mathcal{Y} \subseteq \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_T$	Target domain
$\boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}$	Descriptive and target part of an example
$\mathscr{D} \subseteq \mathcal{X} \times \mathcal{Y}$	A dataset
$\mathscr{D}_{\mathrm{TRAIN}}, \mathscr{D}_{\mathrm{TEST}} \subseteq \mathscr{D}$	Training and testing part of dataset
$M = \lvert\mathscr{D}_{\mathrm{TRAIN}}\rvert$	Number of examples in the training set

4. We give a comparison of ensemble-based and Relief-based scores ($\overline{\text{STR}}$ and MTR rankings) in terms of time complexity and performance.

The notation used throughout the paper is given in Table 1. The rest of the paper is organized as follows. In Sect. 2.1, we define the ensemble-based scoring functions. Next, in Sect. 2.2, we introduce MTR-Relief and present some novel theoretical results that apply also to the standard RReliefF. We then describe the $\overline{\text{STR}}$ approaches. Furthermore, we perform computational complexity analysis of the methods in Sect. 3. Next, in Sect. 4, the experimental design for our extensive evaluation of the proposed methods on benchmark MTR datasets is presented and the obtained results are discussed in Sect. 5. Finally, we conclude in Sect. 6, where we also provide some directions for further work.

2 Methods

In this section, we describe the proposed methods and the necessary background. We start with the ensemble-based methods (Sect. 2.1), continue with MTR-Relief (Sect. 2.2), and finish with the generic $\overline{\text{STR}}$ approach (Sect. 2.3). The PCT framework, as well as MTR-Relief is implemented in the CLUS system (available at http://source.ijs.si/ktclus/clus-public/) (Table 2).

2.1 Ensemble based methods

2.1.1 Predictive clustering trees and ensembles thereof

The PCT framework views a decision tree as a hierarchy of clusters: the root of a PCT corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The leaves represent the clusters at the lowest level of the hierarchy and each leaf is labeled with its cluster's prototype (prediction). PCTs generalize decision trees and can be used for a variety of learning tasks, including clustering and different types of structured output prediction tasks, e.g., multi-target regression, multi-label classification, hierarchical multi-label classification, time series prediction etc. (Blockeel 1998; Kocev et al. 2013). The generalization is based on appropriately adapting the heuristic for inducing PCTs and the prototype function to the given structured output prediction task.

Table 2 The proposed groups of multi-target regression feature ranking scores

Symbolic score, computed from random forests, bagging and extra trees ensemble:
Uses only the depth(s) of nodes at which a feature appears in the trees in the ensemble
Genie3 score, computed from random forests, bagging and extra trees ensemble:
Uses the values of the variance reduction function at the nodes at which a feature appears in the trees in the ensemble
Random Forest score, computed from random forests and bagging ensemble:
Permutes the values of each feature and measures the resulting reduction in performance of the ensemble on out-of-bag examples
MTR-Relief score, an extension of RReliefF, builds no predictive model:
Examines the similarities in feature and output values for randomly selected instances and their neighbours

The first group consists of the Symbolic, Genie3 and Random Forest scores, and the second group consists of MTR-Relief score. For every score, we propose two versions: MTR and $\overline{\text{STR}}$

PCTs are induced with the standard top-down induction of decision trees algorithm (Breiman et al. 1984) presented in Algorithm 1. It takes as input a set of examples E and outputs a tree. The heuristic h that is used for selecting the best test at a node is the weighted impurity of the subgroups of instances of the partitions (lines 3 and 4), induced by the tests. By minimizing it (line 5 of the Algorithm 2), the algorithm is guided towards small trees with good predictive performance. If there are no candidate tests, a leaf is created and the prototype of the instances belonging to that leaf is computed. The main difference between the algorithm for learning PCTs and other algorithms for learning decision trees is that the former considers the impurity function and the prototype function (that computes predictions in leaves) as parameters that can be instantiated for a given learning task.

```
Algorithm 1 PCT($E$)
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$
2: if $t^* \neq none$ then
3:     for each $E_i \in \mathcal{P}^*$ do
4:         $tree_i = \text{PCT}(E_i)$
5:     return node$(t^*, \bigcup_i \{tree_i\})$
6: else
7:     return leaf(Prototype($E$))
```

```
Algorithm 2 BestTest($E$)
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: for each candidate test $t$ do
3:     $\mathcal{P}$ = partition induced by $t$ on $E$
4:     $h = |E| impu(E) - \sum_{E_i \in \mathcal{P}} |E_i| impu(E_i)$
5:     if $h < h^*$ then
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $(t^*, h^*, \mathcal{P}^*)$
```

In this work, we focus on the task of MTR and define the impurity function using the variances of the target variables. First, we denote the variance of the target y_j over subset of examples $E \subseteq \mathscr{D}_{\text{TRAIN}}$ as $Var_j(E)$. We then compute the weights $w_j = Var_j(\mathscr{D}_{\text{TRAIN}})$ and use them as normalization factors in the definition of the impurity function:

$$impu(E) = \frac{1}{T}\sum_{j=1}^{T}\frac{1}{w_j}Var_j(E).$$

In a leaf L, the prototype function returns a vector with the average values of the target variables calculated for the examples belonging to L.

Next, we use three types of ensembles to calculate feature importance scores (i.e., feature rankings). These ensembles have PCTs as their base predictive models (Kocev et al. 2013). An ensemble is a set of base predictive models constructed with a given algorithm. The prediction for each new example is made by combining the predictions of the models from the ensemble. In regression tasks, this is typically achieved by taking the average of the predictions of the base models.

A necessary condition for an ensemble to be more accurate than any of its individual members, is that the members are accurate and diverse models (Hansen and Salamon 1990). There are several ways to introduce diversity among the PCTs in an ensemble. We describe and make use of three of them.

Random forests (RF) and bagging Instead of being learned from the original dataset $\mathscr{D}_{\text{TRAIN}}$, each tree in the ensemble is built from a different bootstrap replicate $\mathcal{B}$ of the dataset $\mathscr{D}_{\text{TRAIN}}$, called bag. The examples $\mathscr{D}_{\text{TRAIN}} \setminus \mathcal{B}$ are called out-of-bag examples (OOB). Additionally, the line 2 of the *BestTest* procedure (see Algorithm 2) is modified to change the feature set during learning by introducing randomization in the test selection. More precisely, at each node in a decision tree, a random subset of the input attributes is taken, and the best test is selected from the splits defined by these attributes. The number of attributes that are retained is given as a function of the total number of descriptive attributes D, e.g., $\lceil\sqrt{D}\rceil$, $\lceil\log_2(D)\rceil$, etc. In the special case when we keep all of the attributes, we obtain the bagging procedure.

Extra trees ensembles (ET) Here, at each node all attributes are considered (as in bagging), but we do not evaluate all tests that the attributes yield. Rather, we choose randomly only one per attribute. Among these D tests, we choose the best one. From the bias-variance point of view, the rationale behind the Extra-Trees method is that the explicit randomization of the cut-point and attribute combined with ensemble averaging should be able to reduce variance more strongly than the weaker randomization schemes used by other methods (Geurts et al. 2006). Note that ET uses the initial dataset $\mathscr{D}_{\text{TRAIN}}$ for learning the base predictive models and does not make bootstrap replicates.

2.1.2 Ensemble based scores

Feature ranking is obtained by exploiting the ensemble structure of the learning algorithm. Due to its simplicity, we first describe the *Symbolic score*. Then, we discuss the *Genie3* (Huynh-Thu et al. 2010) and the *Random Forest score*[1] (Breiman 2001).

In the following, we denote a tree as $\mathcal{T}$, whereas $\mathcal{N} \in \mathcal{T}$ denotes a node. Trees form a forest $\mathcal{F}$. Its size (the number of trees in the forest) is denoted as $|\mathcal{F}|$. The set of all internal nodes of a tree $\mathcal{T}$ in which the attribute x_i appears as part of a test is denoted as $\mathcal{T}(x_i)$.

[1] To prevent any confusion, Random Forest score will be always in singular form and capitalized, whereas the ensemble method random forests will be in plural form and not capitalized.

Symbolic score Let $d(\mathcal{N})$ denote the *depth* of $\mathcal{N} \in \mathcal{T}$: if $\mathcal{N}$ is the root of $\mathcal{T}$, then $d(\mathcal{N}) = 0$. Otherwise, $d(\mathcal{N}) = 1 + d(\text{parent}(\mathcal{N}))$. In the simplest version of the score, we count how many times a given attribute occurs in the tests in the internal nodes of the trees. Since the attributes that appear closer to the root are intuitively more important than those that appear deeper in the trees, we introduce the parameter $w \in (0, 1]$ and define the importance of the feature x_i as

$$importance_{\text{SYMB}}(x_i) = \frac{1}{|\mathcal{F}|} \sum_{\mathcal{T}\in\mathcal{F}} \sum_{\mathcal{N}\in\mathcal{T}(x_i)} w^{d(\mathcal{N})}. \tag{1}$$

The symbolic score can be computed from all three types of ensembles that we use. Its simplest version corresponds to setting w to 1.

Genie3 score The main motivation for the Genie3 ranking is that splitting the current subset $E \subseteq \mathscr{D}_{\text{TRAIN}}$, according to a test where an important attribute appears, should result in high impurity reduction. The Genie3 importance of the attribute x_i is thus defined as

$$importance_{\text{GENIE3}}(x_i) = \frac{1}{|\mathcal{F}|} \sum_{\mathcal{T}\in\mathcal{F}} \sum_{\mathcal{N}\in\mathcal{T}(x_i)} h^*(\mathcal{N}),$$

where $h^*(\mathcal{N})$ is the value of the variance reduction function described in the *BestTest* procedure. Since h^* is proportional to $|E|$, greater emphasis is again put on the attributes higher in the tree, where $|E|$ is larger. The Genie3 score is applicable to all three ensemble methods that we use.

Random Forest (RF) score This feature ranking method tests how much the noise in a given feature decreases the predictive performance of the trees in the forest. The greater the performance degradation, the more important the feature is. This score uses the internal out-of-bag estimates of the error, therefore it cannot be used with ensembles of ET, where all trees are learned on the whole dataset.

Once a tree $\mathcal{T}$ is grown, the algorithm evaluates the performance of the tree by using the corresponding $\text{OOB}_{\mathcal{T}}$ examples. This results in the predictive error $Err(\text{OOB}_{\mathcal{T}}) \geq 0$, where we assume that lower error value corresponds to better predictions. To assess the importance of the feature x_i for the tree $\mathcal{T}$, we randomly permute its values in the set $\text{OOB}_{\mathcal{T}}$ and obtain the set $\text{OOB}^i_{\mathcal{T}}$. Then, the error $Err(\text{OOB}^i_{\mathcal{T}})$ is computed and the importance of the feature x_i for the tree $\mathcal{T}$ is defined as the relative increase of error after noising. The final Random Forest score of the feature is the average of these values across all trees in the forest:

$$importance_{\text{RF}}(x_i) = \frac{1}{|\mathcal{F}|} \sum_{\mathcal{T}\in\mathcal{F}} \frac{Err(\text{OOB}^i_{\mathcal{T}}) - Err(\text{OOB}_{\mathcal{T}})}{Err(\text{OOB}_{\mathcal{T}})}.$$

2.2 MTR-Relief

In this section, we describe the proposed extension of the RReliefF method towards MTR (dubbed MTR-Relief). We provide technical preliminaries, give the description of the score and discuss some theoretical properties of RReliefF.

2.2.1 MTR-Relief definition

The first member of the Relief family of feature ranking algorithms is Relief, which can handle only binary targets (Kira and Rendell 1992). ReliefF and RReliefF extend the Relief family of algorithms for the tasks of multi-class classification (handling nominal target variables) and regression (handling numeric target variables), respectively (Robnik-šikonja and Kononenko 2003).

All methods of the RELIEF family assign to each feature x_i a weight w_i that is a measure of its importance. The expected value of w_i has a nice probabilistic interpretation in the case when both the target and x_i are nominal (Robnik-šikonja and Kononenko 2003): simplified to some extent, we have the relation $\mathbb{E}[w_i] = p_1 - p_2$ where

$$p_1 = P(\text{ different value of } x_i \mid \text{different target value}) \tag{2}$$

$$p_2 = P(\text{different value of } x_i \mid \text{same target value}) \tag{3}$$

However, in the case of regression, this relation serves only as a motivation. For a fixed x_i, we first define the events `diffFeat/sameFeat` (two instances have different/same value of x_i) and `diffTarget/sameTarget` (two instances have different/same target value), and the associated probabilities $P_{\texttt{event}} = P(\texttt{event})$ and $P_{\texttt{event1, event2}} = P(\texttt{event1} \wedge \texttt{event2})$. After applying the Bayes rule to Eqs. (2) and (3), we obtain

$$\mathbb{E}[w_i] = \frac{P_{\text{diffFeat, diffTarget}}}{P_{\text{diffTarget}}} - \frac{P_{\text{diffFeat}} - P_{\text{diffFeat, diffTarget}}}{1 - P_{\text{diffTarget}}} \tag{4}$$

where the probabilities are modeled as distances

$$P_{\text{diffFeat}} \approx d_i, \;\; P_{\text{diffTarget}} \approx d_{\mathcal{Y}} \text{ and } P_{\text{diffFeat, diffTarget}} \approx d_i d_{\mathcal{Y}}.$$

By adapting the distance $d_{\mathcal{Y}}$, we can extend the Relief algorithm to address different machine learning tasks. In our particular case of MTR, the distances are defined as follows. To be consistent with the RReliefF algorithm, we set

$$d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) = \begin{cases} \mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2] & : \mathcal{X}_i \text{ nominal} \\ \frac{|\boldsymbol{x}_i^1 - \boldsymbol{x}_i^2|}{\max_{\boldsymbol{x}} \boldsymbol{x}_i - \min_{\boldsymbol{x}} \boldsymbol{x}_i} & : \mathcal{X}_i \subseteq \mathbb{R} \end{cases}, \tag{5}$$

where max and min go over the known examples $\boldsymbol{x}$. The analogous approach is taken for the metrics d_j on the sets $\mathcal{Y}_j$, $1 \leq j \leq T$, but here only the numeric part of Eq. (5) applies. The distances on the descriptive domain $\mathcal{X}$ and the target domain $\mathcal{Y}$ between $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, and $\boldsymbol{y}^1 = \boldsymbol{y}(\boldsymbol{x}^1)$ and $\boldsymbol{y}^2 = \boldsymbol{y}(\boldsymbol{x}^2)$, respectively, are defined as

$$d_{\mathcal{X}}(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{1}{D}\sum_{i=1}^{D} d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) \tag{6}$$

$$d_{\mathcal{Y}}(\boldsymbol{y}^1, \boldsymbol{y}^2) = \frac{1}{T}\sum_{j=1}^{T} d_j(\boldsymbol{y}^1, \boldsymbol{y}^2) \tag{7}$$

In Eq. (7), we use the ℓ_1 metric to be consistent with the RReliefF definition from Eq. (6), thus making RReliefF a special instance of MTR-Relief when $T = 1$.

The calculation of the weights $w_i = \mathit{importance}(x_i)$ using MTR-Relief is outlined in Algorithm 3, where we generalize the pseudocode of the RReliefF algorithm (Robnik-šikonja and Kononenko 2003). For each of the m iterations, we randomly select an example $\boldsymbol{r}$ from $\mathcal{D}$ (line 4) and find its k nearest neighbors (line 5), using the distance from Eq. (6). After that, we use the neighbors to update the estimates of probabilities that appear in the definition of the weights in Eq. (4) for all attributes (lines 8–10). The estimates of probabilities are updated with the weighted average of the distances between $\boldsymbol{r}$ and its neighbors. The weight $\delta(\ell)$ for the ℓ-th nearest neighbor is proportional to $\exp(-(\sigma\ell)^2)$, and it is normalized ($\sum_{\ell=1}^{k}\delta(\ell) = 1/mk$) to ensure that $w_i \in [-1, 1]$, when the algorithms finishes. The parameter σ is user defined. Finally, the weights w_i are computed (line 12).

Algorithm 3 MTR-Relief($\mathscr{D}, m, k, \sigma$)

```
1: $\boldsymbol{P}_{\text{diffFeat, diffTarget}}, \boldsymbol{P}_{\text{diffFeat}}$ = lists of length $D$ consisting of zeros
2: $P_{\text{diffTarget}} = 0.0$
3: for $\iota = 1, 2, \ldots, m$ do
4:     $\boldsymbol{r}$ = random example from $\mathscr{D}$
5:     $I_1, I_2, \ldots, I_k = k$ nearest neighbors of $\boldsymbol{r}$
6:     for $\ell = 1, 2, \ldots, k$ do
7:         $P_{\text{diffTarget}} \mathrel{+}= \delta(\ell) d_{\mathcal{Y}}(\boldsymbol{r}, I_\ell)$
8:         for $i = 1, 2, \ldots, D$ do
9:             $\boldsymbol{P}_{\text{diffFeat}}[i] \mathrel{+}= \delta(\ell) d_i(\boldsymbol{r}, I_\ell)$
10:            $\boldsymbol{P}_{\text{diffFeat, diffTarget}}[i] \mathrel{+}= \delta(\ell) d_i(\boldsymbol{r}, I_\ell) d_{\mathcal{Y}}(\boldsymbol{r}, I_\ell)$
11: for $i = 1, 2, \ldots, D$ do
12:     $w_i = \frac{\boldsymbol{P}_{\text{diffFeat, diffTarget}}[i]}{P_{\text{diffTarget}}} - \frac{\boldsymbol{P}_{\text{diffFeat}}[i] - \boldsymbol{P}_{\text{diffFeat, diffTarget}}[i]}{1 - P_{\text{diffTarget}}}$
```

As the number of iterations m increases, the estimates of the probabilities are expected to be more accurate. However, note that we do not need more than M iterations. The value of k should be small enough to capture the local structure in the data and capture the interactions between features (Robnik-šikonja and Kononenko 2003). When k gets bigger, we may want to weight neighbors' contributions with σ.

2.2.2 Theoretical analysis of relief algorithms

Below we analyze some properties of the algorithms from the RELIEF family. The motivation for the definition of Relief is, as mentioned before, an elegant probabilistic interpretation of the weights in the case when both the feature and the target are nominal. In that case, the feature x_i is rewarded in a particular iteration, i.e., the weight update is positive, if (loosely speaking) two instances are more probable to have different values of x_i if they belong to different classes as opposed to the case when they are of the same class.

In the following, we fix the feature x_i. First, we specify the condition when the weight w_i is positive. As a consequence, we will derive the result indicating the most rewarded features by the regression versions of Relief. For simplicity, we first assume that $k = 1$ and $\sigma = 0$. After the proof for this case, we will generalize the results. In the following, $\boldsymbol{r}_\iota$ is the randomly chosen instance in the ι-th iteration, and $\boldsymbol{s}_\iota$ is its nearest neighbor.

Theorem 1 *The feature x_i is rewarded (gets a positive weight update) in the ι-th iteration if, and only if, the target distance $d_{\mathcal{Y}}(\boldsymbol{r}_\iota, \boldsymbol{s}_\iota)$ is not smaller than the average target distance, computed by the algorithm.*

Proof We first take two arbitrary events A and B and simplify the expression $P(A \mid B) - P(A \mid \neg B)$ to

$$\frac{P(AB) - P(A)P(B)}{P(B)(1 - P(B))}. \tag{8}$$

We then set A and B to `diffFeat` and `diffTarget`, respectively (defined in Sect. 2.2.1). The weight w_i, computed by Algorithm 3 is a special instance of Eq. (8):

$$w_i = \frac{\frac{1}{m}\sum_\iota \alpha_\iota \tau_\iota - \left(\frac{1}{m}\sum_\iota \alpha_\iota\right)\left(\frac{1}{m}\sum_\iota \tau_\iota\right)}{\left(\frac{1}{m}\sum_\iota \tau_\iota\right)\left(1 - \frac{1}{m}\sum_\iota \tau_\iota\right)}, \tag{9}$$

where we introduced $\tau_\iota = d_\mathcal{Y}(\boldsymbol{r}_\iota, \boldsymbol{s}_\iota)$ and $\alpha_\iota = d_i(\boldsymbol{r}_\iota, \boldsymbol{s}_\iota)$. If we rearrange the terms in Eq. (9) and define $\tau = \sum_\iota \tau_\iota$, we obtain

$$w_i = \frac{m}{\tau(m-\tau)} \sum_\iota \left(\tau_\iota - \frac{1}{m}\tau\right)\alpha_\iota \tag{10}$$

Since $0 \le \tau_\iota \le 1$ and τ/m is the average computed distance in the target space, we have completed the proof. □

Note that this theorem also applies for the standard (single target) regression task. In the case, when $k > 1$ and $\sigma = 0$, we get additional inner sums $\sum_\ell$ over the neighbors and the terms τ_ι, α_ι and $a_\iota\tau_\iota$ are respectively replaced by $\sum_{\ell=1}^k \tau_{\iota,\ell}$, $\sum_{\ell=1}^k \alpha_{\iota,\ell}$ and $\sum_{\ell=1}^k \alpha_{\iota,\ell}\tau_{\iota,\ell}$. Here, the terms $\alpha_{\iota,\ell}$ and $\tau_{\iota,\ell}$ are defined analogously to α_ι and τ_ι. The additional index ℓ specifies the neighbor to which the distance is computed, and instead of Eq. (10) we now have

$$w_i = \frac{mk}{\tau(mk-\tau)} \sum_\iota \sum_\ell \left(\tau_{\iota,\ell} - \frac{1}{mk}\tau\right)\alpha_{\iota,\ell},$$

where $\tau = \sum_\iota \sum_\ell \tau_{\iota,\ell}$.

If $\sigma \neq 0$, the generalization of the upper formula is obvious.

As a corollary, we derive the result indicating the most rewarded features by the regression versions of Relief. Before proceeding to the result, we define the *centralization* of the random variable X as the random variable $X - \mathbb{E}[X]$. In the formulation of the next theorem, we treat the features as random variables.

Theorem 2 *For a given feature, it is optimal if its centralization is locally positively linearly dependent on the target.*

Proof The features are sorted with respect to the weights w_i, but the order of the features does not change if we define the weights as

$$\omega_i = \sum_\iota \left(\tau_\iota - \frac{1}{m}\tau\right)\alpha_\iota,$$

since we have rescaled the weights from Eq. (10) by multiplying them by a positive constant. If we define the vectors $\boldsymbol{\alpha} = (\alpha_\iota)_\iota$ and $\boldsymbol{\tau} = (\tau_\iota - \tau/m)_\iota$ and fix their Euclidean norms, then, by the Cauchy-Schwarz inequality, it follows that $\omega_i \le \|\boldsymbol{\alpha}\|\,\|\tau\|$. We also know that the maximum value for ω_i is achieved when $\boldsymbol{\alpha} = \beta\boldsymbol{\tau}$ for some $\beta \ge 0$. □

Note that the locality in the theorem originates from the pairs of instances for which the distances d_i and $d_\mathcal{Y}$ are computed. Theorem 1 gives some insight into the Relief procedure on its own, but it mostly serves as an intermediate step for proving Theorem 2 which actually characterizes the features that the algorithm may have some bias toward.

2.3 $\overline{\text{STR}}$ approach

If a structured target domain $\mathcal{Y}$ can be decomposed into primitive parts, we can compute a feature ranking (or build a predictive model) for each part separately, and than average (combine) the results to obtain the final ranking. One of the prominent examples of this technique from predictive the modeling field is the binary relevance approach for multi-label classification (Tsoumakas and Katakis 2007).

To the best of our knowledge, there is no previous use of approaches based on binary relevance for performing MTR feature ranking. We build on this idea and formulate a generic algorithm for MTR feature ranking in Algorithm 4. For the base feature ranking method F, we use RReliefF and the three ensemble-based scores from Sect. 2.1.2.

Algorithm 4 $\overline{\text{STR}}$-ranking(F, $\mathscr{D}$, ...)

```
1: for $j = 1, 2, \dots, T$ do
2:     $\mathscr{D}_j = \{(\boldsymbol{x}, y_j(\boldsymbol{x})) \mid (\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})) \in \mathscr{D}\}$
3:     compute $importance_j(x_i)$ by $F(\mathscr{D}_j, \dots)$, for all $i$
4: for $i = 1, 2, \dots, D$ do
5:     $importance(x_i) = \sum_{j=1}^{T} importance_j(x_i)/T$
```

As stated in the discussion earlier in Sect. 1, simple (non-weighted) averaging is an option when one does not have any background knowledge of the data. However, some of the targets can be given a higher importance on the overall ranking if, e.g., a domain expert so desires.

3 Computational complexity

In this section, we analyze the time and space complexity of the proposed algorithms using the big $\mathcal{O}$ notation. The variables that are not explicitly defined here, are listed in Table 1.

3.1 Ensemble scores

To simplify the conclusions, the analysis is carried out under the assumption that the trees in the forest are balanced (Kocev et al. 2013), i.e., a tree consists of $\mathcal{O}(M)$ nodes, has depth $\mathcal{O}(\log_2 M)$ and $M_{\mathcal{N}} = \mathcal{O}(M/2^{d(\mathcal{N})})$ examples reach a node $\mathcal{N}$ at depth $d(\mathcal{N})$. We also assume that all features are numeric, since i) handling them is more time consuming than handling nominal features and ii) the majority of features in the datasets in our experiments (see Sect. 4.2) is numeric.

For each considered x_i, we first need to sort the examples in the given node ($\mathcal{O}(M_{\mathcal{N}} \log M_{\mathcal{N}})$) and then find the best split ($\mathcal{O}(TM_{\mathcal{N}})$). The complexities for inducing an ensemble of size S are then: $\mathcal{O}(SDM \log M(\log M + T))$ for bagging, $\mathcal{O}(S\sqrt{D}M \log M (\log M + T))$ for RF and $\mathcal{O}(SDTM \log M)$ for ET. The additional cost for updating the importances of all features is $\mathcal{O}(SM)$ for the Symbolic and the Genie3 score, and $\mathcal{O}(SDM \log M)$ for the Random Forest score.

If we assume $T = 1$, we obtain the complexity of growing a single STR ensemble. Since we have to grow T of them, the complexities are now $\mathcal{O}(SDMT \log^2 M)$ for bagging, $\mathcal{O}(S\sqrt{D}TM \log^2 M)$ for RF and $\mathcal{O}(SDTM \log M)$ for ET. Hence, the complexity of growing the ET ensemble stays the same, whereas the other two increase. The additional per-tree cost for updating importances is the same, thus we have T-times more work with updating the importances in the $\overline{\text{STR}}$ case.

We draw the following conclusions. First, the cost of updating importances is negligible in the case of Symbolic and Genie3 scores. Second, growing a STR ensemble is by factor $\min\{T, \log M\}$ more time consuming that growing a MTR ensemble. However, STR trees are usually bigger than their MTR analogues and thus tend to take longer to grow in practice. Third, since it is not much more time consuming to evaluate all splits that a feature yields, as compared to evaluating a single split, random forests and bagging ensembles time complexity is (also in practice) comparable to the time complexity of extra trees ensembles.

3.2 Relief scores

First, we analyze the time complexity for MTR-Relief. Since the space-partitioning data structures, such as kD trees (Friedman et al. 1977) do not perform well when the dimension D is high, we use a brute-force method for finding the nearest neighbors. Finding k nearest neighbors takes $\mathcal{O}(MD)$ steps, assuming that an update of the heap of the current k nearest neighbors ($\mathcal{O}(\log k)$) is negligible in comparison to a distance computation ($\mathcal{O}(D)$).

Before we start updating the probabilities, we compute the distances in the target space between $\boldsymbol{r}$ and its neighbors, which takes $\mathcal{O}(kT)$ time. Updating the probabilities now takes only $\mathcal{O}(kD)$ time.

Computation of the weights takes $\mathcal{O}(D)$ time. Thus, the overall time taken by MTR-Relief is $\mathcal{O}(m[MD + kT + kD] + D) = \mathcal{O}(m[MD + kT])$. Since usually $kT < MD$, this amounts to $\mathcal{O}(mMD)$.

Hence, the time needed for computing $importance_j(x_i)$, for all i and a fixed j, in the $\overline{\text{STR}}$-Relief is $\mathcal{O}(mMD)$, hence the overall time needed here is $\mathcal{O}(TmMD)$, which is higher than the time complexity of MTR-Relief by a factor of T.

However, note that the nearest neighbors do not depend on the target space, hence we can make the naive implementation of $\overline{\text{STR}}$-Relief more efficient, if we update the probabilities in parallel, for each target. This comes with the cost of a larger memory consumption. The additional space that is needed in MTR-Relief is $\mathcal{O}(D)$: the lists $\boldsymbol{P}_{\text{diffFeat, diffTarget}}$, $\boldsymbol{P}_{\text{diffFeat}}$ and the list storing the w_is. In the case of parallel updates for $\overline{\text{STR}}$-Relief, we need $\mathcal{O}(TD)$ memory, which might be a considerable amount if the numbers for features and targets are large. The same time-space consumption relation holds when we want to compute the rankings for different values of parameters: either time or space consumption increases by a factor of the number of combinations we want to study.

4 Experimental design

In this section, we present in detail the experimental design used to evaluate the performance of the proposed methods. We begin by stating the experimental questions. We then briefly summarize the MTR datasets used in this study. Next, we present the evaluation procedure. Finally, we give the specific parameters instantiations for each of the methods.

4.1 Experimental questions

We design the experimental evaluation focusing on several research questions referring to the parametrization of the proposed methods, the relevance of the obtained feature rankings, and their (relative) performance:

1. Which ensemble method is the most appropriate for a given importance score in ensemble-based feature ranking?
2. How do the number of neighbors k, the number of iterations m and the weighting parameter σ influence the rankings obtained with MTR-Relief?
3. How do the MTR rankings compare to their single target ($\overline{\text{STR}}$) counterparts?
4. Can the knowledge of feature importance (scores) lead to better predictive performance of a regressor, i.e., are the obtained feature rankings relevant?
5. Which ranking method yields the best feature rankings overall?

4.2 Datasets

We use 24 MTR benchmark problems. Table 3 presents the basic statistics of the datasets. The number of features ranges from 6 to 576 and the features are mainly numeric. The number of targets ranges from 2 to 16, while the number of examples is in the range between 103 and 60607. The datasets come from different domains: the *ATP* datasets concern the prediction of airline tickets prices. The *collembola*, *forestry*, *soil quality*, and *vegetation condition* datasets

Table 3 Description of the benchmark problems in terms of the number of nominal and numeric descriptive attribute, the number of targets, and the number of examples

Dataset	Nominal	Numeric	Targets	Examples
ATP1d (Spyromitros-Xioufis et al. 2016)	0	411	6	337
ATP7d (Spyromitros-Xioufis et al. 2016)	0	411	6	296
collembola (Kampichler et al. 2000)	8	39	3	393
EDM (Karalič and Bratko 1997)	0	16	2	154
ENB (Tsanas and Xifara 2012)	0	8	2	768
Forestry Kras (Stojanova et al. 2010)	0	160	11	60,607
Forestry LIDAR IRS (Stojanova 2009)	0	29	2	2730
Forestry LIDAR Landsat (Stojanova 2009)	0	150	2	6218
Forestry LIDAR Spot (Stojanova 2009)	0	49	2	2730
jura (Goovaerts 1997)	0	15	3	359
OES10 (Spyromitros-Xioufis et al. 2016)	0	298	16	403
OES97 (Spyromitros-Xioufis et al. 2016)	0	263	16	334
osales (Kaggle 2012)	0	401	12	639
RF1 (Spyromitros-Xioufis et al. 2016)	0	64	8	9125
RF2 (Spyromitros-Xioufis et al. 2016)	0	576	8	9125
SCM1d (Spyromitros-Xioufis et al. 2016)	0	280	16	9803
SCM20d (Spyromitros-Xioufis et al. 2016)	0	61	16	8966
scpf (Kaggle 2013)	0	23	3	1137
sigmeareal (Demšar et al. 2005)	0	6	2	817
sigmeasim (Demšar et al. 2005)	2	9	2	10,368
slump (Yeh 2007)	0	7	3	103
soil quality (Demšar 2006)	0	156	3	1944
vegetation condition (Kocev et al. 2009)	1	39	7	16,967
water quality (Džeroski et al. 2000)	0	16	14	1060

contain environmental data, *EDM* stands for electrical discharge machining, *ENB* and *water quality* originate from studies of water quality. *Jura* contains measurements of heavy metals concentrations. *OES* stands for occupational employment survey. The datasets *osales* (online product sales) and *scpf* (see-click-predict fix) originate from two Kaggle competitions, The *RF1* and *RF2* datasets describe river flows. The *SCM1d* and *SCM20d* datasets were constructed for a competition in the domain of supply chain management. The *sigmeareal* and *sigmeasim* datasets describe GMO (herbicide resistant) crops. Finally, the *slump* dataset concerns prediction of properties of concrete slump.

4.3 Evaluation methodology

We adopted the following evaluation methodology to answer the above research questions and to properly assess the performance of the proposed methods. First, we randomly divide each dataset $\mathscr{D}$ into 2/3 for training ($\mathscr{D}_{\text{TRAIN}}$) and 1/3 for testing ($\mathscr{D}_{\text{TEST}}$). A ranking is computed from the training part only, and evaluated on the testing part. This is repeated 10 times and the performance measures are averaged at the end.

The quality of a ranking is assessed by using the kNN algorithm where instead of the standard Euclidean distance, its weighted version was used. For two input vectors $\boldsymbol{x}^1$ in $\boldsymbol{x}^2$, the distance d between them is defined as

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sqrt{\sum_{i=1}^{D} w_i d_i^2(\boldsymbol{x}_i^1, \boldsymbol{x}_i^2)}, \tag{11}$$

where d_i is defined by Eq. (5). The weights are set to $w_i = \max\{importance(x_i), 0\}$, since they need to be made non-negative to ensure that d is well defined, and also to ignore the attributes that are of lower importance than a randomly generated attribute would be.

The evaluation through a kNN predictive model was chosen because of two main reasons. First, this is a distance based model, hence, it can easily make use of the information contained in the feature importances in the learning phase. Second, kNN is simple: its only parameter is the number of neighbors, which we set to 5. In the prediction stage, the neighbors' contributions to the predicted value are equally weighted, so we do not introduce additional parameters that would influence the performance.

A further argument for using kNN as an evaluation model is as follows. If a feature ranking is meaningful, then when the feature importances are used as weights in the calculation of the distances kNN should produce better predictions as compared to kNN without using these weights (Cunningham and Delany 2007; Wettschereck 1994).

We assess the predictive performance with the average relative root mean squared error $\overline{\text{RRMSE}}$. If we denote the predicted value of the target y_j by $\hat{y}_j(\boldsymbol{x})$ and the variance of the target y_j on $\mathscr{D}_{\text{TRAIN}}$ by $Var_j(\mathscr{D}_{\text{TRAIN}})$, then the RRMSE for this target is defined as

$$\text{RRMSE}(y_j) = \sqrt{\frac{1}{|\mathscr{D}_{\text{TEST}}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathscr{D}_{\text{TEST}}} \frac{(\boldsymbol{y}_j(\boldsymbol{x}) - \hat{y}_j(\boldsymbol{x}))^2}{Var_j(\mathscr{D}_{\text{TRAIN}})}},$$

where $\boldsymbol{y}_j(\boldsymbol{x})$ is the true value of the y_j for the example $\boldsymbol{x}$. From this, we compute $\overline{\text{RRMSE}} = \frac{1}{T}\sum_{j=1}^{T} \text{RRMSE}(y_j)$.

4.4 Statistical analysis of the results

For comparing two algorithms, we use the Wilcoxon's test, and for comparing more than two algorithms, we use the Friedman's test. In both cases, the null hypothesis H_0 is that all considered algorithms have the same performance. If H_0 is rejected by the Friedman's test, we additionally apply Nemenyi's post-hoc test to investigate where the statistically significant differences between *any* two algorithms occur. A detailed description of all tests is available in Demšar (2006).

When performing more than one Wilcoxon's test for a given hypothesis, we control the false discovery rate by the Benjamini–Hochberg procedure (Benjamini and Hochberg 1995): let p_i be the i-th smallest among the obtained p values, and t the number of tests. Let i_0 be the largest i, such that

$$p_i \le \frac{i}{t}\alpha =: \hat{\alpha}_i. \tag{12}$$

Then, we can reject the hypotheses that correspond to p values p_i, for $1 \le i \le i_0$.

The results of the Nemenyi tests are presented through average ranks diagrams. Each diagram shows the average rank of the algorithm over the considered datasets, and the critical distance, i.e., the distance by which the average ranks of two considered algorithms must differ to be considered statistically significantly different. Additionally, the groups of algorithms among which no statistically significant differences occur are connected with a line.

Before proceeding with the statistical analysis, we round the performances to three decimal points. In the analysis, the significance level was set to $\alpha = 0.05$.

4.5 Parameter instantiation for the ensemble scores

The algorithm for inducing an ensemble of PCTs for MTR takes as input the following parameters: the number of base predictive models in the forest (all ensemble types), minimal number of examples in a leaf of a tree (all ensemble types), and the feature subset size (random forests only). In all cases, we grow 100 trees, whose leaves must contain at least two examples each. Additionally, the feature subset size in the case of random forests is set to $\lceil\sqrt{D}\rceil$.

Next, recall that the Symbolic score requires selecting a value for w. In a preliminary study of the MTR versions of the scores, we investigated the influence of several values of w, i.e., $w \in \{0.25, 0.5, 0.75, 1\}$, on the performance of the produced feature rankings. We performed Friedman's test and it turns out that the differences among the rankings are not statistically significant in the case of bagging (p value is 0.379) and ET (p value is 0.441), whereas in the case of RF, they are (p value is 0.0148). In the RF case, we can proceed to Nemenyi's test, whose results are shown in Fig. 1.

The diagram reveals that only the Symbolic score with weight $w = 1.0$ is statistically significantly worse than the rankings with weights $w = 0.5$ and $w = 0.25$. This can be

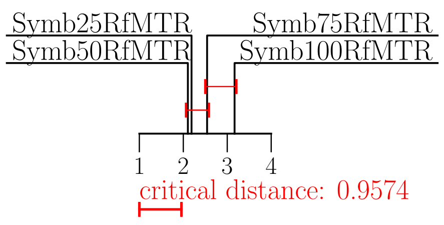


Fig. 1 The average ranks diagram for Nemenyi's test, comparing different values of the parameter w, $w \in \{0.25, 0.5, 0.75, 1.0\}$, for the Symbolic score computed from the RF ensemble

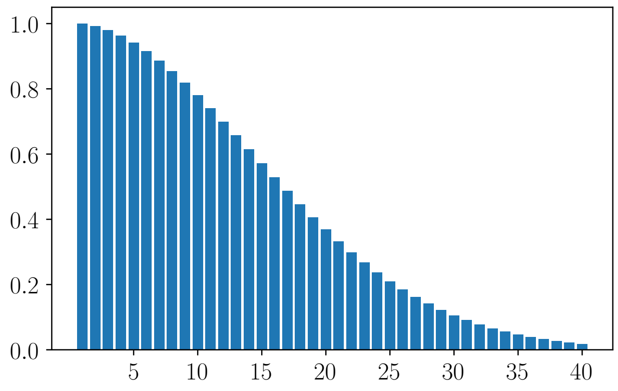


Fig. 2 The influence for the neighbors k in the range $1 \leq k \leq 40$ for the Relief score when $\sigma = 0.05$. The weights are normalized so that the weight of the nearest neighbor equals 1

explained by Eq. (1): the value of 1.0 for the weight is the only one where the depth of the node where an attribute appears is not taken into account when computing the relevance.

Since the average ranks of the ranking methods with $w = 0.25$, $w = 0.5$, $w = 0.75$ and $w = 1.0$ are respectively 2.31, 2.27, 2.56 and 2.85 for bagging, and 2.40, 2.40, 2.35 and 2.88 for ET, the ranking *Symb50* is a reasonable choice for all three ensembles, since it is always ranked at least second. The reason for this is less obvious, but we hypothesize that it could be an artifact of the algorithm for inducing ensembles. Namely, the splits in the nodes of the trees are binary. If we assume that the best test in an internal node $\mathcal{N} \in \mathcal{T}$ partitions the examples approximately in half, then the attribute in $\mathcal{N}$'s test influences one half of the instances that arrive to its parent; hence, the parent should receive twice as large a reward as each of its two children. The value $w = 0.5$ was, for this same reason, selected also in the $\overline{\text{STR}}$ versions of the scores.

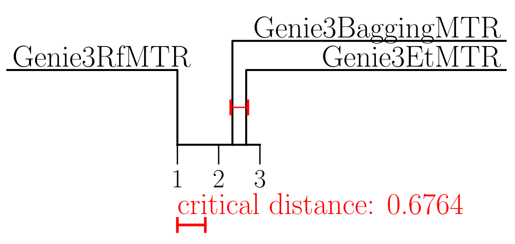
4.6 Parameter instantiation for the relief score

Since the sizes of datasets range over different orders of magnitude, the number of iterations m is specified as a proportion of the size of $\mathcal{D}_{\text{TRAIN}}$. The considered values are $m \in \{1\%, 5\%, 10\%, 25\%, 50\%, 100\%\}$. On the other hand, since the number of neighbors k controls the level of locality, it is better given in absolute values. Our choice is to consider the values $k \in \{1, 5, 10, 15, 20, 25, 30, 40\}$.

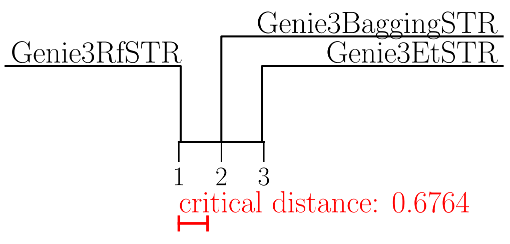
The influence of weighting is investigated with two values of the weighting parameter that result in two extreme weighting schemes: $\sigma = 0$ corresponds to no weighting, i.e., all neighbors have equal influence, and $\sigma = 0.05$, that results in a weighting scheme where the possible 40th nearest neighbor has only $\sim 2\%$ of the influence of the nearest neighbor, see Fig. 2.

5 Results and discussion

In this section, we present and discuss the results of the comprehensive experimental evaluation. First, we discuss the results for the ensemble-based scores. Next, we present the parametrization of MTR-Relief. At the end, we compare the performance of the proposed ensemble-based and Relief-based feature ranking methods.



(a) MTR Genie3



(b) $\overline{\text{STR}}$ Genie3

Fig. 3 The average ranks diagrams from Nemenyi's post-hoc test, applied to the times needed to compute feature rankings with different ensemble methods for the MTR (**a**) and the $\overline{\text{STR}}$ (**b**) version of the Genie3 score

5.1 Ensemble-based feature ranking

5.1.1 Which ensemble method is the most appropriate

For a given fixed score (Genie3, Symb50 or RF) and ranking type (MTR or $\overline{\text{STR}}$), we compare the performance of the different ensemble methods in terms of $\overline{\text{RRMSE}}$ to select the most suitable one for a given score.

For the MTR rankings, the outcome can be summarized as follows. The Friedman test for Genie3 and Symb50 scores did not reject the null hypotheses, with p values of 0.476 and 0.877, respectively. The Wilcoxon test for the RF score rejected the null hypotheses with $p = 0.0170$ and indicated that the bagging method is statistically significantly better than the RF ensemble method.

If methods perform equally well, then the most efficient should be preferred. Hence, we compare the different ensemble methods for the scores Genie3 and Symb50 in terms of time. The differences are significant: the p values are 1.00×10^{-15} and 3.12×10^{-14} respectively. Thus, we can proceed to Nemenyi's post-hoc tests that reveal that, in both cases, the RF ensemble method is significantly faster than the other two methods. This means that it suffices to consider $\sqrt{D}$ features in a tree node to obtain a good split (hence, RF is faster than bagging), but only D test candidates for ET do not suffice (since the trees in ET ensemble must be considerably bigger than the trees in a RF). Since the average ranks diagrams are qualitatively the same, we show only the one for the Genie3 score in Fig. 3a.

For the STR rankings, there are no statistically significant differences regarding the quality of the ranking, i.e., the Friedman tests (for Genie3 and Symb50 scores) and the Wilcoxon test (RF score) do not reject the null hypotheses. Conversely, all three null hypotheses for the time complexity analysis are rejected with p values of 0.0, 0.0 and 1.82×10^{-5} for the scores Genie3, Symb50 and RF, respectively. The differences are now it the case of Genie3 and Symb50 even more profound. In addition to the RF ensemble method being faster than the other two methods, bagging is now also significantly faster than the ET method. The average ranks diagram for the Genie3 score is shown in Fig. 3b.

5.1.2 Comparison of MTR and $\overline{\text{STR}}$ rankings

For each of the 8 combinations of ranking score and ensemble method, we compare the quality of MTR and $\overline{\text{STR}}$ rankings with the Wilcoxon test. The smallest of the eight p values is $p_1 = 0.0298$ in the case of the Genie3 score, coupled with bagging. Since this is not smaller than Benjamini–Hochberg correction $\hat{\alpha}_1 = 0.00625$ from Eq. (12), we cannot reject any hypothesis. On the other hand, every MTR ranking is significantly faster than its $\overline{\text{STR}}$

Table 4 The results of the Wilcoxon tests that compare the performance of standard 5NN to its weighted-distance version

Score-ensemble-version	p_i	$\hat{\alpha}_i$
Genie3-RF-$\overline{\text{STR}}$	$\mathbf{7.59 \times 10^{-5}}$	8.33×10^{-3}
Genie3-RF-MTR	$\mathbf{2.28 \times 10^{-4}}$	1.00×10^{-2}
Symb50-RF-MTR	$\mathbf{3.40 \times 10^{-3}}$	1.25×10^{-2}
Symb50-RF-$\overline{\text{STR}}$	$\mathbf{9.71 \times 10^{-3}}$	1.67×10^{-2}
RF-Bagging-MTR	$\mathbf{2.49 \times 10^{-3}}$	2.50×10^{-2}
RF-RF-$\overline{\text{STR}}$	8.64×10^{-2}	5.00×10^{-2}

The ith row contains the name of the *score-ensemble-version* triplet that provided the feature importances used in the tests of weighted 5NNs against standard 5NN; the p value p_i; and the corrected value $\hat{\alpha}_i$
Statistically significant results are shown in bold

counterpart: all p values now equal 1.8×10^{-5} and all null hypotheses can be rejected. Thus, the MTR rankings should be preferred.

5.1.3 Are the obtained feature rankings relevant?

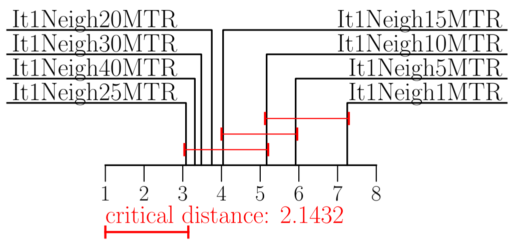
Here, we investigate whether 5NN prediction can benefit from using the additional information from the feature importances. To this end, we compare the performance of 5NN without and with feature importances. We test the relevance of both the MTR and the $\overline{\text{STR}}$ version of each score, coupled with the most suitable ensemble method, as determined in Sect. 5.1.1. Hence, six comparisons are made by using the Wilcoxon's test, where we test the standard 5NN against the weighted 5NN whose feature weights are defined via feature importances as in Eq. (11)

Table 4 gives the results of the statistical evaluation. It shows that the hypotheses are rejected in the case of the Genie3 and the Symb50 scores (in favor of the weighted 5NN). This holds for both the MTR and the $\overline{\text{STR}}$ versions of the scores. The hypothesis was also rejected in the case of the MTR version of the Random Forest score, again in favor of the weighted 5NN. Therefore, we can conclude that using feature ranking is beneficial in these five cases, i.e., the obtained feature importances are relevant and meaningful.

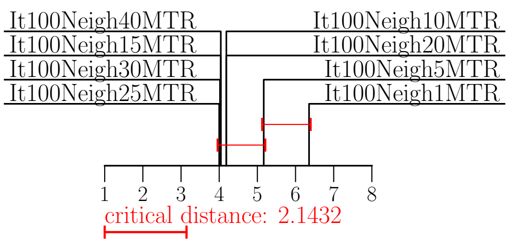
5.2 Relief feature ranking

5.2.1 Influence of the neighbor weighting σ

For both proposed methods (MTR-Relief and $\overline{\text{STR}}$-Relief), we performed 48 Wilcoxon's tests comparing the weighted variant of the algorithm ($\sigma = 0.05$) to its non-weighted variant ($\sigma = 0$). The other two parameters, i.e., m and k were fixed. Here, we will report only the lowest p values for each of the versions. In the case of MTR-Relief, the lowest one was $p_1 = 0.0653$, while in the case of $\overline{\text{STR}}$-Relief, the lowest one was $p_1 = 0.0864$. Thus, the differences are not statistically significant, even before we apply the Benjamini–Hochberg procedure. Since the two quite extreme weighting schemes result in rankings with only minor differences in quality, we conclude that the influence of the weighting parameter is limited. Therefore, we consider only the non-weighted variants of the algorithms in the further analysis.



(a) $m = 1\%$



(b) $m = 100\%$

Fig. 4 The influence of the number of neighbors (k): the average ranks diagrams from Nemenyi's post-hoc test for MTR-Relief with $m = 1\%$ (**a**) and $m = 100\%$ (**b**)

5.2.2 Influence of the number of neighbors *k*

To assess the influence of the number of neighbors, we compare the quality of the rankings obtained by varying the values of k, while the number of iterations (value of m) and ranking type (MTR or $\overline{\text{STR}}$) are fixed. This is done by applying Friedman tests to the described groups of algorithms. The obtained p values corresponding to the 10 groups with $m \leq 50\%$ are all smaller than 10^{-10}. The remaining two p values are $p = 0.00438$ (MTR ranking) and $p = 1.26 \times 10^{-4}$ ($\overline{\text{STR}}$ ranking), hence all null hypotheses can be rejected and we can say that the number of neighbors has considerable influence on the quality of the feature ranking.

We proceed to Nemenyi's post-hoc tests. In Fig. 4, we show average ranks diagrams for the two extreme values of m: $m \in \{1\%, 100\%\}$, for the MTR rankings. The diagrams for $\overline{\text{STR}}$ rankings (and other values of m) are similar. Using more neighbors is obviously better when $m = 1\%$ (Fig. 4a). In this case, only a small portion of $\mathcal{D}_{\text{TRAIN}}$ gets to influence the ranking, thus the ranking overfits to the structure of $\mathcal{D}_{\text{TRAIN}}$ in the neighborhood of the chosen instances. We can mitigate this by examining larger neighborhoods, i.e., choosing more neighbors whose influence is then averaged. In the case of $m = 1\%$, this is the only mechanism that reduces overfitting, hence the strong influence on the performance. In the results with $m = 100\%$ (Fig. 4b), we get to examine the whole $\mathcal{D}_{\text{TRAIN}}$, so the algorithm does not overfit to the extent observed before. However, the diagrams reveal that choosing only one neighbor is still not sufficient, but we do not need more than 10 neighbors to obtain good performance.

5.2.3 Influence of the number of iterations *m*

We perform analogous experiments to the ones described in the previous section, where the number of neighbors k and ranking type are fixed, while the values of the parameter m vary.

The p values from Friedman tests are all smaller than 2.0×10^{-6}, hence in all of the cases, the number of iterations influences the ranking quality. We then perform the follow-up Nemenyi's post-hoc tests and show the resulting average ranks diagrams for the two most extreme cases: $k \in \{1, 40\}$, but now for the $\overline{\text{STR}}$ rankings (Fig. 5).

When $k = 1$ (Fig. 5a), the general more-is-better trend is obvious. This can be again explained by a slightly modified overfitting argument. Since the datasets originate from real-world domains, they must contain some (random) noise. In the case of $k = 1$, its influence can be diminished only by taking into account more and more instances. Approximately the same holds for $k = 40$ (Fig. 5b), but in this case, the average ranks of the scores for $m \geq 10$ are closer. Furthermore, they show that the only values of m that are always part

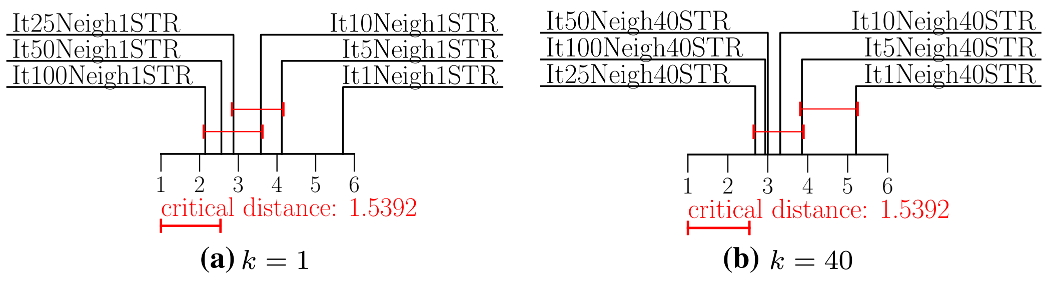


Fig. 5 The influence of the number of iterations (m): the average ranks diagrams from Nemenyi's post-hoc test for $\overline{\text{STR}}$-Relief with $k = 1$ (**a**) and $k = 40$ (**b**)

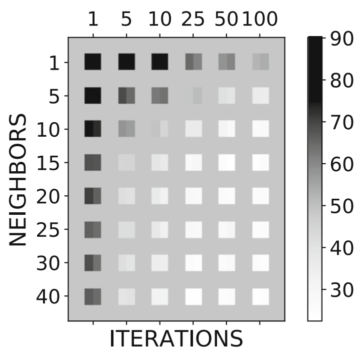


Fig. 6 Heat map of the average ranks of different options for the value pairs (m, k). The background of the map corresponds to the average rank of the non-weighted 5NN, whereas each square with coordinates (m, k) represents the average ranks of the regressor which uses the MTR-Relief (the left part of the square) or $\overline{\text{STR}}$-Relief (the right part of the square) ranking, computed with these parameter values

of the top-performing group of scores, are 25%, 50% and 100%. Similar conclusions can be made by inspecting the other diagrams.

5.2.4 Comparison of MTR and $\overline{\text{STR}}$ rankings

We first compare MTR ans $\overline{\text{STR}}$ rankings based on the Relief score graphically by showing a heat map, where the color corresponds to the average rank of the corresponding regressor. In Fig. 6, there are 48 squares, each of them corresponding to a (m, k) pair of values. Additionally, each square is divided in half. Its left and right part correspond to MTR-Relief and $\overline{\text{STR}}$-Relief, respectively. The background of the heat map is of the color that corresponds to the performance of the non-weighted 5NN regressor. An inspection of the graph reveals that no differences are to be expected between the MTR and $\overline{\text{STR}}$ versions, since most of the time a difference is not visible between the left and right part of each square. The smallest among the 48 p values from the Wilcoxon tests, where we compare MTR and $\overline{\text{STR}}$ rankings is $p = 0.0163$ and is obtained when $m = 10\%$ and $k = 25$. However, if we apply the Benjamini–Hochberg correction, there are no statistically significant differences.

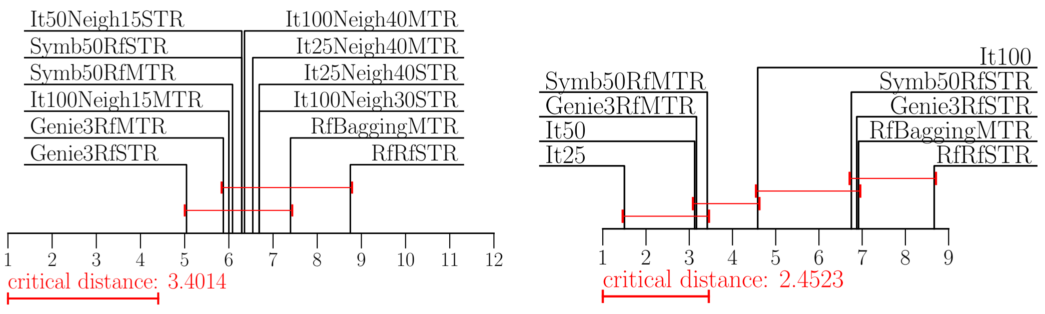
5.2.5 Are the obtained feature rankings relevant?

In Fig. 6, we can additionally see that some parameter settings clearly lead to irrelevant rankings, e.g., $m = 1\%$ and $k = 1$, for which the average ranks for the MTR and $\overline{\text{STR}}$ weighted 5NN predictive performance are 89.4 and 90.1, whereas the baseline 5NN regressor

Table 5 The results of the Wilcoxon's tests that compare the performance of standard 5NN to its weighted-distance version

m-k-version	p_i	$\hat{\alpha}_i$
It100Neigh15MTR	**0.00664**	0.00833
It50Neigh15STR	0.0119	0.01
It25Neigh40STR	0.0975	0.0125
It25Neigh40MTR	0.100	0.0167
It100Neigh30STR	0.145	0.025
It100Neigh40MTR	0.153	0.05

The ith row contains the values of the parameters of the Relief ranking that provided the feature importances and was tested against standard 5NN; the p value p_i; and the corrected value $\hat{\alpha}_i$
Statistically significant result is shown in bold



(a) $\overline{\text{RRMSE}}$ **(b)** time

Fig. 7 Comparison of the 12 ranking scores: the average ranks diagrams from Nemenyi's post-hoc test for $\overline{\text{RRMSE}}$ (**a**) and time (**b**). Due to the implementation of Relief, the running times for Relief depend only on the number of iterations

has the average rank of 48.5. Hence, we take a similar approach to that in Sect. 5.1.3, where we tested for the relevance of ensemble scores. We determine the best three parameter settings for each of the rankings (MTR and $\overline{\text{STR}}$) and then apply the Wilcoxon tests for the weighted 5NN against the non-weighted 5NN baseline. The results are given in Table 5.

In contrast to the ensemble scores relevance analysis, now only one ranking performs statistically significantly better than the baseline. This is the MTR version of the score when $m = 100\%$ and $k = 15$.

5.3 Which ranking method yields the best feature rankings?

In this section, we compare the six candidates from the Ensemble-based rankings and the six candidates from the Relief-based rankings. The comparison is made in terms of $\overline{\text{RRMSE}}$ and time, by applying Friedman's test. The corresponding p values are $p_{\overline{\text{RRMSE}}} = 0.117$ and $p_{\text{time}} = 0.0$. The follow-up Nemenyi's test produced the average ranks diagrams shown in Fig. 7. Note that the critical distance in the $\overline{\text{RRMSE}}$-based diagram should be ignored ($p_{\overline{\text{RRMSE}}} > 0.05$) and we focus only on the average ranks. It seems that the quality of different scores is quite similar: most of them are grouped together with the exception of the $\overline{\text{STR}}$ rankings of Genie3 and RF score as two outliers.

On the other hand, when we compare how much time is needed to compute the rankings (Fig. 7b), we can identify more groups. For example, the fastest group is formed by the Relief scores with $m \in \{25\%, 50\%\}$ and the MTR rankings of the Genie3 and Symbolic scores. The

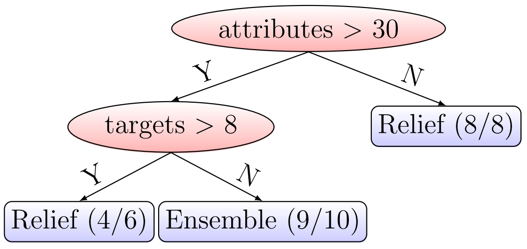


Fig. 8 The PCT constructed from the meta dataset, where Y and N denote Yes and No branches. In every leaf node, the majority target statistics are shown, e.g., in the leaf *Relief (4/6)*, there are 6 examples. On 4 of them, Relief performs better

Table 6 Cross-validated performance of the meta decision tree, where y denotes true values and $\hat{y}$ denotes predicted values

y \ $\hat{y}$	Ensemble	Relief
Ensemble	9	2
Relief	3	10

slowest ones are the STR rankings from the ensemble-based scores, together with the MTR ranking of the RF score where bagging is used.

Since our efficient implementation allows for computing all Relief scores at once, the induction time for these scores depends only on the number of iterations, hence the shortened descriptions of the MTR and $\overline{\text{STR}}$ rankings (only the number of iterations is shown in Fig. 7b).

We next perform a meta-learning analysis of the obtained results. If we want to get a recommendation on using an appropriate feature algorithm for a given new dataset, the average ranks in Fig. 7a do not provide a unique recommendation, since there are too many candidates. Also, we want to understand when the ensemble-based rankings perform better than the Relief-based rankings, considering some basic properties of the datasets. To do so, we describe each dataset as a $(\boldsymbol{x}, y)$ pair, where the descriptive part $\boldsymbol{x}$ consists of the basic dataset statistics shown in Table 3: number of examples, features and targets. The target part for a particular dataset was $y =$ Ensemble if an ensemble method performed better on this dataset, and $y =$ Relief if a Relief method was better. From this meta dataset, we build a single PCT, which is shown in Fig. 8. It seems that the number of examples does not influence the better performing score. We can conclude that the Ensemble group of rankings performs better for datasets that have a higher descriptive dimension $D > 30$ and a lower number of targets ($T \leq 8$).

The accuracy of the tree is 79.2% and was estimated via tenfold cross validation. The corresponding confusion matrix is shown in Table 6.

5.4 Absolute performance of rankings

So far, the statistical analysis shown operated only with the ranks of the algorithms on different datasets and we have not shown yet any actual performance figures in terms of $\overline{\text{RRMSE}}$. The purpose of analysis of these results is twofold: we can explore (i) whether the previously discovered statistically significant differences are also practically relevant, and (ii) which datasets represent harder and which datasets represent easier feature ranking (and predictive modelling) problems.

Inspecting Fig. 7a, we can see that the best average rank among the ensemble-based scores belongs to the $\overline{\text{STR}}$-Genie3 rankings, computed from the random forest ensemble. The best ranked score from Relief group is MTR-Relief ranking, computed with $m = 100\%$ iterations

Table 7 The performance (in terms of $\overline{\text{RRMSE}}$) of the baseline 5NN model and the weighted models that correspond to the best rankings from the ensemble and the Relief group of scores

	Baseline 5NN	Best ensemble score Genie3-RF-STR	Best relief score It100Neigh15MTR
ATP1d	0.467	0.45	0.452
ATP7d	0.664	0.653	0.678
collembola	1.006	0.992	1.006
EDM	0.784	0.776	0.759
ENB	0.306	0.155	0.148
Forestry Kras	0.588	0.59	0.587
Forestry LIDAR IRS	0.425	0.347	0.404
Forestry LIDAR Landsat	0.535	0.49	0.528
Forestry LIDAR Spot	0.414	0.367	0.402
jura	0.836	0.749	0.762
OES10	0.6	0.588	0.598
OES97	0.535	0.532	0.534
osales	0.946	0.868	0.932
RF1	0.239	0.228	0.251
RF2	0.509	0.313	0.529
SCM1d	0.306	0.302	0.301
SCM20d	0.318	0.32	0.317
scpf	1.059	0.989	0.934
sigmeareal	0.977	0.906	0.912
sigmeasim	0.141	0.116	0.052
slump	0.795	0.766	0.756
soil quality	0.732	0.731	0.729
vegetation condition	0.662	0.665	0.68
water quality	0.985	0.983	0.98

and $k = 15$ neighbors. We present performance figures for these two rankings, together with the baseline non-weighted 5NN in Table 7.

We can see that sometimes, the baseline 5NN performs considerably worse than at least one of the other two predictive models. For example, this is the case for the ENB, RF2 and sigmeasim datasets. Interestingly, ENB and sigmeasim have only 8 and 11 features respectively. For the considered methods, these datasets present not that hard feature ranking problems. On the other hand, the statistically significant differences are sometimes not practically relevant, e.g., the relative decrease of the $\overline{\text{RRMSE}}$ after computing the ranking on the SCM datasets is quite small. These datasets present harder feature ranking problems. However, as evident from the statistical tests, the baseline is quite consistently the worst performing of the three predictive models.

6 Conclusion

In this work, we focus on the task of feature ranking for multi-target regression. More specifically, we propose methods for feature ranking that exploit the potential relatedness among multiple targets to produce a single feature ranking valid for all of the targets. Namely, we propose methods for feature ranking that are based on ensemble learning and on the RELIEF family of feature ranking algorithms. In the former group of methods, we consider three ensemble learning methods (bagging, random forests and extremely randomized trees) coupled with three scores (Genie3, Random Forest and Symbolic). The ensemble methods use predictive clustering trees for MTR as base predictive models. The latter group considers an extension of RReliefF towards the task of MTR (MTR-Relief). We have also proposed to use the 'single' target variants of the proposed methods in the context of MTR by averaging the importance scores obtained for each of the targets.

We analyze the proposed methods along several dimensions. First, we perform a theoretical computational complexity analysis of all of the methods. Next, we complement the computational complexity analysis with runtime comparisons. Furthermore, we parametrize the proposed methods. Finally, we benchmark the performance of the proposed methods on 24 datasets by using the feature importances as weights in 5-nearest neighbors (5NN) prediction.

The comprehensive experimental evaluation reveals the following. First, five out of the six ensemble-based ranking methods yield relevant feature rankings when they are computed using the most appropriate tree ensemble, the only exception being the $\overline{\text{STR}}$ ranking with Random Forest score. The evaluation identified two best ranking methods (in terms of performance and efficiency): Genie3 and Symbolic scores coupled with random forests.

Second, with the Relief rankings, only a single setting was found yielding statistically significant differences in performance: the MTR version of Relief with $m = 100\%$ and $k = 15$. When we compared the top 12 ranking algorithms, there were no statistically significant differences among them. However, further analysis revealed that this is due to the properties of datasets, since we can rather accurately determine which of the two groups (ensemble-based or Relief-based) of algorithms will perform better on a given dataset.

Third, the analysis shows that the MTR versions of ensemble-based rankings are statistically significantly faster than their $\overline{\text{STR}}$ counterparts, but not worse in terms of quality. In the case of Relief, there are no differences. All in all, we would suggest using the Genie3 and the Symbolic scoring coupled with random forest ensemble method, since they both yield relevant rankings and are fast enough.

There are several directions for the future work. First, we plan to consider a different approach to defining the distance measure in MTR-Relief (or even in RReliefF). We could use a distance based on the probability densities $f(x_i)$ and $f(y_j)$, since this is the usual continuous analogue of the probabilities $P(x_i = x)$ in the discrete case. Next, we will extend the proposed methods to other structured output prediction tasks, such as multi-label classification, and hierarchical multi-label classification. Furthermore, we will investigate the influence of the ensemble size on the produced feature rankings (thus further reducing the computational cost). Finally, we will develop similar approaches for learning feature rankings for MTR tasks on data streams and for semi/un-supervised MTR tasks.

Acknowledgements We acknowledge the financial support of the Slovenian Research Agency via the Grants P2-0103 and a young researcher Grant to MP, as well as the European Commission, through the Grants MAESTRA (Learning from Massive, Incompletely annotated, and Structured Data) and HBP (The Human Brain Project), SGA1 and SGA2. SD also acknowledges support by Slovenian Research Agency (via Grants J4-7362, L2-7509, and N2-0056), the European Commission (Project LANDMARK) and ARVALIS (Project BIODIV). The computational experiments presented here were executed on a computing infrastructure from the Slovenian Grid (SLING) initiative.

References

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, *57*(1), 289–300.

Blockeel, H. (1998). *Top-down induction of first order logical decision trees*. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium.

Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A survey on multi-output regression. *Data Mining and Knowledge Discovery*, *5*(5), 216–233.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. J. (1984). *Classification and regression trees*. Boca Raton: Chapman & Hall/CRC.

Brobbey, A. (2015). Variable selection in multivariate multiple regression. Master's thesis, Memorial University of Newfoundland, St John's, NL, Canada.

Burnham, A. J., MacGregor, J. F., & Viveros, R. (1999). Latent variable multivariate regression modeling. *Chemometrics and Intelligent Laboratory Systems*, *48*(2), 167–180.

Cunningham, P., & Delany, S. J. (2007). k-Nearest Neighbour Classifiers. Technical report, University College Dublin, Dublin, Ireland.

Demšar, D., Debeljak, M., Džeroski, S., & Lavigne, C. (2005). Modelling pollen dispersal of genetically modified oilseedrape within the field. In *Proceedings of annual meeting of the Ecological Society of America*.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Džeroski, S., Demšar, D., & Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, *13*, 7–17.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, *3*(3), 209–226.

Geurts, P., Erns, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *36*(1), 3–42.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford: Oxford University Press.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Han, Z., Liu, Y., Zhao, J., & Wang, W. (2012). Real time prediction for converter gas tank levels based on multi-output least square support vector regressor. *Control Engineering Practice*, *20*(12), 1400–1409.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*, 993–1001.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., & Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS One*, *5*(9), 1–10.

Kaggle. (2012). Kaggle: Online product sales. https://www.kaggle.com/c/online-sales. Accessed June 12, 2018.

Kaggle. (2013). Kaggle: See click predict fix. https://www.kaggle.com/c/see-click-predict-fix. Accessed June 12, 2018.

Kampichler, C., Džeroski, S., & Wieland, R. (2000). Application of machine learning techniques to the analysis of soil ecological data bases: Relationships between habitat features and Collembolan community characteristics. *Soil Biology and Biochemistry*, *32*(2), 197–209.

Karalič, A., & Bratko, I. (1997). First order regression. *Machine Learning*, *26*(2–3), 147–176.

Kira, K., & Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the tenth national conference on artificial intelligence* (pp. 129–134). AAAI Press, San Jose, California.

Kocev, D., & Džeroski, S. (2013). Habitat modeling with single- and multi-target trees and ensembles. *Ecological Informatics*, *18*, 79–92.

Kocev, D., Džeroski, S., White, M., Newell, G., & Griffioen, P. (2009). Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modelling*, *220*(8), 1159–1168.

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, *46*(3), 817–833.

Petković, M., Džeroski, S., & Kocev, D. (2017). Feature ranking for multi-target regression with tree ensemble methods. In Yamamoto, A., Kida, T., Uno, T., & Kuboyama, T. (Eds.), *Discovery science* (pp. 171–185). Berlin: Springer.

Robnik-šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning Journal*, *55*, 23–69.

Sanchez-Fernandez, M., de-Prado-Cumplido, M., Arenas-Garcia, J., & Perez-Cruz, F. (2004). Svm multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE Transactions on Signal Processing*, *52*(8), 2298–2307.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., & Vlahavas, I. (2016). Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, *104*(1), 55–98.

Stańczyk, U., & Jain, L. C. (Eds.). (2015). *Feature selection for data and pattern recognition. Studies in computational intelligence*. Berlin: Springer.

Stojanova, D. (2009). Estimating forest properties from remotely sensed data by using machine learning. Master's thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

Stojanova, D., Panov, P., Gjorgjioski, V., Kobler, A., & Džeroski, S. (2010). Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics*, *5*(4), 256–266.

Tsanas, A., & Xifara, A. (2012). Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, *49*, 560–567.

Tsoumakas, G., & Katakis, I. (2007). Multi label classification: An overview. *International Journal of Data Warehouse and Mining*, *3*(3), 1–13.

Tuia, D., Verrelst, J., Alonso, L., Perez-Cruz, F., & Camps-Valls, G. (2011). Multioutput support vector regression for remote sensing biophysical parameter estimation. *IEEE Geoscience and Remote Sensing Letters*, *8*(4), 804–808.

Wettschereck, D. (1994). *A study of distance based algorithms*. Ph.D. thesis, Oregon State University, Corvallis, OR.

Yeh, I.-C. (2007). Modeling slump flow of concrete using second-order regressions and artificial neural networks. *Cement and Concrete Composites*, *29*, 474–480.

Chapter 5

Feature Ranking for Multi-Label Classification and HMLC

In this Chapter, we present in detail the second set of contributions of the thesis: those concerning the supervised learning setting and the structured output prediction tasks of multi-label classification (MLC) and hierarchical multi-label classification (HMLC). Recall from the introductory sections (Section 2.2.3) that the task of MLC is to simultaneously predict (possibly multiple) labels that are relevant for a given example. The task of HMLC is similar, but here, the set of labels is also partially ordered, i.e., hierarchically organised. Also recall that we presented our feature ranking methods for MLC and HMLC in Sections 3.1.1 and 3.2.1. Our contributions in this context include:

1. An extension of ensemble-based feature ranking scores from the context of single target prediction to the context of MLC and HMLC.
2. An extension of distance-based feature ranking scores from the context of single target prediction to the context of MLC and HMLC.
3. An extensive experimental evaluation of the newly proposed feature importance scores for MLC in HMLC, respectively, on appropriate collections of benchmark datasets, assessing the performance of the scores individually and in cross-comparison.

Chronologically, we first proposed the extension of distance-based feature ranking methods from the single target prediction to the MLC setting in a paper presented at the DS-2018 conference (Petković, Kocev, et al., 2018). As outlined in Chapter 3, the crucial part of the adaptation is the definition of an appropriate distance measure between the sets of labels. The proposed distance measures are then included in the extension of RReliefF. An extensive evaluation on 24 benchmark datasets revealed that using any of these distance measures always results in rankings of higher quality than the rankings computed by the existing MLC-Relief competitor (Reyes et al., 2015). Moreover, all the proposed feature rankings are relevant – they are significantly better than the uniform rankings that assign the same importance to all the features. The paper is included in this Chapter.

Next, we extended the three ensemble-based scores from single-target prediction to MLC and extensively evaluated them (Petković, Džeroski, et al., 2020c). The main findings are that the proposed methods yield relevant feature rankings and that the best performing method is the one that uses random forests for learning the ensemble and Genie3 for calculating the feature importance scores. Moreover, at the same time, the methods outperform the current state-of-the art, and are computationally very efficient since growing only 10 trees suffices to reach stable quality of the feature rankings. The paper is included in this Chapter.

Our initial experiments with HMLC feature rankings covered Symbolic ranking computed from bagging ensembles. The findings were presented at the conference ETAI 2018 (Petković, Džeroski, et al., 2018). In the paper, not included in this Chapter, we show that the proposed feature rankings are relevant and also time-efficient. We extended this work and included all the other aforementioned ensemble-based and distance-based feature ranking methods for HMLC.

The findings are presented in a journal paper (Petković, Džeroski, et al., 2020a) and are similar to those from MLC experiments. Namely, the ensemble-based feature rankings outperform the Relief ones, and can be computed very efficiently (from an ensemble of 10 trees). The best two performing scores are Genie3 and Symbolic (computed from random forest ensembles).

The work presented in this Chapter pertains to the following hypotheses (as defined in the introduction):

H1: It is possible to extend ensemble- and distance-based feature ranking approaches to the unsupervised feature ranking task, to the tasks of supervised SOP (i.e., MTR, MLC, and HMLC), and to their semi-supervised versions.

H4: The proposed ensemble- and distance-based approaches yield relevant and state-of-the-art feature rankings for MLC.

H5: The proposed ensemble- and distance-based approaches yield relevant and state-of-the-art feature rankings for HMLC.

H6: For unsupervised, MTR, MLC and HMLC problems, the ensemble-based feature ranking approaches on average outperform the distance-based approaches to feature ranking when the number of features is extremely high.

These hypotheses are confirmed with the design and implementation of the MLC and HMLC feature ranking approaches presented in this Chapter and the experimental studies comparing their performance. Hypotheses H4 and H5 are completely confirmed by the results presented in this Chapter. For hypotheses H1 and H6, the parts pertaining to MLC and HMLC are confirmed in this Chapter, while the remaining parts of these hypotheses are addressed in the previous and the next chapters.

The papers included in this Chapter are:

- Petković M., Kocev D., Džeroski S. (2018) Feature Ranking with Relief for Multi-label Classification: Does Distance Matter?. In: Proceedings of the 21st International Conference on Discovery Science. LNCS, 11198: 51–65, Springer, Cham
- Petković, M., Džeroski, S. and Kocev, D. Multi-label feature ranking with ensemble methods. *Machine Learning.* Under review.
- Petković, M., Džeroski, S. and Kocev, D. Feature Ranking for Hierarchical Multi-Label Classification with Tree Ensemble Methods. *Acta Polytechnica Hungarica.* In press.

The contribution of Matej Petković to these papers is as follows. MP contributed to the design of the ensemble-based and distance-based feature ranking methods for MLC and HMLC and implemented these methods in computer code. He also participated in designing the experiments, carried out the experiments, and processed their results. He drafted the papers and revised them following the feedback from the co-authors and reviewers.

Feature Ranking with Relief for Multi-label classification: Does distance matter?[*]

Matej Petković[1,2], Dragi Kocev[1,2], and Sašo Džeroski[1,2]

[1] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
[2] Jožef Stefan Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia
{matej.petkovic,dragi.kocev,saso.dzeroski}@ijs.si

Abstract. In this work, we address the task of feature ranking for multi-label classification (MLC). The task of MLC is to predict which labels from a maximal predefined label set are relevant for a given example. We focus on the Relief family of feature ranking algorithms and empirically show that the definition of the distances in the target space used within Relief should depend on the evaluation measure used to assess the performance of MLC algorithms. By considering different such measures, we improve over the currently available MLC Relief algorithm. We extensively evaluate the resulting MLC ranking approaches on 24 benchmark MLC datasets, using different evaluation measures of MLC performance. The results additionally identify the mechanisms of influence of the parameters of Relief on the quality of the rankings.

Keywords: feature ranking · multi-label classification · Relief

1 Introduction

Classification is a task in predictive modelling, where the goal is to learn a model that takes as the input a vector $\boldsymbol{x}$ of descriptive variables (features) x_i, and predicts the class value y that a given example belongs to. If y can take two different values, the task at hand is referred to as binary classification. Otherwise (y can take more than two values), the task at hand is multi-class classification. In both cases, every example is assigned precisely one value. For example, one can predict whether a person has survived a shipwreck where $y \in \{\texttt{yes}, \texttt{no}\}$ (binary), or what is the blood type of a person where $y \in \{\texttt{A}, \texttt{B}, \texttt{AB}, \texttt{0}\}$ (multi-class). In both cases, class values are mutually exclusive.

A related task is multi-label classification (MLC). As opposed to the standard classification, a MLC predictive model predicts which labels from a predefined set $\mathscr{L}$ are *relevant* for a given example. For example, one can predict which of the genres from the set $\mathscr{L} = \{\texttt{romance}, \texttt{drama}, \texttt{comedy}\}$ are relevant for a given film. Clearly, a film can be `drama` and `comedy` at the same time.

[*] We acknowledge the financial supported of the Slovenian Research Agency via the grants P2-0103, J4-7362, L2-7509, N2-0056, and a young researcher grant to MP, the European Commission, through the grants HBP (The Human Brain Project), SGA2, and LANDMARK, and ARVALIS (project BIODIV).

There are two main approaches to MLC: problem transformation and algorithm adaptation. From the problem transformation group of methods most widely known are binary relevance and label power set. Binary relevance is a simple method that converts a MLC task to several binary classification tasks with $y \in \{\texttt{yes}, \texttt{no}\}$ where we predict the relevance of each label separately. This approach is often criticized for it cannot make use of the interactions among the labels. In the label power set approach [24], the task of predicting a subset of $\mathscr{L}$ is converted to the task of predicting an element of the power set $2^{\mathscr{L}}$, and thus converting a MLC task to multi-class classification task. However, the number of classes can be as high as $2^{|\mathscr{L}|}$, which results in a very sparse dataset.

The second group of methods are method transformation techniques where an existing method is adapted to a new problem. A prominent member of this group are predictive clustering trees which generalize decision trees, so that they can handle MLC [15] and other structure output prediction tasks [12].

Another important task in machine learning is feature ranking, where the goal is to asses the importance of every descriptive attribute (feature) by using some scoring function. The output of a feature ranking algorithm is a list of features that is sorted with respect to the scores.

Feature ranking is typically considered a part of data preprocessing, since it can be used to reduce the dimensionality of the input space, so that only the features that contain the most information about labels (or target(s) in general) are kept in the dataset. By doing this, we decrease the computational cost of building a predictive model, while the performance of the model is not degraded. Another reason to compute a feature ranking is that dimensionality reduction typically results in models that are easier to understand, which is useful when a machine learning expert works in collaboration with a domain expert. Predictive models, such as decision trees, are easier to interpret when a small number of the most relevant features are used to learn them.

There is a plethora of feature ranking methods for the task of classification [22]. A possible approach to MLC feature ranking is to adapt the binary relevance approach from predictive modelling, where at the first stage, feature importances are computed for every label $\ell \in \mathscr{L}$ separately as in the classification case. After that, the feature importances are averaged over the different labels and a single ranking is returned. In this work, we focus on the Relief family of feature ranking algorithms, which are distance based approaches and thus widely applicable. They are part of the *filter methods* which compute the ranking without any additional predictive model [9]. The filters are typically fast, i.e., linear in the number of features, but myopic at the same time, i.e., cannot capture the feature interaction. Relief family of the feature ranking algorithms, however, overcomes this, and can successfully discover, e.g., XOR-relation [14].

The rest of the paper is organized as follows. In Sec. 2, the overview of related work is given. In Sec. 3, the proposed feature ranking algorithms are described and analyzed. In Sec. 4, the detailed description of the experimental is given. In Sec. 5, the results of the experiments are presented. In Sec. 6, conclusions and direction for further work are given.

2 Related Work

We start the overview of the related work with the extensions of the Relief family to MLC setting that are presented in [20]. There, the binary relevance and label power set approach were applied to the feature ranking scenario. More precisely, in the case of binary relevance approach, feature ranking was computed for every label $\ell \in \mathscr{L}$ separately. This was done by using the Relief algorithm for the standard binary classification [11]. After that, the feature importances were averaged to a single score. In the case of the label power set approach the multi-class extension of the Relief (ReliefF) was used [14].

As mentioned before, these two approaches have some drawbacks. Binary relevance approach does not take the label interactions into account and can be expensive to run when the number of labels is high: we have to solve $|\mathscr{L}|$ feature ranking problems which results in high time or space complexity. High space complexity is also a drawback of label power set approach if the number of different relevant label subsets is high. In that case, the data may also become too sparse for the ranking to be relevant.

Both procedures were evaluated on a rather small subset of ten datasets presented in this study (see Sec. 4.2), in a manner similar to our evaluation procedure, which uses k nearest neighbours classifier. No statistical tests were done and the feature rankings were not compared to any baseline.

Another data transformation approach was presented in [13] where the MLC problem is transformed into $|\mathscr{L}|(|\mathscr{L}| - 1)/2$ binary classification problems - one for each of the label pairs (ℓ_1, ℓ_2) where $\ell_1 \neq \ell_2$. For each binary problem, only the examples for which either ℓ_1 or ℓ_2 is relevant (but not both) are retained in the corresponding dataset. The exclusion of the examples for which both labels are relevant is necessary to avoid ill-defined terms in the equations for importance update. The authors motivate this by claiming that the number of the examples for which both labels are relevant, is small in comparison to the number of examples for which precisely one of the two labels is relevant. However, this may not be the case in some data sets, as observed in [18]. The main drawback of this approach is the computational complexity, since the number of feature ranking problems to solve grows quadratically with the number of labels.

A member of the Relief family ReliefF-ML [18] does solve the multi-label ranking problem directly, yet its space complexity is still considerable. The algorithm RReliefF-ML [18] overcomes this issue since it is an extension of the RReliefF version that is suitable for regression tasks [14]. In contrast to the extensions, RReliefF(-ML) computes only one group of the nearest neighbors per example which results in significantly smaller space complexity.

The method was empirically shown to yield relevant feature rankings [18] since it statistically significantly outperformed the baseline. For showing statistical significance, Friedman test was used. However, we need to point out that the very basic assumption of the independence of *data samples* (datasets in this case) was not met, since 10 out of 34 are basically different versions of the same data (Corel16k datasets). In our experiments, we also show that the seemingly ad-hoc choice of the target distance may not lead to the best rankings if we want to optimize for a particular evaluation measure.

Regarding pure predictive modelling setting, the authors in [4] show that in general, different evaluation measures result in different optimal classifiers. However, the authors also show that, e.g., *Hamming Loss* and *Subset Accuracy* have the same optimal classifier under some rather strict conditions.

3 MLC-Relief

Relief family of feature ranking algorithms calculates the feature importance scores by considering differences in the feature values between pairs of examples (an example and its nearest neighbor). More specifically, if the values of features of a pair of examples from the same class are different then the features' importance decreases. Conversely, if the feature values are different for examples from different classes then the features' importance increases.

In the following, we first introduce the distance measures used within the algorithm. Then, the algorithm is described and its computational complexity (including the complexity of computing different distances) is analyzed. Throughout the paper, F and L always denote the number of features and labels respectively.

3.1 Distances: Why and Which

All methods of the Relief family assign feature x_i a weight w_i that is a measure of feature importance in these algorithms. The expected value of the w_i has a nice probability interpretation in the case when both the target and x_i are nominal [14]: simplified to some extent, we have a relation

$$\mathbb{E}[w_i] = \frac{P_{\text{diffAttr, diffTarget}}}{P_{\text{diffTarget}}} - \frac{P_{\text{diffAttr}} - P_{\text{diffAttr, diffTarget}}}{1 - P_{\text{diffTarget}}}, \tag{1}$$

where we define the probabilities $P_{\texttt{ev}} = P(\texttt{ev})$ and $P_{\texttt{ev1, ev2}} = P(\texttt{ev1} \wedge \texttt{ev2})$ that base on the events `diff/sameAttr` (two instances have different/same value of x_i) and `diff/sameTarget` (two instances have different/same target value). The probabilities from the right hand side of Eq. (1) are modeled as the distances in the corresponding spaces: P_{diffAttr} is modeled by the distance d_i on the domain of feature x_i, $P_{\text{diffTarget}}$ is modeled by the distance $d_{\mathscr{L}}$ on the label set $\mathscr{L}$, and $P_{\text{diffAttr, diffTarget}}$ is modeled as their product $d_i d_{\mathscr{L}}$.

First, the distance on the whole descriptive domain $\mathcal{X}$ is defined via the distances d_i on the domains $\mathcal{X}_i$ of features x_i as

$$d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) = \begin{cases} \mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2] & : \mathcal{X}_i \not\subseteq \mathbb{R} \\ \frac{|\boldsymbol{x}_i^1 - \boldsymbol{x}_i^2|}{\max_{\boldsymbol{x}} \boldsymbol{x}_i - \min_{\boldsymbol{x}} \boldsymbol{x}_i} & : \mathcal{X}_i \subseteq \mathbb{R} \end{cases} \qquad d_{\mathcal{X}}(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{1}{F}\sum_{i=1}^{F} d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) \tag{2}$$

where $\mathbf{1}$ is the indicator function with the values $\mathbf{1}[\texttt{true}] = 1$ and $\mathbf{1}[\texttt{false}] = 0$, and max and min go over the examples $\boldsymbol{x}$ in the training set.

For the distance $d_{\mathscr{L}}$ between two sets of labels S^1 and S^2, we consider four options. The use of the first (Hamming Loss) was proposed in [18].

Hamming Loss. This distance is defined as

$$d_{Hamming}(S^1, S^2) = \left|S^1 \setminus S^2 \cup S^2 \setminus S^1\right| \ / \ L. \tag{3}$$

We observe that this is an analogue of $d_{\mathcal{X}}$ from Eq. (2). Encoding a subset $S \subseteq \mathscr{L}$ as a 0/1 vector $\boldsymbol{s}$, where $\boldsymbol{s}_j = 1 \Leftrightarrow \ell_j \in S$, we have $d_{Hamming}(S^1, S^2) = \frac{1}{L}\sum_{j=1}^{L} d_j(\boldsymbol{s}^1, \boldsymbol{s}^2)$, where the numeric part of d_i in Eq. (2) applies in d_j. We believe that there are more suitable choices for the distance $d_{\mathscr{L}}$ that take into account the set structure.

Accuracy. The similarity between two sets can be also measured by their Jaccard index $|S^1 \cap S^2|/|S^1 \cup S^2|$ which is well defined when at least one of the subsets $S^{1,2}$ is not empty (this is the case in our datasets). We then define

$$d_{Accuracy}(S^1, S^2) = 1 - |S^1 \cap S^2| \ / \ |S^1 \cup S^2|. \tag{4}$$

***F*₁ distance.** This distance is defined as

$$d_{F1}(S^1, S^2) = 1 - 2|S^1 \cap S^2| \ / \ (|S^1| + |S^2|), \tag{5}$$

where the second term can be seen as the harmonic mean of the precision and recall [15]. However, these two measures are not symmetric, thus inappropriate as the distance measures.

Subset Accuracy. This distance is defined as

$$d_{SubsetAcc}(S^1, S^2) = \mathbf{1}\left[S^1 \neq S^2\right]. \tag{6}$$

It is the strictest, since it does not differentiate between, e.g., almost the same and disjunctive pairs of subsets. This allows for a faster computation of the distance as compared to the other options (Lemma 2).

Except for the d_{F1}, all distances are also metrics. We named them after the measures that they are expected to optimize (defined in Sec. 4.4), and believe that no other standard measures (see [15, 27]) allow for a direct derivation of distance definitions.

3.2 Algorithm Description

The calculation of the weights $w_i = \mathit{importance}(x_i)$ using the MLC extension of RReliefF is outlined in Alg. 1. RReliefF is an iterative procedure. For each of the m iterations, we randomly select an example $\boldsymbol{r}$ from $\mathscr{D}_{\text{TRAIN}}$ (line 4) and find its K nearest neighbors (line 5) using the distance $d_{\mathcal{X}}$ from Eq. (2). After that, we use the neighbors to update the estimates of probabilities that appear in the definition of the weights (1) for all attributes (lines 8–10). The estimates of probabilities are updated with the weighted average of the distances between $\boldsymbol{r}$ and its neighbors. Here, the distance $d_{\mathscr{L}}$ from the algorithm input is used. The weight $\delta(k) = 1/(mK)$ ensures that $w_i \in [-1, 1]$ when the algorithms finishes. At the end, the weights w_i are computed (line 12) by using the relation (1).

Algorithm 1 MLC-RReliefF($\mathscr{D}_{\text{TRAIN}}$, m, K, $d_{\mathscr{L}}$)

```
1:  $\boldsymbol{P}_{\text{diffAttr, diffTarget}}$, $\boldsymbol{P}_{\text{diffAttr}}$ = zero lists of length $F$
2:  $P_{\text{diffTarget}} = 0.0$
3:  for $\iota = 1, 2, \ldots, m$ do
4:      $\boldsymbol{r}$ = random example from $\mathscr{D}$
5:      $\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_K = K$ nearest neighbors of $\boldsymbol{r}$
6:      for $k = 1, 2, \ldots, K$ do
7:          $P_{\text{diffTarget}}$ += $\delta(\ell) d_{\mathscr{L}}(\boldsymbol{r}, \boldsymbol{n}_k)$
8:          for $i = 1, 2, \ldots, F$ do
9:              $\boldsymbol{P}_{\text{diffAttr}}[i]$ += $\delta(\ell) d_i(\boldsymbol{r}, \boldsymbol{n}_k)$
10:             $\boldsymbol{P}_{\text{diffAttr, diffTarget}}[i]$ += $\delta(\ell) d_i(\boldsymbol{r}, \boldsymbol{n}_k) d_{\mathscr{L}}(\boldsymbol{r}, \boldsymbol{n}_k)$
11: for $i = 1, 2, \ldots, F$ do
12:     $w_i = \frac{\boldsymbol{P}_{\text{diffAttr, diffTarget}}[i]}{P_{\text{diffTarget}}} - \frac{\boldsymbol{P}_{\text{diffAttr}}[i] - \boldsymbol{P}_{\text{diffAttr, diffTarget}}[i]}{1 - P_{\text{diffTarget}}}$
```

The default values of the parameters are set as follows. Typically, we iterate over the whole dataset, i.e., $m = |\mathscr{D}_{\text{TRAIN}}|$. By doing this, the estimates of probabilities are expected to be more accurate. The value of K is typically set small enough to capture the local structure in the data. In that way, we implicitly capture the interactions between features [14].

3.3 Computational complexity

We first analyze the time complexity of a single iteration. Since the space-partitioning data structures, such as kD trees do not perform well when the number of features F is high, we use a brute-force method for finding the nearest neighbors. Hence, the computation of the distances between $\boldsymbol{r}$ and the neighbour candidates takes $\mathcal{O}(MF)$ steps, where $M = |\mathscr{D}_{\text{TRAIN}}|$. In addition to this, the current group of the nearest neighbors must be updated from time to time.

Lemma 1. *The expected number of updates of the group of current nearest neighbors of the instance $\boldsymbol{r}$ is approximately $K \log M$.*

Proof. When we iterate over the neighbors, the group of currently K nearest neighbors is updated if, and only if, at most $K-1$ better candidates have been found so far. Let $\boldsymbol{n}_k$ be the instances from $\mathscr{D}_{\text{TRAIN}} \setminus \{\boldsymbol{r}\}$, sorted increasingly by the distance to $\boldsymbol{r}$, i.e., $\boldsymbol{n}_1$ is the nearest neighbor and $\boldsymbol{n}_{M-1}$ is the farthest neighbor. Let E_k be the expected number of updates when we find the candidate $\boldsymbol{n}_k$. Then, E_k equals the probability p_k of discovering at most $K-1$ of the instances $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_{k-1}$ before $\boldsymbol{n}_k$. Probability $p_{k,s}$ of discovering precisely s of them equals the probability that $\boldsymbol{n}_k$ appears in the $(s+1)$-th position in the random permutation of the instances $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_k$, hence $p_{k,s} = 1/k$, for all $s < k$, and $p_{k,s} = 0$ otherwise. It follows that $p_k = \sum_{s=0}^{K-1} p_{k,s} = \min\{k, K\}/k$.

The total number of expected updates E then equals $E = \sum_{k=1}^{M-1} E_k$, hence $E = \sum_{k=1}^{M-1} p_k = K + K \sum_{k=K+1}^{M-1} \frac{1}{k} = K(1 + H_{M-1} - H_K)$, where k-th harmonic number H_k is defined as $H_k = \sum_{s=1}^{k} 1/s$. Since $\log k < H_k < 1 + \log k$, the leading term in E is indeed $K \log M$.

The overall cost of updating the current nearest neighbours is thus $\mathcal{O}(K \log M \log K)$ if we are using, e.g., the heap structure.

When the neighbours $\boldsymbol{n}_k$, $1 \leq k \leq K$, are computed, the distance between their label set and the label set of $\boldsymbol{r}$ are computed. Considering that we store the label sets as 0/1-lists of length L, this takes $\mathcal{O}(KL)$ steps for the distances $d_{Hamming}$, $d_{Accuracy}$ and d_{F1}, since we have to iterate over all labels. In the case of $d_{SubsetAcc}$, we can do much better, knowing that the labels are typically sparse. To be able to obtain a closed form expression, we will assume that all labels have the same probability to be relevant and that they are independent.

Lemma 2. *The expected value of the labels considered in one computation of* $d_{SubsetAcc}$ *is* $\frac{1-p^L(2-p)^L}{(1-p)^2}$, *where* p *is the probability of a label being relevant.*

Proof. We know that $d_{SubsetAcc}(S^1, S^2) = 1$ as soon as we encounter the label $\ell_l \notin S^1 \cap S^2$. Let X be the number of labels considered. The key observation is that we can easily compute $P(X \geq k) = P(\ell_1, \dots, \ell_{k-1} \in S^1 \cap S^2) = (1 - p)^{2(k-1)}$. This is useful since $\mathbb{E}[X] = \sum_{x=1}^{L} P(X \geq k)$. We obtained geometric series whose sum equals $\mathbb{E}[X] = \frac{1-p^L(2-p)^L}{(1-p)^2}$.

Tab. 1 reveals that the dataset `Delicious` has $L = 983$ labels and label cardinality (average number of labels per example) $\ell_c \doteq 19$. Thus, $p \doteq 0.019$ and $\mathbb{E}[X] \doteq 1.04$, which is considerably smaller than L.

After the distances $d_{\mathscr{L}}$ are computed, the probability estimates are updated in $\mathcal{O}(KF)$ steps. After all iterations, the weights are computed in $\mathcal{O}(F)$ steps, thus the final time complexity is $\mathcal{O}(m[MF + K \log M \log K + KL + KF] + F) = \mathcal{O}(m[MF + KL])$ (in the case of $d_{SubsetAcc}$, L the term KL is replaced by $\mathbb{E}[X]$). If the number of labels is high, then the term KL may not be negligible, which was overlooked in [18].

4 Experimental Design

Here, we give the detailed experimental design for evaluating the performance of the proposed distances. We begin by stating the experimental questions and summarizing the MLC datasets used in this study. Then, we present the evaluation procedure and give the specific parameters instantiations of the methods.

4.1 Experimental questions

The main experimental question is: *Does the choice of the distance* $d_{\mathscr{L}}$ *matter?*

Furthermore, we investigate i) whether the knowledge encapsulated in the feature importances leads to better predictive performance of a model, i.e, are the obtained feature rankings relevant, and ii) how the quality of ranking is influenced by the number of neighbors K and the number of iterations m.

4.2 Datasets

We use 24 MLC benchmark problems. Tab. 1 presents the basic statistics of the datasets. The number of features ranges from 72 to 52350. The features are numeric and nominal. The label set size L ranges from 6 to 983, while the number of training examples ranges from 322 up to 70000. The average number of labels per example (in $\mathscr{D}_{\text{TRAIN}} \cup \mathscr{D}_{\text{TEST}}$), i.e., *label cardinality* is also given. With the exception of *Delicious* dataset, it ranges between 1.0 and 4.38.

The datasets come from different domains. *Arts*, *Business*, *Computers*, *Education*, *Entertainment*, *Health*, *Recreation*, *Reference*, *Science*, *Social* and *Society* describe the problems of finding relevant subtopics of the given main topic of a web page. *Bibtex* and *Bookmarks* are automatic tag suggestion problems, *Birds* deals with predictions of multiple bird species in a noisy environment. *Corel5k* contains Corel images. *Delicious* contains contextual data about web pages along with their tags. *Emotions* deals with emotions in music. *Enron* contains data about emails. *Genbase* and *Yeast* come from biological domain. *Mediamill* was introduced in a video annotation challenge. *Medical* comes from Medical Natural Language Processing Challenge. *Scene* deals with labelling of natural scenes. *TMC2007-500* is about discovering anomalies in text reports.

4.3 Evaluation methodology

We adopted the evaluation methodology that has been previously used in MLC context [18] and in the other types of structured output prediction [17].

We use the same train-test split of the datasets as in the Mulan repository `http://mulan.sourceforge.net/datasets-mlc.html`. A ranking is computed from the training part $\mathscr{D}_{\text{TRAIN}}$ only, and evaluated on the testing part $\mathscr{D}_{\text{TEST}}$.

The quality of the ranking is assessed by using the kNN algorithm where instead of the standard Euclidean distance, its weighted version was used. For two input vectors $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, the distance between them is defined as

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sqrt{\sum_{i=1}^{F} w_i d_i^2(\boldsymbol{x}_i^1, \boldsymbol{x}_i^2)}, \tag{7}$$

where d_i is defined by Eq. (2). The weights are set to $w_i = \max\{\mathit{importance}(x_i), 0\}$, since they need to be made non-negative to ensure that d is well defined, and also to ignore the attributes that have smaller values for importance than a randomly generated attribute would have.

The evaluation through a kNN predictive model was chosen because of two main reasons. First, this is a distance based model, hence, it can easily make use of the information contained in the feature importances in the learning phase. The second reason is kNN's simplicity: its only parameter is the number of neighbors, which we set to 15. In the prediction stage, the neighbors' contributions to the predicted value are equally weighted, so we do not introduce additional parameters that would influence the performance.

Table 1: Data characteristics: sizes of train and test part of the dataset, number of features F, labelset size L and label cardinality ℓ_c.

dataset	$\lvert\mathscr{D}_{\text{TRAIN}}\rvert$	$\lvert\mathscr{D}_{\text{TEST}}\rvert$	F	L	ℓ_c
Arts [26]	3712	3772	23146	26	1.65
Bibtex [10]	4880	2515	1836	159	2.40
Birds [3]	322	323	260	19	1.01
Bookmarks [10]	70000	17856	2150	208	2.04
Business [26]	5710	5504	21924	30	1.60
Computers [26]	6270	6174	34096	33	1.51
Corel5k [7]	4500	500	499	374	3.52
Delicious[25]	12920	3185	500	983	19.02
Education [26]	6030	6000	27534	33	1.46
Emotions[23]	391	202	72	6	1.87
Enron [1]	1123	579	1001	53	3.38
Entertainment [26]	6356	6374	32001	21	1.41
Genbase [6]	463	199	1185	27	1.25
Health [26]	4557	4648	30605	32	1.64
Mediamill [19]	30993	12914	120	101	4.38
Medical [16]	645	333	1449	45	1.25
Recreation [26]	6471	6357	30324	22	1.43
Reference [26]	4027	4000	39679	33	1.17
Scene [2]	1211	1196	294	6	1.07
Science [26]	3214	3214	37187	40	1.45
Social [26]	6037	6074	52350	39	1.28
Society [26]	7273	7239	31802	27	1.67
TMC2007-500 [21]	21519	7077	500	22	2.22
Yeast [8]	1500	917	103	14	4.24

The second rationale for using kNN as an evaluation model is as follows. If a feature ranking is meaningful, then when the feature importances are used as weights in the calculation of the distances kNN should produce better predictions as compared to kNN without using these weights [28].

4.4 Evaluation Measures

In the following, we denote the sets of true and predicted labels for an example $\boldsymbol{x}$ respectively by $y(\boldsymbol{x})$ and $\hat{y}(\boldsymbol{x})$. The measures *Hamming Loss*, *Accuracy*, *F1 Score* and *Subset Accuracy* can be defined in terms of the distances (3)–(6). They are respectively the means (over $\mathscr{D}_{\text{TEST}}$) of the values $d_{Hamming}(y(\boldsymbol{x}), \hat{y}(\boldsymbol{x}))$, $1 - d_{Accuracy}(y(\boldsymbol{x}), \hat{y}(\boldsymbol{x}))$, $1 - d_{F1}(y(\boldsymbol{x}), \hat{y}(\boldsymbol{x}))$ and $1 - d_{SubsetAcc}(y(\boldsymbol{x}), \hat{y}(\boldsymbol{x}))$. Thus, *Hamming Loss* should be minimized while the remaining three should be maximized. We use another four well known measures: *One Error*, *Precision*, *Recall* and area under the pooled precision-recall curve (*pooledAUPRC*). The definitions can be found in [15, 27].

4.5 Statistical Analysis of the Results

For comparing the algorithms, we use the Friedman test. The null hypothesis H_0 is that all considered algorithms have the same performance. If H_0 is rejected by the Friedman's test, we additionally apply Nemenyi or Bonferroni-Dunn post-hoc test. The first is used when we investigate where the statistically significant differences between *any* two algorithms occur, while the second is used when we are interested in the differences between one particular algorithm and the others. A detailed description of all tests is available in [5].

The results of the Nemenyi and Bonferroni-Dunn tests are presented on critical distance diagrams. Each diagram shows the average rank of the algorithm over the considered datasets, and the critical distance, i.e., the distance for which average ranks of two considered algorithms must differ to be considered statistically significantly different. Additionally, the groups of algorithms among which no statistically significant differences occur are connected with a line.

Before proceeding with the statistical analysis, we round the performances to three decimal points. In the analysis, the significance level was set to $\alpha = 0.05$.

4.6 Parameter instantiation

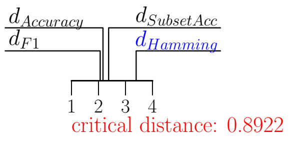
Since the sizes of datasets range over different orders of magnitude, the number of iterations m is given as the proportion of the size of $\mathscr{D}_{\text{TRAIN}}$. The considered values are $m \in \{1\%, 5\%, 10\%, 25\%, 50\%, 100\%\}$. On the other hand, since the number of neighbors K controls the level of locality, it is better given in absolute values. Our choice is to consider the following values $K \in \{1, 5, 10, 15, 20, 25, 30, 40\}$.

5 Results

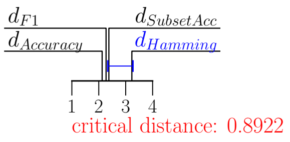
5.1 Does the Distance Matter?

To give every distance as good chance as possible, we compute and evaluate feature rankings for all combinations of the parameters m and K and for every dataset and distance version, the best pair (with respect to the evaluation measure at hand) is chosen.

Friedman test rejected the null hypothesis for three of the four evaluation measures that the distance definitions are part of: *Accuracy* ($p = 5.2 \cdot 10^{-4}$), *F1 Score* ($p = 3.5 \cdot 10^{-4}$) and *Subset Accuracy* ($p = 0.011$). In the case of *Hamming Loss*, the performances are not statistically significantly different ($p = 0.28$). The Bonferroni-Dunn test reveals that $d_{Hamming}$ performs statistically significantly worse than the other three distances, for the evaluation measures *Accuracy* (Fig. 1a) and *F1 Score* (with qualitatively the same diagram). In the case of *Subset Accuracy*, it has still the worst performance, but it is not statistically significantly worse than $d_{SubsetAcc}$ (Fig. 1b). Interestingly enough, the hypotheses was not rejected for the *Hamming Loss* evaluation measure. Also in this case, the rankings with $d_{Hamming}$ have the worst average rank of 2.9 (as compared to the best average rank of 2.1 that belongs to $d_{Accuracy}$), which leads us to a



(a) Evaluation measure: *Accuracy*



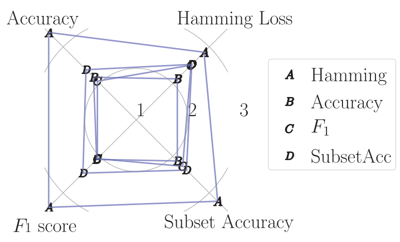
(b) Evaluation measure: *Subset Accuracy*

Fig. 1: Comparison of the four distance functions in terms of a) *Accuracy*, and b) *Subset Accuracy*: Critical distance diagrams from Bonferroni-Dunn test with the baseline $d_{Hamming}$.

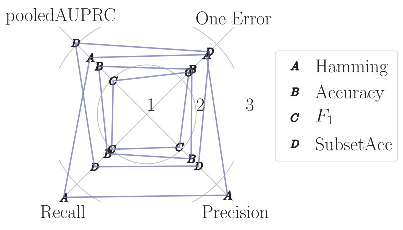
conclusion that the rankings with $d_{Hamming}$ are indeed to some extent optimized for *Hamming Loss*, but not sufficiently. Average ranks for this four measures are shown on the radar plot in Fig. 2a.

The average ranks of the feature rankings with respect to the other four measures are shown in Fig. 2b. Here, the null hypothesis H_0 is rejected in the case of *Precision* ($p = 0.0011$) and *Recall* ($p = 3.8 \cdot 10^{-4}$). This is not that surprising, since optimizing for F_1 *Score* should directly result in optimized *Precision* and/or *Recall*, as noted after the definition of d_{F1} (Eq. (5)). The results of the follow-up Bonferroni-Dunn tests are similar to those for *Subset Accuracy*: rankings obtained with $d_{Hamming}$ have the worst rank, but are not statistically significantly worse than those obtained with $d_{SubsetAcc}$. Additionally, H_0 is also rejected in the case of *pooledAUPRC* ($p = 0.027$), but in this case, no ranking is statistically significantly different from the one that corresponds to $d_{Hamming}$.

Since we have rejected the null hypotheses (all algorithms perform equally well) in 6 of 8 cases, we can already claim that choosing an appropriate distance measure does matter. Moreover, both diagrams in Fig. 2 show that our newly



(a) Distance related measures



(b) Other measures

Fig. 2: Average ranks of the rankings computed with the four distance functions (denoted by A, B, C and D), in terms of measures that a) are, and b) are not directly related to any of distances.

proposed distance definitions result in rankings that outperform those computed with $d_{Hamming}$. A reason for this may be that the latter cannot really capture the possible interactions between the labels since it can be decomposed to the per-label distances, as noted in Sec. 3.1. This may also be the reason why the rankings computed with the newly proposed distances are typically closer to each other than to the rankings computed with $d_{Hamming}$.

To detect the differences among the rankings, we also apply Nemenyi post-hoc test. In addition to the relations discovered with Bonferroni-Dunn test, we now know that there is statistically significant difference between d_{F1} and $d_{SubsetAcc}$, when the quality is measured in terms of *pooledAUPRC*.

5.2 Are the Obtained Rankings Relevant?

To answer this question, we partially repeat the analysis from the previous section: in addition to the evaluation of the four ranking types, also the non-weighted 15NN algorithm is evaluated. If we reject the null hypothesis H_0 with Friedman test, the four rankings are compared to the non-weighted 15NN classifier with Bonferroni-Dunn post-hoc test. If there is a statistically significant difference between the weighted 15NN classifier and non-weighted 15NN classifier (in favour of the weighted one), we proclaim the ranking relevant.

H_0 is rejected for all evaluation measures. The corresponding Bonferroni-Dunn tests identifies the following. The distances $d_{Accuracy}$ and d_{F1} always result in relevant rankings. The distance $d_{SubsetAcc}$ fails to result in relevant rankings in the case of *One Error*. The distance $d_{Hamming}$ results in relevant rankings when the quality is measured in terms of *Subset Accuracy* and *pooledAUPRC*.

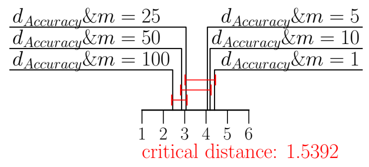
5.3 Influence of the Parameters m and K

To assess how does the number of iterations m influence the quality of ranking, we choose one of the distance functions and a value for the number of neighbors K. When m varies over the values specified in Sec. 4.6, six different rankings are obtained. We compare their quality in terms of the chosen evaluation measure, by applying the Friedman test.

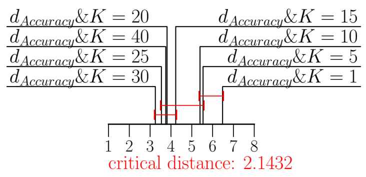
H_0 is rejected for all values of m and for all versions of target distance in the case of *Accuracy*, *F_1 Score*, *Precision*, *Recall* and *Subset Accuracy*. In the case of *Hamming Loss*, it is never rejected. In the case of *One Error*, it is rejected for d_{F1} when $K \geq 25$ and for $d_{SubsetAcc}$ when $K = 40$. In the case of *pooledAUPRC*, the hypothesis is only rejected for $d_{SubsetAcc}$ when $m = 40$.

The only values of m which are always in the top performing group of algorithms, are 25%, 50% and 100%. A typical critical distance diagram (for $d_{Accuracy}$ and $K = 20$) is shown in Fig. 3a.

To assess the influence of the number of Relief neighbours K, a similar analysis is performed, now with the interchanged roles of m and K: the former is fixed and the latter varies. The summary of the results is as follows. Number of neighbors seems to have a lesser influence on the quality, since we do not reject all hypotheses for any of the evaluation measures. However, this is mostly due



(a) Influence of the number of Relief iterations, for $d_{Accuracy}$ when $K = 20$.



(b) Influence of the number of Relief neighbors, for $d_{Accuracy}$ when $m = 25\%$.

Fig. 3: Critical distance diagrams from Nemenyi tests that show the influence of the number of a) iterations, and b) neighbors, on the quality of the $d_{Accuracy}$ rankings, measured in terms of *Precision*.

to the fact that K almost never statistically significantly influence the quality of the $d_{Hamming}$ rankings. For the other distances, the hypothesis is always rejected when the quality is measured in terms of *Accuracy*, *F1 Score*, *Precision*, *Recall*. This also holds for *Subset Accuracy* with two exceptions for $d_{SubsetAcc}$: $m \in \{1\%, 100\%\}$. Again, no hypothesis is rejected in the case of *Hamming Loss*.

Typically, more is better regarding the number of neighbors and the highest values of K, i.e., $K \in \{30, 40\}$ have often the best average rank. This can be explained by the sparsity of the labels. To properly asses the average label space distance in $\mathscr{D}_{\text{TRAIN}}$, one has to consider larger neighborhoods. However, the differences among the algorithms for which $K \geq 15$ are not statistically significant. A typical situation (for $d_{Accuracy}$ and $m = 25\%$) is shown in Fig. 3b.

6 Conclusions and Future Work

In this paper, we propose the use of three distance measures on the target space within an extension of RReliefF approach to feature ranking for MLC tasks. These are the distances that are used within the evaluation measures *Accuracy*, *F1 Score* and *Subset Accuracy* for predictive performance on MLC tasks. We have shown that using any of these distances always results in rankings of higher quality than the rankings computed with the distance used in the evaluation measure *Hamming Loss* [18]. Additionally, the newly proposed measures outperform the old one in terms of *Precision* and *Recall*, since these two are directly connected to the *F1 Score*. For more independent measures, such as *pooledAUPRC* and *One Error* we did not observe any differences, so we can conclude that the use of the proposed distance within RReliefF optimizes the corresponding MLC evaluation measures.

We have also shown that all proposed rankings are relevant by comparing the nearest neighbor classifier that uses feature relevance information, to the standard nearest neighbor classifier. Additionally, we measure the influence of the parameters m (number of Relief iterations) and K (number of Relief neighbors) and show that rankings computed from $m = 25\%$ of the training dataset cannot be statistically significantly outperformed on average. The same goes for rankings that were computed by examining the neighborhoods of size $K = 15$.

There are several directions for future work. We plan to find appropriate distance measures for the hierarchical version of the MLC task: hierarchical multi-label classification. Incorporating probabilities in the distances, the Relief family can be also extended in the direction of data with missing labels and semi-supervised problems. Once these are solved, we also plan to develop an extension of Relief for seemingly much harder context of unsupervised learning, where there are no target variables and the analogous approach cannot be taken.

References

1. UC Berkeley Enron Email Analysis Project. `http://bailando.sims.berkeley.edu/enron_email.html`, Accessed: 2018-06-28
2. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognition **37**(9), 1757–1771 (2004)
3. Briggs, F., Huang, Y., Raich, R., Eftaxias, K., Lei, Z., Cukierski, W., Frey Hadley, S., Hadley, A., Betts, M., Fern, X.Z., Irvine, J., Neal, L., Thomas, A., Fodor, G., Tsoumakas, G., Ng Hong, W., Nguyen, T.N.T., Huttunen, H., Ruusuvuori, P., Manninen, T., Diment, A., Virtanen, T., Marzat, J., Defretin, J., Callender, D., Hurlburt, C., Larrey, K., Milakov, M.: The 9th annual mlsp competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In: IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2013. p. 1 (2013)
4. Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: On label dependence and loss minimization in multi-label classification. Machine Learning **88**(1), 5–45 (2012)
5. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (2006)
6. Diplaris, S., Tsoumakas, G., Mitkas, P., Vlahavas, I.: Protein classification with multiple algorithms. In: 10th Panhellenic Conference on Informatics (PCI 2005). pp. 448–456 (2005)
7. Duygulu, P., Barnard, K., de Freitas, J.F.G., Forsyth, D.A.: Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) Computer Vision — ECCV 2002. pp. 97–112. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
8. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems 14. Springer International Publishing (2001)
9. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3**, 1157–1182 (2003)
10. Katakis, I., Tsoumakas, G., Vlahavas, I.: Multilabel text classification for automated tag suggestion. In: Proceedings of the ECML/PKDD 2008 Discovery Challenge (2008)
11. Kira, K., Rendell, L.A.: The feature selection problem: Traditional methods and a new algorithm. In: Proceedings of the Tenth National Conference on Artificial Intelligence. pp. 129–134. AAAI'92, AAAI Press (1992)
12. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition **46**(3), 817–833 (2013)

13. Kong, D., Ding, C., Huang, H., Zhao, H.: Multi-Label ReliefF and F-statistic Feature Selections for Image Annotation. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2352–2359 (2012)
14. Kononenko, I., Robnik-Šikonja, M.: Theoretical and Empirical Analysis of ReliefF and RReliefF. Machine Learning Journal **55**, 23–69 (2003)
15. Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recognition **45**, 3084–3104 (2012)
16. Pestian, J.P., Brew, C., Matykiewicz, P., Hovermale, D.J., Johnson, N., Bretonnel Cohen, K., Duch, W.: A shared task involving multi-label classification of clinical free text. In: Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing (BioNLP '07). pp. 97–104 (2007)
17. Petković, M., Džeroski, S., Kocev, D.: Feature ranking for multi-target regression with tree ensemble methods. In: Yamamoto, A., Kida, T., Uno, T., Kuboyama, T. (eds.) Discovery Science. pp. 171–185. Springer International Publishing (2017)
18. Reyes, O., Morell, C., Ventura, S.: Scalable extensions of the relieff algorithm for weighting and selecting features on the multi-label learning context. Neurocomputing **161**, 168 – 182 (2015)
19. Snoek, C.G.M., Worring, M., van Gemert, J.C., Geusebroek, J.M., Smeulders, A.W.M.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: Proceedings of the 14th ACM International Conference on Multimedia. pp. 421–430. ACM, New York, NY, USA (2006)
20. Spolaôr, N., Cherman, E.A., Monard, M.C., Lee, H.D.: A comparison of multi-label feature selection methods using the problem transformation approach. Electronic Notes in Theoretical Computer Science **292**, 135–151 (2013)
21. Srivastava, A.N., Zane-Ulman, B.: Discovering recurring anomalies in text reports regarding complex space systems. In: 2005 IEEE Aerospace Conference (2005)
22. Stańczyk, U., Jain, L.C. (eds.): Feature Selection for Data and Pattern Recognition. Studies in Computational Intelligence, Springer Berlin Heidelberg (2015)
23. Trochidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In: 2008 International Conference on Music Information Retrieval (ISMIR 2008). pp. 325–330 (2008)
24. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining pp. 1–13 (2007)
25. Tsoumakas, G., Katakis, I., Vlahavas, I.: Effective and efficient multilabel classification in domains with large number of labels. In: ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08) (2008)
26. Ueda, N., Saito, K.: Parametric mixture models for multi-labeled text. In: Advances in Neural Information Processing Systems 15. pp. 721–728. MIT Press (2003)
27. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning **73**(2), 185–214 (2008)
28. Wettschereck, D.: A study of distance based algorithms. Ph.D. thesis, Oregon State University, USA (1994)

Machine Learning Journal manuscript No.
(will be inserted by the editor)

Multi-label feature ranking with ensemble methods

Matej Petković · Sašo Džeroski · Dragi Kocev

Received: date / Accepted: date

Abstract In this paper, we propose three ensemble-based feature ranking scores for multi-label classification (MLC), which is a generalisation of multi-class classification where the classes are not mutually exclusive. Each of the scores (`Symbolic`, `Genie3` and `Random forest`) can be computed from three different ensembles of predictive clustering trees: *Bagging*, *Random forest* and *Extra trees*. We extensively evaluate the proposed scores on 24 benchmark MLC problems, using 15 standard MLC evaluation measures. We determine the ranking quality saturation points in terms of the ensemble sizes, for each ranking-ensemble pair, and show that quality rankings can be computed really efficiently (typically 10 or 50 trees suffice). We also show that the proposed feature rankings are relevant and determine the most appropriate ensemble method for every feature ranking score. We empirically prove that the proposed feature ranking scores outperform current state-of-the-art methods in the quality of the rankings (for the majority of the evaluation measures), and in time efficiency. Finally, we determine the best performing feature ranking scores. Taking into account the quality of the rankings first and - in the case of ties - time efficiency, we identify the `Genie3` feature ranking score as the optimal one.

Keywords feature ranking · multi-label classification · ensemble-based methods · predictive clustering trees

We acknowledge the financial support of the Slovenian Research Agency via the grants P2-0103, J2-9230, J7-1815, J7-9400 and N2-0128, and a young researcher grant to MP. The computational experiments presented here were executed on a computing infrastructure from the Slovenian Grid (SLING) initiative.

Matej Petković · Sašo Džeroski · Dragi Kocev
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Jožef Stefan International Postgraduate School
Jamova 39, 1000 Ljubljana, Slovenia
Tel.: +386-1477-3635
E-mail: matej.petkovic@ijs.si, E-mail: saso.dzeroski@ijs.si, E-mail: dragi.kocev@ijs.si

1 Introduction

As opposed to the standard classification problems, where the goal is to learn a model that predicts one of the two or more mutually exclusive predefined class values, e.g., predict whether a given board position leads to a `win`, `draw` or `loss` if both players are playing optimally, multi-label classification (MLC) is a predictive modeling task where the examples can be labeled with more than one (or even zero) of the labels from a predefined set of labels $\mathscr{L}$. In this case, we denote examples as $(\boldsymbol{x}, \boldsymbol{y})$, where i) $\boldsymbol{x}$ is a vector of values of features x_i that are either numeric (the domain of x_i is a subset of $\mathbb{R}$) or nominal (the domain of x_i is a finite set of values), and ii) $\boldsymbol{y}$ is a subset of the label set $\mathscr{L}$. The elements of $\boldsymbol{y}$ are the labels that are *relevant* for a given example.

MLC problems are receiving more and more attention from the research community. For example, one of the use cases of MLC is labeling pictures with objects that appear on them. Due to the abundance of data more and more efficient methods are needed, and this is were feature ranking can play a significant role.

One can always approach a MLC problem by transforming the data and then use standard classification method to learn predictive models. The most typical approach is binary relevance where the MLC problem is divided into a set of $L = |\mathscr{L}|$ standard classification problems, and a separate model for each of the labels is build separately, i.e., a model that predicts whether a given label ℓ is relevant for a given example. In the end, these predictions are joined into a final one [Tsoumakas and Vlahavas, 2007]. However, building a separate model for each label might be too costly as well as it prevents the predictive model to take into account dependencies among the labels in the learning phase. A similar, yet even more time consuming approach is to build $L(L-1)/2$ predictive models, one for each pair of labels $\ell_1 \neq \ell_2$ [Elkafrawy et al., 2015]. In the end, the votes for every label are summed up and the ones with the most votes are predicted as relevant.

Another in the group of problem transformation approaches is the power set approach [Tsoumakas and Vlahavas, 2007] where subsets $\boldsymbol{y}$ are looked upon as the elements of the power set $\mathcal{P}(\mathscr{L})$. It transforms the data into a standard classification problem. A drawback of this approach is that the maximal number of classes 2^L may be really high and the data becomes very sparse.

Another group of methods are the so called method adaptation approaches where an existing method is adapted to directly address MLC problems. This has been done, for example, in the case of predictive clustering trees [Madjarov et al., 2012] and support vector machines [Elisseeff and Weston, 2001].

The main topic of this paper is the task of *feature ranking*. It is a machine learning task and it is very closely related to predictive modelling. Input to a feature ranking algorithm is a dataset $\mathscr{D}$ and its output are feature relevance scores $\mathit{relevance}(x_i)$, for all features x_i. The relevance score of a feature x_i assesses how dependent are the target values $\boldsymbol{y}$ on the feature x_i.

By performing a feature ranking, we can explain the predictions of complex black-box models, such as deep neural networks or ensembles. Another motivation for feature ranking is dimensionality reduction, i.e., performing feature selection [Lee and Kim, 2017, Sechidis et al., 2014]. The task of feature selection can be viewed as a subtask of feature ranking, where we select the most relevant features, and ignore the others when building a predictive model. In that way, the obtained models are easier to understand, faster to build and less prone to overfitting. The selection of the most relevant features is done by first computing the feature relevances, and then retaining the features with the top k highest scores, for a $k \in \mathbb{N}$.

Approaches to the MLC feature ranking tasks are analogous to those of MLC predictive modelling. We can convert a dataset into L classification datasets (as in binary relevance), compute ranking for each label separately and aggregate feature relevances across the labels to obtain the final ranking. Other data transformation approaches are possible in combination with the many feature ranking methods for classification are applicable [Guyon and Elisseeff, 2003]: convert the problem into L binary classification tasks, or convert the problem into a multi-class classification task with 2^L classes by using the label power set approach. These two approaches inherit the drawbacks from the MLC task: too high computational cost due to the many binary classification tasks, or sparsity due to the large number of classes in the multi-class classification task. These drawbacks might lead to potentially irrelevant feature rankings.

Method adaptation approaches for MLC feature ranking are divided into three major groups - a heritage from the standard classification scenario [Guyon and Elisseeff, 2003]. The first group are *filters* and they do not need any predictive model. They are typically computationally very efficient, but can be myopic, i.e., do not make use of the feature dependencies. The second group are *wrappers* and they heavily use predictive models but are very appropriate more or less only for feature selection, since they typically return a feature subset without relevances. Usually, the procedure goes as follows: a candidate feature set is chosen at the beginning. Then, a predictive model that uses only features from this feature set is learned, the feature set is updated according to the predictive performance of the model, and the cycle repeats. The third group of feature ranking algorithms are *embedded* methods where a single predictive model is built and feature relevances are computed directly from it.

The rest of the paper is organized as follows. In Sec. 2 we describe some related work. In Sec. 3, the necessary background for the proposed scores is presented, and in Sec. 4 description of the scores is given. In Sec. 5, the experimental setup is presented (with the listing of experimental questions, evaluation procedure etc). Sec. 6 we present and discuss the results of the experiments. In Sec. 7, we conclude and give directions for further work.

2 Related Work

The datasets in our experiments have hundreds of labels (see Tab. 2), hence, the data transformation approaches are inappropriate or even infeasible. These approaches include methods addressing the feature ranking through binary relevance, label power set [Spolaôr et al., 2013] and the 1-vs-1 version of the ReliefF algorithm [Kong et al., 2012]. Consequently, we will only focus on the method adaptation approaches. Furthermore, as explained above, the methods from the wrapper subgroup of method adaptation methods are also not really appropriate for all feature ranking tasks, so we will review the filter and the embedded approaches and start with the former.

There exist adaptations of information theory based feature ranking scores, such as information gain [Pereira et al., 2015], or feature ranking scores that base on statistical tests, such as χ^2 feature ranking [Spolaôr and Tsoumakas, 2013]. However, none of them can exploit the feature dependencies, and they all need additional preprocessing of the data, since the numeric features have to be discretized prior to computing the feature ranking.

These drawbacks are addressed by the RReliefF-ML adaptation [Reyes et al., 2015] of the RReliefF algorithm [Kononenko and Robnik-Šikonja, 2003] to MLC setting. This is a generalization of the regression version of the Relief algorithm, where only one set of neighbours needs to be computed in every iteration, as opposed to the other versions that base on the Relief for classification and need L (binary relevance), $\mathcal{O}(L^2)$ (the version from [Kong et al., 2012]) or 2^L groups (power set approach) in the worst case.

The RReliefF-ML algorithm was empirically shown to yield relevant feature rankings [Reyes et al., 2015] by using it on 34 benchmark problems and applying Friedman statistical test. However, the very basic assumption of test (independence of datasets) was not met [Petković et al., 2018]. Moreover, it has been shown [Dembczyński et al., 2012] that different MLC evaluation measures result in different optimal classifiers when a predictive modelling problem is being solved, which should also reflect in the field of feature ranking. This was the motivation for parameterizing the distance in the RReliefF-ML algorithm and for proposing three different distance definitions that mimic standard MLC evaluation measures: the original hamming loss distance was compared to the distances that base on the multi-label accuracy, multi-label F_1 measure and subset accuracy. In general, the resulting MLC-Relief algorithm outperformed the original.

The only embedded feature ranking method that we could found is computed from a *Random forest* of predictive clustering trees (PCTs) [Kocev et al., 2013a]. There, a *Random forest* of 500 PCTs is built and the ranking is computed out of it as proposed in [Breiman, 2001]. The preliminary study of the method was carried out on the limited number of four benchmark datasets (all four of them are included in our study). In this work, we extend that work in the following ways:

1. In addition to *Random forest*, we propose to use two new ensemble methods: *Bagging* and *Extra trees*
2. In addition to the feature ranking score from [Kocev et al., 2013a], we propose two new: `Symbolic` and `Genie3`
3. We extensively evaluate the methods on 24 benchmark datasets
4. We analyze the computational complexity of the proposed feature ranking algorithms, and show that they can be computed much more efficiently that those in [Kocev et al., 2013a] (up to 50-times faster).
5. We compare the proposed methods to the non-informed features rankings and state-of-the-art feature ranking methods, and show that we outperform the baselines.

3 Background

3.1 Multi-Label Classification PCTs

The proposed feature ranking method is based on ensembles of predictive clustering trees (PCTs) – these are an adaptation of the standard decision trees [Breiman et al., 1984] to different types of structured output prediction tasks [Blockeel, 1998,

Table 1: The algorithm for learning predictive clustering trees. Algorithm 1 gives the top-down induction procedure, while Algorithm 2 gives the best test selection procedure.

Algorithm 1 PCT(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$
2: if $t^* \neq none$ then
3:     for each $E_i \in \mathcal{P}^*$ do
4:         $tree_i = \text{PCT}(E_i)$
5:     return $Internal(t^*, \bigcup_i \{tree_i\})$
6: else
7:     return $Leaf(\text{Prediction}(E))$
```

Algorithm 2 BestTest(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: for each test $t$ do
3:     $\mathcal{P}$ = partition induced by $t$ on $E$
4:     $h = |E| impu(E) - \sum_{E_i \in \mathcal{P}} |E_i| impu(E_i)$
5:     if $h > h^*$ then
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $(t^*, h^*, \mathcal{P}^*)$
```

Kocev et al., 2013b], including MLC, and also clustering. The induction procedure is given in Alg. 1 (from Tab. 1). The input to the method is a set E of training examples, i.e., a training set. The procedure then selects the split test from all candidate splits based on the evaluation of the heuristic score h. The best split is thus chosen to partition the subset E into subsets E_i, an internal node is created (with the selected split), and the method is called recursively on each of the subsets. If no good split exists, a leaf node is created and a prediction for that node is computed.

The adaptation to a given task is completely defined by the impurity measure *impu* used when evaluating the splits (line 5 in Alg. 2, from Tab. 1), and prediction function Prediction. The heuristic greedily guides the tree induction since finding the tree with the smallest depth is a NP-hard problem [Hancock et al., 1996].

In the implementation of PCTs (available at `http://source.ijs.si/ktclus/clus-public`), a label set $\boldsymbol{y}$ is internally represented as its incidence vector $\mathbf{y}$ whose j-th component $\mathbf{y}_j$ takes the value 1 when $\ell_j \in \boldsymbol{y}$ and takes the value 0 otherwise. The impurity of the label ℓ_j for the subset E is then defined as the variance $Var(E)_j$ of $\mathbf{y}_j$. These values are aggregated in order to obtain the impurity of $\boldsymbol{y}$ for the subset E as

$$impu(E) = \sum_{j=1}^{L} \frac{Var(E)_j}{Var(\mathscr{D}_{\text{TRAIN}})_j},$$

i.e., as the average of the normalized variances of the labels.

The prediction $\hat{\boldsymbol{y}}$ for a subset E that belongs to a leaf is obtained by first computing the average incidence vector

$$\overline{\mathbf{y}} = \frac{1}{|E|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in E} \mathbf{y},$$

which gives us the estimates of probabilities $\overline{\mathbf{y}}_j = P(\ell_j \in \boldsymbol{y} \mid \boldsymbol{x})$, and then predicting $\ell_j \in \boldsymbol{y}$ if and only if $\overline{\mathbf{y}}_j \geq 1/2$.

3.2 Ensembles of PCTs

An ensemble is a set of base predictive models. Its prediction for an example $\boldsymbol{x}$ is made by combining the predictions of the ensemble base models. In the case of MLC and ensembles of PCTs, this is typically done by first averaging the probabilities that single trees give, and then applying the threshold of 1/2 as in the single tree case.

The motivation for introducing an ensemble is that it can be viewed as more stable versions of its base models, since - if the members are diverse models [Hansen and Salamon, 1990] - the ensemble predictions have lower variance and are therefore more accurate. Even though we do not use the ensembles as predictive models and rather just compute feature ranking scores out of them, the same motivation still holds. Since every tree is built independently of the others we define the ensemble feature ranking scores as the averages of the scores over the trees in the ensemble.

To introduce some diversity into the tree induction, one has to modify the completely deterministic tree induction algorithm (Alg. 1). There are several ways to do so and we use three of them.

***Bagging* and *Random forest*.** Instead of being learned from the whole training set $\mathscr{D}_{\text{TRAIN}}$, each tree in the *Bagging*/*Random forest* ensemble is built from a different bootstrap replicate $\mathcal{B}$ of $\mathscr{D}_{\text{TRAIN}}$, called bag. The examples $\mathscr{D}_{\text{TRAIN}} \setminus \mathcal{B}$ are called out-of-bag examples (OOB). Additionally, instead of evaluating all possible tests in the *BestTest* procedure (Alg. 2), only the tests that a random subset of features yield, are evaluated. The number of the retained features f is a parameter to the algorithm. Its typical values are $\lceil \log_2 F \rceil$ or $\lceil \sqrt{F} \rceil$, etc. If $f = F$, i.e, we keep all the features, we obtain the *Bagging* procedure, and *Random forest* procedure otherwise.

***Extra trees* ensembles.** As in *Random forest*, we again consider f features in each node, but we do not evaluate all the corresponding tests. Rather, we choose randomly only one test per feature and choose the best one among these f tests. From the bias-variance point of view, the rationale behind the Extra-Trees method is that the explicit randomization of the cut-point and feature combined with ensemble averaging should be able to reduce variance more strongly than the weaker randomization schemes used by other methods [Geurts et al., 2006]. Note that, however, Extra-Tree ensembles do not use bootstrapping.

4 Feature Ranking Scores

We first propose and describe the `Symbolic` score. Then, we proceed explaining the `Genie3` [Huynh-Thu et al., 2010] and the ***`Random forest`** score* [Breiman, 2001] In the following, a tree is denoted by $\mathcal{T}$, whereas $\mathcal{N} \in \mathcal{T}$ denotes a node. Trees form an ensemble $\mathcal{E}$ of the size $|\mathcal{E}|$. The set of all internal nodes of a tree $\mathcal{T}$ in which the feature x_i appears as part of a test is denoted as $\mathcal{T}(x_i)$.

`Symbolic` score. The main motivation for this score is that the *relevance*(x_i) should be proportional to the number of occurrence of the feature x_i in the splits in the trees. However, tests closer to the root influence more examples, hence the depth of the nodes should be also taken into account. The relevance of feature x_i

as defined by the **Symbolic** score is thus given as

$$relevance_{\text{SYMB}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \sum_{\mathscr{N} \in \mathcal{T}(x_i)} \alpha^{\text{depth}(\mathscr{N})}, \tag{1}$$

where the value of the parameter $\alpha \in (0, 1]$ controls how quickly the influence of a node decreases with its depth. Setting it to $\alpha = 1$ results in simple counting of the appearances of the features in the tree nodes.

Genie3 score. The main motivation for this score is that a relevant feature x_i highly reduces the impurity of the target variable if it appears in the split in the tree. The **Genie3** relevance of the feature x_i is thus defined as

$$relevance_{\text{GENIE3}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \sum_{\mathscr{N} \in \mathcal{T}(x_i)} |E(\mathscr{N})| h^*(\mathscr{N}), \tag{2}$$

where $E(\mathscr{N})$ is the set of examples that come to the node $\mathscr{N}$, and $h^*(\mathscr{N})$ is the heuristic value of the split, i.e., the variance reduction, as described in the Alg. 2. Additionally, greater emphasis is again put on the nodes closer to the root that are reached by higher number of examples $E(\mathscr{N})$.

Random forest (RF) score. This score tests how much noising a feature decreases the predictive performance of the trees in the ensemble. The greater the performance degradation, the more important the feature is.

Once a tree $\mathcal{T}$ in an ensemble is grown, it typically overfits to the dataset it was built for, since no pruning is used in the considered ensemble methods. Thus, the performance of the tree has to be assessed by using out-of-bag examples $\text{OOB}_{\mathcal{T}}$ that were not part of the tree induction. This results in the error $Err(\text{OOB}_{\mathcal{T}})$ as measured with Hamming loss. Afterward, the values of feature x_i in the set $\text{OOB}_{\mathcal{T}}$ are randomly permuted. We denote the perturbed OOB set by $\text{OOB}_{\mathcal{T}}^i$, and the corresponding predictive error by $Err(\text{OOB}_{\mathcal{T}}^i)$. The relevance of the feature x_i as the relative increase of the error after noising, averaged over the trees in the ensemble, namely

$$relevance_{\text{RF}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \frac{Err(\text{OOB}_{\mathcal{T}}^i) - Err(\text{OOB}_{\mathcal{T}})}{Err(\text{OOB}_{\mathcal{T}})}. \tag{3}$$

It is evident that this feature ranking score cannot be computed from an ensemble of *Extra trees*, since they do not use bootstrapping. Note also that due to the possibly numerous permutations of the OOB datasets (whose size is on average more than one third of $\mathscr{D}_{\text{TRAIN}}$) and sending the examples through the trees twice, the time complexity of computing the ranking is not negligible compared to that of building the tree, which is the case for the other two feature ranking scores. However, the algorithm can be made more efficient if we note that $Err(\text{OOB}_{\mathcal{T}}^i) = Err(\text{OOB}_{\mathcal{T}})$ for the features x_i that do not appear in $\mathcal{T}$,

Finally, while the first two scores are inherently related to trees, the permutation mechanism of the **Random forest** score can also be used with the other learners. However, building an ensemble of decision trees using *Random forest* or *Bagging* is very appropriate since i) these two ensembles use bootstrapping, and ii) the time complexity of the trees for making predictions is low.

4.1 Theoretical Aspects

Here, we first give the time complexity of the proposed methods, and then the convergence properties of the scores with increasing number of trees.

Under the relaxed balancedness assumption of the trees that a tree has depth $\mathcal{O}(\log n)$, we can quickly derive (by counting the number of operations in the nodes) the following time complexities. The time complexity for inducing *Bagging* is $\mathcal{O}(|\mathcal{E}|FLn\log^2 n)$, where $|\mathcal{E}|$ is the size of the ensemble, F is the number of features, l is the number of labels, n is the number of examples.

Next, we give the following observation for computing the value of $\mathit{relevance}(x_i)$ (using any ensemble method).

Proposition 1 *Variance of the random variable* $R_i = \mathit{relevance}(x_i)$ *decreases linearly with the number of trees in an ensemble.*

Proof Let $R_{i,t}$ be the relevance of some fixed variable x_i as computed from the tree t. Note that the Eqs. (1)–(3) say that $R_i = \frac{1}{|\mathcal{E}|}\sum_{t=1}^{|\mathcal{E}|} R_{i,t}$. Moreover, $R_{i,t}$ are independent and identically distributed variables, thus $\mathit{Var}(R_i) = \mathit{Var}R_{i,1}/|\mathcal{E}|$. □

Corollary 1 *With probability one, the order of features with respect to their ranking converges to some final ranking, when* $\mathcal{E} \nearrow \infty$.

Proof The law of large numbers assures that the distribution of R_i variables converges to the Dirac delta function, positioned at $\mathbb{E}[R_i]$. The fact that the expected values are all different equal with probability one, concludes the proof. □

5 Experimental Setup

5.1 Experimental Questions

We perform experiments to answer the following questions:

1. For a given ranking-ensemble pair, how many trees are needed to reach the saturation point in the quality of the ranking?
2. Are the obtained feature rankings relevant, i.e., are they better than a non-informed ranking that considers all features equally important?
3. What is the most suitable ensemble method for a given feature ranking score?
4. Do the proposed feature ranking scores outperform state-of-the-art baseline?
5. Which feature ranking score is the best?

As state-of-the-art baseline, we take the MLC-Relief method.

5.2 Datasets

We use the same 24 MLC benchmark problems that were used in [Petković et al., 2018]. Tab. 2 presents the basic statistics of the datasets. The number of features ranges from 72 to 52350. The features are numeric and nominal. The label set size L ranges from 6 to 983, while the number of training examples ranges from 322 up to 70000. The average number of labels per example (in $\mathscr{D}_{\text{TRAIN}} \cup \mathscr{D}_{\text{TEST}}$), i.e., *label cardinality* is also given. With the exception of *Delicious* dataset, it ranges between 1.0 and 4.38.

The datasets come from different domains. *Arts*, *Business*, *Computers*, *Education*, *Entertainment*, *Health*, *Recreation*, *Reference*, *Science*, *Social* and *Society* describe the problems of finding relevant subtopics of the given main topic of a web page. *Bibtex* and *Bookmarks* are automatic tag suggestion problems, *Birds* deals with predictions of multiple bird species in a noisy environment. *Corel5k* contains Corel images. *Delicious* contains contextual data about web pages along with their tags. *Emotions* deals with emotions in music. *Enron* contains data about emails. *Genbase* and *Yeast* come from biological domain. *Mediamill* was introduced in a video annotation challenge. *Medical* comes from Medical Natural Language Processing Challenge. *Scene* deals with labelling of natural scenes. *TMC2007-500* is about discovering anomalies in text reports.

Table 2: Data characteristics: Sizes of the train and test parts of the dataset, number of features F, labelset size L and label cardinality ℓ_c.

dataset	$\lvert\mathcal{D}_{\text{TRAIN}}\rvert$	$\lvert\mathcal{D}_{\text{TEST}}\rvert$	F	L	ℓ_c
Arts [Ueda and Saito, 2003]	3712	3772	23146	26	1.65
Bibtex [Katakis et al., 2008]	4880	2515	1836	159	2.40
Birds [Briggs et al., 2013]	322	323	260	19	1.01
Bookmarks [Katakis et al., 2008]	70000	17856	2150	208	2.04
Business [Ueda and Saito, 2003]	5710	5504	21924	30	1.60
Computers [Ueda and Saito, 2003]	6270	6174	34096	33	1.51
Corel5k [Duygulu et al., 2002]	4500	500	499	374	3.52
Delicious [Tsoumakas et al., 2008]	12920	3185	500	983	19.02
Education [Ueda and Saito, 2003]	6030	6000	27534	33	1.46
Emotions [Trochidis et al., 2008]	391	202	72	6	1.87
Enron [UC Berkeley, 2018]	1123	579	1001	53	3.38
Entertainment [Ueda and Saito, 2003]	6356	6374	32001	21	1.41
Genbase [Diplaris et al., 2005]	463	199	1185	27	1.25
Health [Ueda and Saito, 2003]	4557	4648	30605	32	1.64
Mediamill [Snoek et al., 2006]	30993	12914	120	101	4.38
Medical [Pestian et al., 2007]	645	333	1449	45	1.25
Recreation [Ueda and Saito, 2003]	6471	6357	30324	22	1.43
Reference [Ueda and Saito, 2003]	4027	4000	39679	33	1.17
Scene [Boutell et al., 2004]	1211	1196	294	6	1.07
Science [Ueda and Saito, 2003]	3214	3214	37187	40	1.45
Social [Ueda and Saito, 2003]	6037	6074	52350	39	1.28
Society [Ueda and Saito, 2003]	7273	7239	31802	27	1.67
TMC2007-500 [Srivastava and Zane-Ulman, 2005]	21519	7077	500	22	2.22
Yeast [Elisseeff and Weston, 2001]	1500	917	103	14	4.24

5.3 Evaluation Methodology

We adopted the evaluation methodology that has been previously used in MLC context [Reyes et al., 2015] and in the other types of structured output prediction [Petković et al., 2019].

We use the same train-test split of the datasets as in the MULAN repository `http://mulan.sourceforge.net/datasets-mlc.html`. A ranking is computed from the training part $\mathscr{D}_{\text{TRAIN}}$ only, and evaluated on the testing part $\mathscr{D}_{\text{TEST}}$.

The quality of the ranking is assessed by using the kNN algorithm in which as a distance measure the weighted Euclidean distance was used. For two input vectors $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, the distance between them is defined as

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sqrt{\sum_{i=1}^{F} w_i d_i^2(\boldsymbol{x}_i^1, \boldsymbol{x}_i^2)}, \tag{4}$$

where d_i is defined as the absolute difference of the feature values scaled to the $[0, 1]$-interval, if x_i is numeric, and as $\mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2]$ ($\mathbf{1}$ is the indicator function), if x_i is nominal. The weights are set to $w_i = \max\{\mathit{relevance}(x_i), 0\}$, since they need to be made non-negative to ensure that d is well defined, and also to ignore the attributes that have smaller values for importance than a randomly generated attribute would have.

The evaluation through a kNN predictive model was chosen because of three main reasons. First, this is a distance based model, hence, it can easily make use of the information contained in the feature importances in the learning phase. The second reason is kNN's simplicity: its only parameter is the number of neighbors and we set its value to 15 (based on initial experiments in [Petković et al., 2018] and the experiments presented in [Spyromitros et al., 2008]). In the prediction stage, the neighbors' contributions to the predicted value are equally weighted, so we do not introduce additional parameters that would influence the performance. The third reason for using kNN as an evaluation model is as follows. If a feature ranking is meaningful, then when the feature importances are used as weights in the calculation of the distances kNN should produce better predictions as compared to kNN without using these weights [Wettschereck, 1994].

5.4 Evaluation Measures

We evaluate our feature ranking scores using 15 standard MLC evaluation measures [Madjarov et al., 2012, Vens et al., 2008]. These are

- average area under precision-recall curve ($\overline{\text{AUPRC}}$), area under the average precision-recall curve ($\text{AU}\overline{\text{PRC}}$), average area under the ROC curve ($\overline{\text{AUROC}}$),
- micro precision, micro recall, micro F_1 measure,
- multi-label precision, multi-label recall, multi-label F_1 measure, multi-label accuracy,
- Hamming loss, subset accuracy,
- one error, coverage, ranking loss.

We avoided the use of macro-averaged measures and average precision since they are ill-defined in some cases that we also encountered in our evaluation of the results. For example, these cases concern datasets containing examples $(\boldsymbol{x}, \boldsymbol{y})$ whose label set $\boldsymbol{y}$ is empty, thus calculating average precision leads to $0/0$ expressions. Additionally, the problems might occur when a given label is never predicted as relevant. All in all, the datasets where such ill-defined cases occur are Business, Corel5k, Delicious, Genbase, Health, Medical, Reference (macro recall); Birds, Delicious and Mediamill (average precision); and all but Emotions and Scene (macro precision and macro F_1).

5.5 Statistical Analysis of the Results

For comparing the algorithms, we use the Friedman test. The null hypothesis H_0 is that all considered algorithms have the same performance. If it is rejected by the Friedman's test, we additionally apply Nemenyi or Bonferroni-Dunn post-hoc test. The first is used when we investigate where the statistically significant differences between *any* two algorithms occur, while the second is used when we are interested in the differences between one particular algorithm and the others. A detailed description of all tests is available in [Demšar, 2006].

The results of the Nemenyi and Bonferroni-Dunn tests are presented on critical distance diagrams. Each diagram shows the average rank of the algorithm over the considered datasets. Additionally, the groups of algorithms among which no statistically significant differences occur are connected with a line.

Before proceeding with the statistical analysis, we round the performances to three decimal points. In the analysis, the significance level was set to $\alpha = 0.05$.

Sometimes, it is necessary to control the false discovery rate. We do this with the Benjamini-Hochberg procedure [Benjamini and Hochberg, 1995].

5.6 Parameter instantiation

We set the number of features that are considered at each node to $f = \lceil \sqrt{F} \rceil$ [Kocev et al., 2013b] when building a *Random forest*, and to $f = F$ when building an ensemble of *Extra trees* [Kocev and Ceci, 2015]. The remaining parameter that completely defines tree induction is the minimal number of instances in a leaf which was set to 2. We build ensembles of sizes $|\mathcal{E}| \in \{10, 25, 50, 75, 100, 150, 200, 250\}$. The proposed feature ranking scores themselves are parameter-less, except for the `Symbolic` ranking where the value of the parameter α is set to 0.5 as recommended in [Petković et al., 2019].

We set the parameters of the MLC-Relief algorithm to their recommended values [Petković et al., 2018]: the number of iterations m is set to $|\mathscr{D}_{\text{TRAIN}}|/4$, the number of the Relief neighbours K is set to $K = 15$, and the distance between the label sets $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ in the target space is set to the $\boldsymbol{F_1}$ distance

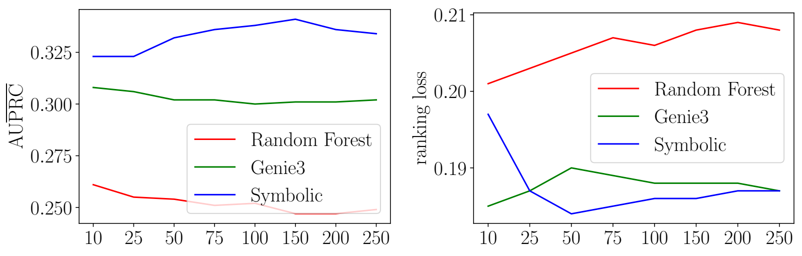
$$d_{F1}(\boldsymbol{y}^1, \boldsymbol{y}^2) = 1 - 2|\boldsymbol{y}^1 \cap \boldsymbol{y}^2| \; / \; (|\boldsymbol{y}^1| + |\boldsymbol{y}^2|).$$

6 Results and Discussion

Due to the large number of datasets and evaluation measures, we will use the *Arts* dataset and the evaluation measures AU$\overline{\text{PRC}}$ and *Ranking Loss* as a running example, for which we show the detailed graphs. The results for the other datasets and measures will be shown in an aggregated form. Their extended version is available at `https://github.com/Petkomat/mlc-ranking`.

6.1 How Many Trees?

For each dataset, ranking-ensemble pair and evaluation measure M, we can define the curves that consist of the points $(e, M(e))$ where e ranges over the possible values of ensemble sizes given in Sec. 5.6. The curves for the *Arts* dataset that belong to feature rankings computed from the *Bagging* ensemble and evaluated in terms of AU$\overline{\text{PRC}}$ and *Ranking Loss* are given in Fig. 1. Since AU$\overline{\text{PRC}}$ should be maximized and *Ranking Loss* minimized, `Symbolic` score performs best in both cases (except for the 10-trees rankings, evaluated with *Ranking Loss*), and `Random forest` score performs worst. Furthermore, it is expected that more trees contribute to a more stable and better feature relevance estimates (the variances of the *relevance* estimates (x_i) is inversely proportional to the number of trees in an ensemble), and the curves are not monotonic and not very steep (except for the `Symbolic` ranking). Consequently, the graphs hint that a small number of trees suffices to achieve the saturation point of the ranking quality. Essentially, we are looking to find the minimal number of trees that need to be included into the ensemble so that the performance of the learned feature rankings is not statistically significantly different compared to the best feature ranking.

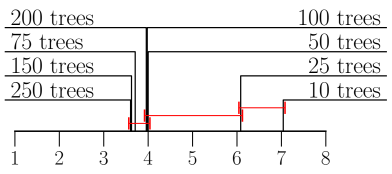


(a) AU$\overline{\text{PRC}}$ (b) *Ranking Loss*

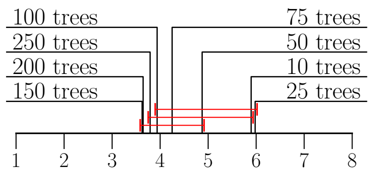
Fig. 1: Quality of the feature rankings computed from the three scores and *Bagging* ensembles, on the *Arts* dataset when the ensemble sizes varies. The quality is measured in terms of AU$\overline{\text{PRC}}$ (a) and *Ranking Loss* (b).

To further investigate that, we fix the ranking-ensemble pair and evaluation measure while the ensemble size varies. We compare the differences among the corresponding rankings over all the datasets by applying Friedman statistical test. Since there are 8 ranking-ensemble pairs and 15 evaluation measures, this results in 120 tests. However, these tests are not independent, since some of them base on the same rankings. Thus, we control the false discovery rate by the Benjamini-Hochberg procedure [Benjamini and Hochberg, 1995]. The null hypothesis that all feature rankings have the same quality is rejected only in 21 cases. This essentially implies that having ensembles with as little as 10 base predictive models already yield good feature ranking. The results from the statistical tests are available at the repository `https://github.com/Petkomat/mlc-ranking`.

Difference in the feature rankings was discovered (i.e., the null hypothesis is rejected) for *i)* 14 out of 15 evaluation measures for the `Symbolic`-*Random forest* pair (the exception is one error), and *ii)* 7 out of 14 evaluation measures for the `Symbolic`-*Bagging* pair. This means that the other feature two feature ranking scores (`Genie3` and `Random forest`) achieve their saturation points much earlier, i.e., 10 trees in an ensemble suffice. The reason that `Symbolic` score needs more trees may be its simplicity, since the actual quality of the nodes that contain a given feature in their tests is ignored when computing this score.



(a) *Random forest*



(b) *Bagging*

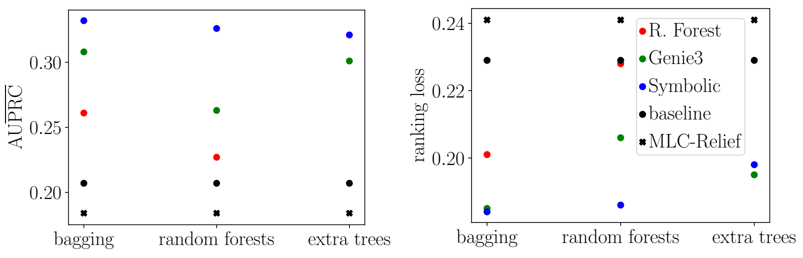
Fig. 2: Average rank diagrams for the `Symbolic` score computed from *Random forest* (a) and *Bagging* ensembles (b) and evaluated through the Nemenyi statistical test using multi-label F_1 as the evaluation measure.

We next use Nemenyi post-hoc test to discover the optimal ensemble size for the remaining two combinations of the `Symbolic` ranking where 10 trees in the ensemble do not suffice: the first with *Random forest* and the second with *Bagging* ensemble. The post-hoc test reveals the groups of the algorithms with statistically significantly different performance. The results of the tests are given in Figs. 2a (*Random forest*) and 2b (*Bagging*): These are obtained by evaluating the feature ranking with the multi-label F_1 measure, however, the same conclusions can be drawn by inspecting the other measures (available at the repository `https://github.com/Petkomat/mlc-ranking`). The best performing group of the algorithms (denoted by the left-most red line) is similar in both cases: It contains the feature rankings that were computed from at least 50 trees which means that 50 trees will suffice to achieve the optimal performance.

In sum, we select ensembles with 10 trees as the optimal setting for the majority of ranking-ensemble pairs, except for the `Symbolic`-*Random forest* and `Symbolic`-*Bagging* – for these two we learn ensembles with 50 trees. In the analysis of the results presented in the next sections, we will use these values. Note that the selected values are conservative for the performance of the proposed methods in terms of predictive performance, however, it is optimal in a sense of predictive performance - efficiency trade-off.

6.2 Are the Feature Rankings Relevant?

Now that we have fix ensemble size for every ranking-ensemble pair, we proceed to the experiments that will reveal whether the proposed feature rankings outperform the non-informed baseline ranking which awards every feature relevance 1. The detailed results are again given for the *Arts* dataset on which the rankings are evaluated with $\text{AU}\overline{\text{PRC}}$ and *Ranking Loss* (Figs. 3a and 3b respectively). In addition to the points that present ranking-ensemble pairs, the non-informed baseline ranking is presented with black circles. The column that belongs to the *Extra trees* have one point less since `Random forest` score cannot be computed from those ensembles. One can see that feature rankings always outperform the baseline for this dataset, however, the difference between the baseline and the `Random forest`-*Random forest* ranking as measured in terms of *Ranking Loss* is not practically significant.



(a) $\text{AU}\overline{\text{PRC}}$ (b) *Ranking Loss*

Fig. 3: Comparison of the proposed scores to the non-informed baseline and the state-of-the art baseline (MLC-Relief), in terms of $\text{AU}\overline{\text{PRC}}$ (a, higher is better) and *Ranking Loss* (b, lower is better) evaluation measures. The legend applies to both graphs.

To check whether the baseline is consistently outperformed, we fix an evaluation measure and perform 8 pairwise comparisons of rankings to the baseline. At the end, we apply Benjamini-Hochberg correction of the p-values and obtain the following results.

For the measures Hamming Loss, One Error and Micro Precision, no statistically significant differences were found. As for the remaining 12 evaluation measures, 7 out of 8 feature rankings outperform the baseline - the exception is the `Symbolic` feature score, computed from *Extra trees* ensembles. This means that every score results in relevant rankings, provided it is computed from an appropriate ensemble of trees.

6.3 Most Appropriate Ensemble Methods

To find the most appropriate ensemble method for a given feature ranking score, we fix the feature ranking score and vary the ensemble method from which this score is computed. We use the Friedman test in the case of the `Symbolic` and `Genie3` scores since they can be computed from three different ensembles, and Wilcoxon test for `Random forest` score. This results in 45 comparisons, one for each score and measure.

The differences are typically not statistically significant (in 30 cases). The `Symbolic` score is the only one for which the differences are mostly significant. However, for the `Symbolic` score, the post-hoc Nemenyi tests do not select a single ensemble method as the best performing. More specifically, *Extra trees* have the worst average ranks according to two evaluation measures: *multi-label precision* (Fig. 4a) and *Ranking Loss* (Fig. 4b). The other evaluation measures did not yield statistically significant differences across the different ensemble types.

The differences among the different ensemble methods in the case of the `Genie3` ranking are statistically significant for micro-averaged F_1 and $\mathrm{AU}\overline{\mathrm{PRC}}$. For both evaluation measures, the Nemenyi post-hoc test reveals that the *Random forest* ensembles have the worst average rank, hence, they should be avoided in combination with `Genie3`.

In the case of the `Random forest` score, the differences are statistically significant for the three area under curves measures and *Ranking Loss*: Wilcoxon test reveals that *Bagging* should be preferred over the *Random forest* ensemble.

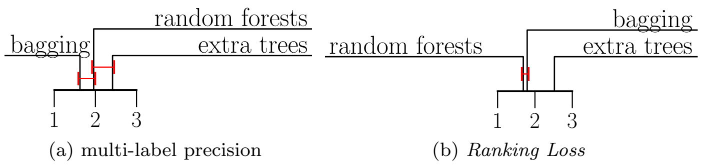


Fig. 4: The average ranks of `Symbolic` feature rankings computed from different ensemble methods. The statistical significance of differences is assessed by the Nemenyi test. The quality of the ranking is measured in terms of multi-label precision (a) and *Ranking Loss* (b).

We next investigate the results where the differences are not statistically significant. We summarize these results (in Tab. 3) by providing the ensemble method with the best average rank for each feature ranking score and for each evaluation measure. The results reveal that the ensemble method that should be used as a default choice are *Extra trees* for the `Genie3` score, *Bagging* for the `Random forest` score, and *Random forest* for the `Symbolic` score.

6.4 Are Feature Rankings State-of-the-art?

To answer this question, we compare the proposed feature ranking scores (computed with the most appropriate ensemble method) to the MLC-Relief baseline. The results for the *Arts* dataset are shown in Fig. 3 where in addition to the colored circles that belong to the proposed scores, black crosses that belong to the MLC-Relief score are shown. We can see that the proposed scores outperform the MLC-Relief score on this dataset.

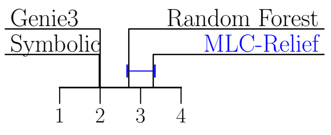
To check whether this is valid in general, we use the Friedman statistical test. Results show that the null hypothesis is rejected for all 15 evaluation measures, so we proceed to the post-hoc test, but now, instead of Nemenyi one, Bonferroni-Dunn test is applied since we are only interested in the comparisons to MLC-Relief.

We show the detailed results for the measures $\mathrm{AU}\overline{\mathrm{PRC}}$ (Fig.5a) and *Ranking Loss* (Fig. 5b). These results show that `Random forest` ranking is not statistically significantly different from the MLC-Relief (according to $\mathrm{AU}\overline{\mathrm{PRC}}$) or MLC-Relief

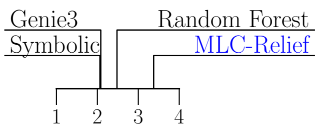
Table 3: The type of ensemble for which the produced ranking has the best average rank, for each feature ranking score, and each evaluation measure. The abbreviation ML in the evaluation measure names stand for multi-label.

	`Genie3`	`Random forest`	`Symbolic`
Hamming loss	*Extra trees*	*Bagging*	*Random forest*
ranking loss	*Extra trees*	*Bagging*	*Random forest*
one error	*Extra trees*	*Bagging*	*Random forest*
coverage	*Extra trees*	*Bagging*	*Random forest*
ML accuracy	*Extra trees*	*Bagging*	*Bagging*
ML precision	*Extra trees*	*Bagging*	*Bagging*
ML recall	*Extra trees*	*Bagging*	*Bagging*
ML F_1 Measure	*Extra trees*	*Bagging*	*Bagging*
subset accuracy	*Bagging*	*Bagging*	*Bagging*
micro precision	*Random forest*	*Bagging*	*Random forest*
micro recall	*Extra trees*	*Bagging*	*Bagging*
micro F_1	*Extra trees*	*Bagging*	*Bagging*
$\overline{\text{AUROC}}$	*Extra trees*	*Bagging*	*Random forest*
$\overline{\text{AUPRC}}$	*Bagging*	*Bagging*	*Random forest*
AU$\overline{\text{PRC}}$	*Extra trees*	*Bagging*	*Random forest*

is statistically significantly worse that all proposed algorithms (according to *Ranking Loss*).



(a) AU$\overline{\text{PRC}}$



(b) *Ranking Loss*

Fig. 5: Average rank diagrams of the proposed ensemble-based scores, compared to the MLC-Relief baseline. Statistically significant differences are assessed by the Bonferroni-Dunn test. The quality of the rankings is measures in terms of AU$\overline{\text{PRC}}$ (a) and *Ranking Loss* (b).

For the other measures, we show only the average ranks diagrams in the form of radar plots where more than one measure can be shown in one graph, see Figs. 6a-6d. The closer the algorithm is to the center of the graph, i.e., rank 1, the better. We can see that MLC-Relief is mostly ranked last, except for the multi-label measures in Fig. 6d, micro recall and F_1 measure and subset accuracy when it is ranked second but last.

Thus, we conclude that our methods are state of the art. Not only that ensemble-based methods have always the best two average ranks. Their computational complexity is also better than the computational complexity of MLC-Relief (recall the time complexities of the proposed methods from Sec. 4.1). The time complexity of MLC-Relief is $\mathcal{O}(Fn^2 + nKL)$ where, additionally, K is the number of MLC-Relief neighbors. This means that MLC-Relief is quadratic in the number of examples, which might be prohibitively large, for bigger data sets.

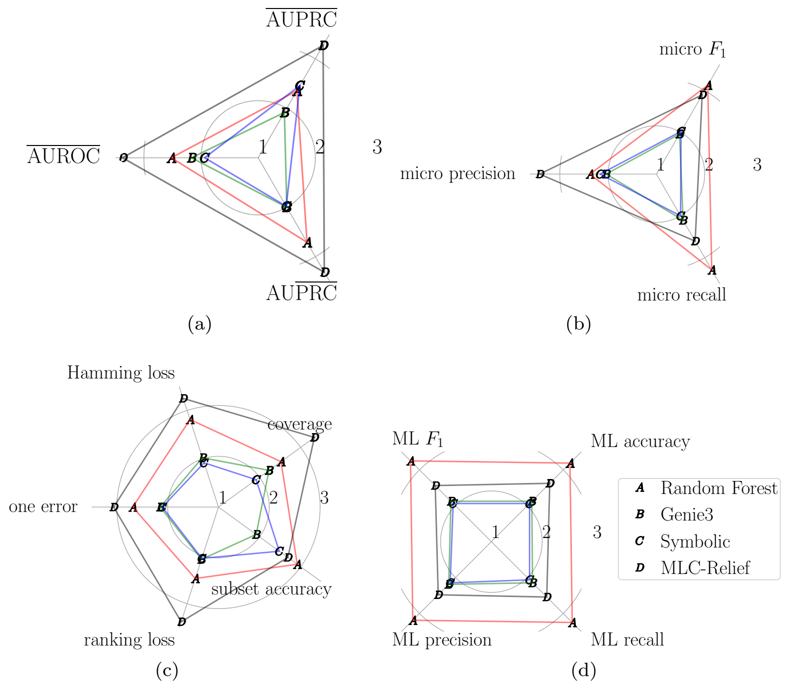
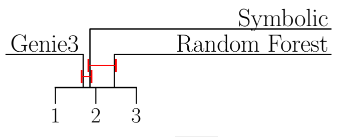


Fig. 6: The average ranks of the ensemble-based scores, compared to MLC-Relief, for all 15 evaluation measures. The legend in (d) applies to all subfigures.

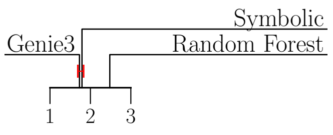
6.5 Which Feature Ranking Score is the Best?

To see which of the proposed ranking scores is the best, we compare the `Symbolic`, `Genie3` and `Random forest` scores, for each of the evaluation measures, using the Friedman test. In 9 cases, the differences among them are statistically significant, therefore we can proceed to the Nemenyi post-hoc test. The average diagrams for two of them, are presented in Figs. 7a ($\text{AU}\overline{\text{PRC}}$) and 7b (*multi-label precision*). In both figures, the order of the scores is the same: `Genie3`, `Symbolic`, `Random forest`. Based on the results from $\text{AU}\overline{\text{PRC}}$, the difference between `Genie3` and `Random forest` is statistically significantly different, while based on *multi-label precision*, the difference between `Random forest` and the other two ranking scores is statistically significantly different.

We can conclude that `Random forest` does not give the best rankings. Since `Genie3` and `Symbolic` score typically result in the rankings of (approximately) the same quality, the final decision is made based on the time complexity: since we need induce 50 trees to obtain the best performance with `Symbolic` score, and only 10 to achieve that with `Genie3` score, we identify `Genie3` feature ranking score as the one that should be preferred.



(a) AU$\overline{\text{PRC}}$



(b) multi-label precision

Fig. 7: The average rank diagrams of the proposed scores. Statistically significant differences are assessed by the Nemenyi post-hoc test. The quality of the rankings is measured in terms of AU$\overline{\text{PRC}}$ (a) and multi-label precision (b).

7 Conclusions

In this paper, we propose three ensemble-based feature ranking scores for multi-label classification (MLC): `Symbolic`, `Genie3` and `Random forest`. We compute each of them from three different ensembles of predictive clustering trees: *Bagging*, *Random forest* and *Extra trees* (with the exception that the `Random forest` score cannot be computed from *Extra trees* ensemble).

We extensively evaluate the proposed scores on 24 benchmark MLC problems, using 15 standard MLC evaluation measures. For each score and ensemble method, we first determine the saturation point in terms of the ensemble size, after which the quality of the ranking does not improve any further. We find out that 10 trees usually suffice, whereas the `Symbolic` score needs *Bagging* and *Random forest* ensembles of size 50.

Next, we show that the proposed feature ranking scores yield relevant feature rankings, by comparing them to the non-informed ranking which values all the features the same. After that, we fist determine the most appropriate ensemble method for every feature ranking score and empirically prove that the proposed feature ranking scores outperform current state-of-the-art methods in the quality of the rankings (for the majority of the evaluation measures), and in time efficiency. Finally, we determine which of the proposed feature ranking scores is the best. Taking into account the quality of the rankings first and - in case of the ties - time efficiency, we find that the `Genie3` feature ranking score is the optimal one.

Future work can follow two major directions. First, since the `Random forest` score typically yields the worst feature rankings, it could be additionally analyzed and parameterized with MLC error measures in Eq. (3) to obtain better performance. Second, other ensemble techniques may be evaluated, such as gradient boosting, which could be even more efficient.

References

[Benjamini and Hochberg, 1995] Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300.

[Blockeel, 1998] Blockeel, H. (1998). *Top-down Induction of First Order Logical Decision Trees.* PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium.

[Boutell et al., 2004] Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Breiman et al., 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. J. (1984). *Classification and Regression Trees.* Chapman & Hall/CRC.

[Briggs et al., 2013] Briggs, F., Huang, Y., Raich, R., Eftaxias, K., Lei, Z., Cukierski, W., Frey Hadley, S., Hadley, A., Betts, M., Fern, X. Z., Irvine, J., Neal, L., Thomas, A., Fodor, G., Tsoumakas, G., Ng Hong, W., Nguyen, T. N. T., Huttunen, H., Ruusuvuori, P., Manninen, T., Diment, A., Virtanen, T., Marzat, J., Defretin, J., Callender, D., Hurlburt, C., Larrey, K., and Milakov, M. (2013). The 9th annual mlsp competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2013*, pages 1–8.

[Dembczyński et al., 2012] Dembczyński, K., Waegeman, W., Cheng, W., and Hüllermeier, E. (2012). On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1):5–45.

[Demšar, 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

[Diplaris et al., 2005] Diplaris, S., Tsoumakas, G., Mitkas, P., and Vlahavas, I. (2005). Protein classification with multiple algorithms. In *10th Panhellenic Conference on Informatics (PCI 2005)*, pages 448–456.

[Duygulu et al., 2002] Duygulu, P., Barnard, K., de Freitas, J. F. G., and Forsyth, D. A. (2002). Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In Heyden, A., Sparr, G., Nielsen, M., and Johansen, P., editors, *Computer Vision — ECCV 2002*, pages 97–112, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Elisseeff and Weston, 2001] Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*. Springer International Publishing.

[Elkafrawy et al., 2015] Elkafrawy, P., Mausad, A., and Esmail, H. (2015). Experimental comparison of methods for multi-label classification in different application domains. *International Journal of Computer Applications*, 114:1–9.

[Geurts et al., 2006] Geurts, P., Erns, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 36(1):3–42.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

[Hancock et al., 1996] Hancock, T., Jiang, T., Li, M., and Tromp, J. (1996). Lower bounds on learning decision lists and trees. *Information and Computation*, 126(2):114–122.

[Hansen and Salamon, 1990] Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001.

[Huynh-Thu et al., 2010] Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS One*, 5(9):1–10.

[Katakis et al., 2008] Katakis, I., Tsoumakas, G., and Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge.*

[Kocev and Ceci, 2015] Kocev, D. and Ceci, M. (2015). Ensembles of extremely randomized trees for multi-target regression. In *Discovery Science: 18th International Conference (DS 2015)*, volume 9356 of *LNCS*, pages 86–100.

[Kocev et al., 2013a] Kocev, D., Slavkov, I., and Džeroski, S. (2013a). Feature ranking for multi-label classification using predictive clustering trees. In *Solving Complex Machine Learning Problems with Ensemble Methods.*

[Kocev et al., 2013b] Kocev, D., Vens, C., Struyf, J., and Džeroski, S. (2013b). Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3):817–833.

[Kong et al., 2012] Kong, D., Ding, C., Huang, H., and Zhao, H. (2012). Multi-Label ReliefF and F-statistic Feature Selections for Image Annotation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2352–2359.

[Kononenko and Robnik-Šikonja, 2003] Kononenko, I. and Robnik-Šikonja, M. (2003). Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning Journal*, 55:23–69.

[Lee and Kim, 2017] Lee, J. and Kim, D.-W. (2017). Scls: Multi-label feature selection based on scalable criterion for large label set. *Pattern Recognition*, 66:342 – 352.

[Madjarov et al., 2012] Madjarov, G., Kocev, D., Gjorgjevikj, D., and Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45:3084–3104.

[Pereira et al., 2015] Pereira, R. B., Plastino, A., Zadrozny, B., and Merschmann, L. H. C. (2015). Information gain feature selection for multi-label classification. *Journal of Information and Data Management*, 6(1):48–58.

[Pestian et al., 2007] Pestian, J. P., Brew, C., Matykiewicz, P., Hovermale, D. J., Johnson, N., Bretonnel Cohen, K., and Duch, W. (2007). A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing (BioNLP '07)*, pages 97–104.

[Petković et al., 2019] Petković, M., Kocev, D., and Džerovski, S. (2019). Feature ranking for multi-target regression. *Machine Learning Journal.* https://doi.org/10.1007/s10994-019-05829-8.

[Petković et al., 2018] Petković, M., Kocev, D., and Džeroski, S. (2018). Feature ranking with relief for multi-label classification: Does distance matter? In Soldatova, L., Vanschoren, J., Papadopoulos, G., and Ceci, M., editors, *Discovery Science*, pages 51–65. Springer International Publishing.

[Reyes et al., 2015] Reyes, O., Morell, C., and Ventura, S. (2015). Scalable extensions of the ReliefF algorithm for weighting and selecting features on the multi-label learning context. *Neurocomputing*, 161:168 – 182.

[Sechidis et al., 2014] Sechidis, K., Nikolaou, N., and Brown, G. (2014). Information theoretic feature selection in multi-label data through composite likelihood. In Fränti, P., Brown, G., Loog, M., Escolano, F., and Pelillo, M., editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 143–152, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Snoek et al., 2006] Snoek, C. G. M., Worring, M., van Gemert, J. C., Geusebroek, J.-M., and Smeulders, A. W. M. (2006). The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th ACM International Conference on Multimedia*, pages 421–430, New York, NY, USA. ACM.

[Spolaôr et al., 2013] Spolaôr, N., Cherman, E. A., Monard, M. C., and Lee, H. D. (2013). A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135–151.

[Spolaôr and Tsoumakas, 2013] Spolaôr, N. and Tsoumakas, G. (2013). Evaluating feature selection methods for multi-label text classification. In *BioASQ Workshop*, pages 1–12.

[Spyromitros et al., 2008] Spyromitros, E., Tsoumakas, G., and Vlahavas, I. (2008). An empirical study of lazy multilabel classification algorithms. In *Artificial Intelligence: Theories, Models and Applications, LNAI 5138*, pages 401–406. Springer Berlin Heidelberg.

[Srivastava and Zane-Ulman, 2005] Srivastava, A. N. and Zane-Ulman, B. (2005). Discovering recurring anomalies in text reports regarding complex space systems. In *2005 IEEE Aerospace Conference.*

[Trochidis et al., 2008] Trochidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. (2008). Multilabel classification of music into emotions. In *2008 International Conference on Music Information Retrieval (ISMIR 2008)*, pages 325–330.

[Tsoumakas et al., 2008] Tsoumakas, G., Katakis, I., and Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08).*

[Tsoumakas and Vlahavas, 2007] Tsoumakas, G. and Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In Kok, J. N., Koronacki, J., Mantaras, R. L. d., Matwin, S., Mladenič, D., and Skowron, A., editors, *Machine Learning: ECML 2007*, pages 406–417, Berlin, Heidelberg. Springer Berlin Heidelberg.

[UC Berkeley, 2018] UC Berkeley (2018). Enron Email Analysis Project. `http://bailando.sims.berkeley.edu/enron_email.html`. Accessed: 2018-06-28.

[Ueda and Saito, 2003] Ueda, N. and Saito, K. (2003). Parametric mixture models for multi-labeled text. In *Advances in Neural Information Processing Systems 15*, pages 721–728. MIT Press.

[Vens et al., 2008] Vens, C., Struyf, J., Schietgat, L., Džeroski, S., and Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214.

[Wettschereck, 1994] Wettschereck, D. (1994). *A study of distance based algorithms.* PhD thesis, Oregon State University, USA.

Acta Polytechnica Hungarica Vol. ??, No. ?, 20??

Feature Ranking for Hierarchical Multi-Label Classification with Tree Ensemble Methods

Matej Petković[1, 2], Sašo Džeroski[1, 2], Dragi Kocev[1, 2]

[1]Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
[2]Jožef Stefan Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia
matej.pektovic@ijs.si, saso.dzeroski@ijs.si, dragi.kocev@ijs.si

In this work, we address the task of feature ranking for hierarchical multi-label classification (HMLC). The task of HMLC concerns problems with multiple binary variables organized into a hierarchy of target attributes. The goal is to learn a model for predicting all of them simultaneously. This task is receiving an increasing attention from the research community, due to its wide application potential in text document classification and functional genomics. Here, we propose a group of feature ranking methods based on three established ensemble methods of predictive clustering trees: Bagging, Random Forests and Extra Trees. Predictive clustering trees are a generalization of decision trees towards predicting structured outputs. Furthermore, we propose to use three scoring functions for calculating the feature importances: Symbolic, Genie3 and Random Forest. We test the proposed methods on 30 benchmark HMLC datasets, show that Symbolic and Genie3 scores yield relevant rankings, that all three scores outperform the HMLC-Relief ranking method, and are computed very time-efficiently. For each scoring function, we find the most appropriate ensemble method and compare the scores among each other to find the best one.

Keywords: hierarchical multi-label classification; feature ranking; ensemble methods; Relief

1 Introduction

Classification is a task in predictive modelling, where we learn a model that takes a vector $\boldsymbol{x}$ of descriptive variables (features) x_i as the input, and predicts the class value y that a given example belongs to. If y can take two different values, the task at hand is referred to as binary classification. Otherwise (y can take more than two values), the task at hand is multi-class classification. In both cases, every example is assigned precisely one value. For example, one can predict whether a person is sick, where $y \in \{\texttt{yes}, \texttt{no}\}$ (binary), or what is the blood type of a person where $y \in \{\texttt{A}, \texttt{B}, \texttt{AB}, \texttt{0}\}$ (multi-class). In both cases, class values are mutually exclusive. A related task is multi-label classification (MLC). As opposed to the standard classification, a MLC predictive model predicts which labels ℓ from a predefined set $\mathscr{L}$ are *relevant* for a given example. For example, one can predict which of the genres from the set $\mathscr{L} = \{\texttt{romance}, \texttt{drama}, \texttt{comedy}\}$ are relevant for a given film. Clearly, a film can be `drama` and `comedy` at the same time.

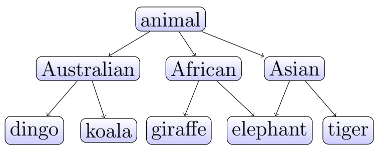


Figure 1
An exemplary hierarchy of animals.

Hierarchical MLC (HMLC) is a generalization of MLC, where the labels are organized into a hierarchy that is given as a binary relation $\prec$, which partially orders the set $\mathscr{L}$. If $\ell_1 \prec \ell_2$, we say that ℓ_1 is a predecessor of ℓ_2. This relation imposes the *hierarchical constraint*: If ℓ is relevant for a given example then all the predecessor labels are also relevant for the example. Fig. 1 shows a toy hierarchy of groups of animals. For example, `animal` $\prec \ell$ for all labels $\ell \in \mathscr{L}$, `Asian` $\prec$ `tiger` etc. The label `elephant` has two parents (`African` and `Asian`) and one additional predecessor (`animal`). Thus, if an example is an `elephant`, then it is also an `African`, `Asian` and an `animal`. Note that hierarchy may result in a tree where every label has at most one parent, or it may result in a general directed acyclic graph (DAG) where the number of parents can be higher, which is the case in the toy hierarchy. In this paper, DAG refers only to the hierarchies that are not also tree-shaped.

A possible approach to HMLC is to ignore the hierarchy at the learning phase and use any of the MLC methods, such as binary relevance or power set approach [24]. Binary relevance is a simple method that converts a MLC task to several binary classification tasks with $y \in \{\texttt{yes}, \texttt{no}\}$ where we predict the relevance of each label separately. This approach is often criticized for it cannot make use of the interactions among the labels. In the label power set approach, the task of predicting a subset of $\mathscr{L}$ is converted to the task of predicting an element of the power set $2^{\mathscr{L}}$, and thus converting a MLC task to multi-class classification task. However, the number of classes can be as high as $2^{|\mathscr{L}|}$, which results in a very sparse dataset. At prediction stage, predecessors of the labels predicted as relevant, must be added to the set of relevant labels, so that the hierarchical constraint is met. A similar two-step approach is possible [1], where support vector machines are learned for each class separately, and then combined using a Bayesian network model so that the predictions meet the hierarchical constraint. However, method adaptation techniques where an existing method is adapted to a new problem may be more appropriate. This can be done with predictive clustering trees (PCTs), which were shown to outperform their basic versions that follow the binary relevance approach [25].

In this paper, we do not address the task of building predictive models for HMLC. Rather, we propose a feature ranking method that is useful in this context. Feature ranking is another important task in machine learning, where the goal is to assess the importance of every descriptive attribute (feature) by using some scoring function. The output of a feature ranking algorithm is a list of features that is sorted with respect to the scores.

Feature ranking is typically considered a part of data preprocessing, since it can be used to reduce the dimensionality of the input space, so that only the features that contain the most information about labels (or target(s) in general) are kept in the dataset. By doing this, we decrease the computational cost of building a predictive model, while the performance of the model is not degraded. Another reason to compute a feature ranking is that dimensionality reduction typically results in models that are easier to understand, which is useful when a machine learning expert works in collaboration with a domain expert. Predictive models, such as decision trees, are easier to interpret when a small number of the most relevant features are used to learn them.

There is a plethora of feature ranking methods for the task of classification [23]. A possible approach to MLC feature ranking is to adapt the binary relevance approach from predictive modelling, where at the first stage, feature importances are computed for every label $\ell \in \mathscr{L}$ separately as in the classification case. After that, the feature importances are averaged over the different labels and a single ranking is returned. However, the landscape of methods for feature ranking for HMLC is not well populated due to the complexity of the task.

In this work, we propose a group of novel feature ranking approaches for HMLC. They base on the scoring functions Symbolic, Genie3 [13] and Random Forest [4], coupled with Bagging, Random Forests and Extra Trees ensembles of PCTs for HMLC [16, 21]. We compare them to two baselines and show that the proposed scoring functions outperform i) the non-informed ranking where all features have the same importance, and ii) the adaptation of Relief algorithm to HMLC [22].

The rest of the paper is organized as follows. In Sec. 2, we first describe predictive clustering trees, ensembles thereof and the three proposed feature ranking scores. Then, we proceed to the HMLC-Relief description. In Sec. 3, the experimental design is described in detail. In Sec. 4, the results are presented. We conclude with the Conclusions.

2 Methods

We first present the PCT-ensemble-based feature rankings and then proceed to the Relief ranking. Both PCT framework and the Relief family of algorithms is implemented in the CLUS system available at `http://source.ijs.si/ktclus/clus-public`.

2.1 Predictive Clustering Trees and Ensembles Thereof

PCTs generalize decision trees and can be used for a variety of learning tasks, including clustering and different types of structured output prediction tasks, e.g., multi-target regression, multi-label classification, hierarchical multi-label classification, time series prediction etc. [3, 16]. PCTs are induced with the standard top-down induction of decision trees algorithm [5] presented in Alg. 1. It takes as input a set of examples E and outputs a tree. The heuristic h that is used for selecting the tests is the weighted impurity of the subgroups of the instances, induced by the tests (line 4 of the Alg. 2). By minimizing it, the algorithm is guided towards small trees with good predictive performance. If there are no candidate tests, a leaf is created and the prototype of the instances belonging to that leaf is computed.

Algorithm 1 PCT(E)

```
1: $(t^*, h^*, \mathscr{P}^*) = \mathrm{BestTest}(E)$
2: if $t^* \neq none$ then
3:     for each $E_i \in \mathscr{P}^*$ do
4:         $tree_i = \mathrm{PCT}(E_i)$
5:     return node$(t^*, \bigcup_i \{tree_i\})$
6: else
7:     return leaf(Prototype$(E)$)
```

Algorithm 2 BestTest(E)

```
1: $(t^*, h^*, \mathscr{P}^*) = (none, 0, \emptyset)$
2: for each candidate test $t$ do
3:     $\mathscr{P}$ = partition induced by $t$ on $E$
4:     $h = |E| impu(E) - \sum_{E_i \in \mathscr{P}} |E_i| impu(E_i)$
5:     if $h < h^*$ then
6:         $(t^*, h^*, \mathscr{P}^*) = (t, h, \mathscr{P})$
7: return $(t^*, h^*, \mathscr{P}^*)$
```

In the HMLC case, the impurity function is defined as follows. First, the label subset $S \subseteq \mathscr{L}$ is converted into $0/1$-vector $\boldsymbol{s}$ of length $|\mathscr{L}|$, where $\boldsymbol{s}_j = 1 \Leftrightarrow \ell_j \in S$. We denote the variance of $\boldsymbol{s}_j$ over subset of examples $E \subseteq \mathscr{D}_{\mathrm{TRAIN}}$ as $Var_j(E)$. Additionally, each label ℓ_j is assigned a weight w_j that is defined as $w_j = \alpha\, \overline{w}(Parents(\ell_j))$ if the set of parents $Parents(\ell_j)$ is not empty and $w_j = 1$ otherwise (in the root(s) of the hierarchy). The function $\overline{w}(P)$ returns the average weight of the label set P, and the parameter $\alpha \in (0,1)$ is user-defined. Finally, we define the impurity function as

$$impu(E) = \frac{1}{|\mathscr{L}|} \sum_{j=1}^{|\mathscr{L}|} w_j Var_j(E),$$

hence the labels ℓ_j that are closer to the root of the hierarchy have bigger influence on the heuristic function. In a leaf L, the prototype function returns a vector whose j-th component equals the average value of $\boldsymbol{s}_j$ of the examples belonging to L.

To calculate feature importance scores (i.e., feature rankings), we grow ensembles of PCTs instead of growing a single one. An ensemble is a set of base predictive models, whose prediction for each new example is made by combining the predictions of the models from the ensemble. In HMLC tasks, this is typically achieved by taking the average of the base-model predictions. In our experiments, we used the following three approaches.

Random Forests, Bagging. In the Random Forests ensemble, instead of being learned from the original dataset $\mathscr{D}_{\mathrm{TRAIN}}$, each tree in the ensemble is learned from a different bootstrap replicate $\mathscr{B}$ of the dataset $\mathscr{D}_{\mathrm{TRAIN}}$, called bag. Additionally, we chose a random subset S of features in every internal node of the tree, and consider only the tests that are yielded by the features in S when looking for the best test in Alg. 2. Typical size of the set S is of the order $\log F$ or $\sqrt{F}$, where F is the number features in $\mathscr{D}_{\mathrm{TRAIN}}$. If $|S| = F$, we obtain Bagging procedure.

Extra Trees. Each tree is being learned directly from $\mathscr{D}_{\mathrm{TRAIN}}$, but the candidate tests in each node are now extremely randomized. Again, we chose a random subset

S of features in every internal node of the tree, and consider only one randomly chosen test per chosen feature, when looking for the best test.

2.2 Ensemble Scores

Once we build an ensemble of PCTs, we can exploit the ensemble structure to compute the feature ranking in three different ways. In the following, we denote a tree as $\mathscr{T}$, whereas $\mathscr{N} \in \mathscr{T}$ denotes a node. Trees form a forest $\mathscr{F}$. Its size (the number of trees in the forest) is denoted as $|\mathscr{F}|$. The set of all internal nodes of a tree $\mathscr{T}$ in which the feature x_i appears as part of a test is denoted as $\mathscr{T}(x_i)$.

Symbolic Ranking. In the simplest version of the score, we would count how many times a given feature occurs in the tests in the internal nodes of the trees. Since the features that appear closer to the root are intuitively more important than those that appear deeper in the trees, we weight these occurrences by the number of examples $e(\mathscr{N})$ that reach a node $\mathscr{N}$, and define the importance of the feature x_i as

$$importance_{\text{SYMB}}(x_i) = \frac{1}{|\mathscr{F}|} \sum_{\mathscr{T} \in \mathscr{F}} \sum_{\mathscr{N} \in \mathscr{T}(x_i)} e(\mathscr{N}) / |\mathscr{D}_{\text{TRAIN}}|. \tag{1}$$

Symbolic ranking can be computed for all three ensemble methods that we use.

Genie3 Ranking. The main motivation for Genie3 ranking is that splitting the current subset $E \subseteq \mathscr{D}_{\text{TRAIN}}$, according to a test where an important attribute appears, should result in high impurity reduction. The Genie3 importance of the attribute x_i is thus defined as

$$importance_{\text{GENIE3}}(x_i) = \frac{1}{|\mathscr{F}|} \sum_{\mathscr{T} \in \mathscr{F}} \sum_{\mathscr{N} \in \mathscr{T}(x_i)} h^*(\mathscr{N}), \tag{2}$$

where $h^*(\mathscr{N})$ is the value of the variance reduction function described in the Alg. 2. Since h^* is proportional to $e(\mathscr{N}) = |E|$, greater emphasis is again put on the attributes higher in the tree, where $|E|$ is larger. Genie3 ranking can be computed for all three ensemble methods that we use.

Random Forest Ranking.[1] This feature ranking method tests how much does noise in a given feature decrease the predictive performance of the trees in the forest. The greater the performance degradation, the more important the feature is. This score uses the internal out-of-bag estimates of the error, therefore it cannot be used with ensembles of Extra Trees.

Once a tree $\mathscr{T}$ is grown, the algorithm evaluates the performance of the tree by using the corresponding $\text{OOB}_{\mathscr{T}}$ examples. This results in the predictive error $Err(\text{OOB}_{\mathscr{T}})$, where lower error value corresponds to better predictions. To assess the importance of the feature x_i for the tree $\mathscr{T}$, we randomly permute its values in the set $\text{OOB}_{\mathscr{T}}$

[1] To avoid confusion, we use the plural form (Random Forests) to refer to the ensemble method, and singular form to refer to feature ranking score (Random Forest).

and obtain the set $\mathrm{OOB}^i_{\mathscr{T}}$. Then, the error $Err(\mathrm{OOB}^i_{\mathscr{T}})$ is computed and the importance of the feature x_i for the tree $\mathscr{T}$ is defined as the relative increase of error after noising. The final Random Forest score of the feature is the average of these values over all trees in the forest, namely

$$importance_{\mathrm{RF}}(x_i) = \frac{1}{|\mathscr{F}|}\sum_{\mathscr{T}\in\mathscr{F}} \frac{Err(\mathrm{OOB}^i_{\mathscr{T}}) - Err(\mathrm{OOB}_{\mathscr{T}})}{Err(\mathrm{OOB}_{\mathscr{T}})}. \tag{3}$$

2.3 HMLC-Relief Ranking

Relief family of feature ranking algorithms calculates the feature importance scores by considering differences in the feature values between pairs of examples (an example and its nearest neighbors). More specifically, if the values of features of a pair of examples from the same class are different then the features' importance decreases. Conversely, if the feature values are different for examples from different classes then the features' importance increases.

The expected value of the importance $importance_{Relief}(x_i)$ in the Relief has a nice probability interpretation in the case when both the target y and x_i are nominal [17]: simplified to some extent, we have a relation

$$\mathbb{E}[importance_{Relief}(x_i)] = \frac{P_{\text{diffAttr, diffTarget}}}{P_{\text{diffTarget}}} - \frac{P_{\text{diffAttr}} - P_{\text{diffAttr, diffTarget}}}{1 - P_{\text{diffTarget}}}, \tag{4}$$

where we define the probabilities $P_{\texttt{ev}} = P(\texttt{ev})$ and $P_{\texttt{ev1, ev2}} = P(\texttt{ev1} \wedge \texttt{ev2})$ that base on the events `diff/sameAttr` (two instances have different/same value of x_i) and `diff/sameTarget` (two instances have different/same target value). The probabilities in Eq. (4) are modeled as the distances in the corresponding spaces: P_{diffAttr} is modeled by the distance d_i on the domain of feature x_i, $P_{\text{diffTarget}}$ is modeled by the distance $d_{\mathscr{L}}$ between two label subsets of $\mathscr{L}$, and $P_{\text{diffAttr, diffTarget}}$ is modeled as their product $d_i d_{\mathscr{L}}$. This enables the generalization not only to numeric attributes and targets, but also to more complex target types, such as hierarchies as described in [22]. However, it must be assured that the upper bound of all distances is 1, which was overlooked in [22]. There, they proceed as follows.

First, the distance on the whole descriptive domain $\mathscr{X}$ is defined via the distances d_i on the domains $\mathscr{X}_i$ of features x_i as

$$d_i(\boldsymbol{x}^1,\boldsymbol{x}^2) = \begin{cases} \mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2] & : \mathscr{X}_i \not\subseteq \mathbb{R} \\ \frac{|\boldsymbol{x}_i^1 - \boldsymbol{x}_i^2|}{\max_{\boldsymbol{x}} \boldsymbol{x}_i - \min_{\boldsymbol{x}} \boldsymbol{x}_i} & : \mathscr{X}_i \subseteq \mathbb{R} \end{cases} \qquad d_{\mathscr{X}}(\boldsymbol{x}^1,\boldsymbol{x}^2) = \frac{1}{F}\sum_{i=1}^{F} d_i(\boldsymbol{x}^1,\boldsymbol{x}^2) \tag{5}$$

where $\mathbf{1}$ is the indicator function with the values $\mathbf{1}[\texttt{true}] = 1$ and $\mathbf{1}[\texttt{false}] = 0$. The distance between two label sets S_1 and S_2 is defined as a weighted Euclidean distance $d_E(S_1,S_2) = \sqrt{\sum_{j=1}^{|\mathscr{L}|} w_j(\boldsymbol{s}_j^1 - \boldsymbol{s}_j^2)^2}$ between the corresponding $0/1$-vectors $\boldsymbol{s}^1$ and $\boldsymbol{s}^2$, where the vectors and the weights are defined as in Sec. 2.1. We correct this and define

$$\mu = 1/\max_{S,S'} d_E(S,S') \quad \text{and} \quad d_{\mathscr{L}}(S_1,S_2) = \mu\, d_E(S_1,S_2). \tag{6}$$

Algorithm 3 HMLC-RReliefF($\mathscr{D}_{\text{TRAIN}}$, m, K)

```
 1: $\boldsymbol{P}_{\text{diffAttr, diffTarget}}, \boldsymbol{P}_{\text{diffAttr}}$ = zero lists of length $F$
 2: $P_{\text{diffTarget}} = 0.0$
 3: for $\iota = 1, 2, \ldots, m$ do
 4:     $\boldsymbol{r}$ = random example from $\mathscr{D}$
 5:     $\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_K$ = $K$ nearest neighbors of $\boldsymbol{r}$
 6:     for $k = 1, 2, \ldots, K$ do
 7:         $P_{\text{diffTarget}} \mathrel{+}= \delta(\ell) d_{\mathscr{L}}(\boldsymbol{r}, \boldsymbol{n}_k)$
 8:         for $i = 1, 2, \ldots, F$ do
 9:             $\boldsymbol{P}_{\text{diffAttr}}[i] \mathrel{+}= \delta(\ell) d_i(\boldsymbol{r}, \boldsymbol{n}_k)$
10:             $\boldsymbol{P}_{\text{diffAttr, diffTarget}}[i] \mathrel{+}= \delta(\ell) d_i(\boldsymbol{r}, \boldsymbol{n}_k) d_{\mathscr{L}}(\boldsymbol{r}, \boldsymbol{n}_k)$
11: for $i = 1, 2, \ldots, F$ do
12:     $importance_{Relief}(x_i) = \frac{\boldsymbol{P}_{\text{diffAttr, diffTarget}}[i]}{P_{\text{diffTarget}}} - \frac{\boldsymbol{P}_{\text{diffAttr}}[i] - \boldsymbol{P}_{\text{diffAttr, diffTarget}}[i]}{1 - P_{\text{diffTarget}}}$
```

2.3.1 Algorithm Description

The iterative procedure of calculating the importances using the HMLC-Relief extension is outlined in Alg. 3. For each of the m iterations, we randomly select an example $\boldsymbol{r}$ from $\mathscr{D}_{\text{TRAIN}}$ (line 4) and find its K nearest neighbors (line 5) using the distance $d_{\mathscr{X}}$ from Eq. (5). After that, we use the neighbors to update the estimates of probabilities from Eq. (4), for all attributes (lines 8–10). The estimates are updated with the average of the distances between $\boldsymbol{r}$ and its neighbors. Here, the distance $d_{\mathscr{L}}$ is used. The weight $\delta(k) = 1/(mK)$ ensures that the computed importances (line 12) are between -1 and 1 when the algorithms finishes.

The default values of the parameters are set as follows. Typically, we iterate over the whole dataset, i.e., $m = |\mathscr{D}_{\text{TRAIN}}|$. By doing this, the estimates of probabilities are expected to be more accurate. The value of K is typically set small enough to capture the local structure in the data. In that way, we implicitly capture the interactions between features [17]. Previous experiments [17, 19] have shown that $K = 10$ or $K = 15$ give the best performance.

2.3.2 Efficient Distance Computation

The normalization factor in the numeric part of the definition of d_i in Eq. (5) is trivial to compute. However, this is not the case with the μ in Eq. (6), if we want to do that efficiently. For tree-shaped hierarchies, we developed an efficient algorithm for computing μ that bases on the fact that the distance-maximizing pair of the labels consists either of two leaves of the hierarchy or the root and a leaf. In both cases, at least one of the leaves in the maximizing pair is the label with the highest depth in the hierarchy. Knowing that, we can carefully design the $\mathscr{O}(|\mathscr{L}|)$ recursive algorithm, which is the optimal time complexity.

If hierarchy is a general DAG we could not do considerably better than computing the maximizing pair by exhaustive search.

3 Experimental Design

3.1 Experimental Questions

The experiments were designed to answer the following experimental questions:

1. Given an ensemble feature ranking score, after which number of trees in the ensemble the quality of the ranking saturates?
2. Do the proposed ensemble feature ranking scores yield relevant rankings, i.e., can the additional information captured in the feature ranking boost the performance of the non-informed baseline classifier?
3. Do the proposed ensemble feature ranking scores outperform the improved version of HMLC-Relief algorithm?
4. Given an ensemble feature ranking score, which ensemble method is the most suitable?
5. Which ensemble feature ranking score yields the best rankings?

3.2 Datasets

We use 30 HMLC benchmark problems whose basic characteristics are summarized in Tab. 1. Most of the datasets have a few thousand of examples while the number of features could be as high as 74435. The label set typically contains a few hundred elements. Approximately one quarter of the hierarchies are DAGs. Many of the datasets are microarray data and come from the field of functional genomics. They describe the connection between description of proteins and their functional classes that are taken from Gene Ontology [7] (the corresponding hierarchies are DAGs), or the MIPS functional hierarchy[2] (the corresponding hierarchies are tree-shaped). Some other datasets are about text categorization of the processed news (`reuters`), patent classification according to the World International Patent Organization (`wipo`) etc.

3.3 Evaluation Procedure

In our experiments, we use the same train/test splits of the datasets as the original authors, for all the data sets. First, a feature ranking is computed from the training set $\mathscr{D}_{\text{TRAIN}}$. Its quality is assessed by the k-nearest neighbor (kNN) algorithm in which the weighted version of Euclidean distance is used instead of the standard one. It is defined as

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sqrt{\sum_{i=1}^{D} w_i d_i^2(\boldsymbol{x}_i^1, \boldsymbol{x}_i^2)}, \tag{7}$$

[2] `http://mips.helmholtz-muenchen.de/funcatDB/`

Table 1
Datasets' characteristics: training and test set size, number of features, label set size, and hierarchy type.

dataset	$\lvert\mathscr{D}_{\text{TRAIN}}\rvert$	$\lvert\mathscr{D}_{\text{TEST}}\rvert$	features	$\lvert\mathscr{L}\rvert$	hierarchy type
cellcycle-yeast-FUN [6]	2482	1284	77	751	tree
church-yeast-FUN [6]	2480	1284	27	751	tree
clef07a-is [9]	10000	1006	80	152	tree
derisi-yeast-FUN [6]	2455	1278	63	751	tree
diatoms [10]	726	372	200	81	tree
eisen-yeast-FUN [6]	1588	837	79	751	tree
enron-corr [14]	988	660	1001	67	tree
expr-yeast-FUN [6]	2494	1294	552	751	tree
exprindiv-ara-FUN [6]	2314	1182	1251	424	tree
exprindiv-ara-GO [6]	7161	3679	1251	627	DAG
gasch1-yeast-FUN [6]	2486	1287	173	751	tree
hom-ara-FUN [6]	2260	1213	72869	420	tree
hom-ara-GO [6]	7119	4002	72869	623	DAG
hom-yeast-FUN [6]	2549	1318	47034	751	tree
icpr2010 [11]	4913	2999	4000	76	tree
interpro-ara-FUN [6]	2455	1264	2815	427	tree
interpro-ara-GO [6]	7778	3985	2815	630	DAG
pheno-yeast-FUN [6]	1010	582	69	751	tree
reuters [18]	3000	3000	47236	143	tree
scop-ara-FUN [6]	2055	1042	2003	407	tree
scop-ara-GO [6]	6507	3336	2003	572	DAG
seq-ara-FUN [6]	2455	1264	4450	424	tree
seq-ara-GO [6]	7778	3985	4450	630	DAG
seq-yeast-FUN [6]	2590	1342	478	751	tree
spo-yeast-FUN [6]	2442	1269	80	751	tree
struc-ara-FUN [25]	2455	1264	14804	427	tree
struc-ara-GO [25]	7778	3985	14804	630	DAG
struc-yeast-FUN [25]	2535	1316	19628	751	tree
wipo [20]	1352	358	74435	528	tree
yeast-GO [1]	2310	1155	5930	133	DAG

where $\boldsymbol{x}^{1,2}$ are the input vectors of nominal/numeric feature values and d_i is defined by Eq. (5). Since the rankings that base on the Random Forest score and HMLC-Relief could contain negative relevance scores, which in both cases means that the feature is more irrelevant that a random feature would be, the weights are defined as $w_i = \max\{0, importance(x_i)\}$.

This evaluation procedure was chosen because kNN classifier is a distance-based model that can directly make use of feature importances, learned in the first phase of procedure. The second reason for our choice was kNN's simplicity: its only parameter is the number of neighbors, which we set to 10.

The rationale for using kNN as an evaluation model is as follows. If a feature ranking is meaningful, then when the feature importances are used as weights in the calculation of the distances kNN should produce better predictions as compared to kNN without using these weights [26]. Once the kNN model is trained on $\mathscr{D}_{\text{TRAIN}}$, its performance on $\mathscr{D}_{\text{TEST}}$ is measured in terms of the area under the average precision-recall curve $\text{AU}\overline{\text{PRC}}$ [25].

It might seem that another possible approach to evaluation is extending a dataset with some randomly generated features and then see whether they are ranked at the bottom of the ranking. However, this approach is more suitable for synthetic data where the ground truth is known and all features can be made relevant. In the real world data, it may very well happen that some of the high-dimensional datasets indeed contain completely irrelevant features, hence, using this approach would yield incorrect performance estimates.

3.4 Statistical Analysis of the Results

For comparing two algorithms, we use the Wilcoxon's test, and for comparing more than two algorithms, we use the Friedman's test. In both cases, the null hypothesis H_0 is that all considered algorithms have the same performance. If H_0 is rejected by the Friedman's test, we additionally apply Nemenyi's post-hoc test to investigate where the statistically significant differences between *any* two algorithms occur. A detailed description of all tests is available in [8].

When performing Wilcoxon's tests whose outcomes are not independent, we control the false discovery rate by the Benjamini-Hochberg procedure [2]: let p_i be the i-th smallest among the obtained p-values, and t the number of tests. Let i_0 be the largest i, such that $p_i \leq \hat{\alpha}_i := (i/t)\alpha$. Then, we can reject the hypotheses belonging to p-values p_i, for $1 \leq i \leq i_0$.

The results of the Nemenyi's tests are presented on average ranks diagrams. Each diagram shows the average rank of the algorithm over the considered datasets, and the critical distance, i.e., the distance for which average ranks of two considered algorithms must differ to be considered statistically significantly different. Additionally, the groups of algorithms among which no statistically significant differences occur are connected with a line. If Friedman's test did not reject the null hypothesis, all algorithms on the average rank diagram are connected with the same line, and no critical distance is given.

Before proceeding with the statistical analysis, we round the performances to three significant digits. In the analysis, the significance level was set to $\alpha = 0.05$.

3.5 Parameter Instantiation

First, we give the parameters used in the process of obtaining the ensemble-based rankings. Afterwards, we give those for HMLC-Relief.

We consider the following ensemble sizes: $|\mathscr{F}| \in \{10, 25, 50, 75, 100, 150, 250\}$. Since 100 trees is a typical recommended value [15], this should be enough. This is the only parameter for Bagging, while Random Forests and Extra Trees method need another one: the number of features F' considered in each internal node as described in Sec. 2.1. The recommended value for Random Forests is $F' = \sqrt{F}$ [15] and $F' = F$ for Extra Trees [15]. However, if we carefully examine the data, we see that are some datasets, e.g., `enron-corr` and `struc-ara-GO`, for which the diversifying mechanism of the Extra Trees algorithm does not work if we set $F' = F$, since every single one of the attributes take at most two values which results in only one possible split per attribute. Thus, we rather choose $F' = \sqrt{F}$ for Extra Trees also, since a necessary condition for an ensemble to be more accurate than any of its individual members, is that the members are diverse models [12][3].

As for Relief algorithm, it has been shown in the previous experiments that the Relief algorithm is quite robust regarding the value of the number of neighbors K, and that no other value outperforms $K = 15$ [17, 19], hence we se stick to this value. Since the datasets are not all of equall size, the number of iterations m in HMLC-Relief algorithm is given as the proportion of the size of $\mathscr{D}_{\text{TRAIN}}$. The considered values are $m \in \{1\%, 5\%, 10\%, 25\%, 50\%, 100\%\}$.

4 Results

In this section, we give answers to our experimental questions from Sec. 3.1.

4.1 Saturation of the Ranking Quality

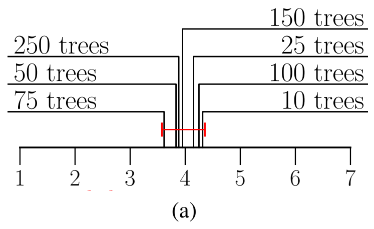
First, we show when do ensemble rankings saturate and then proceed to the HMLC-Relief ranking.

[3] If we fix the ensemble size and feature ranking score and compare the quality of the rankings from the ensembles of Extra Trees that use $\sqrt{F}$- and F-feature subsets via Wilcoxon's test, the results show that $\sqrt{F}$-version of the ensemble statistically significantly outperforms the F-version of the ensemble on the problematic datasets, and that there are no statistically significant differences on the other datasets.

4.1.1 Influence of the Ensemble Size

We analyze each ranking score and ensemble separately. While these two are fixed, we let the number of the trees in the ensemble vary, and apply Friedman's test to discover whether some differences among them occur.

The resulting p-values are all bigger than 0.05, which means that the rankings can be computed very efficiently since it suffices to grow only 10 trees. Therefore, the ensemble size in the subsequent experiments is fixed to 10. Fig. 2a shows the resulting average rank diagram for Genie3 score, computed from a Bagging ensemble. Similar conclusions can be made using the other ensembles and scores.



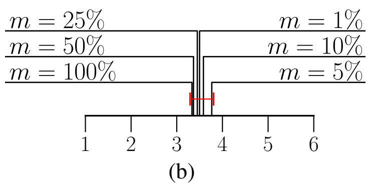


Figure 2
Saturation of the rankings: Friedman's test discovered no statistically significant differences among different (a) ensemble sizes in the ensemble rankings (Genie3 score, coupled with the Bagging ensemble is shown), (b) considered proportions of the dataset by HMLC-Relief.

4.1.2 Influence of the HMLC-Relief's Number of Iterations

Similarly to previous setting, we let the value of the parameter m vary and compare the quality of the corresponding HMLC-Relief rankings by applying Friedman's test. Again, there are no statistically significant differences among the algorithms ($p = 0.96$) and the differences among quality of the rankings are now even smaller. As shown in Fig. 2b, no two average ranks differ by more than 0.43. Since the most time-efficient setting is $m = 1\%$, this is the considered number of iterations in the subsequent HMLC-Relief experiments.

4.2 Are the Ensemble-Based Rankings Relevant?

To answer this question, we compare the predictive performance of the kNN classifier which uses the importances from a particular feature ranking, to a non-weighted kNN baseline by using Wilcoxon's test. This pair-wise comparison is made for every admissible pair of feature ranking score and ensemble method, which results in 8 (not independent) comparisons. After we compute the p-values, the Benjamini-Hochberg correction is applied.

It turns out that all weighted kNN classifiers perform better than the baseline. However, the differences are statistically significant in five out of eight cases, as shown in Tab. 2. The cases where the weighted kNN model is not statistically significantly better than the baseline, are both feature rankings that are computed using Random Forest score and the feature ranking computed from Symbolic score and Extra Trees ensemble.

Table 2
The relevance of ensemble-based feature rankings as assessed by Wilcoxon's test and the Benjamini-Hochberg correction applied. Cases where the weighted kNN performed statistically significantly better than the baseline, are bolded.

Feature Ranking	p-value
(Symbolic, Random Forests)	$\mathbf{2.14 \cdot 10^{-4}}$
(Genie3, Random Forests)	$\mathbf{1.03 \cdot 10^{-3}}$
(Symbolic, Bagging)	$\mathbf{2.41 \cdot 10^{-3}}$
(Genie3, Extra Trees)	$\mathbf{2.06 \cdot 10^{-2}}$
(Genie3, Bagging)	$\mathbf{2.63 \cdot 10^{-2}}$
(Random Forest, Random Forests)	$4.17 \cdot 10^{-2}$
(Random Forest, Bagging)	$4.49 \cdot 10^{-2}$
(Symbolic, Extra Trees)	$3.29 \cdot 10^{-1}$

Thus, using Genie3 score always results in relevant rankings, while Symbolic score fails to yield relevant rankings when used in combination with Extra Trees. The fact that Random Forest ranking fails to yield relevant rankings in all cases, may be at least partially explained by the sparsity of the data, since in that case, the differences of the error estimates on the out of bag examples and out of bag examples with permuted values of a feature, may not be that significant.

4.3 The most Appropriate Ensemble for a Given Score

In this section, we fix the remaining parameter of the ensemble-based rankings, i.e., we find the most appropriate ensemble method for each feature ranking score. This is done by first fixing a feature ranking score, and then comparing the quality of the rankings obtained using this score and one of the possible ensemble methods. In the case of Symbolic and Genie3 score, Friedman's test is applied, since they can be used in combination with three ensemble methods. In the case of Random Forest score which cannot be paired with Extra Trees, Wilcoxon's test is used.

In the case of Symbolic ranking, the obtained p-value is $p = 0.106$, hence the differences are not statistically significant, as shown in Fig. 3a. Following the rationale from the previous section, Random Forests ensemble is proclaimed as the optimal one, since this method is considerably more time-efficient than Bagging. Since the majority of attributes is numeric (or can be considered numeric, because they are nominal and binary), Extra Trees i) have the same $\mathcal{O}$ time complexity of inducing one node as Random Forests, ii) typically result in bigger trees than Random Forests, the Random Forests ensemble is the most time efficient.

We observe a similar situation when comparing different ensemble methods when the feature ranking score is fixed to Genie3, as shown in Fig. 3b. Again, no statistically significant differences are found ($p = 0.705$), hence Random Forests ensemble is again chosen as the most appropriate one.

In the case of Random Forest feature ranking score, Wilcoxon's test detects statistically significant differences ($p = 0.030$): Random Forests ensemble performs worse than Bagging ensemble, hence the latter is the most appropriate one for this feature ranking score.

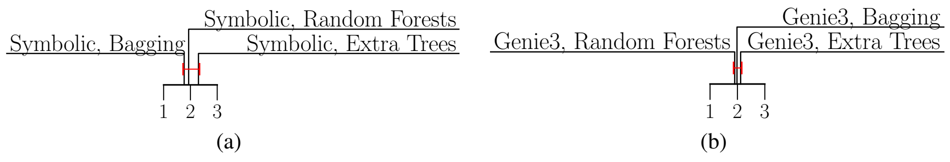


Figure 3
The quality of different feature rankings as assessed by Friedman's test, when ensemble method varies, and feature ranking score is fixed to (a) Symbolic and (b) Genie3. No statistically significant differences were found.

4.4 Comparison of the Scores

In Sec. 4.1, we have shown that we obtain as good as it gets ensemble-based feature rankings when we grow 10 trees, and as good as it gets HMLC-Relief feature rankings when we set then number of iterations to $m = 1\%$ of the training set size $|\mathscr{D}_{\text{TRAIN}}|$. In the previous section, we additionally found the most appropriate ensemble method for a given feature ranking score. In this section, we first check whether the three ensemble scores computed with the optimal parameters outperform the HMLC-Relief score, computed with the optimal parameters. This is done in a similar fashion to the Sec. 4.2 by applying three pairwise comparisons via Wilcoxon's test and correcting the results with Benjamini-Hochberg procedure.

The differences reported here are always in favor of the ensemble-based scores. The obtained p-values are (sorted in the increasing order): $p_1 = 4.86 \cdot 10^{-5}$ (Symbolic, Random Forests), $p_2 = 1.74 \cdot 10^{-4}$ (Genie3, Random Forests), $p_3 = 1.59 \cdot 10^{-3}$ (Random Forest, Bagging). After applying the correction, all three differences are statistically significant, hence all three ensemble-based rankings outperform the HMLC-Relief ranking.

Next, we investigate which of the ensemble-based scores performs best. To this end, we apply Friedman's test. The obtained p-value equals $2.33 \cdot 10^{-3}$ and we can proceed to the Nemenyi's post-hoc test to discover where the differences occur. The results are shown in the Fig. 4.

There are two groups of scores that do not perform statistically significantly different. The first group consist of Symbolic and Genie3 score, and the second one consists of Genie3 and Random Forest score. The graph also reveals Symbolic score

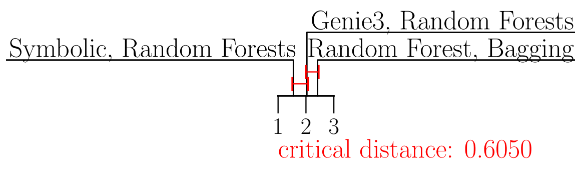


Figure 4
Comparison of the three ensemble-based feature ranking scores.

(with the average rank of 1.55) performs statistically significantly better than Random Forest score (with the average rank of 2.42). Since the Random Forest score has also the worst time complexity, we prefer the other two over it.

4.5 A Closer Look to Some Other Rankings' Characteristics

A high number of benchmark problems allows for a statistical analysis performed in the previous sections, where we were able to draw some definite conclusions. However, when averaging the performances, some information is always lost, therefore we now take a closer look at two additional characteristics of the obtained feature rankings. For example, one of the characteristics that was already mentioned is that the quality of the ranking stabilizes quite quickly, i.e., growing ten trees suffices. In this section, we present some others and use the graphs in Fig. 5 which show the results for `hom-ara-FUN` dataset, as a running example. One of the reasons for choosing this dataset is that it is high dimensional and is also one the datasets where considering all features when inducing the trees in the Extra Trees ensemble does not work. This is visible from Fig. 5a, which shows the qualities of the Symbolic ranking, computed from different ensembles. Considering only a subset of features in each node (Extra Trees, SQRT) as compared to considering all the features (Extra Trees, all), pushes the quality of the rankings over the baseline.

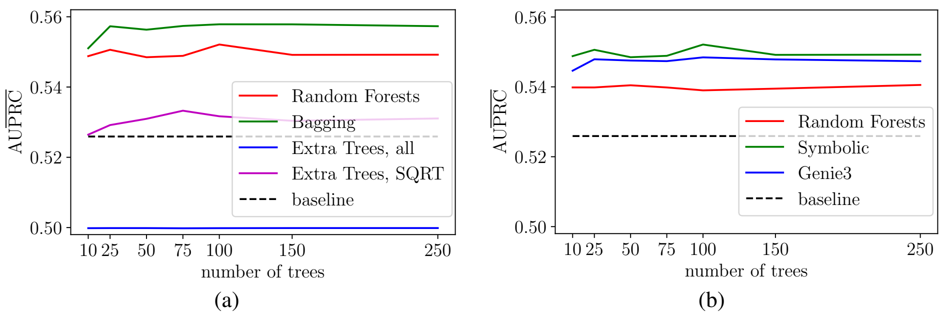


Figure 5
Quality of the feature rankings on the `hom-ara-FUN` dataset when number of trees varies and (a) feature ranking score is fixed to Symbolic, (b) ensemble method is fixed to Random Forests. In the figure (a), all and SQRT denote the number of features considered in the Extra Trees algorithm.

The order of the rankings. Fig. 5b depicts typical situation with respect to the order of the feature ranking scores when an ensemble method is fixed. More precisely, only in 4 out of 30 cases, the ranking that belongs to the Random Forest score is placed in between those that belong to Symbolic and Genie3 score. This can be explained by the fact that the mechanisms for computing feature relevances in the latter two scores are more similar to each other than to the mechanism of Random Forest score. Indeed, the statistics of the nodes that are considered by Symbolic score (number of examples) and Genie3 score (variance reduction), are somewhat related since number of examples is also part of the variance reduction statistic. Random Forest score, on the other hand, takes a look at the error reduction values.

Efficiency of the rankings. Under the efficiency of the ranking, we mean the (relative) number of the features that have a positive feature importance. The lower the number, the more efficient the ranking. In Fig. 5b, all three scores result in a relevant feature rankings, since all three curves lie above the baseline. The dataset at hand has the second highest number of features (72869) and it is surprising that the weighted kNN algorithms which make use of the weights from the rankings, use at most 8% of the features: this is the approximate number of features in the case of Genie3 and Symbolic score. In the case of the Random Forest score, the number is even lower: only 3%. That means that more than 90% of the features were proclaimed completely irrelevant and have weight 0 (or negative). Similar situation was observed for the other extremely high dimensional datasets, e.g., `wipo` where Random Forest rankings proclaim 99% of the features irrelevant. The complete statistics are given in Tab. 3.

By carefully inspecting the table, we make three observations. First, Symbolic and Genie3 score columns are equal, but this can be explained by the fact that they are computed from the same ensemble (Random Forests) and the terms in Eq. (1) and Eq. 2 are always positive which is obvious for the Symbolic score, and can be proven with simple algebra for the Genie3 score.

A more interesting observation is that Random Forest ranking is consistently more efficient that the other two ensemble rankings. Only on the two of 30 datasets, Random Forest score is less efficient that the other two. The reason for this is most likely the following. Note that Random Forest rankings are computed from the Bagging ensemble which always considers all features when inducing a new node of a tree. If the relevant features can be told apart from the irrelevant ones, then, always one of the relevant features would be chosen in a test split. On contrary, Genie3 and Symbolic scores are computed from Random Forests ensemble, which consider only a subspace of features, so all (or most of the) relevant ones can be skipped by chance, so from time to time, an irrelevant feature appears in a test. In addition to that, bootstrapping may also play an important role in this process. If the data is sparse which is true for many of the datasets (including, e.g., `yeast-GO`), then it can happen that different features are important for different bootstrap replicates.

The last observation is that HMLC-Relief feature rankings are typically more efficient than the ensemble-based feature rankings. This is another proof that data is sparse, since the second term in Eq. (4) - which can make the relevance negative - should converge to zero when the domain is populated with more and more ex-

Table 3
The proportion of the features in the dataset with positive importance, for each of the rankings, computed in a optimal setting as given in Sec. 4.4.

dataset	Genie3	Random Forest	Symbolic	HMLC-Relief
cellcycle-yeast-FUN	100%	100%	100%	71%
church-yeast-FUN	85%	81%	85%	26%
clef07a-is	100%	100%	100%	64%
derisi-yeast-FUN	95%	79%	95%	59%
diatoms	95%	78%	95%	78%
eisen-yeast-FUN	100%	100%	100%	62%
enron-corr	76%	54%	76%	16%
expr-yeast-FUN	96%	94%	96%	38%
exprindiv-ara-FUN	86%	68%	86%	45%
exprindiv-ara-GO	98%	94%	98%	42%
gasch1-yeast-FUN	100%	100%	100%	61%
hom-ara-FUN	8%	3%	8%	56%
hom-ara-GO	16%	6%	16%	76%
hom-yeast-FUN	15%	6%	15%	14%
icpr2010	88%	66%	88%	48%
interpro-ara-FUN	19%	8%	19%	6%
interpro-ara-GO	26%	10%	26%	3%
pheno-yeast-FUN	65%	68%	65%	6%
reuters	5%	3%	5%	2%
scop-ara-FUN	20%	17%	20%	5%
scop-ara-GO	29%	23%	29%	6%
seq-ara-FUN	60%	36%	60%	47%
seq-ara-GO	84%	64%	84%	67%
seq-yeast-FUN	91%	89%	91%	14%
spo-yeast-FUN	96%	95%	96%	39%
struc-ara-FUN	36%	16%	36%	53%
struc-ara-GO	62%	27%	62%	53%
struc-yeast-FUN	29%	9%	29%	32%
wipo	4%	1%	4%	5%
yeast-GO	22%	25%	22%	6%

amples (a sketch of a proof can be found in [17]). However, the efficiency is not correlated with the ranking quality as shown in Sec. 4.4.

Conclusions

We proposed three feature ranking scores, i.e., Symbolic, Genie3 and Random Forest score for the task of HMLC. The proposed feature ranking methods can be computed very efficiently, since it suffices to grow only 10 trees in the ensemble. The first two scores yield relevant feature ranking, while Random Forest score fails to do so. The most suitable ensemble method for Symbolic and Genie3 score is Random Forests ensemble. For Random Forest score, the most suitable ensemble method is Bagging. When coupled with the suitable ensemble method, all three scores also outperform the HMLC-Relief feature ranking. When we compare the ensemble scores to each other, Symbolic score statistically significantly outperforms the Random Forest score, therefore we suggest to use either the former or Genie3 score, since there are no statistically significant differences among these two (but the Symbolic score has the lowest rank on average).

We have also shown that Symbolic and Genie3 score are more closely related to each other than to Random Forest score. Especially on the extremely high-dimensional datasets, all three feature ranking scores successfully filter out a majority of the features and still outperform the baseline that uses all of them. The HMLC-Relief feature rankings are even more efficient in that sense, but they are of lower quality.

This work can be extended in at least two directions. First, one could improve the HMLC-Relief, so that its performance would be comparable to the ensemble-based rankings. Next, we could extend the ensemble-based scores to the gradient boosting ensemble technique which is inherently different from the ones presented here, since there, the trees are not independent of each other which also allow for the analysis of the development of the ranking through the iterations.

Acknowledgement

We acknowledge the financial support of the Slovenian Research Agency via the grants P2-0103 and a young researcher grant to MP, as well as the European Commission, through the grants MAESTRA (Learning from Massive, Incompletely annotated, and Structured Data) and HBP (The Human Brain Project), SGA1 and SGA2. SD also acknowledges support by Slovenian Research Agency (via grants J4-7362, L2-7509, and N2-0056), the European Commission (project LANDMARK) and ARVALIS (project BIODIV). The experiments presented here were executed on the computing infrastructure from the Slovenian Grid (SLING) initiative.

References

[1] Barutcuoglu, Z., Schapire, R. E., and Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836.

[2] Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society.*, 57(1):289–300.

[3] Blockeel, H. (1998). *Top-down Induction of First Order Logical Decision Trees*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium.

[4] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[5] Breiman, L., Friedman, J., Olshen, R., and Stone, C. J. (1984). *Classification and Regression Trees*. Chapman & Hall/CRC.

[6] Clare, A. (2003). *Machine learning and data mining for yeast functional genomics*. PhD thesis, University of Wales Aberystwyth, Aberystwyth, Wales, UK.

[7] Consortium, T. G. O. (2000). Gene ontology: tool for the unification of biology. *Natural Genetics*, 25(1):25–29.

[8] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

[9] Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2008). Hierchical annotation of medical images. In *Proceedings of the 11th International Multiconference - Information Society IS 2008*, pages 174–181. IJS, Ljubljana.

[10] Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2011). Hierarchical classification of diatom images using predictive clustering trees.

[11] Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2010). Detection of visual concepts and annotation of images using ensembles of trees for hierarchical multi-label classification. In *Recognizing Patterns in Signals, Speech, Images and Videos*, pages 152–161, Berlin, Heidelberg. Springer Berlin Heidelberg.

[12] Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001.

[13] Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS One*, 5(9):1–10.

[14] Klimt, B. and Yang, Y. (2004). The enron corpus: A new dataset for email classification research. In *ECML '04: Proceedings of the 18th European Conference on Machine Learning – LNCS 3201*, pages 217–226. Springer Berlin / Heidelberg.

[15] Kocev, D. (2011). *Ensembles for predicting structured outputs*. PhD thesis, IPS Jožef Stefan, Ljubljana, Slovenia.

[16] Kocev, D., Vens, C., Struyf, J., and Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3):817–833.

[17] Kononenko, I. and Robnik-Šikonja, M. (2003). Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning Journal*, 55:23–69.

[18] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

[19] Petković, M., Džeroski, S., and Kocev, D. (2018). Feature ranking with relief for multi-label classification: Does distance matter? In Soldatova, L., Vanschoren, J., Papadopoulos, G., and Ceci, M., editors, *Discovery Science*, pages 51–65. Springer.

[20] Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626.

[21] Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., and Džeroski, S. (2010). Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11(1).

[22] Slavkov, I., Karcheska, J., Kocev, D., and Deroski, S. (2018). HMC-ReliefF: Feature ranking for hierarchical multi-label classification. *Computer Science and Information Systems*, 15(1):187–209.

[23] Stańczyk, U. and Jain, L. C., editors (2015). *Feature Selection for Data and Pattern Recognition.* Studies in Computational Intelligence. Springer Berlin Heidelberg.

[24] Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, pages 1–13.

[25] Vens, C., Struyf, J., Schietgat, L., Džeroski, S., and Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214.

[26] Wettschereck, D. (1994). *A study of distance based algorithms.* PhD thesis, Oregon State University, USA.

Chapter 6

Feature Ranking for Semi-Supervised Structured Output Prediction

In this Chapter, we present in detail the third set of contributions of the thesis: those concerning the semi-supervised learning (SSL) setting and the predictive modeling tasks of single target classification (STC) and regression (STR), multi-target regression (MTR), multi-label classification (MLC) and hierarchical multi-label classification (HMLC). Recall from the introductory sections that in the SSL setting, some (or the majority) of examples are unlabeled, i.e., have missing target values (Section 2.2.4). Also recall that we briefly presented our feature ranking methods for SSL in Sections 3.1.3 and 3.2.3. Our contributions in this context include:

1. An extension of ensemble-based feature ranking scores from the context of supervised learning to the context of SSL, for both primitive and structured outputs.
2. An extension of distance-based feature ranking scores from the context of supervised learning to the context of SSL, for both primitive and structured outputs.
3. An extensive experimental evaluation of the newly proposed feature importance scores for SSL on task appropriate collections of benchmark datasets, assessing the performance of the scores individually and in cross-comparison.

Chronologically, we first proposed the extension of ensemble-based feature ranking methods from the supervised classification to the semi-supervised classification setting in a paper presented at the DS-2019 conference (Petković, Džeroski, et al., 2019). As outlined in Chapter 3, the proposed feature ranking methods consist of pairs of ensemble generation methods (bagging, random forests and extra trees) and scores (Symbolic, Genie3, and Random Forest scores). We evaluated the performance of the feature ranking methods on a range of benchmark datasets, and the main findings as follows. First, we show that using unlabeled data improves the quality of feature rankings, even when the clustering hypothesis (Section 2.2.4) is not satisfied. Thus, SSL feature rankings outperform their supervised counterparts. Next, we show that the proposed feature rankings are state-of-the art and, finally, that they are all equally good.

We considerably extended this work (Petković, Džeroski, et al., 2020b) to handle the other aforementioned predictive modeling tasks (STR, MTR, MLC, HMLC). We also proposed extensions of distance-based methods for all the tasks (STC, STR, MTR, MLC, HMLC) in the SSL setting. We also extensively evaluated all the methods on collections of datasets for each task, with additional STC datasets as compared to (Petković, Džeroski, et al., 2019). The main findings of the evaluation on the 38 benchmark datasets are the following.

In the majority of cases, the SSL feature rankings outperform their supervised counterparts, regardless of whether the clustering hypothesis is satisfied for a particular dataset. Next, SSL feature rankings give a more global picture of a dataset, as compared to the supervised feature rankings. Finally, for all the predictive modeling tasks, except for classification (where the number of features in the considered datasets is not extremely high), ensemble-based SSL feature ranking methods outperform the distance-based SSL feature ranking methods. However, the winner among the ensemble-based methods, i.e., the best feature ranking method overall, changes from task to task: Symbolic ranking is the best for STR and MTR, Random forest ranking for MLC and, Genie3 for HMLC.

The work presented in this Chapter addresses the following hypotheses (as defined in the introduction):

H1: It is possible to extend ensemble- and distance-based feature ranking approaches to the unsupervised feature ranking task, to the tasks of supervised SOP (i.e., MTR, MLC, and HMLC), and to their semi-supervised versions.

H6: For unsupervised, MTR, MLC and HMLC problems, the ensemble-based feature ranking approaches on average outperform the distance-based approaches to feature ranking when the number of features is extremely high.

H7: The proposed ensemble- and distance-based feature ranking approaches yield relevant and state-of-the-art feature rankings for feature-ranking problems in SSL for classification.

H8: The proposed ensemble- and distance-based feature ranking approaches yield relevant and state-of-the-art feature rankings for feature-ranking problems in SSL for different types of SOP.

These hypotheses are confirmed with the design and implementation of the feature ranking approaches presented in this Chapter and the experimental studies comparing their performance. Hypotheses H7 and H8 are completely confirmed by the results presented in this Chapter. For hypotheses H1 and H6, the parts pertaining to SSL are confirmed in this Chapter, while the remaining parts of these hypotheses are addressed in the previous and the next chapters.

The papers included in this Chapter are:

- Petković M., Džeroski S., Kocev D. (2019) Ensemble-Based Feature Ranking for Semi-supervised Classification. In: Proceeinds of the 22nd International Conference on Discovery Science, LNCS, 11828: 290-305. Springer, Cham

- Petković, M., Džeroski, S. and Kocev, D. Feature Ranking for Semi-supervised Learning. *Machine Learning.* Under review.

The contribution of Matej Petković to these papers are as follows. MP contributed to the design of the ensemble-based and distance-based feature ranking methods for SSL and implemented these methods in computer code. He also participated in designing the experiments, carried out the experiments, and processed their results. He drafted the papers and revised them following the feedback from the co-authors and reviewers.

Ensemble-Based Feature Ranking for Semi-supervised Classification

Matej Petković[1,2], Sašo Džeroski[1,2], and Dragi Kocev[2]

[1] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[2] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
{name.surname}@ijs.si, http://kt.ijs.si

Abstract. In this paper, we propose three feature ranking scores (Symbolic, Genie3, and Random Forest) for the task of semi-supervised classification. In this task, there are only a few labeled examples in a dataset and many unlabeled. This is a highly relevant task, since it is increasingly easy to obtain unlabeled examples, while obtaining labeled examples is often an expensive and tedious task. Each of the proposed feature ranking scores can be computed by using any of three approaches to learning predictive clustering tree ensembles (bagging, random forests, and extra trees). We extensively evaluate the proposed scores on 8 benchmark datasets. The evaluation finds the most suitable ensemble method for each of the scores, shows that taking into account unlabeled examples improves the quality of a feature ranking, and demonstrates that the proposed feature ranking scores outperform a state-of-the-art semi-supervised feature ranking method SEFR. Finally, we identify the best performing pair of a feature ranking score and an ensemble method.

Keywords: semi-supervised learning · feature ranking · ensembles

1 Introduction

A center task in machine learning is predictive modeling concerned with learning a predictive model, from a given training dataset of values of features-target pairs $(\boldsymbol{x}, y)$, where $\boldsymbol{x} = (x_1, \dots, x_F)$. The learned predictive models can then be used to predict target values for previously unseen values of features. In this paper, we focus on the classification task, where the domain $\mathcal{Y}$ of the target y is a finite set of discrete values. The task at hand is called binary classification if $|\mathcal{Y}| = 2$, and multi-class classification if $|\mathcal{Y}| > 2$. In both cases, we refer to the target values of y as labels or classes.

Typically, all the examples in the training set are labeled, i.e., have a known value of y. In that case, we say that the learning is supervised. However, determining which class an example belongs to, might be very expensive or take too much time in some domains (e.g., compound toxicity). Hence, there is a multitude of datasets with only a handful of labeled examples and many unlabeled ones. To address this challenge, methods that can use the unlabeled data in the learning phase have been developed [6,14]. These semi-supervised learning (SSL) methods are applicable mostly when there are only a few labeled examples and plenty of unlabeled data, which makes applying the supervised learning hard.

Another prominent task in machine learning is feature ranking where, the goal is to discover to what extent each of the features x_i, $1 \leq i \leq F$ is relevant for the class $y(\boldsymbol{x})$. Formally, given a dataset $\mathscr{D}_{\mathrm{TRAIN}}$, the output of a feature ranking algorithm is a list of feature importance scores $\mathit{importance}(x_i)$, where a higher score corresponds to a higher relevance of the feature for the target values. The task of feature ranking is typically seen as a data preprocessing step. We can perform dimensionality reduction on the data to make the learning of a predictive model faster or even at all feasible. This is done by discarding the features that have lower importance than some threshold $\vartheta \in \mathbb{R}$. Lower dimensionality also results in models that are easier to understand – this is particularly important when a data scientist collaborates with a domain expert. Lately, much work is being done in the field of explainable artificial intelligence. In the case of black box models, such as neural networks and ensembles, feature ranking is the only way to at least partially explain the obtained predictions.

A task that is related to feature ranking is feature selection. The goal of feature selection is to identity a subset of the features that yield better (or at least the same) predictive performance when used to learn predictive models (as compared to learning a predictive model on the complete feature set). Note that this is a task different from feature ranking: the former looks for the best subset, while the latter focuses on ordering the features based on their relevance for the target. Notwithstanding, as mentioned above, feature ranking can be used to perform feature selection by applying a threshold on the importance scores.

We propose a method for SSL feature ranking, i.e., learning a feature ranking for datasets with a handful of labeled examples and many unlabeled examples. The proposed method is based on the ensemble learning paradigm [16]. It uses tree-based methods for semi-supervised learning of classification trees [14].

We perform an empirical evaluation of the proposed method on 8 benchmark multi-class classification datasets. In the empirical evaluation, we set out to investigate the influence of the combination of ensemble learning methods and feature ranking scores, the number of labeled examples, and the number of unlabeled examples. Moreover, we compare the performance of the variants of the proposed method to the performance of their fully supervised counterparts as well as to the performance of another competing method for semi-supervised feature ranking based on ensemble learning [1].

The remainder of this paper is organized as follows. In Section 2, we review the background and the related work. Next, in Section 3, we describe the proposed method for feature ranking for semi-supervised classification. Furthermore, we outline the design of the empirical evaluation of the proposed method in Section 4 and then we discuss the results of the evaluation in Section 5. Finally, we conclude and provide directions for further work in Section 6.

2 Background and related work

2.1 Related Work

There are three major groups of methods that address the SSL task. The simplest option is to *discard the unlabeled data*, and then use an existing algorithm for supervised learning. However, when the number of labeled examples is really low (e.g., 10 or 20 examples), this approach has severely limited success.

The second group of methods performs *self-training* [17] where an algorithm for supervised learning is first applied to the labeled examples in $\mathscr{D}_{\text{TRAIN}}$. Next, the resulting model is used to predict the target values of the unlabeled examples, and a heuristic score is used to assess the certainty/reliability of the predictions of the models. The examples for which the algorithm is the most certain in its predictions, keep their labels and the examples are added to the current set of labeled examples. The cycle is iteratively repeated until a stopping criterion is met (e.g., no more unlabelled examples or no more reliable predictions). At the end, a model is learned on the final set of labeled examples.

The last class of SSL methods are *algorithm-adaptation* methods, where an existing algorithm, e.g., for learning decision trees [3], is adapted so that it can also take into account the unlabeled examples. In the case of decision trees, this was done by adapting the heuristic that measures the impurity of the current dataset [14], so that not only the target y is taken into account, but also the features x_i. Under the clustering assumption, i.e., the assumption that the class values correspond to well-defined clusters of data [6], the last two approaches are expected to be superior to the first solution.

Like supervised feature ranking methods, feature ranking methods for SSL belong to three major groups [18]: *filter* methods, where no predictive model is needed for feature ranking; *embedded* methods, where feature ranking is computed directly from a predictive model; and *wrapper* methods, where a predictive model is typically retrained more than once and the ranking is built iteratively.

Filter methods are typically the fastest, but can be myopic. Namely, they do not take into consideration possible feature interactions and have limited scope, e.g., the variance score [2] is applicable to datasets with numeric features only. A representative of embedded methods is the SEFR feature ranking [1] computed from an ensemble of decision trees: this is an SSL adaptation of the random forest ranking [4], where the trees are learned in a self-training fashion. A representative of the wrapper methods is the method for recursive feature elimination with support vector machines (SVMs) [20]. At each iteration, an SVM model is trained and thus a normal $\boldsymbol{w}$ is obtained. Features x_i for which the absolute value of components $|\boldsymbol{w}_i|$ are the smallest are removed, and the procedure is iteratively repeated until a complete ranking is obtained. For other SSL feature ranking methods (which are mostly limited to feature selection of numeric features), we refer the reader to a recent survey [18].

2.2 Semi-supervised Predictive Clustering Trees

The proposed feature ranking method is based on ensembles of predictive clustering trees (PCTs) for classification. The PCT framework views a decision tree as a hierarchy of clusters, which are induced with the standard top-down induction process [5] described in Alg. 1. The root of a PCT corresponds to a cluster containing all data, which is recursively partitioned into subclusters while moving down the tree. The leaves represent the clusters at the lowest level of the tree hierarchy and each leaf is labeled with its cluster's prototype (prediction). PCTs generalize decision trees and can be used for a variety of learning tasks, including clustering tasks, different types of structured output prediction tasks [3,12], as well as SSL tasks [14]. The generalization is based on appropriately adapting the heuristic for inducing PCTs and the prototype function to the given structured output prediction task.

Alg. 1 is used for learning PCTs. Its input is a set of examples $E \subseteq \mathscr{D}_{\text{TRAIN}}$, and its output is a tree. The heuristic h that is used for selecting the best test at a node is the weighted impurity reduction of the subsets of E (lines 3 and 4), induced by the tests. By maximizing it (line 5 of Alg. 2), the algorithm is guided towards small trees with good predictive performance. If no test reduces the impurity, a leaf is created and the prototype (e.g., average) of the instances belonging to that leaf is computed.

Algorithm 1 PCT(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$
2: **if** $t^* \neq none$ **then**
3:     **for each** $E_i \in \mathcal{P}^*$ **do**
4:         $tree_i = \text{PCT}(E_i)$
5:     **return** $Node(t^*, \bigcup_i \{tree_i\})$
6: **else**
7:     **return** $Leaf(\text{Prototype}(E))$
```

Algorithm 2 BestTest(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: **for each** test $t$ **do**
3:     $\mathcal{P}$ = partition induced by $t$ on $E$
4:     $h = |E| impu(E) - \sum_{E_i \in \mathcal{P}} |E_i| impu(E_i)$
5:     **if** $h > h^*$ **then**
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: **return** $(t^*, h^*, \mathcal{P}^*)$
```

In the standard classification scenario, the impurity $impu(E)$ of a data subset E is defined as the Gini impurity of the class y, i.e., $Gini(E, y) = 1 - \sum_c p_E^2(c)$, where $p_E(c)$ is the relative frequency of the class value c in the subset E. In the semi-supervised scenario, where some or most of the target values are missing, the heuristic is adapted so that also impurity of the features is taken into account. To this end, we first define the normalized version of $Gini$, namely $Gini'(E, y) = Gini(E, y)/Gini(\mathscr{D}_{\text{TRAIN}}, y)$. Analogously, we introduce the impurity measure for numeric features as the normalized variance $Var'(E, x_i) = Var(E, x_i)/Var(\mathscr{D}_{\text{TRAIN}}, x_i)$. Finally, the impurity of a set E is defined as $impu(E) = w\ Gini'(E, y) + (1 - w) \cdot \frac{1}{F} \sum_{i=1}^{F} impu(E, x_i)$, where the $impu(E, x_i)$ is calculated as $Gini'(E, x_i)$ if x_i is nominal and as $Var'(E,\ x_i)$ if x_i is numeric, and F is the number of features and the influence of $Gini'(E, y)$ is controlled by the parameter $w \in [0, 1]$, whose optimal value is set by internal cross-validation. When computing the relative frequencies that are used in the $Gini'$ score, only the examples with known values of the target y (or feature x_i)

are taken into account. The prediction, i.e., the prototype in a leaf of a PCT, is defined as the majority class value in the leaf.

2.3 Ensembles of PCTs

An ensemble is a set of base predictive models constructed with a given algorithm. The prediction for a new example $\boldsymbol{x}$ is made by combining the predictions of the models from the ensemble. In classification tasks, this is typically done by voting, where the i-th base model either votes for the class value it predicts, or computes the probabilities $p_i(c \mid \boldsymbol{x})$, for all class values c. In the first case, the prediction of the ensemble is the class with the most votes. In the second case, the prediction is the class with the highest sum $\sum_i p_i(c \mid \boldsymbol{x})$.

A necessary condition for an ensemble to be more accurate than its members, is that the members are accurate and diverse models [9], i.e., that they make different errors on new examples. However, we do not use the ensembles as predictive models. Rather, we use them as a basis for computing feature ranking scores. As it is evident from the scores' definitions in Sec. 3 (Eqs. (1), (2) and (3)), one can also compute the ranking from a single tree, but the variance of feature importances decreases when the number of trees is higher. There are several ways to introduce diversity among the PCTs in an ensemble. We describe and make use of three of them.

Bagging and random forests. Instead of being learned from the original dataset $\mathscr{D}_{\text{TRAIN}}$, each tree in the bagging/random forest ensemble is built from a different bootstrap replicate $\mathcal{B}$ of the dataset $\mathscr{D}_{\text{TRAIN}}$, called bag. The examples $\mathscr{D}_{\text{TRAIN}} \setminus \mathcal{B}$ are called out-of-bag examples (OOB). Additionally, the line 2 of the *BestTest* procedure (see Alg. 2) is modified to change the feature set during learning by introducing randomization in the test selection. More precisely, at each node in a decision tree, a random subset of the features is taken, and the best test is selected from the splits defined by these features. The number of the retained features F' is given as a function of the total number of features F, e.g., $\lceil\sqrt{F}\rceil$, $\lceil F/4\rceil$, etc. We obtain the bagging procedure if we keep all the features, and the random forest procedure otherwise.

Extra tree ensembles. As in random forests, we consider F' features in each node, but we do not evaluate all potential tests that the features could yield. Rather, we choose randomly only one test per feature. Among these F' tests, we choose the best one. From the bias-variance point of view, the rationale behind the Extra-Trees method is that the explicit randomization of the cut-point and feature combined with ensemble averaging should be able to reduce variance more strongly than the weaker randomization schemes used by other methods [7]. Note that originally, Extra-Tree ensembles use no bootstrapping. However, we introduced it due to two main reasons: i) some preliminary experiments showed that it is beneficial to do so from the predictive power point of view, and ii) the Random Forest score (see Eq. (3)) requires OOB examples for its computation.

3 Feature Ranking Scores for SSL Classification

We first propose and describe the *Symbolic score*. Then, we proceed explaining the *Genie3* [11] and the *Random Forest* scores [4]. To avoid confusion, Random Forest score will be always in singular form and capitalized, whereas the ensemble method random forests will be in plural form and not capitalized. In the following, a tree is denoted by $\mathcal{T}$, whereas $\mathcal{N} \in \mathcal{T}$ denotes a node in a tree. Trees form an ensemble $\mathcal{E}$ of size $|\mathcal{E}|$. The set of all internal nodes of a tree $\mathcal{T}$ in which the feature x_i appears as part of a test is denoted as $\mathcal{T}(x_i)$.

Symbolic score. In the simplest version of the score, we count the occurrence of a given feature in the tests in the internal nodes of the trees. Since the features appearing closer to the root influence more examples and are intuitively more important, we define the importance of the feature x_i as

$$\mathit{importance}_{\mathrm{SYMB}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} |E(\mathcal{N})| / |\mathscr{D}_{\mathrm{TRAIN}}|, \tag{1}$$

so that the appearances of the feature x_i are weighted by the number of examples the corresponding node influences. The term $1/|\mathscr{D}_{\mathrm{TRAIN}}|$ is just a scaling factor. This is a parameter-less version of the previously defined Symbolic score [16], where an appearance of a feature x_i in node $\mathcal{N}$ was awarded $\alpha^{\mathrm{depth}(\mathcal{N})}$, where the value of the $\alpha \in (0, 1]$ had to be chosen by the user.

Genie3 score. The main motivation for this score is that splitting the current subset $E \subseteq \mathscr{D}_{\mathrm{TRAIN}}$, according to a test where an important feature appears, should result in high impurity reduction. The Genie3 importance of the feature x_i is thus defined as

$$\mathit{importance}_{\mathrm{GENIE3}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} |E(\mathcal{N})| h^*(\mathcal{N}), \tag{2}$$

where $E(\mathcal{N})$ is the set of examples that come to the node $\mathcal{N}$, and $h^*(\mathcal{N})$ is the value of the variance reduction function described in Alg. 2. Greater emphasis is again put on the features higher in the tree, where $|E|$ is larger.

Random Forest (RF) score. This score tests how much noising a given feature decreases the predictive performance of the trees in the ensemble. The greater the performance degradation, the more important the feature is.

Once a tree $\mathcal{T}$ is grown, the algorithm evaluates the performance of the tree by using the corresponding $\mathrm{OOB}_{\mathcal{T}}$ examples. This results in the accuracy $a(\mathrm{OOB}_{\mathcal{T}})$. Afterward, we randomly permute the values of feature x_i in the set $\mathrm{OOB}_{\mathcal{T}}$ and obtain the set $\mathrm{OOB}_{\mathcal{T}}^i$ with the corresponding accuracy $a(\mathrm{OOB}_{\mathcal{T}}^i)$. The importance of the feature x_i for the tree $\mathcal{T}$ is defined as the relative decrease of accuracy after noising. The final Random Forest score of the feature is the average of these values across all trees in the forest:

$$\mathit{importance}_{\mathrm{RF}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \frac{a(\mathrm{OOB}_{\mathcal{T}}) - a(\mathrm{OOB}_{\mathcal{T}}^i)}{a(\mathrm{OOB}_{\mathcal{T}})}. \tag{3}$$

Note that $a(\mathrm{OOB}_{\mathcal{T}}^{i}) = a(\mathrm{OOB}_{\mathcal{T}})$ if the feature x_i does not appear in $\mathcal{T}$. This can speed up the computation of $importance_{\mathrm{RF}}$, but this feature ranking method is still the most time consuming. While the time complexity of the first two is negligible as compared to the one of growing the forest, this one has an additional linear factor: the number of examples in the training set.

4 Experimental Design

In this section, we define the experimental questions, briefly describe the datasets, define the evaluation procedure and describe which parameters of the algorithms were used in the experiments. With the empirical evaluation, we set out to answer the following four questions:

1. Which ensemble method suits a given feature ranking score the most?
2. Can our SSL-feature ranking scores make effective use of the unlabeled examples, especially when the number of labeled examples is small?
3. Do our SSL-feature ranking scores yield state-of-the-art feature rankings?
4. Which SSL-feature ranking score has the highest quality?

Before proceeding to the rest of the section, let us mention that the proposed SSL feature ranking scores are implemented in the Clus software `http://source.ijs.si/ktclus/clus-public`, and that all the datasets, the results are available at `http://source.ijs.si/mpetkovic/ssl-fr`, together with our implementation of the competing SEFR method [1].

We use 8 benchmark classification datasets from various domains: medicine (*Arrhythmia*, *Dis*), science and technology (*Gasdrift*, *Pageblocks*, *Phishing*), gaming (*Chess*, *Tic-tac-toe*) and economy (*Bank*). The main properties of the datasets, i.e., the number of features, classes, and examples are given in Tab. 1.

Prior to performing any experiments, each dataset $\mathscr{D}$ was randomly split into $x = 10$ stratified folds which resulted in the test sets $\mathscr{D}_{\mathrm{TEST}i}$, $0 \le i < x$. In contrast to the cross-validation in the standard classification scenario where $\mathscr{D}_{\mathrm{TRAIN}i} = \cup_{j \neq i} \mathscr{D}_{\mathrm{TEST}j}$ we first define the copy $\mathscr{D}_{\mathrm{TEST}}{}_{i}^{\ell}$ of $\mathscr{D}_{\mathrm{TEST}i}$ in which we

Table 1: Basic properties of the datasets in the experiments: number of nominal and numeric features, number of examples, number of classes and the proportion of examples belonging to the majority class.

dataset	nominal	numeric	examples	classes	majority class [%]
Arrhythmia [15]	73	206	452	16	54
Bank [15]	9	7	4521	2	88
Chess [15]	36	0	3196	2	52
Dis [8]	22	6	3772	2	98
Gasdrift [15]	0	129	13910	6	22
Pageblocks [15]	0	10	5473	5	90
Phishing [15]	30	0	11055	2	56
Tic-tac-toe [15]	9	0	958	2	65

remove the labels (class values) of all but $\lfloor \ell/(x-1) \rfloor + r_i$ randomly selected examples, where $\lfloor \cdot \rfloor$ is the floor function, r is the reminder of ℓ when divided by $x-1$, and $r_i = 1$ if $i < r$ and 0 otherwise. This assures that every training set $\mathscr{D}_{\text{TRAIN}}{}_i^\ell = \cup_{j \neq i} \mathscr{D}_{\text{TEST}}{}_i^\ell$ contains precisely ℓ labeled examples. We make sure that the implication $\ell_1 \leq \ell_2 \Rightarrow$ *labeled examples of* $\mathscr{D}_{TRAIN}{}_i^{\ell_1}$ *are a subset of the labeled examples in* $\mathscr{D}_{TRAIN}{}_i^{\ell_2}$ holds.

An SSL-ranking score is computed from $\mathscr{D}_{\text{TRAIN}}{}_i^\ell$ and its standard-classification counterpart is computed on the $\mathscr{D}_{\text{TRAIN}}{}_i^\ell$ with the unlabeled examples removed. Afterward, both rankings are evaluated on $\mathscr{D}_{\text{TEST}}{}_i^\ell$. This is done by using the kNN algorithm with $k = 20$ where weighted version of the standard squared Euclidean distance is used. For two input vectors $\boldsymbol{x}^1$ in $\boldsymbol{x}^2$, the distance d between them is defined as $d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sum_{i=1}^F w_i d_i^2(\boldsymbol{x}_i^1, \boldsymbol{x}_i^2)$, where d_i is defined as the absolute difference of the feature values scaled to the $[0,1]$-interval, if x_i is numeric, and as $\mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2]$ ($\mathbf{1}$ is the indicator function), if x_i is nominal.

We first define $w_i = \max\{\mathit{importance}(x_i), 0\}$, since Random Forest ranking can award a feature a negative score. In the degenerated case when the resulting values all equal 0, we define $w_i = 1$, for all features x_i. The first step is necessary to ignore the features that are of lower importance than a randomly generated one would be. The second step is necessary to ensure d is well-defined.

The evaluation through kNN was chosen because of three main reasons. First, this is a distance based method, hence, it can easily make use of the information contained in the feature importances in the learning phase. Second, kNN is simple: Its only parameter is the number of neighbors. In the prediction stage, the neighbors' contributions to the predicted value are equally weighted, so we do not introduce additional parameters that would influence the performance. Finally, if a feature ranking is meaningful, then when the feature importances are used as weights in the calculation of the distances, kNN should produce better predictions as compared to kNN without using these weights [19].

To assess the predictive performance, we first compute the sum M of the x confusion matrices M_i that we obtain from the x-fold cross-validation, i.e., $M = \sum_{i=0}^{x-1} M_i$. We use the confusion matrix M to compute accuracy, Cohen's κ, F_1 score and Matthew's correlation coefficient as evaluation measures. The latter three were used since they do not give misleading results in the case of skewed class distribution. Due to space limitations, we will report only accuracy and F_1 score (considering Cohen's κ, F_1 score and Matthew's correlation coefficient leads to the same conclusions as presented here). We compute 1-versus-other macro-averaged versions of the F_1 measure and Matthew's coefficient (to also consider multi-class classification problems). Note that defining the matrix M is necessary, because averaging the scores over the folds is wrong, as the measures are not additive.

We parametrize the used methods as follows. The number of trees in the ensembles was set to 100. The number of features that are considered in each internal node was set to $\sqrt{F}$ for random forests and F for extra trees [7]. The optimal value of the parameter w for computing the ensembles of PCTs was selected by 5-fold internal cross-validation. The considered values were

$w \in \{0, 0.1, 0.2, \ldots, 0.9, 1\}$. The possible numbers of labeled examples ℓ in the training datasets were $\ell \in \{50, 100, 200, 350, 500\}$ [13].

5 Results and Discussion

5.1 Most Appropriate Ensemble

We start our analysis by choosing the most appropriate ensemble for a given feature ranking score. To this end, we fix the score (Symbolic, Genie3 or Random Forest) and compute it from the three ensembles (random forest, bagging, and extra trees), for all values of the number of labeled examples ℓ. When we evaluate these rankings on a given test set with a given evaluation measure m, we obtain a curve consisting of points $(\ell, m(\ell))$, for every score-ensemble pair. The final measure for the performance of an ensemble for a given score is the area under the corresponding curve, denoted by *auc*. When computing the *auc*, we assume that the points ℓ are equidistant, which effectively puts more weight on the lower values of ℓ, where the SSL methods are the most applicable.

Thus, for every feature ranking score, dataset and evaluation measure, we obtain curves that belong to the three ensemble methods, and rank them with respect to the *auc*: the one with the highest *auc* is assigned rank 1. We then average these ranks over the datasets and evaluation measures.

The summary of the comparison of the pairs (ensemble, score) is given in Tab. 2. We can see that the differences among different ensemble methods are not large which means that for a given score, all three ensemble methods are appropriate to some extent. This is also evident from the fact that no average rank - except maybe that of supervised extra trees coupled with Genie3 score (2.34) is close to the worst possible average rank of 3. However, note that ensembles of extra trees always have the worst average rank, hence we decide only between random forests and bagging. If we are interested only in the quality of a ranking, then the following choices are made: for Symbolic score, random forests are the most appropriate (in both the SSL and the supervised case), whereas for the other two scores (Genie3 and Random Forest) the bagging method performs best. We will stick to these choices throughout the rest of the paper. If the time complexity of inducing the ranking is also taken into account, then random

Table 2: Average ranks of the feature rankings computed from a fixed feature ranking score and varying ensemble method: random forests (RF), bagging and extra trees (ET). The ranks are reported for SSL ensembles and supervised ensembles separately.

	SSL ensembles			supervised ensembles		
score	RF	bagging	ET	RF	bagging	ET
Symbolic	1.86	2.07	2.07	1.93	1.95	2.11
Genie3	2.00	1.93	2.07	1.91	1.75	2.34
Random Forest	2.07	1.82	2.11	1.98	1.89	2.14

Table 3: Differences between the *auc* of SSL feature rankings and their supervised counterparts, measured in terms of accuracy and F_1 measure of 20NN classifier. If $\Delta > 0$, the SSL feature ranking outperforms the supervised one.

	Symbolic		Genie3		Random Forest	
dataset	Δacc	ΔF_1	Δacc	ΔF_1	Δacc	ΔF_1
Arrhythmia	0.015	0.046	-0.010	0.012	0.004	0.031
Bank	-0.030	0.058	-0.028	0.060	-0.039	0.057
Chess	-0.153	-0.168	-0.152	-0.166	-0.091	-0.107
Dis	-0.017	0.095	-0.022	0.113	-0.011	0.117
Gasdrift	-0.237	-0.217	-0.267	-0.244	-0.262	-0.237
Pageblocks	0.017	0.202	0.018	0.228	0.017	0.190
Phishing	-0.257	-0.300	-0.243	-0.288	-0.252	-0.297
Tic-tac-toe	0.110	0.201	0.105	0.191	0.142	0.265

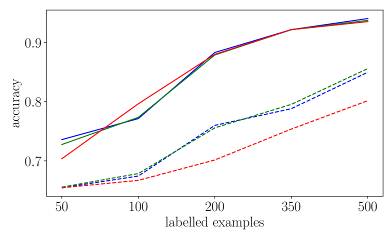
forests would be preferred over bagging, at least in the case of the supervised version of the Symbolic score, where the difference in average ranks is only 0.02, but random forests rankings are computed $\sqrt{F}$-times faster.

5.2 Can Unlabeled Data Improve a Feature Ranking?

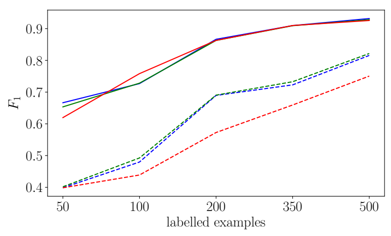
Here, we compare the quality of the SSL feature rankings to their supervised counterparts in more depth. Since the latter use only labeled data, this shows whether SSL feature rankings can make effective use of unlabeled data. We draw the same curves as in the previous section and obtain three types of diagrams as shown in Fig. 1. The Tic-tac-toe dataset (Figs. 1a and 1b) is a representative dataset where SSL rankings outperform their supervised counterparts, for all considered numbers of labeled examples ℓ. Note that for 50 labeled examples, the 20NN model that uses the supervised rankings only achieves a default accuracy (see Tab. 1), hence taking into account unlabeled examples clearly helps.

The next type of diagrams presented in Figs. 1c and 1d shows the curves for the Arrhythmia dataset. This dataset is the only one where the curves of the SSL and supervised rankings intersect: for the lower number of labeled examples (up to 200), the SSL feature rankings outperform their supervised counterparts, which again shows the usefulness of the unlabeled examples. Adding more labeled examples (at least 350), boosts the performance of supervised rankings more and they achieve better performance in this case. Sometimes, taking into account the unlabeled examples does not help, as shown in the diagrams of the last type whose representative, the Gasdrift dataset, is presented in Figs. 1e and 1f.

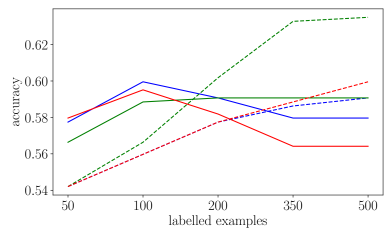
To explain this, we test the validity of the clustering assumption. Recall that, in general, SSL should be the most effective when the clusters in the data are in concordance with the values of the target variable y. To check this, we compute the k-means clustering of a dataset $\mathscr{D}$, where the parameter k is set to the number of class values. We compare the resulting partition of the $\mathscr{D}$ to that induced by the class labels $y(\boldsymbol{x})$, in terms of the Adjusted Rand Index (ARI) [10], which equals 1 when the partitions are equal and 0 if the partitions are



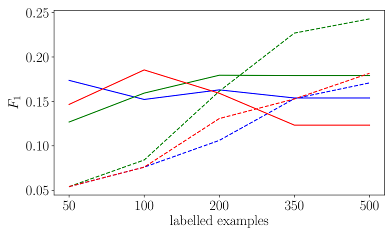
(a) Tic-tac-toe



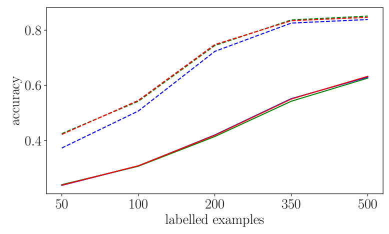
(b) Tic-tac-toe



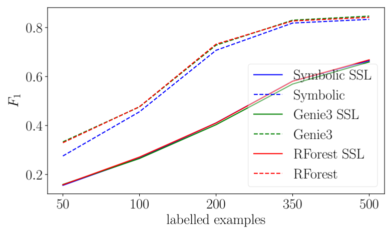
(c) Arrhythmia



(d) Arrhythmia



(e) Gasdrift



(f) Gasdrift

Fig. 1: Comparison of the SSL feature ranking with the supervised feature rankings on the Tic-tac-toe, Arrhythmia and Gasdrift datasets, in terms of accuracy and F_1 measure of the 20NN classifier. The legend in (f) applies to all subfigures.

random. The whole procedure is repeated 10 times and in the end, we compute the median of the ARI scores.

It turns out that the median ARI score for the Gasdrift dataset is quite low ($2.19 \cdot 10^{-2}$), whereas the Tic-tac-toe dataset has high ARI score ($7.00 \cdot 10^{-1}$). For the other datasets, we do not show the graphs, but only the differences between the *auc* values between the SSL and the supervised version of every ranking score. They are given in the Tab. 3. We will focus mainly on the columns that belong to the F_1 measure results, since some of the datasets (Bank, Dis, Pageblocks) are very imbalanced. We see that results are consistent for all three feature ranking scores: SSL rankings outperform their supervised versions in 5 out of 8 cases. The remaining three include the datasets Gasdrift (discussed above), Phishing and Chess. The Phishing dataset has an ARI score of $-7.16 \cdot 10^{-5}$, which effectively means that class labels are randomly distributed among the clusters; Thus, the better performance of the supervised rankings comes as no surprise. The last one is the Chess dataset which has an ARI score of $2.20 \cdot 10^{-1}$.

5.3 Are the Proposed Methods State-of-the-Art?

To answer this question, we compare our SSL feature rankings to the SEFR ranking. We follow the structure of the previous section and first show the graphs for the datasets Tic-tac-toe, Arrhythmia and Gasdrift in Fig. 2.

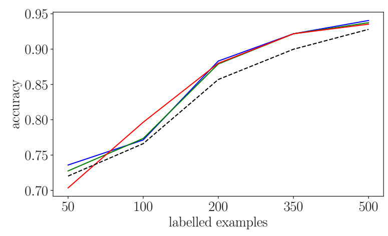
Figs. 2a and 2b depict the quality of the feature rankings computed from the Tic-tac-toe dataset, in terms of the accuracy and F_1 measure respectively. We observe that the differences between SSL and SEFR rankings are not as remarkable as those between the SSL and the supervised rankings. However, all SSL rankings still consistently outperform the SEFR ranking, except for the Random Forest ranking computed from 50 labeled examples.

Similar results are obtained for the Arrhythmia dataset as shown if Figs. 2c and 2d. Again, the differences are in favor of the SSL rankings - in this case, without exceptions. From these two graphs, we can also deduce that adding more labeled examples does not necessarily help: the quality of rankings (except for the Genie3 rankings, shown in green) does not monotonically increase and the F_1 scores of the 20NN classifiers that use the Random Forest, Symbolic and SERF importance scores computed from 500 labeled examples are even slightly lower than the F_1 scores of the same rankings computed from 50 labeled examples. The Gasdrift dataset is the one of the eight datasets for which the differences are the smallest, as shown in Figs. 2e and 2f. The quality of the rankings here increases as more labeled examples are provided. However, the differences equal approximately 0.2 percentage points, as evident from Tab. 4.

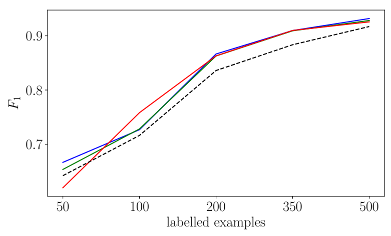
The same table also reveals that typically, the Symbolic, Genie3 and Random Forest rankings outperform the SEFR rankings, since the large majority of the Δ values in the table are positive. In terms of F_1 score, Random Forest and Symbolic rankings outperform the SEFR ranking in 7 out of 8 cases, and Genie3 ranking outperforms it in 6 out of 8 cases. The numbers for accuracy are similar, so we can rightly conclude that our rankings exhibit state-of-the-art performance.

5.4 Which Ranking Score is the Best?

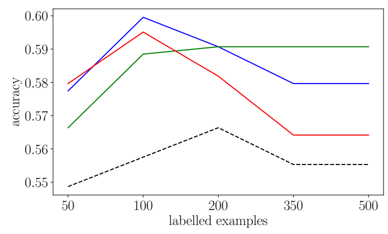
The answers to the previous two experimental questions are pretty clear. However, determining the best ranking is a bit harder. The average ranks (over all datasets and evaluation measures) of the Genie3, Random Forest and Symbolic scores are 1.90, 2.03 and 2.06 respectively, so Genie3 score performs best on average, but at the same time, it is ranked first only five times. Random Forest score is ranked first most frequently (15 times), but is also ranked last most frequently (16 times). The distribution of the Symbolic ranking is the closest to



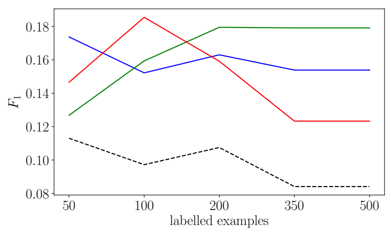
(a) Tic-tac-toe



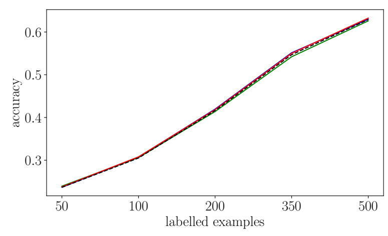
(b) Tic-tac-toe



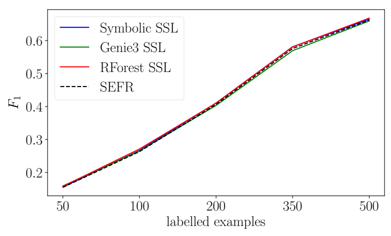
(c) Arrhythmia



(d) Arrhythmia



(e) Gasdrift



(f) Gasdrift

Fig. 2: Comparison of the SSL and SEFR feature rankings on the Tic-tac-toe, Arrhythmia and Gasdrift datasets, in terms of accuracy and F_1 measure of the 20NN classifier. The legend in (f) applies to all subfigures.

the uniform one (it is ranked 1st, 2nd and 3rd 12-times, 11-times, and 9-times respectively) and has the worst average (2.06).

Since all the feature ranking scores offer state-of-the-art performance, we may conclude that they are approximately equally good, so we can again make a decision based on the second criterion: computational complexity. Recall that random forests were the most appropriate method for the Symbolic score, whereas bagging was the most suitable for Genie3 and Random Forest score, which means that Symbolic rankings are computed $\sqrt{F}$-times faster than the Genie3 rankings. The Random Forest rankings demand even more time than Genie3 rankings, due

Table 4: Differences between the *auc* of SSL feature rankings and SEFR feature rankings, measured in terms of accuracy and F_1 measure, of 20NN classifier. If $\Delta > 0$, the SSL feature ranking outperforms the SEFR one.

	Symbolic		Genie3		Random Forest	
dataset	Δacc	ΔF_1	Δacc	ΔF_1	Δacc	ΔF_1
Arrhythmia	$2.9 \cdot 10^{-2}$	$6.1 \cdot 10^{-2}$	$2.9 \cdot 10^{-2}$	$7.1 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$	$5.4 \cdot 10^{-2}$
Bank	$6.0 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$	$8.5 \cdot 10^{-3}$	$1.1 \cdot 10^{-2}$	$-3.4 \cdot 10^{-3}$	$6.1 \cdot 10^{-3}$
Chess	$-2.4 \cdot 10^{-2}$	$-2.6 \cdot 10^{-2}$	$-8.6 \cdot 10^{-4}$	$-2.8 \cdot 10^{-3}$	$6.3 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$
Dis	$-8.9 \cdot 10^{-3}$	$1.3 \cdot 10^{-2}$	$-1.4 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$	$-2.6 \cdot 10^{-3}$	$3.5 \cdot 10^{-2}$
Gasdrift	$2.5 \cdot 10^{-3}$	$3.7 \cdot 10^{-3}$	$-1.6 \cdot 10^{-3}$	$-1.7 \cdot 10^{-3}$	$2.6 \cdot 10^{-3}$	$5.2 \cdot 10^{-3}$
Pageblocks	$8.5 \cdot 10^{-4}$	$1.2 \cdot 10^{-2}$	$2.5 \cdot 10^{-3}$	$4.0 \cdot 10^{-2}$	$-8.0 \cdot 10^{-4}$	$-5.1 \cdot 10^{-3}$
Phishing	$1.4 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$7.9 \cdot 10^{-3}$	$9.7 \cdot 10^{-3}$	$5.4 \cdot 10^{-3}$	$6.4 \cdot 10^{-3}$
Tic-tac-toe	$1.7 \cdot 10^{-2}$	$2.2 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$	$2.2 \cdot 10^{-2}$

to the permutations of feature values and evaluating each tree of the ensemble on the corresponding out-of-bag examples multiple times, as explained in Sec. 3.

6 Conclusions

In this paper, we propose three feature ranking scores (Symbolic, Genie3 and Random Forest). Each can be computed from three different ensembles (bagging, random forests and extremely randomized trees) of predictive-clustering trees (PCTs), which were adapted to the semi-supervised classification task. We evaluate the obtained feature rankings on 8 benchmark classification datasets. We first determine the most suitable ensemble for each of the scores. For the Symbolic score, these are random forests, whereas for the Genie3 and Random Forest score, this is bagging.

Next, we show that using unlabeled data leads to improvements in the ranking since the proposed semi-supervised feature rankings mostly outperform their supervised analogs, which use only labeled data. We analyzed the datasets where this does not hold by checking the validity of the clustering assumption of SSL and show that most probably the assumption is not valid in these cases. After that, we compare our feature ranking scores to the SEFR feature ranking method and empirically show that we consistently outperform this state-of-the-art baseline with every proposed feature ranking score. Finally, we compare the proposed scores among each other and conclude that they are all equally good. We suggest using the Symbolic score because it has the best (lowest) time complexity.

We will continue our work on this topic in three directions. We first plan to extend the number of the benchmark datasets and then to define and evaluate rankings that are obtained from gradient boosting ensembles. Next, we will extend the approach towards regression and more complex predictive modeling tasks including structured output prediction. Finally, we will work to circumvent the use of the internal cross-validation for determining the best value of w in the PCT heuristic by computing the ARI score instead.

7 Acknowledgements

We acknowledge the financial support of the Slovenian Research Agency via the grants P2-0103 and a young researcher grant to MP. SD and DK acknowledge the support by the Slovenian Research Agency (via grants J2-9230, and N2-0056), and the European Commission (project LANDMARK and The Human Brain Project SGA2). The computational experiments presented here were executed on a computing infrastructure from the Slovenian Grid (SLING) initiative.

References

1. Bellal, F., Elghazel, H., Aussem, A.: A semi-supervised feature ranking method with ensemble learning. Pattern Recognition Letters **33**(10), 1426–1433 (2012)
2. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
3. Blockeel, H.: Top-down Induction of First Order Logical Decision Trees. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium (1998)
4. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
5. Breiman, L., Friedman, J., Olshen, R., Stone, C.J.: Classification and Regression Trees. Chapman & Hall/CRC (1984)
6. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press (2010)
7. Geurts, P., Erns, D., Wehenkel, L.: Extremely randomized trees. Machine Learning **36**(1), 3–42 (2006)
8. Gijsbers, P.: OpenML repository (2017), `https://www.openml.org/d/40713`
9. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence **12**, 993–1001 (1990)
10. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2**(1), 193–218 (1985)
11. Huynh-Thu, V.A., Irrthum, A., Wehenkel, L., Geurts, P.: Inferring regulatory networks from expression data using tree-based methods. PLoS One **5**(9), 1–10 (2010)
12. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition **46**(3), 817–833 (2013)
13. Levatić, J.: Semi-supervised Learning for Structured Output Prediction. Ph.D. thesis, Jožef Stefan Postgraduate School, Ljubljana, Slovenia (2017)
14. Levatić, J., Ceci, M., Kocev, D., Džeroski, S.: Semi-supervised classification trees. Journal of Intelligent Information Systems **49**(3), 461–486 (2017)
15. Lichman, M.: UCI machine learning repository (2013), `http://archive.ics.uci.edu/ml`
16. Petković, M., Kocev, D., Džeroski, S.: Feature ranking for multi-target regression. Machine Learning Journal (2019), Accepted
17. Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught learning: Transfer learning from unlabeled data. In: 24th International Conference on Machine Learning. pp. 759–766. ACM (2007)
18. Sheikhpour, R., Sarram, M., Gharaghani, S., Chahooki, M.: A survey on semi-supervised feature selection methods. Pattern Recognition **64**(C), 141–158 (2017)
19. Wettschereck, D.: A Study of Distance Based Algorithms. Ph.D. thesis, Oregon State University, Corvallis, OR (1994)
20. Xu, Z., King, I., Lyu, M.R.T., Jin, R.: Discriminative semi-supervised feature selection via manifold regularization. Transactions on Neural Networks **21**(7), 1033–1047 (2010)

Machine Learning Journal manuscript No.
(will be inserted by the editor)

Feature Ranking for Semi-supervised Learning

Matej Petković · Sašo Džeroski · Dragi Kocev

Received: date / Accepted: date

Abstract The data made available for analysis are becoming more and more complex along several directions: high dimensionality, number of examples and the amount of labels per example. This poses a variety of challenges for the existing machine learning methods: coping with dataset with a large number of examples that are described in a high-dimensional space and not all examples have labels provided. For example, when investigating the toxicity of chemical compounds there are a lot of compounds available, that can be described with information rich high-dimensional representations, but not all of the compounds have information on their toxicity. To address these challenges, we propose semi-supervised learning of feature ranking. The feature rankings are learned in the context of classification and regression as well as in the context of structured output prediction (multi-label classification, hierarchical multi-label classification and multi-target regression). To the best of our knowledge, this is the first work that treats the task of feature ranking within the semi-supervised structured output prediction context. More specifically, we propose two approaches that are based on tree ensembles and the Relief family of algorithms. The extensive evaluation across 38 benchmark datasets reveals the following: Random Forests perform the best for the classification-like tasks, while for the regression-like tasks Extra-PCTs perform the best, Random Forests are the most efficient method considering induction times across all tasks, and semi-supervised feature rankings outperform their supervised counterpart across a majority of the datasets from the different tasks.

We acknowledge the financial support of the Slovenian Research Agency via the grant P2-0103 and a young researcher grant to MP. SD and DK also acknowledge the support by the Slovenian Research Agency (via grants J7-9400, J7-1815, J2-9230, and N2-0128), and the European Commission (project AI4EU).

M. Petković, S. Džeroski, D. Kocev
Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
Jožef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia
Tel.: +386-1477-3635
E-mail: {matej.petkovic,saso.dzeroski,dragi.kocev}@ijs.si

Keywords feature ranking · semi-supervised learning · tree ensembles · Relief · structured output prediction · multi-target prediction

1 Introduction

In the era of massive and complex data, predictive modeling is undergoing some significant changes. Since data are becoming ever more high dimensional, i.e., the target attribute potentially depends on a large number of descriptive attributes, there is a need to provide better understanding of the importance or *relevance* of the descriptive attributes for the target attribute. This is achieved through the task of feature ranking [Guyon and Elisseeff, 2003, Jong et al., 2004, Nilsson et al., 2007, Petković et al., 2019]: the output of a feature ranking algorithm is a list (also called a *feature ranking*) of the descriptive attributes ordered by their relevance to the target attribute. The obtained feature ranking can then be used in two contexts: (1) to better understand the relevance of the descriptive variables for the target variable or (2) as a frequent pre-processing step to reduce the number of descriptive variables. By performing the latter, not only the computational complexity of building a predictive model later on is decreased, but at the same time, the models that use a lesser number of features are easier to explain and understand which is of high importance in a variety of application domains such as medicine [Holzinger et al., 2019, Hoogendoorn et al., 2016, Tjoa and Guan, 2019], life sciences [Grissa et al., 2016, Saeys et al., 2007, Tsagris et al., 2018] and ecological modeling [Bhardwaj and Patra, 2018, Galelli et al., 2014, Zhou et al., 2018].

Another aspect of massiveness is the number of examples in the data. However, for some problems such as sentiment analysis of text, e.g., tweets [Kralj Novak et al., 2015], or determining properties of new chemical compounds [DiMasi et al., 2003], e.g., in QSAR (quantitative structure activity relationship) studies (which is one of the considered datasets in the experiments), one can only label a limited quantity of data, since labeling demands a lot of human effort and time (labelling tweets), or is expensive (performing wet lab QSAR experiments). Since the cases where many examples remain unlabeled are not that rare, advances in predictive modeling have brought us to the point where we can make use of them. In this work, we focus on semi-supervised learning (SSL) techniques that handle data where some examples are labeled and some are not (as opposed to supervised learning (SL) where all examples are labeled). Another direction of research goes into weakly supervised learning [Zhou, 2017] where all examples may be labeled but (some) labels may be inaccurate or of a lower quality.

The SSL approaches are all based on the assumption that the target values are well-reflected in the structure of the data, i.e.,

Assumption 1 (Clustering Hypothesis) *Clusters of data examples (as computed in the descriptive space) well resemble the distribution of target values.*

If the clustering hypothesis is satisfied, then a SSL algorithm that can make use of unlabeled data, may outperform the classical SL algorithms that simply ignore them. This holds for predictive modeling tasks [Levatić, 2017, Zhu et al., 2009], and as we show in this work, for feature ranking tasks also.

In addition to the massiveness, the complexity of the data is also increasing. Predictive modeling is no longer limited to the standard classification and regression, but also tackles their generalizations. For example, in classification, the target variable may take only one of the possible values, for each example in the data. On the other hand, problems such as automatic tagging (e.g., the `Emotions` dataset (see Sec. 6.2) where the task is to determine emotions that a given musical piece carries) allow for more than one label per example (e.g., a song can be *sad* and *dramatic* at the same time). A further generalization of this problem is hierarchical multi-label classification, where the possible labels are organized into a hierarchy, such as the one in Fig. 1, which shows animal-related labels. If a model labels an example as *koala*, it should also label it with the generalizations of this label, i.e., *Australian* and *animal*.

Similarly, the task of regression can be generalized to multi-target regression, i.e., predicting more than one numeric variable at the same time, e.g., predicting canopy density and height of trees in forests (the `Forestry` dataset in Sec. 6.2).

The main motivation for the generalized predictive modeling tasks is that considering all the target variables at the same time may exploit the potential interactions among them which are ignored when one predicts every variable separately. Moreover, building a single model for all targets can dramatically lower the computational costs.

In many cases, the data are at the same time semi-supervised (has missing), high dimensional and has a structured target, as for example in gene function prediction: Labeling genes with their functions is expensive (semi-supervision), the genes can be described with a large number of variables (high dimensionality), and the functions are organized into a hierarchy (structured target). Thus, designing feature ranking algorithms that i) can use unlabeled data, and ii) can handle a variety of target types, including structured ones, is a relevant task that we address in this work. To the best of our knowledge, this is the first work that treats jointly the task of feature ranking in the context of semi-supervised learning for structured outputs.

We propose two general feature ranking approaches. In the first approach, a ranking is computed from an ensemble of predictive clustering trees [Kocev et al., 2013, Blockeel, 1998], adapted to structured outputs and SSL [Levatić, 2017], whereas the second approach is based on the distance-based Relief family of algorithms [Kira and Rendell, 1992]. An initial study, investigated the performance of the ensemble-based approach in the classification task [Petković et al., 2019]. In this work, we substantially extend our previous study in several directions:

1. Additional datasets for classification are considered.

2. Additional four tasks are considered (multi-label and hierarchical multi-label classification, single- and multi-target regression), and the ensemble-based feature ranking methods are evaluated in these cases.
3. The Relief family of algorithms is extended to SSL, and evaluated for all five tasks (in comparison to the ensemble-based feature ranking methods).

The rest of the paper is organized as follows. In Sec. 2, we give the formal definitions of the different predictive modeling tasks, and introduce the notation. Sec. 3 surveys the related work, whereas Secs. 4 and 5 define the ensemble-based and Relief-based feature importance scores, respectively. Sec. 6 fully describes the experimental setup. We present and discuss the results in Sec. 7, and conclude with Sec. 8.

The implementation of the methods, as well as the results are available at `http://source.ijs.si/mpetkovic/ssl-ranking`.

2 Preliminaries

Basic notation. The data $\mathscr{D}$ consist of examples $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is a vector of values of D descriptive variables (features), and $\boldsymbol{y}$ is the value of the target variable(s). The domain $\mathcal{X}_i$ of the feature x_i is either numeric, i.e., $\mathcal{X}_i \subseteq \mathbb{R}$, or categorical, i.e., it is a finite set of categorical values, e.g., $\mathcal{X}_i = \{A, B, AB, 0\}$ if a feature describes blood type. Both numeric and categorical types are considered primitive *unstructured* types. The domain $\mathcal{Y}$ of the target variable depends on the predictive modeling task at hand. In this paper, we consider five tasks, two having unstructured, and three having structured target data types.

Regression (STR). In this case, the target is a single numeric variable. Since we later consider also its generalization (multi-target regression), we refer to this task as single-target regression (STR).

Multi-target regression (MTR). Here, the target variable is a vector with T numeric variables as components, i.e., $\mathcal{Y} \subseteq \mathbb{R}^T$. Equivalently, we can define MTR as having T numeric targets, hence the name. In the special case of $T = 1$, MTR boils down to STR.

Classification. In this case, the target is categorical. Since the algorithms considered in this paper can handle any classification task, we do not distinguish between binary ($|\mathcal{Y}| = 2$) and multi-class classification ($|\mathcal{Y}| > 2$).

Multi-label classification (MLC). The target domain is a power set $\mathcal{P}(\mathscr{L})$ of some set $\mathscr{L}$ of categorical values, whose elements are typically referred to as labels. Thus, the target values are sets. Typically, the target value $\boldsymbol{y}$ of the example $(\boldsymbol{x}, \boldsymbol{y})$ is referred to as a set of labels that are relevant for this example. The sets $\boldsymbol{y}$ can be of any cardinality, thus the labels are not mutually exclusive, as is the case with the task of (standard) classification.

Hierarchical multi-label classification (HMLC). This is a generalization of MLC where the domain is again a power set of some label set $\mathscr{L}$, which, additionally, is now partially-ordered via some ordering $\prec$. An exemplary hierarchy (of animal-related labels), which results from such an ordering is shown in the corresponding Haase diagram in Fig. 1.

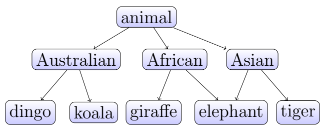


Fig. 1: An exemplary hierarchy of animal related labels.

If $\ell_1 \prec \ell_2$, the label ℓ_1 is predecessor of the label ℓ_2. If, additionally, there is no such label ℓ, such that $\ell_1 \prec \ell \prec \ell_2$, we say that ℓ_1 is a parent of ℓ_2. If a label does not have any parents, it is called a root. A hierarchy can be either tree-shaped, i.e., every label has at most one parent, or it can be directed acyclic graph (DAG). Since the label `elephant` has two parents (`African` and `Asian`), the hierarchy in Fig. 1 is not a tree.

Regarding predictive modeling, the ordering results in a hierarchical constraint, i.e., if a label ℓ is predicted to be relevant for a given example, then, also its predecessors must be predicted relevant, e.g., if a given example is `koala`, it must also be `Australian` and `animal`.

In the cases of MLC and HMLC, each set of relevant labels $S \subseteq \mathscr{L}$ is conveniently represented by the 0/1 vector $\boldsymbol{s}$ of length $|\mathscr{L}|$, whose j-th component equals one if and only if $\ell_j \in S$. Thus, we will also use the notation $T = |\mathscr{L}|$.

Semi-supervised learning (SSL). The unknown target values will be denoted by question marks (?). If the target value of the example is known, we say that the example is labeled, otherwise the example is unlabeled. This applies to all types of targets and is not to be confused with the labels in the tasks of MLC and HMLC.

3 Related Work

In general, feature ranking methods are divided into three groups [Stańczyk and Jain, 2015]. *Filter* methods do not need any underlying predictive model to compute the ranking. *Embedded* methods compute the ranking directly from some predictive model. *Wrapper* methods are more appropriate for feature selection, and build many predictive models which guide the selection.

Filters are typically the fastest but can be myopic, i.e., can neglect possible feature interactions, whereas the embedded methods are a bit slower, but can additionally serve as an explanation of the predictions of the underlying model. The prominence of the feature ranking reflects in numerous methods solving this task in the context of classification and STR [Guyon and Elisseeff, 2003, Stańczyk and Jain, 2015], however, the territory of feature ranking for SSL is mainly uncharted, especially when it comes to structured output prediction.

An overview of SSL feature ranking methods for classification and STR is given in [Sheikhpour et al., 2017]. However, the vast majority of the methods described there are either supervised or unsupervised (ignoring the labels completely). An exception is the SSL Laplacian score [Doquire and Verleysen, 2013], applicable to the STR problems.

This method is a filter and stems from graph theory. It first converts a dataset into a graph, encoded as a weighted incidence matrix whose weights correspond to the distances among the examples in the data. The distances are measured in the descriptive space but more weight is put on the labeled examples. One of the drawbacks of the original method is that it can only handle numeric features. Our modification that overcomes this is described in Sec. 6.6.

For structured output prediction in SSL, we could not find any competing feature ranking methods. Our ensemble-based scores belong to the group of embedded methods, and crucially depend on ensembles of SSL predictive clustering trees (PCTs) [Levatić, 2017]. We thus describe bellow SSL PCTs and ensembles thereof.

3.1 Predictive clustering trees

PCTs are a generalization of standard decision trees. They can handle various structured output prediction tasks and have been recently adapted to SSL [Levatić, 2017]. This work considers the SSL of PCTs for classification, STR, MTR [Levatić et al., 2018], MLC, and HMLC.

For each of these, one has to specify the impurity function *impu* that is used in the best test search (Alg. 2), and the prototype function *prototype* that creates the predictions in the leaf nodes. After these two are specified, a PCT is induced in the standard top-down-tree-induction manner.

Starting with the whole dataset $\mathscr{D}_{\text{TRAIN}}$, we find the test (Alg. 1, line 1) that greedily splits the data so that the heuristic score of the test, i.e., the decrease of the impurity *impu* of the data after applying the test, is maximized. For a given test, the corresponding decrease is computed in line 4 of Alg. 2.

If no useful test is found, the algorithm creates a leaf node and computes a leaf node with the prediction (Alg. 1, line 3). Otherwise, an internal node $\mathscr{N}$ with the chosen test is constructed, and the PCT-induction algorithm is recursively called on the subsets in the partition of the data, defined by the test. The resulting trees become child nodes of the node $\mathscr{N}$ (Alg 1, line 7).

Algorithm 1 PCT(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$
2: if $t^* = none$ then
3:     return $Leaf(prototype(E))$
4: else
5:     for each $E_i \in \mathcal{P}^*$ do
6:         $tree_i = \text{PCT}(E_i)$
7:     return $Node(t^*, \bigcup_i \{tree_i\})$
```

Algorithm 2 BestTest(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: for each test $t$ do
3:     $\mathcal{P}$ = partition induced by $t$ on $E$
4:     $h = |E| impu(E) - \sum_{E_i \in \mathcal{P}} |E_i| impu(E_i)$
5:     if $h > h^*$ then
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $(t^*, h^*, \mathcal{P}^*)$
```

The impurity functions for a given subset $E \subseteq \mathscr{D}_{\text{TRAIN}}$ in the considered tasks are defined as weighted averages of the feature impurities $impu(E, x_i)$, and target impurities $impu(E, y_j)$.

For nominal variables z, the impurity is defined in terms of the Gini Index $Gini(E, z) = 1 - \sum_v p_E^2(v)$, where the sum goes over the possible values v of the variable z, and $p_E(v)$ is the relative frequency of the value v in the subset E. In order not to favoritize any variable a priori, the impurity is defined as the normalized Gini value, i.e., $impu(E, z) = Gini(E, z)/Gini(\mathscr{D}_{\text{TRAIN}}, z)$. This applies to nominal features and the target in classification.

For numeric variables z, the impurity is defined in terms of their variance $Var(E, z)$, i.e., $impu(E, z) = Var(E, z)/Var(\mathscr{D}_{\text{TRAIN}}, z)$. This applies to numeric features and targets in other predictive modeling tasks, since the sets in MLC and HMLC are also represented by 0/1 vectors. However, note that computing the Gini-index of a binary variable is equivalent to computing the variance of this variable if the two values are mapped to 0 and 1. When computing the single-variable impurities, missing values are ignored.

In a fully-supervised scenario, the impurity of data is measured only on the target side. However, the majority of target values may be missing in the semi-supervised case. Therefore, for SSL, also the features are taken into account when calculating the impurity, which is defined as

$$impu(E) = w \cdot \frac{1}{T} \sum_{j=1}^{T} \alpha_j impu(E, y_j) + (1 - w) \cdot \frac{1}{D} \sum_{i=1}^{D} \beta_i impu(E, x_i), \qquad (1)$$

where the level of supervision is controlled by the user-defined parameter $w \in [0, 1]$. Setting it to 1 means fully-supervised tree-induction (and consequently ignoring unlabeled data). The other extreme, i.e., $w = 0$, corresponds to fully-unsupervised tree-induction (also known as clustering). The dimensional weights α_j and β_i are typically all set to 1, except for HMLC where $\alpha_i = 1$ for the roots of the hierarchy, and $\alpha_i = \alpha \cdot mean(\text{parent weights})$ otherwise, where $\alpha \in (0, 1)$ is a user-defined parameter. A MLC problem is considered a HMLC problem where all labels are roots.

The prototype function returns the majority class in the classification case, and the per-component mean $[\bar{y}_1, \ldots, \bar{y}_T]$ of target vectors otherwise. In all cases, the prototypes (predictions) are computed from the training examples in a given leaf. In the cases of MLC and HMLC, the values $\bar{y}_j$ can be additionally thresholded to obtain the actual subsets, i.e., $\hat{\boldsymbol{y}} = \{\ell_j \mid \bar{y}_j \geq \vartheta, 1 \leq j \leq T\}$, where taking $\vartheta = 0.5$ corresponds to computing majority values of each label.

3.2 Ensemble methods

To obtain a better predictive model, more than one tree can be grown, for a given dataset, which results in an ensemble of trees. Predictions of an ensemble are averaged predictions of trees (or, in general, arbitrary base models) in the ensemble. However, a necessary condition for an ensemble to outperform its base models is, that the base models are diverse [Hansen and Salamon, 1990]. To this end, some randomization must be introduced into the tree-induction process, and three ways to do so have been used [Levatić, 2017].

Bagging. When using this ensemble method, instead of growing the trees using $\mathscr{D}_{\text{TRAIN}}$, a bootstrap replicate of $\mathscr{D}_{\text{TRAIN}}$ is independently created for each tree, and used for tree induction.

Random Forests (RFs). In addition to the mechanism of Bagging, for each internal node of a given tree, only a random subset (of size $D' < D$) of all features is considered when searching for the best test, e.g., $D' = \mathit{ceil}(\sqrt{D})$.

Extremely Randomized PCTs (ETs). As in Random Forests, a subset of features can be considered in every internal node (this is not a necessity), but additionally, only one test per feature is randomly chosen and evaluated. In contrast to Random Forests (and Bagging), the authors of original ETs did not use bootstrapping [Geurts et al., 2006]. However, previous experiments [Petković et al., 2019] showed that it is beneficial to do so when the features are (mostly) binary, since otherwise ets can offer only one possible split and choosing one at random has no effect.

4 Ensemble-Based Feature Ranking

The three proposed importance scores can be all computed from a single PCT, but to stabilize the scores, they are rather computed from an ensemble: Since the trees are grown independently, the variance of each score $\mathit{importance}(x_i)$ decreases linearly with the number of trees.

Once an ensemble (for a given predictive modeling task) is built, we come to the main focus of this work: Computing a feature ranking out of it. There are three ways to do so: Symbolic [Petković et al., 2019], Genie3 [Petković et al., 2019] (its basic version (for standard classification and regression) was proposed in [Huynh-Thu et al., 2010]), and Random Forest score [Petković et al., 2019] (its basic version was proposed in [Breiman, 2001]):

$$importance_{\mathrm{SYMB}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} |E(\mathcal{N})| / |\mathcal{D}_{\mathrm{TRAIN}}|, \tag{2}$$

$$importance_{\mathrm{GENIE3}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} h^*(\mathcal{N}), \tag{3}$$

$$importance_{\mathrm{RF}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \frac{e(\mathrm{OOB}_{\mathcal{T}}^{i}) - e(\mathrm{OOB}_{\mathcal{T}})}{e(\mathrm{OOB}_{\mathcal{T}})}. \tag{4}$$

Here, $\mathcal{E}$ is an ensemble of trees $\mathcal{T}$, $\mathcal{T}(x_i)$ is the set of the internal nodes $\mathcal{N}$ of a tree $\mathcal{T}$ where the feature x_i appears in the test, $E(\mathcal{N}) \subseteq \mathcal{D}_{\mathrm{TRAIN}}$ is the set of examples that reach the node $\mathcal{N}$, h^* is the heuristic value of the chosen test, $e(\mathrm{OOB}_{\mathcal{T}})$ is the value of the error measure e, when using $\mathcal{T}$ as a predictive model for the set $\mathrm{OOB}_{\mathcal{T}}$ of the out-of-bag examples for a tree $\mathcal{T}$, i.e., examples that were not chosen into the bootstrap replicate, thus not seen during the induction of $\mathcal{T}$. Similarly, $e(\mathrm{OOB}_{\mathcal{T}}{}^{i})$ is the value of the error measure e on the $\mathrm{OOB}_{\mathcal{T}}$ with randomly permuted values of the feature x_i.

Thus, Symbolic and Genie3 ranking take into account node statistics: The Symbolic score's award is proportional to the number of examples that reach this node, while Genie3 is more sophisticated and takes into account also the heuristic value of the test (which is proportional to $|E(\mathcal{N})|$, see Alg. 2, line 4.

The Random Forest score, on the other hand, measures to what extent noising, i.e., permuting, the feature values decreases the predictive performance of the tree. In Eq. (4), it is assumed that e is a loss, i.e., lower is better as is the case, for example, in the regression problems where (relative root) mean squared errors are used. Otherwise, e.g., for classification tasks and the F_1 measure, the importance of a feature is defined as $-importance_{\mathrm{RF}}$ from Eq. (4). Originally, it was designed to explain the predictions of the RFs ensemble [Breiman, 2001] (hence the name), but it can be used with any predictive model. However, trees are especially appropriate, because the predictions can be obtained fast, provided the trees are balanced.

4.1 Ensemble-based ranking for SSL structured output prediction

The PCT ensemble-based feature ranking methods for different structured output prediction (SOP) tasks have been introduced by [Petković et al., 2019, Petković et al., 2020], and evaluated for different SL SOP tasks. In this case, PCTs use a heuristic based on the impurity reduction on the target space, as defined by Eq. (1), in a special case when $w = 1$. As for SSL, the general case of Eq. (1) applies. Once we have SSL PCTs, the ensemble-based feature ranking methods technically work by default. They have been evaluated in the case of SSL classification. However, they have not been evaluated on STR and SOP tasks.

4.2 Does the ensemble method matter?

From Eqs. (2)–(4), it is evident that all three feature ranking scores can in theory be computed from a single tree, and averaging them over the trees in the ensemble only gives a more stable estimate of $\mathbb{E}[\mathit{importance}(x_i)]$. However, one might expect that bagging, RFs and ETs on average yield the same order of features (or even the same importance values) since the latter two are more randomized versions of the bagging method. Here, we sketch a proof that this is not (necessarily) the case.

One of the cases when the expected orders of features are equal, is a dataset where each of the two binary features x_1 and x_2 completely determine the target y, e.g., $y = x_1$ and $y = 1 - x_2$, and the third feature is effectively random noise. It is clear that the expected values of the importances are in all cases $\mathit{importance}(x_1) = \mathit{importance}(x_2) > \mathit{importance}(x_3)$.

One of the cases where bagging gives rankings different from those of RFs, is a dataset where knowing the values of ranking pairs (x_1, x_2) and (x_3, x_i), for $4 \leq i \leq D$ again completely reconstructs the target value y, and $h(x_1) > h(x_i) > \max\{h(x_2), h(x_3)\}$, for $i \geq 4$. In this case, bagging will first choose x_1 and then x_2 in the remaining two internal nodes of the tree, so x_1 and x_2 would be the most important features. On the other hand, RFs with $D' = 1$ and D sufficiently large, will in the majority of the cases first choose one of the features x_i, $i \geq 4$, and then, sooner or later, x_3. Unlike in the bagging-based ranking, x_3 is now more important than x_1.

4.3 Time complexity

In predictive clustering, the attributes in the data belong to three (not mutually exclusive) categories: i) Descriptive attributes are those that can appear in tests of internal nodes of a tree, ii) Target attributes are those for which predictions in leaf nodes of a tree are made, and iii) Clustering attributes are those that are used in computing the heuristic when evaluating the candidate tests. Let their numbers be D, T and C, respectively, and let M be the number of examples in $\mathscr{D}_{\text{TRAIN}}$. Note that in the SSL scenario (if $w \notin \{0, 1\}$), we have the relation $C = D + T$. Assuming that the trees are balanced, we can deduce that growing a single semi-supervised tree takes $\mathcal{O}(MD' \log M(\log M + C))$ [Levatić, 2017].

After growing a tree, ranking scores are updated in $\mathcal{O}(M)$ time (where M is the number of internal nodes) for the Symbolic and Genie3 score, whereas updating the Random Forest scores takes $\mathcal{O}(DM \log M)$. Thus, computing the feature ranking scores does not change the $\mathcal{O}$-complexity of growing a tree, and we can compute all the rankings from a single ensemble. Thus, growing an ensemble $\mathcal{E}$ and computing the rankings takes $\mathcal{O}(|\mathcal{E}|MD' \log M(\log M + C))$.

5 Relief-based Feature Ranking

The RELIEF family of feature ranking algorithms does not use any predictive model. Its members can handle various predictive modeling tasks, including classification [Kira and Rendell, 1992], regression [Kononenko and Robnik-ikonja, 2003], MTR [Petković et al., 2019], MLC [Petković et al., 2018, Reyes et al., 2015], and HMLC [Petković et al., 2020]. The main intuition behind Relief is the following: the feature x_i is relevant if the differences in the target space between two neighboring examples are notable if and only if the differences in the feature values of x_i between these two examples are notable.

5.1 Supervised Relief

More precisely, if $\boldsymbol{r} = (\boldsymbol{x}^1, \boldsymbol{y}^1) \in \mathscr{D}_{\text{TRAIN}}$ is randomly chosen, and $\boldsymbol{n} = (\boldsymbol{x}^2, \boldsymbol{y}^2)$ is one of its nearest k neighbors, then the computed importances $\mathit{importance}_{\text{Relief}}(x_i)$ of the Relief algorithms equal the estimated value of

$$P_1 - P_2 = P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \mid \boldsymbol{y}^1 \neq \boldsymbol{y}^2) - P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \mid \boldsymbol{y}^1 = \boldsymbol{y}^2), \tag{5}$$

where the probabilities are modeled by the distances between $\boldsymbol{r}$ and $\boldsymbol{n}$ in appropriate subspaces. For the descriptive space $\mathcal{X}$ spanned by the domains $\mathcal{X}_i$ of the features x_i, we have

$$d_{\mathcal{X}}(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{1}{F}\sum_{i=1}^{F} d_i(\boldsymbol{x}^1, \boldsymbol{x}^2); \;\; d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) = \begin{cases} \mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2] & : \mathcal{X}_i \not\subseteq \mathbb{R} \\ \frac{|\boldsymbol{x}_i^1 - \boldsymbol{x}_i^2|}{\max_{\boldsymbol{x}} \boldsymbol{x}_i - \min_{\boldsymbol{x}} \boldsymbol{x}_i} & : \mathcal{X}_i \subseteq \mathbb{R} \end{cases} \tag{6}$$

where $\mathbf{1}$ denotes the indicator function. The definition of the target space distance $d_{\mathcal{Y}}$ depends on the target domain. In the cases of classification and MTR, the categorical and numeric part of the definition d_i in Eq. (6) apply, respectively. Similarly, in multi-target regression, $d_{\mathcal{Y}}$ is the analogue of $d_{\mathcal{X}}$ above.

In the cases MLC and HMLC, we have more than one option for the target distance definition [Petković et al., 2018], but in order to be as consistent as possible with the STR and MTR cases, we use the Hamming distance between the two sets. Recalling that sets $S \subseteq \mathscr{L}$ are presented as 0/1 vector $\boldsymbol{s}$ (Sec. 2), the Hamming distance $d_{\mathcal{Y}}$ is defined as

$$d_{\mathcal{Y}}(S^1, S^2) = \gamma \sum_{i=1}^{|\mathscr{L}|} \alpha_i \mathbf{1}[\boldsymbol{s}_i^1 \neq \boldsymbol{s}_i^2] \tag{7}$$

where the weights α_i are based on the hierarchy and are defined as in Eq. (1), and γ is the normalization factor that assures that $d_{\mathcal{Y}}$ maps to $[0, 1]$. It equals $\frac{1}{|\mathscr{L}|}$ in the MLC case, and depends on the data in the HMLC case [Petković et al., 2020].

To estimate the conditional probabilities $P_{1,2}$ from Eq. (5), they are first expressed in the unconditional form, e.g., $P_1 = P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \wedge \boldsymbol{y}^1 \neq \boldsymbol{y}^2)/P(\boldsymbol{y}^1 \neq \boldsymbol{y}^2)$. Then, the numerator is modeled as the product $d_i d_{\mathcal{Y}}$, whereas the nominator is modeled as $d_{\mathcal{Y}}$. The probability P_2 is estimated analogously.

5.2 Semi-supervised Relief

In the SSL version of the above tasks, we have to resort to the predictive clustering paradigm, using descriptive and clustering attributes instead of descriptive and target ones. More precisely, the descriptive distance is defined as above. As for the clustering distance, it equals $d_{\mathcal{Y}}$ when the target value of both $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$ are known, and equals $d_{\mathcal{X}}$ otherwise. The contribution of each pair to the estimate of probabilities is weighted according to their distance to the labeled data. The exact description of the algorithm is given in Alg. 3.

Algorithm 3 SSL-Relief($\mathscr{D}_{\text{TRAIN}}$, m, k, $[w_0, w_1]$)

```
1: **imp** = zero list of length $D$
2: $\boldsymbol{P}_{\text{diffAttr, diffCluster}}$, $\boldsymbol{P}_{\text{diffAttr}}$ = zero lists of length $D$
3: $P_{\text{diffCluster}} = 0.0$
4: $\boldsymbol{w} = computeInstanceInfluence(\mathscr{D}_{\text{TRAIN}}, w_0, w_1)$
5: $s = 0$                    # sum of weights of the pairs, used in normalization
6: **for** iteration $= 1, 2, \ldots, m$ **do**
7:     $\boldsymbol{r}$ = random example from $\mathscr{D}$
8:     $\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_k = k$ nearest neighbors of $\boldsymbol{r}$
9:     **for** $\ell = 1, 2, \ldots, k$ **do**
10:        $w = \boldsymbol{w}[\boldsymbol{r}] \cdot \boldsymbol{w}[\boldsymbol{n}_\ell]$
11:        $s \mathrel{+}= w$
12:        **if** $\boldsymbol{r}$ and $\boldsymbol{n}_\ell$ are labeled **then**
13:            $d_{\text{cluster}} = d_{\mathcal{Y}}(\boldsymbol{r}, \boldsymbol{n}_\ell)$
14:        **else**
15:            $d_{\text{cluster}} = d_{\mathcal{X}}(\boldsymbol{r}, \boldsymbol{n}_\ell)$
16:        $P_{\text{diffCluster}} \mathrel{+}= w\, d_{\text{cluster}}(\boldsymbol{r}, \boldsymbol{n}_\ell)$
17:        **for** $i = 1, 2, \ldots, D$ **do**
18:            $\boldsymbol{P}_{\text{diffAttr}}[i] \mathrel{+}= w\, d_i(\boldsymbol{r}, \boldsymbol{n}_\ell)$
19:            $\boldsymbol{P}_{\text{diffAttr, diffCluster}}[i] \mathrel{+}= w\, d_i(\boldsymbol{r}, \boldsymbol{n}_\ell)\, d_{\text{cluster}}(\boldsymbol{r}, \boldsymbol{n}_\ell)$
20: **for** $i = 1, 2, \ldots, D$ **do**
21:    **imp**$[i] = \frac{\boldsymbol{P}_{\text{diffAttr, diffCluster}}[i]}{P_{\text{diffCluster}}} - \frac{\boldsymbol{P}_{\text{diffAttr}}[i] - \boldsymbol{P}_{\text{diffAttr, diffCluster}}[i]}{s - P_{\text{diffCluster}}}$
22: **return imp**
```

SSL-Relief takes as input the standard parameters ($\mathscr{D}_{\text{TRAIN}}$, the number of iterations m, and the number of Relief neighbors k), as well as the interval $[w_0, w_1] \subseteq [0, 1]$, which the influence levels of $\boldsymbol{r}$-$\boldsymbol{n}$ pairs are computed from (line 4): First, for every $(\boldsymbol{x}, \boldsymbol{y}) \in \mathscr{D}_{\text{TRAIN}}$, we find the distance $d_{\boldsymbol{x}}$ to its nearest labeled neighbor. If $d = 0$, i.e., the value $\boldsymbol{y}$ is known, the influence w of this example is set to 1. Otherwise, the influence of the example is defined by a linear function $d \mapsto w(d)$ that goes through the points $(\max_{(\boldsymbol{x},?)} d_{\boldsymbol{x}}, w_0)$ and $(\min_{(\boldsymbol{x},?)} d_{\boldsymbol{x}}, w_1)$. Thus, the standard regression version of Relief is obtained when no target values are missing.

5.3 Time complexity

For technical reasons, the actual implementation of SSL-Relief does not follow the Alg. 3 word for word, and first computes all nearest neighbors. This takes $\mathcal{O}(mMD)$ steps, since the majority of the steps in this stage is needed for computing the distances in the descriptive space. We use the brute-force method, because it is, for the data at hand, still more efficient than, for example, k-D trees. Since the number of iterations is typically set to be a proportion of M (in our case $m = M$), the number of steps is $\mathcal{O}(M^2D)$. When computing the instances' influence (line 4), only the nearest neighbor of every instance is needed, so this can be done after the K-nearest neighbors are computed, within a negligible number of steps.

In the second stage, the probability estimates are computed and the worst-case time complexity is achieved when all examples are labeled since this is the case when we have to additionally compute $d_{\mathcal{Y}}$ (otherwise, we use the stored distances $d_{\mathcal{X}}$). The number of steps needed for a single computation od $d_{\mathcal{Y}}$ depends on the domain: $\mathcal{O}(1)$ suffices for classification and STR, whereas $\mathcal{O}(T)$ steps are required in the MTR, MLC and HMLC cases.

The estimate updates themselves take $\mathcal{O}(D)$ steps per neighbor, thus, the worst case time complexity is $\mathcal{O}(M^2D + kM(T + D)) = \mathcal{O}(M^2D + kMC)$ where $C = D + T$ is (again) the number of clustering attributes.

6 Experimental Setup

In this section, we undertake to experimentally evaluate the proposed feature ranking methods. We do so by answering a set of experimental questions listed below. We then describe in detail how the experimental evaluation is carried out.

6.1 Experimental questions

The evaluation is based on the following experimental questions:

1. For a given ensemble-based feature ranking score, which ensemble method is the most appropriate?
2. Are there any qualitative differences between the semi-supervised and supervised feature rankings?
3. Can the use of unlabeled data improve feature ranking?
4. Which feature ranking algorithm performs best?

6.2 Datasets

All datasets are well-known benchmark problems that come from different domains. For classification, we have included five new datasets (those below the splitting line of Tab. 1), in addition to the previous ones [Petković et al., 2019].

Since MLC can be seen as a special case of HMLC with a trivial hierarchy, we show the basic characteristics of the considered MLC and HMLC problems in a single table (Tab. 2), separating the MLC and HMLC datasets by a line.

Similarly, the regression problems (for STR and MTR) are shown in Tab. 3.

The given characteristics of the data differ from tasks to task, but the last column of every table (CH) always gives the estimate of how well the clustering hypothesis (Asm. 1) holds. For all predictive modeling tasks, this estimate is based on k-means clustering [Arthur and Vassilvitskii, 2007] or, more precisely, on the agreement between the distribution of the target values in these clusters. The number of clusters was set to the number of classes in the case of classification, and to 8 otherwise, i.e., the default Scikit Learn's [Pedregosa et al., 2011] parameters are kept. The highest agreement of the five runs of the method is reported.

CH computation. In the case of classification, the measure at hand is the Adjusted Random Index [Hubert and Arabie, 1985] (ARI) that we have already used earlier [Petković et al., 2019]. It computes the agreement between the classes that examples are assigned via clustering, and the actual class values. The optimal value of ARI is 1, whereas the value 0 corresponds to the case when clustering is independent of class distribution.

In the other cases, we compute the variance of each target variable, i.e., an actual target in the STR and MTR case, and a component of the 0/1 vector which a label set in the case of MLC and HMLC is represented by. Let $\mathcal{C}$ be the set of the obtained clusters, i.e., $c \subseteq \mathscr{D}$, for each cluster $c \in \mathcal{C}$. Then, for every target variable y_j, we compute $v_j = \sum_c p(c)\, \mathit{Var}(c, y_j) / \mathit{Var}(\mathscr{D}, y)$, i.e., the relative decrease of the variance after the clustering is applied, where $p(c) = |c|/|\mathscr{D}|$. It can be proved (using the standard formula for the estimation of sample variance and some algebraic manipulation) that $v_j \leq 1$. Trivially, $v_j \geq 0$. We average the contributions v_j over the target variables to obtain the score v. In the case of HMLC, we use weighted average where the weights are proportional to the hierarchical weights α_i, defined in Sec. 3.1. Finally, the tables report the values of $\mathit{CH} = 1 - v \in [0, 1]$, to make the value 1 optimal.

6.3 Parameter instantiation

We parametrize the used methods as follows. The number of trees in the ensembles was set to 100 [Kocev et al., 2013]. The number of features that are considered in each internal node was set to $\sqrt{D}$ for RFs and D for ETs [Geurts et al., 2006]. The optimal value of the level of supervision parameter w for computing the ensembles of PCTs was selected by internal 4-fold cross-validation. The considered values were $w \in \{0, 0.1, 0.2, \dots, 0.9, 1\}$.

The amount of supervision in SSL-Relief is adaptive, which allows for coarser set of values, and we consider $w_{1,2} \in \{0, 0.25, 0.5, 0.75, 1\}$ (where $w_1 \leq w_2$). The considered numbers k of Relief neighbors were $k \in \{15, 20, 30\}$, and the best hyper-parameter setting option (the values of w_1, w_2, and k) was again chosen via internal 4-fold cross-validation. Since more is better when the number of iterations m in Relief is concerned, this parameter was set to $m = |\mathscr{D}|$.

The possible numbers of labeled examples L in the training datasets were $L \in \{50, 100, 200, 350, 500\}$ [Levatić, 2017].

6.4 Evaluation pipeline

For the tasks of MLC and HMLC, the data come with predefined training and test parts ($\mathscr{D}_{\text{TRAIN}}$ and $\mathscr{D}_{\text{TEST}}$). This is not the case for the tasks of classification, STR and MTR, therefore, 10-fold cross validation is performed. To obtain the training-test pairs in cross-validation, we follow the procedure used by [Petković et al., 2019], as shown in Fig. 2.

Table 1: Basic properties of the classification datasets: number of examples $|\mathscr{D}|$, number of features D, number of classes (the y-domain size $|\mathcal{Y}|$), the proportion of examples in the majority class (MC), and the CH value.

dataset	$\lvert\mathscr{D}\rvert$	D	$\lvert\mathcal{Y}\rvert$	MC	CH
Arrhythmia [Lichman, 2013]	452	279	16	0.54	0.02
Bank [Lichman, 2013, Moro et al., 2011]	4521	16	2	0.88	-0.00
Chess [Lichman, 2013]	3196	36	2	0.52	0.22
Dis [Gijsbers, 2017]	3772	28	2	0.98	0.00
Gasdrift [Lichman, 2013]	13910	128	6	0.22	0.02
Pageblocks [Lichman, 2013]	5473	10	5	0.90	0.03
Phishing [Lichman, 2013]	11055	30	2	0.56	-0.00
Tic-tac-toe [Lichman, 2013]	958	9	2	0.65	0.70
Aapc [Džeroski et al., 1997]	335	84	3	0.47	0.34
Coil2000 [Van Der Putten and Van Someren, 2004]	9822	85	2	0.94	-0.00
Digits [Xu et al., 1992]	1797	64	10	0.10	-0.00
Pgp [Levatić et al., 2013]	932	183	2	0.52	0.00
Thyroid [Lichman, 2013]	3772	27	2	0.94	0.01

Table 2: Basic properties of the MLC (above the line) and HMLC (below the line) datasets: number of examples $|\mathscr{D}|$, number of features D, number of labels $|\mathscr{L}|$, label cardinality (average number of labels per example) ℓ_c, the depth of hierarchy, and the CH value.

dataset	$\lvert\mathscr{D}\rvert$	D	$\lvert\mathscr{L}\rvert$	ℓ_c	depth	shape	CH
Bibtex [Katakis et al., 2008]	7395	1836	159	2.4	1	tree	0.02
Birds [Briggs et al., 2013]	645	260	19	1.0	1	tree	0.05
Emotions [Trochidis et al., 2008]	593	72	6	1.9	1	tree	0.04
Genbase [Diplaris et al., 2005]	662	1185	27	1.3	1	tree	0.26
Medical [Pestian et al., 2007]	978	1449	45	1.3	1	tree	0.04
Scene [Boutell et al., 2004]	2407	294	6	1.1	1	tree	0.21
Clef07a-is [Dimitrovski et al., 2008]	11006	80	96	3.0	3.0	tree	0.05
Ecogen [Chen et al., 2004]	1893	138	56	15.5	3.0	tree	0.03
Enron-corr [Klimt and Yang, 2004]	1648	1001	67	5.3	3.0	tree	0.03
Expr-yeast-FUN [Clare, 2003]	3788	552	594	8.9	4.0	tree	0.00
Gasch1-yeast-FUN [Clare, 2003]	3773	173	594	8.9	4.0	tree	0.01
Pheno-yeast-FUN [Clare, 2003]	1592	69	594	9.1	4.0	tree	0.00

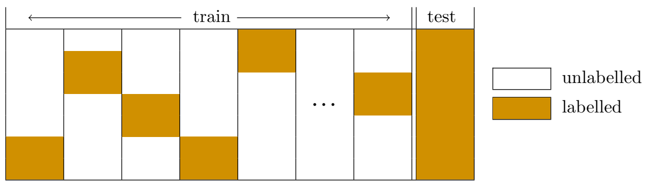


Fig. 2: Training and test set creation in SSL cross-validation: In the test fold, all examples keep their labels, whereas the folds that form the training set, together contain (approximately) L labeled examples.

Each dataset $\mathscr{D}$ is randomly split into $x = 10$ folds which results in the test sets $\mathscr{D}_{\text{TEST}i}$, $0 \leq i < x$. In contrast to cross-validation in the SL scenario, where $\mathscr{D}_{\text{TRAIN}i} = \cup_{j \neq i} \mathscr{D}_{\text{TEST}j}$, we first define the copy $\mathscr{D}_{\text{TEST}}{}_i^L$ of $\mathscr{D}_{\text{TEST}i}$ in which we keep the target values for $\lfloor L/(x-1) \rfloor + r_i$ randomly selected examples (orange parts of columns in Fig. 2) and remove the others (white parts). Here, $\lfloor \cdot \rfloor$ is the floor function, r is the reminder of L when divided by $x - 1$, and $r_i = 1$ if $i < r$ and 0 otherwise. This assures that every training set $\mathscr{D}_{\text{TRAIN}}{}_i^L = \cup_{j \neq i} \mathscr{D}_{\text{TEST}}{}_i^L$ contains a number of labeled examples as close as possible to L.

For the MLC and HMLC data, we can choose L labeled instances from the training set and delete the target values for the others. This is done for different numbers L of labeled examples, and we make sure that the implication $L_1 \leq L_2 \Rightarrow$ *labeled examples of* $\mathscr{D}_{TRAIN}{}_i^{L_1}$ *are a subset of the labeled examples in* $\mathscr{D}_{TRAIN}{}_i^{L_2}$ holds.

The ranking evaluation proceeds as follows. First, SSL-ranking is computed from $\mathscr{D}_{\text{TRAIN}}{}_i^L$ and its SL counterpart is computed on the $\mathscr{D}_{\text{TRAIN}}{}_i^L$ with the unlabeled examples removed. Afterward, both rankings are evaluated on $\mathscr{D}_{\text{TEST}}{}_i^L$ (in the cases of MLC and HMLC, $\mathscr{D}_{\text{TRAIN}}{}^L$ and $\mathscr{D}_{\text{TEST}}{}^L$ are used).

Table 3: Basic properties of the STR and MTR datasets: number of examples $|\mathscr{D}|$, number of features D, number of targets T, and the CH value.

dataset	examples	D	T	CH
CHEMBL2850 [Gijsbers, 2017]	1211	1024	1	0.09
CHEMBL2973 [Gijsbers, 2017]	1521	1024	1	0.18
Mortgage [Bilken University, 2020]	1049	15	1	0.57
Pol [Bilken University, 2020]	5000	26	1	0.12
QSAR [Gijsbers, 2017]	2145	1024	1	0.20
Treasury [Bilken University, 2020]	1049	15	1	0.54
Atp1d [Spyromitros-Xioufis et al., 2016]	337	411	6	0.49
CollembolaV2 [Kampichler et al., 2000]	393	47	3	0.02
Edm1 [Karalič and Bratko, 1997]	154	16	2	0.23
Forestry-LIDAR-IRS [Stojanova, 2009]	2730	28	2	0.19
Oes10 [Spyromitros-Xioufis et al., 2016]	403	298	16	0.63
Scm20d [Spyromitros-Xioufis et al., 2016]	8966	61	16	0.16
Soil-quality [Demšar et al., 2006]	1944	142	3	0.07

This is done by using the kNN algorithm with $k \in \{20, 40\}$ where weighted version of the standard squared Euclidean distance is used. For two input vectors $\boldsymbol{x}^1$ in $\boldsymbol{x}^2$, the distance d between them is defined as $d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sum_{i=1}^{D} w_i d_i^2(\boldsymbol{x}_i^1, \boldsymbol{x}_i^2)$, where d_i is defined as in Eq. (6). The dimensional weights w_i are defined as $w_i = \max\{\mathit{importance}(x_i), 0\}$, since Random Forest and Relief ranking can award a feature a negative score. In the degenerated case when the resulting values all equal 0, we define $w_i = 1$, for all features x_i. The first step is necessary to ignore the features that are of lower importance than a randomly generated one would be. The second step is necessary to ensure d is well-defined. We chose more than one value of k to show the qualitative differences between the supervised and semi-supervised feature rankings.

The evaluation through kNN was chosen because of three main reasons. First it can be used for all the considered predictive modeling tasks. Second, this is a distance based method, hence, it can easily make use of the information contained in the feature importances in the learning phase. Third, kNN is simple: Its only parameter is the number of neighbors. In the prediction stage, the neighbors' contributions to the predicted value are equally weighted, so we do not introduce additional parameters that would influence the performance.

6.5 Evaluation measures

To asses the predictive performance of a kNN model, the following evaluation measures are used: F_1 for classification (macro-averaged for multi-class problems), Root Relative Squared Error (RRMSE) for STR and MTR, and area under the average precision-recall curve for MLC and HMLC (AU $\overline{\mathrm{PRC}}$). Their definitions are given in the Tab. 4. In the cross-validation setting, we average the scores over the folds (taking test set sizes into account).

Table 4: Evaluation measures, for different predictive modeling tasks. The F_1 measure and AU $\overline{\mathrm{PRC}}$ are defined in terms of precision $p = \mathit{tp}/(\mathit{tp} + \mathit{fp})$ and recall $r = \mathit{tp}/(\mathit{tp} + \mathit{fn})$, where the numbers tp, fp and fn denote the number of true positive, false positive and false negative examples, respectively.

tasks	measure	definition
classification	F_1	$2/(1/p + 1/r)$
MLC, HMLC	AU $\overline{\mathrm{PRC}}$	area under the micro-averaged precision-recall curve
STR, MTR	RRMSE	$\frac{1}{T}\sum_{j=1}^{T}\sqrt{\frac{1}{\lvert\mathscr{D}_{\mathrm{TEST}}\rvert}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathscr{D}_{\mathrm{TEST}}}\frac{(\hat{\boldsymbol{y}}_j-\boldsymbol{y}_j)^2}{\mathit{Var}(\mathscr{D}_{\mathrm{TEST}},\boldsymbol{y}_j)}}$

For each ranking and dataset, we construct a curve that consist of points $(L, \mathit{performance}_L)$. The comparison of two methods is then based either i) on these curves directly (see Fig. 3), or ii) on the area under the computed curves.

6.6 The considered methods

The methods that our proposed methods are compared to, depend on the predictive modeling task:

– Classification: We have shown [Petković et al., 2019] that ensemble-based ranking algorithms have state-of-the-art performance. Thus, their and Relief's SSL and SL versions are compared against each other.
– STR: As mentioned before (Sec. 3), the existing SSL state-of-the-art competitor is Laplace, thus we compare Laplace, and the SL/SSL versions of both ensemble-based rankings and Relief-based rankings, against each other.
– MTR, MLC, HMLC: To the best of our knowledge, there are no existing methods that can perform feature ranking in the SSL structured output prediction scenarios, thus, we compare both versions of ensemble-based rankings and Relief-based rankings against each other.

Despite our best efforts, we could not obtain any existing implementation of the Laplace method, so we provide ours together with the rest of the code. Also note that the ensemble-based and Relief-based methods work out of the box, i.e., no data preprocessing is necessary, whereas by design, Laplace can handle only numeric features. To overcome this issue, we extend the method by the following procedure: i) transform the nominal features using 1-hot encoding, ii) compute the Laplace scores s_i, iii) for the originally nominal features x_i, define their score s_i as the sum of the scores of the corresponding 1-hot encoded features, and, finally, iv) define the importance scores $\mathit{importance}_{\text{Laplace}}(x_i) = S + s - s_i$ (where S and s denote the maximum and the minimum of the scores, respectively). The last step is necessary since less is better, for the originally computed Laplace scores. The transformation $s_i \mapsto S + s - s_i$ maps S to s and vice-versa, thus, the scale remains intact. The other problem of the method are constant features (they cause 0/0 values), present, for example, in QSAR data: These had to be manually removed.

7 Results

Unless stated otherwise, the rankings are compared in terms of the areas under the performance curves (see Sec. 6.5). When a SSL-ranking is compared to a SL-ranking, and the difference $\varDelta$ between the two performances is computed, $\varDelta > 0$ always corresponds to the case when the SSL-ranking performs better.

7.1 The optimal ensemble method for ensemble-based ranking

We first determine the most appropriate ensemble method, for each of the three ensemble scores, and their two versions (SSL and SL). The results in Tab. 5 give the average ranks of the ensemble methods in each setting, in terms of the areas under the performance curves.

Table 5: Average ranks of the considered SSL and SL ensembles, for a fixed ensemble-based score and predictive modeling task. The best ranks are shown in bold, unless all three methods perform equally well. In the case of ties, we bold the most efficient method (see Tab. 6).

task	score	SSL ensemble			SL ensemble		
		RFs	ETs	bagging	RFs	ETs	bagging
classification	Genie3	2.00	2.15	**1.85**	**1.69**	2.46	1.85
	Random Forest	**1.92**	2.15	1.92	2.00	2.08	**1.92**
	Symbolic	**1.77**	2.23	2.00	**1.77**	2.23	2.00
MLC	Genie3	**1.50**	2.67	1.83	2.17	2.17	**1.67**
	Random Forest	**2.00**	2.00	2.00	**1.67**	2.00	2.33
	Symbolic	**1.33**	2.33	2.33	2.00	2.50	**1.50**
HMLC	Genie3	**1.67**	2.00	2.33	**1.67**	1.83	2.50
	Random Forest	2.17	**1.83**	2.00	1.83	**1.67**	2.50
	Symbolic	2.17	2.00	**1.83**	**1.67**	2.00	2.33
STR	Genie3	**1.67**	2.00	2.33	2.17	2.00	**1.83**
	Random Forest	2.00	**1.83**	2.17	2.67	**1.67**	1.67
	Symbolic	2.17	**1.50**	2.33	2.33	**1.83**	1.83
MTR	Genie3	2.14	**1.86**	2.00	2.43	**1.71**	1.86
	Random Forest	2.29	2.14	**1.57**	2.43	**1.71**	1.86
	Symbolic	**2.00**	2.00	2.00	2.29	**1.71**	2.00

We observe that for both regression tasks (STR and MTR), RFs ensembles almost never perform best (with the exception of Genie3 SSL-rankings), whereas for the other three classification-like tasks, they quite consistently outperform the other two ensemble methods. The differences among the average ranks are typically not considerable (with the exception of the most of the MLC rankings, and supervised MTR rankings) which is probably due to the fact that the split selection mechanisms of the considered ensemble methods are still quite similar, and the trees are fully-grown, so sooner or later, a relevant feature appears in the node. In the case of ties, we choose the more efficient one (see Tab. 6): RFs are always the most efficient, whereas the second place is determined by the number of possible splits per feature. For lower values (e.g., when most of the features are binary, as is the case in MLC and HMLC data), bagging is faster than ETs.

Table 6: Average ranks of the ensemble methods, in terms of induction times.

task	RFs	ETs	bagging
classification	**1.00**	2.23	2.77
MLC	**1.00**	2.67	2.33
HMLC	**1.00**	2.50	2.50
STR	**1.17**	2.33	2.50
MTR	**1.29**	1.71	3.00

To make the later graphs more readable, we plot, for every score, only the curve that corresponds to the most suitable ensemble method for this score.

7.2 Qualitative difference between SSL and SL rankings

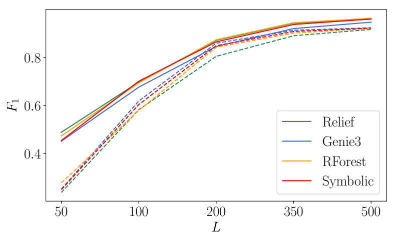
We first discuss the qualitative difference between the SSL-rankings and their supervised counterparts. In the process of obtaining a feature ranking, the SSL-version of the ranking algorithm sees more examples than its supervised version, and it turns out that this is well-reflected in the results. Fig. 3 shows the results for five datasets (one dataset, for each task) and the performance of the rankings, as assessed by kNN models, for $k \in \{20, 40\}$. Those two values of k are used to show that SSL-rankings tend to capture a more global picture of data, whereas the supervised ones reflect a more local one. This phenomenon

Table 7: Proportions of the computed feature rankings whose SSL-version captures more global properties of the data, as compared to its supervised version. The differences δ_{20} and δ_{40} of the areas under the performance curves of 20NN and 40NN models are computed (always in a way that $\delta > 0$ means that SSL-version performs better). Therefore, if $\varDelta = \delta_{40} - \delta_{20} > 0$, then the SSL-version of the ranking is more global, and is more local if $\varDelta < 0$.

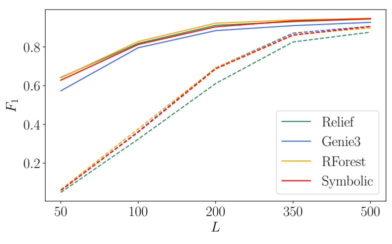
task	classification	MLC	HMLC	STR	MTR
$P[\varDelta > 0]$	0.73	0.83	0.96	1.00	0.93

is most visible in the two regression datasets. In the case of the `treasury` dataset, SSL-rankings perform worse than supervised ones on the local scale for smaller numbers L of labeled examples (Fig. 3g), and are equal or better for $L \geq 200$. However, on the global scale (Fig. 3h), the SSL-rankings are clear winners. A similar situation is observed for the other datasets in Fig. 3, and also in general.

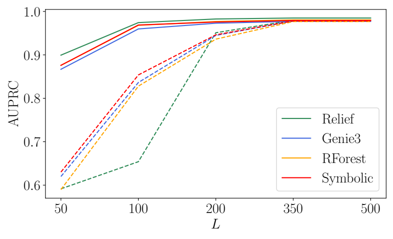
Tab. 7 reveals that for the vast majority of the rankings (and datasets), the SSL rankings are more global. This proportion is the highest for STR data (it even equals 100%), and is understandably the lowest for classification, where the datasets have the smallest number of examples on average.



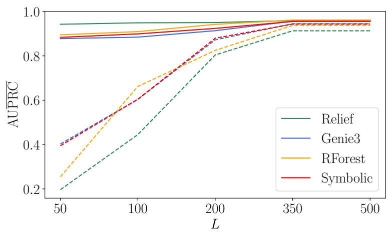
(a) classification: digits, 20NN



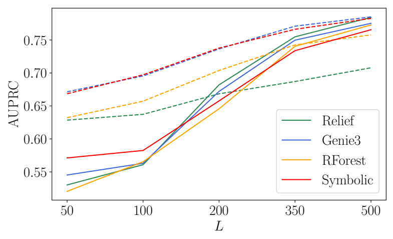
(b) classification: digits, 40NN



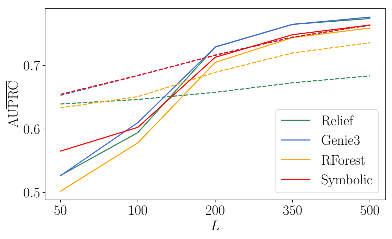
(c) MLC: genbase, 20NN



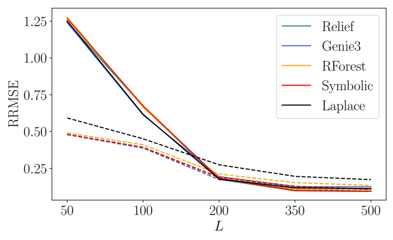
(d) MLC: genbase, 40NN



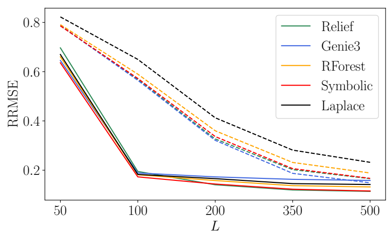
(e) HMLC: ecogen, 20NN



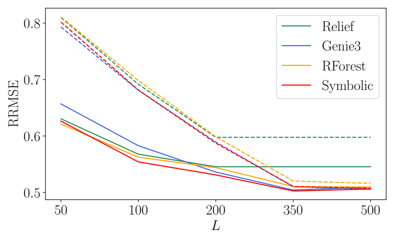
(f) HMLC: ecogen, 40NN



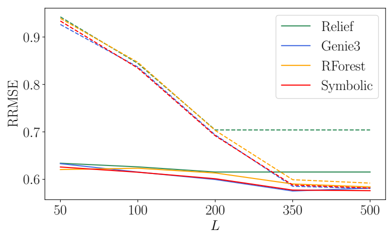
(g) STR: treasury, 20NN



(h) STR: treasury, 40NN



(i) MTR: oes10, 20NN



(j) MTR: oes10, 40NN

Fig. 3: Comparison of the SL and SSL feature rankings, for different predictive modeling tasks. The curves for the SSL and the SL versions of a ranking are shown as a solid and a dashed line of the same color. The graphs in the left column use 20NN models in the evaluation, whereas those in the right, use 40NN models.

7.3 Can unlabeled data improve feature rankings?

To answer this question, we compare the SSL versions of the proposed feature rankings to their supervised counterparts. In the previous section, we explained why sometimes the answer is not straightforward and depends on whether one is interested in a global or local scale. Since the question is whether the ranking can be improved by using unlabeled data, and given the qualitative differences between the SSL- and SL-versions of the rankings from the previous section, we fix the number of neighbors to $k = 40$.

We start with the classification results given in Tab. 8.

Table 8: The differences Δ of areas under the curves of F_1-values of the 40NN models with distance weights based on SSL-rankings and SL-rankings.

	datasets	Genie3	RForest	Symbolic	Relief	CH
classification	Arrhythmia	0.039	0.008	0.022	0.006	0.02
	Bank	0.064	0.067	0.061	0.050	-0.00
	Chess	-0.084	0.021	-0.081	0.022	0.22
	Dis	0.066	0.046	0.050	0.123	0.00
	Gasdrift	0.041	0.038	0.053	0.109	0.02
	Pageblocks	0.272	0.250	0.250	0.243	0.03
	Phishing	-0.125	-0.128	-0.132	-0.115	-0.00
	Tic-tac-toe	0.148	0.225	0.152	0.141	0.70
	Aapc	0.115	0.041	0.067	0.110	0.34
	Coil2000	0.019	0.029	0.022	0.020	-0.00
	Digits	0.170	0.204	0.198	0.245	-0.00
	Pgp	0.043	0.113	0.096	0.139	0.00
	Thyroid	0.288	0.268	0.285	0.265	0.01

From the mainly positive numbers in the table, one can conclude that SSL-rankings successfully recognize the structure of data, and outperform their supervised analogs, even in most of the cases where the CH values are low, e.g., for `digits` dataset in Fig. 3a, or, most notably, for `pageblocs`.

Continuing with the results for MLC (the upper part of Tab. 9), we first see that CH values are rather low, since, in contrast to the ARI values from classification, correction for chance is not incorporated into these CH values. An exception to this are the `genbase` (see Figs. 3c and 3d) and the `scene` dataset. For both datasets, the SSL-versions of the rankings outperform their SL-analogs. This also holds for the `birds` and `emotions` datasets, for all rankings, and additionally for the `medical` dataset in the case of Relief.

The bottom part of Tab. 9 gives the results for HMLC datasets. One can notice that Asm. 1 is never satisfied (low CH values), and that SSL-scores mostly could not overcome this, with the exception of Relief rankings on the `ecogen` dataset. However, inspecting the corresponding curves in detail (Fig. 3f), reveals that the negative differences in the performance of SSL-rankings and SL-rankings are mostly due to the bad start of SSL-rankings: For $L \geq 200$, the SSL-versions prevail.

Table 9: The differences $\varDelta$ of areas under the curves of AU $\overline{\text{PRC}}$-values of the 40NN models whose distance weights base on SSL-ranking and SL-ranking.

	datasets	Genie3	RForest	Symbolic	Relief	CH
MLC	Bibtex	-0.115	-0.078	-0.100	-0.100	0.02
	Birds	0.039	0.052	0.021	0.029	0.05
	Emotions	0.012	0.028	0.011	0.044	0.04
	Genbase	0.091	0.121	0.094	0.189	0.26
	Medical	-0.067	0.008	-0.058	0.014	0.04
	Scene	0.045	0.048	0.063	0.119	0.21
HMLC	Clef07a-is	-0.097	-0.066	-0.102	-0.041	0.05
	Ecogen	-0.007	-0.003	-0.018	0.051	0.03
	Enron-corr	-0.068	-0.062	-0.023	-0.064	0.03
	Expr-yeast-fun	-0.090	-0.103	-0.086	-0.071	0.00
	Gasch1-yeast-FUN	-0.080	-0.087	-0.084	-0.096	0.01
	Pheno-yeast-FUN	-0.032	-0.031	-0.036	-0.029	0.00

Table 10: The differences $\varDelta$ of areas under the curves of RRMSE-values of the 40NN models with distance weights based on SSL-rankings and SL-rankings.

	datasets	Genie3	RForest	Symbolic	Relief	Laplace	CH
STR	CHEMBL2850	-0.047	-0.063	0.014	-0.092	-0.010	0.09
	CHEMBL2973	-0.143	-0.103	-0.114	-0.168	-0.109	0.18
	Mortgage	0.074	0.092	0.097	0.078	0.120	0.57
	Pol	0.027	0.249	0.127	-0.049	0.278	0.12
	QSAR	-0.347	-0.446	-0.442	-0.523	-0.262	0.20
	Treasury	0.118	0.172	0.165	0.155	0.215	0.54
MTR	Atp1d	0.048	0.024	0.048	0.093		0.49
	CollembolaV2	-0.048	-0.014	-0.010	-0.002		0.02
	Edm1	0.002	0.018	0.004	0.006		0.23
	Forestry-LIDAR-IRS	-0.115	-0.070	-0.101	-0.114		0.19
	Oes10	0.083	0.084	0.083	0.122		0.63
	Scm20d	-2.357	-2.317	-2.295	-2.281		0.16
	Soil-quality	-0.044	-0.085	-0.080	-0.111		0.07

We finish this section with the regression results. The upper part of Tab. 10 shows that when CH is well-setisfied, i.e., for the datasets `mortgage` and `treasury` (see Fig. 3h), the SSL-rankings outperform the SL-rankings. Moreover, this also holds for the `pol` data (except for the Relief rankings). Inspecting the datasets where negative values are present (most notably the `qsar` dataset) reveals the same phenomenon as in HMLC case: for extremely low values of L, e.g., $L = 50$, the SSL-rankings do not perform well, possibly because knowing the labels of 50 out of approximately 2000 examples simply does not suffice. With more and more labels known, the performance of SSL-rankings drastically improves, while the performance of SL-rankings stagnates. Finally, for $L \geq 200$ or $L \geq 350$, all SSL-rankings again outperform the SL-ones.

Similar findings hold for the MTR data and the results in the bottom part of Tab. 10. The SSL-rankings perform well from the very beginning on the three datasets where CH holds the most, i.e., `oes10` (see Fig. 3j), `atp1d`, and `edm1`, but can only catch up with the SL-rankings (and possibly outperform them) for larger values of L in the other cases.

7.4 Which SSL-ranking performs best?

To answer this question, we compare the predictive performances of the corresponding 40NN models and report their ranks in Tab. 11. The results reveal that, for majority of the tasks, ensemble-based rankings perform best, however, in some cases, the winners are not clear, e.g., in the case of the classification. Still, Symbolic ranking quite clearly outperforms the others on both regression tasks, STR and MTR.

To complement this analysis, we also compute the average ranks of the algorithms for their induction times. As explained in Sec. 7.1, for the ensemble-

Table 11: The average ranks of different SSL-ranking algorithms that base on the performance of the corresponding 40NN models.

task	Genie3	Random Forest	Symbolic	Relief	Laplace
classification	2.62	2.46	2.62	**2.31**	
MLC	3.00	**2.17**	2.50	2.33	
HMLC	**2.00**	2.83	2.50	2.67	
STR	3.00	3.33	**1.83**	4.17	2.67
MTR	2.57	2.57	**1.71**	3.14	

based rankings, RFs are always preferable in terms of speed. They can still be outperformed by Relief if the number of features is higher and the number of examples is moderate, which follows directly from the $\mathcal{O}$-values in Secs. 4.3 and 5.3. All these methods are implemented in the Clus system (Java), whereas our implementation of the Laplace score is, as mentioned before, Python-based (Scikit Learn and numpy). Thus, even though Laplace and Relief have the same core operations (finding nearest neighbors), using higly-optimized Scikit Learn's methods (such as kNN) puts Laplace at the first place, whereas Relief is (second but) last, for STR problems.

Table 12: The average ranks of different SSL-ranking algorithms in terms of their induction times. Since the time complexity of ensemble-based rankings (almost) equals the induction time of the ensembles, we report the latter. For each task, we show the ranks for both extreme values of L.

task	L	RFs	ETs	bagging	Relief	Laplace
classification	50	**1.15**	2.46	3.15	3.23	
	500	**1.31**	2.69	3.54	2.46	
MLC	50	1.67	3.50	3.33	**1.50**	
	500	2.00	3.00	3.67	**1.33**	
HMLC	50	**1.17**	2.83	3.17	2.83	
	500	**1.33**	3.00	3.50	2.17	
STR	50	2.17	3.50	3.83	4.50	**1.00**
	500	2.67	3.17	4.67	3.50	**1.00**
MTR	50	**1.86**	2.29	3.71	2.14	
	500	**1.71**	2.43	3.71	2.14	

8 Conclusions

In this work, we focus on **semi-supervised learning of feature ranking**. The feature rankings are learned in the context of simple (single-target) classification and regression as well as in the context of structured output prediction (multi-label classification, hierarchical multi-label classification and multi-target regression). This is the first work that treats the task of feature ranking within the semi-supervised structured output prediction - it treats all the different prediction tasks in an unified way.

We propose, develop and evaluate **two approaches for SSL feature ranking for SOP** based on tree ensembles and the Relief family of algorithms. The tree ensemble-based rankings can be learned using three ensemble learning methods (Bagging, Random Forests, Extra Trees) coupled with three scoring functions (Genie3, Symbolic and random forest scoring). The Relief-based rankings use the regression variant of the Relief algorithm for extension towards the SOP tasks. This is the first extension of a Relief algorithm towards semi-supervised learning.

An **experimental evaluation of the proposed methods is carried out on 38 benchmark datasets** from the five machine learning tasks: 13 from classification, 6 from multi-label classification, 6 from hierarchical multi-label classification, 6 from regression and 7 from multi-target regression. Whenever available, we compared the performance of the proposed methods to the performance of state-of-the-art methods. Furthermore, we compared the performance of the semi-supervised feature ranking methods with their supervised counterparts.

The results from the extensive evaluation are best summarized through the answers of the research questions:

1. *For a given ensemble-based feature ranking score, which ensemble method is the most appropriate?*
   Generally, Random Forests perform the best for the classification-like tasks (classification, muilti-label classification and hierarchical multi-label classification), while for the regression-like tasks (regression, multi-target regression) Extra-PCTs perform the best. Furthermore, across all tasks, Random Forests are the most efficient method considering induction times.
2. *Are there any qualitative differences between the semi-supervised and supervised feature rankings?*
   The semi-supervised rankings tend to capture a more global picture of the data, whereas the supervised ones reflect a more local one.
3. *Can the use of unlabeled data improve feature ranking?*
   Semi-supervised feature rankings outperform their supervised counterpart across a majority of the datasets from the different tasks.
4. *Which feature ranking algorithm performs best?*
   Different SSL feature ranking methods perform the best for the different tasks: Symbolic ranking is the best for the regression and multi-target regression, Random forest ranking for multi-label classification, Genie3 for hierarchical multi-label classification, and Relief for classification.

Acknowledgements The computational experiments presented here were executed on a computing infrastructure from the Slovenian Grid (SLING) initiative, and we thank the administrators Barbara Krašovec and Janez Srakar for their assistance.

References

Arthur and Vassilvitskii, 2007. Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 07, page 10271035, USA. Society for Industrial and Applied Mathematics.

Bhardwaj and Patra, 2018. Bhardwaj, K. and Patra, S. (2018). An unsupervised technique for optimal feature selection in attribute profiles for spectral-spatial classification of hyperspectral images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 138:139–150.

Bilken University, 2020. Bilken University (2020). Function approximation repository. Accessible at `http://funapp.cs.bilkent.edu.tr/DataSets/`.

Blockeel, 1998. Blockeel, H. (1998). *Top-down Induction of First Order Logical Decision Trees*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium.

Boutell et al., 2004. Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

Breiman, 2001. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Briggs et al., 2013. Briggs, F., Huang, Y., Raich, R., Eftaxias, K., Lei, Z., Cukierski, W., Frey Hadley, S., Hadley, A., Betts, M., Fern, X. Z., Irvine, J., Neal, L., Thomas, A., Fodor, G., Tsoumakas, G., Ng Hong, W., Nguyen, T. N. T., Huttunen, H., Ruusuvuori, P., Manninen, T., Diment, A., Virtanen, T., Marzat, J., Defretin, J., Callender, D., Hurlburt, C., Larrey, K., and Milakov, M. (2013). The 9th annual mlsp competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2013*, pages 1–8.

Chen et al., 2004. Chen, B.-J., Chang, M.-W., and lin, C.-J. (2004). Load forecasting using support vector machines: A study on EUNITE competition 2001. *IEEE Transactions on Power Systems*, 19(4):1821–1830.

Clare, 2003. Clare, A. (2003). *Machine learning and data mining for yeast functional genomics*. PhD thesis, University of Wales Aberystwyth, Aberystwyth, Wales, UK.

Demšar et al., 2006. Demšar, D., Džeroski, S., Larsen, T., Struyf, J., Axelsen, J., Bruus, M., and Krogh, P. H. (2006). Using multi-objective classification to model communities of soil microarthropods. *Ecological Modelling*, 191:131–143.

DiMasi et al., 2003. DiMasi, J. A., Hansen, R. W., and Grabowski, H. G. (2003). The price of innovation: new estimates of drug development costs. *Journal of Health Economics*, 22(2):151 – 185.

Dimitrovski et al., 2008. Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2008). Hierchical annotation of medical images. In *Proceedings of the 11th International Multiconference - Information Society IS 2008*, pages 174–181. IJS, Ljubljana.

Diplaris et al., 2005. Diplaris, S., Tsoumakas, G., Mitkas, P., and Vlahavas, I. (2005). Protein classification with multiple algorithms. In *10th Panhellenic Conference on Informatics (PCI 2005)*, pages 448–456.

Doquire and Verleysen, 2013. Doquire, G. and Verleysen, M. (2013). A graph laplacian based approach to semi-supervised feature selection for regression problems. *Neurocomputing*, 121:5–13.

Džeroski et al., 1997. Džeroski, S., Potamias, G., Moustakis, V., and Charissis, G. (1997). Automated revision of expert rules for treating acute abdominal pain in children. In *Proceedings of the 6th Conference on Artificial Intelligence in Medicine in Europe*, AIME '97, page 98109, Berlin, Heidelberg. Springer-Verlag.

Galelli et al., 2014. Galelli, S., Humphrey, G. B., Maier, H. R., Castelletti, A., Dandy, G. C., and Gibbs, M. S. (2014). An evaluation framework for input variable selection algorithms for environmental data-driven models. *Environmental Modelling & Software*, 62:33 – 51.

Geurts et al., 2006. Geurts, P., Erns, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 36(1):3–42.

Gijsbers, 2017. Gijsbers, P. (2017). Dis data. Retrieved from OpenML repository `https://www.openml.org/d/40713`.

Grissa et al., 2016. Grissa, D., Ptra, M., Brandolini, M., Napoli, A., Comte, B., and Pujos-Guillot, E. (2016). Feature selection methods for early predictive biomarker discovery using untargeted metabolomic data. *Frontiers in Molecular Biosciences*, 3:30.

Guyon and Elisseeff, 2003. Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

Hansen and Salamon, 1990. Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001.

Holzinger et al., 2019. Holzinger, A., Langs, G., Denk, H., Zatloukal, K., and Müller, H. (2019). Causability and explainability of artificial intelligence in medicine. *WIREs Data Mining and Knowledge Discovery*, 9(4):e1312.

Hoogendoorn et al., 2016. Hoogendoorn, M., Szolovits, P., Moons, L. M., and Numans, M. E. (2016). Utilizing uncoded consultation notes from electronic medical records for predictive modeling of colorectal cancer. *Artificial Intelligence in Medicine*, 69:53–61.

Hubert and Arabie, 1985. Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

Huynh-Thu et al., 2010. Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS One*, 5(9):1–10.

Jong et al., 2004. Jong, K., Mary, J., Cornuéjols, A., Marchiori, E., and Sebag, M. (2004). Ensemble feature ranking. In *PKDD - LNCS 2302*, pages 267–278.

Kampichler et al., 2000. Kampichler, C., Džeroski, S., and Wieland, R. (2000). Application of machine learning techniques to the analysis of soil ecological data bases: relationships between habitat features and collembolan community characteristics. *Soil Biology and Biochemistry*, 32(2):197 – 209.

Karalič and Bratko, 1997. Karalič, A. and Bratko, I. (1997). First Order Regression. *Machine Learning*, 26(2-3):147–176.

Katakis et al., 2008. Katakis, I., Tsoumakas, G., and Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge*.

Kira and Rendell, 1992. Kira, K. and Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 129–134. AAAI Press.

Klimt and Yang, 2004. Klimt, B. and Yang, Y. (2004). The enron corpus: A new dataset for email classification research. In *ECML '04: Proceedings of the 18th European Conference on Machine Learning – LNCS 3201*, pages 217–226. Springer Berlin / Heidelberg.

Kocev et al., 2013. Kocev, D., Vens, C., Struyf, J., and Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3):817–833.

Kononenko and Robnik-ikonja, 2003. Kononenko, I. and Robnik-ikonja, M. (2003). Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning Journal*, 55:23–69.

Kralj Novak et al., 2015. Kralj Novak, P., Smailović, J., Sluban, B., and Mozetič, I. (2015). Sentiment of emojis. *PloS one*, 10.

Levatić, 2017. Levatić, J. (2017). *Semi-supervised Learning for Structured Output Prediction*. PhD thesis, Jožef Stefan Postgraduate School, Ljubljana, Slovenia.

Levatić et al., 2013. Levatić, J., Cúrak, J., Kralj, M., Šmuc, T., Osmak, M., and Supek, F. (2013). Accurate models for p-gp drug recognition induced from a cancer cell line cytotoxicity screen. *Journal of medicinal chemistry*, 56(14):5691–5708.

Levatić et al., 2018. Levatić, J., Kocev, D., Ceci, M., and Džeroski, S. (2018). Semi-supervised trees for multi-target regression. *Information Sciencies*, 450(C):109127.

Lichman, 2013. Lichman, M. (2013). UCI machine learning repository. `http://archive.ics.uci.edu/ml`.

Moro et al., 2011. Moro, S., Cortez, P., and Laureano, R. (2011). Using data mining for bank direct marketing: An application of the crisp-dm methodology.

Nilsson et al., 2007. Nilsson, R., Peña, J. M., Björkegren, J., and Tegnér, J. (2007). Consistent feature selection for pattern recognition in polynomial time. *Journal of Machine Learning Research*, 8:589–612.

Pedregosa et al., 2011. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pestian et al., 2007. Pestian, J. P., Brew, C., Matykiewicz, P., Hovermale, D. J., Johnson, N., Bretonnel Cohen, K., and Duch, W. (2007). A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing (BioNLP '07)*, pages 97–104.

Petković et al., 2019. Petković, M., Džeroski, S., and Kocev, D. (2019). Ensemble-based feature ranking for semi-supervised classification. In Kralj Novak, P., Šmuc, T., and Džeroski, S., editors, *Discovery Science*, pages 290–305. Springer International Publishing.

Petković et al., 2020. Petković, M., Džeroski, S., and Kocev, D. (2020). Feature ranking for hierarchical multi-label classification with tree ensemble methods. *Acta Polytechnica Hungarica*. To appear.

Petković et al., 2019. Petković, M., Kocev, D., and Džeroski, S. (2019). Feature ranking for multi-target regression. *Machine Learning Journal*. Accepted.

Petković et al., 2018. Petković, M., Kocev, D., and Džeroski, S. (2018). Feature ranking with relief for multi-label classification: Does distance matter? In Soldatova, L., Vanschoren, J., Papadopoulos, G., and Ceci, M., editors, *Discovery Science*, pages 51–65. Springer International Publishing.

Reyes et al., 2015. Reyes, O., Morell, C., and Ventura, S. (2015). Scalable extensions of the relieff algorithm for weighting and selecting features on the multi-label learning context. *Neurocomputing*, 161:168 – 182.

Saeys et al., 2007. Saeys, Y., Inza, I., and Larraaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517.

Sheikhpour et al., 2017. Sheikhpour, R., Sarram, M., Gharaghani, S., and Chahooki, M. (2017). A survey on semi-supervised feature selection methods. *Pattern Recognition*, 64(C):141–158.

Spyromitros-Xioufis et al., 2016. Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., and Vlahavas, I. (2016). Multi-target regression via input space expansion: Treating targets as inputs. *Machine Learning*, 104(1):5598.

Stańczyk and Jain, 2015. Stańczyk, U. and Jain, L. C., editors (2015). *Feature Selection for Data and Pattern Recognition*. Studies in Computational Intelligence. Springer Berlin Heidelberg.

Stojanova, 2009. Stojanova, D. (2009). Estimating forest properties from remotely sensed data by using machine learning, MSc Thesis. Jožef Stefan International Postgraduate School.

Tjoa and Guan, 2019. Tjoa, E. and Guan, C. (2019). A survey on explainable artificial intelligence (xai): Towards medical xai.

Trochidis et al., 2008. Trochidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. (2008). Multilabel classification of music into emotions. In *2008 International Conference on Music Information Retrieval (ISMIR 2008)*, pages 325–330.

Tsagris et al., 2018. Tsagris, M., Lagani, V., and Tsamardinos, I. (2018). Feature selection for high-dimensional temporal data. *BMC Bioinformatics*, 19(1):17.

Van Der Putten and Van Someren, 2004. Van Der Putten, P. and Van Someren, M. (2004). A bias-variance analysis of a real world learning problem: The coil challenge 2000. *Machine Learning*, 57(1-2):177–195.

Xu et al., 1992. Xu, L., Krzyzak, A., and Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on systems, man, and cybernetics*, 22(3):418–435.

Zhou et al., 2018. Zhou, Y., Zhang, R., Wang, S., and Wang, F. (2018). Feature selection method based on high-resolution remote sensing images and the effect of sensitive features on classification accuracy. *Sensors*, 18(7).

Zhou, 2017. Zhou, Z.-H. (2017). A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53.

Zhu et al., 2009. Zhu, X., Goldberg, A. B., Brachman, R., and Dietterich, T. (2009). *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers.

Chapter 7

Feature Ranking for Unsupervised Learning

In this Chapter, we present in detail the fourth set of contributions of the thesis: those concerning the unsupervised learning setting. Recall from the introductory sections that in unsupervised learning, no target variable/attribute is present in the data (Section 2.2). Also recall that we presented our unsupervised feature ranking methods in Sections 3.1.2 and 3.2.2. Our contributions in this context include:

1. An extension of ensemble-based feature ranking scores from the context of predictive modeling to unsupervised learning.
2. An extension of distance-based feature ranking scores from the context of predictive modeling to unsupervised learning.
3. An extensive experimental evaluation of the newly proposed unsupervised feature importance scores on a collection of benchmark datasets assessing the performance of the scores individually and in cross-comparison, as well as in comparison to other existing unsupervised feature ranking methods.

We proposed the extensions of ensemble-based and distance-based feature ranking methods to the unsupervised learning setting in a journal paper under review (Petković, Škrlj, et al., 2020). As outlined in Sections 3.1.2 and 3.2.2, the proposed feature ranking methods follow the intuition of the predictive clustering paradigm. The group of ensemble-based methods consist of pairs of ensemble generation methods (bagging, random forests and extra trees) and scores (Genie3, and Random Forest scores). Since the Symbolic score can be seen as a more rudimentary version of the Genie3 score, we did not experiment with it this time. The group of distance-based methods consists of URelief – the extension of the Relief algorithm to the unsupervised learning setting.

We evaluated the performance of the methods on a range of benchmark datasets. Ensembles of extra trees that were chosen as the main underlying ensemble method for their time-efficiency also outperform the other two methods in terms of quality of the corresponding feature rankings. The better of the two ensemble-based feature ranking scores was Genie3.

Both ensemble-based and distance-based feature ranking scores outperform the current state-of-the-art unsupervised feature rankings. To be more precise, the Genie3 score outperforms them (and URelief) statistically significantly and is most frequently the best performing method. URelief has the second best average rank (after the Genie3 score).

The work presented in this Chapter refers to the following hypotheses (as defined in the introduction):

H1: It is possible to extend ensemble- and distance-based feature ranking approaches to the unsupervised feature ranking task, to the tasks of supervised SOP (i.e., MTR, MLC, and HMLC), and to their semi-supervised versions.

H6: For unsupervised, MTR, MLC and HMLC problems, ensemble-based approaches on average outperform the distance-based approaches when the number of features is extremely high.

H9: The proposed ensemble-based and distance-based approaches yield relevant and state-of-the-art feature rankings for unsupervised feature-ranking problems.

These three hypotheses are confirmed with the design and implementation of the unsupervised feature ranking approaches presented in this Chapter and the experimental study comparing their performance. Hypothesis H3 is completely confirmed by the results presented in this Chapter. For hypotheses H1 and H6, the parts pertaining to unsupervised scenario, are confirmed in this Chapter, while the remaining parts of these hypotheses are addressed in the previous chapters.

The paper included in this Chapter is:

- Petković, M., Kocev, D., Škrlj, B. and Džeroski, S. Ensemble- and Distance-Based Feature Ranking for Unsupervised Learning. *Information Fusion.* Under Review.

The contribution of Matej Petković to this paper are as follows. MP contributed to the design of the ensemble-based and distance-based feature ranking methods for unsupervised learning and implemented these methods in computer code. He also participated in designing the experiments, carried out the experiments, and processed their results. He drafted the paper and revised it following the feedback from the co-authors and reviewers.

Ensemble- and Distance-Based Feature Ranking for Unsupervised Learning

Matej Petković*, Dragi Kocev, Blaž Škrlj, Sašo Džeroski

Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Jozef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia

Abstract

In this work, we propose two novel (groups of) methods for unsupervised feature ranking and selection. The first group includes feature ranking scores (Genie3 score, RandomForest score) that are computed from ensembles of predictive clustering trees. The second method is URelief, the unsupervised extension of the Relief family of algorithms. Using 26 benchmark data sets and 5 baselines, we show that both the Genie3 score (computed from the ensemble of extra trees) and the URelief method outperform the existing methods and that Genie3 performs best overall. Additionally, we analyze the influence of the hyper-parameters of the proposed methods on their performance, and show that for the Genie3 score the highest quality is achieved by the most efficient parameter configuration. Finally, we propose a way of discovering the location of the most relevant features in the ranking.

Keywords: feature ranking, unsupervised learning, explainability, tree ensembles, extra trees, Relief

1. Introduction

High-dimensional data are becoming part of everyday life, making the learning of models much harder, slower or even infeasible, unless some dimensionality reduction is performed. Moreover, explainable AI (or XAI) plays a central role in the human-centric approach to trustworthy AI [1, 2], especially in domains such as medicine [3, 4], life sciences [5, 6, 7, 8] and ecological modeling [9, 10, 11].

Both problems (high dimensionality and lack of explainability) may be solved by *feature ranking* approaches. In the supervised scenario, where the examples in the data are described as $(\boldsymbol{x}, y)$ pair, with $\boldsymbol{x}$ a tuple of values of descriptive attributes (features) x_i and y a target value, the goal of feature ranking is

*Corresponding author: matej.petkovic@ijs.si, +38614773635

Email address: `{matej.petkovic, dragi.kocev, blaz.skrlj, saso.dzeroski}@ijs.si` (Matej Petković*, Dragi Kocev, Blaž Škrlj, Sašo Džeroski) to assign a real-valued score to each feature, which defines how relevant the feature is for predicting the target y. By assessing the relevance correctly, one can i) discard the irrelevant features and make learning predictive models faster/feasible, or ii) understand the data better, which helps when a machine learning expert collaborates with a domain expert.

However, sometimes the examples in the data are hard to label since labeling either demands a lot of manual work (e.g., labeling tweets [12]), or is expensive for some other reason (e.g., determining the activity of chemical compounds [13]). Therefore, semi-supervised learning that uses both labeled and unlabeled data is gaining importance. Methods for feature ranking in semi-supervised learning have been recently developed [14, 15] based on SSL with PCTs as well as existing distance-based approaches for feature ranking such as Relief.

On the spectrum of major machine learning paradigms/ settings, semi-supervised learning is situated between supervised learning, which uses only labeled examples (with known values of the targets) to learn predictive models, and unsupervised learning, where only examples without labels are used, typically to group examples into clusters of similar examples. Similarity between examples is determined with respect to a given distance measure, typically Euclidean. Just as we need feature ranking in the context of supervised and semi-supervised learning, we also need it in unsupervised learning.

Since we do not have targets to predict in unsupervised learning, feature importance in this setting cannot mean how relevant the features are for predicting the targets (and building a predictive model for that purpose). Rather, feature importance in this context measures how relevant a given feature is for constructing a clustering from the given data. Other formulations and motivations of relevance in ranking in an unsupervised context have been considered, such as the one recently proposed by Doquet and Sebag [16]: We do not know how relevant is a given feature (for predicting a target), but we can assess how redundant it is in the context of the other features (a feature is redundant if we can predict it accurately from other features). However, we believe the relevance to constructing a clustering captures more appropriately the spirit of unsupervised feature ranking and it is this notion that is embedded within the approaches for unsupervised feature ranking that we put forward in this paper. The intuition of Doquet and Sebag [16] corresponds more to feature selection, the task of finding an optimal set of features, where redundancy is to be avoided. Feature selection can be viewed as a special case of feature ranking/ relevance estimation, where every feature is assigned a score of either 1 (the feature is kept) or 0 (the feature is discarded). We can obtain a feature selection by first obtaining a ranking (in terms of relevance scores) and then keeping only the features with the scores exceeding some user-defined threshold [17].

In this work, we propose two groups of novel methods for unsupervised feature ranking[1]. The first group consists of two ensemble-based feature ranking

[1]The implementation of the methods, as well as the results, are available at `https://gitlab.com/Petkomat/unsupervised-feature-ranking`

scores that are computed from ensembles of predictive clustering trees used for hierarchical clustering [18]. At each internal node of a tree, the algorithm finds a feature along which the data are most efficiently divided into subclusters. In the leaves, prototypes of the clusters are computed. These prototypes can be seen as a prediction for feature values for a new-coming example, thus the features that are (more often) used in the splits, are (more) relevant for predicting the values of all the features.

The second group consists of only one member named URelief, i.e., the unsupervised extension of the Relief family of algorithms [19, 20]. It does not use any underlying model, but rather estimates the feature relevances via the influence of the features to the distances between the neighboring examples in the data. In other words, if the distance along a given feature strongly contributes to the overall distance between examples, the feature is deemed important. In distance-based clustering, the feature would be important for the clustering of the examples.

The rest of the paper is organized as follows. Sec. 2 gives a brief description of related work, in particular the baseline methods that we compare to. In Sec. 3, a detailed description of the proposed ensemble-based scores is given, whereas in Sec. 4 this is done for URelief. Sec. 5 describes the experimental setup, including the evaluation procedure. Sec. 6 discusses the experimental results, and Sec. 7 concludes the paper.

2. Related Work

While the literature on supervised feature selection has been extensively reviewed and studied [21, 22], the literature on the topic of unsupervised feature ranking has not been extensively and systematically surveyed. The closest overview is the recent work by Solorio-Fernández et al. [23] that overviews methods for unsupervised feature selection. It divides the methods based on the strategy they use for selecting the features, into three groups: filter, wrapper and hybrid. The filter methods evaluate the features based on intrinsic properties of the data, the wrapper methods evaluate the features based on the results of a specific clustering algorithm, and the hybrid methods exploit the advantages of both filter and wrapper methods with a special focus on the balance between the computational cost and the quality of the selected features. Here, we briefly summarize the most prominent and relevant work on the topic of *unsupervised feature ranking*.

To begin with, the Scikit-feature repository [24] offers unified implementations of some unsupervised feature ranking algorithms[2]. One of them is the Laplace score [25]. It is based on the graph theory and the creation of the Laplace matrix of a graph, whose vertices are the examples and whose edges correspond to the level of similarity among the examples.

[2]`https://github.com/jundongl/scikit-feature`

Next, SPEC (spectral clustering) [26] is a unification of many existing feature ranking scores. These include the Relief and the Laplace scores. The actual score that the authors experiment with is thus a result of particular instantiations, such as choosing an appropriate similarity measure, e.g., radial basis kernels.

Furthermore, the similarity matrix (calculated by using a radial-basis kernel) is also one of the building blocks of the MCFS algorithm [27]. However, in addition to calculating this matrix, the algorithm proceeds to solving a generalized eigenvalue problem. It ensures the sparsity of the scores by including a L1-regularization term in the objective function.

Next, another variation of the above approaches is NDFS [28]. Here, L2,1-regularization is applied rather than L1-regularization. The best features are those which are the most closely related to the labels of the clusters constructed within the algorithm.

The last method that we discuss here is AgnoS-S. This is the best performing variation among the unsupervised ranking methods proposed by Doquet and Sebag [16] in their award-wining paper at the ECML 2019 conference. This is an auto-encoder-based feature ranking method, where slack variables are included into the auto-encoder (and the loss function) to ensure that more important features get higher weights. The method also estimates the intrinsic dimension of the data [29], determining the width of the middle layer of the auto-encoder, which the method optimizes.

We use the above approaches as baselines when evaluating the quality of the methods we introduce in this paper.

3. Ensemble-Based Feature Ranking

The first two proposed feature ranking scores are embedded in ensembles of predictive-clustering trees (PCTs). In this section, we first describe the PCTs, proceed to ensembles thereof, and conclude with feature ranking scores.

3.1. Single PCTs

PCTs are a fully modular generalization of standard decision trees [30] where three groups of attributes need to be defined.

The attributes that can be used in the tests (splits) in the internal nodes of the trees (line 2 in Alg. 2) are *descriptive* attributes (or features). The attributes that are used to assess the quality of split candidates, i.e., are part of the formula of impurity function *impu*, are *clustering* attributes. The attributes that are predicted in the leaves, i.e., returned by the *prototype* function, are *target* attributes.

When PCTs are used in predictive modeling, clustering and target attributes mostly coincide (semi-supervised learning is one of the exceptions [31]), and there is typically only one such attribute (multi-target tasks are an exception [32]). For example, in classification, we are predicting the value of a single nominal attribute y, which is also used in impurity calculations. The remaining attributes are descriptive.

However, we use PCTs for (hierarchical) clustering. Here, the groups of descriptive, clustering, and target attributes coincide, since any of the attributes can be used in a test, all of them are taken into account when computing the impurity of clusters, and all of them are "predicted" when the prototypes of the clusters are computed in the leaves.

The exact pseudocode for inducing a PCT is given in Alg. 1, which takes as its input a subset of examples $E \subseteq \mathscr{D}_{\text{TRAIN}}$, and returns a PCT.

Algorithm 1 PCT(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$
2: if $t^* = none$ then
3:     return $Leaf(prototype(E))$
4: else
5:     for each $E_i \in \mathcal{P}^*$ do
6:         $tree_i = \text{PCT}(E_i)$
7:     return $Node(t^*, \bigcup_i \{tree_i\})$
```

Algorithm 2 BestTest(E)

```
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: for each test $t$ do
3:     $\mathcal{P}$ = partition induced by $t$ on $E$
4:     $h = |E| impu(E) - \sum_{E_i \in \mathcal{P}} |E_i| impu(E_i)$
5:     if $h > h^*$ then
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $(t^*, h^*, \mathcal{P}^*)$
```

In clustering (unsupervised learning), the impurity function for a given subset $E \subseteq \mathscr{D}_{\text{TRAIN}}$ is defined as the average of $impu(E) = \frac{1}{n}\sum_{i=1}^{n} impu(E, x_i)$ of the clustering-attribute impurities $impu(E, x_i)$, which are defined as follows.

For nominal variables x, the impurity is defined in terms of the Gini Index $Gini(E, x) = 1 - \sum_v p_E^2(v)$, where the sum goes over the possible values v of the variable x, and $p_E(x)$ is the relative frequency of the value v in the subset E. To avoid any bias towards variables with high overall impurity, the impurity is defined as the normalized Gini value, i.e., $impu(E, x) = Gini(E, x)/Gini(\mathscr{D}_{\text{TRAIN}}, x)$.

For numeric variables x, the impurity is similarly defined as the normalized variance of x, i.e., $impu(E, x) = Var(E, x)/Var(\mathscr{D}_{\text{TRAIN}}, x)$.

The prototype function returns a vector $[\hat{x}_1, \dots, \hat{x}_n]$, where $\hat{x}_i$ is a mean of the attribute x_i if the attribute is numeric, and the mode of the attribute, if the attribute is nominal. The means and modes are computed from the training examples in a given leaf.

3.2. Ensembles of PCTs

Single trees are known for their instability, i.e., undesirably high variance [33]. To overcome that, ensemble methods such as bagging [34], random forests [33] and ensembles of extremely randomized trees (extra trees) [35] have been proposed. All of these methods have been adapted to learn ensembles of PCTs [32, 36].

A tree ensemble is a set of trees which are grown on a possibly manipulated training set and/or a randomized tree induction algorithm. More precisely, in **bagging**, every tree in an ensemble is grown on an independent bootstrap replicate of the training set $\mathscr{D}_{\text{TRAIN}}$. **Random forests** additionally consider only a random subset of descriptive attributes, when finding the best test (line 2 of Alg. 2). The subset is drawn independently in every internal node.

Extra trees, as the name suggests, go even further regarding randomization and, as the random forests, first choose a subset of descriptive attributes, and then evaluate only one test per attribute. For example, if x_i is one of the chosen numeric features, random forests evaluate the tests $x_i \leq \vartheta$, for all possible values of ϑ, whereas extra trees additionally randomly choose only one ϑ_i, for each chosen feature x_i.

Originally, extra trees do not use bootstrapping. However, we have included it in learning extra PCTs since i) our preliminary experiments show that it is beneficial to use it (especially when there are many binary features in the data), and ii) one of the feature scores described in the next section requires bootstrapping.

3.3. Feature Ranking Scores

We propose to use two feature ranking scores that can be computed from any of the ensembles described in the previous section.

The first one is GENIE3 [37], originally defined for predictive modeling tasks, such as classification or regression. Within the framework of predictive clustering, we can extend the score to unsupervised modeling as

$$\mathit{importance}_{\text{GENIE3}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} h^*(\mathcal{N}), \tag{1}$$

where $\mathcal{E}$ is an ensemble of trees $\mathcal{T}$, and $\mathcal{T}(x_i)$ is the set of internal nodes of the tree $\mathcal{T}$ where x_i is part of the test. Finally, $h^*(\mathcal{N})$ is the quality of the test in the node $\mathcal{N}$, as assessed by the heuristic in Alg. 2, line 4.

Note that $h^*(\mathcal{N})$ is proportional to the number of examples that reach the node $\mathcal{N}$ (since those nodes have bigger influence), and the impurity reduction value. One could thus also use a simpler version of the GENIE3 score (namely Symbolic score [38]) that ignores the impurity reduction factor, and weights every appearance of the feature x_i by the number of examples that reach that node.

The second feature ranking score that we propose is a generalization of the RANDOMFOREST score [33]. Instead of evaluating a feature directly via the quality of the tests, this approach compares how much introducing some noise in the feature degrades the performance of a model. More precisely, for every tree $\mathcal{T}$ in the ensemble, we first define the set of out-of-bag examples $\text{OOB}_{\mathcal{T}}$, i.e., the examples that were not chosen into the bootstrap replicate of the training set. Next, we define the set $\text{OOB}_{\mathcal{T}}{}^i$ that is obtained from the set $\text{OOB}_{\mathcal{T}}$ by permuting the values of the feature x_i. Let e be some error measure (e.g., the average of the per-feature variances in a cluster). Then, the RANDOMFOREST (RF) score is defined as

$$\mathit{importance}_{\text{RF}}(x_i) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{T} \in \mathcal{E}} \frac{e(\text{OOB}^i_{\mathcal{T}}) - e(\text{OOB}_{\mathcal{T}})}{e(\text{OOB}_{\mathcal{T}})}, \tag{2}$$

i.e., as the relative increase of the error measure e. The larger the increase, the more important the feature. The score has been originally defined for random forests (hence the name), but can be used with practically any ensemble approach based on bagging.

3.4. Time complexity

Since the sets of descriptive, clustering and target attributes coincide with the number of all attributes, we denote their sizes by n. The other dimension of the data set, i.e., the number of examples, is denoted by m.

In the analysis, we assume that the trees are approximately balanced, i.e., their depth is $\mathcal{O}(\log m)$. The largest amount of computation is performed when estimating the quality of the tests. In a given node, evaluating split candidates that a single attribute yields, requires $\mathcal{O}(m' \log m' + m'n)$ steps (sorting the m' examples that reach the node according to their attribute values) and efficiently evaluating either one (extra trees) or all thresholded tests (bagging, random forests).

Taking into account that there are n' chosen features in each node (for bagging, $n' = n$), and the assumed depth of the tree, we can derive the total time complexity of growing a tree: $\mathcal{O}((m \log m + mn)n' \log m) = \mathcal{O}((\log m + n)n'm \log m)$[3]. One can loose the inner $\log m$ term by presorting the data with respect to all the features in advance.

Since the trees are grown independently, the algorithm is easy to parallelize, and the time complexity of growing any of the ensembles is the same.

After that, the cost of computing the GENIE3 ranking is negligible, since the ensemble stores the quality of the nodes and we can traverse the nodes of a single tree in time $\mathcal{O}(m)$. Computing the RANDOMFOREST ranking is a bit more time-consuming. We have to permute the values of a feature, and send the examples trough each tree, for every feature. This is done in $\mathcal{O}(nm \log m)$ time. This is still faster than inducing a tree.

4. URELIEF-based Feature Ranking

In contrast to ensemble-based feature ranking scores, the RELIEF family of feature ranking algorithms does not use any predictive model. Its members can handle various predictive modeling tasks, including classification [19], regression [20], and others [38, 40, 41].

In this work, we extend the RELIEF family to unsupervised ranking, by again taking the predictive-clustering point of view. We follow the main intuition behind RELIEF [20], which for two examples that are close to each other (in the clustering space) states: The feature x_i is relevant if the differences in the

[3]Thus, bagging is quadratic in the number of features which is a problem if trees are not grown in parallel. This is the reason why we did not introduce feature importance scores using gradient-boosted trees [39].

Algorithm 3 URelief($\mathscr{D}_{\text{TRAIN}}$, I, K)

```
1:  $\boldsymbol{w}$, $\boldsymbol{P}_{\text{diffAttr, diffTarget}}$, $\boldsymbol{P}_{\text{diffAttr}}$ = zero lists of length $n$
2:  $P_{\text{diffTarget}} = 0.0$
3:  for iteration $= 1, 2, \dots, I$ do
4:      $\boldsymbol{r}$ = random example from $\mathscr{D}_{\text{TRAIN}}$
5:      $\boldsymbol{n}_1, \boldsymbol{n}_2, \dots, \boldsymbol{n}_K = K$ nearest neighbors of $\boldsymbol{r}$ with respect to $d_{\mathcal{X}}$
6:      for $k = 1, 2, \dots, K$ do
7:          $P_{\text{diffClus}}$ += $d_{\mathcal{X}}\big(\boldsymbol{r}, \boldsymbol{n}_k\big)/(IK)$
8:          for $i = 1, 2, \dots, n$ do
9:              $\boldsymbol{P}_{\text{diffAttr}}[i]$ += $d_i\big(\boldsymbol{r}, \boldsymbol{n}_k\big)/(FIK)$
10:             $\boldsymbol{P}_{\text{diffAttr, diffClus}}[i]$ += $d_i\big(\boldsymbol{r}, \boldsymbol{n}_k\big) d_{\mathcal{X}}\big(\boldsymbol{r}, \boldsymbol{n}_k\big)/(FIK)$
11: for $i = 1, 2, \dots, n$ do
12:     $w_i = \frac{\boldsymbol{P}_{\text{diffAttr, diffClus}}[i]}{P_{\text{diffClus}}} - \frac{\boldsymbol{P}_{\text{diffAttr}}[i] - \boldsymbol{P}_{\text{diffAttr, diffClus}}[i]}{1 - P_{\text{diffClus}}}$
13: return $\boldsymbol{w}$
```

target space between the two examples are notable if and only if the differences in the feature values of x_i between these two examples are notable.

As in PCT induction above, we again define clustering and target space to consist of all n attributes. We estimate the relevance of the descriptive attributes, which again comprise all n attributes.

Being close or having a notable difference is operationalized in the algorithm via the distances in the appropriate spaces. For the clustering (target) space $\mathcal{X}$ spanned by the domains $\mathcal{X}_i$ of the features x_i, we have

$$d_{\mathcal{X}}(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{1}{n} \sum_{i=1}^{n} d_i(\boldsymbol{x}^1, \boldsymbol{x}^2); \quad d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) = \begin{cases} \mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2] & : \mathcal{X}_i \nsubseteq \mathbb{R} \\ \frac{|\boldsymbol{x}_i^1 - \boldsymbol{x}_i^2|}{\max\limits_{\boldsymbol{x}} \boldsymbol{x}_i - \min\limits_{\boldsymbol{x}} \boldsymbol{x}_i} & : \mathcal{X}_i \subseteq \mathbb{R} \end{cases} \tag{3}$$

where $\mathbf{1}$ denotes the indicator function. These distances are then used in the estimation of the expression

$$P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \mid \boldsymbol{x}^1 \neq \boldsymbol{x}^2) - P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \mid \boldsymbol{x}^1 = \boldsymbol{x}^2), \tag{4}$$

which is the output of the URelief algorithm. Above, $\boldsymbol{x}^{1,2} \in \mathcal{X}$ are two examples, and $\boldsymbol{x}_i^{1,2}$ values of the feature x_i for these two examples.

The unsupervised version of the algorithm, URelief is shown in Alg. 3. Within the code, the probability $P_{\text{diffAttr},i} = P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2)$ is modeled as the distance d_i (line 9), whereas the probability $P_{\text{diffClus}} = P(\boldsymbol{x}^1 \neq \boldsymbol{x}^2)$ is modeled as the distance $d_{\mathcal{X}}$ (line 7). The conditional probabilities are estimated via the Bayes formula (line 12). The product of two events, e.g., $P(\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2 \wedge \boldsymbol{x}^1 \neq \boldsymbol{x}^2)$ is modeled as the product of the two probabilities: This is used together with the Bayes formula in line 12.

The algorithm is iterative. On every iteration, an example $\boldsymbol{r}$ is chosen from the training set (line 4), and its nearest neighbors $\boldsymbol{n}_k$ are computed in the clustering space. After that, the estimates for the needed probabilities are updated according to the distances between $\boldsymbol{r}$ and $\boldsymbol{n}_k$. The updates are normalized, so that the final estimate of every probability falls into the interval $[0, 1]$. Finally, in line 12, the weights (feature importance scores) are computed.

The **time complexity** of URelief can be estimated as follows. To make full use of the efficient vectorized implementation of scipy's [42] `cdist` for computing the distances among the examples, we first compute all nearest neighbors. This takes $\mathcal{O}(Imn)$ steps. Since the number of iterations is typically set to be a proportion of m, the number of steps is $\mathcal{O}(m^2 n)$. When updating the probability estimates, only the K nearest neighbors are considered, so the total number of steps is also $\mathcal{O}(m^2 n)$, since K is upper-bounded by some constant.

Thus, regarding the time complexity, URelief is more appropriate for high-dimensional data than ensemble-based feature rankings, since it is linear in the number of features. However, the results show that this comes at a performance price.

5. Experimental Setup

In this section, we describe the experimental setup of the empirical evaluation of the proposed methods. We state the experimental questions and competing methods, describe the evaluation procedure, give a brief description of the data sets, and finish with the parameter instantiation of the different ranking methods.

The evaluation is based on the following experimental questions:

1. Which of the proposed ensemble scores is better?
2. Are the proposed feature ranking scores state-of-the-art (SOTA)?
3. What is the influence of the parameters of the proposed methods on the quality of the produced rankings?
4. How to locate the relevant features in a ranking?

To answer the first question, we compare the performance of the two ensemble-based ranking scores with the Wilcoxon (statistical) test [43]. The evaluation procedure is discussed in Section 5.2.

To answer the second question, URelief, the better of the two ensemble-based scores, and five current SOTA methods are compared via Friedman's test, together with Nemenyi's post-hoc test [43]. The considered competitors are AgnoS-S, the Laplace score, NDFS, MCFS and SPEC, all briefly described in Sec. 2.

To answer the third question, we compare the performances of the proposed scores obtained with different parameter configurations. We vary the ensemble method and the subset size for ensemble-based scores. We vary the numbers of neighbors and iterations for URelief.

To answer the fourth question, we construct a curve that measures the performance of the sets (of different sizes) of top-ranked features. A detailed description of the approach taken is given in Sec. 6.4.

5.1. Datasets

We first planned to use all the 29 available datasets from the Scikit-feature repository. To assure that the considered datasets are independent, as the statistical tests assume, we had to exclude three of them: `lung_small` (but `lung` is present), `orlraws10P` (but `ORL` is present) and `warpAR10P` (but `warpPIE10P` is present). Those included - together with the number of features and examples, are listed in Tab. 1 and available for download from the repository[4]. Note that they typically have a low number of examples and a high number of features.

Table 1: The basic characteristics of the considered data sets.

data set	examples	features	domain
ALLAML	72	7129	biology
arcene	200	10000	mass spectrometry
BASEHOCK	1993	4862	text data
Carcinom	174	9182	biology
CLL-SUB-111	111	11340	biology
COIL20	1440	1024	face image
colon	62	2000	biology
gisette	7000	5000	digit recognition
GLI-85	85	22283	biology
GLIOMA	50	4434	biology
Isolet	1560	617	spoken letter recognition
leukemia	72	7070	biology
lung	203	3312	biology
lymphoma	96	4026	biology
madelon	2600	500	artificial
nci9	60	9712	biology
ORL	400	1024	face image
PCMAC	1943	3289	text data
pixraw10P	100	10000	face image
Prostate-GE	102	5966	biology
RELATHE	1427	4322	text data
SMK-CAN-187	187	19993	biology
TOX-171	171	5748	biology
USPS	9298	256	hand written image
warpPIE10P	210	2420	face image
Yale	165	1024	face image

5.2. Evaluation Procedure

In the supervised learning setting, one of the popular procedures for evaluating feature ranking algorithms is the use of the weighted k-nearest neigh-

[4]`https://github.com/jundongl/scikit-feature`

bors (kNN) algorithm [44] in which the distance among the examples employed weights features by their importance and is defined as

$$d(\boldsymbol{x}^{\mathbf{1}}, \boldsymbol{x}^{\mathbf{2}}) = \left(\sum_{i=1}^{n} \mathit{importance}(x_i)(\boldsymbol{x}_i^{\mathbf{1}} - \boldsymbol{x}_i^{\mathbf{2}})^p\right)^{1/p}$$

for some $p > 0$. For example, $p = 2$ results in the weighted Euclidean distance, and the better the performance, the better are the feature-importance estimates [45]. In the unsupervised scenario, choosing the correct evaluation procedure is trickier.

For example, the above procedure cannot be extended to the unsupervised scenario, since we would predict the same attributes that are used for the distance computation. As a consequence, if one, for example, measures the performance of the model in terms of mean squared error, it can be proven that the performance of the nearest-neighbors model is optimized when the weights $\mathit{importance}(x_i)$ are all equal.

Another popular (yet sometimes not appropriate procedure) for evaluating unsupervised feature rankings is to i) use a supervised data set, ii) compute a ranking ignoring the target variable, iii) take some top-ranked features and learn a model that predicts the target variable. This approach only makes sense when the so-called clustering hypothesis holds [46], i.e., when *clusters of data examples (as computed in the descriptive space) well resemble the distribution of target values*. Since the considered datasets also contain the target variable, they allow us to check this hypothesis by computing the Adjusted Random Index (ARI) [47] between the actual class labels and the labels obtained from the k-means clustering, where k was set to the number of classes in a given data set.

The ARI can take the values from the interval $[-1, 1]$, and is adjusted so that ARI $= 0$ if the two labelings are independent, i.e., the clusters do not tell us anything about the target. The distribution of the ARI scores over the considered data sets is given in Fig. 1 and shows that there are many data sets with ARI close to 0. Therefore, the supervised approach to evaluating unsupervised feature rankings is also inappropriate.

Therefore, we adopt another widely-used approach [16] where some fixed number of features is chosen, and a predictive model is built using only this many top-ranked features. In our experiments, we choose 16 features[5]. When choosing a model, we again follow Doquet and Sebag [16], and use a nearest-neighbors classifier. However, since some of the data sets contain a rather low number of examples (e.g., 50), we set the number of neighbors to 1.

Note that the evaluation procedure of Doquet and Sebag [16] is not entirely correct, since it uses the same data for learning the ranking and evaluating it,

[5]To see where in the ranking are actually located the most relevant features, we built models that use top 1, 2, 4, 8, 16, etc. features (see Sec. 6.4). Number 16 is the closest to the 20 features that Doquet and Sebag [16] use.

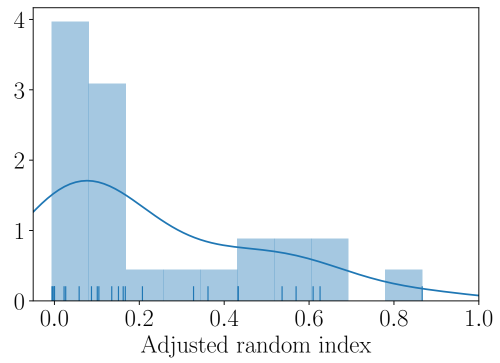


Figure 1: The distribution of the ARI values (as estimated by kernel-density estimation in Python's module seaborn (Available at `https://seaborn.pydata.org/`) over the considered data sets. The ARI value for a given data set (represented as a single tick) is the median of the ARI values over ten runs of the k-means algorithm.

and therefore measures only how well the ranking *overfits* the data. We fix this by using 10-fold cross-validation: A feature ranking is learned from a training fold, and the 1NN classifier with data from the training fold is tested on the testing fold. As the evaluation measure, we use mean squared error: All the features in the data are numeric, since some competing methods cannot handle nominal features.

5.3. Parameter instantiation

Our previous experiments with the semi-supervised version of Relief [15] showed that the optimal number of neighbors is around 30, so we set $K = 30$ in URelief. Regarding the number of iterations I, higher values means more accurate estimates, so we use $I = m$, m being the number of examples in the data. Since the number of examples is not very high, this value of I is not prohibitively large. We experiment with the other parameter configurations in Sec. 6.3, considering different numbers of neighbors and iterations.

For calculating the ensemble-based rankings, we set the size of the subset of chosen features to $n' = \log_2 n$ (rounded up), as suggested by Breiman [33]. Moreover, this number is sufficiently low to make tree induction time-efficient. We use the extra tree ensembles (they evaluate only n' tests per internal node) and induce the trees in parallel, using the Slovenian national supercomputing network. The number of trees was set to 100 and they were fully grown. We also experiment with other parameter configurations from Section 6.3, considering different subset sizes and ensemble methods.

For the baselines implemented in the Scikit-feature repository, we use the default settings, following Doquet and Sebag [16]. If a method demands the number of clusters at input (MCFS, NDFS), we set it to the number of classes in the given data set. For AgnoS-S, we use the shape of the network and the training parameters as reported by Doquet and Sebag [16].

6. Results

Below we discuss the results of the evaluation in light of our experimental questions.

6.1. Which Ensemble Score is Better?

To answer this question, we directly compare the cross-validated MSE performance of the 1NN classifiers that use the top-ranked features, as determined by the Genie3 and RandomForest scores calculated from extra tree ensembles. Their performance is reported in Table 2, together with the corresponding average rank: for a given data set, the better score gets rank 1, and the other gets rank 2.

Results suggest that Genie3 feature rankings are, on average, better than RandomForest ones. The differences in performance between the two methods are also statistically significant (at the significance level of 0.05): The p-value from the Wilcoxon's test is 0.0397. This is why we will use the Genie3 feature importance score in the main comparison of the methods presented below.

6.2. Are the Proposed Scores SOTA?

To answer this question, we compare the performance of Genie3 rankings, URelief rankings, and the baseline rankings (or, rather, the performance of the corresponding 1NN classifiers), on different data sets. The corresponding MSE values are reported in Table 3.

Comparing the methods we propose (Genie3, URelief) to their competitors in terms of average ranks, cf. Table 2, the best-performing feature-ranking method is Genie3. URelief also performs well, as it has the second-best average rank. To investigate whether the differences in performance among the algorithms are statistically significant, we apply Friedman's statistical test (since we compare more than two methods). The null hypothesis is rejected at the significance level of 0.05, since the p-value equals $9.4 \cdot 10^{-8}$. Thus, we can proceed to the Nemenyi's post-hoc test that reveals where the differences occur. The results of these tests are presented with the average rank diagram in Figure 2.

In addition to the average ranks of the algorithms (the lower, the better), the critical distance (CD) is reported. If the average ranks of two algorithms are at least CD apart, the difference in the performance of the algorithms is considered statistically significant. The groups of algorithms, for which this does not hold, are connected by red lines.

The interpretation of the results is thus as follows. The top performing group of algorithms, among which there are no statistically significant differences, consists of Genie3 and URelief. Moreover, Genie3 is statistically significantly better than all the remaining methods (i.e., those outside the top group). The

Table 2: The performance (in terms of MSE of the corresponding 1NN classifiers using the 16 top-ranked features) of Genie3 and RandomForest scores computed from ensembles of extra trees. Additionally, the average ranks of both methods are reported.

Dataset	Genie3	Random Forest
ALLAML	1.15	**1.14**
arcene	40.92	**40.79**
BASEHOCK	**0.17**	0.18
Carcinom	**0.33**	0.34
CLL-SUB-111	**1726.00**	2449.00
COIL20	0.12	**0.10**
colon	1.43	**1.42**
gisette	**269.00**	275.00
GLI-85	1516.00	**1508.00**
GLIOMA	**0.23**	0.24
Isolet	0.37	**0.36**
leukemia	**1.84**	1.87
lung	0.29	0.29
lymphoma	**1.73**	1.87
madelon	**31.79**	33.61
nci9	**1.62**	1.66
ORL	**24.90**	34.20
PCMAC	**0.18**	0.19
pixraw10P	6.65	**6.19**
Prostate-GE	0.22	**0.21**
RELATHE	**0.22**	0.23
SMK-CAN-187	0.34	0.34
TOX-171	**206.00**	217.00
USPS	**0.33**	0.48
warpPIE10P	**16.90**	22.30
Yale	**46.70**	51.80
Average Rank	**1.31**	1.69

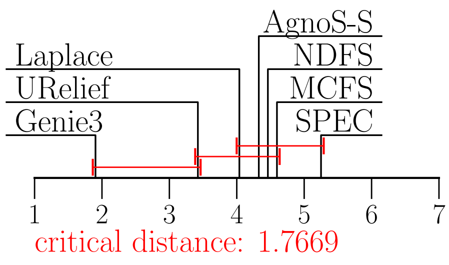


Figure 2: Average rank diagram.

Table 3: The performance of the 1NN classifiers that correspond to the Genie3, URelief and the baseline rankings, for every data set. The best performance per dataset is given in bold typeface. Additionally, the average rank for each of the feature ranking methods is given at the bottom.

Dataset	Genie3	URelief	Laplace	SPEC	MCFS	NDFS	AgnoS-S
ALLAML	1.15	**1.11**	1.16	1.15	1.16	1.15	1.22
arcene	**40.90**	51.30	42.40	76.20	42.40	42.40	80.60
BASEHOCK	0.17	0.18	0.18	0.18	0.18	0.18	**0.16**
Carcinom	**0.33**	**0.33**	0.36	0.36	0.35	0.36	0.36
CLL-SUB-111	**1726.00**	1823.00	1735.00	2350.00	2429.00	1744.00	2374.00
COIL20	0.12	0.11	0.16	0.23	0.28	**0.09**	0.18
colon	**1.43**	1.49	1.50	1.50	1.50	1.50	1.49
gisette	**269**	278.00	291.00	315.00	291.00	291.00	304.00
GLI-85	**1516.00**	1759.00	1521.00	1783.00	1558.00	1655.00	1712.00
GLIOMA	**0.23**	**0.23**	0.24	0.27	**0.23**	0.24	0.27
Isolet	0.37	0.48	0.37	0.45	0.42	0.38	**0.36**
leukemia	1.84	**1.83**	1.91	1.90	1.91	1.91	1.94
lung	**0.29**	**0.29**	0.30	0.33	**0.29**	0.31	0.31
lymphoma	**1.73**	1.81	2.01	2.01	2.01	2.01	1.85
madelon	31.79	31.83	33.62	33.62	33.62	33.62	**34.61**
nci9	1.62	1.64	**1.60**	1.66	**1.60**	**1.60**	1.76
ORL	**24.98**	27.60	34.19	34.19	34.19	34.19	23.19
PCMAC	0.18	0.19	**0.16**	0.18	0.18	0.19	**0.16**
pixraw10P	6.65	7.93	9.30	8.48	9.30	9.30	**6.48**
Prostate-GE	**0.22**	0.23	**0.22**	0.30	0.23	**0.22**	0.24
RELATHE	**0.22**	**0.22**	0.23	0.24	0.23	0.23	0.32
SMK-CAN-187	**0.34**	0.35	0.35	**0.34**	0.35	0.35	0.38
TOX-171	**206.00**	212.00	207.00	221.00	237.00	219.00	209.00
USPS	0.33	0.35	0.33	0.38	0.40	**0.29**	0.34
warpPIE10P	16.97	16.45	30.28	30.28	30.28	30.28	**15.30**
Yale	46.77	51.29	56.76	56.76	56.76	56.76	**42.73**
Avevare rank	**1.90**	3.42	4.04	5.25	4.60	4.46	4.33

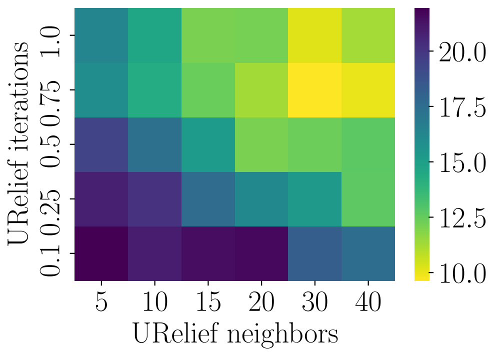


Figure 3: Average ranks (over the considered data sets) of the considered parameter configurations of URelief feature ranking. Lower ranks correspond to better performance.

other two groups are i) URelief, Laplace, AgnoS-S, NDFS, and MCFS, and ii) Laplace, AgnoS-S, NDFS, MCFS, and SPEC. Thus, the Genie3 *method is the new SOTA* in unsupervised feature ranking, whereas URelief is closer in performance to the previous SOTA methods.

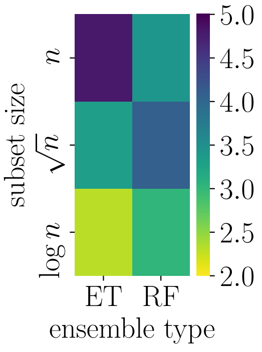
6.3. Parameter Setting Influence on Ranking Performance

In the main line of experiments, we set the parameters of our feature ranking methods to the most time-efficient values, where the time complexity critically depends on them. For ensemble-based scores, we chose extra tree ensembles and $\log_2 n$ as the number of considered features in internal nodes. As for URelief, we followed the previous findings since the (maximal) number of iterations is not a problem.

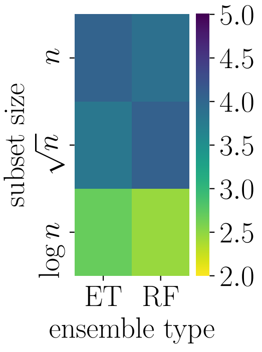
In this section, we investigate how the quality of the produced rankings depends on the parameters of the ranking methods. We start with URelief and then proceed to ensemble-based scores.

URelief. The heat-plot in Figure 3 shows the average ranks of the feature rankings computed by using all the pairs of different values for the number of iterations $I \in \{0.1m, 0.25m, 0.5m, 0.75m, 1.0m\}$ and the number of neighbors $K \in \{5, 10, 15, 20, 30, 40\}$. We can see that more neighbors and more iterations in general lead to better performance (lower ranks). Both observations are somewhat expected, since i) more iterations mean more stable estimates of the probabilities present in Eq. (4), and ii) a higher number of neighbors was also a suitable choice for our semi-supervised experiments [15]. The explanation for why the algorithm prefers a higher number of neighbors is that the number of features can be really high, thus some noise is definitely present. Therefore, having a higher number of neighbors averages out the noise at least to some extent. Note that we did not lose much by choosing $I = m$ and $K = 30$, since this is the third-best option.

Ensemble-based scores. Here, we investigate the influence of the ensemble type (extra trees, random forests, bagging), and the feature subset size. Note that it is enough to consider only extra trees and random forests, since we consider the subset sizes of $n' \in \{\log_2 n, \sqrt{n}, n\}$, and the bagging approach is a special case of random forests when $n' = n$. The heat-plots in Fig. 4 show the performance of the rankings for both considered scores: Genie3 (Fig. 4a) and Random Forest (Fig. 4b).



(a) Genie3



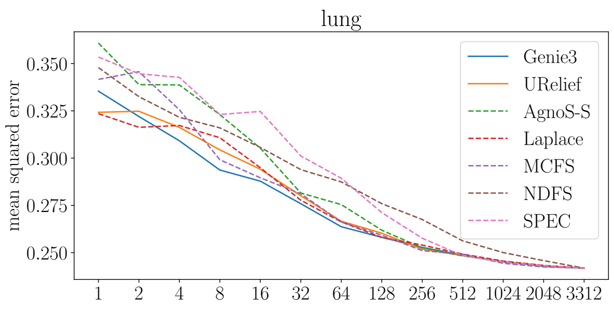
(b) RandomForest

Figure 4: Average ranks (across all data sets) of the considered parameter configurations (three different subset sizes and two ensemble methods), when the feature ranking score is fixed to one of two options (Genie3, RandomForest). Lower ranks correspond to better performance.

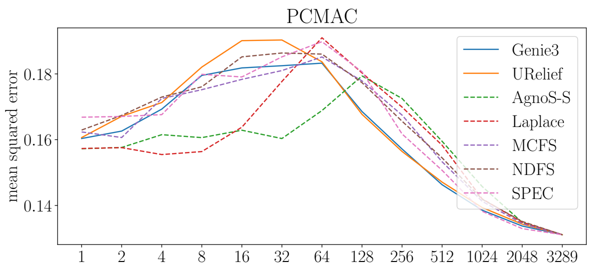
The performance of both scores mostly increases when the subset size n' decreases. A possible explanation for that is that such models overfit less to the data given that the number of features is high. It seems that this parameter is more important that the ensemble method, especially for the RandomForest score. Again, we did not lose much (performance-wise) when choosing the parameters for the main line of experiments to optimize for time complexity, avoiding the potentially quadratic number of operations in the number of features n.

6.4. Where in a Ranking are the Relevant Features?

Choosing a number of features to include in a model comes in handy when one wants to evaluate different feature rankings over different domains and many data sets. However, if a domain expert is interested in a particular data set, a more global view on the ranking might be desirable to understand the problem better. To achieve that, one can build predictive models that use the top 1, 2, 4, ..., 2^j, ..., $2^{floor(\log_2 n)}$, and n features and show the performance of the obtained models as a curve. Using the geometric (rather than the linear 1, 2, 3,...) sequence of numbers of top-ranked features makes the curve construction feasible (building, for example, more than 20 000 predictive models for the GLI-85 data set for the linear sequence might be too time-consuming), while still showing enough details at the beginning of the ranking, which is its most interesting part.



(a)



(b)

Figure 5: Error curves on two datasets (a) lung and (b) PCMAC for the feature rankings produced by the competing unsupervised ranking methods. Lower placement of a curve corresponds to better performance.

Here, we chose two data sets, for which the quality of the rankings is completely different[6]. Figure 5a shows the results for the `lung` data set, where with the increasing number of features, the MSE of the models mostly decreases. This means that, the rankings are mostly correct since relevant features have the highest importance. After URelief and Laplace both discover the most relevant feature, Genie3 quickly becomes the one (after four features) that ranks the features best.

The situation in Fig. 5b is different. These are the rankings for the `PCMAC` data set. We can see that all the algorithms successfully discover a few most relevant features, but then, mostly useless and noisy features are placed in the

[6]The results for the rest of the curves are available at `https://gitlab.com/Petkomat/unsupervised-feature-ranking`

ranking after them. This is evident from the increasing MSE. Only after 32 or 64 features, the next relevant ones are positioned and MSE decreases again.

When a domain expert sees those two figures, the expert can locate the position of the relevant features, which will help him understand the problem better. Of course, when we use all the features (the last point of every curve), MSE does not depend on the ranking and all the curves end at the same point.

7. Conclusions

In this work, we propose two novel approaches for unsupervised feature ranking. The first approach uses ensemble-based scores (Genie3 and RandomForest), computed from ensembles of predictive clustering trees. The second approach is an unsupervised version of the Relief algorithm (URelief).

After carefully choosing and discussing the evaluation procedure, we conduct an extensive empirical evaluation. We determine how the parameters of both approaches (ensemble-based scores and URelief) influence the quality of the rankings they produce. We show that for the ensemble-based scores, where parameters critically influence the time efficiency, the most efficient rankings fortunately also have the highest quality.

The comparative evaluation shows that the Genie3 score works better than the RandomForest score. We then compare Genie3 and URelief with five baselines showing that, on average, both proposed methods outperform the baselines. Since the difference in performance between Genie3 and the baselines is statistically significant, we recommend, all in all, to use the Genie3 score. This score can be efficiently computed from a parallelized ensemble of extremely randomized PCTs, where the feature subset size is set to $n' = \log_2 n$, which typically results in the time complexity of $\mathcal{O}(mn \log m \log n)$, where m and n are the numbers of examples and features.

Acknowledgements

The computational experiments presented here were executed on a computing infrastructure from the Slovenian Grid (SLING) initiative, and we thank the administrators Barbara Krašovec and Janez Srakar for their assistance.

Funding

This work was supported by the Slovenian Research Agency via the grant P2-0103 and a young researcher grant to MP and BŠ. SD and DK also acknowledge the support by the Slovenian Research Agency (via grants J7-9400, J7-1815, J2-9230, and N2-0128), and the European Commission (projects AI4EU (grant number 825619) and TAILOR (grant number 952215)).

[1] European Commission, On Artificial Intelligence - A European approach to excellence and trust, https://ec.europa.eu/digital-single-market/en/artificial-intelligence (2020).

[2] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI", Information Fusion 58 (2020) 82 – 115. `doi:https://doi.org/10.1016/j.inffus.2019.12.012`.

[3] A. Holzinger, G. Langs, H. Denk, K. Zatloukal, H. Müller, Causability and explainability of artificial intelligence in medicine, WIREs Data Mining and Knowledge Discovery 9 (4) (2019) e1312.

[4] M. Hoogendoorn, P. Szolovits, L. M. Moons, M. E. Numans, Utilizing uncoded consultation notes from electronic medical records for predictive modeling of colorectal cancer, Artificial Intelligence in Medicine 69 (2016) 53–61.

[5] D. Grissa, M. Ptra, M. Brandolini, A. Napoli, B. Comte, E. Pujos-Guillot, Feature selection methods for early predictive biomarker discovery using untargeted metabolomic data, Frontiers in Molecular Biosciences 3 (2016) 30.

[6] Y. Saeys, I. Inza, P. Larraaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19) (2007) 2507–2517.

[7] B. Pes, N. Dessì, M. Angioni, Exploiting the ensemble paradigm for stable feature selection: A case study on high-dimensional genomic data, Information Fusion 35 (2017) 132 – 147. `doi:https://doi.org/10.1016/j.inffus.2016.10.001`.

[8] M. Tsagris, V. Lagani, I. Tsamardinos, Feature selection for high-dimensional temporal data, BMC Bioinformatics 19 (1) (2018) 17.

[9] K. Bhardwaj, S. Patra, An unsupervised technique for optimal feature selection in attribute profiles for spectral-spatial classification of hyperspectral images, ISPRS Journal of Photogrammetry and Remote Sensing 138 (2018) 139–150.

[10] S. Galelli, G. B. Humphrey, H. R. Maier, A. Castelletti, G. C. Dandy, M. S. Gibbs, An evaluation framework for input variable selection algorithms for environmental data-driven models, Environmental Modelling & Software 62 (2014) 33 – 51.

[11] Y. Zhou, R. Zhang, S. Wang, F. Wang, Feature selection method based on high-resolution remote sensing images and the effect of sensitive features on classification accuracy, Sensors 18 (7).

[12] P. Kralj Novak, J. Smailović, B. Sluban, I. Mozetič, Sentiment of emojis, PloS one 10.

[13] J. A. DiMasi, R. W. Hansen, H. G. Grabowski, The price of innovation: new estimates of drug development costs, Journal of Health Economics 22 (2) (2003) 151 – 185.

[14] M. Petković, S. Džeroski, D. Kocev, Ensemble-based feature ranking for semi-supervised classification, in: P. Kralj Novak, T. Šmuc, S. Džeroski (Eds.), Discovery Science, Springer International Publishing, 2019, pp. 290–305.

[15] M. Petković, D. Sašo, D. Kocev, Feature ranking for semi-supervised learning, Machine Learning JournalSubmitted.

[16] G. Doquet, M. Sebag, Agnostic feature selection, in: Machine Learning and Knowledge Discovery in Databases European Conference, ECML PKDD 2019, Wrzburg, Germany, September 1620, 2019, LNCS 11906, 2020, pp. 343–358. `doi:10.1007/978-3-030-46150-8_21`.

[17] B. Seijo-Pardo, V. Bolón-Canedo, A. Alonso-Betanzos, On developing an automatic threshold applied to feature selection ensembles, Information Fusion 45 (2019) 227 – 245. `doi:https://doi.org/10.1016/j.inffus.2018.02.007`.

[18] H. Blockeel, D. L. Raedt, J. Ramon, Top-down induction of clustering trees, ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning (1998) 55–63.

[19] K. Kira, L. A. Rendell, The feature selection problem: Traditional methods and a new algorithm, in: Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI'92, AAAI Press, 1992, pp. 129–134.

[20] I. Kononenko, M. Robnik-ikonja, Theoretical and Empirical Analysis of ReliefF and RReliefF, Machine Learning Journal 55 (2003) 23–69.

[21] R. Zhang, F. Nie, X. Li, X. Wei, Feature selection with multi-view data: A survey, Information Fusion 50 (2019) 158 – 167. `doi:https://doi.org/10.1016/j.inffus.2018.11.019`.

[22] V. Bolón-Canedo, A. Alonso-Betanzos, Ensembles for feature selection: A review and future trends, Information Fusion 52 (2019) 1 – 12. `doi:https://doi.org/10.1016/j.inffus.2018.11.008`.

[23] S. Solorio-Fernndez, J. Carrasco-Ochoa, J. Martnez-Trinidad, A review of unsupervised feature selection methods, Artificial Intelligence Review 53 (2020) 907–948.

[24] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, ACM Computing Surveys (CSUR) 50 (6) (2018) 94.

[25] X. He, D. Cai, P. Niyogi, Laplacian score for feature selection, in: Proceedings of the 18th International Conference on Neural Information Processing Systems, NIPS05, MIT Press, Cambridge, MA, USA, 2005, p. 507514.

[26] Z. Zhao, H. Liu, Spectral feature selection for supervised and unsupervised learning, in: Proceedings of the 24th International Conference on Machine Learning, ICML 07, Association for Computing Machinery, New York, NY, USA, 2007, p. 11511157. `doi:10.1145/1273496.1273641`.

[27] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 10, Association for Computing Machinery, New York, NY, USA, 2010, p. 333342. `doi:10.1145/1835804.1835848`.
URL `https://doi.org/10.1145/1835804.1835848`

[28] Z. Li, Y. Yang, J. Liu, X. Zhou, H. Lu, Unsupervised feature selection using nonnegative spectral analysis, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI12, AAAI Press, 2012, p. 10261032.

[29] E. Facco, M. dErrico, A. Rodriguez, A. Laio, Estimating the intrinsic dimension of datasets by a minimal neighborhood information, Scientific Reports 7. `doi:10.1038/s41598-017-11873-y`.

[30] L. Breiman, J. Friedman, R. Olshen, C. J. Stone, Classification and Regression Trees, Chapman & Hall/CRC, 1984.

[31] J. Levatić, Semi-supervised learning for structured output prediction, Ph.D. thesis, Jožef Stefan Postgraduate School, Ljubljana, Slovenia (2017).

[32] D. Kocev, C. Vens, J. Struyf, S. Džeroski, Tree ensembles for predicting structured outputs, Pattern Recognition 46 (3) (2013) 817–833.

[33] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.

[34] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123140. `doi:10.1023/A:1018054314350`.
URL `https://doi.org/10.1023/A:1018054314350`

[35] P. Geurts, D. Erns, L. Wehenkel, Extremely randomized trees, Machine Learning 36 (1) (2006) 3–42.

[36] D. Kocev, M. Ceci, T. Stepišnik, Ensembles of extremely randomized predictive clustering trees for predicting structured outputs, Machine learning In press.

[37] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, P. Geurts, Inferring regulatory networks from expression data using tree-based methods, PLoS One 5 (9) (2010) 1–10.

[38] M. Petković, D. Kocev, S. Džeroski, Feature ranking for multi-target regression, Machine Learning JournalAccepted.

[39] J. H. Friedman, Greedy function approximation: A gradient boosting machine, The Annals of Statistics 29 (5) (2001) 1189–1232.

[40] M. Petković, D. Kocev, S. Džeroski, Feature ranking with relief for multi-label classification: Does distance matter?, in: L. Soldatova, J. Vanschoren, G. Papadopoulos, M. Ceci (Eds.), Discovery Science, Springer International Publishing, 2018, pp. 51–65.

[41] O. Reyes, C. Morell, S. Ventura, Scalable extensions of the relieff algorithm for weighting and selecting features on the multi-label learning context, Neurocomputing 161 (2015) 168 – 182.

[42] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods 17 (2020) 261–272. `doi:https://doi.org/10.1038/s41592-019-0686-2`.

[43] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[44] D. Wettschereck, A study of distance based algorithms, Ph.D. thesis, Oregon State University, USA (1994).

[45] P. Cunningham, S. J. Delany, k-Nearest Neighbour Classifiers, Tech. Rep. UCD-CSI-2007-4, University College Dublin (2007).

[46] X. Zhu, A. B. Goldberg, R. Brachman, T. Dietterich, Introduction to Semi-Supervised Learning, Morgan and Claypool Publishers, 2009.

[47] L. Hubert, P. Arabie, Comparing partitions, Journal of Classification 2 (1) (1985) 193–218.

Chapter 8

Case Study: Predicting Mars Express Thermal Power Consumption

In this chapter, we present a case study where MTR feature ranking is used to explain the MTR model that predicts the electrical energy (or rather, the electrical current) needed for thermal regulation of the Mars Express (MEX) satellite – a European Space Agency satellite that has been orbiting Mars since 2008. To plan and optimize the scientific operations of MEX, its operators need to estimate in advance, as accurately as possible, the power consumption of the thermal subsystem, since only the remaining power can be allocated for scientific purposes. The energy allocated to the thermal subsystem is spent by 33 heaters placed in the main cube of the satellite, which keep the instruments at their working temperature.

A machine learning model for predicting the amount of the energy spent by the thermal subsystem is needed since, over the years, the heat-consumption model of the satellite has changed to the extent that the initial simulations do not suffice any more. Thus, ESA organized its first data challenge (Lucas & Boumghar, 2017) to motivate teams from around the globe to construct accurate models for predicting thermal power consumption of MEX.

It is worth mentioning that ESA has provided the competitors with five types of data. These were

- **SAA** (Solar Aspect Angles) data give the relative position of the satellite and the Sun.
- **DMOP** (Detailed Mission Operations Plans) data contain information about the execution of different subsystems' commands at a specific time.
- **FTL** (Flight dynamics TimeLine events) data contain the pointing and action commands that impact the position of MEX, such as pointing the spacecraft towards Earth or Mars.
- **EVT** (Miscellaneous Events) data contain the time intervals during which MEX was in Mars's shadow or records of the time points when MEX is in an apsis of its elliptical orbit.
- **LTDATA** (Long Term Data) contain the Sun–Mars distances and the solar constant.

All raw data entries were time-stamped (expressed in milliseconds) indicating when the entry was logged. The time span between two consecutive entries varies from less than a minute (SAA) to several hours (LTDATA). Thus, the data as given was not directly usable to a machine learning algorithm due to the incompatible time resolutions of the different components of the raw data, and the unstructured format of some of the entries given, for example, as text.

The competitors had to first extract meaningful features out of the data, and only then proceed to predictive modeling. Our team won the challenge by using random forests of predictive clustering trees (Breskvar et al., 2017). However, in a thorough subsequent analysis (Petković, Boumghar, et al., 2019), random forests were outperformed by gradient boosting (Friedman, 2002).

Both random forests and gradient boosting ensembles are tree ensemble methods that offer state-of-the-art predictive performance and can be explained by computing feature rankings from them. The latter is of crucial importance for the operators of MEX to understand and trust the developed machine learning models. After building the models, we thus computed a feature ranking from each of them, so that the operators could – in some sense – validate them. After the operators confirmed that the models follow their intuition, we continued our collaboration on a MEX-related task, where the obtained feature rankings played the main role.

More precisely, in April 2018, a new software was deployed to MEX, which reduces its gyroscope usage by 90%. This considerably changed the way the operators fly the satellite. Our task was to quantify the effects of gyroless flying on the thermal subsystem.

We approached this task by computing Genie3 feature rankings from two random forests, which both predicted the electrical currents through the 33 heaters of MEX. The first model was learned on the data from the time period before April 2018, when the gyroscopes were still in full use. The second model was learned from the data after this period, when the gyroscopes were mostly turned off.

When the obtained feature rankings were compared, some apparent differences popped up. For example, the feature importance scores (of all features) for one of the heaters located next to the gyroscopes were orders of magnitude lower than when the gyroscopes were still in full use. The reason for this is that gyroscopes (when turned on) emit heat, therefore, the heater located next to the gyroscope was mostly turned off. Moreover, besides the different scales of the scores, the relative importances, i.e., the ranks of the features are also quite different (Petković, Lucas, et al., 2019).

The papers included in this Chapter are:

- Petković M., Boumghar R., Breskvar M., Džeroski S., Kocev D., Levatić J., Lucas L., Osojnik A., Ženko B., Simidjievski N., Machine Learning for Predicting Thermal Power Consumption of the Mars Express Spacecraft. *IEEE Aerospace and Electronic Systems Magazine*, 34(7), 46-60, 2019

- Petković, Lucas, L., Kocev, D, Džeroski, S., Boumghar, R., and Simidjievski, N. Quantifying the Effects of Gyroless Flying of the Mars Express Spacecraft with Machine Learning. In *Proceedings of the 2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 9-16, Pasadena, CA, USA, 2019.

The contribution of Matej Petković to these papers are as follows. MP implemented the missing parts of the code for the experiments, participated in designing the experiments, carried out most experiments, and processed their results. Together with NS, he drafted the papers and revised them following the feedback from the co-authors and reviewers.

Feature Article:

DOI. No. 10.1109/MAES.2019.2915456

Machine Learning for Predicting Thermal Power Consumption of the Mars Express Spacecraft

***Matej Petković, Martin Breskvar, Sašo Džeroski, Dragi Kocev,* Jožef Stefan Institute, Slovenia, Jožef Stefan International Postgraduate School, Slovenia**
***Redouane Boumghar,* European Space Agency, ESOC, Data Analytics Team for Operations, Germany**
***Jurica Levatić,* Institute for Research in Biomedicine, Spain**
***Luke Lucas,* Mars Express, Mission Planning and Spacecraft Operations, Germany**
***Aljaž Osojnik, Bernard Ženko, Nikola Simidjievski,* Jožef Stefan Institute, Slovenia**

INTRODUCTION

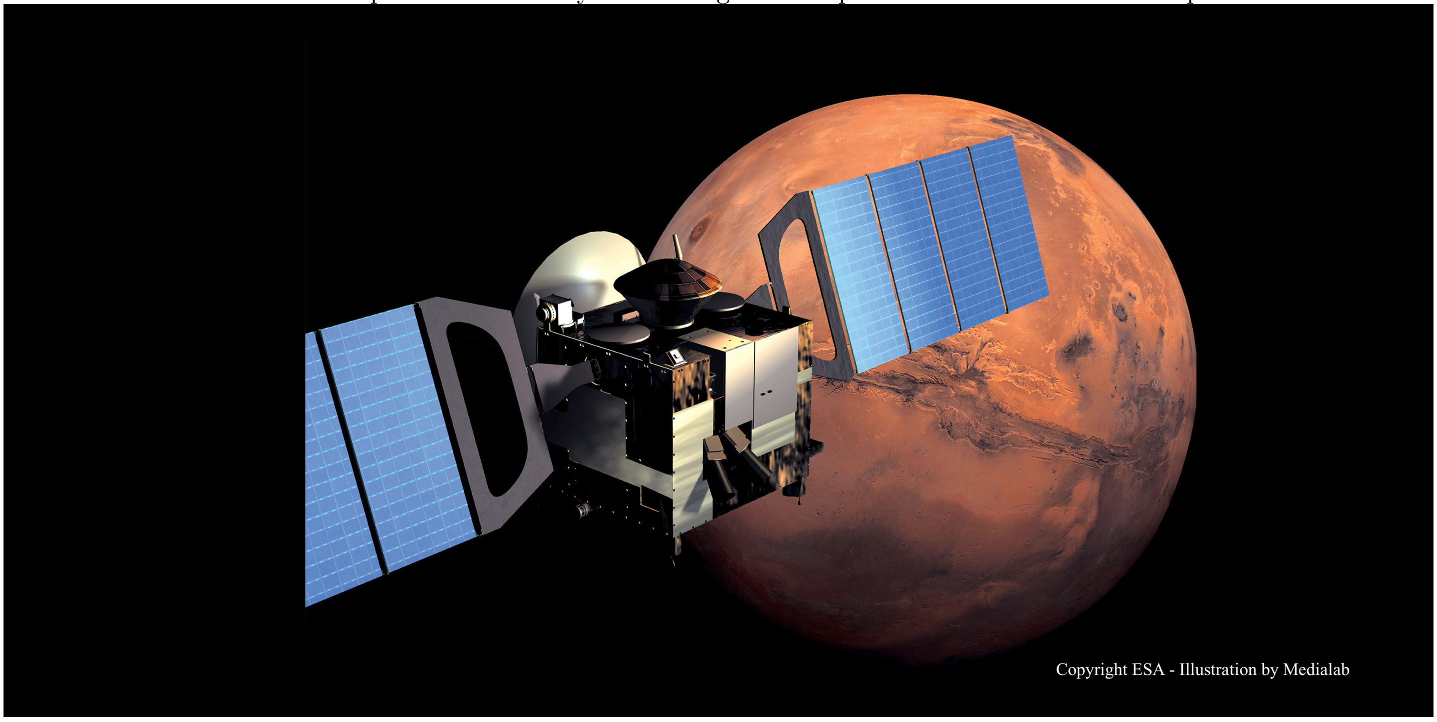
MARS Express (MEX), a spacecraft operated by the European Space Agency (ESA), is Europe's first spacecraft that orbits Mars. During its science operations, since the beginning of 2004, it has provided evidence of the presence of water above and below the surface of the planet [1], an ample amount of three-dimensional (3-D) renders of the surface as well as the most complete map of the chemical composition of Mars's atmosphere [2].

MEX is powered by electricity generated by its solar arrays and stored in batteries to be used during the eclipse periods. The scientific payload of the MEX consists of seven instruments that provide global coverage of the planet's surface, subsurface, and atmosphere. The instruments and on-board equipment have to be kept within their operating temperature ranges, spanning from room temperature for some instruments, to temperatures as low as -180 °C for others. In order to maintain these predefined operating temperatures, the spacecraft is equipped with an autonomous thermal system composed of 33 heater lines as well as coolers. The thermal system, together with the platform units, consumes a significant amount of the total generated electric power, leaving a fraction to be used for science operations.

Predicting the power consumption of the thermal system is a nontrivial task. However, due to the aging of the spacecraft and the decaying capacity of its batteries, it is a very crucial one for optimal planning and execution of science operations on MEX. The power consumption is a dynamic process that changes through time, depending on various external and internal factors, such as long-term exposure of the spacecraft to the Sun or heat generated by the on-board instruments. For instance, Figure 1 shows the effect of the radio transmitter during a communication pass, with interpolation between different temperature sensors of the $-Y$ face of the spacecraft. Temperatures fluctuate by up to 28 °C due to these two different ON/OFFconditions. Current attempts at modeling and predicting the power consumption involve manually constructed models that are based on simplified first-principle models, expert knowledge, and experience. Given MEX's current condition, this prompts for a more accurate predictive model of the thermal power consumption (TPC), which would yield prolonged operating life.

This motivated the organization of the first ESA's data mining competition—the Mars Express Power Challenge [3]. The focus of the challenge was the development of

Authors' current addresses: M. Petković, M. Breskvar, S. Džeroski, D. Kocev, Jožef Stefan Institute 1000, Ljubljana, Slovenia, Jožef Stefan International Postgraduate School SI-1000, Ljubljana, Slovenia, E-mail: (matej.petkovic@ijs.si). R. Boumghar, Data Analytics Team for Operations, ESOC, European Space Agency, Cologne, Germany. J. Levatić, Institute for Research in Biomedicine 08028, Barcelona, Spain. L. Lucas, Mars Express, Mission Planning and Spacecraft Operations 64293, Darmstadt, Germany. A. Osojnik, B. Ženko, N. Simidjievski, Jožef Stefan Institute 1000, Ljubljana, Slovenia and also with University of Cambridge, Cambridge CB3 0FD, United Kingdom. E-mail: (nikola.simidjievski@ijs.si).
Manuscript received August 31, 2018, revised November 26, 2018, and ready for publication May 6, 2019.
Review handled by M. D. R-Moreno.
0885-8985/19/$26.00 © 2019 IEEE



specialized approaches for constructing models that are able to accurately estimate and predict the MEX's TPC given only measured telemetry data. For this task of predictive modeling, machine learning approaches offer a different, yet more accurate solution to modeling the complex relationship between the telemetry and the power consumption than a human expert.

Machine learning is an area in the realm of artificial intelligence, which studies algorithms with the ability to learn, i.e., algorithms that improve their performance through knowledge gathered from experience (data). Their ability to capture and describe patterns in complex data makes them a valuable asset for studying a variety of phenomena in different domains from life sciences, earth sciences, social and behavioral sciences. In the context of the MEX challenge, machine learning algorithms for predictive modeling aim at constructing a model that can capture complex relationships in the data. In turn, such models accurately estimate future values of power consumption for each of the 33 heater lines and coolers (target variables/features) given measured telemetry data (descriptive variables/features).

In our previous work [4], we presented the machine learning pipeline solution that won the Mars Express Power Challenge. The winning solution first transforms the raw telemetry data into carefully constructed features with 1-min time resolution between values, rendering a massive dataset. Next, it uses the method of Random Forest of Predictive Clustering Trees (RF-PCTs) [5] to construct 33 predictive models for each of the 33 target variables. Finally, it outputs a predicted value of each target variable for every hour of one Martian year in the future (1.88 Earth years). The proposed solution performed better than the ~40 other competing solutions while being more accurate than the models currently in use at ESA by an order of magnitude.

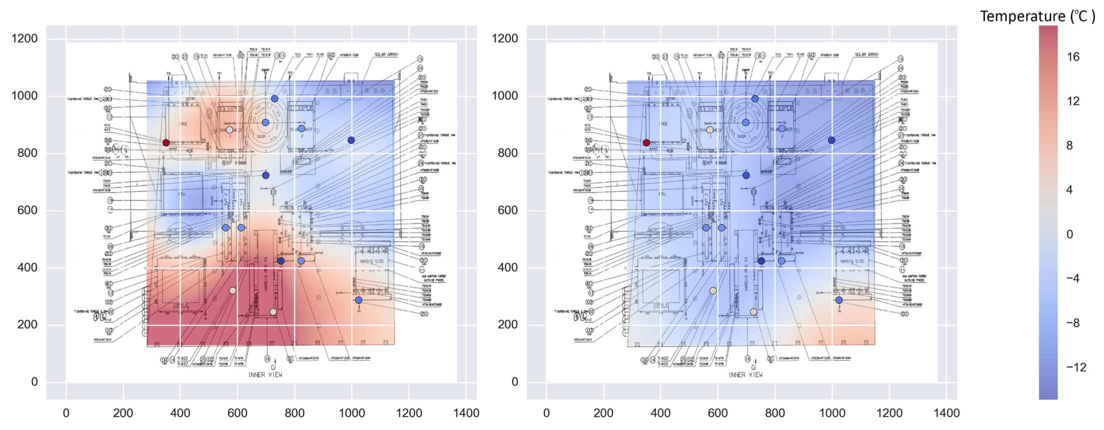


Figure 1.
Interpolated thermal effects on the MEX's $-Y$ face, with radio transmitter turned on (left) and off (right).

However, the premium predictive accuracy of the winning solution came at a cost of substantial computational overhead. In this paper, we extend the work presented in [4] to address this issue. In particular, we propose an update to the winning solution which aims at efficiently constructing predictive models of MEX's TPC, while still being able to maintain good predictive performance. More specifically, we consider updates of the pipeline along two dimensions: 1) constructing data with different data granularity in the learning process and 2) using different machine learning methods which can efficiently learn accurate predictive models. The former considers engineering features from the raw telemetry data at different time resolutions coarser than 1 min, thus, reducing the size of the dataset used in the learning phase. The latter considers both local and global methods for multitarget regression. Local methods construct a model for each target variable separately. Here, besides the winning method RFs of PCTs [5], we also consider XGBoost [6], a recent efficient implementation of Stochastic Gradient Boosting [7]. In contrast, global methods produce a predictive model able to predict several target variables simultaneously [8]. To this end, we consider global RFs of PCTs for multitarget regression, an extension of the local version, which can construct single model for all 33 target variables, therefore, substantially reducing the computational time needed for obtaining a solution [5], [9].

In sum, the main task that we address is: Given three Martian years of telemetry data (August 22, 2008 to April 14, 2014), use machine learning to efficiently construct predictive model that accurately predicts the values of the electric current through the 33 thermal power consumers for the subsequent Martian year (April 14, 2014 to March 1, 2016).

The remainder of this paper is organized as follows. In Section "RELATED WORK," we provide an overview of the work related to machine learning applications for space-exploration research. Section "DATA" presents and discusses the tasks of data preparation and preprocessing. Section "MACHINE LEARNING METHODS" presents the machine learning methods used in this study. In Section "EXPERIMENTAL SETUP," we present the experimental setup for evaluating the proposed extensions of the machine learning pipeline. Section "RESULTS" presents and discusses the results of the empirical evaluation. Finally, Section "CONCLUSION" concludes the paper and suggests directions for further work.

RELATED WORK

Machine learning offers an ample amount of methods that tackle predictive tasks in real-life domains [10], [11]. These methods have been applied for predicting discrete output values (classification), continuous output values (regression), even structured outputs as in gene networks, image classification, text categorization, etc., [12].

The challenges typically addressed in space-exploration research are associated with high cost of failure [13]. For instance, the remote spacecraft are typically equipped with processors and memory lagging decades behind the state of the art. Next, the development and launch of a space mission is expensive and there is little or no opportunity for repair. In this context, the utility of machine learning approaches has been proven to be a valuable asset. In particular, many applications of machine learning address the task of anomaly detection in spacecraft, where a typical task considers monitoring the status of the on-board equipment. Such analyses of telemetry data are performed using neural networks [14], relevance vector machine [15], or by applying seasonal decomposition methods (linear regression together with the nearest neighbors) [16].

Machine learning can be used not only for estimating the current state of a spacecraft, but also predicting its future ones and therefore allowing for autonomous decisions. In our previous work [4], we propose a solution for predicting spacecraft's power consumption, which can be used for optimizing its operation. This works closely relates to the one of [17], where the authors propose Random Forests (RFs) for predicting temperature of the instruments to optimize battery usage during eclipses.

Another important challenge relates to safe ground movement of autonomous (space) rovers or robots. Hernández et al. [18] address this issue by using support vector machines to recognize and avoid dangerous objects. In similar context, Giusti et al. [19] address the task of image analysis for automated navigation systems. Here, with deep neural networks as the underlying machine learning method, autonomous drones are utilized for tracking forests paths.

Finally, machine learning can be also utilized to learn or simplify a physical model of a spacecraft or a model of its environment. In [20], Finn et al. employ RFs to simplify the exact physical model for complex and dynamic radiation environment in the Van Allen belts. In a similar context, McGovern and Wagstaff [13] outline several studies which address the challenges of spacecraft operating in high-radiation environments (beyond Earth's magnetosphere and ionosphere) and the reliability of machine learning algorithms applied in these scenarios. In particular, these studies propose variants of traditional ML approaches (*k*-means and SVMs) robust to potential data corruption on the disks due to the various levels of radiation.

DATA

In the typical machine learning setting, the input in a learning algorithm is (training) data which embodies the experience. The data consist of training examples (also referred to as instances or measurements) and their properties (also referred to as features or attributes). The features, numerical (i.e., continuous), or nominal (i.e., discrete), can either describe the data or specify the desired output of the algorithm. In the context of predicting MEX's TPC, an example is a time period, while features are derived from context and observations data.

In this paper, we use data provided by ESA [3] that consists of raw telemetry data (context data) and measurements of the electric current of 33 thermal power lines (observation data), for three Martian years of MEX operations. We refer to these data as the training data $\mathscr{D}_{\mathrm{TRAIN}}$. For the fourth Martian year of the operation, the context part of the data was used for generating the predicted values, that in turn where evaluated using the real measured observation data. We refer to these data as the test data $\mathscr{D}_{\mathrm{TEST}}$.

The *observation data* consists of the electric current measurements of the 33 power consumers, recorded once or twice per minute. The *context data* consists of five components:

- Solar Aspect Angles (SAA) data contain the angles between the Sun–MEX line and the axes of the MEX's coordinate system.
- Detailed Mission Operations Plans (DMOP) data contain the information about the execution of different subsystems' commands at a specific time.
- Flight dynamics TimeLine events (FTL) data contain the pointing and action commands that impact the position of MEX, such as pointing the spacecraft toward Earth or Mars.
- Miscellaneous Events (EVTF) data contain time intervals during which MEX was in Mars's shadow or records of the time points when the MEX is in apsis of its elliptical orbit.
- Long Term Data (LTDATA) contain the Sun–Mars distances and the solar constant.

All raw data entries are time-stamped (expressed in milliseconds) indicating when the entry was logged. The time span between the two consecutive entries varies from less than a minute (SAA) to several hours (LTDATA). For a detailed description of the task and the data, we refer the reader to [3], [21].

The raw data are not directly applicable to a machine learning algorithm due to two main reasons: i) incompatible time resolutions of the different components of the raw data, and ii) unstructured format of some of the entries, such as text, that are not readily usable for machine learning algorithms. Therefore, to construct an appropriate dataset for a learning algorithm, we preprocess the raw data in two phases: conveying data time resolution (time interval between two consecutive examples) and engineering new (more informative) features from different parts of the context data that may yield to better predictive performance.

The first phase relates to choosing an appropriate time resolution Δt of the dataset, and divide the time span $[t_{\mathrm{FIRST}}, t_{\mathrm{LAST}})$ into subintervals $[t_i, t_{i+1})$, where $t_{i+1} = t_i + \Delta t$. Here, t_{FIRST} is the first time-stamp in the $\mathscr{D}_{\mathrm{TRAIN}}$, and t_{LAST} is the last time-stamp in the $\mathscr{D}_{\mathrm{TEST}}$.

The second phase considers constructing more informative features. The value of a given feature for a particular time interval is obtained by aggregating measurements from the time interval at hand that correspond to one or more components from the raw telemetry data.

Due to issues with the spacecraft communication in some periods, some measurements are missing from the both the context and observations data. In principle, the machine learning methods employed in this study can handle data with missing values. However, longer periods with contiguous missing values can substantially damage the accuracy of the learned predictive models as well as add an additional computational overhead. For this reason, we remove examples with missing observation data for time periods longer than 10 min. On the other hand, in the context data, we interpolate the examples with missing values for time periods shorter than 10 min, or leave them intact otherwise.

In the following sections, we describe the groups of features constructed in the preprocessing step of the pipeline.

ENERGY INFLUX FEATURES

There are seven features in this group: one for solar panels and one for each of the six sides of the cuboid of MEX. The features describe the amount of solar energy that is collected through a given surface in a given time interval $[t_i, t_{i+1})$. The solar energy collected by a side of cuboid directly influence the amount of the energy used by the thermal lines that maintain the temperature in that part of the spacecraft. The solar energy collected by the solar panels influence the amount of available energy that can be stored in spacecraft's batteries.

The amount of energy collected by a given surface is proportional to the product of the effective area A_{eff} of the surface and the solar constant c. If the area A is given, we compute A_{eff} as $A_{\mathrm{eff}} = A \max\{\cos\alpha, 0\}$, where α is the angle between the Sun–MEX line and the outer normal $\vec{n}$ to the surface (see Figure 2). Without any loss of generality, we assume $A = 1$ for all surfaces, as the machine learning methods that we use, are invariant to monotonic transformations of features. The values of α for each of the seven surfaces were computed directly from the SAA data, while c was given in LTDATA. In addition to the effective area and the solar constant, (pen)umbras have a considerable impact on the energy influx. We define the

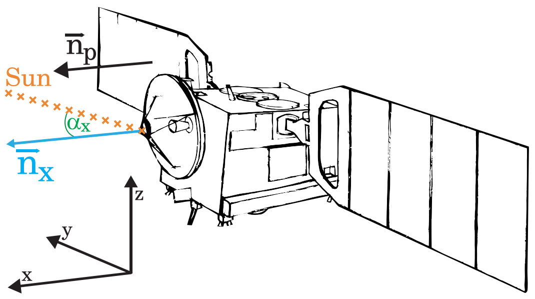


Figure 2.
Illustration of the MEX spacecraft and its coordinate axes x, y, and z, that correspond to *front*, *left*, and *up* sides of MEX, respectively. α_x denotes the SAA of the front side, i.e., the angle between the normal $\vec{n}_x$ and the Sun-MEX line. $\vec{n}_p$ denotes the normal of the panels.

amount of the energy E_S^i that pass through the surface S at time interval $[t_i, t_{i+1})$ as

$$E_S^i = \int_{t_i}^{t_{i+1}} A_{\text{eff}}(t)c(t)U(t)\text{d}t$$

where U is the *umbra coefficient*, an approximation of the proportion of Sun visible from the spacecraft. U takes the value $U(t) = 0$ if the spacecraft is in an umbra, $U(t) = 0.5$ if the spacecraft is in a penumbra, and $U(t) = 1$ otherwise. Instead of calculating exact integrals for E_S^i, we approximate the values using the trapezoid-rule.

HISTORICAL ENERGY INFLUX FEATURES

The thermal state of the spacecraft depends not only on the current energy influx, but also on the energy influx in the past. To capture this, we construct historical energy influx features for each of the seven surfaces. A given historical feature for surface S at time t_i is computed as a sum of energy influx during given historical time-frame:

$$H_S^i = \sum_{j=1}^{H} E_S^{i-(j-1)} \tag{1}$$

where H is the number of time intervals included in the historical feature. To account for different impacts of the historical energy influx, we construct historical features with different time-frames for different values of H, given in Section “PARAMETER INSTANTIATION.”

DMOP FEATURES

The DMOP data contain log of commands issued to different MEX subsystems. The names of commands have been obfuscated; however, the available documentation reveals two variants of events: 1) events that contain information about the subsystem and command that has been executed (e.g., ASXX383C) and 2) events that represent flight dynamics events (e.g., MAPO.000005).

The first four characters of the first variant represent the subsystem while the rest represent the command and its parameters. In the second variant the first four characters represent the name of the event, followed by a number that indicates the number of occurrences. Given that these events have different impact on the temperatures of various subsystems of the spacecraft, it is safe to assume that they impact the thermal subsystems differently.

More specifically, we assume that there is a significant delay between triggering of a subsystems' command and its actual effect on the thermal state of the spacecraft. Therefore, from the raw DMOP data, we construct features that encode this information of delayed effect in terms of “time since last activation” of a specific subsystem command.

The values of the DMOP features are calculated as follows:

$$f_k^i = \begin{cases} 0, & \text{if } k \text{ is activated at } t_i \\ \min(f_k^{i-1} + \Delta t, \theta) & \text{otherwise,} \end{cases}$$

where f_k^i denotes the value of feature corresponding to event k at time t_i. Note that, here we also assume that all of the subsystems were deactivated at the first time point (i.e., $f_k^0 = \theta$). The θ regulates the effect of a given event diminishing with time, rendering its influence unimportant at some point. We selected this threshold to be 1 day ($\theta = 1440$, the number of minutes in a day). Table 1 presents these calculations of features.

We construct such features for each flying dynamic event (17 features), each subsystem–command pair (345 features) and each subsystem in case different commands are issued to it (15 features). We also construct binary indicators for each subsystem and flying dynamic event (34 features in total), where a feature f_k^i has value of 1 if the subsystem was triggered within the time-step t_i, and 0 otherwise.

FTL FEATURES

The FTL data contain logs of pointing events and their time ranges, where simultaneous events are also possible. For each pointing event in time interval $[t_i, t_{i+1})$, a feature has value that equals the proportion of the time in $[t_i, t_{i+1})$, during which the event is in progress. Since the duration of events is typically longer than Δt, the values of the features are mostly 1 (event is in progress), or 0 (event not in progress). This approach renders 23 FTL features in total.

FINAL DATASETS

Table 2 presents the important details regarding the final constructed datasets used further in the experiments.[1]

[1] The data are accessible at http://spacelab.ijs.si/.

Table 1.

Illustration of the DMOP Features that Encode the Time Since Last Activation of a Given Subsystem Command

Raw Data		DMOP Features			
t	Command	APSF28A1	ASXX383C	ATTTF030A	ASXX303A
t_1	*none*	1440	1440	1440	1440
t_2	**ASXX383C**	1440	**0**	1440	1440
t_3	**ATTTF030A**	1440	Δt	**0**	1440
⋮	⋮	⋮	⋮	⋮	⋮
t_{14}	**ASXX303A**	1440	$12\Delta t$	$11\Delta t$	**0**

MACHINE LEARNING METHODS

Considering predictive modeling, many machine learning methods struggle with the problem of overfitting. Overfitting occurs when a method learns a model with a very good performance on the provided training data, but has limited generalization power and performs poorly on data unseen during learning. In machine learning, there is a long tradition of developing methods that address this problem by learning multiple (diverse) models and combining their outputs instead of just learning a single model. These methods are referred to as ensemble methods or ensembles. An ensemble is a set of (base) predictive models constructed with a given algorithm, that is expected to lead to predictive performance gain over an individual model by combining the predictions of its constituents. In this study, we employ two types of ensembles: RF-PCTs (both local and global version of the algorithm for multi-target regression) [5], [22] and Stochastic Gradient Boosted Trees (XGBoost) [6], [7].

Table 2.

Properties of the Final Datasets Given Different Granularity Δt

Δt [min]	Number of examples	Number of features	Size [MB]
1	3922 895	459	10 090
5	784 773	459	2048
10	392 474	459	1045
15	261 697	459	702
30	130 900	459	353
60	65 493	445	168

RF OF PREDICTIVE CLUSTERING TREES

RF [22] is an ensemble method that learns a set of tree-based predictive models and combines their prediction. The base models are learned from random bootstrap samples of the training set, where for each tree at each tree node a random feature subset (with a user defined size) is considered for selecting the best split. Such approach allows for constructing a set of diverse predictive models that can differ both in size and performance.

In this study, the RF ensembles consist of Predictive Clustering Trees (PCTs) [23]. PCTs have a tree structure that includes internal nodes and leaves. The internal nodes contain tests on the descriptive variables (i.e., the different features extracted with preprocessing), while leaves give predictions for the target variable (i.e., power consumption of a thermal line). PCT refers to a hierarchy of clusters with each node corresponding to a cluster. In particular, the top-node of a PCT corresponds to one cluster (group) containing all data points. This cluster is then recursively partitioned into smaller clusters while moving down the tree. The leaves represent the clusters at the lowest level of the hierarchy and each leaf is labeled with its cluster's centroid/prototype (the average of the target variable is the prediction made by the leaf).

RF of PCTs is a generalization of the traditional RF ensemble of regression trees [24], in terms of addressing structured output prediction tasks [5], [25]. While the traditional RF is able to predict values of a single numeric target variable at a time (i.e., is a local method for Multi-target Regression), the RF of PCTs ensemble allows also for predicting several target variables simultaneously (i.e., is a global method for Multitarget Regression). The algorithm for learning a RF of PCTs is presented in Algorithm 1. Namely, it takes three inputs: 1) training data $\mathscr{D}_{\mathrm{TRAIN}}$, as well as two hyperparameters denoting, 2) the number of trees M in the ensemble, and 3) the size of the feature subset considered at each node split f. Each PCT in the ensemble is learned with greedy recursive top-down induction algorithm on a random bootstrap sample of the input dataset (line 3 of the Algorithm 1).[2]

The PCT-induction algorithm (lines 8–15 of the Algorithm 1) starts in the root node of the tree by selecting a test from set of candidate tests that are generated by a random feature subset $\mathcal{F}$ of size f. The best test is greedily chosen by a heuristic function that typically measures the impurity of the target values of the examples in the subsets E_i of the set E that this test results in. The goal of the heuristic function is to guide the algorithm toward small trees with good predictive performance. In the global MTR setting, this heuristic function is the average variance of the targets. In our case, with numeric features, every test is of form $x_i < \vartheta$, for some threshold ϑ, and partitions the set E into two subsets. These are the set E_1 of test-positive instances for which $x_i < \vartheta$ and the set E_2 of test-negative instances for which $x_i \geq \vartheta$, i.e., $\mathcal{P}^* = \{E_1, E_2\}$ (line 9).

The procedure is recursively repeated on the subsets E_i to construct the subtrees (line 12) until a stopping criterion is satisfied (e.g., the minimal number of examples in a leaf is reached or the heuristic score no longer changes, etc.). In turn, a leaf node is created and its prototype is computed (line 15). In the global MTR setting, the prototype is a vector of average target values of the examples in the leaf. The prototypes are used for prediction.

Algorithm 1: Random Forest of PCTs ($\mathscr{D}_{\text{TRAIN}}$, M, f)

```
1:  $\mathcal{RF} = \emptyset$
2:  **for** $m = 1, 2, \ldots, \mathrm{M}$ **do**
3:      $E = bootstrapSample(\mathscr{D}_{\text{TRAIN}})$
4:      $\mathcal{T} = inducePCT(E, f)$
5:      add $\mathcal{T}$ to $\mathcal{RF}$
6:  **return** $\mathcal{RF}$
7:  **where** $inducePCT(E, f)$
8:      $\mathcal{F}$ = random sample of $f$ features
9:      $(t^*, \mathcal{P}^*) = findBestTest(E, \mathcal{F})$
10:     **if** $t^* \neq none$ **then**
11:         **for each** $E_i \in \mathcal{P}^*$ **do**
12:             $subtree_i = inducePCT(E_i, f)$
13:         **return** $\mathscr{N}(t^*, \bigcup_i \{subtree_i\})$      // internal node
14:     **else**
15:         **return** $\mathscr{L}(\mathrm{Prototype}(E))$      // leaf node
```

Finally, the RF of PCTs algorithm outputs a set of PCTs, whose predictions are combined (averaged per value) to obtain the final ensemble prediction. The reasons for using RF of PCTs are twofold: its state-of-the-art predictive performance [5], and ability to calculate feature importance scores, i.e., ranking of the features w.r.t. their importance for the target variables.

[2] The RF of PCT framework is implemented in the CLUS system available at http://source.ijs.si/ktclus/clus-public.

Namely, RFs can measure how much each feature contributed to the quality of the predictive model. For this purpose, we used the Genie3 algorithm [26], for which the motivation is the following. If a relevant feature x_i is part of a test $x_i < \vartheta$, then the heuristic score h^* (reduction of the variance) of this split is high. Additionally, the features that appear in the tests of nodes at lower depths, e.g., in the root, influence more examples compared to the ones appearing deeper in the tree, so the former are intuitively more important. Therefore, the Genie3 importance score is defined as

$$\mathrm{importance}_{\mathrm{GENIE3}}(x_i) = \frac{1}{|\mathcal{RF}|} \sum_{\mathcal{T} \in \mathcal{RF}} \sum_{\mathscr{N} \in \mathcal{T}(x_i)} h^*(\mathscr{N}) |E(\mathscr{N})| \tag{2}$$

where $\mathcal{T}(x_i)$ is the set of nodes in the tree $\mathcal{T}$ in which x_i is part of the test, $h^*(\mathscr{N})$ is the value of the variance reduction function in the node $\mathscr{N}$, and $|E(\mathscr{N})|$ is the number of examples that come to the node $\mathscr{N}$.

Further in this paper, we denote the local and the global versions of RF PCTs for multitarget regression with L-RF and G-RF, respectively.

GRADIENT BOOSTED TREES

Gradient boosting [27] refers to a class of boosting ensemble methods [10] which aims at learning a set of predictive models focusing on difficult observations in the data. In contrast to RF ensembles that first learn the ensemble constituents from random parts of the dataset and in turn combine their outputs, boosting ensembles are constructed iteratively. At each boosting iteration, a new weak base model that corrects the error made by the ensemble thus far is learned and added to the set creating a stronger model at each step. Typically, such weak models have simple structure and thus perform slightly better than random models.

In general, boosting methods differ in the type of base models they employ and how the learning is performed. The former is task related, where typical base models considered include: logistic regressors (for classification tasks), linear regressors (for regression tasks), or decision trees (for both classification and regression tasks). Regarding how the learning is performed, boosting base models are either constructed to focus on hard examples identified in previous iterations [28] or by minimizing the empirical risk via steepest gradient descent [27]. In this paper, we focus on the latter category, referred to as Gradient Boosting.

An outline of the gradient boosting algorithm for constructing ensembles with decision trees is presented in Algorithm 2. The algorithm takes three inputs: 1) a training dataset $\mathscr{D}_{\text{TRAIN}}$, 2) number of boosting iterations M, 3) a loss function $\mathcal{L}$, and 4) a learning rate η. First, an initial default (weak) model is learned on the whole dataset that minimizes the loss function $\mathcal{L}$. Given that the gradient boosted method aims at optimizing the loss function, it is important that $\mathcal{L}$ should be convex and differentiable. A typical choice of $\mathcal{L}$ is L_2 square-loss function [10]. In turn, gradient boosting at each step m learns a new model on the pseudo-residuals, i.e., the discrepancy value between the true and the predicted value of the ensemble in the previous iterations (line 4).

However, such straightforward approach can very easily overfit to the training data. In order to address the problem of overfitting, more sophisticated gradient boosting methods implement two different mechanisms: a learning rate and random data sampling procedure. The former regulates the influence of the prediction of each subsequent model added in the ensemble set (line 6). The latter, referred to as Stochastic Gradient Boosting [7], employs additional random data sampling procedure: Each model is learned and evaluated on different random subsamples of the training data (line 3).

Algorithm 2: Stohastic Gradient Boosted Trees ($\mathscr{D}_{\mathrm{TRAIN}}$, M, $\mathcal{L}$, η)

```
1: $GBT_\emptyset = defaultModel(\mathscr{D}_{\mathrm{TRAIN}}, \mathcal{L})$
2: for $m = 1, 2, \ldots, \mathrm{M}$ do
3:     $E = randomSample(\mathscr{D}_{\mathrm{TRAIN}})$
4:     $\mathscr{R}_{\mathrm{m}} = \frac{\partial \mathcal{L}(E, GBT_{m-1}(E))}{\partial GBT_{m-1}}$    // compute pseudo-residuals
5:     $\mathcal{T} = induce(\mathscr{R}_{\mathrm{m}}, \mathcal{L})$
6:     $GBT_m = GBT_{m-1} + \eta\mathcal{T}$
7: return $GBT$
```

In this study, we employ XGBoost [6]—a recent efficient implementation of Stochastic Gradient Boosting [7] that employs regression trees as proposed in [24] as base models. In the paper, we denote the XGBoost ensembles with XGB.

EXPERIMENTAL SETUP

PARAMETER INSTANTIATION

Granularity: The data granularity is defined by the length Δt of the time interval that corresponds to one example in the dataset. We construct the predictive models using datasets with $\Delta t \in \{1, 5, 10, 15, 30, 60\}$ (measured in minutes) where $\Delta t = 1$ corresponds to the dataset used in [4].

Historical features: For the dataset with finest granularity $\Delta t = 1$, we consider the following numbers H of historical intervals from (1): $H \in \{4, 16, 32, 64, 128\}$. Consequently, the historical time span ranges from 4 min to 128 min. The choices of the historical intervals in the datasets with courser granularity are presented in Table 3. In total, these values yield 35 features in the dataset with $\Delta t \leq 30$ and 21 features dataset with $\Delta t = 60$.

Table 3.

Values of the Number of Historic Intervals H and the Corresponding Historic Time Spans, for Different Granularities. Δt

Δt	Values of H	Time spans
1	$\{4, 16, 32, 64, 128\}$	$\{4, 16, 32, 64, 128\}$
5	$\{1, 3, 6, 13, 25\}$	$\{5, 15, 30, 65, 125\}$
10	$\{1, 2, 3, 6, 13\}$	$\{10, 20, 30, 60, 130\}$
15	$\{1, 2, 3, 4, 9\}$	$\{15, 30, 45, 60, 135\}$
30	$\{1, 2, 3, 4, 5\}$	$\{30, 60, 90, 120, 150\}$
60	$\{1, 2, 3\}$	$\{60, 120, 180\}$

RF parameters: To constrain the size of the trees in the RFs, we specify a minimal number of examples in the leaves m_{LEAF} for each tree. Since the number of instances in the datasets is inversely proportional to Δt, we set the minimal number of instances for the $\Delta t = 1$ experiments to 500, while for the others we set them to $m_{\mathrm{LEAF}} = 500/\Delta t$. Additionally, we set the total number of trees in the RFs to 200, where one quarter of the features is considered at every split when growing the trees, i.e., $f = 0.25|\mathcal{F}|$ in Algorithm 1.

XGBoost Parameters: Analogously, to constrain the size of the trees, we set maximal depth of each tree in the ensemble to 11. The learning rate parameter is set to 0.1. Additionally, to address potential overfitting issues, for every boosting iteration 60% of the features and 60% of the examples are randomly chosen for training. The maximum number of boosting iteration (ensemble constituents) is set to 200 with an early-stop option, i.e., if the newly added trees in the ensemble do not improve the performance of the ensemble over five consecutive boosting iterations the algorithm stops.

EVALUATION PROCEDURE

The dataset $\mathscr{D}$ consists of examples $(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x}$ is a vector of feature values (features are described in Section "DATA"), and $\boldsymbol{y}$ is a vector of 33 target values, i.e., the electrical currents trough the heaters and coolers.

The dataset is divided into two parts: $\mathscr{D}_{\mathrm{TRAIN}}$ that describes the state of the spacecraft throughout the first three Martian years, and $\mathscr{D}_{\mathrm{TEST}}$ that describes the state of the spacecraft throughout the fourth Martian year. All predictive models, i.e., the approximations $\hat{\boldsymbol{y}} : \boldsymbol{x} \mapsto \hat{\boldsymbol{y}}(\boldsymbol{x})$ of the true mappings $\boldsymbol{y} : \boldsymbol{x} \mapsto \boldsymbol{y}(\boldsymbol{x})$, were learned on $\mathscr{D}_{\mathrm{TRAIN}}$.

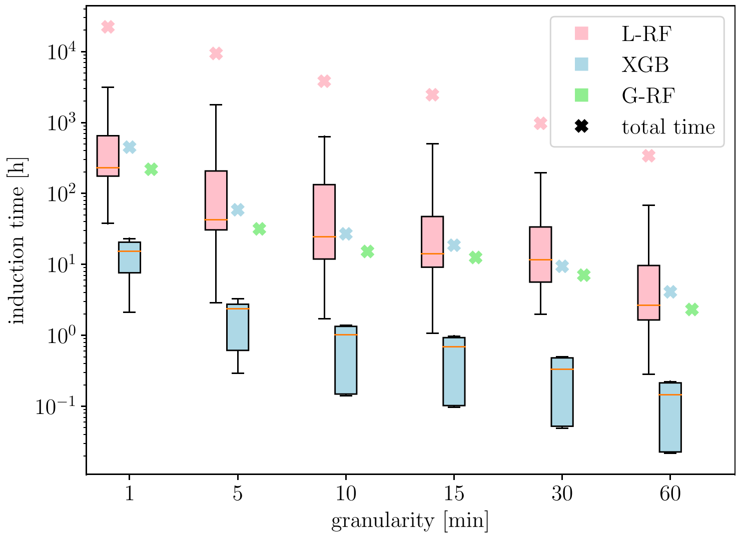


Figure 3.
Learning times for different algorithms and time granularity Δt. The crosses present the total learning time of the algorithm. Since L-RF and XGB build a model for each target (power line) separately, the distribution of learning times per power lines is additionally presented with the box plot. The per-target times add up to the total running time.

The ith component of vectors $\hat{\boldsymbol{y}}(\boldsymbol{x})$ and $\boldsymbol{y}(\boldsymbol{x})$, i.e., the predicted and true value for ith target, are denoted by $\hat{y}_i(\boldsymbol{x})$ and $y_i(\boldsymbol{x})$.

The quality of the predictions $\hat{\boldsymbol{y}}(\boldsymbol{x})$ is evaluated on a separate test set $\mathscr{D}_{\text{TEST}}$ not used for learning the models. It is measured in terms of the average root mean squared error $\overline{\text{RMSE}}$, defined via the root mean squared errors of each target variable, as follows:

$$\overline{\text{RMSE}}(\hat{\boldsymbol{y}}) = \sqrt{\frac{1}{T}\sum_{i=1}^{T}\text{RMSE}^2(\hat{y}_i)} \tag{3}$$

$$\text{RMSE}(\hat{y}_i) = \sqrt{\frac{1}{|\mathscr{D}_{\text{TEST}}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathscr{D}_{\text{TEST}}}(y_i(\boldsymbol{x}) - \hat{y}_i(\boldsymbol{x}))^2} \tag{4}$$

where $|\mathscr{D}_{\text{TEST}}|$ denotes the size of $\mathscr{D}_{\text{TEST}}$ and $T = 33$ is the number of target variables.

We also compare the machine learning methods in terms of their time efficiency (for constructing a predictive model). By doing so, we estimate the tradeoff between the predictive performance of the models and time needed for constructing them, in turn determining the optimal combination of time resolution in the data and machine learning method. The time efficiency refers to single-threaded runs of the algorithms, computed as follows. First, only an α portion of the ensemble is constructed where we measure the learning time t_α. Subsequently, the total learning time is estimated as $t = t_\alpha/\alpha$. Such estimations were necessary, since some methods do not allow for single threaded runs (as reported in the next section).

RESULTS

The goal of the experiments is to determine whether we can improve the efficiency of our approach for predicting TPC, while retaining good predictive performance. In particular, we evaluate tree different algorithms in terms of time efficiency (for learning a model) and predictive performance.

First, we report on the running times of the three algorithms given training data with different granularity as well as the impact on their predictive performance. Next, we discuss different alternatives for improving both the efficiency and the predictive performance of the algorithms. Finally, we discuss the quality of the learned predictive models using feature importance diagrams.

PERFORMANCE

We first focus on the learning times for different algorithms given different data granularity Δt. Figure 3 presents the time needed for each algorithm to construct a model for each target as well as the total time to complete the task.

As expected, in general, for all approaches the learning times decrease with the granularity Δt increasing. Nevertheless, the L-RF approach, that constructs an ensemble model for each of the 33 target variables, has the longest learning time of approximately 22 000 h ($\sim$ 2.5 years). In particular, learning an L-RF ensemble from the $\Delta t = 1$ dataset takes approximately 60 times longer than constructing an XGB ensemble which takes 450 h.

While the constituents of an XGB ensemble are considerably smaller than the ones of an RF, constructing a

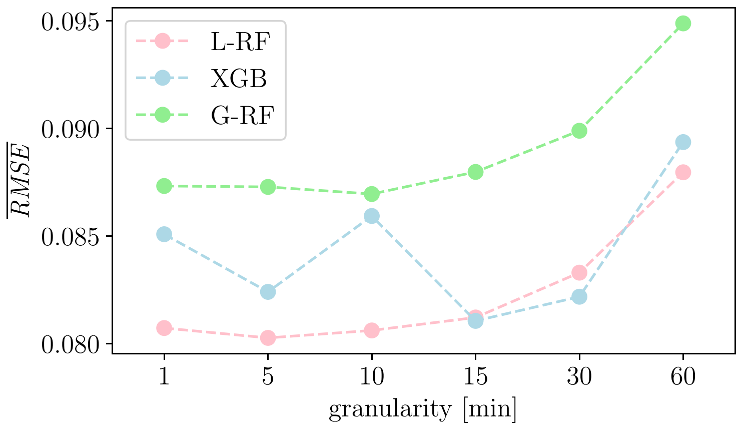


Figure 4.
Predictive performance of the three ensemble methods given data with different granularity.

XGB ensemble still takes approximately twice as much time as constructing a G-RF ensemble (220 h). Note that the constituents of both types of RF ensembles can be constructed in parallel, thus, additionally reducing the learning time for L-RF and G-RF by a factor of 100. In contrast, the most resource friendly is the $\Delta t = 60$ dataset, where the estimated learning times of L-RF, XGB, and G-RF are 330, 4.1, and 2.3 h, respectively.

Next, Figure 4 presents the impact of the data granularity on the predictive performance of the models constructed with the three different methods. Note that there is a tradeoff between learning-time efficiency and predictive performance. In particular, the ability of G-RF to construct predictive models in the shortest amount of time comes at a cost of decreased predictive performance compared to the other two methods. In this context, XGB performs best in two cases ($\Delta t \in \{15, 30\}$) while being substantially more efficient than L-RF which performs best in the remaining four cases ($\Delta t \in \{1, 5, 10, 60\}$).

The predictive error of both RF methods increases substantially with $\Delta t > 15$. In the case of XGB, however, the best predictive performance is obtained with the medium grained datasets $\Delta t \in \{15, 30\}$. Note that, the L-RF trained on $\Delta t = 1$ dataset (as used in [4]) still has some competitive advantage in terms of predictive performance. However, similar (or slightly better) performance can be obtained either by learning from the $\Delta t \in \{5, 10\}$ which is considerably more efficient, or by constructing XGB ensemble using the $\Delta t = 15$ dataset.

FURTHER OPTIMIZATION

So far, we reported on ensembles consisting of 200 constituents. In general, the size of the ensemble has different effects on the quality of the predictions in the case of RF ensembles and Boosting ensembles. In the former case, a general rule-of-thumb is that the predictive performance increases with the ensemble size, until it (effectively) saturates at some point. The reason for this is that every tree in RF is grown independently of the others. Thus, a RF ensemble may only be *unnecessarily* large.

In contrast, the boosting ensemble creation is different. Here, every additional tree focuses on minimizing the errors of the previously grown trees, therefore, such ensembles have a tendency to overfit to the training data. As a consequence, the ensemble size can have substantial effect on the predictive performance.

We conjecture that such artefacts are also present in the models evaluated thus far, and therefore we aim to further optimize the learning methods. To estimate the optimal number of trees in the ensembles, we take the $\Delta t = 60$ dataset and perform fivefold cross validation on $\mathscr{D}_{\text{TRAIN}}$. More specifically, we randomly divide $\mathscr{D}_{\text{TRAIN}}$ into five equally sized parts P_i, $1 \le i \le 5$. We train ensembles with different sizes (1, ..., 200) on every group of four parts, and estimate their performance on the remaining part not used for training. The average error of the five attempts (so called *cross-validated* error) is the final score of a given ensemble. We perform the same procedures for the RF (G-RF) and XGB ensembles.

The results in Figure 5 show that both types of methods can achieve good performance with considerably smaller ensembles. In particular, the RF method is able to achieve good performance very early (which does not drastically improve over time) due to accurate and deeper trees. On the other hand, as expected, XGB starts with very poor performance which considerably improves after approximately 50 iterations.

Given these evidence, we once again evaluate the predictive performance of the three methods on the test data, however, instead of constructing ensembles with 200 trees we construct them with only 50.

Table 4 confirms our conjecture: The ensemble size in the case of RF methods (L-RF and G-RF) has in general

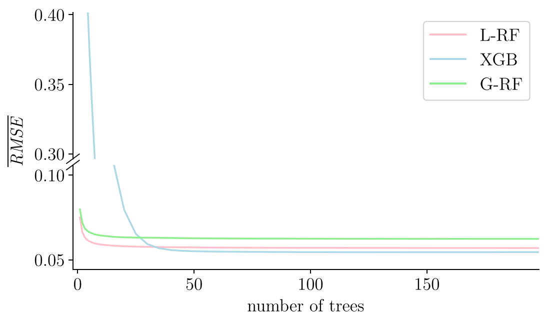


Figure 5.
Effect of the ensemble size on the performance of a Gradient Boosting ensemble (XGB), a local RF ensemble (L-RF), and a global RF ensemble (G-RF), evaluated with fivefold cross validation on the $\Delta t = 60$ dataset.

insignificant effect on the predictive performance. Note that the RF method is comprised of random trees, hence in some cases, like the G-RF constructed on the $\Delta t = 15$ dataset, small ensemble sizes hurt the predictive performance. Further analysis shows that in this case the performance stabilizes with at least 100 constituents to $\overline{\text{RMSE}} = 0.087$. In the case of XGB, the effect of the ensemble size is more prominent and constant. More specifically, the predictive performance of an XGB ensemble learned on a dataset with granularity $\Delta t = 15$ yields the best overall performance we obtained so far. Note that obtaining such performance is also considerably faster than the one reported in our previous study, i.e., L-RF learned on $\Delta t = 1$ dataset.

Nevertheless, in terms of time efficiency, all ensemble methods benefit from the reduction of the ensemble size. In particular, on average the learning time is reduced by factor of 4 in the case of RF ensembles and by a factor of almost 5 in the case of boosting. Figure 6 presents the results of the overall learning time and the predictive performance of all three methods with reduced ensemble size.

More specifically, here we aim at identifying the optimal choice of algorithm given both criteria of predictive performance and time efficiency. Therefore, the optimal solution should be as close as possible to the lower left

Table 4.

Predictive Performance ($\overline{\text{RMSE}}$) of the Three Ensemble Methods with 50 Constituents

Method \ Δt	1	5	10	15	30	60
L-RF	0.0825	0.0804	0.0812	0.0808	0.0844	0.0876
G-RF	0.0865	0.0869	0.0881	0.0989	0.0899	0.0954
XGB	0.0835	0.0796	0.0835	**0.0785**	0.0816	0.0889

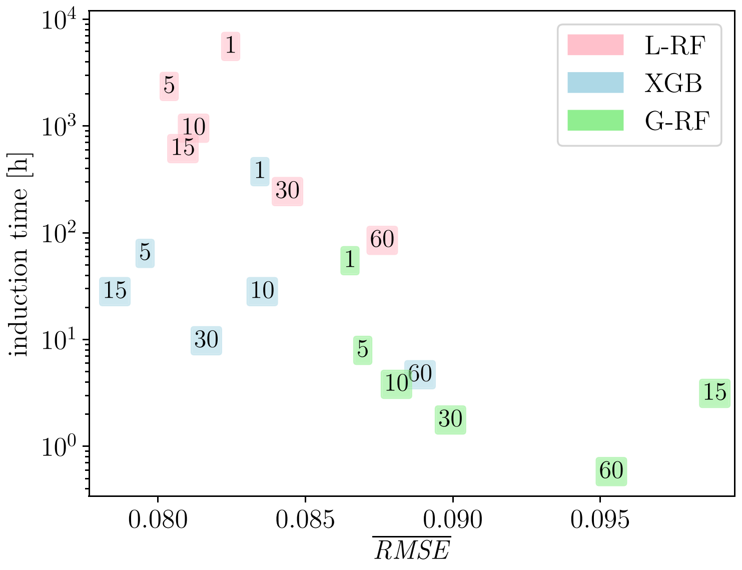


Figure 6.
Tradeoff between the learning time and the predictive performance ($\overline{\text{RMSE}}$) of the three ensembles with 50: Colors of the rectangles denote the induction algorithm, while the numbers in the rectangles determine the time granularity Δt.

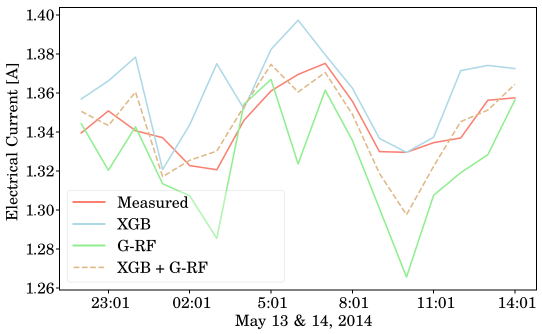


Figure 7.
Measured (red line) and predicted behavior on a sample time-period of 2 days for the 12th power line, obtained from G-RF and XGB ensembles as well as their combination.

corner of the graph. A point A in the graph is *dominated* by another point B in the graph if B is better than A in both criteria. For example, the performance of L-RF on $\Delta t = 10$ is dominated by the performance of XGB with both $\Delta t \in \{5, 15\}$. The nondominated points form a so-called *Pareto front*. In our case, the Pareto front consists of the two XGB points ($\Delta t \in \{15, 30\}$) and four G-RF points ($\Delta t \in \{5, 10, 30, 60\}$). All these points are considered optimal, unless we further specify our preferences over the criteria: If one aims at obtaining the fastest (but less accurate) solution one should consider G-RF ($\Delta t = 60$). On the other hand, XGB ($\Delta t = 15$) yields the most accurate (but less efficient) solution.

ENSEMBLE OF ENSEMBLES

The performance of an ensembles is a consequence of the performance and diversity of its constituents. Moreover, ensembles usually perform better when compared to each individual constituent [29]. Given that in this study, we consider different types of ensembles (*RFs* and XGB) with good predictive performance, to further improve the overall performance, we can also combine their outputs in an Ensemble of Ensembles (EoE). As a proof-of-principle, Figure 7 presents the predictions of XGB and G-RF during sample time-period of 2 days for the 12th power line. We can see that, while the XGB overestimates and G-RF underestimates the measured data (red line), their combination performs better.

Given the results from Figure 6, we construct two EoEs. The member ensembles are trained on $\Delta t = 15$ dataset, since all results so far point to a good tradeoff between efficiency and performance in this case. Moreover, both EoEs have one XGB member combined either with G-RF (both methods are efficient) or L-RF (both methods are accurate).

The results are summed up in Table 5. While the XGB& G-RF ensemble is very efficient to construct, its

Table 5.

Comparison of the Two Ensembles of Ensembles (EoE) in Terms of Predictive Performance ($\overline{\text{RMSE}}$) to the Performance of the Individual Ensembles

Ensembles	$\overline{\text{RMSE}}$
XGB& G-RF	0.0803
XGB&L-RF	**0.0778**
G-RF(100)	0.087
L-RF(50)	0.0808
XGB(50)	0.0785

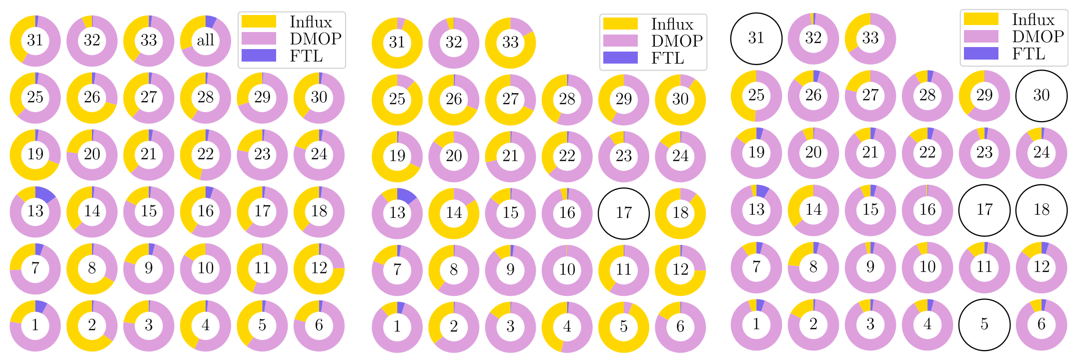
All models are learned on the $\Delta t = 15$ dataset.

performance is only better than one of the members, G-RF. On the other hand, the XGB&L-RF achieves the best performance overall of $\overline{\text{RMSE}} = 0.07779$. However, in practice, obtaining such an ensemble takes ten times more than learning only a XGB ensemble that has practically similar performance ($\overline{\text{RMSE}} = 0.07853$).

FEATURE IMPORTANCE

In our last set of experiments, we assess the importance of the features obtained from the three methods. Typically, different features have different influence on the target variables, which in turn affects the predictive performance of the constructed models. Figure 8 illustrates how different groups of features (energy influx, DMOP and FTL) influence the 33 power consumers. The feature importance diagrams were calculated using the $\Delta t = 15$ dataset. More specifically, in the case of L-RF and XGB [see Figure 8(b) and (c)], the proportions in the diagrams for each target were computed according to (2). On the other hand, in the case of G-RF [see Figure 8(a)] where the model predicts all targets simultaneously, (2) leads only to the global feature importance (*all* diagram). The per-target importance diagrams were computed from the local models, considering one target when computing the heuristic function. Note that, for some targets, the feature importance diagrams are blank since all ensemble constitutes in these cases are constant models (trees without internal nodes).

In the case of G-RF, the most important feature group overall is DMOP. In particular, in the majority of the power lines (27) their influence is at least 50%. The Energy influx features have a major influence on five power lines, while the FTL features have a considerable impact only on the 13th power line. Similarly, L-RF



(a) *G-RF* (b) *L-RF* (c) *XGB*

Figure 8.
Distribution of different feature groups in the feature rankings produced by (a) G-RF, (b) L-RF and (c) XGB ensembles, for the 33 power lines. In each of the individual diagrams, the presence of a given feature group type is proportional to the sum of Genie3 relevance over the features from the group. The empty diagrams denote ensembles constructed from constant models. The G-RFmethod allows for the additional *global* feature importance denoted as *all*.

finds the DMOP features as most important overall. However, as opposed to G-RF, here we can see more power lines which are almost exclusively influenced by energy influx features. These differences in the computed feature importance between G-RF and L-RF is a consequence of how the ensembles are constructed: While the former is able to capture the global phenomena across all power lines, the latter captures more detailed behavior that relates to each individual power line.

On the other hand, XGB mostly relates to the DMOP features. Compared to the other two methods, in this case their influence is considerably greater. In particular, the Energy Influx features have some significant importance ($> 20\%$) on only four power lines, while the FTL features do not contribute greatly. Additionally, XGB was not able to produce feature importance diagrams for 5 of the target variables.

CONCLUSION

In this paper, we propose a machine learning pipeline for predicting the power of the thermal subsystem on board the Mars Express spacecraft. More specifically, we propose novel solutions in the machine learning pipeline that focus on efficiently constructing predictive models of MEX's TPC, while still being able to maintain high predictive performance. More specifically, we employ state-of-the-art feature engineering approaches for transforming raw telemetry data, which in turn is used for constructing accurate predictive models with different machine learning methods. These solutions are the main contribution of this paper, since they considerably improve our competition-winning solution [4] in two directions: efficiency and accuracy.

The proposed improvements in the pipeline consider 1) preparing training data with different time granularity, as well as 2) employing different machine learning methods for constructing accurate predictive models. Regarding the former, we carefully transformed the raw telemetry data at different time resolutions ($\Delta t \in \{1, 5, 10, 15, 30, 60\}$ min) which resulted in significant reductions of the size of the datasets used in the learning phase. Regarding the latter, we considered different state-of-the-art local and global machine learning ensemble methods for multitarget regression. These methods include: local and global RFs of predictive clustering trees (L-RF and G-RF) as well as stochastic gradient boosted trees (XGB). We evaluated our proposed solutions on the task of predicting hourly values of the electric current through the 33 thermal power consumers on board MEX for one Martian year, given raw telemetry data of three preceding Martian years.

In terms of time efficiency, our empirical study shows that the time resolution of the data has a significant impact on both the construction time of the predictive models as well as on their accuracy. The former is an expected result, given that coarser granularity yields a reduced dataset and therefore shorter learning time. However, learning methods using coarser data usually yield less accurate models. The latter result though provides a significant insight into this problem: Given a dataset with moderate granularity ($\Delta t \in \{10, 15\}$), all three methods are able to obtain models with comparable (or better) predictive performance, in substantially shorter time as compared to models learned on dataset with finer granularity.

In terms of predictive performance, the local ensemble methods perform better than the global method. More specifically, in most cases, L-RF and XGB have comparable performance, with XGB being slightly better. While both methods perform better than G-RF, the difference in performance is neither substantial nor significant. Note that learning a global model also takes considerably less time than learning a local model. In the same context, our results show that, while the size of the ensembles has a significant effect on the learning time, it can also improve the predictive performance. In particular, we showed that with all methods we can obtain similar or better (XGB) predictive performance with smaller ensembles. Moreover, we also demonstrated that by further combining the predictions of the different ensembles into an ensemble-of-ensembles, we additionally improve the predictive performance and obtain premium accuracy of $\overline{\mathrm{RMSE}}_{XGB\&\mathrm{L-RF}} = 0.07779$.

Finally, our feature importance analysis indicates that, for this particular problem of predicting TPC, the DMOP have a significant role in the quality of the predictive models. Their importance is more prominent in the XGB ensembles models than in the *RF* ensembles, which also rely on the Energy Influx features when constructing a model.

There are several directions to extend the work presented in this paper. Considering the data, note that DMOP information is available only after a certain command is executed on the spacecraft and its effect measured subsequently. This means that using these data for predicting longer time horizons is not possible. Moreover, given the findings of this paper, omitting them from the learning process might have a severe consequence on the performance of the predictive models. Therefore, an immediate continuation of the work presented here is to further optimize the constructed features as well as investigate different approaches for engineering new (informative) features. Finally, while the proposed methodology focuses on the thermal subsystem of the MEX spacecraft, it can also be readily applied to the other subsystems. Moreover, it can also be extended to other spacecraft such as the XMM Newton [17], Integral [20], and ExoMars as well as rovers (such as Curiosity and ExoMars) exploring Mars.

ACKNOWLEDGEMENTS

We acknowledge the financial support of the Slovenian Research Agency (via the grants P2-0103, J4-7362, L2-7509, J2-9230 and the young researcher grants to MP and MB).

REFERENCES

[1] R. Orosei et al., “Radar evidence of subglacial liquid water on mars,” *Science*, vol. 361, pp. 490–493, 2018.

[2] A. Chicarro, P. Martin, and R. Trautner, “The Mars express mission: An overview,” *Mars Express, Sci. Payload*, vol. 1240, pp. 3–13, 2004.

[3] L. Lucas and R. Boumghar, “Machine learning for spacecraft operations support—The Mars express power challenge,” in *Proc. 6th Int. Conf. Space Mission Challenges Inf. Technol.*, 2017, pp. 82–87.

[4] M. Breskvar et al., “Predicting thermal power consumption of the Mars Express satellite with machine learning,” in *Proc. 6th Int. Conf. Space Mission Challenges Inf. Technol.*, 2017, pp. 88–93.

[5] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, “Tree ensembles for predicting structured outputs,” *Pattern Recognit.*, vol. 46, no. 3, pp. 817–833, 2013.

[6] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.

[7] J. H. Friedman, “Stochastic gradient boosting,” *Comput. Statist. Data Anal.*, vol. 38, no. 4, pp. 367–378, 2002.

[8] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, “A survey on multi-output regression,” *Data Mining Knowl. Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

[9] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, and I. Vlahavas, “Multi-target regression via input space expansion: Treating targets as inputs,” *Mach. Learn.*, vol. 104, no. 1, pp. 55–98, 2016.

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, (ser. Springer Series in Statistics). New York, NY, USA: Springer, 2013.

[11] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2005.

[12] G. H. Bakır, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, *Predicting Structured Data*, ser. Neural Information Processing. Cambridge, MA, USA: The MIT Press, 2007.

[13] A. McGovern and K. L. Wagstaff, “Machine learning in space: Extending our reach,” *Mach. Learn.*, vol. 84, no. 3, pp. 335–340, 2011.

[14] Z. Li, “Machine learning in spacecraft ground systems,” in *Proc. 6th Int. Conf. Space Mission Challenges Inf. Technol.*, 2017, pp. 76–81.

[15] T. Yairi, Y. Kawahara, R. Fujimaki, Y. Sato, and K. Machida, “Telemetry-mining: A machine learning approach to anomaly detection and fault diagnosis for space systems,” in *Proc. 2nd IEEE Int. Conf. Space Mission Challenges Inf. Technol.*, 2006, vol. 2006, pp. 476–483.

[16] M. Muñoz, Y. Yue, and R. Weber, “Telemetry anomaly detection system using machine learning to streamline mission operations,” in *Proc. 6th Int. Conf. Space Mission Challenges Inf. Technol.*, 2017, pp. 70–75.

[17] G. De Canio, T. Godard, R. Boumghar, and U. Weissmann, “Optimization of the battery usage during eclipses using a machine learning approach,” in *Proc. 15th Int. Conf. Space Oper.*, Marseille, France, 2018, https://doi.org/10.2514/6.2018-2607

[18] A. C. Hernández, C. Gómez, J. Crespo, and R. Barber, "Adding uncertainty to an object detection system for mobile robots," in *Proc. 6th Int. Conf. Space Mission Challenges Inf. Technol.*, 2017, pp. 7–12.

[19] A. Giusti et al., "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 661–667, Jul. 2016.

[20] T. J. Finn, R. Boumghar, J. Martinez, and A. Georgiadou, "Machine learning modeling methods for radiation belts profile predictions," in *Proc. 15th Int. Conf. Space Oper.*, Marseille, France, 2018, https://doi.org/10.2514/6.2018-2639

[21] R. Boumghar, L. Lucas, and A. Donati, "Machine learning in operations for the mars express orbiter," in *Proc. 15th Int. Conf. Space Oper.*, Marseille, France, 2018, https://doi.org/10.2514/6.2018-2551

[22] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[23] H. Blockeel, L. De Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 55–63.

[24] L. Breiman, J. Friedman, R. Olshen, and C. J. Stone, *Classification and Regression Trees*. London, U.K.; Chapman & Hall, 1984.

[25] H. Blockeel, "Top-down induction of first order logical decision trees," Ph.D. dissertation, Katholieke Universiteit Leuven, Leuven, Belgium, 1998.

[26] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts, "Inferring regulatory networks from expression data using tree-based methods," *PLOS ONE*, vol. 5, no. 9, pp. 1–10, 2010.

[27] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[28] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[29] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.

2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)

Quantifying the effects of gyroless flying of the Mars Express Spacecraft with machine learning

Matej Petković[1,2,✉], Luke Lucas[3], Dragi Kocev[1,2,4], Sašo Džeroski[1,2], Redouane Boumghar[5], Nikola Simidjievski[1,4,6]
e-mail: name.surname@ijs.si[1]
Jožef Stefan Institute, Slovenia[1]
Jožef Stefan International Postgraduate School, Slovenia[2]
e-mail: luke.lucas@lsespace.com[3]
Mars Express, Mission Planning & Spacecraft Operations, Germany[3]
Bias Variance Labs, Slovenia[4]
e-mail: redouane.boumghar@free.fr[5]
LibreSpace Foundation, Greece[5]
University of Cambridge, United Kingdom[6]

***Abstract*—The gyroscopes on-board the Mars Express (MEX) spacecraft, responsible for orientation and pointing actions, are slowly decaying. On 16th April 2018, a new software was deployed to MEX which reduces gyroscope usage by 90%. In this paper, we investigate the effect of gyroless flight on the power consumption of MEX's thermal subsystems. In particular, we train predictive models from telemetry data obtained before the event on 16th April 2018 and estimate their performance on time periods before and after the event. This offers the chance to evaluate machine learning models on new situations (e.g., gyroless scenarios) while these models have been trained on scenarios that were yet unencountered.**

The results show that the predictive performance of the models estimated in the gyroless period is lower when compared to the estimated performance before the event. Notwithstanding, the estimated performance in both scenarios is very good and the differences are not practically significant. Hence, the models can be utilized for accurate prediction of the thermal power consumption in practical sense.

Given MEX's new gyroless situation, which is likely to occur also on any other long lasting mission, we investigate the impact of the gyroscopes on the thermal power consumption and quantify the effects. Such modelling and analysis provides important insight into the spacecraft's new behavior, and still allows for planning optimization of MEX's operations, despite the radical change in the operational methodology.

***Keywords*-machine learning, Mars Express, gyroscopes, ensemble learning, predictive modeling, feature engineering**

I. Introduction

Mars Express (MEX), a spacecraft operated by the European Space Agency (ESA), has been orbiting Mars since the beginning of 2004. It has provided a wealth of scientific data about Mars such as evidence of the presence of water on and below the surface of the planet, three-dimensional renders of the surface as well as data about the chemical composition of the Martian atmosphere.

The scientific payload of MEX consists of a suite of seven instruments that provide global coverage of the planet's surface, subsurface and atmosphere. The instruments and on-board equipment must be kept within their operating temperature ranges, spanning from $+50°C$ to temperatures as low as $-180°C$. In order to maintain these predefined operating temperatures, the spacecraft is equipped with an autonomous thermal control system composed of 33 heater lines as well as coolers. Power is supplied by electricity generated by its solar arrays and stored in batteries to be used during the eclipses. The thermal control system, together with the platform units, consumes a significant amount of the total available electric power, leaving a fraction to be used for science operations.

After more than 16 years in space, MEX's components are slowly decaying, leading to reduced functionality and ever decreasing remaining lifetime, e.g., the batteries are now over 60% degraded, making accurate planning and use of the available power essential. In 2017, the gyroscopes on-board the spacecraft critical for maintaining its attitude, orientation and control were approaching end of life. Without gyroscopes the mission would end. To extend MEX's life, a novel functionality was developed; new software reduced gyroscope usage by 90% and enabled gyroless flight. The new software was deployed on 16th April 2018. A drastic effect on the thermal control subsystem was immediately evident. Every electronic unit consumes power and emits heat, when in operation, including the gyroscopes. Consequently, once the gyroscopes were switched off the thermal subsystem had to compensate for the lack of the heat source the gyroscopes had been.

In addition, gyroless flight has also required new spacecraft orientations, pointing the top face of MEX to the Sun, unusually illuminating and warming it, adding further unknowns to the thermal model. As thermal regulation on MEX is a complex and interwoven system, a change in any one area of the spacecraft may have unforeseen and dangerous consequences in other areas. Since going gyroless, the thermal subsystem's performance has altered markedly, previously established patterns and behaviors are no longer present.

This lifesaving move to gyroless flight resulted in a conundrum. A new gyroless thermal model was required yet there was not enough data to create an empirical model, which requires a good understanding of the changed thermal behavior. At the same time, an accurate model was essential to plan observations without risking dangerously discharging the batteries during eclipses. In contrast, an overcautious model would maintain MEX safety but at the expense of missed science opportunities.

In this paper, we thus investigate the effect of operating MEX without its gyroscopes on its thermal power subsystem. We first learn predictive models from telemetry data obtained before the "event" of 16th April 2018 and evaluate their performance on two periods spanning before and after the event. Next, to assess the impact of gyroless flight on the thermal subsystems, we analyze the feature importance scores obtained from models learned on data gathered before and after the event. Finally we show that despite radical changes in MEX operations, the models provide insight to thermal behavior and an accurate predictive tool.

The remainder of the paper is organized as follows. Section II presents the data pre-processing procedure and the machine learning methods used in this study. Section III elaborates the design of the experiments, the results of which are reported and discussed in Section IV. Finally, Section V concludes the paper.

II. Materials and Methods

A. Data

We use data provided by the European Space Operations Centre at ESA, which consists of raw telemetry data (context data) and measurements of the electric current of 33 thermal power lines (observation data), for one Martian year of MEX operations. We refer to these data as the training data $\mathscr{D}_{\text{TRAIN}}$ as shown in Fig. 1. For estimating the performance of the learned models, the context part of the data was used for generating the predicted values, that in turn where evaluated using the real measured observation data.

Since in this paper we investigate the effect of the gyroscopes on the behavior of the spacecraft, we generate two test data sets with equal length of approximately $1/3$ of a Martian year. The first test data set, referred to as $\mathscr{D}_{\text{G}}$ (cf. Fig. 1), spans in the period when the gyroscopes were fully operational, i.e., between 17th August 2017 and 16th April 2018. The second data set, referred to as $\mathscr{D}_{\text{GLESS}}$ (cf. Fig. 1), spans in the gyroless period, i.e., between 17th May 2018 and 15th December 2018.

The *observation data* consists of the electric current measurements of the 33 power consumers, recorded once or twice per minute. The *context data* consists of five components:

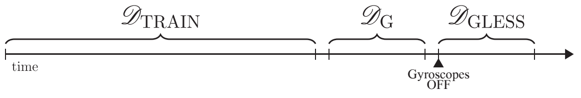


Figure 1: The three data sets that were extracted from the data, together with the two 1-month pauses between them. The lengths of the time intervals are proportional to the actual time spans of the data sets.

SAA - (Solar Aspect Angles) data contain the angles between the Sun–MEX line and the axes MEX's coordinate system.
FTL - (Flight dynamics TimeLine events) data contain the pointing and action commands that impact the position of MEX, such as pointing the spacecraft towards Earth or Mars.
EVT - (Miscellaneous Events) data contain time intervals during which MEX was in Mars's shadow or records when MEX is at apoapsis of its elliptical orbit. Note that, some of these events can also be present in the FTL.
LTDATA - (Long Term Data) contain the distances between different bodies i.e., MEX, Earth, Mars, the Sun as well as the solar constant.

All raw data entries are time-stamped indicating when the entry was logged. The time span between the two consecutive entries varies from less than a minute (SAA) to several hours (LTDATA). For a detailed description of the task and the data, we refer the reader to [1].

To construct an appropriate data set for a learning algorithm, we pre-process the raw data by first applying an appropriate time resolution Δt (time interval between two consecutive samples). In particular, we divide the time span $[t_{\text{FIRST}}, t_{\text{LAST}})$ into subintervals $[t_i, t_{i+1})$, where $t_{i+1} = t_i + \Delta t$, where t_{FIRST} is the first time-stamp in the $\mathscr{D}_{\text{TRAIN}}$, and t_{LAST} is the last time-stamp in the $\mathscr{D}_{\text{GLESS}}$. The value of a given feature for a particular time interval is obtained by aggregating measurements from the time interval at hand that correspond to one or more components from the raw telemetry data. Following the findings from [2], in this paper we set $\Delta t = 15min$.

Using these data, we next focus on engineering new (more informative) features from different parts of the context data that may yield better predictive performance. More specifically, following [2], we constructed *Energy influx features* that describe the amount of solar energy falling on a given face of the spacecraft in a given time interval $[t_i, t_{i+1})$. The solar energy falling upon a side of the spacecraft cuboid directly influences the amount of energy used by the thermal lines that maintain the temperature in that part of the spacecraft. The solar energy collected by the solar panels influence the amount of available energy for science operations and for recharging the spacecraft's batteries.

The amount of energy received by a given surface is proportional to the product of the effective area A_{eff} of the surface and the solar constant c. If the area A is given, we compute A_{eff} as $A_{\text{eff}} = A \max\{\cos\alpha, 0\}$, where α is the angle between the Sun–MEX line and the outer normal $\vec{n}$ to the surface. Without any loss of generality, we assume $A = 1$ for all surfaces, as the machine learning methods that we use, are invariant to monotonic transformations of features. The values of α for each of the seven surfaces were computed directly from the SAA data, while c was given in LTDATA. In addition to the effective area and the solar constant, (pen)umbras have a considerable impact on the energy influx, as MEX is plunged into the darkness of Mars' shadow. We define the amount of energy E_S^i that is received by the surface S at time interval $[t_i, t_{i+1})$ as

$$E_S^i = \int_{t_i}^{t_{i+1}} A_{\text{eff}}(t)\, c(t)\, U(t)\, \mathrm{d}t,$$

where U is the *umbra coefficient*, an approximation of the proportion of Sun visible from the spacecraft. U takes the value $U(t) = 0$ if the spacecraft is in an umbra, $U(t) = 0.5$ if the spacecraft is in a penumbra, and $U(t) = 1$ otherwise. Instead of calculating exact integrals for E_S^i, we approximate the values using the trapezoid-rule.

Furthermore, the thermal state of the spacecraft depends not only on the current energy influx, but also on the energy influx in the past. To capture this, we construct *Historical energy influx features* for each of surfaces. A given historical feature for surface S at time t_i is computed as a sum of energy influx during given historical time-frame:

$$H_S^i = \sum_{j=1}^{H} E_S^{i-(j-1)},$$

where H is the number of time-intervals included in the historical feature. To account for different impacts of the historical energy influx we construct historical features with different time-frames, for different values of $H \in \{1, 2, 4, 6, 8\}$

Finally, to construct the final data sets, we add features extracted from both the FTL and the EVT data. The FTL data contain logs of pointing events and their time ranges. For each pointing event in time interval $[t_i, t_{i+1})$, a feature has value that equals the proportion of the time in $[t_i, t_{i+1})$, during which the event is in progress. The EVT data contain logs of events and their duration referring to the relative position of MEX to Mars. From these we construct features that encode this information in terms of "time since last occurrence" of a specific event.

The data can be accessed at http://spacelab.ijs.si (after the completion of the review process).

B. Methods

To learn the predictive models and analyze the data we employ predictive clustering methods able to address the task of multi-target regression (MTR). These methods can learn both local and global models for multi-target regression (Fig. 2). The former models are learned for each target variable separately. In contrast, the latter models are able to predict several numeric target variables simultaneously [3]. In this paper, we consider Random Forest of Predictive Clustering Trees (RF-PCTs) - an ensemble method with state-of-the-art predictive performance [2, 4], which can construct single model for all target variables, therefore substantially reducing the computational time needed for obtaining a solution. This ability is inherited from the base models in the ensemble, Predictive Clustering Trees (PCTs), a generalization of decision trees used for a variety of structured output learning tasks. Besides its performance, RF-PCT is able to produce feature importance scores, i.e., ranking of the features w.r.t. their importance for each of the target variables. In the remainder, we describe the important details of the employed methods.

Random Forest of PCTs[1] [5] is an ensemble method [6] that learns a set of PCT predictive models and combines their predictions. The base models are learned from bootstrap samples of the training set, where for each tree at each tree node a random feature subset (with a user defined size) is considered for selecting the best split. Such approach allows for constructing a set of diverse predictive models that can differ both in size and performance. In particular, it takes three inputs: (1) training data $\mathscr{D}_{\text{TRAIN}}$, as well as two hyper-parameters denoting (2) the number of trees M in the ensemble and (3) the size of the feature subset considered at each node split f. Each PCT in the ensemble is learned with greedy recursive top-down induction algorithm on a bootstrap sample of the input data set (line 3 of the Alg. 1).

The PCT algorithm (line 7 of Alg. 1) starts at the root node of the tree by selecting a split test from a set of candidate tests that are generated by a random feature subset $\mathcal{F}$ of size f. The best test is greedily chosen by a heuristic function that typically measures the impurity of the target values of the examples in the subsets E_i of the set E that this test results in. The goal of the heuristic function is to guide the algorithm towards small trees with good predictive performance (Algorithm 2). The number of features that are retained is given as a function of the total number of descriptive features F_{ALL}, e.g., $f = \lceil\sqrt{F_{\text{ALL}}}\rceil$, $f = \lceil\log_2(F_{\text{ALL}})\rceil$, $f = F_{\text{ALL}}/4$, etc.

Note that, in the case of the global MTR task, the heuristic function is the average variance of the T targets, while the predictions $\hat{\boldsymbol{y}}$ of each tree are defined as the averages of the

[1]The RF of PCT framework is implemented in the CLUS system developed at the Jožef Stefan Institute, and available at http://source.ijs.si/ktclus/clus-public

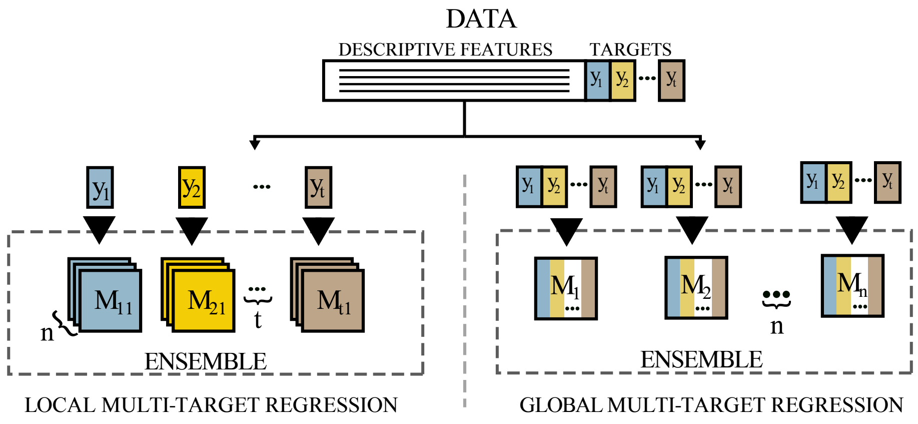


Figure 2: An overview of the local and global models for multi-target regression. The former models M are learned for each of the t target variables separately. The latter model is learned for all t target variables simultaneously. For constructing ensembles, n models are learned on different samples of the training data. Subsequently, the predictions of the ensemble constituents are combined to obtain the final ensemble prediction.

Algorithm 1 Random Forest of PCTs ($\mathscr{D}_{\text{TRAIN}}$, M, f)

```
1: $\mathcal{RF} = \emptyset$
2: **for** $m = 1, 2, \ldots, \mathrm{M}$ **do**
3:     $E = bootstrapSample(\mathscr{D}_{\text{TRAIN}})$
4:     $\mathcal{T} = inducePCT(E, f)$
5:     add $\mathcal{T}$ to $\mathcal{RF}$
6: **return** $\mathcal{RF}$

7: **where** $inducePCT(E, f)$
8:     $\mathcal{F}$ = random sample of $f$ features
9:     $(t^*, \mathcal{P}^*) = findBestTest(E, \mathcal{F})$
10:    **if** $t^* \neq none$ **then**
11:        **for each** $E_i \in \mathcal{P}^*$ **do**
12:            $subtree_i = inducePCT(E_i, f)$
13:        **return** $\mathscr{N}(t^*, \bigcup_i \{subtree_i\})$        // internal node
14:    **else**
15:        **return** $\mathscr{L}(\hat{\boldsymbol{y}}(E))$        // leaf node
```

target values, namely

$$\begin{aligned} impu(E) &= \frac{1}{T}\sum_{j=1}^{T} Var_j(E) \\ \hat{\boldsymbol{y}}(E) &= [\overline{y}_1(E), \ldots, \overline{y}_T(E)] \end{aligned}$$

where $Var_j(E)$ denotes the variance of the target y_j in the subset $E \subseteq \mathscr{D}_{\text{TRAIN}}$ and $\overline{y}_j(E)$ denotes its average. Finally, the RF-PCTs algorithm outputs a set of PCTs, whose predictions are combined (averaged per value) to obtain the final ensemble prediction.

Genie3 feature ranking score. Random forests can measure how much each feature contributed to the quality of the predictive model. For this purpose, we use the Genie3 algorithm [7, 8]. Genie3 considers the amount of variance reduction when a subset $E \subseteq \mathscr{D}_{\text{TRAIN}}$ is split given a particular feature. More specifically, if a relevant feature x_i is part of a test $x_i < \vartheta$, then the heuristic score h^* (reduction of the variance) of this split is high. Greater emphasis is put on the features higher in the tree, i.e., on the splits where $|E|$ is larger. The Genie3 importance I_{GENIE3} of the feature x_i is defined as

$$I_{\text{GENIE3}}(x_i) = \frac{1}{|\mathcal{RF}|}\sum_{\mathcal{T}\in\mathcal{RF}}\ \sum_{\mathscr{N}\in\mathcal{T}(x_i)} |E(\mathscr{N})|\ h^*(\mathscr{N}),$$

where $\mathcal{RF}$ is a forest of trees $\mathcal{T}$, $E(\mathscr{N})$ is the set of examples that come to the node $\mathscr{N}$, and $h^*(\mathscr{N})$ is the value of the variance reduction function described in Algorithm 2.

III. Experimental Setup

In this paper, we investigate the effect of gyroless flying of MEX on its thermal power subsystems. Our experimental questions are formulated as follows:

Algorithm 2 $findBestTest(E, \mathcal{F})$

```
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: **for each** $x_i \in \mathcal{F}$ **do**
3:     **for each** candidate test $t$ **do**
4:         $\mathcal{P}$ = partition induced by $t$ on $E$
5:         $h = |E| impu(E) - \sum_{E_i \in \mathcal{P}} |E_i| impu(E_i)$
6:         **if** $h > h^*$ **then**
7:             $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
8: **return** $(t^*, h^*, \mathcal{P}^*)$
```

1) How much do the predictions of the MEX power consumption change when one ceases to use the gyroscopes?
2) Are our models robust enough to offer a good predictive performance in both flight scenarios?
3) Which parts of the spacecraft are mostly affected by the changes?

A. Evaluation Procedure

To properly address the afore-mentioned experimental questions, we carry out two lines of experiments. The first line of experiments considers analyzing the predictive performance of models learned on the training set $\mathscr{D}_{\text{TRAIN}}$ and evaluated on both test sets $\mathscr{D}_{\text{G}}$ (when the gyroscopes were still in full use) and $\mathscr{D}_{\text{GLESS}}$ (when the gyroscopes turned off). More specifically, we first investigate whether there is a change between the two modes of flight. Therefore, we learn and evaluate the performance of global and local PCTs. Next, using the same setup, we construct local and global random forests and evaluate their performance on both test data sets. Note that, to prevent data leaks, we separate the three data sets with two 1-month gaps as shown in Fig. 1.

The quality of the predictions $\hat{\boldsymbol{y}}(\boldsymbol{x})$ is measured in terms of the average root mean squared error $\overline{RMSE}$, defined as

$$\overline{RMSE}(\hat{\boldsymbol{y}}) = \sqrt{\frac{1}{T}\sum_{i=1}^{T} RMSE^2(\hat{y}_i)}, \quad \text{where}$$

$$RMSE(\hat{y}_i) = \sqrt{\frac{1}{|\mathscr{D}_{\text{TEST}}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathscr{D}_{\text{TEST}}} (y_i(\boldsymbol{x}) - \hat{y}_i(\boldsymbol{x}))^2}$$

If the actual gyroless flying dynamics is different from the standard flying dynamics, then the model predictions should be less accurate on $\mathscr{D}_{\text{GLESS}}$ as compared to $\mathscr{D}_{\text{G}}$. This gives us an indirect comparison of the flying scenarios.

In the second line of experiments, we analyze the influence of the individual features in the two flying scenarios. In particular, we directly compare the feature importance scores obtained from two random-forest ensembles learned on the $\mathscr{D}_{\text{G}}$ and $\mathscr{D}_{\text{GLESS}}$ separately. For each ensemble we compute the Genie3 feature ranking and compare their rankings in terms of Fuzzy Jaccard Index (FUJI). FUJI measures the similarity between the different sets of top ranked features, where lower values indicate large discrepancies between the two rankings.

FUJI is defined as follows. Given two rankings $r_i = (x^i_{(1)}, \ldots, x^i_{(F_{\text{ALL}})})$, $1 \le i \le 2$, where $x^i_{(j)}$ denotes the j-th top-ranked feature in ranking r_i, accompanied by the feature importance scores $f^i_j = I_{\text{GENIE3}}(x^i_{(j)})$: We define the sets $F^i_j = \{x^i_{(1)}, \ldots, x^i_{(j)}\}$ as the sets of top-ranked features of ranking r_i, for $1 \le i \le 2$ and $1 \le j \le F_{\text{ALL}}$.

The membership function μ of the feature $x^i_{(k)}$ for the set F^i_j is defined as:

$$\mu(F^i_j, x^i_{(k)}) = \begin{cases} 1 & ; x^i_{(k)} \in F^i_j \\ f^i_k / f^i_j & ; \text{otherwise} \end{cases}.$$

Finally, $\text{FUJI}(F^1_j, F^2_j)$ is defined as

$$\text{FUJI}(F^1_j, F^2_j) = \frac{\sum_{x\in F^1_j \bigcup F^2_j} \min\{\mu(F^1_j, x), \mu(F^2_j, x)\}}{\sum_{x\in F^1_j \bigcup F^2_j} \max\{\mu(F^1_j, x), \mu(F^2_j, x)\}}$$

We plot the curves that consist of points $(j,\ \text{FUJI}(F^1_j, F^2_j))$, $1 \le j \le F_{\text{ALL}}$, as shown in Fig. 4b.

B. Parameter instantiation

In this paper we follow a similar experimental setup as in [2] and set the hyperparameters as follows: When learning a single PCT, the minimal number of samples in a leaf was set to 33 and M5 post-pruning technique was used. For constructing a random forest of PCTs, we learn 200 trees and consider 25% of the number of features at each splitting node. We keep the minimal number of samples in a leaf setting to 33 and we do not use post-pruning.

IV. Results and Discussion

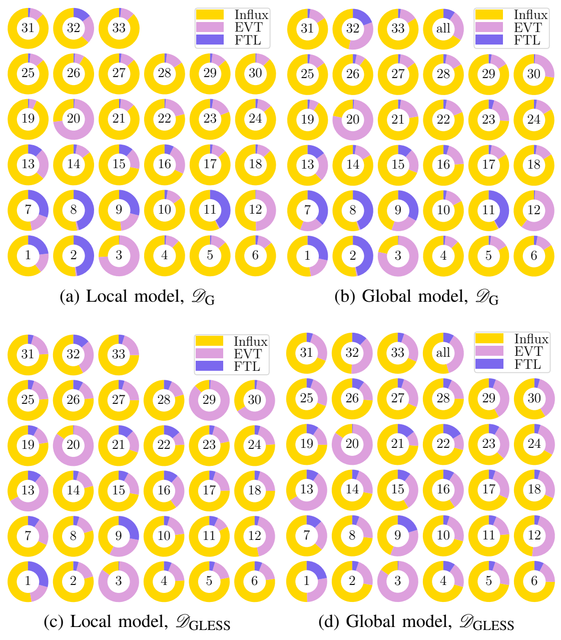
In the first line of experiments, we evaluate the predictive performance of the models learned on $\mathscr{D}_{\text{TRAIN}}$. Tab. I presents the performance of the global and local PCTs, where we observe two important results. First, both global and local PCTs exhibit degraded performance (by cca 30%) when evaluated on $\mathscr{D}_{\text{GLESS}}$ as compared to their performance on $\mathscr{D}_{\text{G}}$. This confirms our first hypothesis: There is a difference between the standard and the gyroless-flying dynamic.

Second, when evaluated on both data sets, the global model gives more accurate predictions than their local counterpart. These results indicate that the local models tend to overfit to the training data since most of the local models (30 out of 33) are larger in size than the global model that models all 33 targets simultaneously.

Table I: Predictive performance of global and local single PCTs evaluated on $\mathscr{D}_{\text{G}}$ and $\mathscr{D}_{\text{GLESS}}$.

test set	local model	global model
$\mathscr{D}_{\text{G}}$	$2.98 \cdot 10^{-2}$	$2.83 \cdot 10^{-2}$
$\mathscr{D}_{\text{GLESS}}$	$3.75 \cdot 10^{-2}$	$3.62 \cdot 10^{-2}$

However, in general, single models are prone to overfitting and have a limited ability to accurately predict future phenomena. In the next line of experiments, we address the second experimental question. Tab. II presents the results of random forest ensembles with 200 PCTs base models each. As expected, the results show that the ensembles perform better and are more robust than their single-model



(a) Local model, $\mathscr{D}_{\mathrm{G}}$ (b) Global model, $\mathscr{D}_{\mathrm{G}}$

(c) Local model, $\mathscr{D}_{\mathrm{GLESS}}$ (d) Global model, $\mathscr{D}_{\mathrm{GLESS}}$

Figure 3: Distribution of different feature groups in the feature rankings, for the 33 power lines. The rankings were produced by a) the local and b) global ensembles, learned on $\mathscr{D}_{\mathrm{G}}$, and c) the local and d) the global model, learned on $\mathscr{D}_{\mathrm{GLESS}}$. In each of the individual diagrams, the presence of a given feature group type is proportional to the sum of Genie3 relevance over the features from the group. The global models allow for the additional *global* feature importance denoted as *all*.

counterparts. In terms of the performance of the ensembles on the standard and gyroless periods, the ensembles still exhibit better performance (by cca. 30%) when tested on $\mathscr{D}_{\mathrm{G}}$ as compared to $\mathscr{D}_{\mathrm{GLESS}}$. This indicates that more robust models are also able to identify the difference between the two flying modes. However, the difference in performance is not practically significant and therefore such models can be still utilized for accurate prediction of the thermal power consumption regardless of the time period used in the learning phase.

Table II: Predictive performance of global and local Random Forest ensembles of 200 trees evaluated on $\mathscr{D}_{\mathrm{G}}$ and $\mathscr{D}_{\mathrm{GLESS}}$.

test set	local model	global model
$\mathscr{D}_{\mathrm{G}}$	$2.51 \cdot 10^{-2}$	$2.50 \cdot 10^{-2}$
$\mathscr{D}_{\mathrm{GLESS}}$	$3.27 \cdot 10^{-2}$	$3.21 \cdot 10^{-2}$

To address our last experimental question in terms of assessing where the differences occur, we evaluate the feature rankings obtained from two different random forest ensembles. In particular, we first report on these analyses on a ”macro” scale where we depict the resulting feature rankings for all 33 targets obtained from local and global random forests learned on the $\mathscr{D}_{\mathrm{G}}$ and $\mathscr{D}_{\mathrm{GLESS}}$. We then inspect in more detail and compare the feature rankings of two local ensemble models that model the behavior of two particular power lines *NPWD2531* and *NPWD2851* which are located close to the gyroscopes. We refer to this analysis as a ”micro” scale analysis.

Macro Scale. We start our analysis of feature rankings with the comparison of influence of three groups of features: Influx, EVT and FTL. The group of influx features contains 42 energy influx features, including the historical influx features. The second group, EVT, contains 24 features that base only on the EVT data, while the third group contains 17 features that were constructed from the FTL data. The diagrams that present the amount of influence of the three groups on given target, are shown in Fig. 3.

The results show that, while there is no significant difference between local and global ensemble models, there are some obvious differences between the feature rankings obtained from the different periods. In particular, in the $\mathscr{D}_{\mathrm{GLESS}}$ period, the Influx features have a smaller influence on some targets as compared to their influence in the $\mathscr{D}_{\mathrm{G}}$ period. Moreover, FTL features have less influence on the targets y_2, y_8 and y_{11}, whereas EVT features have more influence on the targets y_{29} and y_{30}. The latter is explained by the fact that (pen)umbras were 2.6-times more frequent in the time span of $\mathscr{D}_{\mathrm{GLESS}}$, making some features from the EVT group more important.

This difference can be further analyzed on Fig. 4, that depicts the difference of the rankings obtained from the two global ensembles learned on $\mathscr{D}_{\mathrm{G}}$ and $\mathscr{D}_{\mathrm{GLESS}}$ respectively. In particular, Fig. 4 a depicts the feature importance scores of all features obtained from the two ensembles. Note that, they are sorted in a descending order with respect to their score on $\mathscr{D}_{\mathrm{G}}$. It can be seen that the most substantial difference between the rankings occurs at the beginning, i.e., the top most relevant features. This result is also confirmed by the FUJI plot, depicted in Fig. 4 b. It shows that the biggest discrepancy between the feature rankings is in the top 20 features. This means that the global random forest method mostly relates to the Influx features when learning a model on $\mathscr{D}_{\mathrm{G}}$ period, while in the $\mathscr{D}_{\mathrm{GLESS}}$ period the FTL and EVT features are more important. Note, however, that this difference besides being influenced by the abundance of the gyroscopes, is also likely to be influenced by the larger total amount of penumbras in $\mathscr{D}_{\mathrm{GLESS}}$ as compared to $\mathscr{D}_{\mathrm{G}}$.

Micro Scale. In our last set of experiments, we report on the feature importance analysis of the two thermal power lines *NPWD2531* (y_{11}) and *NPWD2851* (y_{28}) located close to the gyroscopes. Fig. 5 ab depicts the results for *NPWD2531*, while Fig. 5 cd for *NPWD2851*.

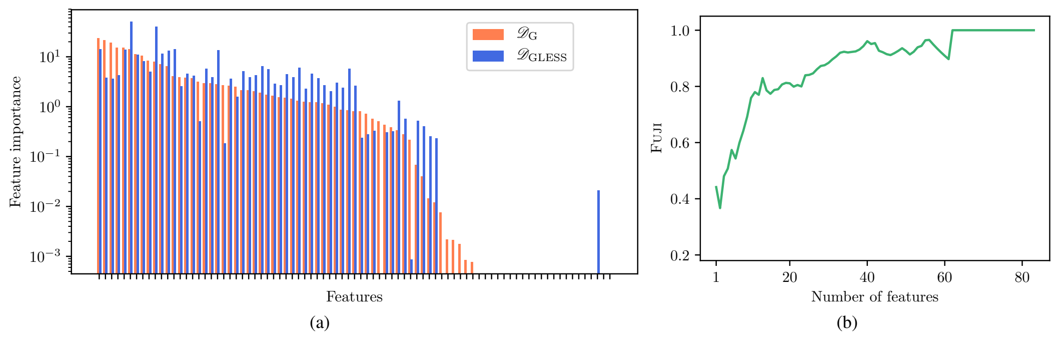


Figure 4: Comparison of the rankings obtained from global ensemble model on the datasets $\mathscr{D}_G$ and $\mathscr{D}_{GLESS}$. Figure a) plots all the feature importance scores sorted in descending order with respect to their score on $\mathscr{D}_G$. Figure b) plots the FUJI values computed from the two rankings.

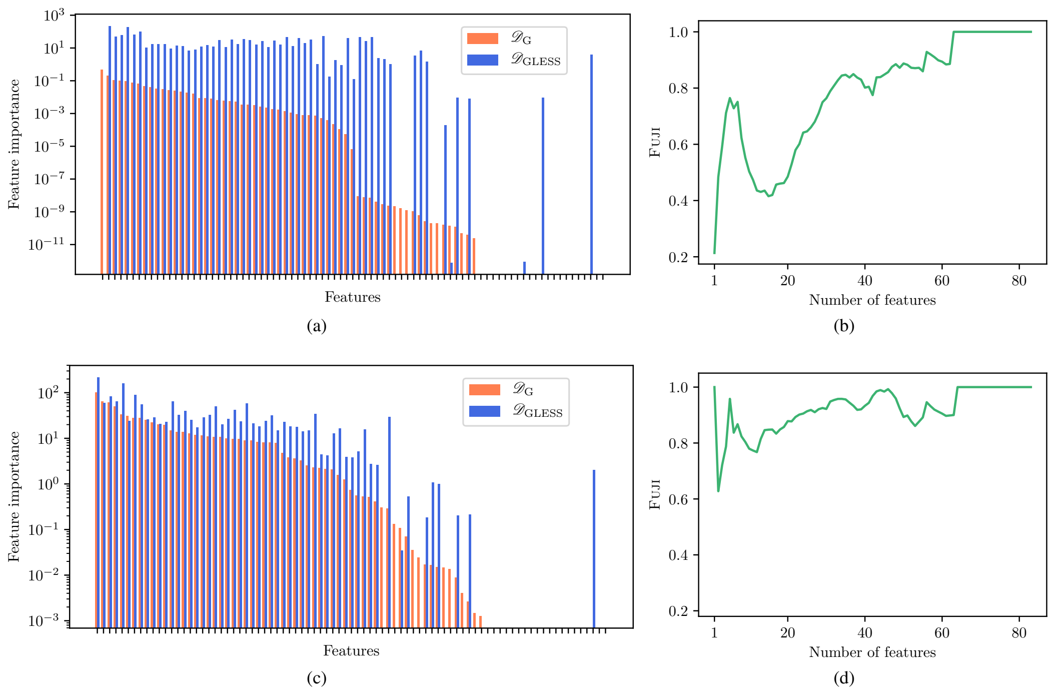


Figure 5: Comparison of the rankings, computed with the local model on the datasets $\mathscr{D}_G$ and $\mathscr{D}_{GLESS}$, for the power line *NPWD2531* (first row) and *NPWD2851* (second row). Figures a) and c) plot feature importances of the features, where features are sorted by their importance on $\mathscr{D}_G$. Figures b) and d) give the values of FUJI, computed from these two rankings.

We first analyze the power line *NPWD2531* and see that feature importance score from $\mathscr{D}_G$ are orders of magnitude smaller than $\mathscr{D}_{GLESS}$. The reason for this is, that the heater was turned off during the time span of $\mathscr{D}_G$, and it was mostly used in the time span of $\mathscr{D}_{GLESS}$. Next, besides having different scales of the scores, as shown in Fig. 5 b the rankings of the features are also different. The FUJI value at the beginning is very low, however, since both rankings have (different) Influx at the top, the similarity starts to increase. After that, the similarity decreases again since the model learned on $\mathscr{D}_G$ continues to relate to Influx feature while the model learned on $\mathscr{D}_{\text{GLESS}}$ finds the EVT features more valuable.

On the other hand, the situation with the rankings for *NPWD2851* is a bit different. Here, Fig. 5c already reveals that both rankings are quite similar having similar top ranked features (Influx). This is confirmed by the high FUJI values in Fig. 5d. The explanation for this is that the *NPWD2851* power line was used throughout the whole time span of $\mathscr{D}_{\text{G}}$ and also in some periods of $\mathscr{D}_{\text{GLESS}}$, except for the period when was changed with *NPWD2531*.

V. CONCLUSIONS

As MEX ages, operational methods continue to change and the development of a reliable model of the thermal subsystems becomes a repeating issue. On 16th April 2018 a new software, that reduces the usage of the decaying gyroscopes, was deployed on MEX. While the effect on the thermal control subsystem was immediately evident, the lack of data made manual creation of accurate empirical model virtually impossible.

In this paper, we employ state-of-the-art machine learning methods to investigate the effects of gyroless flight on the power consumption of MEX's thermal subsystems. Using these methods, able to adress both local and global multi-target regression tasks, we first aim to identify and quantify the difference between the two modes of operating MEX. Next, we employ these machine learning methods that address the data scarcity issues and are able to produce robust and accurate predictive models.

The results show that, while the employed machine learning methods perform very good in terms of predictive performance. The difference in performance when evaluated on periods with different flight modes has no practical significance. More specifically, both local and global ensemble models show better performance when evaluated on the period with operational gyroscopes. When evaluated on the gyroless period, the models exhibit slightly worse performance. However, given the limited availability of training data such performance deterioration, from an engineering perspective, makes no substantial difference to the total power predictions. Moreover, in terms of comparing the predictive performance of the local vs. global models, our experimental study shows that the global model perform better than their local counterpart. And while this difference in performance is not substantial, note that the global ensemble models are significantly more efficient to construct.

In addition to predictive power, the ensembles allow also for the interpretation of the results via feature rankings. The feature rankings detect that the conditions that MEX was operating in, differ between $\mathscr{D}_{\text{G}}$ and $\mathscr{D}_{\text{GLESS}}$. In particular, we show that i) turning the gyroscopes off influences some of the power lines, and ii) there are also other factors of influence, such as the length of (pen)umbras. The former can be particularly well observed in the case of the power line *NPWD2531* that is located right next to the gyroscopes, and was turned off while the gyroscopes were fully working, since it was more efficient to use power line *NPWD2851* instead. However, in the gyroless scenario, it is more efficient to use the power line *NPWD2531*, and the change is clearly reflected in the feature rankings.

There are several directions to continue the work presented in this paper. First, note that the FTL data is available immediately after a certain operation has been executed. This means that using such data in predictive scenarios for longer time horizons is not possible. On the other hand, omitting them from the learning process might have a severe consequence on the performance of the predictive models. Therefore an immediate continuation of the work presented in this paper is to investigate different approaches for engineering new features. Such features, besides being used for learning accurate predictive models, will also be used for investigating more complex relations present in the data that influence the behaviour of the spacecraft. Finally, the methods presented in this paper can be extended to other subsystems of MEX and other spacecraft.

The machine learning model's robustness and accuracy despite changing operational contexts, especially in comparison to the empirical models inherent development lag and inaccuracy, are push factors for the integration of machine learning into the flight control team's suite of tools. The ultimate goal being to use these machine learning methods for science observation planning.

ACKNOWLEDGMENT

We acknowledge the financial support of the Slovenian Research Agency via the grants P2-0103, J2-9230, J4-7362, L2-7509 and the young researcher grants for MP.

REFERENCES

[1] Luke Lucas and Redouane Boumghar. Machine learning for spacecraft operations support - The Mars Express Power Challenge. In *Sixth International Conference on Space Mission Challenges for Information Technology, SMC-IT 2017*, pages 82–87, 2017.

[2] Matej Petković, Redouane Boumghar, et al. Machine learning for predicting thermal power consumption of the mars express spacecraft. *IEEE Aerospace and Electronic Systems Magazine*, 34(6), 2019.

[3] Hanen Borchani, Gherardo Varando, et al. A survey on multi-output regression. *Data Mining and Knowledge Discovery*, 5(5):216–233, 2015.

[4] Martin Breskvar, Dragi Kocev, et al. Predicting thermal power consumption of the mars express satellite with machine learning. In *Sixth International Conference on Space Mission Challenges for Information Technology, SMC-IT 2017*, pages 88–93, 2017.

[5] Dragi Kocev, Celine Vens, et al. Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3):817–833, 2013.

[6] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[7] Vân Anh Huynh-Thu, Alexandre Irrthum, et al. Inferring regulatory networks from expression data using tree-based methods. *PLOS ONE*, 5(9):1–10, 09 2010.

[8] Matej Petković, Dragi Kocev, et al. Feature ranking for multi-target regression. *Machine Learning Journal*, 2019. Under Review.

Chapter 9

Relational Feature Generation and Ranking

In this chapter, we present in detail the fifth and final set of contributions of the thesis: those concerning the relational learning setting. Our contributions in this context include:

1. An extension of decision trees for classification and ensembles thereof to the context of classification in a relational setting.

2. An extension of ensemble-based feature ranking scores for classification from the context of tabular data to the relational setting.

3. An extensive experimental evaluation of the newly proposed relational tree ensembles and feature importance scores on a collection of benchmark datasets.

Chronologically, we first proposed an extension of classification trees and ensembles thereof (together with a relational adaptation of the Genie3 score) in a paper presented at the ISMIS-2020 conference (Petković, Ceci, et al., 2020). Somewhat different to the other experiments presented in this thesis, we first extended the gradient boosting ensembles since they tend to allow for shallower trees, which is beneficial in terms of the time-complexity of the method. The main finding regarding the feature rankings was that they are relevant and make sense, as long as the underlying model successfully learns the target concept.

We then extended this paper to a journal submission (Petković, Pio, et al., 2020). We included the bagging and random forest ensembles, and experimented also with the Symbolic ranking score. Moreover, we compared the performance of our (ensembles of) relational trees to the performance of the other state-of-the-art relational classifiers, and analyzed the sensitivity of boosting ensembles to their hyperparameters, since tuning all of them takes a considerable amount of time if not parallelized.

The main findings are that the proposed classifiers achieve high accuracy and outperform the current state-of-the art. For many datasets, critical to achieving such performance is the inclusion of aggregates in the tests in the internal nodes of the trees. Moreover, the obtained models are explainable since the proposed two feature ranking scores (Genie3 and Symbolic) yield intuitively correct feature rankings for the considered datasets.

The work presented in this Chapter refers to the following hypotheses (as defined in the introduction):

H2: It is possible to extend ensemble-based approaches to classification and the corresponding feature ranking methods to handle relational data.

H10: The proposed ensemble-based approaches to relational classification yield state-of-the-art performance as well as relevant feature rankings.

These two hypotheses are confirmed with the design and implementation of the ensemble-based approaches to relational classification and feature-ranking, presented in this Chapter, and the experimental study comparing their performance.

The papers included in this Chapter are:

- Petković M., Ceci M., Kersting K., Džeroski S. Estimating the Importance of Relational Features by using Gradient Boosting. In *Proceedings of the 25th International Symposium on Methodologies for Intelligent Systems.* Graz, 2020.
- Petković, M., Pio, G., Ceci, M., Škrlj, B., Kersting, K., and Džeroski, S. Relational Tree Ensembles and Feature Rankings. *Artificial Intelligence*, 2020. To be submitted.

The contribution of Matej Petković to these papers are as follows. MP contributed to the design of the ensemble-based relational classification and feature ranking methods, and implemented these methods in computer code. He also participated in designing the experiments, carried out the experiments, and processed their results. He drafted the papers and revised them following the feedback from the co-authors and reviewers.

Estimating the Importance of Relational Features by using Gradient Boosting*

Matej Petković[1,2][0000−0002−0495−9046], Michelangelo Ceci[1,3][0000−0002−6690−7583], Kristian Kersting[4][0000−0002−2873−9152], and Sašo Džeroski[1,2][0000−0003−2363−712X]

[1] Jozef Stefan Institute, Jamova 39, Ljubljana, Slovenia
[2] Jozef Stefan Postgraduate School, Jamova 39, Ljubljana, Slovenia
{matej.petkovic,saso.dzeroski}@ijs.si
[3] Università degli Studi di Bari Aldo Moro, via E. Orabona 4, Bari, Italy
michelangelo.ceci@uniba.it
[4] CS Department, TU Darmstadt, Hochschulstrasse 1, Darmstadt, Germany
kersting@cs.tu-darmstadt.de

Abstract. With data becoming more and more complex, the standard tabular data format often does not suffice to represent datasets. Richer representations, such as relational ones, are needed. However, a relational representation opens a much larger space of possible descriptors (features) of the examples that are to be classified. Consequently, it is important to assess which features are relevant (and to what extent) for predicting the target. In this work, we propose a novel relational feature ranking method that is based on our novel version of gradient-boosted relational trees and extends the Genie3 score towards relational data. By running the algorithm on six well-known benchmark problems, we show that it yields meaningful feature rankings, provided that the underlying classifier can learn the target concept successfully.

Keywords: feature ranking · relational trees · gradient boosting

1 Introduction

One of the most frequently addressed tasks of machine learning is predictive modeling, where the goal is to build a model by using a set of known examples, which are given as pairs of target values and vectors of feature values. The model should generalize well to previously unseen combinations of feature values and accurately predict the corresponding target value. In this paper, we limit ourselves to classification, i.e., to the case when the target can take one of finitely many nominal values. For example, when modeling the genre of a movie, the possible values might be `thriller`, `drama`, `comedy` and `action`.

In the simplest and most common case, the data comes as a single table where rows correspond to examples and columns to features, including the target.

* This is financially supported by the Slovenian Research Agency (grants P2-0103, N2-0128, and a young researcher grant to MP).

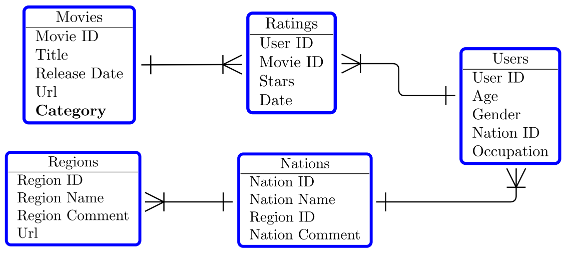


Fig. 1: The `movies` dataset consists of 5 tables and 4 relations among them. Single and multiple ends on the connections among the tables represent one-to- (or -to-one) and many-to- (or -to-many) relations, respectively. The task at hand is to predict the property Category (genre) of a movie (shown in bold).

However, when predicting the genre of a movie, it might be beneficial to not only know the properties of the movie (i.e., the feature values), but also to have access to the properties of ratings of the movie (e.g., number of stars and date) on some website. As a consequence, the data is now represented by two tables: one describing the movies and the other one describing the ratings. A link between them might then be the relation $\mathit{belongsTo}(\text{rating}, \text{movie})$ that tells which movie a given rating rates. The actual `movies` dataset used in our experiments (shown in Fig. 1) contains five additional tables and four relations among them.

In the era of abundance of data, such complex datasets are more and more frequent in various domains, e.g., sports statistics, businesses, epidemiology [14], among others. In biology, for instance, protein networks [18] encode complex data about proteins and their functions, together with different protein-protein interactions. Concrete examples are given in Sec. 5. In all these cases, relational data are very much of interest because they offer a powerful representation language where a broad spectrum of predictive modeling tasks can be applied. A notable example is link prediction where the goal is to predict whether two examples are in a given relation or not [4], say, two researchers are co-authors of a paper [16]. It is also possible to predict links of multiple types [3]. Another prominent task is classification. For example, movies can be categorized into different genres. Again, predicting multiple targets is possible [12,14].

The main focus of the present paper, however, is not predictive modeling, but rather feature ranking. In the simplest case when the data is given as a single table, the task of feature ranking is to estimate the importance of each descriptive feature when predicting the target value. The features are then ranked with respect to their importance. A possible motivation for performing feature ranking is dimensionality reduction. Discarding the less relevant features from the dataset (before proceeding to predictive modeling) results in lower time and space complexity of learning the models, which may also be more accurate since a lower number of features reduces overfitting. Moreover, feature ranking can explain black box models such as ensembles of decision trees [1,5] or neural networks, which may be of crucial importance in domains such as medicine.

Feature ranking on relational data, however, is not a trivial task. In fact, with two tables or more (and relations between them) at hand, the notion of a feature is a more general concept. It can refer to a relation between tables or to a descriptive attribute, which is part of the description of the objects in a given table. Furthermore, the existing relations and descriptive attributes can be combined into new ones. For example, the schema in Fig. 1 implicitly defines a feature whose value for a given movie is the name of the nation which contributes the highest number of ratings for the movie. Consequently, the feature space can be extremely large, and discovering only the most relevant features for a given predictive modeling task might have a high value. A feature ranking algorithm can estimate the importance of the features explicitly given (such as release date in the `movies` dataset), or it can use some heuristic to construct new features from the ones existing in the descriptive relations.

In this paper, we propose a feature ranking algorithm that focuses on the second approach, i.e., heuristically introduces new features. The novel heuristic for building relational trees starts the search for relevant features at those which are explicitly given, and gradually proceeds towards more and more complex ones if needed. To improve the quality (and stability) of feature rankings, we use gradient boosting ensembles [5] instead of a single tree. Once the ensemble is built, feature ranking is seamlessly computed out of it by using the adaptation of the Genie3 score [9].

The remainder of the paper is organized as follows. Sec. 2 explains the necessary background and notation, Sec. 3 reviews related work, Sec. 4 introduces our extension of gradient boosting to relational data and our novel feature ranking algorithm, Sec. 5 gives the experimental setup, while Sec. 6 the results are discussed. Finally, Sec. 7 concludes the work.

2 Background on Relational Predictive Modeling

Let $\mathcal{X}$ be a generic domain that identifies an object type and x be an example (instance) of such type. Every object $x \in \mathcal{X}$ is represented in terms of its ID (object identifier) and a list of attribute values. A relation r of arity $t \geq 1$ is defined as a subset of a Cartesian product of the domains $\mathcal{X}_i$ of relation's arguments X_i, $1 \leq i \leq t$. The fact that $x_1 \in \mathcal{X}_1, \dots, x_t \in \mathcal{X}_t$ are in a relation r, is denoted either as $(x_1, \dots, x_t) \in r$ or $r(x_1, \dots, x_t)$.

If $t = 1$ the relation describes a property of a single object, e.g., *isMale*(user). If $t = 2$, the relation is said to be binary. A binary relation is one-to-many if each $x_1 \in \mathcal{X}_1$ is in a relation with *at least* one $x_2 \in \mathcal{X}_2$, and each x_2 is in a relation with *exactly* one x_1. The other three cases (one-to-one, many-to-one and many-to-many) are defined analogously.

The goal of the relational classification task is thus to learn a model that uses descriptive relations to predict the target relation $r(X_0, Y)$ where the first component corresponds to the example to be classified and the last component corresponds to the target values. For the link prediction task, the target relation is of form $r(X_0^1, X_0^2, Y)$ where both X_0^1 and X_0^2 correspond to examples and the last component corresponds to the target values. Thus, a relational representation of the data elegantly unifies different classification tasks into the same framework. However, in this work, we focus on the $r(X_0, Y)$ target relations only.

A dataset is typically stored as a group of tables and relations between them. For example, the `movies` dataset in Fig. 1 contains five tables that describe five different types of objects: movies, ratings, users, nations, and regions. Every movie is given as a 5-tuple, consisting of movie ID, title, release date, url, and category. Similarly, ratings are given as 4-tuples of user ID (the author of the rating), movie ID (the movie that rating refers to), stars and date.

Since both the fist component of movies and the second component of ratings are of the same type (movie ID), the dataset also implicitly defines the one-to-many relation via which we can find, for example, all ratings of a given movie. Similar links exist between users and ratings (connected via user ID columns), between users and nations, and between nations and regions. Should there be a relation among users, e.g., *isARelative*, we would need an additional table, e.g., relatives, that would contain two columns (both of the type user ID) and would explicitly list all the pairs of relatives.

Prior to applying our algorithm (and also the majority of those discussed in the related work), the data at hand should be converted into a pure relational representation, coherent with the formal description above, where all facts are given as relation elements $r(x_1, \dots, x_t)$, e.g., $\mathit{gender}(\text{Ana}, \text{female})$.

3 Related Work

Since our feature ranking is embedded into a classifier [7], it is closely related to relational ensemble techniques. Gradient boosting of relational trees is proposed in [11]. It takes relations (given as sets of tuples) as the input and builds gradient-boosted regression trees. As usual, multi-class problems demand 1-hot encoding of the target variable, which converts the original dataset into a series of 1-versus-all classification problems, and an ensemble is built for each of them separately. In turn, a tree induction in an ensemble bases on the TILDE learner [17] and its predecessor FOIL [13], which results in two possible limitations of the method.

First, it allows only for the nominal descriptive features, thus the numeric ones (e.g., age of a user) should first be discretized into bins which results in the loss of ordering of values. The second possible limitation are candidate tests in the internal nodes of the trees. Without loss of generality, we assume that the variable X_0 that corresponds to the example ID whose target values is to be predicted, always appears as the first component of a relation r. In this case, the candidate splits are of two types. First, a split can be a conjunction of predicates

$$r_1(X_0, x_1^2, \dots) \wedge \dots \wedge r_j(X_0, x_j^2, \dots) \wedge r_{j+1}(x_{j+1}^1, x_{j+1}^2, \dots) \wedge \dots \wedge r_\ell(x_\ell^1, x_\ell^2, \dots)$$

where $\ell \geq 1$, and x_i^k is the value of the variable at position k for relation r_i. Second, some of the variables X_i^k may not be grounded yet (i.e., their value is not determined). In that case, a split is of the form

$$\exists\, X_{i_1}^{k_1} \dots \exists\, X_{i_n}^{k_n} :\; r_1(X_0, x_1^2, \dots) \wedge \dots \wedge r_\ell(x_\ell^1, x_\ell^2, \dots) \tag{1}$$

where n is the number of non-grounded variables. When the actual example ID x_0 reaches a split, X_0 takes its value, the split is evaluated, and the example follows the YES or NO branch accordingly. That means that having splits like *Is the average age of users that rated a movie, larger than 60?* is not possible. That was overcome by introducing aggregates into TILDE [17]. There, also constrained aggregation is possible, e.g., *Is the average age of users that have contributed at least 5 ratings in total, and have rated the given movie, larger than 60?*. For the exact formulation of the possible split tests, we refer the reader to [17] where the extension of the method to relational random forests is described.

Regarding relational feature ranking methods, there is only the FARS method [8], which belongs to the group of filters [7], i.e., no predictive model is needed for computing the ranking. As such, it cannot be used for explaining the decisions of classifiers. It is suitable for classification datasets and is based on propagation of the class values from the table that contains the target attribute to the other tables. The method supports neither estimation of numeric attributes nor the estimation of implicitly defined features.

4 Our Method

Here, we describe our two contributions. We first present the proposed test split generation for relational trees (and boosting ensembles). Afterward, the proposed feature ranking approach is introduced.

4.1 Relational Gradient Boosting with Aggregates

Relational trees are built with the standard top-down induction procedure [2] whose main part is greedily finding the optimal test (according to a heuristic h that is based on an impurity measure, e.g., Gini index [2]) that splits the data into two subgroups. The point at which relational tree induction differs from the standard one is how the candidate splits are created. Indeed, one option is using TILDE with aggregates. However, also from the feature ranking perspective, we find the following feature-value back-propagation splits more appropriate, since by doing so (in contrast to TILDE), we can directly link the values of a given feature to the given example ID, no matter which table is the feature present in.

Consider Fig. 2, which depicts our candidate test generation. Each test is generated in two stages. First (as shown in Fig. 2a), we start from an example ID x_0, and follow any relation r_1 where x_0 can appear in. The group g_1 of all tuples $\boldsymbol{x}_1^i \in r_1$, which x_0 is part of, is thus found. Then, for each of the tuples $\boldsymbol{x}_1^i$, we recursively repeat the search from $\boldsymbol{x}_1^i$, thus finding the group of examples g_2^i by following any relation r_2 that shares at least one input domain $\mathcal{X}$ with relation r_1. The search is finished after at most ℓ steps which is a user-defined

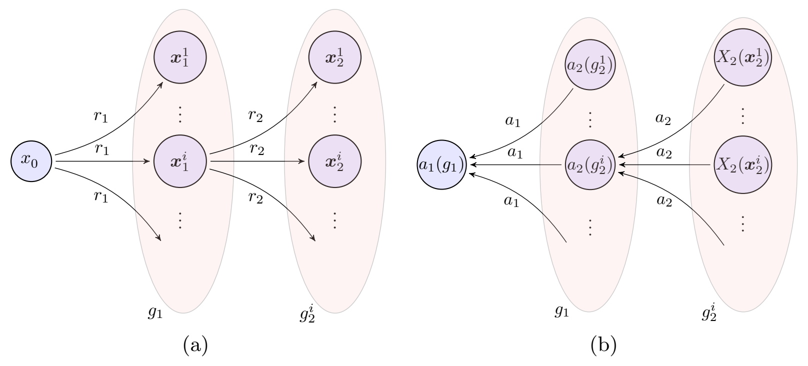


Fig. 2: Candidate test generation: Finding related tuples by following descriptive relations (a), and back-propagation of the feature values by aggregation (b).

parameter. In Fig. 2a, we have $\ell = 2$. For example, following the schema of the `movie` data set in Fig. 1, we might start from $x_0 = \texttt{titanic}$, and find the group g_1 of all pairs $\boldsymbol{x}_1^i = (\texttt{titanic}, \texttt{ratingID}^i)$. For each such pair, a (singleton) group g_2^i of all pairs $(\texttt{ratingID}^i, \texttt{stars})$ is found.

After the search is finished, the back-propagation of feature values by aggregation starts, by choosing one of the variables that were introduced in the last step of the search. Let this be X_ℓ, i.e., X_2. Its type defines possible starting aggregates. We can always use *count* or *countUnique*. Additionally, we can use *max*, *min*, *mean*, and *sum* if X_2 is numeric, and *mode* if X_2 is nominal. The data type returned by the first aggregate in turn defines the possible options for the next one in the chain. In general, we proceed form right to left (Fig. 2b).

By using aggregate a_2, we aggregate every group g_2^i over X_2. Following the example above, the variable at hand would be the number of stars. Then, the tuple $\boldsymbol{x}_1^i \in g_1$ is effectively replaced by the aggregated value $a_2(g_2^i)$. After this is done for all tuples $\boldsymbol{x}_1^i$, the group g_1 is similarly aggregated into the final value $a_1(g_1)$. If we set a_2 to *mean* and a_1 to *max*, in the above example, we compute the maximal rating of a movie (the mean of a single value is the value itself).

Finally, the procedure for generating candidate splits proceeds to finding the optimal threshold ϑ the aggregated values are compared against, and chooses the best test among the candidates. This covers also the existentially quantified split tests in Eq. (1) if $\vartheta = 0$ and the aggregates are set to *count*. This motivates the idea to allow the algorithm to continue the search for a good split in the YES-child at any of the steps from its parent node.

Therefore, the evaluation of the tests becomes more time-consuming deeper in the tree, so gradient boosting [5] where the trees are shallower is more efficient than, e.g., bagging [1] where bias-variance decomposition of the error reveals that trees should be grown to a full depth.

Using aggregates is necessary since the preliminary experiments (not part of this work) show that this increases the expressiveness of the splits which reflects in substantially improved predictive power of the models.

4.2 Feature Ranking

Now, we are ready to introduce our novel relational feature ranking. Let (R, A, ϑ) be a triplet denoting a test in a node $\mathcal{N}$ in a tree $\mathcal{T}$, where R and A are lists of relations and aggregates used in the test. The size s of a test is defined as $s = |R|$ $(= |A|)$. Let $E(\mathcal{N})$ be the set of examples that reach the node $\mathcal{N}$, and $h(\mathcal{N})$ the heuristic value of the split in $\mathcal{N}$ [2]. We write $r \in \mathcal{N}$ if $r \in R$. Then, a natural extension of the Genie3 score [9] for the already existing relations r is

$$\mathit{importance}(r) = \sum_{\mathcal{T}} \sum_{\mathcal{N} \in \mathcal{T}} \frac{\mathbf{1}[r \in \mathcal{N}]}{s(\mathcal{N})} h(\mathcal{N}) |E(\mathcal{N})|, \tag{2}$$

where $\mathbf{1}$ is the indicator function. The definitions says that all relations that appear in a given node are rewarded equally and proportionally to the heuristic value. The term $|E(\mathcal{N})|$ assures that relations that appear higher in a tree (and influence more examples) receive bigger award.

Please note that Eq. (2) is naturally extended to (parts of) lists R of relations by summing up the importances of their atomic parts. Note also that the relations and combinations thereof that do not appear in the ensemble, have the importance score with the value 0.

5 Experimental Setting

In order to investigate the performance of our relational feature ranking methods, we computed feature rankings for six well-known datasets. In addition to the `movie` dataset (shown in Fig. 1), these were: `basket` [10] (basketball players, coaches, teams, etc.), `IMDB` [6] (movies, actors, directors, etc.), `Stack` [15] (user posts, users and comments), and `Yelp` [19] (different businesses, their reviews, users etc.). Additional statistics for the data are given in Tab. 1.

In our experiments, the quality of the underlying predictive model will be used as a proxy for the quality of the ranking, thus 10-fold cross-validation (CV) is performed. For each training set, internal 3-fold CV was performed to tune the boosting parameters via grid-search. The parameters (and their possible values) that were optimized are shrinkage ($\{0.05, 0.2, 0.4, 0.6\}$), proportion of chosen examples ($\{0.6, 0.8, 1.0\}$), proportion of the evaluated tests in a node ($\{0.2, 0.4, \ldots, 1.0, \mathrm{sqrt}\}$), and maximal depth of trees ($\{2, 4, 6, 8\}$). Ensemble size was set to 50 and the size of splits was $\ell \leq 2$.

6 Results and Discussion

The experimental results are summarized in Fig. 3. It shows the feature importances, averaged over the 10-folds of CV, and the three most relevant features, for each dataset. We observe two qualitatively different results: for the datasets `basket`, `Stack` and `UWCSE` (and to some extend `Yelp`), it is evident that feature importance score have mostly converged to their final values, meaning that the Gini heuristic in the splits goes down to 0.0 and the trees with higher indices do not influence the ranking or predictions of the model. This is confirmed by the accuracy of the corresponding models: 0.98 (`basket`), 0.95 (`Stack`), 0.83 (`basket`) and 0.88 (`Yelp`). Since accuracy on the training sets are even higher, this means that only a few training examples are being miss-classified and the target values at 50-th iteration are mostly close to 0.0.

The two most prominent members of the second class of results are `IMDB` and `movie` where the feature importance scores are still noticeably growing. On the other hand, the order of the features is mostly fixed, so trees are similar to each other, but unable to fully solve the predictive problem. Indeed, the accuracy of the corresponding two models is 0.63 (`IMDB`) and 0.57 (`movie`) which is closer to the default accuracy than in the previous cases (see Tab. 1).

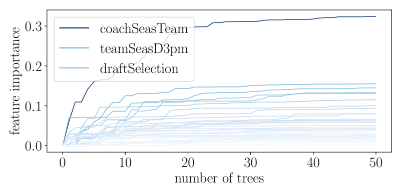
The next observation is that for all six datasets, a group of 1–3 most important features is established. The difference between this group and the other features is most notable for the `UWCSE` dataset. Here, the goal is to predict, which discipline a given course belongs to, and the most important relation is $\mathit{taughtBy}(\text{course}, \text{person}, \text{session})$. This is not surprising as it is also the link to the $\mathit{advisedByDiscipline}(\text{person}, \text{person}, \text{discipline})$, ranked second. Since professors typically work only in one discipline, the discipline a professor advises someone in, is likely equal to a discipline that the professor teaches.

A similar difference between the top feature and the others is visible also in the cases of the `Yelp` dataset (as well as for `basketball`). In the case of `Yelp`, the goal is to predict the category of a business (e.g., restaurant, Health&Medical etc.), and counting in how many tuples of the *attributes* relation a business appears, is a good indication of the target value. For example, restaurants tend to have a lot of attributes such as classy, hipster, romantic, dessert etc. whereas Health&Medical places are sometimes described only by `ByAppointmentOnly`.

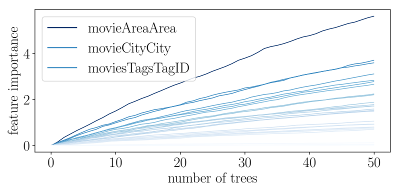
In the cases where the models are not that accurate, feature ranking is a way of seeing whether the model overfits and if some relations should be excluded from the descriptive space. This happens in the case of the `IMDB` dataset where

Table 1: Data characteristics: number of tables, number of relations in the final representation, sum of relation sizes (number of descriptive facts), number of target facts, number of classes, and the proportion of the majority class.

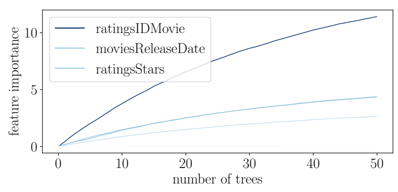
dataset	tables	relations	descriptive facts	target facts	classes	majority class
basket	9	118	630038	95	2	0.70
IMDB	21	57	614662	8816	4	0.58
movie	5	16	183469	1422	4	0.51
Stack	7	52	383040	5855	5	0.36
UWCSE	12	15	1961	115	5	0.25
Yelp	9	51	3348181	24959	4	0.57



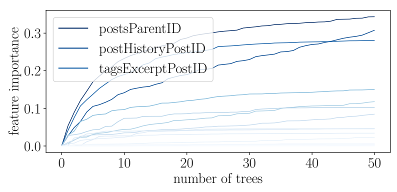
(a) `basket`



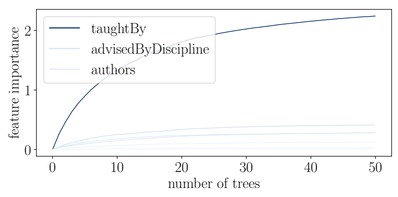
(b) `IMDB`



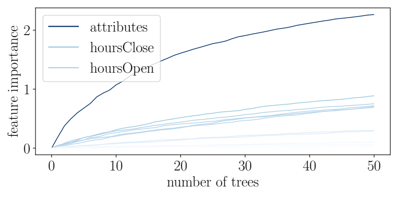
(c) `movie`



(d) `Stack`



(e) `UWCSE`



(f) `Yelp`

Fig. 3: Development of feature importance values with the number of trees, for different datasets. Every line corresponds to a feature that is present in an ensemble. It's color corresponds to the final feature importance at 50 trees. Additionally, the three most relevant features are listed.

the goal is to predict the genre of a movie but the most relevant feature is the area where the movie was taken.

7 Conclusions and Future Work

We have proposed an adaptation of the Genie3 feature ranking to relational data, using our adaptation of gradient boosting as the underlying ensemble, and evaluated its appropriateness empirically. The main motivation for choosing boosting was that the trees learned in boosting can be quite shallow as compared to those learned in bagging. However, parameter-tuning for boosted relational trees takes a considerable amount of time, so we plan to extend the proposed approach to other ensemble methods, such as bagging and random forests (and parallelize them). Also, the feature-ranking-motivated definition of the possible split tests will be evaluated in a predictive modeling scenario. Finally, we plan to compare the different relational ensembles against each other and against other learners.

References

1. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123140 (1996)
2. Breiman, L., Friedman, J., Olshen, R., Stone, C.J.: Classification and Regression Trees. Chapman & Hall/CRC (1984)
3. Davis, D., Lichtenwalter, R., Chawla, N.V.: Multi-relational link prediction in heterogeneous information networks. In: 2011 International Conference on Advances in Social Networks Analysis and Mining. pp. 281–288 (2011)
4. Dong, Y., Tang, J., Wu, S., Tian, J., Chawla, N.V., Rao, J., Cao, H.: Link prediction and recommendation across heterogeneous social networks. In: 2012 IEEE 12th International Conference on Data Mining. pp. 181–190 (2012)
5. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics **29**(5), 1189–1232 (2001)
6. GroupLens Research: Imdb dataset, data retrieved from `https://grouplens.org/datasets/hetrec-2011/`
7. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3**, 1157–1182 (2003)
8. He, J., Liu, H., Hu, B., Du, X., Wang, P.: Selecting effective features and relations for efficient multi-relational classification. Computational Intelligence **26**, 258–281 (2010)
9. Huynh-Thu, V.A., Irrthum, A., Wehenkel, L., Geurts, P.: Inferring regulatory networks from expression data using tree-based methods. PLOS ONE **5**(9), 1–10 (09 2010). https://doi.org/10.1371/journal.pone.0012776
10. Moore, A.W.: Basket dataset, data retrieved from `http://www.cs.cmu.edu/~awm/10701/project/data.html`
11. Natarajan, S., Kersting, K., Khot, T., Shavlik, J.: Boosted statistical relational learners: From benchmarks to data-driven medicine, pp. 1–74. Springer (2014)
12. Pio, G., Serafino, F., Malerba, D., Ceci, M.: Multi-type clustering and classification from heterogeneous networks. Information Sciences **425**, 107–126 (2018)
13. Quinlan, J.R.: Boosting first-order learning. In: Arikawa, S., Sharma, A.K. (eds.) Algorithmic Learning Theory. pp. 143–155. Springer Berlin Heidelberg (1996)
14. Serafino, F., Pio, G., Ceci, M.: Ensemble learning for multi-type classification in heterogeneous networks. IEEE Transactions on Knowledge and Data Engineering **30**(12), 2326–2339 (2018)
15. Stack Exchage: Stack dataset, data retrieved from `https://archive.org/details/stackexchange`
16. Sun, Y., Barber, R., Gupta, M., Aggarwal, C.C., Han, J.: Co-author relationship prediction in heterogeneous bibliographic networks. In: 2011 International Conference on Advances in Social Networks Analysis and Mining. pp. 121–128 (2011)
17. Vens, C.: Complex aggregates in Relational Learning. Ph.D. thesis, Faculteit Ingenieurswetenschappen, Katholieke Univeristeit Leuven (2007)
18. Škrlj, B., Kralj, J., Lavrač, N.: Targeted end-to-end knowledge graph decomposition. In: Riguzzi, F., Bellodi, E., Zese, R. (eds.) Inductive Logic Programming. Springer (2018)
19. Yelp: Yelp dataset, data retrieved from `www.yelp.com/dataset_challenge`

Relational tree ensembles and feature rankings

Matej Petković[a,b], Gianvito Pio[c], Michelangelo Ceci[c,a], Blaž Škrlj[a,b], Kristian Kersting[d], Sašo Džeroski[a,b]

[a]*Jozef Stefan Institute, Jamova 39, Ljubljana, Slovenia*
[b]*Jozef Stefan Postgraduate School, Jamova 39, Ljubljana, Slovenia*
[c]*Università degli Studi di Bari Aldo Moro, via E. Orabona 4, Bari, Italy*
[d]*CS Department, TU Darmstadt, Hochschulstrasse 1, Darmstadt, Germany*

Abstract

As the complexity of data increases, so does the importance of powerful representations, such as relational and logical representations. The need for machine learning methods that can learn predictive models in such representations is strong. A characteristic of these representations is that they give rise to a huge number of features to be considered, thus drastically increasing the difficulty of learning in terms of computational complexity and the curse of dimensionality. Despite this, methods for ranking features in this context, i.e., estimating their importance are practically non-existent.

Among the most well-known methods for feature ranking are those based on ensembles, and in particular tree ensembles. To develop methods for feature ranking in a relational context, we adopt the relational tree ensemble approach. We thus first develop methods for learning ensembles of relational trees, extending a wide spectrum of tree-based ensemble methods from the propositional to the relational context, resulting in methods for bagging and random forests of relational trees, as well as gradient boosted ensembles thereof.

Complex relational features are considered in our ensembles: by using complex aggregates, we extend the standard collection of features that correspond to existential queries, such as 'Does this person have any children?', to more complex features that correspond to aggregation queries, such as 'What is the average age of this person's children?'. We also calculate feature importance scores and rankings from the different kinds of relational tree ensembles learned, with different kinds of relational features. The rankings provide insight into and explain the ensemble models, which would be otherwise difficult to understand.

We compare the methods for learning single trees and different tree ensembles, using only existential qualifiers and using the whole set of relational features, against 8 state-of-the-art methods on a collection of benchmark relational datasets, deriving also the corresponding feature rankings. Overall, the bagging ensembles perform the best, with gradient boosted ensembles following closely. The use of aggregates is beneficial and in some datasets drastically improves performance. In the latter cases, the aggregate-based features clearly stand out in the feature rankings derived from the ensembles.

1. Introduction

Nowadays, data are organized in complex structures, and the standard tabular representation, where each row corresponds to an object and each column corresponds to a characteristic, does not suffice anymore. Indeed, in real-world scenarios, we can easily find objects of multiple types, possibly described by different sets of characteristics, that are interconnected to each other through multiple types of relationships. Finance, biology, epidemiology and geography are just examples of application domains where such a scenario is very common. In biology, for instance, protein networks [1] encode complex data about proteins and their functions, together with different protein-protein interactions. Additional examples are reported in Sec. 5.1.

The application of standard machine learning methods to such data is not straightforward, due to the additional complexity introduced by the need to "navigate" the relationships. On the other hand, discarding the additional information provided by connected objects may lead to learn weak models. For example, if we are interested in learning a classification model for the genre of a movie, the characteristics of movies themselves, e.g., titles and release dates, might not be enough to discriminate among different genres. On the other hand, if we include additional data about the ratings (with their characteristics, e.g., date and value), as well as about users who provided such ratings (with their characteristics, e.g., age, gender and job), the model may have a better chance of accurately predicting the genre since, for example, comedies attract different kinds of viewers as compared to dramas.

In this scenario, data can naturally be represented through a relational database where data are stored in multiple tables, that are interconnected via foreign keys that implicitly represent and define the relationships among objects and their characteristics (i.e., columns). For example, the dataset

Movies (see Sec. 5.1) can be represented through the relational schema depicted in Fig. 1, where the table storing movies (that is the target of the learning task) is linked to the table *Ratings* from which it is possible to understand which *Users* provided such ratings as well as their characteristics, spread over the table *Users* itself and other tables representing their geographical location (*Nations* and *Regions*).

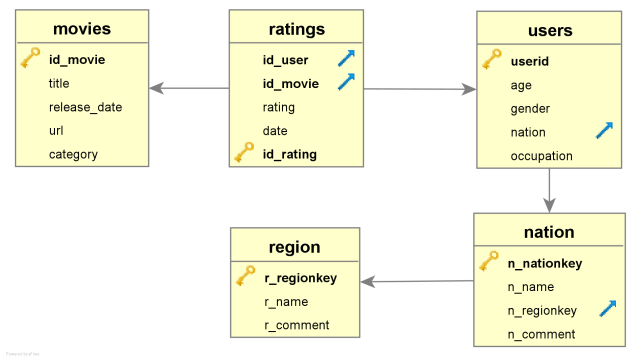


Figure 1: The relational schema of the `Movies` dataset, consisting of 5 tables. Fields shown in bold represent primary keys (with a key symbol) and foreign keys (with an arrow). The task at hand is to predict the property Category (genre) in the target table Movies.

A possible solution to analyze such data consists in the adoption of *propositionalization* approaches. They are mainly based on the transformation of relational data into a propositional (or attribute-value or feature-based) form, by constructing additional features that represent the relationships and the properties of data related to target objects. This approach decouples the construction of features from the phase of learning the predictive model [2], in order to apply standard classification methods to transformed data [3]. Solutions based on propositionalization mainly resort to two different approaches, namely, database-oriented [4] or logic-oriented techniques [5]. The former work by materializing joins according to foreign keys and allowing for fast aggregations. The latter require data represented as Prolog facts and are able to consider complex background knowledge. This aspect makes logic-oriented propositionalization able to provide expressive first-order models by constructing first-order logic features, at the cost of possible lower efficiency.

Differently, *structural* approaches take into account the structure of original data and directly navigate the tables following the foreign keys. Therefore, the whole hypothesis space is directly explored during the learning process (see a more comprehensive overview in Sec. 2).

Following this line of research, in this paper, we propose a novel method, called RE3PY[1], to solve the classification task in the relational setting, namely on the prediction of the value of a categorical attribute of previously unseen examples of a target table. RE3PY is based on the top-down induction of decision trees [6], properly extended to the relational setting. Split candidates are based on conditions on *paths* involving multiple tables (following the foreign keys) and aggregates of attributes. Following the example in Fig. 1, a path can convey information on: 1) the rating of a movie; 2) the author of the rating; 3) the age of the author, that can be aggregated through the average, minimum, maximum aggregation functions, back to the movie. In this case, we obtain new aggregated features associated to the movie, representing the average, the minimum and the maximum age, respectively, of the authors of the ratings of such movie. Similarly, we can generate new features representing the nationality of the majority of the authors of ratings associated to a given movie.

In RE3PY, by iterating over the possible combinations of relations and aggregates, we construct multiple new features for the examples of the target table during the learning phase and, once the most promising representation has been selected, others are discarded, making the process much more space-efficient than approaches based on propositionalization [7]. Moreover, RE3PY exploits different ensemble approaches such as Bagging, Random Forests and Gradient Boosting, that have proven to provide a significant contribution in terms of prediction accuracy [8, 9, 10].

An additional advantage of our method is the possibility to "motivate" the returned predictions. This is in line with recent advances in machine learning where the iterpretability of the models and the explainability of the predictions are gaining increasing attention, especially in particular application domains such as medicine [11, 12, 13]. Similarly to classical classification and regression trees, models learned by RE3PY are inherently interpretable. However, the adoption of ensemble techniques significantly compromises the explainability of the predictions. Our method also considers this aspect by providing interesting clues about the contribution provided by each feature in the learned model. This is achieved by extending ensemble-based feature

[1]The implementation is available at `https://github.com/re3py/re3py`

ranking approaches [9, 14] to the relational setting. Feature ranking is a task, tightly coupled with predictive modelling, which goal is that of determining to what extent a given descriptive attribute (feature) influences the prediction of a target attribute. In the standard tabular setting, approaches for feature ranking have been widely investigated [15] whereas, at the best of our knowledge, this is the first work able to perform feature ranking in the relational setting.

The remainder of the paper is organized as follows: Sec. 2 briefly describes the work related to this paper; in Sec. 3 we provide the problem statement, while in Sec. 4 we describe in detail the proposed method; in Sec. 5, we describe the experimental setup, whereas in Sec. 6 we present and discuss the results of our experimental evaluation; finally, Sec. 7 concludes the paper and outlines some future works.

2. Related Work

The method proposed in this paper has its roots in the fields of relational learning and heterogeneous network classification. In the following, we briefly present existing work related to these research fields.

In the literature, we can find several historical methods, mainly based on Inductive Logic Programming (ILP), that can solve the classification task in the relational setting. Noteworthy approaches include Quinlan's FOIL [16], that learns function-free Horn clauses (a subset of first-order predicate calculus), PROGOL [17], that combines inverse entailment with general-to-specific search through a refinement graph that guarantees a solution with the maximum compression in the search space, and TILDE [18], that learns a predicate logic theory by means of a so-called logical decision tree, a first-order logic upgrade of the classical decision trees. Such historical approaches, although able to produce highly-interpretable and possibly accurate models, easily turned out to be very inefficient in real-world scenarios.

The more recent method BoostSRL [19] extends TILDE towards the induction of gradient boosting ensembles. However, in contrast to TILDE, it does not exploit the aggregates, which makes tree (ensemble) induction faster at the price of a lower quality of the models. Moreover, neither TILDE nor BoostSRL, which can be considered the closest approaches to the method proposed in this paper, offer the possibility to perform feature ranking. Additional differences, mainly related to the language bias (that originate from the strategy adopted to construct the splits in the internal nodes) are discussed in Sec. 4, where our method is presented.

A probabilistic approach is that followed by 1BC [20] and Mr-SBC [21][22], that are based on the naïve Bayes classification method and use first-order classification rules for the computation of the posterior probability for each class. In addition, Mr-SBC exploits: *i)* logical factorization to simplify the computation of the posterior probabilities [23]; *ii)* both discrete and continuous attributes, that are discretized through a supervised approach; *iii)* the database schema, during the generation of classification rules. In [24], two relevant extensions of Mr-SBC have been proposed, namely ST-MrSBC (Self-Training MrSBC), an ensemble-based approach that can capture autocorrelation phenomena by resorting to a self-training method, and MT-MrSBC (Multi-Type MrSBC) that also predicts possible missing values and catches dependencies among the models built for different relations.

In [25], the authors introduced the tool RelWEKA, which extends the well-known WEKA toolkit in order to allow it to work in the relational setting. In particular, it exploits specific relational distance measures (e.g., the Relational Instance-Based Learning (RIBL) measure [26]), and kernels (e.g., the Minkowski RIBL set distance [27]) to implement the relational variant of classical algorithms, such as k-NN and Support Vector Machines.

More recent approaches model relational data as information networks. In this setting, tuples in a relational database correspond to nodes in the network, while foreign keys correspond to edges [7]. Most of existing works consider the within-network (or transductive) setting, that is, they model a partially labeled network and estimate the label for unlabeled nodes. Moreover, initial works mainly focused on homogeneous networks, where all the nodes are of the same type, related by one type of relationships. Approaches in this area are mainly based on collective inference (see [28], [29], [30] and [31]), active inference (see [32]), semi-supervised and transductive inference (see [33], [34] [35] and [36]). One relevant example in the transductive classification setting is the work in [37], where the authors proposed GNetMine, a graph-based regularization framework, that models the link structure in arbitrary information networks, with an arbitrary number of types of objects and links. GNetMine analyzes each sub-network associated with each type of link and aims at preserving its consistency. Another relevant example is RankClass [38], that combines ranking and classification tasks, by assuming that highly-ranked objects within a class should play more important roles in classification or, vice-versa, that class labels can be relevant to build a good ranking. RankClass iteratively builds a graph-based ranking model and, on the basis of the current ranking, adjusts the graph structure, in order to strengthen the weights of links in the subnetwork corresponding to each specific class, and to weaken links in the rest of the network.

Recently, in [39] the authors proposed a collective classification approach which aims at classifying objects of the same type in a heterogeneous network, on the basis of the analysis of meta-paths [40]. Meta-paths are sequences of link types connecting two objects to be classified, and are used to effectively assign labels to groups of interconnected instances. Classification is performed on a probabilistic fashion, on the basis of feature values of the objects under analysis, on "relational features" associated with meta-paths, and on the labels associated with objects appearing in meta-paths. For further insights on this stream of research the reader can refer to [41].

Some recent works also proposed the adoption of the predictive clustering framework in the context of network data. However, most of them are able to work only with homogeneous networks and, therefore, cannot analyze the possible heterogeneity of data and relationships, that is typical of the relational setting. For example, in [42] the authors follow an approach that combines a descriptive task (clustering) with a predictive task, and argue that using the network to represent data provides a unified framework for identifying and characterizing patterns in climate data. The network is actually built a-posteriori according to the Pearson's correlation coefficient, measured between pairs of nodes on the time series collected for the same variable. A clustering algorithm then groups interconnected nodes, aiming to minimizing the pairwise walking distance between nodes in the same cluster. Finally, the prediction model is learned locally for each cluster. However, this method tackles a regression task and, as clarified before, is not able to work with heterogeneous networks.

In [43], the authors proposed to learn predictive clustering trees to classify nodes in a network. Although the trees are learned with a classical top-down procedure, the adopted heuristics exploits an autocorrelation-based measure, thus explicitly taking autocorrelation into account. The same principle is adopted in [44], where network predictive clustering trees are used to predict gene functions. The difference is that, in this last work, a hierarchical multi-label classification scheme is considered. However, both methods are not able to work with heterogeneous networks and, therefore, cannot analyze real-world scenarios, that show different types of entities, related through multiple types of relationships.

The first attempt that really exploits a predictive clustering approach for classification tasks in heterogeneous networks (and, equivalently, in the relational setting) is the work proposed in [45]. The authors proposed the algorithm HENPC, that is able to solve multi-type classification tasks on heterogeneous networks. Methodologically, HENPC extracts possibly overlapping and hierarchically-organized heterogeneous clusters and exploits them to classify unlabeled nodes according to labeled nodes falling in the same clusters. Differently from the approach proposed in the present paper, HENPC is not based on the induction of trees, but rather a bottom-up approach for the identification of heterogeneous clusters, that generally leads to a non-optimal computational efficiency.

In conclusion, multiple attempts have been done to solve the classification task in the relational setting, but existing approaches are affected by one or more limitations, namely they: *i)* are able to work only with a restricted number of object types (often one type), *ii)* do not properly exploit the information coming from related objects (discarding relevant information), *iii)* suffer from a computational viewpoint, *iv)* do not appear highly accurate, or *v)* are not able to provide clues about the motivations behind a given prediction. In this respect, the work presented in this paper can be considered the first attempt aiming to achieve all such characteristics simultaneously.

3. Problem Statement

Before presenting our method, we introduce some useful notations and definitions which are necessary to formally define the problem we solve.
Given:

- A training database D composed of a set of n relational tables $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$;
- A set of primary key constraints on tables in $\mathcal{T}$;
- A set of foreign key constraints on tables in $\mathcal{T}$;
- A target table $T_t \in \mathcal{T}$;
- A discrete target attribute y belonging to T_t, different from the primary key of T_t, with values in $\mathcal{Y}_t$.

Find: a classification model $\psi_t : T_t \rightarrow \mathcal{Y}_t$, which is able to classify all the unlabeled instances (i.e., tuples) of the target table T_t in a database D' which has the same schema of D.

It is noteworthy that the learning setting we consider is not transductive, i.e., the applicability of the learned classification model is not limited to unlabelled instances which existence is known during the training, but can be applied to any database D', with the same schema of D, for which the values of the target attribute y are unknown.

Example 1. *Let us consider the database schema reported in Fig. 1 related to the* `movies` *database. The target table T_t is the table* `Movies`*, the target attribute y is the attribute* `Category` *and the foreign keys are represented by the relationships among the tables.*

The learner should exploit information from D, including non-target tables. To better clarify the way non-target tables are used, we introduce the definition of *task-relevant* objects and *reference* objects. Intuitively, a reference object ro corresponds to a tuple of the target table T_t, while a task-relevant object tro corresponds to a tuple of a non-target table, which is connected to ro by means of a foreign key path.

We formally introduce the concepts of *foreign key path* and *task-relevant objects* by means of the following definitions:

Def. 1. *A foreign key path is defined as an ordered sequence of tables $p_i = \langle T_{i_1}, T_{i_2}, \ldots, T_{i_s} \rangle$, where: $\forall_{j=1,2,\ldots,s}\ T_{i_j} \in \mathcal{T}$ and $\forall_{j=1,2\ldots,s-1}\ T_{i_j}$ has a foreign key to the table $T_{i_{j+1}}$ (or vice-versa). Note that the same table may be traversed multiple times within the same path if it is involved in multiple foreign keys.*

Example 2. *With reference to the database of Fig. 1, an example of foreign key path is $\langle$*`Movies`*,* `Ratings`*,* `Users`*,* `Nations`*,* `Regions`*$\rangle$. This foreign key path is of length 5 (i.e., $s = 5$). It is noteworthy that also paths of smaller length representing sub-sequences are valid, such as $\langle$*`Movies`*,* `Ratings`*,* `Users`*$\rangle$.*

Def. 2. *A task-relevant object $tro \in T_k$ ($T_k \in \mathcal{T}$) is* 1-related *to a reference object $ro \in T_t$ according to a foreign key path $p_i = \langle T_{i_1}, T_{i_2} \rangle$ if and only if all the following conditions are satisfied:*

- *$T_{i_1} = T_t$;*

- $T_{i_2} = T_k$;
- *there exists a foreign key constraint from T_k to T_t (or vice-versa) such that the foreign key of tro assumes the same value of the primary key of ro (or vice-versa).*

Def. 3. *A task-relevant object tro $\in T_k$ ($T_k \in \mathcal{T}$) is* l-related *to a reference object ro $\in T_t$ according to a foreign key path $p_i = \langle T_{i_1}, T_{i_2}, \ldots, T_{i_s}\rangle$ if and only if all the following conditions are satisfied:*

- $s = l + 1$;
- $T_{i_1} = T_t$;
- $T_{i_s} = T_k$;
- *there exists a task-relevant object newTro $\in T_{i_{s-1}}$ such that newTro is* $(l-1)$-related *to ro according to p_i;*
- *there exists a foreign key constraint fk from $T_{i_{s-1}}$ to T_s (or vice-versa) such that the foreign key of newTro assumes the same value of the primary key of tro (or vice-versa).*

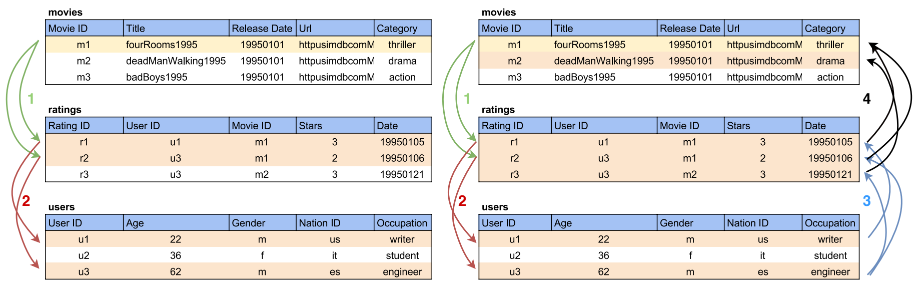
Def. 4. *A task-relevant object tro $\in T_k$ is* related *to a reference object ro $\in T_t$ according to a foreign key path p_i if and only if there exists $l \geq 1$ such that tro is l-related to ro according to p_i. We indicate as $R(ro, l, p_i)$, the set of task-relevant objects l-related to the reference object ro according to p_i, and as $R(ro, p_i)$ the set of all the task-relevant objects related to the reference object ro according to p_i, for any $l \geq 1$.*

In Fig. 2, we report a small example of a database instance, which refers to the schema of Fig. 1. From this figure, it is possible to identify the task-relevant objects for two foreign key paths, namely $\langle$`Movies`, `Ratings`, `Users`$\rangle$ (on the left) and $\langle$`Movies`, `Ratings`, `Users`, `Ratings`, `Movies`$\rangle$ (on the right). Following a path, it is possible to navigate the objects and their features involved when analyzing a specific reference object.

Example 3. *Considering the specific movie m1, the objects and their features considered by following the foreign key path $\langle$`Movies`, `Ratings`, `Users`$\rangle$, expressed in the logical formalism, are:*

```
movies(m1),
   movies_title(m1, fourRooms1995),
   movies_releaseDate(m1, 19950101),
   movies_url(m1, httpusimdbcomMtitle-exactFour20Rooms201995),
   movies_category(m1,thriller),
ratings(r1),
   ratings_userID(r1, u1),
   ratings_movieID(r1, m1),
   ratings_stars(r1, 3),
   ratings_date(r1, 19950105),
ratings(r2),
   ratings_userID(r2, u3),
   ratings_movieID(r2, m1),
   ratings_stars(r2, 2),
   ratings_date(r2, 19950106),
users(u1),
   users_age(u1, 22),
   users_nationID(u1, us),
   users_gender(u1, m),
   users_occupation(u1, writer),
users(u3),
   users_age(u1, 62),
   users_nationID(u1, es),
   users_gender(u2, m),
   users_occupation(u1, engineer).
```

From this example, it is possible to see that, in the logical formalism, we have two types of predicates: *i)* unary predicates (e.g., `movies(·)`, and `ratings(·)`), used to identify a row (object) in a table (e.g., the predicate `movies(·)` is used to instantiate a row in the table *movies*); *ii)* binary predicates (e.g., `movies_title(·,·)` and ratings_userID(·,·)), used to instantiate attribute values (e.g., `movies_title(m1, fourRooms1995)` is used to instantiate the attribute *title* of the table *movies*, for the row *m1*). It is noteworthy that binary predicates, by means of variables, are also able to implicitly instantiate relationships (e.g. user *u1* gave three stars to the movie "fourRooms1995" in the rating *r1*).
Finally, since our method is strongly based on the concept of aggregates, as introduced in Section 1, in the following we formally define them.



movies

Movie ID	Title	Release Date	Url	Category
m1	fourRooms1995	19950101	httpusimdbcomM	thriller
m2	deadManWalking1995	19950101	httpusimdbcomM	drama
m3	badBoys1995	19950101	httpusimdbcomM	action

ratings

Rating ID	User ID	Movie ID	Stars	Date
r1	u1	m1	3	19950105
r2	u3	m1	2	19950106
r3	u3	m2	3	19950121

users

User ID	Age	Gender	Nation ID	Occupation
u1	22	m	us	writer
u2	36	f	it	student
u3	62	m	es	engineer

movies

Movie ID	Title	Release Date	Url	Category
m1	fourRooms1995	19950101	httpusimdbcomM	thriller
m2	deadManWalking1995	19950101	httpusimdbcomM	drama
m3	badBoys1995	19950101	httpusimdbcomM	action

ratings

Rating ID	User ID	Movie ID	Stars	Date
r1	u1	m1	3	19950105
r2	u3	m1	2	19950106
r3	u3	m2	3	19950121

users

User ID	Age	Gender	Nation ID	Occupation
u1	22	m	us	writer
u2	36	f	it	student
u3	62	m	es	engineer

Figure 2: An example of a database instance, together with two possible foreign key paths: $\langle \texttt{Movies}, \texttt{Ratings}, \texttt{Users} \rangle$ (left) and $\langle \texttt{Movies}, \texttt{Ratings}, \texttt{Users}, \texttt{Ratings}, \texttt{Movies} \rangle$ (right). In both cases, the reference object *m1* is emphasized in light orange, whereas the corresponding task-relevant objects that are found in two (left), respectively four steps (right), are emphasized in pink.

Def. 5. *Let $p_i = \langle T_{i_1}, T_{i_2}, \ldots, T_{i_s} \rangle$ be a foreign key path of length s, X be a generic attribute of the table T_{i_s}, and $agg = [aggr_1(), aggr_2(), \ldots, aggr_{s-1}()]$ be a list of aggregate functions, all compatible with the type*[2] *of X, where $aggr_j$ applies to the j-related objects. The aggregate of X with respect to the reference object ro and the path p_i is defined as:*

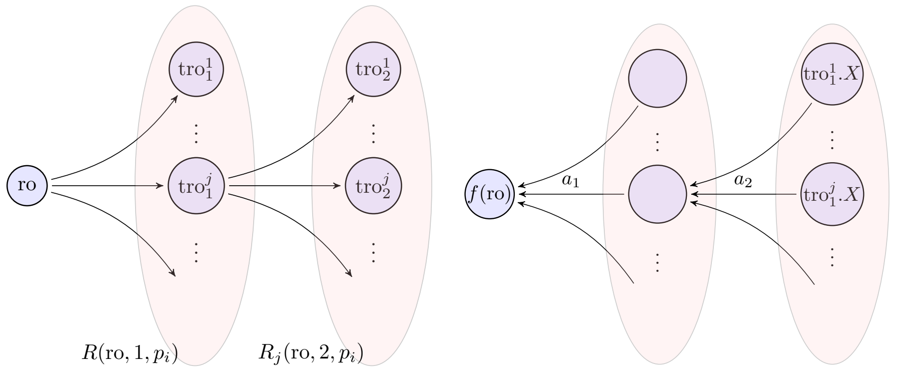
$$A(ro, p_i, agg, X) = \underset{tro_1 \in R(ro,1,p_i)}{\mathrm{aggr}_1} \left(\underset{\substack{tro_2 \in R(ro,2,p_i),\\ tro_2 \frown tro_1}}{\mathrm{aggr}_2} \left(\ldots \underset{\substack{tro_{s-1} \in R(ro,s\text{-}1,p_i),\\ tro_{s-1} \frown tro_{s-2}}}{\mathrm{aggr}_{s\text{-}1}} (tro_{s-1}.X) \ldots \right)\right) \tag{1}$$

where $tro_j \frown tro_{j-1}$ indicates that there is a foreign key between the two objects tro_j and tro_{j-1}.

It is noteworthy that aggregation functions may also be applied to one single value, in the cases of many-to-one or one-to-one relationships from the target table, while multiple values are actually aggregated into one value in the case of one-to-many relationships.

Using these definitions, in the following section we describe the proposed method RE3PY, which learns a classification function ϕ_t.

[2] *Average*, *minimum*, *maximum*, *count*, *countDistinct* for numeric types; *mode*, *count* and *countDistinct* for categorical types.



(a) Finding task-relevant objects (b) Aggregating the values

Figure 3: Two-step construction of a feature f in the internal nodes of a tree. In the example, the the considered foreign-key path p_i contains three tables. (a) Identification of all 1-related (i.e., $R(ro, 1, p_i)$) and 2-related (i.e., $R(ro, 2, p_i)$) task-relevant objects following p_i. (b) Aggregation of the value of the feature X, back to the target object.

4. The proposed method RE3PY

In this section, we formally introduce the proposed method. We start with the language bias and the role of aggregates, then we describe the tree induction and finally we discuss the proposed ensemble approaches.

4.1. Language Bias and Aggregates

The core characteristic of our method is the on-the-fly construction of new features according to Eq. (1), that are considered as candidate splits by our algorithm for the top-down induction of decision trees (TDIDT) [6]. Such new features are constructed by navigating the foreign key paths, as shown in Example 3.

Algorithmically, given a foreign-key path and a reference object ro, we start navigating from the target table. Considering the database instance in Fig. 2 and the path ⟨`movies, ratings, users`⟩, the first step is to identify all 1-related task-relevant objects, e.g., all the ratings provided to a given movie. For example, for the movie *m1*, we identify the ratings *r1* and *r2*. Then, for each of the 1-related task-relevant objects, the walk continues towards the 2-related task relevant objects, namely the users *u1* (accessed from *r1*) and *u3* (accessed from *r2*). The general schema of finding the task-relevant objects is shown in Fig. 3a.

The features of the traversed objects are aggregated back into a single value through a chain of aggregation functions, as shown in Eq. (1). In other words, each $(l-1)$-related object is replaced by the aggregated value obtained by applying the function aggr_l to the feature under consideration of the corresponding group of l-related objects (see Fig. 3b).

The possible foreign key paths depend on the dataset at hand, while the possible aggregation functions depend on the type of properties, namely:

- average (avg), minimum (min), maximum (max), count and countDistinct, for numeric types;
- mode, count, countDistinct, for categorical types.

Any combination of foreign-key path, feature of the corresponding s-related objects (where s is the length of the path), and list of aggregation functions, defines a candidate feature f, i.e., a mapping $ro \mapsto f(ro)$.

Example 4. *The path* $p_i = \langle \texttt{movies, ratings, users} \rangle$*, the feature Age of the table users, and the list of aggregates* $agg = [avg, avg]$ *represent the feature* f *corresponding to the average age of a user who rated a given movie. More formally:*

$$f(ro) = A(ro, p_i, agg, Age) = \text{avg}_{tro_1 \in R(ro,1,p_i)}(\text{avg}_{\substack{tro_2 \in R(ro,2,p_i), \\ tro_2 \frown tro_1}} (tro_2.Age))$$

In this particular case, $R(ro, 1, p_i)$ *is the set of ratings of a given movie ro, and* $R(ro, 2, p_i)$ *is the set of users who provided the ratings.*

Note that the *existence* of related task-relevant objects, e.g., the existence of some ratings associated to a given movie (that is the only aspect evaluated by approaches like BoostSRL [19]), can be evaluated by adopting a *count* aggregation function and the test $f(ro) > 0$.

4.2. Single Tree Induction

Our classification method is based on the top-down induction of decision trees (TDIDT). In particular, both original features associated to the target table and those generated through aggregates, as shown in the previous section, are considered for the evaluation of possible splits in the internal nodes of the tree.

Algorithm 1 *Tree*(examples E, current foreign-key path p)

```
1: $(h^*, f^*, S^*) = (0, none, none)$      # best quality, feature and splitting set
2: for all valid pairs $(f, S)$, according to $E$ and $p$ do
3:     if $h(E, f, S) > h^*$ then
4:         $(h^*, f^*, S^*) = (h(E, f, S), f, S)$
5: if $f^* = none$ then
6:     return $Leaf$(majority_class($E$))
7: else
8:     $E_+, E_- = splitToPositiveNegative(E, ro \mapsto f^*(ro) \in S^*)$
9:     $tree_+ = Tree(E_+, path(f^*))$
10:    $tree_- = Tree(E_-, path(f^*))$
11:    return $Internal(ro \mapsto f^*(ro) \in S^*, tree_+, tree_-)$
```

Tree induction starts with the call $Tree(E, \emptyset)$, where E is the set of all the examples of the target table. The pseudocode of the procedure $Tree(\cdot, \cdot)$ is described in Algorithm 1, which describes how the decision tree is actually induced.

First, all valid tests in the form "f(ro) $\in S$" are generated, based on the possible features f and domain subsets S. A test is not valid if any of the following stopping criteria applies: *i)* the maximum depth of tree is reached, *ii)* the heuristic score of the test is too low, *iii)* the set of examples falling in the positive or negative branch is too small.

Then, the best combination (f^*, S^*) is selected, according to a given heuristic h (line 2) that, in this work, corresponds to the GINI index [6]. If the set of candidate tests is not empty, the optimal test is of the form $f^*(ro) \in S^*$. If f is a numeric feature, then $S^* = (\vartheta, \infty)$ or $(-\infty, \vartheta]$, for an appropriate ϑ. Otherwise (i.e., when f^* is nominal), S^* is a subset of the domain of the attribute.

In Algorithm 1, p represents the current foreign key path. For the identification of valid features f during the first call, p contains only the target table. From p, the algorithm can define a new path that starts from it, and is able to perform up to ℓ steps. The algorithm follows a depth-first search strategy and is able to perform backtracking, that is, it is able to consider any (non-empty) prefix of the already defined path p and move forward, up to at most ℓ steps. Note that ℓ defines the maximum look-ahead (i.e., depth) from the current foreign-key path and does not correspond to the maximum

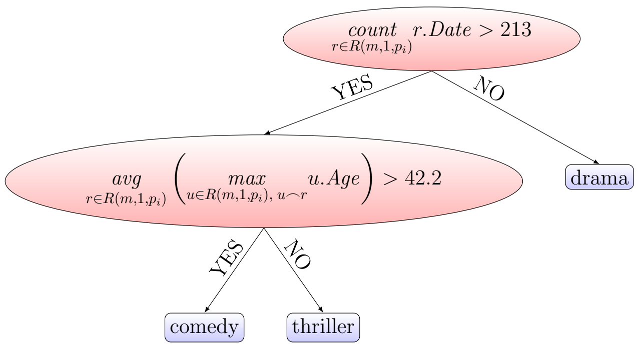

$\underset{r \in R(m,1,p_i)}{count}\ r.Date > 213$
YES
NO
$\underset{r \in R(m,1,p_i)}{avg}\left(\underset{u \in R(m,1,p_i),\ u \frown r}{max}\ u.Age\right) > 42.2$
drama
YES
NO
comedy
thriller

Figure 4: An example of tree built for the *Movies* dataset. In the root node, the reviews of the input movie m are counted (for technical reasons, we still need a predicate from the table *Ratings*, so the attribute Data was chosen). If the number of ratings is not greater than 213, the genre is predicted as *drama*. Otherwise, the average age of a user u who rated the movie m is computed (note that the age of a user is one single value and the aggregation function, *max* in this case, does not affect the result). If the average age is greater than 42.2, the predicted genre is *comedy*, otherwise it is *thriller*.

length of foreign-key paths.

If no valid test can be found, a leaf node with the majority class as the prediction is returned (line 6). Otherwise, the examples are split into positive (E_+) and negative (E_-), according to the identified test. In such a case, an internal node with the selected test is created, and the left and right children are created by recursively calling the algorithm $\mathit{Tree}(E_+, \mathit{path}(f^*))$ for the positive (left) branch and $\mathit{Tree}(E_-, \mathit{path}(f^*))$ for the negative (right) branch, where $\mathit{path}(f^*)$ is the foreign-key path of the selected test. An example of a relational tree built from the Movies dataset is shown in Fig. 4.

4.3. Relational ensembles in RE3PY

In the literature, one well-known approach to improve predictive performance of tree-based predictive models is that ensemble learning. Typical solutions are based, among the others, on Bagging [8], Random Forests [9] and Gradient Boosting [10].

The concept of a *bag* (a bootstrap replicate), used in Bagging and Random Forests, is different from that in the standard tabular setting. The main difference is that, in the relational setting, the objects are connected and random sampling would lead to lose relationships. In RE3PY, bootstrapping is performed only on the target table, while task-relevant objects are implicitly sampled according to their relationships with the sampled reference objects.

In addition to sampling the instances, Random Forests also sample a subset of features for each learned tree. In the specific relational setting considered by our system, possible features are represented by original features associated to the target table, or additional features computed by following a given foreign-key path and computing aggregates. Therefore, in RE3PY, we extend the ensemble approach of Random Forests by sampling random subsets of possible features from such an extended set of features. It is noteworthy that this is a cheap operation from a computational viewpoint. Indeed, for the purpose of selecting a sample, we simply enumerate the possible features without actually computing and evaluating their discriminatory power through the heuristic h. In other words, possible tests are actually evaluated only on sampled features.

Finally, boosting-based approaches come with their own randomization mechanisms. Similarly to Random Forests, a subset of examples and features is considered for each tree. The difference is that sampling happens without replacement up to a predefined sample size. Random sampling is not the only way adopted by boosting-based approaches to improve the predictive power of a single tree. More precisely, trees are built in a sequential manner where the first model ϕ_t^1 is built using the target values from the training data, whereas each subsequent model ϕ_t^{j+1}, $j \geq 1$, is built - loosely speaking - on the gradient of the error of the previous model [10]. The final model is then defined as:

$$\phi_t = \sum_j \phi_t^j \tag{2}$$

4.4. Feature ranking in the relational setting

With the increasing complexity of data under analysis, as well as of the application domains in which predictive models are adopted, the need to explain the output is becoming crucial, especially in specific sectors like medicine [11, 12, 13] and life sciences [46, 47, 48]. To explain black-box models, such as neural networks, typical approaches rely on permutation-based feature scores: the importance of a given feature is estimated by evaluating the relative increase of the error rate of the predictive model when the values of the feature are permuted.

This strategy cannot be directly applied in the relational setting, since the concept of permutation is not defined for a predicate. On the other hand, we recall that single trees are inherently interpretable, and can naturally be inspected to understand which aspects are influencing the predictions. This is not true for their ensemble counterparts that, although generally appear more accurate, lose this important characteristic. However, the tree induction mechanism allows us to define an alternative strategy to compute feature scores, that can be easily extended to ensembles thereof. In the following, we first define our approach to compute a feature score in the relational setting for single trees and then we extend it to ensembles.

Given a tree whose internal nodes contain tests based on predicates, as described in Sec. 4, the number of occurrences for a predicate may be considered the most straightforward way to evaluate the importance of that predicate. However, also the depth in the tree is important, since appearing close to the root means that the predicate has been selected earlier during the greedy search performed by the algorithm for TDIDT and, therefore, on the basis of a larger subset of examples (reference objects).

Therefore, an occurrence of a predicate in the test of an internal node should be weighted by the number of reference objects that reach the node. Such intuitive intuition would be the direct extension of the concept of *Symbolic* ranking [49] for the tabular data. However, it is not adequate in our case due to the look-ahead parameter $\ell \geq 1$, which makes a chosen predicate only a part of the feature constructed in the node. For example, a generic feature given in Eq. (1) consists of s parts: $s - 1$ hops between the tables, and the predicate on X. Although the condition expressed in a node $\mathcal{N}$ involves the set of features $f(\mathcal{N})$, only the last steps($\mathcal{N}$) of the condition are constructed in the node $\mathcal{N}$, where steps($\mathcal{N}$) ($\leq \ell$) represents the number of steps added in the node $\mathcal{N}$ of the tree but not present in its parent node.

Thus, the symbolic importance of the feature X for a tree τ can be formally defined as:

$$imp_{\mathrm{S}}(X) = \sum_{\mathcal{N} \in \tau} \mathbb{1}[X \in f(\mathcal{N})]/\mathrm{steps}(\mathcal{N}) \tag{3}$$

where $\mathbb{1}$ is the indicator function.

Symbolic ranking can appear coarse since the reward for a predicate that appears in a given node is always a discrete "jump" that does not depend on the actual quality of the split. This issue can be overcome by adopting the Genie3 score [14], that weights the contribution of each node, when computing the importance of a feature, on the basis of the quality of the split. Genie3 was originally designed for tabular data, but in the following we extend it to the relational setting. Formally, let $h^*(\mathcal{N})$ be the quality of the split performed at node $\mathcal{N}$ measured through the heuristic h. The importance of a predicate X measured through the Genie3 score extended to the relational setting is therefore computed as follows:

$$imp_{\text{Genie3}} = \sum_{\mathcal{N} \in \tau} \mathbb{1}[X \in f(\mathcal{N})] \; h^*(\mathcal{N})/\text{steps}(\mathcal{N}). \tag{4}$$

Note that, when data consist of only one table (containing the reference objects) both measures boil down to their original counterparts designed for tabular data.

The extension of such feature scores to the ensemble-based approaches is quite straightforward. In particular, since the trees of Random Forests and Bagging can be seen as independent realizations of the same random variable, we can simply compute the average of the feature score observed over the trees of the ensemble. On the other hand, the predictions of the trees in Gradient Boosting ensembles need to be summed according to equation (2). Coherently, the correct aggregation of per-tree scores feature rankings is also the sum. However, since $\sum_{i=1}^n x_i > \sum_{i=1}^n y_i \Leftrightarrow (\sum_{i=1}^n x_i)/n > (\sum_{i=1}^n y_i)/n$, for any numbers x_i and y_i, both averaging and summation lead to the same feature ranking.

Note that computing partial results, while building the set of trees, may reveal how the importance of a feature evolves with each additional tree. In Sec. 6.5, we will show how feature rankings computed through Bagging, Random Forests and Boosting converge with different running times.

4.5. Differences and advantages over TILDE and BoostSRL

As briefly introduced in Sec. 2, there have been several attempts in the literature to build classification models from relational data. Among existing approaches, TILDE and BoostSRL are the closest to RE3PY. Therefore, in the following we highlight the main differences and advantages of RE3PY, with respect to them.

First, RE3PY simultaneously exhibits the predictive performance of ensembles and the possibility to explain the predictions, thanks to its ability to produce a ranking of the features.

Second, both TILDE and BoostSRL can handle only nominal values, while RE3PY can also handle numeric attributes[3].

[3]Actually, the values may be also ordinal, since the trees are invariant to the

Third, aggregates are not fully exploited by TILDE and BoostSRL. In particular, BoostSRL is only based on the existence of related task-relevant objects. This means that it is able to construct additional features that would correspond to a particular case of our *count* aggregation function (with $\vartheta = 0$). On the other hand, TILDE also uses additional aggregates like RE3PY, but in a different way. In particular, RE3PY uses a sequence of aggregates to compute complex aggregates on the values of variables introduced at the last step of the foreign-key path, while the intermediate objects are only used to reach the last object. On the contrary, TILDE uses the intermediate steps for constraining the sets of task-relevant objects. For example, it can produce[4] tests like *Is the average age of users that have provided at least 5 ratings in total, and have rated a given movie, greater than 60?*, but can not produce tests like *Is the average rate provided by the user that have provided at least 5 ratings in total and have rated a given movie, equal to five?*. One consequence of this aspect is that RE3PY is able to aggregate every variable multiple times (see the example in Figure 4), but TILDE is not. For the exact formulation of the possible tests in TILDE, we refer reader to [50].

Fourth, more operationally speaking, RE3PY, available as a Python package[5], is able to build models based on single trees, Random Forests, Bagging and Gradient Boosting ensembles. On the contrary, with BoostSRL, it is possible to build only single trees and Gradient Boosting ensembles. Using TILDE, we could not find any available implementations that allow for the induction of ensembles.

4.6. Computational Complexity

In this section, we estimate the computational complexity of RE3PY, starting from that of single trees and then extending it to ensembles thereof. In the following, we work under the standard assumption that the induced trees are balanced, i.e., their depth is $\mathcal{O}(n)$, where n is the number of reference objects (i.e., rows in the target table). Moreover, we introduce the notation for the following upper bounds. Let:

- F be denote the the maximal number of features in the internal nodes;

monotonous transformation of the attributes. Thus, only the order is important and not the actual values.

[4]By combining test in a path from the root to the leaf of a decision tree.

[5]`https://pypi.org/project/re3py/`

- s be the maximum length of foreign-key paths;
- b be the branching factor, i.e., the maximum number of 1-related objects for a given object.

According to such assumptions, we can derive that the time complexity of growing a single tree is $\mathcal{O}(Fnb^s \log n)$. A thorough derivation of the formula is given in Appendix A.

In the case of ensembles, the time complexity can be estimated as follows: bagging boils down to growing T trees, so the time complexity corresponds to T times the time complexity of a single tree. However, T is a constant that does not asymptotically affect the complexity. Moreover, trees are grown independently and, as clarified by Breiman [8], *bagging is almost a dream procedure for parallel computing.* Thus a bagging ensemble can be grown in the same time as a single tree, provided we have enough computing units.

Similarly holds for Random Forests: sequential induction is T times more expensive, but the procedure can be easily parallelized. Moreover, the algorithm considers only $F' \leq F$ features at each internal node of a tree. Typical values for F' are $F' = \lceil \sqrt{F} \rceil$ or $F' = \lceil \log_2 F \rceil$ which can result in substantial speed-ups when the number of features F is large.

Finally, for Gradient Boosting, trees are not grown independently, since the target values are being updated according to the gradients of the predictive errors of the ensemble built so far. However, similar to Random Forests, they can focus on a subset of features and a subset of examples. Moreover, in contrast to the trees with Bagging and Random Forests, which are fully grown (motivated by the bias-variance decomposition of the error [8]), boosted trees are typically prepruned according to a maximum depth and/or a minimum amount of examples per leaf.

5. Experimental evaluation

In this section, we describe the experiments we performed to evaluate the performance of the proposed method. In particular, we will investigate the following research questions:

Q1 Does the considered relational setting provide advantages over a classical propositional counterpart that only focuses on the target table?

Q2 Does the exploitation of the proposed aggregates provide advantages over the exploitation of only existentially-qualified features?

- *Q3* Do the ensembles provide further advantages over single trees? If yes, which ensemble strategy performs best?
- *Q4* How does the proposed method perform with respect to state-of-the-art competitors?
- *Q5* Does the feature ranking provide real clues on the importance of the features?
- *Q6* How is the proposed method sensitive to its input parameters?

In order to answer to the research questions *Q2* and *Q1*, we compared the obtained results with those achieved with two different settings, namely, considering only existential qualifiers (to answer to research question *Q2*), and discarding all the tables other than the target tables (to answer to research question *Q1*). Moreover, to properly answer to the research question *Q3*, we evaluated the performance of our system with the three different ensemble approaches introduced in Section 4.3, namely bagging, boosting and random forests, and compared the results with those achieved when learning a single decision tree.

All the experiments have been performed through a 10-fold cross validation and the results were measured in terms of classification accuracy.

In the following subsections, we first describe the adopted datasets and the considered competitor systems. Finally, we report the results, together with some statistical tests, and discuss them with respect to the each research question.

5.1. Datasets

In the following, we briefly describe the datasets considered in our experiments, while in Table 1, we report their quantitative characteristics.

- **BASKET[6].** This dataset stores the statistics of basketball players of NBA and ABA collected during the 2004 and 2005. It includes data regarding the regular season, playoff and all star games, as well as data related to coaches, players and drafts. The target table is *teams* and the target attribute is *league*, which values can be *national* or *american* (see Figure 5).

[6]http://www.cs.cmu.edu/˜awm/10701/project/data.html

- **IMDB[7].** This dataset is an extension of the MovieLens10M dataset, published by GroupLens. In particular, it includes data from the Movielens dataset, data about the pages from Internet Movie Database (IMDb) and reviews from Rotten Tomatoes. We kept only the users with both rating and tagging information. The target table is *movies*, and the target attribute is *genre*, which values can be *comedy*, *thriller*, *drama* or *action* (see Figure 6).
- **MOVIE.** This dataset is built from MovieLens100k[8] and contains the ratings assigned to movies by users, collected through the *movielens* recommender system. The target table is *movies*, and the target attribute is *category*, which values can be *comedy*, *thriller*, *drama* or *action* (see Figure 7).
- **STACK[9].** This dataset is an anonymized dump of the Stack Exchange network. We considered the data about the Stack Overflow website, that consist of users, comments, posts, votes, history and links. The target table is *posts*, and the target attribute is *posttypeid*, which values can be *1(Questions)*, *2(Answers)*, *3(Wiki)*, *4(TagWikiExcerpt)* and *5(TagWiki)* (see Figure 8).
- **UWCSE[10].** This dataset contains data about the Department of Computer Science and Engineering of the University of Washington, including faculty members, projects, publications and the courses they teach. Moreover, it also contains data describing the relationships among faculty members (professors, assistant professor, students' adviser, etc.). The target table is *courses* and the target attribute is *discipline*, which values can be *graphics*, *theory*, *ai*, *language* or *systems* (see Figure 9).
- **YELP[11].** This dataset contains data related to Yelp, that is a website collecting reviews and comments about business activities. Data include businesses and their characteristics, users, check-in information, friendship relationships, tips and reviews. The target table is *business*

[7]http://grouplens.org/datasets/hetrec-2011/
[8]http://grouplens.org/datasets/movielens/
[9]https://archive.org/details/stackexchange
[10]http://alchemy.cs.washington.edu/data/uw-cse/
[11]http://www.yelp.com/dataset_challenge

Table 1: Quantitative characteristics of the considered datasets. *Predicates* represents the total number of predicates summed over all the tables in the dataset; *Target facts* is number of reference objects (i.e., rows in the target table); *Descriptive facts* corresponds to the number of facts describing object attributes and relationships through predicates.

Dataset	**Tables**	**Predicates**	**Descriptive Facts**	**Target Facts**	**Classes**
BASKET	9	118	630038	95	2
IMDB	21	57	614662	8816	4
MOVIE	5	16	183469	1422	4
STACK	7	52	383040	5855	5
UWCSE	12	15	1961	115	5
YELP	9	51	3348181	24959	4
WEBKB	6	9	36131	500	3
CARCINOGENESIS	6	12	64640	329	2
MUTAGENESIS	6	13	32942	188	2

and the target attribute is *category*, which values can be *Restaurants*, *Beauty & Spas*, *Health & Medical* and *Shopping* (see Figure 10).

- **WEBKB**[12]**.** This dataset contains the textual content, links and anchors of webpages collected by the World Wide Knowledge Base project of the CMU group. The target table is *pages* and the target attribute is *category*, which value is *student*, *faculty* and *course* (see Figure 11).
- **CARCINOGENESIS**[13]. This dataset contains data about molecules, e.g., their atoms, charge, etc., which are used to predict their carcinogenicity. The target table is *canc* and the target attribute is *class*, which value can be *1* (cancerous) or *0* (not cancerous) (see Figure 12).
- **MUTAGENESIS**[13]. Similarly to CARCINOGENESIS, this dataset contains data about molecules, e.g., their atoms, charge, etc., which are used to predict their mutagenicity. The target table is *drugs* and the target attribute is *active*, which value can be *1* (active) or *0* (not active) (see Figure 13).

[12]`www.cs.cmu.edu/afs/cs/project/theo-20/www/data/`
[13]`http://kt.ijs.si/janez_kranjc/ilp_datasets/`

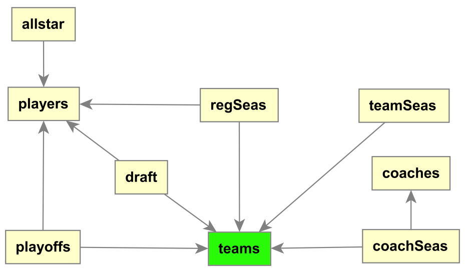


Figure 5: Schema of the dataset BASKET, showing the tables and the foreign keys. The target table is emphasized in green.

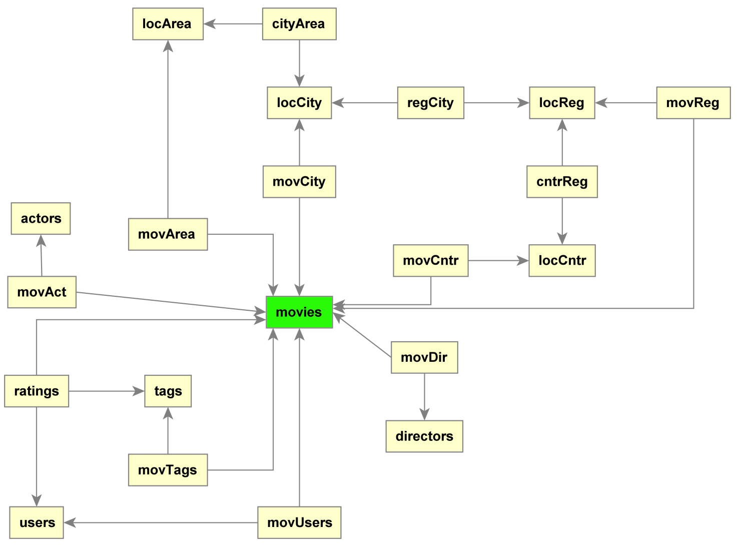


Figure 6: Schema of the dataset IMDB, showing the tables and the foreign keys. The target table is emphasized in green.

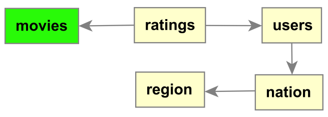


Figure 7: Schema of the dataset MOVIE, showing the tables and the foreign keys. The target table is emphasized in green.

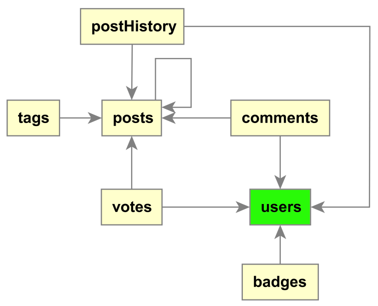


Figure 8: Schema of the dataset STACK, showing the tables and the foreign keys. The target table is emphasized in green.

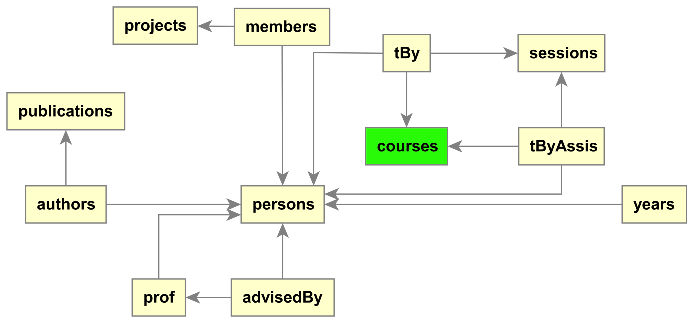


Figure 9: Schema of the dataset UWCSE, showing the tables and the foreign keys. The target table is emphasized in green.

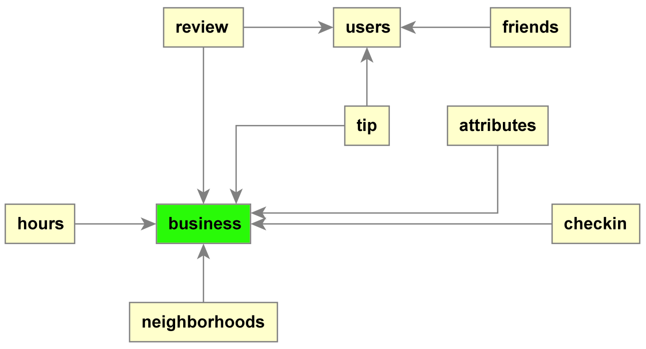


Figure 10: Schema of the dataset YELP, showing the tables and the foreign keys. The target table is emphasized in green.

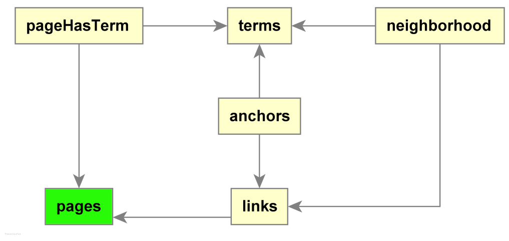


Figure 11: Schema of the dataset WEBKB, showing the tables and the foreign keys. The target table is emphasized in green.

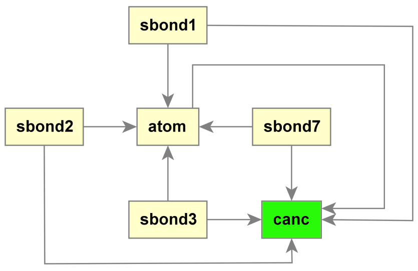


Figure 12: Schema of the dataset CARCINOGENESIS, showing the tables and the foreign keys. The target table is emphasized in green.

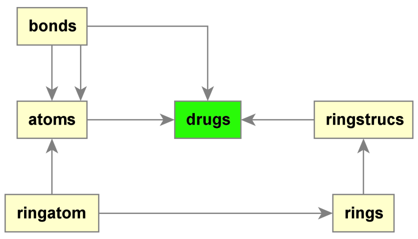


Figure 13: Schema of the dataset MUTAGENESIS, showing the tables and the foreign keys. The target table is emphasized in green.

5.2. Competitor Approaches

For comparison purposes, we compare the results of our approach with 8 state-of-the-art methods that are able to solve the classification task in a relational setting, namely:

- **RelIBk** [25], that is the relational variant, available in RelWeka, of the well-known k-nearest neighbors algorithm. As distance measure, we adopted the Relational Instance-Based Learning (RIBL) measure [26].
- **RelSMO** [25], that is the relational variant, available in RelWeka, of the Platt's Sequential Minimal Optimization algorithm [51]. This algorithm is based on kernel Support Vector Machines and adopts the Minkowski RIBL set distance [27].
- **GNetMine** [37], that is a graph-based regularization framework that works in the transductive setting. It explicitly aims at preserving the consistency over each relation and each link when assigning the label to unlabeled instances.
- **MrSBC** [21][14], that is a method that induces a set of first-order rules from the tables of the relational schema and exploits a naïve Bayes classification method to classify unlabeled instances.
- **ST-MrSBC** (Self-Training MrSBC) [24][15], that is a method based on MrSBC, that can capture autocorrelation phenomena by resorting to a variant of the self-training method. It bases its predictions on the ensemble of models built over all the self-training iterations.
- **MT-MrSBC** (Multi-Type MrSBC) [24][15], that is a method based on MrSBC that iteratively analyzes instances of multiple relations, in order to predict unknown labels in multiple target tables, in order to learn multiple classification functions simultaneously. It builds an ensemble of classifiers for each relation, and it is able not only to exploit autocorrelation phenomena, but also dependencies among the models built

[14]`www.di.uniba.it/~ceci/micFiles/systems/MURENA.html`

[15]`https://figshare.com/articles/Ensemble_MT-MrSBC_and_Ensemble_ST-MrSBC_systems/4334048/7`

for different target tables. Here we consider both its variants, namely **LexicographicMT-MrSBC**, that analyzes the relations in a predefined, lexicographic order, and **RandomMT-MrSBC**, that analyzes the relations in a random order at each iteration.

- **HENPC** [45][16], that is a method that exploits the predictive clustering framework on heterogeneous networks represented as relational databases. In particular, it extracts a hierarchy of heterogeneous clusters and exploit it for classification purposes in a transductive setting.
- **BoostSRL** (Boosting for statistical relational learning) [19] is a generalization of gradient boosting method towards relational data. As introduced in Sec. 2, the tests in the trees are roughly similar to those of our system when no aggregation is used (only existential qualifiers).

5.3. Parameter setting

As regards the parameters of RE3PY, we set $\ell = 2$ and the number of trees equal to 50 for its ensemble-based variants. For Bagging, no additional parameters are necessary since the bias-variance decomposition suggests the induction of fully-grown trees [8]. As for Random Forests, we set the number of features to sample for each tree equal to the square root of the number of features, as suggested in [9]. As for the Gradient Boosting and single trees, we performed a grid search via internal 3-fold cross-validation. In particular, for Gradient Boosting, we searched over *shrinkage* $\in \{0.05, 0.2, 0.4, 0.6\}$, *chosen_examples* $\in \{0.6, 0.8, 1.0\}$ (i.e., the proportion of the examples that are randomly chosen for a tree induction), $chosen\_features \in \{0.2F, 0.4F,\ 0.6F, 0.8F, F, \sqrt{F}\}$ (where F is the total number of features), and *depth* $\in \{2, 4, 6, 8\}$. For single trees, we searched over *leaf_size* $\in \{1, 5, 10, 15, 20\}$ (minimum number of examples per leaf) and *impurity* $\in \{0, 0.01, 0.02, 0.05, 0.1, 0.2\}$ (minimal relative decrease of the impurity in a split). Therefore, the configurations also include fully grown trees.

We adopted the same grid search approach for BoostSRL, but focusing only on *shrinkage* and *depth*, since the other parameters are not supported. Moreover, since BoostSRL cannot operate with numeric variables, we used a equal-width discretization into 10 bins wherever needed. As regards the other

[16]`http://www.di.uniba.it/~gianvitopio/systems/henpc/index.html`

competitors, we took their best results obtained in previous experiments on the same datasets from [24, 45].

6. Results

In Table 3 we show the results obtained by RE3PY considering the following dimensions of analysis:

i) the dataset;

ii) the considered relational setting/aggregates: all aggregates (**AGG-All**), existential aggregates only (**AGG-Exist**), target table only (**NO-Rel**), i.e., by ignoring task-relevant objects;

iii) the adopted ensemble strategy.

Starting from the presented results, in the following subsections we illustrate and discuss the different aspects covered by the research questions introduced in Sec. 5.

6.1. Q1 - Contribution of the Relational Setting

In this section we focus on evaluating the advantages of analyzing related data stored in additional tables (i.e., task-relevant objects and their relationships to reference objects) with respect to focusing only on the target table. This analysis allows us to evaluate the contribution provided, in general, by the relational setting. In Table 3, we can evaluate this aspect by comparing AGG-All and AGG-Exist with NO-Rel. From the results, we can see that our full system that exploits all the aggregates (AGG-All) outperforms the variant NO-Rel in 32 out of 36 cases (i.e., in all the cases except for the combinations single trees and random forests on CARCINOGENESIS, and Boosting on WEBKB that did not finish within two days).

Intuitively, the obtained results are reasonable. For example, the only descriptive facts of the Movies table of the MOVIE dataset are the release date, the url and the title. Clearly, they cannot reveal much about the category of a movie. On the other hand, knowing something about the users who provided some ratings on the movies (e.g., their gender or age) can help. The general superiority of the variant AGG-All over AGG-All is also confirmed by a Wilcoxon test, that revealed that the difference is statistically significant (see Table 2 - AGG-All vs NO-Rel).

Table 2: Results of the Wilcoxon test at $\alpha = 0.05$. P-values are corrected , with False Discovery Rate correction.

AGG-All vs **NO-Rel**	Winner	p-value
Single trees	AGG-All	$2.09 \cdot 10^{-2}$
RF	AGG-All	$1.52 \cdot 10^{-2}$
Boosting	AGG-All	$1.10 \cdot 10^{-1}$
Bagging	AGG-All	$7.69 \cdot 10^{-3}$

AGG-Exist vs **NO-Rel**	Winner	p-value
Single trees	AGG-Exist	$8.59 \cdot 10^{-1}$
RF	AGG-Exist	$7.22 \cdot 10^{-1}$
Boosting	AGG-Exist	$6.36 \cdot 10^{-1}$
Bagging	AGG-Exist	$6.78 \cdot 10^{-1}$

AGG-All vs **AGG-Exist**	Winner	p-value
Single trees	AGG-All	$2.60 \cdot 10^{-1}$
RF	AGG-All	$1.10 \cdot 10^{-1}$
Boosting	AGG-All	$7.69 \cdot 10^{-3}$
Bagging	AGG-All	$7.69 \cdot 10^{-3}$

On the other hand, the variant exploiting only existential qualifiers (AGG-Exist) outperforms NO-Rel in only 17 cases out of 36. Coherently, the Wilcoxon test showed that AGG-Exist generally performs better than NO-Rel, but the difference does not appear to be statistically significant (see Table 2 - AGG-Exist vs NO-Rel).

Overall, these results confirm that the exploitation of additional information conveyed by task-relevant objects is clearly beneficial, but that it may introduce noise if not properly handled. In particular, the adoption of only existential qualifiers appears to be not enough and can, on the contrary, negatively affect the results in several situations. This becomes clear looking at the Avg. Rank column in Table 3.

6.2. Q2 - Contribution of the Aggregates

In this subsection, we focus on evaluating the advantages of using aggregates in the splits of the trees (the possible aggregates that we used are listed in Sec. 4.1). In Table 3, we can evaluate this aspect by comparing AGG-All with AGG-Exist.

Table 3: Accuracy measured for RE3PY on all the datasets, with different relational setting/aggregates and with different ensemble strategies. The last column denotes the average rank measured for the given relational setting/aggregates on the specific dataset (the best result is emphasized in bold).

		Single trees	RF	Boosting	Bagging	Avg. Rank
BASKET	AGG-All	0.968	0.958	0.979	0.979	**1.5**
	AGG-Exist	0.979	0.968	0.968	0.968	**1.5**
	NO-Rel	0.695	0.695	0.695	0.695	3
IMDB	AGG-All	0.587	0.612	0.632	0.628	**1**
	AGG-Exist	0.578	0.576	0.577	0.574	3
	NO-Rel	0.586	0.610	0.625	0.616	2
MOVIE	AGG-All	0.517	0.556	0.568	0.567	**1**
	AGG-Exist	0.510	0.510	0.510	0.510	2
	NO-Rel	0.508	0.495	0.507	0.493	3
STACK	AGG-All	0.963	0.950	0.951	0.951	**1.13**
	AGG-Exist	0.950	0.950	0.950	0.950	1.87
	NO-Rel	0.801	0.801	0.801	0.801	3
UWCSE	AGG-All	0.826	0.826	0.835	0.861	1
	AGG-Exist	0.200	0.209	0.191	0.200	3
	NO-Rel	0.252	0.252	0.243	0.252	2
YELP	AGG-All	0.670	0.779	0.879	0.855	**1**
	AGG-Exist	0.588	0.587	0.579	0.588	2.75
	NO-Rel	0.596	0.599	0.574	0.599	2.25
WEBKB	AGG-All	0.924	0.814	NA	0.932	2.25
	AGG-Exist	0.966	0.968	NA	0.970	**1.5**
	NO-Rel	0.456	0.388	0.488	0.430	2.5
CARCINO GENESIS	AGG-All	0.523	0.547	0.562	0.562	**1.25**
	AGG-Exist	0.535	0.538	0.550	0.538	2.25
	NO-Rel	0.553	0.553	0.553	0.553	2.5
MUTA GENESIS	AGG-All	0.910	0.920	0.872	0.904	**1.25**
	AGG-Exist	0.665	0.665	0.665	0.665	3
	NO-Rel	0.856	0.888	0.878	0.867	1.75

As anticipated in the previous subsection, using aggregates and, therefore, a richer set of possible splits, turns out to be beneficial. Indeed, AGG-All outperforms AGG-Exist in 30 out of 36 cases (i.e., all the combinations except single trees and random forests induced from BASKET and WEBKB, Bagging induced from WEBKB, and single trees induced from CARCINOGENESIS). These results are also confirmed by the Wilcoxon test, that shows that the difference among them is statistically significant (see Table 2 - AGG-All vs AGG-Exist).

A closer analysis of the results revealed that in the BASKET dataset many trees show the test on `coach_seas_team`, *count* as aggregation and the threshold 0 in their root. This means that, in terms of the adopted heuristic, other (more complex) aggregates, possibly exploitable by AGG-All, did not provide a relevant contribution. This situation was also observed for WEBKB and CARCINOGENESIS.

On the contrary, on UWCSE, we noticed a significant improvement. In this dataset, the goal is to determine what is the discipline a professor is teaching. By looking at the models, we noticed that one of the splits identified by AGG-All that filters out 18 professors in the `ai` field, is based on the number of persons who had already taught this course, and checking if such a number is less than 5. In this case, the adoption of only existential quantifiers does not provide much information, since every course is taught by at least one professor, and such a discriminative pattern could not be exploited.

6.3. Q3 - Effect of the ensemble

In this subsection, we evaluate the possible contribution provided by the adoption of relational ensemble approaches, with respect to inducing one single decision tree.

The bias-variance decomposition of the predictive error for Bagging and Random Forests [8] shows that they are expected to exhibit better performances than a single tree, since their bias is the same and the variance is lower. A similar advantage has been (mostly empirically) shown for gradient boosting [52]. In the case of relational data, the derivation of the decomposition is the same. Therefore, the adoption of Bagging or Random Forests is generally advisable. As for Boosting, growing many trees might be computationally too expensive, especially since they cannot be grown in parallel (see the missing results for the WEBKB dataset in Table 3).

Looking at the results of our experiments reported in Table 3 in a columnwise fashion, we now discuss whether such observations also apply in our relational setting with the considered datasets.

Comparing **Bagging** with single trees, the only difference would be the bootstrapping of the training set. Therefore, the performance of Bagging should never be substantially worse than that of single trees, provided that the number of examples in the training set is not too small. Looking at the results, we can observe that Bagging wins in 13 cases out of 27, with 11 ties. In the 3 cases in which single trees perform better, the difference is always almost negligible (see, for example, 1% in the STACK dataset). On the contrary, the improvement observed in some situations with the adoption of Bagging may be substantial (see, for example, 19% on the YELP dataset).

As regards **Random Forests**, which additionally sample the features, a possibly extremely large number of split candidates may lead to some performance reduction if the sample size is too small and the majority of the possible splits is irrelevant. This effect theoretically vanishes with an infinite number of trees, but in practice the number of trees is finite. The experimental results show that, in comparison with single trees, the performances of Random Forests are similar to those of Bagging. In particular, we observe 10 wins, 10 ties and 7 loses. A relevant exception is the WEBKB dataset, where the number of possible splits is quite high, due to some nominal variables. As a result, considering $\sqrt{F}$ features per sample appears to be not enough, and a single tree leads 11% advantage in terms of accuracy.

Looking at the results obtained with **Boosting**, we can observe 10 wins and 6 ties with respect to single trees. Also in this case, we can see substantial performance improvements for the winning cases (see, for example, the results of the YELP dataset), and tiny differences when Boosting loses.

A general comparison among the different ensemble approaches is shown in Table 4. Here, we report the rank of each ensemble approach on each dataset, as well as the average rank.

Although the absolute differences are typically not large, Bagging and Boosting consistently outperform Random Forests, which are ranked in the last position in most of cases. The (average) ranks of the other two methods are pretty similar and, performance-wise, it is hard to prefer one approach over the other. However, it is noteworthy that every Boosting model is based on 288 internal cross-validations used for determining the best parameter setting. Moreover, as already emphasized, trees in Boosting cannot be induced in parallel. On the contrary, Bagging has no hyperparameters to tune, and it is easily parallelizable. Therefore, we can conclude that in our relational setting, although RE3PY offers all the available options, learning an ensemble based on Bagging should be generally preferred.

6.4. Q4 - Comparative evaluation

In this subsection, we compare the results obtained by RE3PY when exploiting all the aggregates (i.e., AGG-All) with Bagging, with those achieved by state-of-the-art methods in their best configurations (see [24, 45]). The results of such a comparison are shown in Table 5.

The comparison reveals that RE3PY outperforms all the competitors on 6 out of 7 datasets, and shows the second best result for the WEBKB dataset. It is noteworthy that, except for BoostSRL, the results shown for competitors can be considered over-optimistic, since they are the outcome of an aposteriori selection (on the testing set) of the best parameter configurations, and not that of an internal cross-validation (on the training set).

Table 4: The ranks of the AGG-All version of the ensembles, for the considered data sets. NA results in having rank 3

	RF	Bagging	Boosting
BASKET	3	1.5	1.5
IMDB	3	2	1
MOVIE	3	2	1
STACK	3	1.5	1.5
UWCSE	3	1	2
YELP	3	2	1
WEBKB	2	1	3
CARCINOGENESIS	3	1.5	1.5
MUTAGENESIS	1	2	3
Average Rank	2.67	1.61	1.72

When the above two columns are passed to Friedman's statistical test, it reveals that the differences are also statistically significant ($p = 0.0465$).

6.5. Q5 - Feature Ranking

In this section, we show how feature ranking can provide clues on an otherwise black-box model. Indeed, as introduced in Sec. 4.4, RE3PY includes a feature ranking approach for the relational setting that allows us to identify the contribution of predicates in the construction of ensemble-based models.

As an example, we take a detailed look at the YELP dataset (Fig. 14), for which we show the rankings for the two best performing ensemble approaches, i.e., Bagging and Boosting, and with AGG-All and AGG-Exist. We recall that, in this dataset, the goal is to learn a predictive model that discriminates between `Beauty and Spas`, `Health and Medical`, `Restaurants` and `Shopping` types of business.

For the Bagging-based models, the difference between AGG-All and AGG-Exist is relevant. Indeed, the importance of the features when using aggregates (Fig. 14a) converges quickly by increasing the number of trees, and the top-ranked predicates are clearly meaningful (see `businessReviewCount` - the number of reviews, or `businessStars` - the number of stars in a review). On the other hand, the importance of the features for the model learned without exploiting aggregates (Fig. 14b) does not converge quickly and the top-ranked attributes appear to be meaningless for the task at hand. Indeed, the `neighborhood` intuitively cannot determine the business type, nor can

Table 5: Best accuracy values of the models learned by RE3PY and its competitors. The best result for each dataset is shown in bold.

	MrSBC	ST MrSBC	MT MrSBC Lex	MT MrSBC Rand	HENPC	Rel Weka SMO	Rel Weka IBk	Boost SRL	RE3PY
BASKET	0.294	0.264	0.716	0.716	0.664	0.959	0.959	0.706	**0.979**
IMDB	0.612	0.586	0.588	0.585	0.599	0.554	0.581	0.570	**0.628**
MOVIE	0.433	0.469	0.467	0.462	0.562	0.510	0.510	0.510	**0.567**
STACK	0.810	0.758	0.755	0.754	0.597	0.127	0.876	0.428	**0.951**
UWCSE	0.274	0.289	0.289	0.289	0.545	0.205	0.152	0.230	**0.861**
YELP	0.460	0.581	0.527	0.581	0.628	0.593	0.538	0.253	**0.855**
WEBKB	0.585	0.570	0.570	0.570	**0.978**	0.612	0.606	0.504	0.932

the existence of the `closingHours`, since every business is located somewhere and every business has a closing time. This difference is also confirmed by the absolute values of the feature scores: they are much lower if the aggregates are not used, meaning that the variance reduction in splits is typically low and that the features cannot effectively discriminate among the classes.

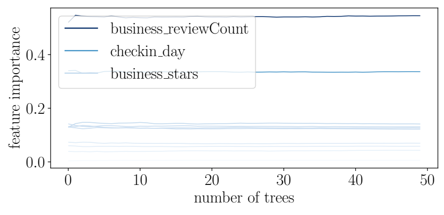
Similar conclusions can be drawn for the Boosting-based models (see Figures 14c and 14c). Note that although both versions of Boosting rank `hoursClose` among the top-3 (probably due to a random chance), AGG-All can exploit it more effectively thanks to the multiple possible aggregates that can be computed on it.

As for the other data sets, we show the Genie3 feature importance score computed from the Bagging-based models that exploits all the aggregates (AGG-All) in Fig. 15.

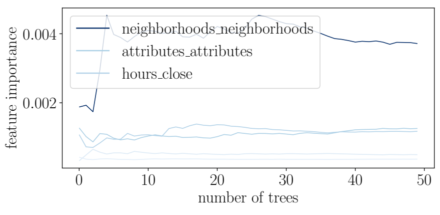
6.6. Q6 Parameter Sensitivity

In this subsection, we aim to answer to the last research question, related to the sensitivity of RE3PY to its parameters. Since Bagging is a special case of Random Forests, whose only additional parameter is the size of the subset of the features that is considered in internal splits and, in turn, it is also one of the parameters of Boosting, in the following we only show a detailed analysis for Boosting ensembles.

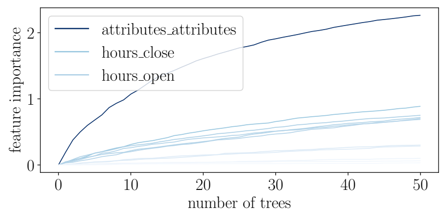
As described in Sec. 5.3, the analyzed parameters are: depth of the tree, (proportion of) sampled features, (proportion of) sampled examples, and shrinkage. In order to analyze the influence of a specific parameter, we



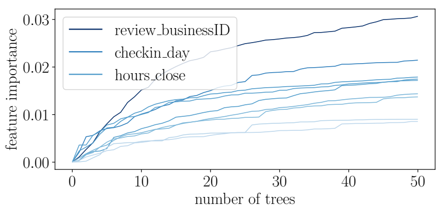
(a) Bagging - AGG-All



(b) Bagging - AGG-Exist



(c) Boosting - AGG-All



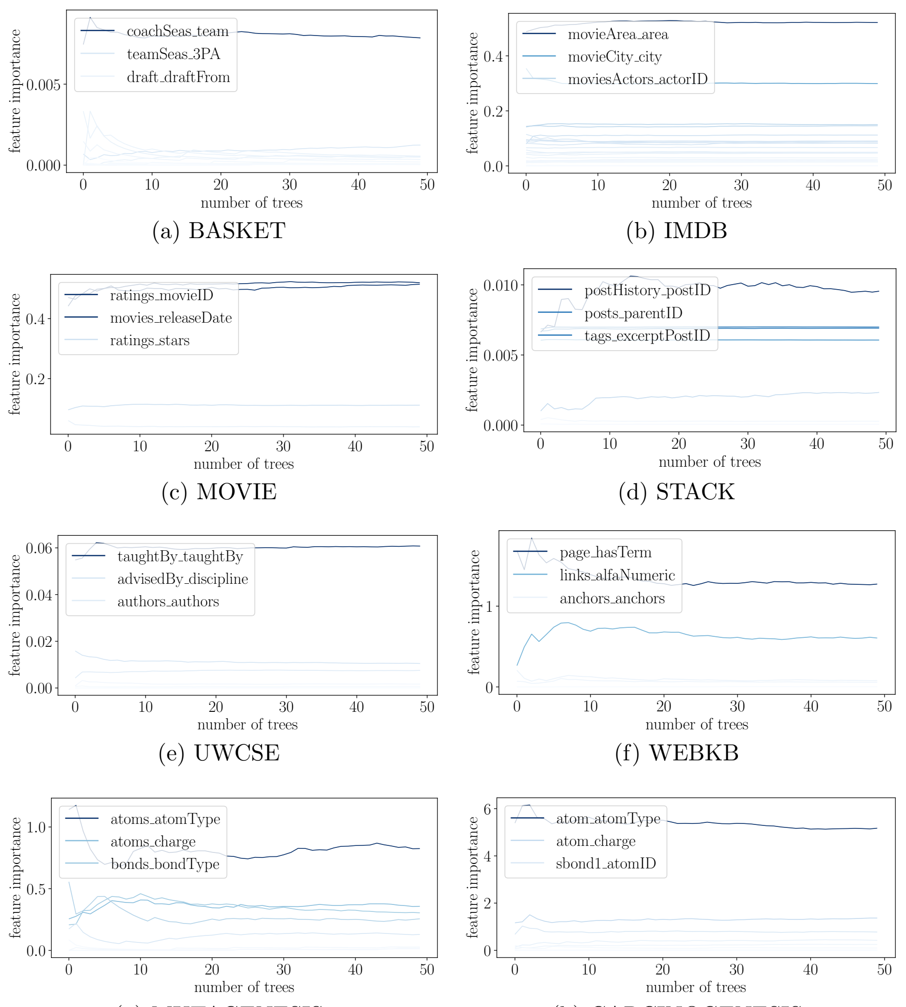
(d) Boosting - AGG-Exist

Figure 14: Feature importance scores for the YELP data set, for different ensemble configurations. The curve for a given feature relates the measured Genie3 feature importance scores with the number of induced trees in the ensemble.

estimate the distribution of the accuracy values when such a parameter is fixed to a chosen value and the other parameters vary, through the Python module *seaborn*[17]. The bigger the differences among the distributions, the larger the influence of the parameter.

In Fig. 16, we show the result of the analysis on the UWCSE dataset, that mostly summarizes the results observed on all the datasets. The depth of the trees typically does not play a major role, provided that it is higher than 2 (see Fig. 16a). As regards the number of sampled features (see Fig. 16b), it is clear that taking $\sqrt{F}$ generally does not suffice. Although the optimal value of this parameter depends on the specific dataset, it is clearly beneficial to diversify the trees and choose less then F features (i.e., a proportion less than 1.0). As for the proportion of the sampled examples (see Fig. 16c), typically, the more examples the better results. However, the differences between the performances observed for the higher two values (0.8 and 1.0) are almost negligible. On the other hand, taking only 60% of examples seems to negatively affect the results. This is interesting since the expected proportion

[17]`https://seaborn.pydata.org/`



(a) BASKET (b) IMDB

(c) MOVIE (d) STACK

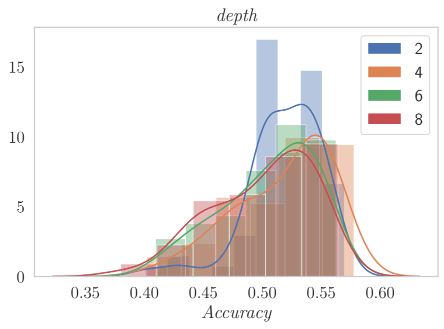
(e) UWCSE (f) WEBKB

(g) MUTAGENESIS (h) CARCINOGENESIS

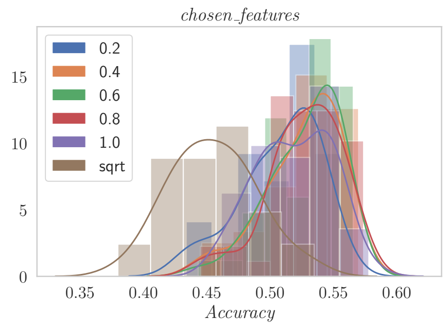
Figure 15: Feature importance scores returned by RE3PY (Bagging - AGG-All), for all the datasets (except for the YELP dataset, which is already shown in Fig. 14a).

of examples sampled by Bagging (and Random Forests) is $1 - 1/e \approx 0.63$, but Bagging still achieves state of the art results.

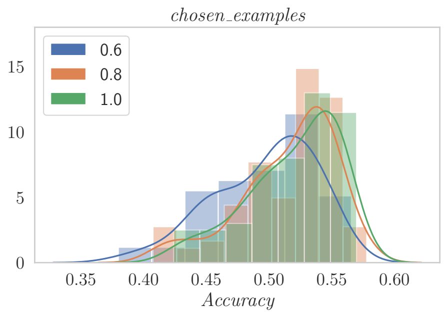
For the dataset at hand, the most influential parameter turns out to be



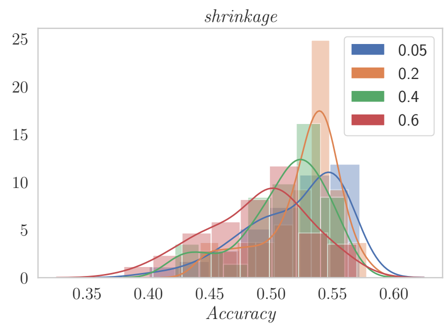
(a) Tree depth



(b) Proportion of sampled features



(c) Proportion of sampled examples



(d) Shrinkage

Figure 16: Analysis of the parameter influence through histograms (together with the estimated distributions) of the accuracy values measured on the UWCSE dataset.

the shrinkage (see Fig. 16d). Here, there is a clear preference towards lower values which means that the dataset is quite hard to model and it is preferable to converge to the final model in a slower manner.

The results of the parameter analysis for the other datasets are available at `https://github.com/re3py/re3py`.

7. Conclusions

In this work, we addressed the need for machine learning methods that can learn predictive models in powerful relational and logical representations. A characteristic of these representations is that they give rise to a huge number of features to be considered, thus drastically increasing the difficulty of learning in terms of computational complexity and the curse of dimensionality. Despite this, methods for ranking features in this context, i.e., estimating their importance are practically non-existent.

Among the most well-known methods for feature ranking are those based on ensembles, and in particular tree ensembles. To develop methods for feature ranking in a relational context, we adopt the relational tree ensemble approach. We thus first develop methods for learning ensembles of relational trees, extending a wide spectrum of tree-based ensembles methods from the propositional to the relational context, resulting in methods for bagging and random forests of relational trees, as well as gradient boosted ensembles thereof.

The novel procedure for generating relational features in the (ensembles of) trees covers two kinds of relational features. The first corresponds to existential queries, such as 'Does this person have any children?'. The second corresponds to aggregation queries, such as 'What is the average age of this person's children'. We also calculate feature importance scores and rankings from the different kinds of relational tree ensembles learned, with different kinds of relational features. The rankings provide insight into and explain the ensemble models, which would be otherwise difficult to understand.

The proposed method covers the Cartesian product of aggregated/existential splits and single trees/boosting/bagging/random forests. To some extent similar methods cover this options only partially. For example, FOIL [16] and BoostSRL [19] do not use aggregates, and relational random forests [53] and BoostSRL implement only one ensemble method.

The implementations of such methods, except for BoostSRL, do not follow the open science philosophy of reproducibility and reusability. They are not fully forking (e.g., TILDE), or are not publicly available. The code for our tree ensembles, on the other hand, is fairly easy to use and publicly available for download. Also the data used in our experiments are freely available.

The main findings, regarding the experimental evaluation of the proposed method, are the following:

- Ensembles outperform single trees, and the bagging ensembles perform the best, with gradient boosted ensembles following closely.
- Learning trees in a relational setting, i.e., considering more than just the target table, is clearly beneficial and can lead to substantial improvements (of even more than 50 percentage points) of the obtained predictive models, e.g., on the data sets UWCSE and WEBKB.

- Similarly goes for the use of aggregates in feature construction. Using them is beneficial, and drastically improves the quality of the obtained models on some data sets, e.g., UWCSE.
- Taking the maximal quality of the state-of-the-art competitors, and comparing it to our Bagging ensembles that use aggregation shows that the proposed method uniformly outperforms the others. The only case where this is not true is the WEBKB data set, where HENPC is better. However, note that HENPC models are learned in a transductive setting, i.e., need to see the descriptive parts of testing examples in advance.
- Our method yields models that are not only of high predictive performance, but are also explainable. As in the propositional case, this is achieved by defining feature ranking scores, for which we show that they return meaningful results.
- Parameter sensitivity analysis for Boosting shows that the most critical parameter is shrinkage, here - for the data at hand - smaller values are preferred. As for the number of features, it does not have major influence on the quality of the obtained models, provided it is high enough.

As for further work, there are many directions to explore. First, the proposed method, as implemented, is modular enough to handle not only classification, but also, e.g., regression. Moreover, by appropriately defining and implementing the heuristics for choosing the best candidate split, we can go beyond the primitive outputs, and also tackle structured outputs, such as multi-target regression and (hierarchical) multi-label classification.

Second, tree-based methods can also handle missing values in the target part of the data, so we plan to extend the approach towards the semi-supervised or unsupervised context. To achieve that, the tree induction algorithm should be further modified, since the standard predictive clustering approaches to learning in these two contexts can not be applied, due to the high number of features.

Third, we can also think about quantitatively evaluating feature rankings in a relational context. A possible approach from the propositional setting is to weight the dimensions (features) in the distance used by nearest-neighbor algorithm. However, even though relational NNs are one of the competitors in this paper, including the estimated feature importances as distance weights in the relational setting is a task on its own.

Next, another task that is easier to define or execute in the propositional setting, is feature selection where only a subset of most relevant features is selected to enter the subsequent experiments. However, in the relational setting, features (that can be constructed) may depend on the learning method. Thus, we plan to first formulate the problem correctly, and then evaluate our method in this context.

Finally, we plan to extend the evaluation of relational ensembles to real-world case studies, e.g., larger QSAR datasets.

Appendix A. Number of candidate features

When deriving the computational complexity of the method in Section 4.6, we assumed an upper bound for the number of features in internal nodes equal to F. In this Appendix, we discuss this upper bound more thoroughly, and explain the factor Fb^s in the time complexity $\mathcal{O}(Fnb^s \log n)$, where, as explained in Section 4.6, s is the length of the longest considered foreign-key paths, b is the branching factor and n is the number of reference objects (i.e., rows in the target table).

First, we count the number of foreign-key paths. In the following, we assume that each table contains K foreign keys, and that all predicates in the tables are numeric (this leads to the worst-case scenario, because of the additional sorting operation).

If the foreign key paths contain at most s tables, then the number of possible paths is $\mathcal{O}(K^s)$. If a is the number of used (numeric) aggregates, then every paths can be aggregated in a^s ways. This means that $F = a^s K^s = (aK)^s$. Note that, potentially, this result can be infeasibly high. However, the parameter s is indirectly controlled by the user-defined look-ahead parameter ℓ, that defines the maximal number of additional steps made in an internal node of a tree. In our experiments, we have shown that we can achieve state-of-the-art results with $\ell = 2$.

The evaluation of a feature $f(ro)$ for a given reference object ro proceeds as follows. Our implementation makes finding all tros at a given step in the path a constant operation (i.e., we store the pointers to all 1-related sets). Every such set will be aggregated exactly once, so the total time needed to evaluate a single split is proportional to the sum of the sizes of the discovered 1-related sets. Given b the branching factor, the aggregation is done in $\mathcal{O}(\sum_{i=1}^{s} b^i) = \mathcal{O}(b^s)$ operations (assuming $b > 1$). This step needs to be performed for every reference objects, so the total time for evaluating a single split is $\mathcal{O}(n'b^\ell)$ where n' is the number of reference objects that arrive to a given node.

To sum up, the time needed for computing the value of a single feature for a single reference object is $\mathcal{O}(b^s)$. Therefore, summing over different tree depths $d \in \{1, 2, \ldots, \log n\}$ (assuming balancing), the total tree-induction time corresponds to:

$$
\begin{aligned}
&= \mathcal{O}\left(\sum_{d=1}^{\log n} \underbrace{2^d}_{\substack{\text{number of nodes}\\ \text{at depth } d}} \cdot \left[\underbrace{\frac{n}{2^d} \cdot F \cdot b^s}_{\substack{\text{feature value computation}\\ \text{for all } n/2^d\ ros \text{ in a node}}} + F \cdot \underbrace{\left(\frac{n}{2^d} \log \frac{n}{2^d} + \frac{n}{2^d} \right)}_{\substack{\text{sorting the } ros\\ \text{with respect to a feature}\\ +\\ \text{evaluation of the splits}}} \right] \right) \\
&= \mathcal{O}\left(\sum_{d=1}^{\log n} Fn[b^s + \log n] \right) \\
&\overset{(*)}{=} \mathcal{O}\left(\sum_{d=1}^{\log n} Fnb^s \right) \\
&= \mathcal{O}\left(Fnb^s \log n \right) \\
&= \mathcal{O}\left((aK)^s nb^s \log n \right)
\end{aligned}
$$

$(*)$ Note that $s \approx \ell \log n$, since the number of additional hops increases by at most ℓ at every depth, thus $b^s \approx b^{\ell \log n}$. Since the logarithms above are binary and $b \geq 2$, it also follows that $b^{\ell \log n} \geq 2^{\ell \log n} = n^\ell$ and that $n^\ell \geq \log n$.

References

[1] B. Škrlj, J. Kralj, N. Lavrač, Targeted end-to-end knowledge graph decomposition, in: F. Riguzzi, E. Bellodi, R. Zese (Eds.), Inductive Logic Programming, Springer, 2018.

[2] S. Kramer, N. Lavrač, P. Flach, Relational Data Mining, LNAI, Springer-Verlag, Berlin Heidelberg Germany, 2001, Ch. Propositionalization Approaches to Relational Data Mining, pp. 262–291.

[3] M. Krogel, S. Rawles, F. Zelezny, P. Flach, N. Lavrac, S. Wrobel, Comparative evaluation of approaches to propositionalization, in: T. Horvath, A. Yamamoto (Eds.), Proceedings of International Conference on Inductive Logic Programming, Vol. 2835 of LNAI, Springer-Verlag, 2003, pp. 197–214.

[4] J. Knobbe, M. Haas, A. Siebes, Propositionalisation and aggregates, in: L. D. Raedt, A. Siebes (Eds.), Proceedings of PKDD 2001, Vol. 2168 of LNAI, Springer-Verlag, 2001, pp. 277–288.

[5] N. Lavrač, S. Džeroski, Inductive Logic Programming: Techniques and Applications, Ellis Horwood, Chichester, UK, 1994.

[6] L. Breiman, J. Friedman, R. Olshen, C. J. Stone, Classification and Regression Trees, Chapman & Hall/CRC, 1984.

[7] N. Lavrač, B. Škrlj, M. Robnik-Šikonja, Propositionalization and embeddings: two sides of the same coin, Mach. Learn. 109 (7) (2020) 1465–1507.

[8] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[9] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32. doi:10.1023/A:1010933404324.
URL `https://doi.org/10.1023/A:1010933404324`

[10] J. H. Friedman, Greedy function approximation: A gradient boosting machine, The Annals of Statistics 29 (5) (2001) 1189–1232.

[11] A. Holzinger, G. Langs, H. Denk, K. Zatloukal, H. Müller, Causability and explainability of artificial intelligence in medicine, WIREs Data Mining and Knowledge Discovery 9 (4) (2019) e1312.

[12] M. Hoogendoorn, P. Szolovits, L. M. Moons, M. E. Numans, Utilizing uncoded consultation notes from electronic medical records for predictive modeling of colorectal cancer, Artificial Intelligence in Medicine 69 (2016) 53–61.

[13] E. Tjoa, C. Guan, A survey on explainable artificial intelligence (xai): Towards medical xai (10 2019).

[14] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, P. Geurts, Inferring regulatory networks from expression data using tree-based methods, PLOS ONE 5 (9) (2010) 1–10. doi:10.1371/journal.pone.0012776.

[15] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157–1182.

[16] J. R. Quinlan, R. M. Cameron-Jones, Foil: A midterm report, in: Proceedings of the 6th European Conference on Machine Learning, ECML'93, Springer-Verlag, Berlin, Heidelberg, 1993, p. 1–20.

[17] S. Muggleton, Inverse entailment and progol, New Generation Computing 13 (3) (1995) 245–286.

[18] H. Blockeel, L. D. Raedt, Top-down induction of first-order logical decision trees, Artificial Intelligence 101 (1) (1998) 285 – 297.

[19] S. Natarajan, K. Kersting, T. Khot, J. Shavlik, Boosted statistical relational learners: From benchmarks to data-driven medicine, 2014.

[20] P. A. Flach, N. Lachiche, Naive bayesian classification of structured data, Mach. Learn. 57 (3) (2004) 233–269.

[21] M. Ceci, A. Appice, D. Malerba, Mr-sbc: A multi-relational naïve bayes classifier, in: N. Lavrac, D. Gamberger, H. Blockeel, L. Todorovski (Eds.), Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings, Vol. 2838 of Lecture Notes in Computer Science, Springer, 2003, pp. 95–106.

[22] M. Ceci, A. Appice, D. Malerba, Discovering emerging patterns in spatial databases: A multi-relational approach, in: J. N. Kok, J. Koronacki, R. L. de Mántaras, S. Matwin, D. Mladenic, A. Skowron (Eds.), Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007, Proceedings, Vol. 4702 of Lecture Notes in Computer Science, Springer, 2007, pp. 390–397. doi:10.1007/978-3-540-74976-9_38.
URL `https://doi.org/10.1007/978-3-540-74976-9\_38`

[23] M. Ceci, A. Appice, Spatial associative classification: propositional vs structural approach, J. Intell. Inf. Syst. 27 (3) (2006) 191–213. doi:10.1007/s10844-006-9950-x.
URL `https://doi.org/10.1007/s10844-006-9950-x`

[24] F. Serafino, G. Pio, M. Ceci, Ensemble learning for multi-type classification in heterogeneous networks, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2326–2339.

[25] A. Woznica, A. Kalousis, M. Hilario, Learning to combine distances for complex representations, in: Z. Ghahramani (Ed.), Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007, Vol. 227 of ACM International Conference Proceeding Series, ACM, 2007, pp. 1031–1038.

[26] M. Kirsten, S. Wrabel, T. Horváth, Distance Based Approaches to Relational Learning and Clustering, Springer-Verlag, Berlin, Heidelberg, 2001, p. 213–230.

[27] J. B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika 29 (1) (1964) 1–27.

[28] S. A. Macskassy, F. Provost, Classification in networked data: A toolkit and a univariate case study, J. Mach. Learn. Res. 8 (2007) 935–983.

[29] B. Gallagher, H. Tong, T. Eliassi-Rad, C. Faloutsos, Using ghost edges for classification in sparsely labeled networks, in: Proc. 14th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining, ACM, 2008, pp. 256–264.

[30] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data., AI Magazine 29:3 (2008) 93–106.

[31] D. Jensen, J. Neville, B. Gallagher, Why collective inference improves relational classification, in: Proc. 10th ACM SIGKDD, ACM, 2004, pp. 593–598.

[32] M. Bilgic, L. Getoor, Effective label acquisition for collective classification, in: ACM SIGKDD, ACM, 2008, pp. 43–51.

[33] X. Zhu, Z. Ghahramani, J. D. Lafferty, Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions, in: Proc. 20th ICML, AAAI Press, 2003, pp. 912–919.

[34] D. Malerba, M. Ceci, A. Appice, A relational approach to probabilistic classification in a transductive setting, Eng. Appl. Artif. Intell. 22 (1) (2009) 109–116. doi:10.1016/j.engappai.2008.04.005.
URL `https://doi.org/10.1016/j.engappai.2008.04.005`

[35] H. Rahmani, H. Blockeel, A. Bender, Predicting the functions of proteins in protein-protein interaction networks from global information, Journal of Machine Learning Research 8 (2010) 82–97.

[36] A. Appice, M. Ceci, D. Malerba, An iterative learning algorithm for within-network regression in the transductive setting, in: Discovery Science 2009, Springer, 2009, pp. 36–50.

[37] M. Ji, Y. Sun, M. Danilevsky, J. Han, J. Gao, Graph regularized transductive classification on heterogeneous information networks, in: J. L. Balcázar, F. Bonchi, A. Gionis, M. Sebag (Eds.), Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part I, Vol. 6321 of Lecture Notes in Computer Science, Springer, 2010, pp. 570–586.

[38] M. Ji, J. Han, M. Danilevsky, Ranking-based classification of heterogeneous information networks, in: SIGKDD '11, ACM, NY, USA, 2011, pp. 1298–1306.

[39] X. Kong, P. S. Yu, Y. Ding, D. J. Wild, Meta path-based collective classification in heterogeneous information networks, in: X. Chen, G. Lebanon, H. Wang, M. J. Zaki (Eds.), 21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012, ACM, 2012, pp. 1567–1571.

[40] C. Yang, M. Liu, F. He, X. Zhang, J. Peng, J. Han, Similarity modeling on heterogeneous networks via automatic path discovery, in: M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, G. Ifrim (Eds.), Machine Learning and Knowledge Discovery in Databases - European Conference,

ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part II, Vol. 11052 of Lecture Notes in Computer Science, Springer, 2018, pp. 37–54.

[41] Y. Dong, Z. Hu, K. Wang, Y. Sun, J. Tang, Heterogeneous network representation learning, in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, ijcai.org, 2020, pp. 4861–4867. doi:10.24963/ijcai.2020/677. URL `https://doi.org/10.24963/ijcai.2020/677`

[42] K. Steinhaeuser, N. V. Chawla, A. R. Ganguly, Complex networks as a unified framework for descriptive analysis and predictive modeling in climate science, Statistical Analysis and Data Mining 4 (5) (2011) 497–511.

[43] D. Stojanova, M. Ceci, A. Appice, S. Dzeroski, Network regression with predictive clustering trees, Data Min. Knowl. Discov. 25 (2) (2012) 378–413.

[44] D. Stojanova, M. Ceci, D. Malerba, S. Dzeroski, Using PPI network autocorrelation in hierarchical multi-label classification trees for gene function prediction, BMC Bioinformatics 14 (2013) 285.

[45] G. Pio, F. Serafino, D. Malerba, M. Ceci, Multi-type clustering and classification from heterogeneous networks, Inf. Sci. 425 (2018) 107–126.

[46] D. Grissa, M. Pétéra, M. Brandolini, A. Napoli, B. Comte, E. Pujos-Guillot, Feature selection methods for early predictive biomarker discovery using untargeted metabolomic data, Frontiers in Molecular Biosciences 3 (2016) 30.

[47] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19) (2007) 2507–2517.

[48] M. Tsagris, V. Lagani, I. Tsamardinos, Feature selection for high-dimensional temporal data, BMC Bioinformatics 19 (1) (2018) 17.

[49] M. Petković, D. Kocev, S. Džeroski, Feature ranking for multi-target regression, Machine Learning 109 (2020) 1179–1204.

[50] C. Vens, Complex aggregates in relational learning, Ph.D. thesis, Faculteit Ingenieurswetenschappen, Katholieke Univeristeit Leuven (2007).

[51] J. Platt, Fast training of support vector machines using sequential minimal optimization, in: Advances in Kernel Methods - Support Vector Learning, MIT Press, 1998.

[52] Y. L. Suen, P. Melville, R. J. Mooney, Combining bias and variance reduction techniques for regression trees, in: J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, L. Torgo (Eds.), Machine Learning: ECML 2005, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 741–749.

[53] A. Van Assche, C. Vens, H. Blockeel, S. Džeroski, A random forest approach to relational learning, in: T. Dietterich, L. Getoor, K. Murphy (Eds.), Proceedings of ICML 2004 workshop on Statistical Relational Learning and its Connections to Other Fields, 2004, pp. 110–116.

Chapter 10

Conclusions and Further Work

In this thesis, we developed feature ranking methods tailored to a variety of learning settings, bridging the gap between the ever more complex data on the one hand, and the lack of feature ranking methods that would explain the models learned on these data on the other hand. **The developed feature ranking methods address complex machine learning tasks from supervised, semi-supervised and unsupervised learning**. They learn feature rankings in the context of supervised and semi-supervised structured output prediction (SOP), including multi-target regression (MTR), multi-label classification (MLC), and hierarchical multi-label classification (HMLC). We have also **extended the feature ranking methods to relational learning**, where the data representation is even richer and more complex.

The developed feature ranking methods **handle these various learning contexts in an elegant and unified way**. At the same time, the feature rankings computed by these methods offer **state-of-the-art performance in all the learning contexts**, as proven by extensive empirical studies presented in this thesis. Moreover, the developed feature ranking methods **scale-up well**, since they are subquadratic in the number of features and easily parallelizable.

There are two corner stones that make the unified and elegant approach to the various problems possible. The first corner stone is that we follow the philosophy of predictive clustering, where attributes in data are treated as descriptive, clustering and target ones, and can belong to more than one of these groups. The second corner stone is the inherent generality of the algorithms that we extend.

The generality of the considered tree-ensemble-based and distance-based feature ranking methods originates from the fact that only a rather well-isolated part of the original algorithm needs to be changed in order to extend it to another task. For tree-ensemble based feature rankings, this is (mostly) the heuristic for assessing the quality of the splits, whereas for the distance-based methods, this is the distance in the target (or clustering) space. The impurity measures that are the main part of the tree-induction heuristics also measure distances between the examples. On the other hand, a target space that prohibits any metric definition is most probably a mathematical construct and, loosely speaking, rather unrelated to real life. This gives us the intuition why the considered methods are so widely applicable and extendable.

We have developed two groups of feature ranking methods. The first group of feature ranking methods is **tree-ensemble-based** and contains the Symbolic, Genie3 and Random Forest scores that are computed from different ensembles of predictive clustering trees (PCTs), i.e., random forests, bagging, and ensembles of extremely randomized PCTs. The second group of methods are **distance-based** methods that follow the Relief approach for feature ranking.

The Symbolic score is calculated using the specific variant of the ensemble methods used for the given task. For the extension of Genie3, one needs to understand that the score is expressed in terms of the quality of the splits in the trees of an ensemble, as assessed by the heuristic function. For the Random Forest score, one needs to adapt the error measure (which is used in the score) to the task at hand. The distance-based scores are extended by generalizing the original distance definitions on the target space (applicable to nominal sets and $\mathbb{R}$) in ReliefF to more general target spaces, such as $\mathbb{R}^D$, power-sets and partially ordered power-sets of nominal values.

Once the extensions are defined in the supervised learning setting, one can proceed to the semi-supervised versions of the above predictive modeling tasks, and to unsupervised learning as well. This holds both for the ensemble- and distance-based approaches. The extensions are made by following the philosophy of predictive clustering and appropriately defining the groups of descriptive and clustering attributes.

Extending the tree-ensembles as predictive models, as well as the corresponding feature ranking scores, from propositional to relational data, demands one step more, since only the features from the table that contains the target attribute(s) can be used during tree induction in the same way as in the propositional case. The values of the features from the other tables have to be first passed to the examples from the target table. We additionally had to define a procedure for constructing new, complex features whose building blocks are the ones provided in the data. This is to some extent related to representation learning, as we know it in the propositional setting. For example, when classifying images with neural networks, the hidden layers of a neural network construct new complex features from the original data representation. These features are then used to classify examples. Other methods for representation learning also exist, e.g., sum-product networks (Vergari et al., 2018). Moreover, we use UMAP (McInnes et al., 2018) to learn new representations of high-dimensional data (Škrlj et al., 2020).

To keep the interpretability of the constructed relational models, feature ranking scores should be calculated accordingly. Note that this is done only for the two tree-ensemble-based feature ranking scores that can be computed only from the trees; Symbolic and Genie3, since the Random Forest score cannot be easily extended to the relational setting. For similar reasons, the Relief family is not extended to the relational setting: Defining a distance in the relational case is possible, but is a line of research on its own.

In addition to the limitation mentioned in the above paragraph, the only notable weakness of the developed methods is that they may yield suboptimal feature rankings when the rankings are used to perform feature selection and there are redundant features among the most important one. The reason for this is that the developed tree-ensemble-based and distance-based methods assign similar scores to, for example, correlated features.

10.1 Summary of the Contributions

We summarize the contributions of this dissertation as follows:

- We have designed *two novel groups of feature ranking algorithms for different tasks (MTR, MLC, and HMLC) of (supervised) structured output prediction*: a group of ensemble-based and a group of distance-based algorithms. We started with MTR problems extending the feature ranking methods that can handle both nominal and numeric features. Regarding feature ranking for MLC, we first proposed the MLC-Relief algorithm that outperformed a preexisting competitor that ignores label interactions. Later, the ensemble-based scores achieved even better quality and at a lower computational cost. As for feature ranking methods for HMLC, our methods were the first that could handle such problems. In the cases of MLC and HMLC, ensemble-based feature ranking scores performed better than distance-based scores, whereas in the MTR case, there were no significant differences among them.

- In addition to the ability of the above feature ranking methods to handle nominal and numeric features on the one hand, and various output types on the other hand, they can also easily handle missing values both in the descriptive and target part of a data set. Thus, *we further extended the developed feature ranking methods for SOP tasks to their semi-supervised versions*, which is our next contribution. Since semi-supervised learning (SSL) means a new layer of complexity on its own, we also developed feature ranking methods that handle single target classification (STC) and single target regression (STR).

  The robustness of the proposed feature ranking methods came into play when we showed that they can use unlabeled data to improve the ranking quality, even when the basic clustering hypothesis of SSL is not satisfied. To the best of our knowledge, in most cases (STC, MTR, MLC, HMLC), the proposed methods were the first that could handle such problems. The exception was STR, where they outperformed the competitor, which also cannot handle nominal features. An interesting finding that holds regardless of the output type is that semi-supervised feature ranking captures a more global picture of a data set, when compared to its supervised version. Again, the ensemble-based approaches typically outperformed the distance-based ones.

- Our next contribution is the development of **feature ranking methods for unsupervised learning**. Even though this setting could be seen as a special case of SSL, the absence of the target attribute(s) simplifies matters to the extent that in this setting, many more feature ranking methods exist, compared to SSL. However, *both our approaches, distance-based and ensemble-based, clearly outperformed all the competitors*, including AgnoS-S (Doquet & Sebag, 2020) that was awarded the best paper award at the ECML 2019 conference.

- In the contributions listed above, we already covered two dimensions of data complexity: the type of the output and the amount of supervision. To describe the next contribution, we have to move along the third data complexity dimension, leave the propositional setting and enter the relational one. The next contribution is a **new algorithm for inducing relational trees (and ensembles thereof)**. The integral part of the algorithm is the construction of the expressive features that consist of combinations of descriptive relations, as given in a data set, and aggregates, which – as proved in the experiments – make the features even better. We evaluated the *relational trees and ensembles thereof in the context of relational classification and showed that they outperform the other competitors.*

  Moreover, we also **explained the obtained relational models by computing feature rankings** out of them. This was done by extending the Symbolic and Genie3 scores to the relational setting. This is, to the best of our knowledge, the first feature ranking approach in the relational setting. The obtained rankings were shown to be meaningful when the learned relational model reflected the target concept well.

- The final contribution of this thesis is a case study that uses the developed ensemble-based feature ranking algorithms for MTR to explain the models that predict the thermal power consumption of ESA's Mars Express spacecraft in two operating conditions, with or without gyroscopes. Based on the obtained rankings, the spacecraft operators can understand and verify the predictive models, and also quantify the differences between them.

Following the philosophy of **open, reproducible and transparent science**, the implementations of the methods above are freely available, as well as the results, upon which the analysis of their performance is made (the exact locations are given in the corresponding papers). Note that the methods are always extensively evaluated on a number of data sets that typically exceeds 20.

10.2 Further Work

The presented work is comprehensive in terms of the addressed machine learning tasks and challenges. It answers the questions posed in the introduction, and **opens new directions for further research**. The major directions of further research can be summarized as follows.

First of all, although we carefully designed the distances and the impurity measures in the proposed methods, we plan to further investigate the effect of the use of other distances and impurity measures on the performance of the proposed methods. The distances can be designed in a way to include some peculiarities of the output space under consideration (e.g., in HMLC one can aggregate the contribution of the labels at the different levels in a variety of ways). We will also explore the wealth of existing kernels for inclusion in the core of the developed methods.

Second, in the SOP tasks, besides labeled and unlabeled examples, partially labeled examples can also be encountered. An example is partially labeled if not all values for the target variables are available (e.g., in QSAR, not always all compounds are tested for all potential adverse effects), and some are. The proposed methods can be easily adapted to learning from partially labeled examples: the ensemble-based ones already support this to an extent (PCTs can be learned from partially labeled examples). In the distance-based methods, the distance calculation could be updated to consider only the portions of the target variables that are available. Once the scores are extended, we will evaluate and benchmark them in a variety of SOP domains.

Third, the proposed methods (especially the ensemble-based ones) can be used to explain the predictions made for individual examples by the predictive models as well as to explain the predictions made for a set of examples. The explanation will be provided as feature rankings relevant for those specific examples. For example, the Symbolic and Genie3 scores can be calculated using only the splits reached by the examples in question, and the Random forest score can be calculated by permuting the values only on the examples in question.

Forth, in the context of relational learning, we plan to extend the proposed approach to regression as well as to the SOP tasks considered in this dissertation. The proposed methods could also be extended towards semi-supervised and unsupervised relational learning. Moreover, relational feature rankings could be used to improve the random-walk-based graph embedding algorithm by guiding the random walks using the feature importance to assign probability to every type of edge. This is possible since the constructed features in our relational trees correspond to walks in the underlying network.

Last, but not least, since the developed methods cannot handle well redundant features, we plan to investigate how these methods can be joined with other unsupervised feature ranking methods into a chain, where we would first use our method to compute the ranking, and then further filter this ranking with another method to remove redundant features from the top-ranked ones.

References

Amaldi, E., & Kann, V. (1998). On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, *209*, 237–260.

Arthur, D., & Vassilvitskii, S. (2007). K-Means++: The Advantages of Careful Seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035.

Bakır, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (Eds.). (2007). *Predicting structured data*. The MIT Press.

Barutcuoglu, Z., Schapire, R. E., & Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, *22*(7), 830–836.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society.*, *57*(1), 289–300. https://doi.org/10.2307/2346101

Blockeel, H. (1998). *Top-down Induction of First Order Logical Decision Trees* (Doctoral dissertation). Katholieke Universiteit Leuven. Leuven, Belgium.

Blockeel, H., & Raedt, L. D. (1998). Top-down induction of first-order logical decision trees. *Artificial Intelligence*, *101*(1), 285–297.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. J. (1984). *Classification and Regression Trees*. Chapman & Hall/CRC.

Breskvar, M., Kocev, D., Levatić, J., Osojnik, A., Petković, M., Simidjievski, N., Ženko, B., Boumghar, R., & Lucas, L. (2017). Predicting Thermal Power Consumption of the Mars Express Satellite with Machine Learning. *Proceedings of the 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 88–93.

Cai, D., Zhang, C., & He, X. (2010). Unsupervised Feature Selection for Multi-Cluster Data. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 333–342. https://doi.org/10.1145/1835804.1835848

Chicarro, A., Martin, P., & Trautner, R. (2004). The Mars Express mission: An overview. *European Space Agency, (Special Publication) ESA SP*, *1240*, 3–13.

Corani, G., Benavoli, A., Demšar, J., Mangili, F., & Zaffalon, M. (2017). Statistical comparison of classifiers through Bayesian hierarchical modelling. *Machine Learning*, *106*, 1817–1837.

Cover, T. M., & Thomas, J. A. (2006). *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience.

Cristianini, N., & Shawe-Taylor, J. (2010). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Cunningham, P., & Delany, S. J. (2007). *k-Nearest Neighbour Classifiers* (tech. rep.). University College Dublin, Dublin, Ireland.

de Bruijne, M. (2016). Machine learning approaches in medical image analysis: From detection to diagnosis [20th anniversary of the Medical Image Analysis journal (MedIA)]. *Medical Image Analysis*, *33*, 94–97. https://doi.org/https://doi.org/10.1016/j.media.2016.06.032

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

DiMasi, J. A., Hansen, R. W., & Grabowski, H. G. (2003). The price of innovation: new estimates of drug development costs. *Journal of Health Economics*, *22*(2), 151–185.

Doquet, G., & Sebag, M. (2020). Agnostic Feature Selection. https://doi.org/10.1007/978-3-030-46150-8_21

Elisseeff, A., & Weston, J. (2001). A Kernel Method for Multi-Labelled Classification. *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 681–687.

Elkafrawy, P., Mausad, A., & Esmail, H. (2015). Experimental Comparison of Methods for Multi-label Classification in different Application Domains. *International Journal of Computer Applications*, *114*, 1–9.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378.

Fürnkranz, J., Gamberger, D., & Lavrač, N. (2014). *Foundations of Rule Learning*. Springer Publishing Company, Incorporated.

Geurts, P., Erns, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *36*(1), 3–42.

Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*(1), 389–422.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*, 993–1001.

He, X., Cai, D., & Niyogi, P. (2005). Laplacian Score for Feature Selection. *Proceedings of the 18th International Conference on Neural Information Processing Systems*, 507–514.

Hoy, M. B. (2018). Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants [PMID: 29327988]. *Medical Reference Services Quarterly*, *37*(1), 81–88. https://doi.org/10.1080/02763869.2018.1404391

Hrbacek, K., & Jech, T. (1999). *Introduction to set theory: Third Edition, Revised and Expanded*. Marcel Dekker, Inc.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., & Geurts, P. (2010). Inferring Regulatory Networks from Expression Data Using Tree-Based Methods. *PLoS One*, *5*(9), 1–10.

Hyafil, L., & Rivest, R. L. (1976). Constructing Optimal Binary Decision Trees is NP-Complete. *Information Processing Letters*, *5*, 15–17.

International Organization for Standardization. (2007). *ISO/IEC 11404:2007 - Information technology — General-Purpose Datatypes (GPD)* (tech. rep.). International Organization for Standardization. https://www.iso.org/standard/39479.html

Kira, K., & Rendell, L. A. (1992). A practical approach to feature selection. *ML92: Proceedings of the 9th international workshop on Machine learning*, 249–256.

Kittler, J. (1978). Pattern Recognition and Signal Processing. Sijthoff; Noordhoff.

Kocev, D. (2011). *Ensembles for predicting structured outputs* (Doctoral dissertation). IPS Jožef Stefan. Ljubljana, Slovenia.

Kocev, D., Ceci, M., & Stepišnik, T. (2020). Ensembles of extremely randomized predictive clustering trees for predicting structured outputs. *Machine learning*, *In press.*

Kocev, D., & Džeroski, S. (2013). Habitat modeling with single- and multi-target trees and ensembles. *Ecological Informatics*, *18*, 79–92.

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, *46*(3), 817–833.

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*, 3–24.

Kralj Novak, P., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of Emojis. *PloS one*, *10*(12), 1–22.

Levatić, J. (2017). *Semi-supervised Learning for Structured Output Prediction* (Doctoral dissertation). Jožef Stefan Postgraduate School. Ljubljana, Slovenia.

Li, Z., Yang, Y., Liu, J., Zhou, X., & Lu, H. (2012). Unsupervised Feature Selection Using Nonnegative Spectral Analysis. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 1026–1032.

Lucas, L., & Boumghar, R. (2017). Machine Learning for Spacecraft Operations Support - The Mars Express Power Challenge. *Proceedings of the Sixth International Conference on Space Mission Challenges for Information Technology, SMC-IT 2017*, 82–87.

Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, *45*, 3084–3104.

McInnes, L., Healy, J., Saul, N., & Grossberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *The Journal of Open Source Software*, *3*(29), 861.

Mitchell, T. (1997). *Machine learning*. McGraw Hill.

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons* (Doctoral dissertation). Princeton University. Princeton, NY, USA.

Nielsen, F. (2016). Introduction to HPC with MPI for Data Science. Springer. https://doi.org/10.1007/978-3-319-21903-5_8

Pearson, K. (1895). Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, *58*, 240–242.

Petković, M., Boumghar, R., Breskvar, M., Džeroski, S., Kocev, D., Levatić, J., Lucas, L., Osojnik, A., Ženko, B., & Simidjievski, N. (2019). Machine Learning for Predicting Thermal Power Consumption of the Mars Express Spacecraft. *IEEE Aerospace and Electronic Systems Magazine*, *34*(7), 46–60.

Petković, M., Ceci, M., Kersting, K., & Džeroski, S. (2020). Estimating the Importance of Relational Features by using Gradient Boosting. *Proceedings of the 25th International Symposium on Methodologies for Intelligent Systems*, *12117*, 362–371.

Petković, M., Džeroski, S., & Kocev, D. (2017). Feature Ranking for Multi-target Regression with Tree Ensemble Methods. In A. Yamamoto, T. Kida, T. Uno, & T. Kuboyama (Eds.), *Discovery Science* (pp. 171–185). Springer International Publishing.

Petković, M., Džeroski, S., & Kocev, D. (2018). Feature ranking for hierarchical multi-label classification with tree ensemble methods. *ETAI 2018 : proceedings of abstracts*.

Petković, M., Džeroski, S., & Kocev, D. (2019). Ensemble-Based Feature Ranking for Semi-supervised Classification. In P. Kralj Novak, T. Šmuc, & S. Džeroski (Eds.), *Discovery Science* (pp. 290–305). Springer International Publishing.

Petković, M., Džeroski, S., & Kocev, D. (2020a). Feature Ranking for Hierarchical Multi-Label Classification with Tree Ensemble Methods [In Press]. *Acta Polytechnica Hungarica*.

Petković, M., Džeroski, S., & Kocev, D. (2020b). Feature Ranking for Semi-supervised Learning [Under Review]. *Machine Learning.*

Petković, M., Džeroski, S., & Kocev, D. (2020c). Multi-label feature ranking with ensemble methods [In press]. *Machine Learning.* https://doi.org/10.1007/s10994-020-05908-1

Petković, M., Kocev, D., & Džeroski, S. (2018). Feature Ranking with Relief for Multi-label Classification: Does Distance Matter? In L. Soldatova, J. Vanschoren, G. Papadopoulos, & M. Ceci (Eds.), *Discovery Science, LNCS* (pp. 51–65).

Petković, M., Kocev, D., & Džeroski, S. (2020). Feature Ranking for Multi-Target Regression. *Machine Learning*, *109*, 1179–1204.

Petković, M., Lucas, L., Kocev, D., Džeroski, S., Boumghar, R., & Simidjievski, N. (2019). Quantifying the Effects of Gyroless Flying of the Mars Express Spacecraft with Machine Learning. *Proceedings of the 2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 9–16.

Petković, M., Pio, G., Ceci, M., Škrlj, B., Kersting, K., & Džeroski, S. (2020). Relational Tree Ensembles and Feature Rankings [To be submitted]. *Artificial Intelligence.*

Petković, M., Škrlj, B., Kocev, D., & Džeroski, S. (2020). Feature Ranking for unsupervised Learning [Under Review]. *Information Fusion.*

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, *1*, 81–106.

Quinlan, J. R., & Cameron-Jones, R. M. (1993). FOIL: A Midterm Report. *Proceedings of the 6th European Conference on Machine Learning*, 1–20.

Reyes, O., Morell, C., & Ventura, S. (2015). Scalable extensions of the ReliefF algorithm for weighting and selecting features on the multi-label learning context. *Neurocomputing*, *161*, 168–182.

Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning*, *55*, 23–69.

Rousu, J., Saunders, C., Szedmak, S., & Shawe-Taylor, J. (2006). Kernel–based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, *7*, 1601–1626.

Ryan, T. P. (2008). *Modern Regression Methods* (2nd ed.). Wiley-Interscience.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, *27*(3), 379–423.

Siedlecky, W., & Sklansky, J. (1988). On Automatic Feature Selection. *International Journal of Pattern Recognition and Artificial Intelligence*, *2*(2), 197–220.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*, 484–489. https://doi.org/10.1038/nature16961

Škrlj, B., Džeroski, S., Lavrač, N., & Petković, M. (2020). ReliefE: High dimensional feature ranking via manifold embeddings [Under review]. *Machine Learning.*

Slavkov, I. (2012). *An Evaluation Method for Feature Rankings* (Doctoral dissertation). IPS Jožef Stefan. Ljubljana, Slovenia.

Stańczyk, U., & Jain, L. C. (Eds.). (2015). *Feature Selection for Data and Pattern Recognition.* Springer Berlin Heidelberg.

Tsoumakas, G., & Vlahavas, I. (2007). Random k-Labelsets: An Ensemble Method for Multilabel Classification. In J. N. Kok, J. Koronacki, R. L. d. Mantaras, S. Matwin, D. Mladenič, & A. Skowron (Eds.), *Machine Learning: ECML 2007* (pp. 406–417). Springer Berlin Heidelberg.

Vens, C. (2007). *Complex aggregates in Relational Learning* (Doctoral dissertation). Faculteit Ingenieurswetenschappen, Katholieke Univeristeit Leuven.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, *73*(2), 185–214.

Vergari, A., Peharz, R., Di Mauro, N., Molina, A., Kersting, K., & Esposito, F. (2018). A comparison of different data transformation approaches in the feature ranking context. *Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence*, 4163–4170.

Welch, W. (1982). Algorithmic Complexity: Three NP-Hard Problems in Computational Statistics. *Journal of Statistical Computation and Simulation*, *15*, 17–25. https://doi.org/10.1080/00949658208810560

Wettschereck, D. (1994). *A Study of Distance Based Algorithms* (Doctoral dissertation). Oregon State University. Corvallis, OR.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques, 3rd edition*. Morgan Kaufmann.

Zhao, Z., & Liu, H. (2007). Spectral feature selection for supervised and unsupervised learning. *ACM International Conference Proceeding Series*, *227*, 1151–1157. https://doi.org/10.1145/1273496.1273641

Zhu, X., Goldberg, A. B., Brachman, R., & Dietterich, T. (2009). *Introduction to Semi-Supervised Learning*. Morgan; Claypool Publishers.

Bibliography

Publications Included in the Thesis

Journal Articles

Petković, M., Boumghar, R., Breskvar, M., Džeroski, S., Kocev, D., Levatić, J., Lucas, L., Osojnik, A., Ženko, B., & Simidjievski, N. (2019). Machine Learning for Predicting Thermal Power Consumption of the Mars Express Spacecraft. *IEEE Aerospace and Electronic Systems Magazine*, *34*(7), 46–60.

Petković, M., Džeroski, S., & Kocev, D. (2020a). Feature Ranking for Hierarchical Multi-Label Classification with Tree Ensemble Methods [In Press]. *Acta Polytechnica Hungarica*.

Petković, M., Džeroski, S., & Kocev, D. (2020b). Feature Ranking for Semi-supervised Learning [Under Review]. *Machine Learning*.

Petković, M., Džeroski, S., & Kocev, D. (2020c). Multi-label feature ranking with ensemble methods [In press]. *Machine Learning*. https://doi.org/10.1007/s10994-020-05908-1

Petković, M., Kocev, D., & Džeroski, S. (2020). Feature Ranking for Multi-Target Regression. *Machine Learning*, *109*, 1179–1204.

Petković, M., Pio, G., Ceci, M., Škrlj, B., Kersting, K., & Džeroski, S. (2020). Relational Tree Ensembles and Feature Rankings [To be submitted]. *Artificial Intelligence*.

Petković, M., Škrlj, B., Kocev, D., & Džeroski, S. (2020). Feature Ranking for unsupervised Learning [Under Review]. *Information Fusion*.

Conference Papers

Petković, M., Ceci, M., Kersting, K., & Džeroski, S. (2020). Estimating the Importance of Relational Features by using Gradient Boosting. *Proceedings of the 25th International Symposium on Methodologies for Intelligent Systems*, *12117*, 362–371.

Petković, M., Kocev, D., & Džeroski, S. (2018a). Feature Ranking with Relief for Multi-label Classification: Does Distance Matter? In L. Soldatova, J. Vanschoren, G. Papadopoulos, & M. Ceci (Eds.), *Discovery Science, LNCS* (pp. 51–65).

Petković, M., Lucas, L., Kocev, D., Džeroski, S., Boumghar, R., & Simidjievski, N. (2019). Quantifying the Effects of Gyroless Flying of the Mars Express Spacecraft with Machine Learning. *Proceedings of the 2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 9–16.

Publications Related to the Thesis

Journal Articles

Petković, M., Popovski, G., Koroušić Seljak, B., Kocev, D., & Eftimov, T. (2020). DIETHUB: Dietary Habits Analysis through Understanding the Content of Recipes [Under review]. *Trends in Food Science & Technology.*

Petković, M., Škrlj, B., Kocev, D., & Simidjievski, N. (2020). Fuzzy Jaccard Index: A robust comparison of ordered lists [Under review]. *Information Fusion.*

Petković, M., Slavkov, I., Kocev, D., & Džeroski, S. (2020). Biomarker Discovery by Feature Ranking: Evaluation on a case study of Embryonal Tumors [Under review]. *Computers in Biology and Medicine.*

Škrlj, B., Džeroski, S., Lavrač, N., & Petković, M. (2020b). ReliefE: High dimensional feature ranking via manifold embeddings [Under review]. *Machine Learning.*

Slavkov, I., Petković, M., Geurts, P., Kocev, D., & Džeroski, S. (2020). Error curves for evaluating the quality of feature rankings [Under review]. *PeerJ Computer Science.*

Slavkov, I., Petković, M., Kocev, D., & Džeroski, S. (2018). Quantitative score for assessing the quality of feature rankings. *Informatica: An International Journal of Computing and Informatics*, *42*(1), 43–52.

Conference Papers

Breskvar, M., Kocev, D., Levatić, J., Osojnik, A., Petković, M., Simidjievski, N., Ženko, B., Boumghar, R., & Lucas, L. (2017). Predicting Thermal Power Consumption of the Mars Express Satellite with Machine Learning. *Proceedings of the 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 88–93.

Petković, M., Džeroski, S., & Kocev, D. (2017). Feature Ranking for Multi-target Regression with Tree Ensemble Methods. In A. Yamamoto, T. Kida, T. Uno, & T. Kuboyama (Eds.), *Discovery Science* (pp. 171–185). Springer International Publishing.

Petković, M., Džeroski, S., & Kocev, D. (2018a). Feature ranking for hierarchical multi-label classification with tree ensemble methods. *ETAI 2018 : proceedings of abstracts.*

Petković, M., Panov, P., & Džeroski, S. (2015). Izboljšano ocenjevanje pomembnosti zveznih značilk. In R. Piltaver & M. Gams (Eds.), *Intelligent systems : proceedings of the 18th International Multiconference Information Society - IS 2015* (pp. 92–95).

Petković, M., Panov, P., & Džeroski, S. (2016). A comparison of different data transformation approaches in the feature ranking context. In T. Calders, D. Malerba, & M. Ceci (Eds.), *Discovery Science, LNCS* (pp. 310–324).

Petković, M., Tanevski, J., Maver, A., Vidmar, L., Peterlin, B., & Džeroski, S. (2017). Comparison of feature ranking approaches for discovery of rare genetic variants related to multiple sclerosis. In M. Luštrek, R. Piltaver, & M. Gams (Eds.), *Slovenian Conference on Artificial Intelligence : proceedings of the 20th International Multiconference Information Society - IS 2017* (pp. 11–14).

Škrlj, B., Džeroski, S., Lavrač, N., & Petković, M. (2020a). Feature Importance Estimation with Self-Attention Networks. *24th European Conference on Artificial Intelligence (ECAI 2020).*

Other Publications

Journal Articles

Eftimov, T., Popovski, G., Petković, M., Koroušić Seljak, B., & Kocev, D. (2020). COVID-19 pandemic changes the food consumption patterns [In press]. *Trends in Food Science & Technology*.

Biography

Matej Petković was born on 25 June 1991 in Ljubljana, Slovenia, where he finished primary (OŠ Zalog) and secondary school (Gimnazija Bežigrad). In 2010, he started the bachelor's study of mathematics at the Faculty of Mathematics and Physics, University of Ljubljana. In 2013, he was awarded Third Prize at the International Mathematics Competition for University Students. After graduating with distinction in 2013 (GPA 10.00), he started his master's study at the same faculty. In 2014, he was awarded Second Prize at the International Mathematics Competition for University Students, and Special distinction for extraordinary study success at the University of Ljubljana (slo. Svečana listina za najboljši študijski uspeh). He finished his master's study with distinction in 2015 (GPA 10.00) with the defence of the master's thesis entitled "Estimation of the relevance of continuous features with ReliefF" (slo. Ocenjevanje pomembnosti zveznih značilk z metodo ReliefF) under the mentorship of Prof. Dr. Sašo Džeroski. The thesis was awarded the faculty Prešeren prize.

In 2015, he started working as a young researcher at the Department of Knowledge Technologies (Jožef Stefan Institute) under the supervision of Prof. Dr. Marko Bohanec, and enrolled in the PhD program entitled Information and Communication Technologies at the Jožef Stefan International Postgraduate School under the supervision of Prof. Dr. Sašo Džeroski. In 2017, he became an assistant at the Faculty of Mathematics and Physics (University of Ljubljana) in the field of mathematics.

During his studies, he took part in two EU funded projects including HBP (The Human Brain Project) and MAESTRA (Learning from Massive, Incompletely annotated, and Structured Data), and two projects funded by the European Space Agency ESA: GalaxAI – Machine Learning for Space Operations and AiTLAS – Artificial Intelligence Toolbox for Earth Observation.

His research interest is in the area of machine learning, mostly in the area of feature ranking for structured output prediction. He has published several scientific papers and has presented his work at several international conferences and workshops, both in the area of machine learning and the areas of applications.