# TREE AND RULE ENSEMBLES FOR MULTI-TARGET PREDICTION WITH RANDOM OUTPUT SUBSPACES

Martin Breskvar

**Doctoral Dissertation**
**Jožef Stefan International Postgraduate School**
**Ljubljana, Slovenia**

**Supervisor:** Prof. Dr. Sašo Džeroski, Jožef Stefan Institute, Ljubljana, Slovenia
**Co-Supervisor:** Dr. Dragi Kocev, Jožef Stefan Institute, Ljubljana, Slovenia

**Evaluation Board:**
Asst. Prof. Dr. Bernard Ženko, Chair, Jožef Stefan Institute, Ljubljana, Slovenia
Asst. Prof. Dr. Celine Vens, Member, Faculty of Medicine, KU Leuven, Campus KULAK, Kortrijk, Belgium
Asst. Prof. Dr. Petra Kralj Novak, Member, Jožef Stefan Institute, Ljubljana, Slovenia

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL


Martin Breskvar

# TREE AND RULE ENSEMBLES FOR MULTI-TARGET PREDICTION WITH RANDOM OUTPUT SUBSPACES

**Doctoral Dissertation**

# ANSAMBLI DREVES IN PRAVIL ZA VEČCILJNO NAPOVEDOVANJE Z NAKLJUČNO IZBRANIMI IZHODNIMI PODPROSTORI

**Doktorska disertacija**

**Supervisor:** Prof. Dr. Sašo Džeroski

**Co-Supervisor:** Dr. Dragi Kocev

Ljubljana, Slovenia, March 2019

*The power of willful ignorance cannot be overstated.*

*To those who have no voice.*

# Acknowledgments

I vividly remember walking towards my car when I received the first phone call from my soon-to-be supervisor Prof. Dr. Sašo Džeroski. Sašo "pitched" an open PhD position to me and it was then that I have first witnessed how preparation truly can be the key to success. With his well-prepared monologue my concerns and self-doubt faded away before reaching my car and before I knew it, I was a member of his research group. Sašo, even though I might have not been the obvious candidate for this position, you gave me a chance and for that, I am very grateful. Thank you.

I thank the members of the evaluation board, Asst. Prof. Dr. Bernard Ženko, Asst. Prof. Dr. Celine Vens and Asst. Prof. Dr. Petra Kralj Novak for taking the time to read, evaluate and consequently improve this work.

This dissertation was financially supported by the Slovenian Research Agency through a Young Researcher grant and by the European Commission through the projects *MAESTRA – Learning from Massive, Incompletely annotated and Structured Data* (ICT-2013-612944) and *The Human Brain Project* (FP7-ICT-604102, HBP SGA1 720270).

I thank my co-supervisor and office mate Dr. Dragi Kocev for enduring the recurring floods of annoying questions and finding the will to answer them all. Dragi, this dissertation would have not been possible without your input.

My gratitude also extends to Jurica Levatić, Nikola Simidjievski, Matej Petković and Jasmin Bogatinovski – the remaining N404 "inhabitants". I have learned a lot from you guys and I thank you for making the office, in spite of my bad jokes, a pleasant place of work. You all gave it your best to improve my sense of humor but, truth be told, some things are beyond repair. However, as is often the case in research, even negative results can yield a positive outcome: Thank you for instituting the Breskvar index (BI, more is better) for measuring the quality of jokes and additionally amending its original value scale of 0 to 100 to include real numbers in order to adequately assess my jokes, usually strongly dominating the BI $< 1.0$ region.

I would like to thank the colleagues from the Department of Knowledge Technologies and especially members of our research group for providing a friendly and professional work environment. I would also like to acknowledge the MMMe8 team and state that I will fondly remember the time we spent working on the Mars Power Challenge. Special thanks goes to Milica Bauer for tirelessly navigating all of us through the bureaucratic mazes, for organizing great events and for thousands of small chats.

The work presented in this dissertation carries my name but I consider it to be an achievement of many people. People, who were involuntarily volunteered into suffering my daily ramblings or worse. People, who helped me with advice, with kind words of encouragement, or with simple things like preparing lunch or kindly suggesting to restart my kindness machine in order to avoid the backlash of my behavior. In other words, my friends and family. My sincerest thanks to you all.

The greatest burden of my PhD journey fell onto my wonderful wife. Mateja, without you, this dissertation would never see the light of day. I thank you for helping me in each

and every (un)imaginable way. Thank you for enduring through all the ups and downs. I am indescribably grateful for all your support and extremely lucky to have you by my side. You are my life – I love you.

# Abstract

This dissertation deals with supervised learning, an area within the field of machine learning, where the goal is to learn predictive models. We focus on the challenging task of multi-target prediction (MTP). Contrary to classical predictive modeling, where the goal is to predict the value of one target, MTP requires the prediction of values of multiple targets.

The central premise of MTP is that the target variables are parts of a structure, where the structure is seen as a set of connected building blocks that individually carry information, but jointly represent a meaningful concept. We focus on solving tasks where all target attributes are either continuous (multi-target regression, MTR) or binary (multi-label classification, MLC).

MTP tasks can be solved either locally or globally. Local approaches decompose the original MTP problem into a set of single-target problems, which are then solved by applying existing methods that do not account for the possible relations between the target variables. Alternatively, global approaches learn a single model that predicts all target variables simultaneously and explicitly or implicitly take into account interrelations between the target variables.

Ensemble models are often used to obtain premium predictive performance. An ensemble is a collection of base models, whose predictions are combined to obtain an overall prediction. Base models are often built by using the same learning algorithm on variations of the provided data. Variations are typically made by sampling data instances, input variables or both. We focus on ensemble methods for MTP that introduce such variations in the input space and use predictive clustering trees (PCTs) as base predictive models.

When solving MTP tasks, data variations are also possible in the target space. We extend tree ensemble methods by introducing Random Output Selections (ROS) to learn ensembles of PCTs for MTP on subsets of target variables and thus consider interrelations (within each PCT) only among the selected subset of target variables.

The proposed method was implemented for several types of tree ensembles for MTP. We have applied tree ensembles with ROS to MTR as well as MLC tasks and have extensively evaluated our method on a variety of benchmark datasets from various domains. Moreover, a novel approach to combining base model predictions has been introduced. Usually, all ensemble members give predictions for all target variables, but in ROS ensembles, for a given target, we can choose to combine only the predictions of base predictive models, learned by using that target variable. Tree ensembles with ROS outperform current state-of-the-art MTR methods and perform comparably to current MLC methods.

Ensembles of PCTs are the state-of-the-art in MTP, but are generally not considered interpretable. Consequently, less powerful, but more interpretable models are often used instead. Fitted rule ensembles for MTR use ensembles of PCTs to derive a large pool of candidate rules of which only a small set is retained after optimization. We extend this approach by using tree ensembles with ROS to generate candidate rules. This approach performs slightly better than state-of-the-art rule-based methods for MTR.

# Povzetek

Pričujoča disertacija se ukvarja z nadzorovanim učenjem, področjem strojnega učenja, katerega cilj je učenje napovednih modelov. Osredotočamo se na reševanje naloge večciljnega napovedovanja (VCN). V nasprotju s klasičnim napovednim modeliranjem, kjer napovedujemo vrednost ene, v VCN napovedujemo vrednosti večih ciljnih spremenljivk.

Osrednja predpostavka VCN je, da so ciljne spremenljivke del strukture, kjer je struktura razumljena kot množica med seboj povezanih gradnikov, ki posamično vsebujejo informacije, skupaj pa tvorijo smiseln koncept. Rešujemo naloge, kjer so vse ciljne spremenljivke ali številske (večciljna regresija, VCR) ali binarne (večoznačna klasifikacija, VOK).

Naloge VCN se lahko rešujejo z uporabo lokalnih ali globalnih pristopov. Lokalni pristopi razgradijo prvotni problem VCN v množico problemov napovedovanja ene vrednosti, ki se nato rešujejo z uporabo obstoječih metod, ki pa ne upoštevajo morebitne povezanosti ciljnih spremenljivk. Po drugi strani, se z globalnimi pristopi učimo samo en napovedni model, ki napoveduje vse ciljne spremenljivke hkrati ter neposredno ali posredno upošteva morebitno medsebojno povezanost ciljnih spremenljivk.

Ansambelske metode se uporabljajo za doseganje najvišje napovedne moči. Ansambel je zbirka osnovnih modelov, katerih napovedi so združene v eno samo napoved. Osnovni modeli v ansamblu so pogosto naučeni z uporabo enega učnega algoritma, pri čemer vsak osnovni model učimo na vzorčenih učnih podatkih. Običajno se poslužujemo vzorčenja primerov, vhodnih spremenljivk ali obojega. V tej disertaciji se osredotočamo na ansambelske metode za VCN, ki se uporabljajo omenjena vzorčenja podatkov na vhodnih spremenljivkah ter kot osnovne modele uporabljajo drevesa za napovedno razvrščanje (DNR).

Pri reševanju nalog VCN je vzorčenje možno tudi v izhodnem prostoru. Ansambelske metode na osnovi DNR za VCN razširjamo z vpeljavo Naključnega Izbiranja Izhodnih Podprostorov (NIIP). Posamezna DNR se učimo na podmnožici ciljnih spremenljivk, kar morebitno povezanost ciljnih spremenljivk omeji le na izbrane ciljne spremenljivke.

NIIP je integriran v več metod za grajenje ansamblov DNR za VCN. Ansamble z NIIP smo uporabili na problemih VCR in VOK, ter jih temeljito ovrednotili na vrsti podatkovij iz različnih domen. Predlagan je tudi nov način združevanja napovedi v ansamblih. Osnovni modeli v ansamblu običajno napovedujejo vse ciljne spremenljivke. Z uporabo NIIP, lahko za vsako ciljno spremenljivko združujemo samo napovedi tistih dreves v ansamblu, ki so se to spremenljivko učila. Rezultati pokažejo, da imajo ansambli DNR z uporabo NIIP višjo napovedno moč, kot trenutno najboljše metode za reševanje nalog VCR, ter primerljivo napovedno moč, kot trenutno najboljše metode za reševanje nalog VOK.

Ansambli DNR, sicer najmočnejši napovedni modeli za VCN, so v splošnem nerazložljivi. Posledično se uporabljajo modeli z manjšo napovedno močjo, katerih obrazložitev je mogoča. Metoda za grajenje ansamblov pravil za VCR gradi množico kandidatnih pravil z uporabo ansamblov DNR, katerih število je po optimizaciji občutno manjše. Metodo razširjamo tako, da kandidatna pravila gradimo z NIIP ansambli. Rezultati pokažejo, da predlagana metoda dosega primerljivo napovedno moč v primerjavi z znanimi metodami za učenje pravil za VCR.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Abbreviations

| | | |
|---|---|---|
| Bag | ... | Bagging ensemble of PCTs |
| Bag-ROS | ... | Bagging ensemble of PCTs with ROS |
| ET | ... | Ensemble of extremely randomized PCTs |
| ET-ROS | ... | Ensemble of extremely randomized PCTs with ROS |
| FIRE | ... | Fitted rule ensemble |
| FIRE-ROS | ... | Fitted rule ensemble with ROS |
| MLC | ... | Multi-label classification |
| MTP | ... | Multi-target prediction |
| MTR | ... | Multi-target regression |
| PC | ... | Predictive clustering |
| PCR | ... | Predictive clustering rule |
| PCT | ... | Predictive clustering tree |
| RF | ... | Random forest ensemble of PCTs |
| RF-ROS | ... | Random forest ensemble of PCTs with ROS |
| ROS | ... | Random output selections |
| SOP | ... | Structured output prediction |
| TDI | ... | Top-down induction |

# Chapter 1

# Introduction

Mankind has technologically advanced from being able to understand the potential of using computers to an era where a large portion of business is done online and many people cannot imagine their lives without smartphones. The times when one computer occupied an entire room are long over and several orders of magnitude more powerful computers now fit into our pockets. Such technological breakthroughs and changes in global social dynamics were bound to introduce many opportunities and challenges.

Nowadays digitally centered organizations understand that perpetually monitoring and improving on all key areas as well as experimenting and occasionally even failing are the necessary ingredients for success. All mentioned activities are at least partially supported by niche software systems and analysts because correctly interpreting and leveraging the increasingly complex and abundant data has never been so overwhelming and important as it is today. Substantial efforts have been made to address this problem but the technological progress keeps pushing the boundaries of the possible. Today, it is no longer only a matter of using the right tools for the job: It is also (and very much so) about speed. The uncharted territories are expanding faster than they are being tackled and the rate at which data is generated and upon which decisions have to be made is non-trivially high. Another important aspect we need to mention is the global trend of tailoring services and products to individuals. All of this directly influences the amount of generated data that further deepens the problem of scale and puts additional pressure on the data analytics teams.

For quite some time, it is clear that automated approaches for data analysis are needed. However, mere automation is not enough to cope with the dynamics of the modern world. The potential of using *artificial intelligence* (Russell & Norvig, 2009) (AI) is greater than ever and some applications of it are well-known to the general public, such as the chess match between the then reigning world chess champion Garry Kasparov and Deep Blue, a chess-playing computer. AI is often equated with physical objects, such as humanoids, self-organizing industrial robotic systems or self-driving vehicles. But in its essence, AI boils down to computer algorithms that perceive the environment and utilize mathematical and statistical approaches to make decisions that ultimately lead to achieving the predefined goals.

*Data mining* (Witten & Frank, 2005) is the process of exploiting the available data in order to extract useful information, which often includes using *machine learning* (ML) (Mitchell, 1997) approaches. This dissertation falls within the scope of ML, a subfield of AI, concerned with the development and evaluation of computer algorithms that are able to learn from experience. According to Džeroski (2007a), data mining tasks are divided into four major groups: probability distribution estimation, predictive modeling, clustering and pattern discovery. In this dissertation, we are particularly interested in predictive modeling,

which focuses on finding models that predict specific properties of data instances. In ML, these properties of interest are often called *targets* and are used to guide the model learning procedure. The guidance is commonly known as supervision, because the model-building decisions are made based on the values of properties of interest, i.e., models are tailored towards making predictions for specific properties of interest.

In *supervised learning*, a major area of ML, the goal is to exploit the provided labeled data for learning a predictive model, which can provide predictions on previously unseen data. Traditionally, models predict a single scalar value, be it numerical or categorical. Many real-life problems can be represented as single target (value) predictive modeling tasks and a multitude of ML algorithms has been developed to solve them. However, all challenges are not easily mapped into problems of predicting a single scalar value. The need for more advanced analytical approaches has thus emerged, necessitating the development of methods that can model data with higher complexity on the output side. In that sense, we need to transition from classical predictive modeling, where a single value is predicted, to the more challenging area of structured output prediction (SOP). In this dissertation, we focus on the tasks of predicting multiple targets, i.e., multi-target prediction (MTP).

There are many practically relevant MTP problems. As an illustration, we highlight one problem from the domain of environmental studies and another one from the domain of image recognition. In the first example, water samples are collected from Slovenian rivers (Džeroski, Demšar, & Grbović, 2000). Each water sample is analyzed and the information about the chemical properties of the sampled water and abundances of animal and plant species are extracted. The abundance of animal or plant species is represented with a numeric value – one numeric value for each species. The goal is to find a model that predicts abundances of all plant and animal species based on the chemical properties of the water. The MTP task from this example is commonly known as multi-target regression (MTR), because the goal is to predict a tuple of numeric values.

In our second MTP example, the goal is to detect objects shown in an image. Here, the target attributes are boolean values that indicate the presence or absence of an object in the image. Consider, as an example, a set of wildlife images, where the goal is to detect which animal species are shown in the image. Note, that more that one animal can be present in the image. Each species is represented with a boolean target attribute, e.g., bear (yes/no), fish (yes/no), etc. Such MTP tasks are known as multi-label classification (MLC) tasks, because the image is either labeled (tagged) with a given animal species or not. Ontologically, MTP tasks fall within the scope of structured output prediction. Solving such tasks is important in many domains and has been listed as one of the most challenging problems in ML (Kriegel et al., 2007; Q. Yang & Wu, 2006).

## 1.1 Motivation

Machine learning is becoming common in many domains and a plethora of methods exists that solve the traditional tasks of supervised learning, namely classification and regression. A similar trend is observed with methods for solving MTP tasks. The main premise behind MTP (and SOP in general) is the potential relatedness between the components in the *target structure*. The building blocks of the target structure are not considered isolated from each other, i.e., they are in relation to each other. Methods for SOP usually assume such relatedness and directly aim to exploit it in order to achieve better predictive performance. In this dissertation, we challenge this assumption. Before we can explain the idea, we need to mention the two contrasting approaches to solving MTP tasks: local and global.

The local approach is the simplest way of solving MTP tasks. It generates models for

predicting each component of the target structure separately, thus ignoring the relatedness between them. This gets the job done, but has two main deficiencies. The first one is the computational cost, which increases linearly with the number of target attributes, because one model is learned for each target attribute. The second one is the fact that this approach does not exploit the potential relations between the target attributes. However, this approach does also have its benefits, such as the ability to use any of the many appropriate existing off-the-shelf methods to learn the individual predictive models. Models yield predictions for the corresponding target attributes, which are then combined into the final prediction – a vector, where each component represents the prediction for one target attribute.

With the global approach, a single model for predicting all components of the target structure simultaneously is learned. The methods that are capable of doing this are explicitly designed for solving MTP tasks and assume that all target attributes are interrelated. This results in learning only one predictive model, which can reduce overfitting, that yields a prediction for the whole target structure. Using the global approach typically produces predictive models in a more efficient manner in terms of computational complexity, because only one model is learned instead of many. It also explicitly aims to exploit the relations between the components of the target structure, potentially improving the predictive performance over local approaches.

We generally do not learn models that predict semantically unrelated things, so the relatedness assumption makes sense. But, does this still make sense when the relations are weak or even nonexistent? Assuming that all targets are always related seems optimistic. The idea of this dissertation is to explore what happens if the relatedness assumption is relaxed. Instead of assuming that all target attributes are related, we want to design an approach that considers subsets of target attributes and resides between local and global approaches.

The main goal of predictive modeling is to build models with good predictive performance. To that end, ensemble models are often used to obtain a premium predictive performance. An ensemble model is a set of base predictive models. In ensemble learning, predictions are made by combining predictions of individual base predictive models. Generally, two types of ensemble models exist: homogeneous, where one algorithm is used on different variants of provided data, and heterogeneous, where different algorithms are used on the same data. Homogeneous ensembles typically sample the provided data in terms of data instances, attributes or both.

Recently, the field of MTP has gained substantial attention and many ensemble methods have been extended towards MTP. Within this dissertation, we focus on those ensemble methods that use decision trees as base predictive models. In particular, bagging and random forest ensembles (Kocev, Vens, Struyf, & Džeroski, 2013) as well as ensembles of extremely randomized decision trees (Kocev & Ceci, 2015) are used to learn the tree ensemble models for MTP.

This dissertation presents a tree ensemble extension method for MTP - Random Output Selections (ROS) - where randomly selected subsets of target attributes are considered by each tree in the ensemble. This is analogous to the random subspaces method (Ho, 1998), which builds a forest of decision trees, where each tree considers a randomly selected subset of input attributes for learning. This approach has not yet been considered in the MTP setting.

Rule models are one of the most expressive and interpretable model types. A rule learning algorithm for MTR was proposed by Aho, Ženko, Džeroski, and Elomaa (2012) (Fitted rule ensembles for MTR – FIRE for MTR) that internally uses random forest ensembles of PCTs to generate candidate MTR decision rules from MTR decision trees.

We extend the existing FIRE algorithm to consider candidate rules obtained with a forest of MTR decision trees with ROS extension. With rule ensembles a small amount of predictive performance is sacrificed in favor of interpretability.

## 1.2 Goals, Hypotheses and Methodology

The goals of this dissertation are divided into two categories: (i) the design and implementation of new algorithms, and (ii) the evaluation of the implemented algorithms. The ROS ensembles, by design, operate in the context of multi-target prediction. The goals of the dissertation are defined in more detail below.

**Design and implementation goals**

**Goal 1** Adapt and implement the existing TDI of PCTs algorithm to induce single PCTs that can use only a subset of the target structure components in their search heuristics.

**Goal 2** Extend and implement decision tree ensemble methods (bagging and random forests of PCTs and ensembles of extremely randomized PCTs) with ROS, to learn PCTs that focus on subsets of the target structure.

**Goal 3** Extend and implement the FIRE (fitted rule ensembles) for MTR method that uses ensembles of PCTs to generate candidate rules, to use ROS ensembles of PCTs.

**Goal 4** Implement a novel mechanism for ensemble voting, i.e., for combining the predictions of individual models in ROS ensembles, where individual ensemble constituents vote only for the target components that have been used in the search heuristic during learning.

**Evaluation goals**

**Goal 5** Evaluate the ROS ensemble extension in the context of PCT ensembles, constructed by using bagging and random forests of PCTs and ensembles of extremely randomized PCTs.

**Goal 6** Evaluate the FIRE rule ensemble method for MTR with bagging and extremely randomized PCTs (not using the ROS extension).

**Goal 7** Evaluate the ROS ensemble extension in the context of rule ensembles (FIRE for MTR method), constructed by using bagging and random forests of PCTs and ensembles of extremely randomized PCTs.

We explore the effects of using random subsets of the output space with different machine learning model types. Based on that, we form several hypotheses, which address different aspects of the proposed approach. The hypotheses investigated in this dissertation are:

**Hypothesis 1** The existing supervised ensemble methods for MTP available in the predictive clustering framework can be adapted to consider subsets of the output space.

**Hypothesis 2** The ROS extension improves predictive performance of tree and rule ensembles for MTP.

**Hypothesis 3** The optimal output subspace size depends on the considered domain/dataset.

In order to achieve the above goals of the dissertation and confirm or reject the hypotheses, we use the following methodology:

**Design and implementation**

1. In order to adapt the existing algorithms for TDI of PCTs, we change the way the impurity is calculated. We rewrite the part of the heuristic function that calculates the variance reduction, making it consider only a specific subset of the target components.

2. We extend the existing tree ensemble methods for MTP with ROS, by changing how individual PCTs in the ensemble are learned. Originally, all PCTs in the ensemble predict the whole target structure. We change this by randomly sampling the output space (similar to what the random subspaces algorithm does in the input space) and then provide the heuristic function with only the sampled subspace of the output space to learn from. The learned PCTs are able to predict only the parts of the target structure used in learning, as well as the whole target structure.

3. We extend the existing MTR rule-learning system FIRE in several directions. First, we replace the tree building method with one that uses the ROS extension. Then, we modify the calculations of covariance within FIRE to take into account the fact that (depending on the parameters) not all rules give predictions for all target attributes: Namely, FIRE calculates and uses the covariances between the predictions of different rules within the ensemble.

4. The ensemble methods considered are bagging (Breiman, 1996), random forests (Breiman, 2001) and extremely randomized trees (Geurts, Ernst, & Wehenkel, 2006). All three ensemble methods have been extended towards SOP in (Kocev & Ceci, 2015; Kocev et al., 2013). We develop extended versions of these. First, we implement a parametrized sampler of the output space, where the user is able to select the amount of original output space to sample. Then, we integrate the sampling into the ensemble learning procedure by adding an additional parameter to the induction of individual PCTs in the ensembles, i.e., the sampled output space.

5. The fitted rule ensembles (FIRE) algorithm for MTR internally learns a random forest ensemble of PCTs that is used to generate candidate rules. We introduce ROS-extended random forest ensembles as well as bagging ensembles of PCTs and ensembles of extremely randomized PCTs as methods to generate candidate rules.

6. For both tree and rule ensemble methods for MTP, we implement a novel base model aggregation function, i.e., voting mechanism. With tree ensembles, the individual base models get the capability to vote for only those components of the target structure that were used during the learning of that particular base model. The final predictions are, as is the case with the original ensembles, the arithmetic mean of all the votes. Now, however, the arithmetic mean for each individual target component is calculated only for the predictions of the base models that use that target component during learning. With rule ensembles, the novel voting mechanism is applied in an indirect manner. The individual candidate rules give predictions only for a subset of the target structure, but the optimization procedure decides whether such rules are relevant or not. In other words, we provide the optimization procedure with the possibility to select such rules, but we do not force it to use them.

7. All proposed methods are implemented within an open-source machine learning system CLUS, available at http://source.ijs.si/ktclus/clus-public/.

**Evaluation**

1. To evaluate the tree-based ROS ensembles, we use various output subspace sizes (a ROS parameter) as well as the original and the novel voting mechanisms. This is done for all three considered ensemble methods: bagging and random forests of PCTs and extremely randomized PCTs.

2. We compare the tree-based ROS ensembles to their original (non-extended) counterparts as well as their single-target variants and state-of-the-art methods.

3. To evaluate the rule-based ensembles with ROS, we use various output subspace sizes (a ROS parameter). To further explore the benefits of output space sampling, we allow the optimization procedure to select from not only fully-predictive rules (rules that predict the whole target structure), but also from partially-predictive rules (rules that predict only parts of the target structure).

4. We evaluate the FIRE rule ensemble method for MTR with candidate rules obtained from bagging ensembles of PCTs and ensembles of extremely randomized PCTs that do not use ROS, which has not been considered so far.

5. We compare the FIRE rule-based ensembles with ROS to their respective counterparts without ROS, as well as with other rule-based methods for MTR.

## 1.3 Contributions

In this dissertation, we propose a novel methodology for multi-target prediction. In particular, we address the tasks of multi-target regression and multi-label classification. The methods we propose are ensemble methods for MTP that learn ensemble constituents by considering subsets of the output space.

The contributions of this dissertation, described in this section, have been either published or submitted for publication at international conferences and in journals. The relevant publications are listed in the Bibliography section at the end of this dissertation. We summarize the contributions of this dissertation as follows:

**Contribution 1** *A novel methodology for learning tree ensembles with Random Output Selections (ROS) in the context of the predictive clustering framework.*

We propose a novel ensemble method for solving MTP tasks. In particular, we introduce a tree ensemble method for solving MTR and MLC tasks, which we call Random Output Selections (ROS). The proposed ensemble method builds on top of the existing family of predictive clustering (PC) ensemble methods, where predictive clustering trees (PCTs) are used as base predictive models.

Two commonly used approaches to solving MTP tasks exist. The first approach is called local, because it learns a predictive model for each target variable. The second approach is called global, because only one predictive model is trained to predict all target variables. When learning ensembles, the local approach learns an ensemble for each target variable, whereas the global approach learns an ensemble of global predictive models. The proposed method is an ensemble approach that works on the spectrum between these two approaches, while still yielding a predictive model capable of global predictions. Individual

base predictive models in the ensemble are learned on randomly selected subsets of the original target space.

The proposed methodology also introduces a new way of aggregating predictions in an ensemble. Usually, all base predictive models in an ensemble take part in the overall ensemble prediction by contributing their respective predictions (votes) to the prediction aggregation function. The same logic transfers to ensembles for MTP: predictions are made by aggregating the votes of individual base predictive models on a per-target basis. In this dissertation, we propose to use ROS ensembles, where individual base models are learned on subsets of target variables. In addition to that, we also propose a new prediction aggregation function which calculates global predictions based on ROS subsets of target attributes. In particular, only those base predictive models that have learned a specific target attribute are allowed to contribute their votes to the overall prediction of the ensemble for that target attribute.

**Contribution 2** *A novel methodology for learning rule ensembles for multi-target regression with Random Output Selections (ROS) in the context of the predictive clustering framework.*

Fitted rule ensembles for MTR (FIRE for MTR) is an ensemble method within the predictive clustering framework, where base predictive models are predictive clustering rules (PCRs). The model is obtained in three steps. First, a forest of PCTs is learned. Second, the individual PCTs in the forest are decomposed into a set of candidate predictive clustering rules (PCRs). Optionally, linear terms are added to the set of candidate rules. The last step is optimization of weights of individual rules and linear terms in the candidate set. The result is a rule set, containing PCRs and linear terms with nonzero weights.

The FIRE model originally uses random forests of PCTs to generate candidate rules. In this dissertation, we propose to use ROS ensembles to generate the candidate rules. In addition to the ROS-extended random forest ensembles of PCTs, we also use bagging of PCTs and ensembles of extremely randomized PCTs to derive candidate rules.

FIRE models make predictions by summing the per-target predictions of triggered decision rules. Similarly to the tree ensembles, rule ensembles can also use different prediction strategies. In particular, there are three possibilities: (i) summing per-target predictions of decision rules that predict all target variables, (ii) summing per-target predictions of decision rules that predict only those target variables that were used during the learning of the respective decision rule, or (iii) the combination of both previous approaches. The idea is to let the optimization procedure decide which rules are beneficial and which are not.

The optimization procedure itself requires changes. The calculations of covariances between two decision rules, between two linear terms and between a decision rule and a linear term must be adapted accordingly. In particular, the new calculations must take into account that two decision rules do not necessarily share the same set of target variables when making predictions. It is therefore necessary to adjust these pairwise calculations so that only the overlapping target variables are included in the calculation. Consequently, two decision rules with completely disjoint sets of target attributes have zero covariance.

**Contribution 3** *Extensive empirical evaluation of the developed methods across benchmark problems from various domains, including a comparison to state-of-the-art supervised approaches for multi-target prediction, which shows better or comparable performance of the proposed approaches.*

We perform an extensive empirical evaluation of the proposed methods on a series of benchmark datasets, gathered from various application domains. In particular, we focus on the

predictive performance and size of the obtained predictive models. We use three different ROS-extended ensemble methods: bagging, random forests and ensembles of extremely randomized PCTs. All three ensemble methods are used in the context of learning tree ensembles for MTP as well as rule ensembles for MTR.

ROS-extended methods have parameters additional to those of the original ensemble methods, namely: target subset size and selection of the prediction aggregation function. Before we could compare the proposed method to the state-of-the-art method, we had to select the best performing parameters for ROS. We have done so, by evaluating the proposed methods on the benchmark datasets, and selecting the parameter values that, on average, performed best. We give recommendations for ROS parameter values for all three ensemble methods.

We show that tree ensembles extended with ROS can outperform their original counterparts. Moreover, we show that the tree ensembles perform best by using ensembles of extremely randomized PCTs extended with ROS. Rule ensembles originally use random forest ensembles of PCTs. We confirm that this ensemble method performs best also when extended with ROS.

We also compare the proposed methods to the state-of-the-art MTP methods. More specifically, the best performing tree ensembles with ROS for MTR are compared to other output space transformation methods. We selected the three most closely related methods to compare with. The first is the Random Linear Target Combinations method (Tsoumakas, Spyromitros-Xioufis, Vrekou, & Vlahavas, 2014), where the authors construct new target variables via random linear combinations of existing ones. The second and the third methods are two variants of ensembles with multi-output regression trees as base predictive models, proposed by Joly (2017). There, the authors use a similar approach to the one presented in this dissertation. However, instead of omitting parts of the target space, they transform it by using projections. Results show that the ROS ensembles outperform all other compared methods in terms of predictive performance, but the differences in performance are not statistically significant.

Tree ensembles with ROS for MLC are also compared to three methods for solving MLC tasks. The first method, called Random $k$-Labelsets (RAkEL) (Tsoumakas & Vlahavas, 2007), transforms a MLC task into a classical classification task by considering randomly selected sets of $k$ target attributes as a single class. The second method, proposed by Joly, Geurts, and Wehenkel (2014), uses a random forest ensemble of multi-output classification trees with random projections that transform the output space. The third method we compare with is a random forest ensemble of PCTs (Kocev, 2011). The results show that ROS ensembles perform comparably to the models obtained with the existing methods.

Rule ensembles with ROS for MTR (FIRE-ROS) were compared to predictive clustering trees (PCTs) (Blockeel, Raedt, & Ramon, 1998), predictive clustering rules (PCRs) (Ženko, 2007) and original FIRE rule ensembles. In addition to that, we also compared the proposed method to tree ensembles with ROS, random forest ensembles with random projections and Random Linear Target Combinations. Results show that FIRE-ROS outperform PCRs with statistical significance. PCTs and the original FIRE ensembles are outperformed without statistical significance. Other competing methods all produce models that perform better than FIRE-ROS rule ensembles. These results are in line with previous research related to FIRE ensembles.

## 1.4 Organization of the Thesis

The current chapter is introductory and starts with the motivation for this dissertation, followed by a general outline of the pursued goals and tested hypotheses. The methodology

used to achieve the set goals and to investigate the hypotheses is also described. Following these introductory preliminaries, the main scientific contributions are described. The remainder of the dissertation is organized as follows.

Chapter 2 describes predictive data mining. The description of this important and well-researched field of machine learning would not be complete without the formal definitions and descriptions of predictive data mining tasks. In order to aid this task, we also describe some data mining concepts that are used throughout this dissertation. We mention the classical single-target predictive data mining tasks and then move to the more general area of predicting multiple target attributes. This chapter also gives an overview of other research related to this dissertation.

Chapter 3 describes the predictive clustering (PC) framework. This chapter describes methods that are most closely related to this dissertation and form the basis for our work. The first part describes predictive clustering trees and their use for solving MTP tasks. The second part of this chapter is dedicated to ensemble learning. Specifically, we describe ensembles of predictive clustering trees and ensembles of predictive clustering rules. All methods described in this chapter are subject to adaptations as a direct result of this dissertation.

Chapter 4 describes the central part and main contribution of this dissertation: a novel approach for learning ensembles of predictive clustering trees with random output selections – ROS. The proposed approach can be placed between the two generally accepted approaches for solving MTP tasks: local and global. Here, we describe both approaches and how ROS ensembles differ from them. This chapter also includes the previously mentioned adaptations of the original tree ensemble methods towards using ROS. We conclude this chapter with a computational complexity analysis of the proposed approach.

Chapters 5 and 6 describe the evaluation of the proposed approach of tree-based ROS ensembles for solving the tasks of multi-target regression and multi-label classification, respectively. Here, we present the experimental design, the results of the experiments and their discussion.

Chapter 7 describes the approach of using ROS for learning rule ensembles. In particular, we take the fitted rule ensembles (FIRE) approach for multi-target regression and describe the necessary adaptations for ROS integration. The result of using this method is an ensemble of predictive clustering rules. The second part of this chapter describes the experimental evaluation of the FIRE-ROS approach. In particular, it presents the experimental design, results and their discussion.

Chapter 8 concludes this dissertation. We first present a summary of the dissertation and its original contributions. We then outline several directions for possible further work.

# Chapter 2

# Predictive Data Mining

The process of *data mining* (Witten & Frank, 2005) connects several technical areas such as statistics, databases and data storage related technologies, soft computing, pattern recognition, data visualization, language technologies, text mining and finally machine learning (ML) (Mitchell, 1997) with the goal to tackle the challenge of knowledge extraction from the provided data. More precisely, ML is a subfield of artificial intelligence (AI) (Russell & Norvig, 2009) aiming to design and develop computer algorithms able to learn from experience. Generally, ML methods expect some data to learn from and ultimately yield a model. The interpretation of the model as well as the model type is highly dependent on the type of ML task we are solving. In this dissertation, we focus on *predictive modeling tasks*, which are the ones most widely known and researched, next to the *clustering* and *pattern mining* tasks.

In predictive modeling, the goal is to induce a model capable of making predictions. The idea is to learn a model using the provided data and later use it for making predictions on unseen data. If we are predicting a discrete value from a predefined set of possible values or *classes*, the task at hand is (multi-class) classification. A special case of this task is binary classification, where the number of possible classes is exactly two. If the predicted value is numeric, the task is called regression. A wide range of real-life problems can be represented as one of the mentioned predictive modeling tasks and a multitude of ML algorithms have been developed to solve such predictive modeling tasks. In this dissertation, we consider more challenging problems, where several values need to be predicted. This particular area of ML is called multi-target prediction. The ML tasks belonging to this area are multi-target regression (MTR) and multi-label classification (MLC).

This chapter is divided into three sections. We begin with some preliminaries regarding the composition of the input data. In particular, we formally describe data types and how they form data examples, which ultimately constitute a dataset - the input to ML methods. Next, we give formal descriptions for the MTR and MLC tasks. Here, we also present two illustrative datasets often used for benchmarking. The last section in this chapter describes the related work.

## 2.1 Data Types, Data Examples and Datasets

ML methods require data to learn from. The data is available in its entirety at learning time. This type of ML is called *batch learning*. Another field of ML is concerned with online learning, i.e., learning data examples from high-frequency and theoretically infinite data sources. In this dissertation, we do not address such tasks. Therefore, the reader should assume the batch learning setting throughout this dissertation.

Nowadays, many different structured formats are used to transport, store and process

data. However, in its core, data is represented with data types. The purpose of this section is to provide formal definitions of data types. A generally accepted nomenclature is needed to describe the technical properties of data. The acceptability and generality are important not only from the data re-use point of view but also because it helps in determining the applicability of ML methods to datasets, i.e., ML methods are usually designed to solve one of many possible ML tasks. Here, we describe only a relevant subset of the general-purpose data types otherwise described in the OntoDT data type taxonomy, a part of OntoDM - ontology for data mining (Panov, Soldatova, & Džeroski, 2016). OntoDT is based on the ISO/IEC 11404, the international standard for representing data types in computer systems (International Organization for Standardization, 2007).

Technically, a *data example* is a record with one or more properties. Properties can be thought of as placeholders with certain constraints, which apply to the values that a given property of a data example can take. Each data property must therefore specify its own constraints on the values which is done through the use of data types.

Data types can be either primitive or generated. Primitive data types used in data mining are the `boolean` data type, the `real` and the `discrete` data type. A `boolean` data type can take only two values: a true ($\top$) or a false ($\bot$) value. A `real` data type takes values from the set of real numbers $\mathbb{R}$. A `discrete` data type can take the value of one of the items, specified by a predefined final set of possible values. This makes the discrete data type somewhat special, because it can only be fully defined when the possible values are specified. An example of a discrete data type is `discrete(A,B,C,D)`, denoting that a property, restricted to the values of this data type, can only store values `A`,`B`,`C` or `D`.

The difference between primitive and generated data types is that generated data types are defined partially or fully by primitive data types. There are many generated data types. In this dissertation, we will only define the `tuple` data type. A tuple is an ordered list of data types. An example of a tuple would be `tuple(real, discrete(A,B,C))`. This specifies a tuple that takes two values. The first value must be numeric and can take any value from the set $\mathbb{R}$. The second value of the tuple must be one of the three specified letters `A`, `B` or `C`. Examples of such tuples are: `(1.23, B)` and `(2.72, A)`.

Datasets are bags of data examples [1]. Examples that belong to a dataset share the same tuple data type. There are many ways of how data can be represented. Consequently, ML methods should have the ability to consume any given data representation, which is obviously intractable. In ML, this problem is often bridged by providing the learning data in a tabular data format. Such representation is easily understood by humans and can be efficiently processed by machines. Rows of the table represent different data examples, whereas the columns denote the properties of said examples. Moreover, in ML, columns of such datasets are typically referred to as *features* or *attributes*. Each attribute can take values from the assigned data type.

The attributes in a dataset do not have a specific purpose, for which they can or should be used. However, in predictive modeling, attributes are divided into two special disjunct sets. The first set contains all the attributes that need to be predicted. Traditionally, this set only contains one attribute but in this dissertation we address SOP tasks, where this set contains at least two attributes. We use the terms *target attributes* or *outputs* to refer to this set and its members. The other attributes belong to the set of *descriptive attributes* or *inputs* because they are used by the model to "describe" the decisions that ultimately lead to predictions.

[1] We purposely use the word *bag* instead of *set* because repetitions of data examples in the dataset are possible.

## 2.2 Predicting Multiple Outputs

Multi-target prediction tasks actually belong to a broader area of *structured output prediction* (SOP) (Džeroski, 2007a). Predictive modeling tasks that could be considered as SOP tasks are those, where the predicted value is not primitive. With this in mind, we are left with a vast number of possible combinations of predicted values. Semantically, predicting several concepts that are not in any way connected, does not give competitive advantage. In practice, however, anything goes and one can jointly model completely unrelated concepts. Nevertheless, the central assumption of SOP is that the predicted values are in fact parts of a structure. In particular, structures are seen as a set of well-connected building blocks that individually carry information but jointly represent a meaningful concept.

Unlike with predictive modeling tasks that consider only one predicted value, SOP tasks do not have special names. However, some SOP tasks are frequently tackled and consequently do have generally accepted and recognizable names, such as MTR and MLC. This dissertation focuses solely on MTP tasks which we describe next.

### 2.2.1 Predicting multiple continuous variables

The MTP task, where the goal is to predict several continuous values, is called multi-target regression (MTR). Many such methods exist and they are surveyed in Borchani, Varando, Bielza, and Larrañaga (2015). We define the MTR task as follows:
**Given:**

- A descriptive (input) space $X$, with tuples of dimension $d$, containing values of primitive data types, i.e., $\forall x_i \in X, x_i = (x_{i_1}, x_{i_2}, ..., x_{i_d})$,
- A target (output) space $Y$, with tuples of dimension $t$, containing real values, i.e., $\forall y_i \in Y, y_i = (y_{i_1}, y_{i_2}, ..., y_{i_t})$, where $y_{i_k} \in \mathbb{R}$ and $1 \leq k \leq t$,
- A set of examples $\mathsf{S}$, where each example is a pair of tuples from the descriptive and the target space, i.e., $\mathsf{S} = \{(x_i, y_i) | x_i \in X, y_i \in Y, 1 \leq i \leq N\}$ and $N$ is the number of examples in $\mathsf{S}$ ($N = |\mathsf{S}|$),
- A quality measure $\mathcal{M}$, which rewards models with high predictive accuracy and low complexity.

**Find:** A function $f : X \to Y$ such that $f$ maximizes $\mathcal{M}$. In this dissertation, function $f$ will be represented by ensembles of predictive clustering trees and ensembles of predictive clustering rules.

An example dataset used to benchmark MTR methods is presented in Figure 2.1. The data is about water quality of Slovenian rivers (D. Demšar et al., 2006), where water samples are used to determine the chemical properties of the water and abundance of certain plants and animals. The learning goal is to generate a model that predicts abundances based on the chemical properties of water samples.

### 2.2.2 Predicting multiple discrete variables

The MTP task, where the goal is to predict several binary values, is called multi-label classification (MLC). Many such methods exist and they are surveyed in Gibaja and Ventura (2014), Madjarov, Kocev, Gjorgjevikj, and Džeroski (2012). We define the MLC task as follows:

| | Descriptive attributes | | | | Target attributes | | | |
|---|---|---|---|---|---|---|---|---|
| # | $KMnO_4$ | $CO_2$ | $\cdots$ | $K_2Cr_2O_7$ | Baetis rh. | Tubifex sp. | $\cdots$ | Simulium sp. |
| 1 | 0.66 | 0.15 | $\cdots$ | 2.7 | 3 | 0 | $\cdots$ | 3 |
| 2 | 2.05 | 0.56 | $\cdots$ | 2.8 | 0 | 0 | $\cdots$ | 5 |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ |
| 1060 | 1.3 | 1.23 | $\cdots$ | 1.1 | 5 | 3 | $\cdots$ | 1 |

Figure 2.1: The Water quality dataset: An example dataset used for benchmarking MTR methods.

**Given:**

- A descriptive space $X$, with tuples of dimension $d$, containing values of primitive data types, i.e., $\forall x_i \in X, x_i = (x_{i_1}, x_{i_2}, ..., x_{i_d})$,
- A label space $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots, \lambda_q\}$ that holds tuples of length $q$, where each tuple contains only binary values, i.e., $\forall \lambda_i \in \{\top, \bot\}$ and $1 \leq i \leq q$,
- A set of examples $\mathsf{S}$, where each example is a pair of tuples from the descriptive and the label space, i.e., $\mathsf{S} = \{(x_i, y_i) | x_i \in X, y_i \in \mathcal{L}, 1 \leq i \leq N\}$ and $N$ is the number of examples in $\mathsf{S}$ ($N = |\mathsf{S}|$),
- A quality measure $\mathcal{M}$, which rewards models with high predictive accuracy and low complexity.

**Find:** A function $f : X \rightarrow 2^{\mathcal{L}}$ such that $f$ maximizes $\mathcal{M}$. In this dissertation, function $f$ will be represented by ensembles of predictive clustering trees.

An example dataset used to benchmark MLC methods is presented in Figure 2.2. The data is about detecting bird species in noisy environments from short audio recordings (Briggs et al., 2013).

| | Descriptive attributes | | | | Target attributes | | | |
|---|---|---|---|---|---|---|---|---|
| # | $audio_1$ | $audio_2$ | $\cdots$ | Location | Pacific wren | Dark-eyed Junco | $\cdots$ | Steller's jay |
| 1 | 0.016 | 0.039 | $\cdots$ | 1 | $\bot$ | $\bot$ | $\cdots$ | $\bot$ |
| 2 | 0.006 | 0.035 | $\cdots$ | 1 | $\bot$ | $\top$ | $\cdots$ | $\top$ |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ |
| 322 | 0.030 | 0.014 | $\cdots$ | 17 | $\bot$ | $\top$ | $\cdots$ | $\bot$ |

Figure 2.2: The Birds dataset: An example dataset used for benchmarking MLC methods.

## 2.3 Related Work

This dissertation focuses on predictive modeling. In particular, solving multi-target prediction (MTP) tasks. A generally accepted grouping of existing methods divides them into *local* and *global* methods (Bakır et al., 2007). This division stems from the approach that the methods take when solving MTP tasks. Recall that in MTP, the goal is to predict several target variables. The first group of methods tackles this problem by learning a separate predictive model for each target attribute. This group of methods is called local because, when learning a model for predicting a single target attribute, they purposely ignore the available information contained in other target attributes, i.e., they operate locally. An opposite approach is taken by the global methods that learn a single model for all target attributes. Such an approach can be advantageous because it can take advantage of potential relations between the target attributes and can thus lead to models with better predictive performance.

In this section, we give an overview of methods for multi-target prediction. Specifically, we describe the methods for multi-target regression in Section 2.3.1 and methods for multi-label classification in Section 2.3.2.

### 2.3.1 Multi-target regression

Recently, a plethora of methods have been proposed that address the multi-target regression (MTR) task. MTR, also referred to as multi-target, multi-variate or multi-response regression, is a machine learning task, where the goal is to predict multiple real-valued variables simultaneously. Borchani et al. (2015) divide MTR methods into two categories: *problem transformation methods* and *algorithm adaptation methods*.

#### Problem transformation methods

These methods transform the initial multi-target problem into several single-target problems. Each single-target problem is then modelled separately and the predictions of individual models concatenated to form a multi-target prediction. This straightforward method is often referred to as one-vs-all or binary relevance. The main drawback of this approach is that the models are learned individually. This makes exploiting potential relations between the target variables impossible. However, a very nice property that this approach exhibits is that any existing regression method can be applied to model individual target variables. Another benefit of this local approach is the possibility to add target variables without having to rebuild all models.

In the work of Spyromitros-Xioufis, Tsoumakas, Groves, and Vlahavas (2012, 2016), the ideas from several MLC approaches (Godbole & Sarawagi, 2004; Read, Pfahringer, Holmes, & Frank, 2009) have been transferred to MTR. In multi-target regressor stacking (MTRS), a ST approach is applied in the first phase. In the second phase, the training dataset is augmented by adding additional input variables. The values of original input variables remain untouched. The values of the newly generated variables are obtained by taking the original input variables and applying the ST models on them. The predictions of the ST models on the $j^{th}$ example are added as values of the new input variables to the $j^{th}$ example. The number of added variables is the same as the number of target variables in the original dataset. At this point, another round of training is applied. Authors believe that this stacking approach corrects the errors made by the training in the first phase. The second approach that the authors propose are regressor chains (RC), where ST models are chained, i.e., learned in a particular sequence. The chaining sequence is a random permutation of target variables. For every item in the chain a ST model is learned.

Here, similarly as with MTRS, predictions of ST models are added to the training data before learning the next ST model (fist ST model uses only the original input variables). In addition to that a corrected variant was proposed (RCC), where cross-validated estimates were added instead of actual predictions. Results showed that the order of the target variables in the chain is important, which lead to another extension of this method, this time in the form of an ensemble. Each ensemble member was learned with the RC/RCC methods, but this time, chain sampling was done for each ensemble member separately. The approaches are called ensembles of RC/RCC (ERC/ERCC).

In the work of W. Zhang, Liu, Ding, and Shi (2012), a multi-output support vector regression method is proposed. This approach builds a multi-output model that uses the vector virtualization method that captures correlations between the target variables. The multi-target problem is transformed into a single-target problem by introducing additional input variables. The number of new variables $d$ is the same as the number of original target variables. This transforms one example from the original dataset into $d$ examples in the new dataset. In addition to that, the values of new variables are all zero, except for the value $d_i$, where $i$ corresponds to the $i^{th}$ original target variable. The generated single-target problem is then solved by applying the least-squares support vector regression.

Tsoumakas et al. (2014) propose an ensemble method called Random Linear Target Combinations. They propose to construct new target variables via random linear combinations of existing ones. The output space is transformed in such a way that each linear combination consists of $k$ original output features. Each combination is then considered for learning one ensemble member. The transformation of the output space is achieved via a coefficient matrix filled with random values. This method approaches MTR locally but also models dependencies between target attributes in the process. It could be considered as a method working on the spectrum between the local and global approaches.

### Algorithm adaptation methods

Algorithm adaptation methods are global in their capability to handle multi-target tasks naturally, i.e., no transformation of the training data is needed. Methods of this kind can exploit the potential relatedness among the target variables in order to learn models with better predictive performance.

**Statistical methods.** The statistical approaches are considered to be one of the first attempts towards multi-output regression. The early work of Izenman (1975) introduces reduced-rank regression. Later, Brown and Zidek (1980) propose adaptive multi-target ridge regression – an extension of the standard single-target ridge regression (Hoerl & Kennard, 1970). Almost two decades later, Breiman and Friedman (1997) proposed the Curds & Whey method, where they address the problem of taking advantage of correlations between the target variables in order to gain predictive accuracy w.r.t. to taking the single-target regression approach for each target variable. Abraham, Tan, Winkler, Zhong, Liszewska, et al. (2013) present a framework for multiple output regression that preserves the relationships among the response variables while minimizing the residual errors of prediction by coupling regression methods with geometric quantile mapping. Recently, Gillberg et al. (2016) proposed a Bayesian reduced-rank regression method that tackles the problem of structured noise in MTR.

**Support vector machines.** Traditionally, support vector machines were used to model single-target classification problems. This has been extended towards regression problems with support vector regression (SVR). Many kernel-based methods for modeling multi-target regression problems have been proposed (Alvarez, Rosasco, Lawrence, et al., 2012;

Brouwer, Kubicki, Sofo, & Giles, 2014; Cai & Cherkassky, 2009; Chung, Kim, Lee, & Kim, 2015; Sánchez-Fernández, de-Prado-Cumplido, Arenas-García, & Pérez-Cruz, 2004; Vazquez & Walter, 2003; S. Xu, An, Qiao, Zhu, & Li, 2013; Yanyan Yang, Chen, & Dong, 2015; W. Zhang et al., 2012) as a consequence of extending the SVR to multi-target support vector regression. The listed methods mainly focus on the problem of modeling the dependencies between the target variables. For example, Vazquez and Walter (2003) adapted the Kriging method to exploit correlations between the targets, while Yanyan Yang et al. (2015) used an additional regularization term to capture the effect of correlations among the targets.

**Instance-based methods.** Pugelj and Džeroski (2011) have introduced a k-Nearest neighbour method extended towards structured prediction. This method can be applied to MTR as well as MLC. The authors show that the proposed method, although simple, performs comparably to more complicated methods such as decision trees and rules.

**Tree methods.** A substantial amount of tree-based methods have been proposed. One of the earliest works is by De'Ath (2002), where a well-known CART method (Breiman, Friedman, Olshen, & Stone, 1984) was transferred to multi-output prediction. The process of tree induction is identical to CART, only the impurity function was changed to be the sum of squared error over the multi-variate response.

Hothorn, Hornik, and Zeileis (2006) proposed a unified framework for recursive partitioning which embeds tree-structured regression models into a well defined theory of conditional inference procedures. Statistical tests are used internally to make decisions regarding tree growth. The method is applicable to single- and multi-target regression problems.

Several methods from the family of predictive clustering have been proposed. Struyf and Džeroski (2006) proposed a method that introduces constraints into learning multi-target regression trees. This approach is an adaptation of the work of Blockeel et al. (1998). The method is capable of inducing trees with the size or accuracy constraint. Constrained trees can be used to obtain better interpretations.

This work was later extended by Appice and Džeroski (2007), where they use model trees. Model trees are decision trees that contain models within leaf nodes. Typically, a linear model is used, but other more complex model types can be used. In this particular work, several linear models are possible in each leaf node. The learning process determines whether to split a node or introduce a new regression variable. Each linear model in the leaf node then uses the introduced regression variables.

Ensembles of multi-target regression trees have been proposed by Kocev (2011), Kocev and Ceci (2015), Kocev, Džeroski, White, Newell, and Griffioen (2009), Kocev et al. (2013). In particular bagging, random forests and ensembles of extremely randomized predictive clustering trees were used.

Joly et al. (2014) propose ensembles of multi-output regression trees, where each individual tree is built by using a projected output space. Their approach uses Johnson-Lindenstrauss lemma. If the output space projection matrix satisfies the lemma, the variance computations in the projected space will be $\epsilon$-approximations of the variance in the original output space. Gaussian, Rademacher, Hadamard and Achlioptas projections are used. The goal is to truncate the output space in order to reduce the number of calculations needed to find the best split, which is the main computational burden while building a decision tree. While learning the ensemble, each tree is given a different output space projection. They use two different ensemble methods: Random forests and Extra trees. A similar approach is proposed in Joly (2017), where they use gradient boosting with random

projections and multi-target regression trees as base models. The idea is to use random projections to automatically adapt to the output correlation structure at each boosting iteration.

Recently, Van Wolputte, Korneva, and Blockeel (2018) introduced multi-directional ensembles of regression and classification trees. This method is an extension of random forest ensembles towards multi-directional prediction. The base models can be (but do not need to be) multi-output decision trees. The idea is to build a general model that is capable of answering any possible query of the data of the form "predict Y from X".

**Rule methods.** Of all described MTR methods, rule methods are the least researched. To the best of our knowledge, two approaches exist. The first approach was proposed by Ženko (2007) and is called Predictive Clustering Rules (PCRs). As the name suggests, this method belongs in the predictive clustering family of methods. The proposed approach, loosely based on the CN2 method (Clark & Niblett, 1989), is a generalization of the standard sequential covering approach that uses a general-to-specific beam search to find candidate rules. The space of possible rule conditions is exhaustively traversed and each potential rule condition expansion is evaluated using a heuristic function. The heuristic function considers several rule quality measures which can be enabled/disabled according to the requirements. The rule quality measures used by the heuristic function are dispersion, to control for the accuracy of the rule, coverage, to make rules as general as possible, distance to existing rules, to not prefer rules that cover the same examples as existing rules, and prototype dissimilarity, which should be different from the dissimilarity of the whole training set. This method can be used for single- and multi-target regression/classification. The method is capable of learning ordered and unordered rule sets.

The other existing method is called Fitted Rule Ensembles and has been proposed by Aho, Ženko, and Džeroski (2009), Aho et al. (2012). This method also belongs in the family of predictive clustering methods. In contrast to PCRs, where the rules are learned directly, this method learns rule sets in an indirect manner. The process starts by learning an ensemble of predictive clustering trees (PCTs). Then, all PCTs in the ensemble are decomposed into PCRs and each rule is assigned an initial weight. At this point, an optimization procedure is used to optimize the weights of the rules to obtain a smaller, more concise rule set. Optionally, simple linear functions (linear terms) can be added to the rule set to improve overall predictive performance.

### 2.3.2 Multi-label classification

Compared to multi-target regression, multi-label classification (MLC) is a substantially more researched field. MLC is a machine learning task, where the goal is to predict multiple binary variables simultaneously. Tsoumakas and Katakis (2007) divide MLC methods into two categories: *problem transformation methods* and *algorithm adaptation methods*. Several other survey papers have been published (Gibaja & Ventura, 2014; Madjarov, Kocev, et al., 2012; Tsoumakas, Katakis, & Vlahavas, 2009; M.-L. Zhang & Zhou, 2014).

#### Problem transformation methods

**Binary relevance.** Binary relevance is the simplest approach to solving MLC tasks. The initial multi-label problem is decomposed into single-label problems. Then, each problem is solved separately. This approach does not account for possible label dependencies, it can introduce sample imbalance and is borderline useful in high dimensional output spaces. The listed disadvantages were the cause of criticism (Zhou, Tao, & Wu, 2012) but Read, Pfahringer, Holmes, and Frank (2011) highlight the advantages of using this method, such

as linear scaling with the increase of labels and the ability to add labels/models without losing previous efforts.

**Classifier chains.** The classifier chains B. Chen, Li, Zhang, and Hu (2016), Enrique Sucar et al. (2014), Read et al. (2009, 2011) method builds upon the binary relevance approach. It generates a model by chaining single-target (ST) models. The chaining sequence is a random permutation of labels. For every item in the chain a ST model is learned. The predictions of ST models are added to the training data before learning the next ST model (fist ST model uses only the original input variables). Recently, Alali and Kubat (2015) proposed a method that stacks two layers of classifiers. The BR models, representing the first layer, make predictions which are inputs to the classifier at the second layer. To address the scalability issues, Read, Martino, Olmos, and Luengo (2015) propose an approach that can be used to efficiently learn thousands of labels.

**Label powerset methods.** The label powerset approaches (Boutell, Luo, Shen, & Brown, 2004; Read, Pfahringer, & Holmes, 2008; Tsoumakas & Vlahavas, 2007) transform a MLC problem into a multi-class classification problem. Each possible subset of labels that appear in the dataset becomes a new meta-class. This approach has issues with scalability because label powersets grow combinatorially in size. The individual base models can be learned by using any of the existing multi-class classification methods. The shortcoming of scalability has been tackled by Read (2008) with pruning the powerset. Tsoumakas, Katakis, and Vlahavas (2008) approach this problem differently by organizing labels into hierarchies and then training classifiers that predict the labelsets for each node of the hierarchy. Another approach (Szymański, Kajdanowicz, & Kersting, 2016) exists, where the authors propose to extend the work of Tsoumakas and Vlahavas (2007) by not using the original random partitioning of subsets, but rather a data-driven approach. They use community detection approaches from social networks to partition the label space, which can find better subspaces than random search.

**Binary pairwise methods.** When using the binary pairwise method, a binary classification model is trained for each possible pair of labels (Fürnkranz, Hüllermeier, Loza Mencía, & Brinker, 2008). The individual models are learned from an augmented dataset where the examples labelled with the first label are deemed as positive and examples labelled with the second label as negative. A majority vote is applied over all label pairs to obtain predictions. This approach is computationally intensive which is why its use quickly becomes intractable. Madjarov, Gjorgjevikj, and Džeroski (2012) propose to use a two-stage approach, where the first stage learns BR models and the second one consists of pairwise models. Madjarov, Gjorgjevikj, Dimitrovski, and Džeroski (2016) also use a data-driven approach. They use label hierarchies which they obtain from hierarchical clustering of flat label sets by using annotations that appear in the training data.

### Algorithm adaptation methods

**Rule methods.** Ženko (2007) proposed Predictive Clustering Rules (PCRs). This method is applicable to MTR as well as MLC tasks and has been described with MTR methods. Loza Mencía and Janssen (2016) propose two approaches: a stacking and a separate-and-conquer approach to learn single-label rules. The condition part of a rule can contain tests on the predictions of other labels, making this an approach that models label dependencies.

**Bayesian networks.** The approach proposed by Wang, Wang, Wang, and Ji (2014) can be used to enhance the performance of any multi-label classification method. The idea is to model the label dependencies with a Bayesian network. Nodes in the network represent labels, connected by edges with conditional probabilities. The edges simulate label dependencies.

**Neural networks.** M.-L. Zhang and Zhou (2006) propose an adaptation of the standard backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986) by using an error function that ranks the labels according to presence within examples. M.-L. Zhang (2009) adapt the radial basis function neural network towards MLC. Recently, Yeh, Wu, Ko, and Wang (2017) proposed deep neural networks with multiple hidden layers.

**Instance-based methods.** The ML-$k$NN method was introduced by M.-L. Zhang and Zhou (2005, 2007) and is an adaptation of the method proposed by Cover and Hart (1967). This method was later extended by Liu and Cao (2015) which did not assume independence between labels and examples. Binary relevance was used together with $k$NN by Spyromitros, Tsoumakas, and Vlahavas (2008). A $k$NN method is used by Cheng and Hüllermeier (2009), where they add a separate step of using neighbouring instances to get their labels which are then fed into a logistic model. Pugelj and Džeroski (2011) proposed a method (described in the MTR section) also applicable to MLC.

**Support vector machines.** Many approaches use some kind of combination of support vector machines and binary relevance (Boutell et al., 2004; W.-J. Chen, Shao, Li, & Deng, 2016; Elisseeff & Weston, 2002; E. C. Gonçalves, Plastino, & Freitas, 2013; T. Gonçalves & Quaresma, 2004) A recent improvement (J. Xu, 2014) of the ranking-based support vector machine substantially contributed to the applicability of the method by reducing its computational time. Jayadeva, Khemchandani, and Chandra (2007) implemented a support vector machine method that allows for non-parallel separating hyperplanes.

**Tree methods.** Predictive clustering trees (Blockeel & De Raedt, 1998) support various SOP tasks, including MLC. A modified entropy definition was used with decision trees (Clare & King, 2001) to make predictions for multi-label gene expression data. The method De Comité, Gilleron, and Tommasi (2003a) produces sets of rules that can be viewed as a decision tree. Madjarov and Gjorgjevikj (2012) introduced a method that first learns a multi-label model tree, where each leaf node holds binary relevance SVMs for each of the labels. Gjorgjevikj, Madjarov, and Džeroski (2013) propose a combination of SVMs and decision trees, where a multi-output decision tree is trained and then SVMs are used to learn individual labels in the leaf nodes. Decision trees that capture local label dependencies at each node of the tree were proposed by (Al-Otaibi, Kull, & Flach, 2014; Al-Otaibi, Kull, & Flach, 2016). For a set of dependent labels, a single (global) tree is constructed, while for independent labels, one (local) tree per-label is built. Q. Wu, Ye, Zhang, Chow, and Ho (2015) learn decision trees that use SVMs internally. Each tree node trains a one-vs-all SVM classifier which is then used to evaluate candidate splits. This method has problems with scalability due to the computational costs of learning SVMs.

**Ensemble approaches.** Two extensions of the boosting algorithm have been proposed by Schapire and Singer (2000). Alternating decision trees (Freund & Mason, 1999) were adapted by De Comité, Gilleron, and Tommasi (2003b) for multi-label classification. Kocev et al. (2013) propose to use bagging and random forests ensembles of predictive clustering trees. Su and Rousu (2015) proposed an ensemble of maximum margin graph labelling

classifiers, where a graph is defined over a set of labels to model possible dependencies. The bagging method was recently modified by Q. Wu, Tan, Song, Chen, and Ng (2016). The proposed method automatically keeps track of relevant labels for given examples during the learning process.

## 2.4 Summary

This chapter begins with the introduction of the basic terminology related to machine learning. We describe data types, data examples and datasets – concepts that we use throughout this dissertation.

We introduce predictive modeling, where the idea is to learn a model using the provided data and later use the learned model for making predictions on unseen data. Typically, two predictive modeling tasks are addressed: classification and regression. The former is concerned with classifying examples into one of the predefined classes whereas the latter is concerned with predicting a continuous value.

Many real-life problems can be formulated as a regression of a classification task. However, more challenging problems exist where more than one value must be predicted. The area within machine learning, concerned with solving such problems of predicting multiple outputs, is called multi-target prediction (MTP). We introduce two specific and widely-studied MTP tasks: multi-label classification (MLC) and multi-target regression (MTR). The former is concerned with predicting presence of individual labels which are represented as binary values. The latter is concerned with predicting several continuous values.

We conclude the chapter with a short description of related work. In particular, we describe state-of-the-art methods for MLC and MTR.

# Chapter 3

# Predictive Clustering Framework and Multi-Target Prediction

The *predictive clustering* (PC) paradigm has been introduced by Blockeel (1998). It can be seen as a generalization of supervised and unsupervised learning. The two approaches are traditionally considered as two completely distinct machine learning tasks. However, there are supervised methods that partition the instance space into subsets, which makes it possible to interpret them as clustering approaches (Langley, 1996). Examples of models built with such methods are decision trees and rules. The primary goal of supervised learning is to build models capable of making predictions whereas in unsupervised learning, the goal is to group examples similar in nature into groups (clusters). The PC paradigm connects the supervised and unsupervised learning into one streamlined approach. Next to the obvious benefit of using one approach over two, the resulting clusters are also symbolically described.

Predictive modeling techniques explore the descriptive space of available data in order to find good models that can make predictions for unseen examples. Models are often built by using a *heuristic search function* that estimates the quality of model-constructing steps during learning [1]. It does so by querying the target space of provided data. In other words, the search through the descriptive space is *supervised* by values in the target space. If the querying part is omitted, the task becomes a clustering task, i.e., unsupervised. Many methods are capable of solving either supervised or unsupervised tasks but not both.

In Chapter 2, we already mentioned the two groups of descriptive ($\mathsf{A}_d$) and target ($\mathsf{A}_t$) attributes used in traditional predictive modeling which correspond to the descriptive and target spaces respectively. The PC paradigm introduces the third group of attributes called *clustering attributes*. The conceptual understanding of the learning process remains the same as with traditional predictive modeling. Now, however, the heuristic queries the clustering space instead of the target space, i.e., the resulting clusters are homogeneous in the clustering space and described within the descriptive space. Finally the predictions are calculated in the target space, completely detached from the learning process. Another important difference between the PC paradigm and traditional predictive modeling is that in PC, overlap between the three groups of attributes is allowed which opens up many new possibilities of knowledge extraction from data. In this dissertation, however, we do not address such scenarios. The reader should therefore assume that the groups of clustering and target attributes are equal and that they have zero overlap with the group of descriptive

[1] The words *heuristic search function* are often abbreviated to *heuristic*. This can sometimes be confusing, because the result of a heuristic search function (the actual calculated value) is also often called "heuristic". Although not encouraged, the meaning of the word *heuristic* can appear in several contexts. The actual meaning therefore depends on the context that the word is used in.

attributes. In other words: We train our models against the same set of attributes that are eventually predicted by the model.

The PC paradigm distances itself from specifying the data mining task. It does so by generalizing over all of them. The generalization is achieved by first recognizing that all single-value predicting tasks are in fact special cases of MTP tasks (Džeroski, 2007b). Next, the heuristic functions that guide the search through the descriptive space calculate many different candidate options. Usually, one of those options is then used to refine the current model [2]. At the level of solving a specific task, these calculations can no longer be abstract but need to be explicitly defined, e.g., classification and regression problems are handled differently. In order to become general, the PC paradigm introduces three non-optional parameters:

1. the *heuristic search function*, responsible for guiding the search process,
2. the *impurity function*, used to calculate the homogeneity of a given set of data instances,
3. the *prototype function*, applied in order to calculate the predictions for a given data instance.

For now, let us only state that the three mentioned parameters instantiate the learning method within the PC paradigm and adapt it to address the ML task at hand. The actual examples of parameter values will be explained in the following sections.

The PC paradigm has been initially implemented with generalized decision trees, called *predictive clustering trees* (Blockeel et al., 1998). Later, it has been extended towards rule learning with *predictive clustering rules* (Ženko, 2007). Both model types can also be used for ensemble learning (Aho et al., 2012; Kocev, 2011; Kocev & Ceci, 2015). All PC methods are implemented and readily available for use in the CLUS software package.[3]

The approaches mentioned in this chapter are the basis of this dissertation, which are later extended. In Section 3.1, we describe the predictive clustering trees. Sections 3.2 and 3.3 pertain to learning ensembles of predictive clustering trees and ensembles of predictive clustering rules for multi-target prediction respectively.

## 3.1 Predictive Clustering Trees for Multi-Target Prediction

Decision trees are one of the most recognizable, interpretable and widely used model types in ML, applied to solving classification as well as regression tasks (Breiman et al., 1984). In this dissertation, we use them in the context of PC paradigm. From the PC point of view, decision trees are a hierarchy of non-overlapping clusters and are consequently called predictive clustering trees (PCTs).

Figure 3.1 depicts a standard PCT which consists of two types of nodes: (i) split nodes and (ii) leaf nodes. The topology of a PCT is the same as the topology of a standard decision tree. Split nodes typically have one data inflow and two data outflows. Decision trees containing split nodes with more data outflows do exist, but are not used in this dissertation and are therefore not discussed. The uppermost split node is also called the *root node* as it represents the beginning of the decision tree. There, the inflow is not visible but it should be assumed that the data enters there. Split nodes contain *tests* or conditions, which partition the incoming data into smaller chunks. Nodes without tests

[2] A greedy search is often used in the heuristic functions. There, the best heuristic value determines the best option. However, other approaches exist, where more than one option is retained, e.g., beam search.

[3] CLUS software package is available at http://source.ijs.si/ktclus/clus-public/.

are called leaf nodes. They have only one data inflow and are responsible for specifying predictions for the examples that end up reaching them.

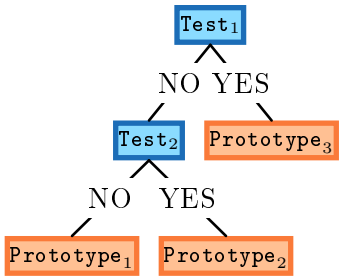



Figure 3.1: A standard predictive clustering tree. Blue and orange rectangles represent split nodes and leaf nodes respectively. The test condition $\texttt{Test}_1$ is in the root node. Each leaf node contains a prototype function for calculating predictions.

Once a PCT is trained, it can be used for making predictions. Predictions are made by using the tree in a top-to-bottom manner. A data example enters at the root node and travels through the tree until a leaf node is reached. The paths between two nodes are mutually exclusive directions that an example can take. The decision regarding which path an example will follow is based on the values of example's descriptive attributes. The decision process is such that the incoming examples are tested against the conditions in the split nodes. The outcome of each individual test can either be positive or negative, causing the example to take the positive (yes) or negative (no) path. Considering a numeric attribute $A$, an example of a condition in a split node could be $A > 2.72$, indicating that data examples with value of attribute $A$ higher than 2.72 should take the *yes* path and *no* otherwise. This process is known as *decision tree traversal*.

### 3.1.1 Learning a single predictive clustering tree

The main advantage of using PCTs over standard decision trees is that PCTs can be instantiated to solve several different tasks. In the introductory paragraphs of this chapter, we have mentioned that the PC paradigm introduces three non-optional parameters which adapt the general PC approach to the task at hand. Here, we demonstrate how PCTs can be used for solving the tasks of multi-target regression and multi-label classification by defining all three parameters.

PCTs are induced using the standard *top-down induction of decision trees* (TDIDT) algorithm (Breiman et al., 1984). The function PCT, depicted by Algorithm 3.1, takes as input a dataset $\mathsf{S}$, sets of descriptive ($\mathsf{A}_d$) and target ($\mathsf{A}_t$) attributes and a sampling function $\delta(\mathsf{X})$ that samples uniformly at random and without replacement attributes from the attribute set $\mathsf{X}$. The result of this algorithm is a predictive model called predictive clustering tree. The model consists of split nodes, containing tests on descriptive attributes $\mathsf{A}_d$, and leaf nodes, specifying predictions for the target attributes $\mathsf{A}_t$.

The induction starts with sampling of the descriptive attributes by using $\delta$ function. This step is ensemble-specific and is not a part of the standard PCT induction setup. We include it here because we later describe ensemble methods that use this algorithm to generate their base predictive models and some of the ensemble methods require such descriptive attribute sampling. This will be further explained in Section 3.2. Nevertheless, please note that $\delta$ is passed to the PCT function as a reference in order for it to be used within the PCT function. In other words: When we write $\delta$, we refer to the actual function. When we write $\delta(\cdot)$, we refer to applying $\delta$ to the input set $\cdot$ and expect as a result a set

**Algorithm 3.1:** The top-down induction of predictive clustering trees.

**Function:** PCT
**Input:** A dataset $\mathsf{S}$, a set of descriptive attributes $\mathsf{A}_d$, a set of target attributes $\mathsf{A}_t$, an attribute sampling function $\delta$
**Output:** A predictive clustering tree
$\mathsf{A'}_d \leftarrow \delta(\mathsf{A}_d)$
$(t^*, h^*, \mathcal{P}^*) \leftarrow \mathsf{BestTest}(\mathsf{S}, \mathsf{A'}_d, \mathsf{A}_t)$
**if** $t^* = none$ **then**
  **return** $\mathsf{Leaf}(\mathsf{Prototype}(\mathsf{S}, \mathsf{A}_t))$
**else**
  **foreach** $\mathsf{S}_i \in \mathcal{P}^*$ **do**
    $\mathrm{tree}_i \leftarrow \mathsf{PCT}(\mathsf{S}_i, \mathsf{A}_d, \mathsf{A}_t, \delta)$
  **end**
  **return** $\mathsf{Node}(t^*, \bigcup_i \{\mathrm{tree}_i\})$
**end**

of randomly sampled items from that set.

The next step of the PCT induction constitutes a greedy search encapsulated within the function BestTest and is depicted by Algorithm 3.2. The search for an acceptable split is done within the dataset $\mathsf{S}$ and is constrained to the descriptive attributes $\mathsf{A'}_d$ and internally evaluated against the target attributes $\mathsf{A}_t$. If a test is not found ($t^* = none$), the algorithm returns a leaf node, initialized to predict values for attributes $\mathsf{A}_t$, based on the currently examined data $\mathsf{S}$ and indicating that the search within this particular subspace of the original data $\mathsf{S}$ should stop. If a test is found, the BestTest function returns a non-empty test $t^*$, a heuristic score of said test $h^*$ and a partitioning $\mathcal{P}^*$. The partitioning is a direct result of dividing the data $\mathsf{S}$ according to the test $t^*$. The induction then continues with the algorithm recursively calling itself for every discovered partition in $\mathcal{P}^*$ in order to find additional tests. When the function eventually returns from recursion, it creates a node with test $t^*$ and child nodes $\mathrm{tree}_i$.

**Algorithm 3.2:** The search for the best test within the provided set of data examples and constraints, used with standard PCTs.

**Function:** BestTest
**Input:** A dataset $\mathsf{S}$, a set of sampled descriptive attributes $\mathsf{A'}_d$, a set of target attributes $\mathsf{A}_t$
**Output:** The best test $t^*$, heuristic score $h^*$ of test $t^*$ and the partitioning $\mathcal{P}^*$ induced by $t^*$ on dataset $\mathsf{S}$
$(t^*, h^*, \mathcal{P}^*) \leftarrow (none, 0, \emptyset)$
**foreach** attribute $a$ in $\mathsf{A'}_d$ **do**
  **foreach** possible test $t$ on attribute $a$ in $\mathsf{S}$ **do**
    $\mathcal{P} \leftarrow$ partitioning induced by $t$ on $\mathsf{S}$
    $h \leftarrow \mathsf{Impurity}(\mathsf{S}, \mathsf{A}_t) - \sum_{\mathsf{S}_i \in \mathcal{P}} \frac{|\mathsf{S}_i|}{|\mathsf{S}|} \mathsf{Impurity}(\mathsf{S}_i, \mathsf{A}_t)$
    **if** $(h > h^*) \wedge \mathrm{Acceptable}(t, \mathcal{P})$ **then**
      $(t^*, h^*, \mathcal{P}^*) \leftarrow (t, h, \mathcal{P})$
    **end**
  **end**
**end**
**return** $(t^*, h^*, \mathcal{P}^*)$

The default **heuristic search function** used with predictive clustering trees is the *impurity reduction* (IR) heuristic shown by Equation 3.1, where $\mathsf{S}$ is the dataset before it is partitioned ($\mathsf{S}_i$ are the partitions) and $\mathsf{Impurity}(X, Y)$ is the function measuring the impurity of dataset $X$ according to the target attributes in set $Y$. The idea is to reward those candidate splits during learning of the PCT that reduce the impurity of the resulting partitions the most.

$$\mathrm{IR} = \mathsf{Impurity}(\mathsf{S}) - \sum\nolimits_{\mathsf{S}_i \in \mathcal{P}} \frac{|\mathsf{S}_i|}{|\mathsf{S}|} \mathsf{Impurity}(\mathsf{S}_i) \tag{3.1}$$

The definition of impurity differs depending on the ML task but it generally connects to the concept of homogeneity between the data examples. When examples are similar enough for them to be indiscernible, the evaluated set of data examples has the highest possible purity. The impurity reduction heuristic is used by the $\mathsf{BestTest}$ function depicted in Algorithm 3.2.

### 3.1.2 Predictive clustering trees for predicting multiple continuous variables

Equation 3.2 defines the **impurity function** $\mathsf{Impurity}(\mathsf{S}, \mathsf{A}_t)$, where $\mathsf{S}$ represents a set of data examples and $\mathsf{A}_t$ represents the set of attributes, against which the calculations are made. In the case of predicting multiple continuous variables, the impurity is calculated as the arithmetic mean of variances of individual target attributes.

$$\mathsf{Impurity}(\mathsf{S}, \mathsf{A}_t) = \frac{1}{|\mathsf{A}_t|} \sum_{a}^{\mathsf{A}_t} \mathrm{Var}_a(\mathsf{S}) \tag{3.2}$$

$$\mathrm{Var}_a(\mathsf{S}) = \frac{1}{|\mathsf{S}|} \sum_{i=1}^{|\mathsf{S}|} (a_i - \overline{a})^2 \tag{3.3}$$

The variance of individual target attribute is calculated as shown by Equation 3.3, where $\mathrm{Var}_a(\mathsf{S})$ represents the variance of the attribute $a$ on dataset $\mathsf{S}$, $a_i$ represents the value of the attribute $a$ in the $i^{th}$ data example of dataset $\mathsf{S}$, and $\overline{a}$ represents the mean value of attribute $a$ in the dataset $\mathsf{S}$. The **prototype function** for the standard PCT predicting multiple continuous variables is defined as a vector $\hat{y}$, where each component of the vector ($\hat{y}_a$) represents a placeholder for the prediction for a target attribute $a$ ($a \in \mathsf{A}_t$). An individual component $y_a$ of the prototype vector is calculated as an arithmetic mean of values in the dataset $\mathsf{S}$ for attribute $a$, i.e., $\hat{y}_a = \overline{a}$. An example of a PCT that predicts multiple continuous variables is depicted in Figure 3.2.

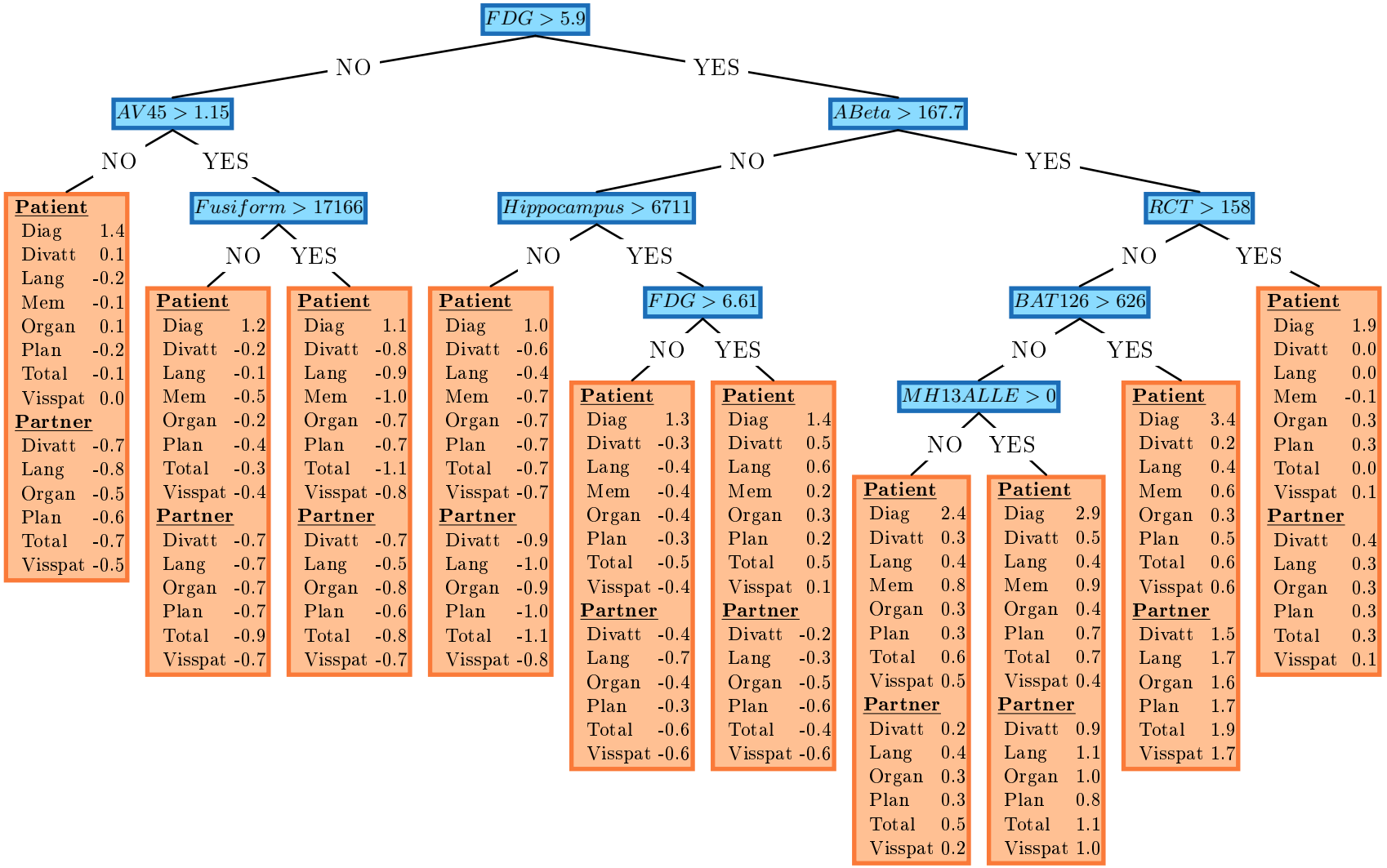



Figure 3.2: An example of a predictive clustering tree for predicting multiple continuous variables. This predictive clustering tree was obtained from the *ADNI* dataset (Alzheimer's Disease Neuroimaging Initiative), which is described in Breskvar, Ženko, and Džeroski (2015). It predicts the diagnosis and various scores in connection to everyday cognition of Alzheimer's patients and their corresponding study partners.

### 3.1.3 Predictive clustering trees for predicting multiple nominal variables

Equation 3.4 defines the **impurity function** $\mathsf{Impurity}(\mathsf{S}, \mathsf{A}_t)$, where $\mathsf{S}$ represents a set of data examples and $\mathsf{A}_t$ represents the set of attributes, against which the calculations are made. In the case of predicting multiple nominal variables, the impurity is calculated as the arithmetic mean of Gini indices of individual target attributes.

$$\mathsf{Impurity}(\mathsf{S}, \mathsf{A}_t) = \frac{1}{|\mathsf{A}_t|} \sum_{a}^{\mathsf{A}_t} \mathrm{Gini}_a(\mathsf{S}) \tag{3.4}$$

$$\mathrm{Gini}_a(\mathsf{S}) = 1 - \sum_{j=1}^{|Classes(a)|} p_j^2 \tag{3.5}$$

The Gini index of individual target attribute is calculated as shown by Equation 3.5, where $\mathrm{Gini}_a(\mathsf{S})$ represents the Gini index of the attribute $a$ on dataset $\mathsf{S}$, function $Classes(a)$ returns the classes contained within attribute $a$ of dataset $\mathsf{S}$, and $p_j$ represents the probability of a class $c_j$ ($c_j \in Classes(a)$). Probability is estimated with the proportion of examples in $\mathsf{S}$ belonging to the class $c_j$. If the number of classes in $|Classes(a)|$ equals two, modeling target attribute $a$ becomes a binary classification problem.

The **prototype function** for the standard PCT predicting multiple discrete variables is defined as a vector $\hat{y}$, where each component of the vector ($\hat{y}_a$) represents a placeholder for the prediction for a target attribute $a$ ($a \in \mathsf{A}_t$). An individual component $y_a$ of the prototype vector holds probabilities for each class $c_j$ ($c_j \in Classes(a)$). At this point, thresholding can be applied to determine the predicted class or the majority class can be used. In this dissertation, we use the latter. An example of a PCT that predicts multiple nominal variables is depicted in Figure 3.3.

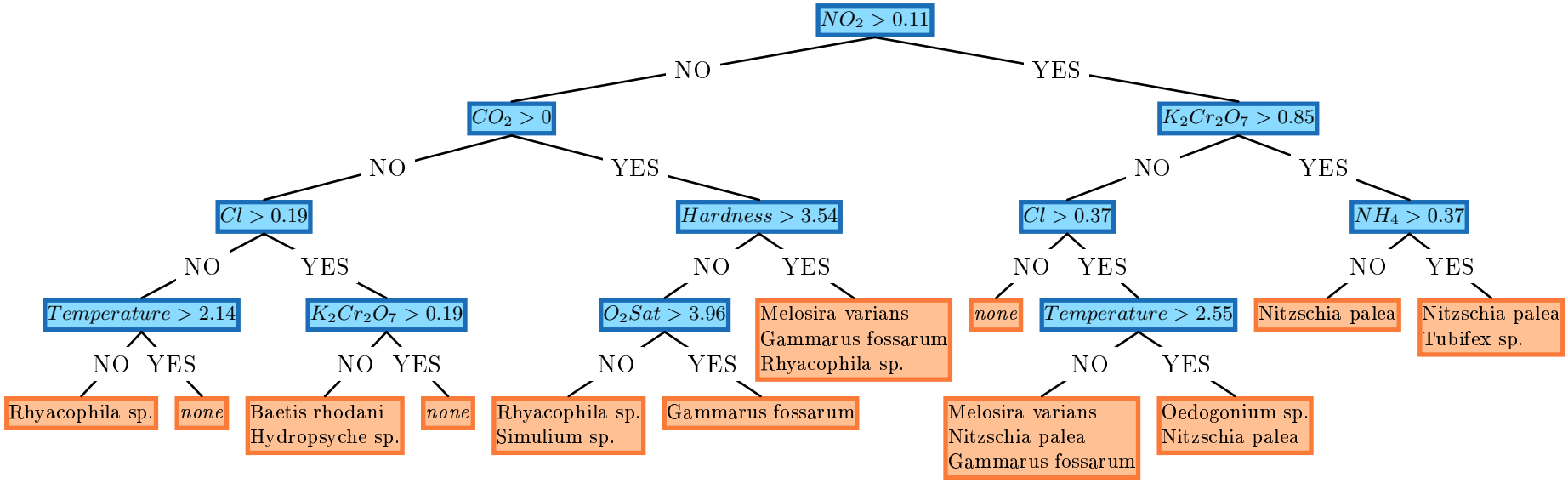



Figure 3.3: An example of a predictive clustering tree for predicting multiple discrete variables. This predictive clustering tree was obtained from the *Water quality* dataset, which is described in Džeroski, Demšar, and Grbović (2000). It uses chemical properties obtained from water samples to predict the presence of plant and animal species in Slovenian rivers. Leaf nodes with the content *none* do not predict any plant or animal species.

## 3.2 Ensembles of Predictive Clustering Trees for Multi-Target Prediction

An ensemble is a set of models, called base predictive models, where each model contributes to the overall prediction. It is a common practice to use ensembles to lift the predictive performance of the base predictive models and thus obtain premium predictive performance. This goal is achievable if the base predictive models are accurate and diverse (Hansen & Salamon, 1990). A predictive model is considered to be accurate when it performs better than random guessing. It does not make sense to use any kind of trained model, when flipping a coin produces better predictions.

As mentioned, base predictive models must also be diverse. Diversity of models is quantified indirectly. Two predictive models are diverse when they make predictions with different error rates. Note that two completely distinct predictive models can, on average, have the same error rate. The topological/logical differences between the two models with such properties do not constitute as diversity, making one of the two models redundant. Retaining both models does therefore not contribute to the overall diversity within the ensemble. In such cases, the Occam's razor principle should be applied and one of the two models should be removed or replaced with a different one.

We have defined ensembles as sets of models. However, we have not constrained the elements that can belong to such sets. Generally, two types of ensembles exist: homogeneous and heterogeneous. The former comprise models that have been learned by using the same learning algorithm. Examples of methods that produce such ensembles are Bagging (Breiman, 1996), Boosting (Freund & Schapire, 1996), Random subspaces (Ho, 1998), Random forests (Breiman, 2001) and ensembles of extremely randomized trees (Geurts et al., 2006). On the other hand, heterogeneous models can be combined within an ensemble with the stacking approach (Wolpert, 1992), where different learning algorithms are used on the same data.

Figure 3.4 depicts the flow of learning a homogeneous ensemble and obtaining predictions with it. The typical process is such that the preprocessed and sanitized dataset is presented to the ensemble learning method. Internally, the ensemble method distributes the data to different instances of the same base learning algorithm. Each instance produces a predictive model. Note that the input data can optionally be internally manipulated before it is presented to the learning algorithm. In particular, bootstrap aggregations of the dataset can be used to make perturbations within the dataset with the goal of making the learning more robust to potential noise in the data. When a prediction for an unseen data instance is to be made, each ensemble model is queried to produce a prediction for it. The predictions of the individual base predictive models are then combined by using an aggregation function. The aggregation function can internally treat predictions of individual base predictive models equally, or it can introduce bias, such as weighting of the individual ensemble models.

Typically, there are as many predictions as there are models in the ensemble although ensemble models exist, where predictions can be expected to arrive only from a subset of base predictive models in the ensemble. An example of such ensembles are rule ensembles, where individual rules in the ensemble can be viewed as base models. In those cases, individual rules are triggered only if the conditions are met, thus making it possible for a rule not to be triggered at all and consequently not yielding a prediction.

Ensemble models are generally not considered interpretable, because they hide the logic behind the predictions of base models and due to the fact that the predictions are aggregated, thus losing "individuality" of base models. However, ensemble models generally achieve better predictive performance than individual base predictive models, making

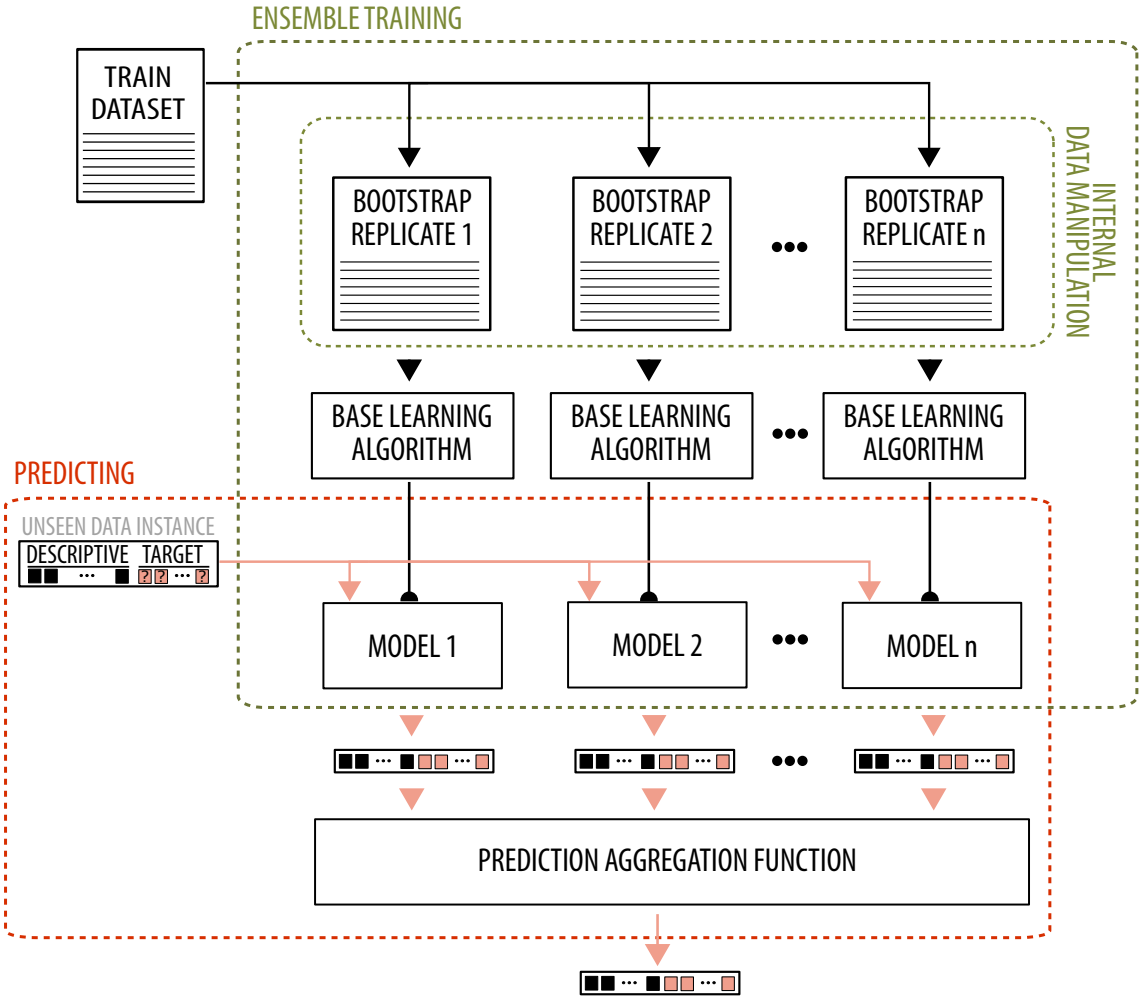



Figure 3.4: The process of learning a homogeneous ensemble and obtaining predictions with it. A training dataset is presented to the ensemble method (bounded with the black dotted rectangle) which produces several base predictive models. When an ensemble prediction for an unseen data instance has to be made, each learned base predictive model is queried for a prediction. Predictions of base predictive models are aggregated with an aggregation function and the ensemble produces a prediction.

them a popular weapon of choice. In contrast to what has just been stated, rule ensembles are an exception to this rule because they are interpretable. It will be made clear that the interpretability of rule ensembles comes with a price: The predictive power of rule ensembles is lower than that of tree ensembles. Regardless of ensemble type and method, another downside of using them is their computational complexity. The cost of learning and an ensemble model is the sum of the corresponding costs of all base models.

All mentioned ensemble methods were extended towards MTP within the PC framework. Consequently, PCTs and PCRs have also been used in the ensemble setting. PCTs have been used with bagging, random forests, random subspaces, bagging of subspaces (Kocev, 2011; Kocev et al., 2013) and ensembles of extremely randomized PCTs (Kocev & Ceci, 2015). Ensembles of PCRs were learned indirectly by first transcribing a random forest ensemble of PCTs to a set of PCRs and then using an optimization procedure to select only the relevant rules as members of the rule ensemble (Aho et al., 2012).

In this chapter, we describe the tree and rule-based ensemble methods for solving MTP

tasks. The presented methods are implemented within the predictive clustering framework and are presented in Sections 3.2 and 3.3, respectively.

An ensemble makes predictions by combining the predictions of its base models. When a prediction for an example has to be made, each base predictive model is queried for its prediction or *vote* for the given example. The votes are passed to the aggregation function, which then combines them and yields the final prediction of the ensemble. In general, the aggregation function is a parameter and there are many ways to combine the votes of the base predictive models. When solving regression tasks, the arithmetic mean is the most common. For classification tasks, majority and probability distribution voting (Bauer & Kohavi, 1999) are often used. In addition to how the votes are aggregated, the aggregation function can also use other information, such as preferences based on domain knowledge or (optimization derived) weights for individual base models (Kuncheva, 2004).

In our work, we use PCTs and variants thereof to solve MTP tasks. The aggregation functions therefore have to be appropriately extended towards MTP. For MTR tasks, the arithmetic mean is calculated on a per-target basis. The per-target probability distribution voting is used for MLC tasks. Note that all base learners provide predictions for all target attributes. The relevance of this statement will become evident in Chapter 4, where we describe ROS – the proposed approach.

The remainder of this section describes the three ensemble methods relevant to this dissertation. All three methods have been used in the context of predictive clustering trees.

### 3.2.1 Bagging

Bagging is short for bootstrap aggregating. It is an ensemble method that uses bootstrap replication (also known as bootstrapping) of the training data to introduce randomization in the learning dataset. A bootstrap replicate $\mathsf{S}^*$ of a dataset $\mathsf{S}$ is again a dataset that has been randomly sampled from $\mathsf{S}$. Sampling with replacement is repeated until both datasets are of equal size (i.e., $|\mathsf{S}| = |\mathsf{S}^*|$). Such perturbations of the learning set have proven useful for unstable base models, such as decision trees, but can generally be used by any model type. Base models use the whole descriptive space $\mathcal{D}$ while searching for the best splits. The descriptive attribute sampling is not required by the bagging ensemble method. We therefore define the descriptive attribute sampling function as: $\delta(\mathsf{A}_d) = \mathsf{A}_d$, i.e., no sampling is done and all descriptive attributes are available to all instances of the learning algorithm during learning. Algorithm 3.3 shows the bagging induction algorithm, which has been extended towards MTP and uses PCTs as base predictive models.

**Algorithm 3.3:** Bagging of PCTs.

**Function:** Bagging
**Input:** Dataset $\mathsf{S}$, number of base models $\mathsf{b}$, a set of descriptive attributes $\mathsf{A}_d$, a set of target attributes $\mathsf{A}_t$, a dummy attribute sampling function $\delta$
**Output:** Forest of PCTs
$\mathsf{F} \leftarrow \emptyset$
**for** $i \leftarrow 1$ **to** $b$ **do**
  $\mathsf{S}^* \leftarrow \mathsf{Bootstrap}(\mathsf{S})$
  $T_i \leftarrow \mathsf{PCT}(\mathsf{S}^*, \mathsf{A}_d, \mathsf{A}_t, \delta)$
  $F \leftarrow F \cup \{T_i\}$
**end**
**return** $\mathsf{F}$

### 3.2.2 Random forests

Random forest algorithm works similarly to bagging. This ensemble method also starts with the introduction of randomization in the instance space by applying the bootstrapping technique. However, random forest algorithm additionally introduces randomization in the descriptive attribute space. It does so by internally randomizing the PCT induction algorithm. The randomization is such that the BestTest function considers only a subset of randomly selected descriptive attributes from the set of all descriptive attributes $\mathsf{A}_d$. The process of random selection of descriptive attributes yields different subsets of descriptive attributes at each split node. The randomization is achieved by using the random attribute sampling function $\delta(\mathsf{A}_d)$. The recommended subset size of descriptive attributes for regression is $\sqrt{|\mathsf{A}_d|}$ and $\lfloor 0.1 \cdot |\mathsf{A}_d| \rfloor + 1$ for classification tasks. Algorithm 3.4 shows the random forest induction algorithm, which uses internally randomized PCTs as base predictive models.

**Algorithm 3.4:** Random forest of PCTs.

**Function:** RandomForest
**Input:** Dataset $\mathsf{S}$, number of base models $\mathsf{b}$, a set of descriptive attributes $\mathsf{A}_d$, a set of target attributes $\mathsf{A}_t$, an attribute sampling function $\delta$
**Output:** Forest of PCTs
$\mathsf{F} \leftarrow \emptyset$
**for** $i \leftarrow 1$ **to** $b$ **do**
  $\mathsf{S}^* \leftarrow \mathsf{Bootstrap}(\mathsf{S})$
  $T_i \leftarrow \mathsf{PCT}(\mathsf{S}^*, \mathsf{A}_d, \mathsf{A}_t, \delta)$
  $F \leftarrow F \cup \{T_i\}$
**end**
**return** $\mathsf{F}$

### 3.2.3 Extremely randomized trees

Extremely randomized trees or extra-trees are very unstable decision trees. It therefore only makes sense to use them in the ensemble setting. They differ from the other two ensemble methods by not perturbing the input data with bootstrapping and by randomizing the test selections. Extra-trees select at random $k$ descriptive attributes and for each of them randomly select a split point. This differs from the usual BestTest function which considers all possible split points for each descriptive attribute, before selecting the one with the best heuristic score. The random selection of $k$ attributes is achieved by using the random attribute sampling function $\delta(\mathsf{A}_d)$. The recommended number of attributes to sample is $k = |\mathsf{A}_d|$. Each randomly selected split is evaluated and the one with the best heuristic value $h^*$ is selected. Algorithm 3.5 shows the procedure for the induction of extra-tree ensembles, which has been extended towards MTP and uses extra-PCTs as base models.

At this point, we must also address the particularities of extremely randomized PCTs with respect to how they find best tests. The BestTest function depicted by Algorithm 3.2 is used for inducing standard PCTs, used within bagging and random forest ensembles. Ensembles of extremely randomized PCTs use a different BestTest function, depicted by Algorithm 3.6. Algorithm 3.7 describes the SelectRandomTest function, responsible for generating random tests for a given attribute $a$ within a dataset $\mathsf{S}$. If attribute $a$ is numeric, a random split point is selected between the lowest and highest value of attribute $a$ in the dataset $\mathsf{S}$. Based on this, a test is created, stating that the value of attribute $a$ should be lower than the selected cut point $a_{cut}$. If attribute $a$ is nominal, a (sub)set is

**Algorithm 3.5:** Random forest of extremely randomized PCTs.

**Function:** ExtraTrees
**Input:** Dataset $\mathsf{S}$, number of base models $\mathsf{b}$, a set of descriptive attributes $\mathsf{A}_d$, a set of target attributes $\mathsf{A}_t$, an attribute sampling function $\delta$
**Output:** Forest of extremely randomized PCTs
$\mathsf{F} \leftarrow \emptyset$
**for** $i \leftarrow 1$ **to** $b$ **do**
  $T_i \leftarrow \mathsf{PCT}(\mathsf{S}, \mathsf{A}_d, \mathsf{A}_t, \delta)$
  $F \leftarrow F \cup \{T_i\}$
**end**
**return** $\mathsf{F}$

randomly selected from the original possible values for attribute $a$ from $V_a$. Based on this, a test is created, stating that the value of attribute $a$ should be contained in the randomly selected set $V'_a$.

**Algorithm 3.6:** The search for the best test within the provided set of data examples and constraints, used with extremely randomized PCTs.

**Function:** BestTest
**Input:** A dataset $\mathsf{S}$, a set of sampled descriptive attributes $\mathsf{A'}_d$, a set of target attributes $\mathsf{A}_t$
**Output:** The best test $t^*$, heuristic score $h^*$ of test $t^*$ and the partitioning $\mathcal{P}^*$ induced by $t^*$ on dataset $\mathsf{S}$
$(t^*, h^*, \mathcal{P}^*) \leftarrow (none, 0, \emptyset)$
**foreach** attribute $a$ in $\mathsf{A'}_d$ **do**
  $t \leftarrow \mathsf{SelectRandomTest}(\mathrm{a}, \mathsf{S})$
  $\mathcal{P} \leftarrow$ partitioning induced by $t$ on $\mathsf{S}$
  $h \leftarrow \mathsf{Impurity}(\mathsf{S}, \mathsf{A}_t) - \sum_{\mathsf{S}_i \in \mathcal{P}} \frac{|\mathsf{S}_i|}{|\mathsf{S}|} \mathsf{Impurity}(\mathsf{S}_i, \mathsf{A}_t)$
  **if** $(h > h^*) \wedge \mathrm{Acceptable}(t, \mathcal{P})$ **then**
    $(t^*, h^*, \mathcal{P}^*) \leftarrow (t, h, \mathcal{P})$
  **end**
**end**
**return** $(t^*, h^*, \mathcal{P}^*)$

**Algorithm 3.7:** Random test selection when using extremely randomized PCTs.

**Function:** SelectRandomTest
**Input:** A dataset $\mathsf{S}$, an attribute $a$ to make the test for
**Output:** A randomly selected test $t$
$V_a \leftarrow \text{GetAttributeValues}(a, \mathsf{S})$
**if** IsNumeric(a) **then**
  $a_{max} \leftarrow \text{Max}(V_a)$
  $a_{min} \leftarrow \text{Min}(V_a)$
  $a_{cut} \leftarrow \text{RandomBetween}(a_{min}, a_{max})$
  **return** Test("$a < a_{cut}$")
**else if** IsNominal(a) **then**
  $V'_a \leftarrow \text{RandomNonEmptySet}(V_a, \mathsf{S})$
  **return** Test("$a \in V'_a$")
**end**
**return** *none*

## 3.3 Rule Ensembles for Multi-Target Prediction

The predictive clustering framework was extended towards rule learning with predictive clustering rules (PCRs, Ženko (2007)). This method is a generalization of the standard sequential covering approach that uses general-to-specific beam search to find candidate rules. It is capable of learning single-target as well as multi-target decision rules and has been applied to MTR as well as MLC tasks. PCRs have not been used in an ensemble setting. However, a different method for learning PCRs has been proposed and we describe this alternative method in the rest of this section.

Predictive learning with rule ensembles was first introduced with the method RuleFit, proposed by J. H. Friedman and Popescu (2008). The idea behind this approach is to generate a large pool of candidate decision rules, assign an initial weight of zero to all of them and then optimize the weights towards the smallest possible model with good predictive performance. The candidate rules are generated with an ensemble, built with the importance sampled learning ensemble – ISLE (J. H. Friedman, Popescu, et al., 2003). The authors used the proposed approach to learn rule ensembles for solving both regression and classification tasks. Moreover, they also extended their approach towards determining the importance of the descriptive (input) attributes.

### 3.3.1 Learning rule ensembles with FIRE

The RuleFit approach has been extended towards MTP. In particular, it has been extended towards MTR (Aho et al., 2012) and dubbed Fitted Rule Ensembles for MTR or FIRE for short. The rule model takes the form given by Equation 3.6. The model consists of three parts: (i) an average vector *avg*, providing predictions for all target attributes ($\mathsf{A}_t$), (ii) M weighted decision rules, triggered only when the conditions of individual rules are met, and (iii) optional weighted linear terms, one for every *descriptive-target attribute* pair $(t, d)$ where $d \in \mathsf{A}_d$ and $t \in \mathsf{A}_t$. Linear terms have been introduced in order to improve the predictive power of induced models.

$$f(x) = w_0 avg + \sum_{i=1}^{M} w_i r_i(x) + \underbrace{\sum_{t}^{\mathsf{A}_t} \sum_{d}^{\mathsf{A}_d} w_{(t,d)} x_{(t,d)}}_{\text{optional}} \tag{3.6}$$

**Algorithm 3.8:** The FIRE algorithm: Induction of rule ensembles for multi-target regression.

**Function:** FIRE
**Input:** Dataset $\mathsf{S}$, number of base models in the tree ensemble $\mathsf{b}$, a set of descriptive attributes $\mathsf{A}_d$, a set of target attributes $\mathsf{A}_t$, an attribute sampling function $\delta$
**Output:** An ensemble of rules and linear terms $\mathsf{F}$ with their corresponding weights $\mathsf{W}$

$\mathsf{RF} \leftarrow \text{RandomForest}(\mathsf{S}, \mathsf{A}_d, \mathsf{A}_t, \delta)$
$\mathsf{F} \leftarrow \text{ConvertForestToRules}(\mathsf{RF})$
$\mathsf{F} \leftarrow \mathsf{F} \cup \text{GenerateLinearTerms}(\mathsf{A}_d, \mathsf{A}_t)$ // Optional
$ERR_{min} \leftarrow \infty$
**for** $\tau \leftarrow 1.0$ **to** $0.0$ **with** *step* **do**
  $(W_\tau, ERR_\tau) \leftarrow \text{OptimizeWeights}(\mathsf{F}, \mathsf{S}, \tau)$
  **if** $ERR_\tau < ERR_{min}$ **then**
    $(W_{opt}, ERR_{min}) \leftarrow (W_\tau, ERR_\tau)$
  **else if** $ERR_\tau > threshold \cdot ERR_{min}$ **then**
    **break**
  **end**
**end**
$(\mathsf{F}, \mathsf{W}) \leftarrow \text{RemoveZeroWeightedTerms}(\mathsf{F}, W_{opt})$
**return** $(\mathsf{F}, \mathsf{W})$

FIRE models are induced in several steps which are depicted by Algorithm 3.8. First, a random forest (RF) ensemble of PCTs (Kocev et al., 2013) is learned. Next, a large quantity of decision rules is generated by transcribing individual PCTs from the tree ensemble to decision rules. A decision rule is obtained by concatenating all tests from the root node to the last non-leaf node of the PCT. The leaf node holds the predictions which are copied to the decision rule. Every PCT therefore decomposes into as many decision rules as there are leaf nodes in it. The acquired decision rules are given the same initial weight of zero and are put in the pool of candidate rules. Next comes the most computationally complex step of this approach: the optimization. Gradient directed optimization is used to set as many candidate weights to zero as possible by minimizing the squared loss, defined as $L_{sqrd}(f(x), y) = \frac{1}{2}(f(x) - y)^2$, where $f(x)$ and $y$ correspond to predicted and true values, respectively. However, this loss function is only applicable to single-target problems. Aho et al. (2012) transfer it to the multi-target regression scenario. The appropriate convex multi-target loss function is shown by Equation 3.7 and corresponds to the average loss over all target attributes. Before the optimization, the predictions of candidate rules must be normalized in order to equalize the importance of individual rules and targets. This is done in three steps: (i) The targets are zero-centered, (ii) the rule's predictions are scaled to equalize the effect of it on the optimization process and (iii) the differing scales of target attributes are normalized. These pre-processing steps, along with the optimization procedure itself, are explained in detail in Aho et al., 2012.

$$L(f(x), y) = \frac{1}{T} \sum_{t=1}^{T} L_{sqrd}(f_t(x), y_t) \tag{3.7}$$

### 3.3.2 Making predictions with FIRE models

Historically, decision rules take the form: IF *condition* THEN *consequence*. An example of a decision rule that predicts many numerical target attributes is depicted in Figure 3.5. The example is taken from the same dataset as the one used to showcase an example PCT in Figure 3.3. This time, however, the data was interpreted as being numeric[4].

Baetis rhodani, Tubifex sp., ..., Simulium sp.

IF $O_2 > 3.52$ AND $CO_2 > 0.27$ THEN $(0.61, 0.41, \ldots, 1.13)$

Figure 3.5: An example decision rule for predicting several numerical variables.

We can see that the example rule in Figure 3.5 predicts several numerical values, in contrast to standard decision rules, where only one value is predicted. The interpretation of this example decision rule is the following: If $O_2$ levels are above 3.52 and at the same time $CO_2$ levels are above 0.27, the rule should be triggered and the prediction should be made. Note that nothing is specified in case when the conditions of the decision rule are not fulfilled. With rule ensembles, it is certainly possible that the input data instance does not trigger any rules. In those cases, mechanisms must be in place to still provide predictions, because the model is incomplete if it does not always yield a prediction. FIRE models inherently solve this problem with the average prediction vector (the first term in Equation 3.6).

The model produced by FIRE comprises decision rules and optional linear terms. With tree ensembles, it is clear that each base predictive model will receive a data instance for which a prediction has to be made. The data instance will traverse through all base predictive models and each traversal will result in a prediction. In other words, all base models always yield a prediction for any kind of (valid) input. With rule ensembles, things are somewhat different. Let us examine how, by examining the structure of FIRE models through the eyes of making predictions.

We have already established that the FIRE models contain two mandatory terms and an optional one. The first term is the average prediction vector. The second and third terms represent the learned rules and optional linear terms, respectively. We can consider the average prediction vector as well as each decision rule or linear term in the ensemble as a separate base predictive model. The final prediction is the result of a function that sums the contributions of individual base predictive models, as opposed to ensembles of decision trees for MTR, where the aggregation function predicts the arithmetic mean over all predictions of base predictive models. Since the average prediction vector and linear terms do not have any conditions, they are always triggered, i.e., their condition is always evaluated to *true*. This immediately solves the problem of model completeness, because now, the predictions will always be yielded, regardless of whether any of the decision rules are triggered or not.

The prediction for a data instance comes in the form of a numerical prediction vector, where each component corresponds to one target variable. The prediction vector is calculated as the per-component wighted sum of the average prediction vector and predic-

[4]Many predictive modeling problems can be converted from regression to classification or vice versa. This does not make sense for all problems but this particular one allows for such transformations.

$(0.95, 0.73, 0.42, \ldots, 0.75, 0.74)$

rules:
- $+0.34 \cdot$ [IF $O_2 > 3.52$ AND $Cl \leq 0.38$ AND $CO_2 > 0.27$ THEN (0.61,0.41,...,1.13)]
- $-0.85 \cdot$ [IF $K_2Cr_2O_7 > 0.43$ AND $O_2Sat > 3.60$ THEN (0.6,0.57,...,0.64)]
- ...
- $+0.77 \cdot$ [IF $K_2Cr_2O_7 \leq 0.43$ AND $conduct \leq 2.46$ THEN (2.24,3.3,...,2.69)]

linear terms:
- $+0.15 \cdot (0, 0, x_8, 0, \ldots, 0)$
- ...
- $+0.5 \cdot (0, \ldots, 0, x_1)$

Figure 3.6: A part of the example rule ensemble model for predicting the abundance of animal and plant species in Slovenian rivers. The prediction vectors have been shortened in order to fit the model on the page. Due to space limitations, the majority of decision rules and linear terms has also been omitted.

tion vectors of individual decision rules and linear terms. Figure 3.6 depicts a part of an otherwise large rule ensemble.

### 3.3.3 Linear terms in FIRE models

Optionally, linear terms can be added to the set of candidate rules. A linear term is a simple linear model where only numeric descriptive attributes can appear in the function. One linear term is created for each target attribute-input attribute pair. Thus, the number of linear terms is at most $|\mathsf{A}_d| \cdot |\mathsf{A}_t|$. The linear terms are represented with the double sum in Equation 3.6.

$$x_{(t,d)} = (\underset{1}{0}, ..., \underset{t-1}{0}, \underset{t}{x_d}, \underset{t+1}{0}, ..., \underset{T}{0}) \tag{3.8}$$

Each linear term can be perceived as a decision rule without a triggering condition which means that they are always triggered. A linear term influences the prediction of the ensemble by introducing a linear model that applies to target $t$ and uses a descriptive attribute $d$, as depicted in Equation 3.8.

The idea of using linear terms is to improve overall predictive performance of the rule ensemble. Each linear term is assigned a weight (exactly as with decision rules) that is subject to optimization. Contrary to the average vector, which is a constant, linear terms give different predictions based on the values of the descriptive attributes of the instance being predicted. Consequently, the final prediction of the rule ensemble does not only depend on constant predictions of the average vector and (triggered) decision rules but also on the descriptive values of the predicted instance.

Let us consider a fictional MTR problem with ten descriptive and three target variables and an example rule ensemble with the following properties:

- an average prediction vector $(C_1, C_2, C_3)$,
- two decision rules $R_1$ and $R_2$ with corresponding weights $w_1$ and $w_2$, and
- four linear terms: $x_{(1,5)}, x_{(2,1)}, x_{(3,9)}, x_{(3,3)}$ with corresponding weights $w_3, w_4, w_5$ and $w_6$.

This example rule ensemble is shown below. Note that the linear terms can be stacked together and combined with the average prediction vector. It can be seen that the ensemble predictions contain decision rules as well as linear models that influence the predictions based on the values of descriptive attributes.

$$\begin{aligned} Model =& (C_1, C_2, C_3) \\ &+ w_1 \cdot R_1 + w_2 \cdot R_2 \\ &+ w_3 \cdot (x_5, 0, 0) + w_4 \cdot (0, x_1, 0) + w_5 \cdot (0, 0, x_9) + w_6 \cdot (0, 0, x_3) \\ =& (C_1 + w_3 \cdot x_5, C_2 + w_4 \cdot x_1, C_3 + w_5 \cdot x_9 + w_6 \cdot x_3) \\ &+ w_1 \cdot R_1 + w_2 \cdot R_2 \end{aligned}$$

## 3.4 Summary

In this chapter, we introduced the predictive clustering framework for MTP. We described two types of machine learning models, namely decision trees and decision rules.

We described the predictive clustering trees (PCTs) – generalized decision trees for MTP. We use PCTs as individual models as well as base predictive models in tree-based ensembles. We described several ensemble methods that will later be used for building predictive models. In particular, we described three algorithms for building tree-based ensemble models: Bagging, Random forest and ensembles of extremely randomized trees. All three ensemble methods have been extended towards MTP.

The described rule-based ensemble models use predictive clustering rules (PCRs) as base predictive models. We describe the algorithm called *Fitted rule ensembles* for MTR, an extension of the RuleFit algorithm towards MTR. In contrast to the high-performing and uninterpretable tree-based ensemble models, rule-based ensemble models have lower predictive power but can be highly interpretable. The predictive performance of rule ensembles for MTR is comparable to that of a single PCT. In order to lift the performance of rule ensembles, one can optionally include linear terms into the ensemble model. These terms augment the predictions of the rule ensemble by using the input values rather than the knowledge obtained during the training of the ensemble model.

# Chapter 4

# Random Output Selections – ROS

In this chapter, we introduce an ensemble extension method called Random Output Selections (ROS) that can be applied to solving MTP tasks. The idea of ROS is to force the individual members of the ensemble into learning only from subsets of the target space. In particular, we extend the ensemble methods described in Section 3.2 to learn predictive clustering trees (PCTs) on randomly selected subsets of the target space. This approach has shown promise as it was able to outperform state-of-the-art MTP methods (Breskvar, Kocev, & Džeroski, 2017, 2018a).

Decision trees are obtained by using the top-down induction algorithm (TDIDT). The driving force behind the building process is a greedy heuristic function responsible for finding the most appropriate sub-partitions of input data according to some criteria. The sub-partitions are usually not obtained directly but as a consequence of applying a test, which is in fact represented in the form of a condition. All examples that meet the condition travel along a different path than the examples that do not meet that condition. In general, the input data can be partitioned in many different ways. One of the most commonly used approaches is to use a greedy heuristic function to evaluate all the candidate tests and select the one with the highest heuristic score as the best.

Greedy approaches, including the one used for inducing PCTs, exhibit a property that is best described as myopia. This simply means that the algorithm purposely ignores the fact that the candidate splits with the best heuristic scores are not necessarily optimal and could actually push the induction process onto a suboptimal path. When using the top-down approach to building decision trees, we must therefore accept the risk that the algorithm can select a suboptimal split. Due to the hierarchical nature of the induction process itself, it is obvious that if we select a suboptimal split in the early stages, we cannot recover from this mistake. This problem is present in all decision tree methods that use the top-down induction approach and becomes even more challenging when the heuristic score is calculated against several target attributes, as is the case with MTP tasks.

This chapter is structured as follows. First, we describe the two approaches to solving MTP tasks. Next, we describe how the individual ensemble members subset the output space. Then, we present the method for learning ROS ensembles with PCTs. Finally, we provide a computational complexity analysis of the proposed approach.

## 4.1 Approaches to Multi-Target Prediction

Before we describe the intricacies of ROS, we must first talk about the two commonly used approaches to solving MTP tasks: local and global. The majority of existing methods can be classified as either local or global but ROS operates on the spectrum between these two extremes.

Each target of the multi-target structure can be considered as a separate building block that one would like to predict. It is therefore obvious that the most straightforward way of solving a MTP task is to learn a separate predictive model for each target attribute of the multi-target structure. As it was already pointed out in Chapter 2.2, the main premise of SOP is the relatedness among the components of the target structure. Consequently, if we model each component separately, we do not take advantage of the potential relations among the components, which can lead to learning models with inferior predictive power. Nevertheless, this approach is known as local and is widely used.

On the other hand, the complete target structure can be predicted with only one predictive model, considering the target structure as a set of connected problems rather than many unrelated subproblems. This approach is known as global or "big bang", because the algorithms have the opportunity to see and exploit the information in all components of the target structure as well as the potential relations among them. This can lead to improved predictive performance, simplification of the learned model(s), higher computational efficiency and reduced overfitting as compared to local methods (Kocev et al., 2009; Kocev et al., 2013; Levatic, Kocev, & Džeroski, 2014).

These two polarized approaches both have their advantages as well as some drawbacks. Local approaches are simple to implement and they often outperform the global approaches, albeit not always with statistical significance. Another important advantage of the local approach to MTP is the abundance of available methods. When a MTP task is decomposed into individual single-target prediction subtasks, one can use many available single-target methods to solve them. However, the main problem with this approach is its computational complexity, which increases linearly with the number of target attributes.

Substantially less methods of global nature exist. The ones that do, however, are usually computationally more efficient as compared to using the local approach. The global methods produce one model that predicts all target attributes. This can greatly improve interpretability, which inherently depends on several key factors. First, the model type itself must be interpretable, such as decision trees or rules. Second, if the models are grouped into an ensemble, the interpretability is again lost. There are cases, where this is not true (e.g., with rule ensembles) but it generally holds. And lastly, if the models are too big, interpretation is difficult but still considerably less tedious in comparison with the interpretation of local models.

The main drawback of global methods is that they explicitly focus on exploiting the relations among the target attributes. When used appropriately, these relations can influence the model-learning process in a very beneficial way. In particular, the resulting models can have better predictive performance. However, the relations among the target attributes are often weak. In addition to this, they can even be fictional, i.e., they can appear to exist due to noisy data. Most global methods do not address this issue explicitly and force the "relatedness agenda" throughout the whole learning process. We therefore propose a new global ensemble approach that learns and combines *localized* models with the goal to increase the overall predictive performance of the ensemble by reducing the effects of weak and noisy relations among the target attributes.

## 4.2 Generating Random Subspaces

We have mentioned that the relations among the target attributes can potentially be weak or nonexistent. Such properties of the data can negatively influence the learning process in global methods because of their inherent assumption of relatedness among all target attributes. ROS aims to constrain this problem by learning a specially crafted ensemble of decision trees, in particular, PCTs.

Until now, we have not discussed a very important part of ROS: the localization of base models in the ensemble. We purposely use the term *localized* over *local* because of two reasons: (i) the term local indicates a different approach, which we explain below and (ii) because ROS ensembles are in fact global in nature. The difference between global and ROS ensembles is that individual base models in a ROS ensemble do not learn from all target attributes but a subset of them. Although individual base models could be instantiated to learn from only one target attribute, ROS ensembles were designed to use base models that were learned from two or more target attributes, making them localized towards the target attributes used for learning.

Figure 4.1 depicts three strategies that can be applied when solving a generic MTP task with four target variables: local (left), global (right) and ROS (middle). All three approaches are presented in the context of learning ensembles, where the number of base models in the ensemble is $n$. A global ensemble would contain $n$ base models, where each base model would be trained on all available target attributes. A ROS ensemble would also contain $n$ base models. In this case, however, the base models would have each been trained on a different subset of target attributes. Finally, the local approach is in fact an "ensemble of ensembles". Here, an ensemble of size $n$ is trained for each target attribute. Note that the number of models using the local approach linearly increases with the number of target attributes.

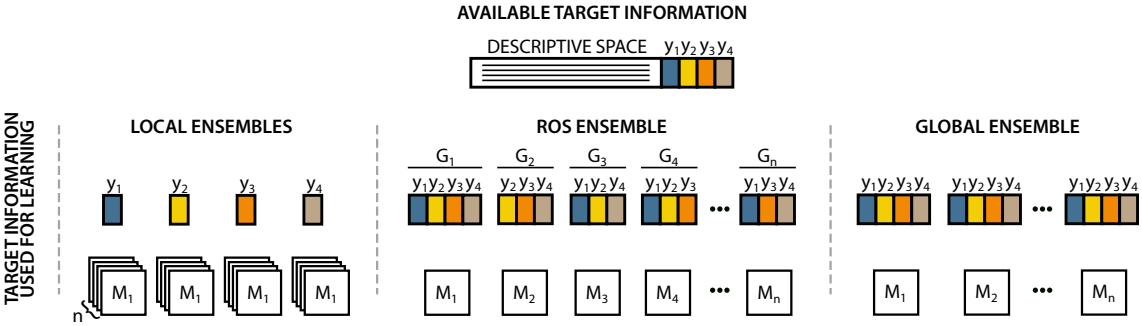



Figure 4.1: The local approach (far left) learns one ensemble model for each target attribute. The global approach (far right) does exactly the opposite: it learns one base predictive model for all target attributes (for this figure, the subset size is 75% of the number of targets). The approach in the middle is ROS. It can be seen that ROS ensemble contains base models that consider only a subset of all target attributes. The figure also illustrates that individual base models in the ROS ensemble use different subspaces of target attributes.

It is at this point that we address the issue of selecting target subspaces. The subsets of the target space must be selected in such a way that the resulting ensemble model will have better predictive power compared to the non-ROS ensemble. Let us assume that target attributes $y_1$ and $y_2$ are highly related. If we learn a global ensemble, we will be able to model $y_1$ and $y_2$ better than if we model them with a local approach. However, local approach will most likely outperform the global one on the attribute $y_3$, because the global approach is forced to also consider $y_1$ and $y_2$, thus deviating from the purpose of finding a good fit for $y_3$. This issue with the global approach actually worsens, if the target variables are in fact completely unrelated. By using a ROS ensemble, we aim to reduce or completely eliminate such noise otherwise overlooked by traditional global ensembles.

Subspaces of the target space can be crafted in (at least) three ways:

1. A computational approach could be used to determine the (non-)linear relations

**Algorithm 4.1:** Algorithm for generating ROS subspaces.

**Function:** GenerateSubspaces
**Input:** Number of subspaces to generate $b$, function $\theta(\mathsf{X}, v)$ that samples uniformly at random and without replacement a subset of the set $\mathsf{X}$, set of target attributes $\mathsf{A}_t$, percentage $v$ to sample
**Output:** A list $\mathsf{G}$ of length $b$, containing sampled target subspaces

```
G ← emptyList()
AddToList(G, A_t)          // The first model predicts all target attributes
for i ← 2 to b do
    AddToList(G, θ(A_t, v))
end
return G
```

among the target attributes. With that information, the target space subsets could be generated by always coupling (highly) related attributes together.

2. Semantics of the data could be taken into account by asking a domain expert to suggest the appropriate target subspaces.

3. The subspaces could be created in a randomized manner.

The proposed ROS ensemble approach generates the subspaces with random sampling from the pool of target attributes. Whereas regular PCTs simultaneously consider the whole target space in the heuristic used for tree construction, ROS considers a different random subset of it for each base model in the ensemble. Each base model is consequently learned by considering only those targets that are included in the randomly generated partition provided to it.

ROS creates the output space partitions (subspaces) in advance, i.e., the partitions are independent of the algorithm using them. ROS generates a different subspace for every ensemble constituent. Thus, the number of generated subspaces must be the same as the number of base models in the ensemble. We refer to this number with variable $b$. Algorithm 4.1 depicts the construction process of the target subspaces.

In the first step, we create an empty list that will eventually contain all the generated subspaces, i.e., $\mathsf{G} = [\mathsf{G}_1, \mathsf{G}_2, \ldots, \mathsf{G}_b]$. The first generated subspace is the set $\mathsf{A}_t$ and includes all target variables, i.e., the first PCT in the ensemble considers all target attributes. This is needed to ensure that all targets are being considered at least once. We generate the remaining $b-1$ subspaces with the $\theta$ function, which has a parameter $v$. An example of a $\theta$ function could return a random selection of 25% ($v = \frac{1}{4}$) of items in the set provided as input. If one defines $\theta(\mathsf{X}, v) = \mathsf{X}$, then all ensemble constituents will always consider all targets, which is what a regular ensemble of PCTs does. This function is a parameter of our overall ensemble learning algorithm.

Algorithm 4.2 describes the random sampling $\theta$ function used in our experiments. The function $\theta(\mathsf{X}, v)$ samples uniformly at random and without replacement $\lceil v \cdot |\mathsf{X}| \rceil$ items from the set $\mathsf{X}$, where $v$ represents the percentage of $\mathsf{X}$ we want to sample. This algorithm always samples a fixed number of attributes. We use this sampling function on the set of target attributes $\mathsf{A}_t$, consequently obtaining subsets of the target space.

**Algorithm 4.2:** A sampling function that returns a percentage of elements provided in the input set.

**Function:** $\theta$
**Input:** A set $\mathsf{X}$ to sample from, the desired percentage of items to sample $v$
**Output:** A subset of items $\mathsf{Q}$ ($\mathsf{Q} \subseteq \mathsf{X}$)
$\mathsf{Q} \leftarrow \emptyset$
**while** $|\mathsf{Q}| < \lceil v \cdot |\mathsf{X}| \rceil$ **do**
  $a \leftarrow \text{RandItemFromSet}(\mathsf{X} \setminus \mathsf{Q})$
  $\mathsf{Q} \leftarrow \mathsf{Q} \cup \{a\}$
**end**
**return** $\mathsf{Q}$

## 4.3 Predictive Clustering Trees with ROS

We have already described the processes of learning regular single PCTs and ensembles thereof in Sections 3.1 and 3.2, respectively. In order to use ROS, the algorithm for inducing a single PCT as well as ensemble algorithms require changes. In this section, we describe the required changes pertaining to the PCT induction algorithm. Changes are required in the PCT function, which in turn affects the BestTest function and impurity calculations therein. Algorithm 4.3 depicts the new PCT induction algorithm, adapted for using ROS. Additional input parameter $\mathsf{G}$ is introduced, which is passed to the PCT and BestTest functions.

**Algorithm 4.3:** The top-down induction of predictive clustering trees with ROS.

**Function:** PCT
**Input:** A dataset $\mathsf{S}$, a set of descriptive attributes $\mathsf{A}_d$, a set of target attributes $\mathsf{A}_t$, an attribute sampling function $\delta$, ROS subspace $\mathsf{G}$ ($\mathsf{G} \subseteq \mathsf{A}_t$)
**Output:** A predictive clustering tree
$\mathsf{A'}_d \leftarrow \delta(\mathsf{A}_d)$
/* Injection of ROS subspace instead of $\mathsf{A}_t$ */
$(t^*, h^*, \mathcal{P}^*) \leftarrow \text{BestTest}(\mathsf{S}, \mathsf{A'}_d, \mathsf{G})$
**if** $t^* = none$ **then**
  /* Predictions are still calculated for all target attributes */
  **return** $\text{Leaf}(\text{Prototype}(\mathsf{S}, \mathsf{A}_t))$
**else**
  **foreach** $\mathsf{S}_i \in \mathcal{P}^*$ **do**
    /* Recursively passing the ROS subspace $\mathsf{G}$ */
    $\text{tree}_i \leftarrow \text{PCT}(\mathsf{S}_i, \mathsf{A}_d, \mathsf{A}_t, \delta, \mathsf{G})$
  **end**
  **return** $\text{Node}(t^*, \bigcup_i \{\text{tree}_i\})$
**end**

The BestTest procedure enumerates all possible candidate tests and calculates their heuristic scores. The original implementation calculates the heuristic score $h^*$ against the provided target attributes. This part of the procedure remains the same. However, now, ROS forces the individual PCTs to learn from provided ROS subspace $\mathsf{G}$ rather than from the original set of all target attributes $\mathsf{A}_t$.

## 4.4 Learning ROS Ensembles

With all preliminaries laid out, we can now describe the overall process of learning ROS ensembles. This dissertation uses three different ensemble building methods (bagging, random forests and extremely randomized trees) that have been extended to MTP (Kocev et al., 2013) and use PCTs (dubbed Bag and RF) and extremely randomized PCTs (dubbed ET) as base models.

**Algorithm 4.4:** A generic wrapper algorithm for building ROS ensembles.

**Function:** ROSEnsemble
**Input:** A dataset $\mathsf{S}$, a set of descriptive attributes $\mathsf{A}_d$, a set of target attributes $\mathsf{A}_t$, function $\gamma(\mathsf{X})$ that perturbs the dataset $\mathsf{X}$, an attribute sampling function $\delta$, ROS percentage $v$ to sample from $\mathsf{A}_t$
**Output:** Forest of PCTs with ROS
$\mathsf{G} \leftarrow \mathsf{GenerateSubspaces}(b, \theta, \mathsf{A}_t, v)$
$\mathsf{F} \leftarrow \emptyset$
**foreach** $\mathsf{G}_i \in \mathsf{G}$ **do**
  $\mathsf{S}^* \leftarrow \gamma(\mathsf{S})$
  $tree_{\mathsf{G}_i} \leftarrow \mathsf{PCT}(\mathsf{S}^*, \mathsf{A}_d, \mathsf{A}_t, \delta, \mathsf{G}_i)$
  $\mathsf{F} \leftarrow \mathsf{F} \cup \{tree_{\mathsf{G}_i}\}$
**end**
**return** $\mathsf{F}$

Algorithm 4.4 is a generic wrapper demonstrating how the ROS ensembles are instantiated. We use the values in Table 4.1 to describe the values of initialization parameters used by the generic algorithm to instantiate the three mentioned ensemble methods with random output selections. Note that the PCT function is now called with an additional parameter $\mathsf{G}_i$, specifying the ROS sampled subspace.

Table 4.1: Parameters for building ROS ensembles.

| Ensemble method | $\delta(\mathsf{X})$ | $\gamma(\mathsf{X})$ | BestTest |
|---|---|---|---|
| Bagging of PCTs | $\mathsf{X}$ | $\mathsf{Bootstrap}(\mathsf{X})$ | Alg. 3.2 |
| Random forest of PCTs | For MTR: $\theta\left(\mathsf{X}, \frac{1}{\sqrt{\lvert\mathsf{X}\rvert}}\right)$ | $\mathsf{Bootstrap}(\mathsf{X})$ | Alg. 3.2 |
| | For MLC: $\theta\left(\mathsf{X}, \frac{\lfloor 0.1 \cdot \lvert\mathsf{X}\rvert \rfloor + 1}{\lvert\mathsf{X}\rvert}\right)$ | | |
| Random forest of extra-PCTs | $\mathsf{X}$ | $\mathsf{X}$ | Alg. 3.6 |

## 4.5 Making Predictions with ROS Ensembles

We have described the process of making predictions with tree ensembles in Section 3.2. In this section, we propose a new aggregation function. To that end, we introduce a naming convention to differentiate between the original aggregation function and the proposed one. We refer to the original aggregation function as *total averaging*. The name stems from the fact that all base predictive models contribute a prediction for all target attributes. Hence the word *total*. In contrast to total averaging, we propose *subspace averaging*. As the name suggests, only a subset of predictions is considered in the overall voting procedure.

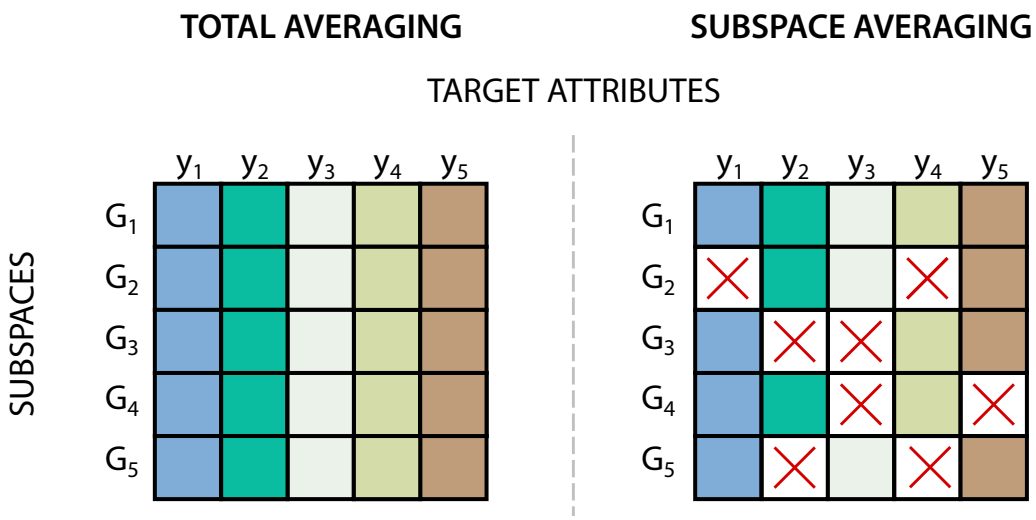



Figure 4.2: Visual differences between total and subspace averaging.

Figure 4.2 visually demonstrates the difference between total and subspace averaging. Both sides of the figure represent an ensemble with five base predictive models. The standard approach to prediction aggregation (total averaging) is shown on the left side. Rows correspond to five considered subspaces ($\mathsf{G}_1 \dots \mathsf{G}_5$), each marked with its own color, belonging to their corresponding base predictive models. Each base predictive model (subspace) considers five target attributes ($y_1 \dots y_5$). Consequently, each target attributes is predicted based on five votes: one from each base predictive model. The right side of the figure depicts a different behavior, used when voting with subspace averaging. There, not all base predictive models give predictions for all target attributes, e.g., model with subspace $\mathsf{G}_4$ does not give a prediction for target attributes $y_3$ and $y_5$. Note that the subspaces used with subspace averaging are the same as those used for learning the base predictive models, which we discuss in Section 4.2. It can also be seen that the first base predictive model ($\mathsf{G}_1$) gives a prediction for all target attributes. This behavior is by design, as shown by Algorithm 4.1. Next, we describe both prediction mechanisms in a more formal manner.

**Total averaging** combines the predictions of all base models for all target attributes. In case the task is MTR, the predictions are averaged. We calculate the average as the arithmetic mean: Final prediction of the ensemble for the $i^{th}$ target attribute ($\widehat{y}_i$) is computed as $\widehat{y}_i = \frac{1}{b}\sum_{j=1}^{b} y_i^j$, where $y_i^j$ represents the prediction of $j^{th}$ base model for the $i^{th}$ target attribute in an ensemble with $b$ base predictive models. In case the task is MLC, the predictions are combined using probability per-target distribution voting (Bauer & Kohavi, 1999).

**Subspace averaging** considers only the predictions made by the base predictive models for the targets used to learn them. In other words, the prediction for a given target is averaged over only those base models, for which that target was considered in the heuristic during learning. In an ensemble with $b$ base predictive models, a prediction for the $i^{th}$ target attribute is computed as shown by Equation 4.1.

$$\widehat{y}_i = \frac{\sum_{j=1}^{b} \mathbb{1}(t_i \in \mathsf{G}_j) \cdot y_i^j}{\sum_{j=1}^{b} \mathbb{1}(t_i \in \mathsf{G}_j)} \tag{4.1}$$

The indicator function $\mathbb{1}(\cdot)$ evaluates to 1, if the argument $\cdot$ is true and 0 otherwise. $\mathsf{G}_j$ is the subspace of target attributes, which was considered during learning of the $j^{th}$ ensemble base predictive model. The symbol $t_i$ denotes the $i^{th}$ target attribute. The denominator

is the number of ensemble constituents for which $t_i$ was considered during learning and is non-zero because every target attribute is considered by at least the first base predictive model (subspace $\mathsf{G}_1$).

## 4.6 Computational Complexity Analysis

From the work of Kocev et al. (2013) and the assumption that the decision tree is balanced and bushy (Witten & Frank, 2005), it follows that the computational complexity of learning a **single multi-target PCT** is $\mathcal{O}(dNlog^2N) + \mathcal{O}(dtNlogN) + \mathcal{O}(NlogN)$, where $N$ is the number of instances, $d$ is the number of predictive attributes and $t$ is the number of target attributes in the dataset. Similarly, from the work of Kocev and Ceci (2015) and with the same assumption, it follows that the computational complexity of learning a **single multi-target extra-PCT** is $\mathcal{O}(ktNlogN) + \mathcal{O}(NlogN)$, where $k$ is the number of randomly sampled predictive attributes at each split. In general, learning an **ensemble** of $b$ base models has the complexity of learning all of its constituents. In our case, that amounts to $b(\mathcal{O}(dNlog^2N) + \mathcal{O}(dtNlogN) + \mathcal{O}(NlogN))$ for bagging and random forests of PCTs and $b(\mathcal{O}(ktNlogN) + \mathcal{O}(NlogN))$ for random forests of extra-PCTs.

The computational complexity also depends on the use of **bootstrapping** and the amount of predictive and/or target attributes considered for each base model. Computational cost of bootstrapping is $\mathcal{O}(N)$ and the number of instances considered in that case equals $N' = 0.632 \cdot N$ (Breiman, 1996). Bootstrapping is not used for learning extremely randomized PCTs.

Taking into account the fact that random forests also **sample the input space** (through the sampling function $\delta$), the number of predictive variables actually considered by the base models is $d'$ ($d' < d$). The sampling of predictive variables happens at every split node, so the complexity of data subsampling is $\mathcal{O}(d'logN')$.

ROS proposes additional **sampling of the target space**. The function $\theta(\mathsf{X}, v)$ (see Algorithm 4.2) is used to sample from the target space. The sampled subsets are always of equal cardinality, which is controlled by the parameter $v \in (0.0, 1.0)$. However, the first subset always includes all target attributes. We define the variable $t'$ as the average target subspace cardinality considered by the base models as: $t' = \frac{1}{b}\big((b-1)\cdot\lceil v\cdot t\rceil + t\big)$. The complexity of the sampling function $\theta(\mathsf{X}, v)$ is low. All operations of the sampling algorithm are in the order of a constant ($\mathcal{O}(1)$). The while loop is executed $\lceil v \cdot |\mathsf{X}|\rceil$-times. Each randomly sampled attribute $a$ has equal probability of being included in the resulting set $\mathsf{Q}$. Thus, the complexity of the sampling algorithm, which samples $b-1$ times, is linear and proportionate to $(b-1)\cdot\mathcal{O}(v\cdot t)$. Considering all of the above, the complexity of the ROS ensembles is:

$$\begin{aligned}\text{Bag-ROS } \textit{or } \text{RF-ROS} &= b\cdot\big[\mathcal{O}(d'N'log^2N') + \mathcal{O}(d't'N'logN') \\ &\quad + \mathcal{O}(N'logN') + \mathcal{O}(d'logN') + \mathcal{O}(N)\big] \\ &\quad + (b-1)\cdot\mathcal{O}(v\cdot t) \\ \text{ET-ROS} &= b\cdot\big[\mathcal{O}(d't'NlogN) + \mathcal{O}(NlogN) \\ &\quad + \mathcal{O}(d'logN')\big] + (b-1)\cdot\mathcal{O}(v\cdot t).\end{aligned} \tag{4.2}$$

The ratio between the full output space size and the one considered by ROS is constant and is proportionate to $\frac{t'}{t} = \frac{(b-1)\cdot v\cdot t + t}{b\cdot t} = \lim_{b\to\infty}\frac{(b-1)\cdot v+1}{b} = v$. The overall complexity of ROS is consequently reduced in the parts that correspond to the selection of the best split. We expect a linear decrease in complexity in those terms. Otherwise, the overall complexity is still as described by Kocev and Ceci (2015), Kocev et al. (2013).

Ensembles usually contain many base models which results in longer times to make a prediction. Therefore, we also address the complexity of **making predictions**. Under the previously mentioned assumption about decision trees being balanced and bushy, the average depth of a decision tree is actually the average length of the path that has to be traversed by an instance in order to get to the prediction. The complexity of making a prediction with a single-target decision tree is therefore $\mathcal{O}(log(N))$. When we use the global approach to MTP, all target variables are predicted simultaneously with the same complexity as that of making a prediction with a single-target tree. When we switch to the ensemble setting, the complexity increases linearly with the number of base models in the ensemble to $b \cdot \mathcal{O}(log(N))$. If we approach the MTP task locally, each target is predicted with its own ensemble and that additionally increases the complexity in proportion to the number of target variables to $bT \cdot \mathcal{O}(log(N))$.

## 4.7 Summary

This chapter introduces a novel global ensemble extension approach for addressing MTP tasks. The proposed method randomly selects subsets of targets for learning individual base models. This can yield better predictive performance and shorter learning times. The proposed method is called *Random Output Selections* – ROS.

A novel ensemble prediction aggregation function for joining the predictions of base models in ROS ensembles was also introduced. By default, each tree in the ensemble learns from and consequently gives predictions for all targets. With the newly-proposed aggregation function, only targets that were considered during the learning of an individual tree contribute to the final predictions.

We conclude the chapter with a theoretical computational complexity analysis of the proposed ensemble extension method.

# Chapter 5

# Evaluation of Tree Ensembles with Random Output Selections for Multi-Target Regression

In this chapter, we address the task of multi-target regression (MTR), described in Section 2.2.1. We solve the task of MTR by learning ensembles of predictive clustering trees (PCTs). In particular, we learn three different types of ensembles of PCTs that use ROS: the proposed ensemble extension method, described in Chapter 4.

The main focus of this study is to determine whether the proposed method can improve the predictive performance and shorten the learning times of the considered ensemble methods. An extensive empirical evaluation over a variety of benchmark datasets is performed in order to determine the effects of using ROS on predictive performance. In addition to this, we also perform an analysis with respect to time and space complexity. The results show that the proposed ensemble extension can yield better predictive performance, reduce learning time or both, without a considerable change in model size. The newly proposed aggregation function gives best results when used with ensembles of extremely randomized PCTs.

We perform an extensive empirical evaluation of the three different ensemble methods on 17 benchmark datasets, which provide performance assessment for the original and extended ensemble methods, as well as individual multi-target regression trees and ensembles of single-target regression trees (all over a range of ensemble sizes). The study also includes parameter setting recommendations for the proposed method. Moreover, we compare the performance to other competing methods that also transform the output space.

This chapter is composed of two sections. The first section provides details regarding the experimental design. We start by posing experimental questions. We also explain and describe the evaluation measures used in this chapter. Then, we describe the benchmark datasets used for the evaluation of the proposed method. Finally, we describe the experimental setup, specifically designed to answer the posed questions. The second section of this chapter contains results of the experiments and a comparison with the other output space transformation methods.

## 5.1 Experimental Design

To evaluate the performance of the ROS ensembles for MTR, we performed extensive experiments on benchmark datasets. This section presents: (i) the experimental questions addressed, (ii) the evaluation measures used, (iii) the benchmark datasets and (iv) the

experimental setup (including the parameter instantiations for the methods used in the experiments).

### 5.1.1 Experimental questions

In our experiments, we construct PCT ensembles for MTR by using the described ensemble extension method ROS. In order to better understand the effects of ROS, we investigate the resulting ensemble models across three dimensions.

First, we are interested in the **convergence** of their predictive performance as we increase the number of PCTs in the ensemble. We want to establish the number of base models needed in an ensemble to reach the point of performance saturation. We consider an ensemble saturated, when adding additional base models to it would not bring statistically significant improvement in terms of predictive power.

Next, we are interested, whether the proposed extension can improve the **predictive performance** over the performance of the original ensembles. Learning on subsets of targets could exploit additional structural relations that may be overlooked by the original ensemble approaches.

Finally, as we have theoretically derived in Section 4.6, we expect that the dimensionality reduction of the output space will yield improvements in terms of **computational efficiency**. Specifically, we are interested in the running times of the ROS ensemble approaches and the sizes of the resulting models.

The specific experimental questions we pose relate to the above three dimensions we are interested in. The experiments and their evaluation have been designed with the following research questions in mind:

1. How many base models do we need in ROS ensembles in order to reach the point of performance saturation?
2. What is the best value for the portion of target space to be used within such ensembles? Is this portion equal for all evaluated ensemble methods?
3. Does it make sense to change the default aggregation function of the ensemble that uses the prediction for all targets? Can this improve predictive performance?
4. Considering predictive performance, how do ROS ensemble methods compare to the original ensemble methods?
5. Is ROS helpful in terms of time efficiency?
6. Do ROS models use less memory than the models trained with the original ensemble methods?
7. How the ROS models compare to other output transformation methods?

### 5.1.2 Evaluation measures

In order to understand the effects that ROS has on the learning process, we first need to evaluate the models induced by the ROS ensemble approaches. In machine learning, empirical evaluation is most commonly used to achieve this goal that assesses the performance of a given model in terms of evaluation measures. Below we describe the measures that we use for assessing predictive power, time and space complexity.

The **predictive performance** of a MTR model is assessed by using the average relative root mean squared error (aRRMSE), which averages the relative root mean squared errors (RRMSE) for the individual target variables. RRMSE is a relative measure calculated

against the baseline model that predicts the arithmetic mean of all values of a given target in the learning set. Specifically, the value $\overline{y}_i$ in Equation 5.1 is the prediction of the baseline model for the $i^{th}$ target variable, while the value $\hat{y}_i^{(e)}$ represents the predicted value for the $i^{th}$ target variable of the example $e$.

$$\text{aRRMSE} = \frac{1}{t}\sum_{i=1}^{t}\text{RRMSE}_\text{i} = \frac{1}{t}\sum_{i=1}^{t}\sqrt{\frac{\sum_{e=1}^{\text{N}_{test}}(y_i^{(e)} - \hat{y}_i^{(e)})^2}{\sum_{e=1}^{\text{N}_{test}}(y_i^{(e)} - \overline{y}_i)^2}} \tag{5.1}$$

We also monitor how much our models **overfit** the training data by calculating their relative decrease of performance on the testing data with respect to that on the training data. Smaller values mean less overfitting, with zero being the ideal score. We calculate the overfitting score with Equation 5.2.

$$\text{OS} = \frac{\text{aRRMSE}_{test} - \text{aRRMSE}_{train}}{\text{aRRMSE}_{train}} \tag{5.2}$$

The **efficiency** is measured in terms of execution times and sizes of the induced models. Time efficiency is measured with the CPU time needed to induce (train) the model (i.e., learning time). For ROS ensembles, this includes the target space decomposition. We also measure the average time needed to make a prediction (i.e., prediction time). Space efficiency is measured with the total number of nodes in the tree model (intermediate and leaf nodes): the smaller the better. Induction times and model sizes are summed over all ensemble constituents.

### 5.1.3 Data description

To evaluate the proposed method, we use 17 benchmark datasets that contain multiple continuous target attributes. Datasets come from various domains. Table 5.1 shows the main characteristics of the considered datasets. In order to have as general evaluation as possible, we use datasets of different sizes in terms of number of instances, number of predictive and number of target attributes.

**ADNI.** The dataset comes from Alzheimer's Disease Neuroimaging Initiative (ADNI) database. ADNI is an international observational study of healthy, cognitively normal elders, people with mild cognitive impairment and people with Alzheimer's disease. It collects a wide range of clinical and biological data for each patient at multiple time points. We used the ADNIMERGE table, which is a joined dataset from multiple ADNI data collection domains (Gamberger, Ženko, Mitelpunkt, Shachar, & Lavrač, 2016). Each patient is described with biological and clinical attributes. We use the biological attributes to predict 14 clinical target attributes related to everyday cognition of patients and their study partners.

**ATP 1D and ATP 7D.** These datasets contain Airline Ticket Prices (Spyromitros-Xioufis et al., 2016). The goal is to predict prices of airline tickets for the following day (ATP 1D) or to predict the minimal price in the next seven days (ATP 7D). We predict prices of six different flight preferences: any airline, any non-stop airline, and four different airline companies.

**Forestry Kras.** The data in this dataset was derived from multi-spectral multi-temporal Landsat satellite imagery and 3D LIDAR recordings of a part of the Kras region in Slovenia (Džeroski, Kobler, Gjorgjioski, & Panov, 2006). Each example corresponds to a rectangular region of interest (spatial unit). Attributes for a specific spatial unit were obtained

Table 5.1: Properties of the considered MTR datasets with multiple continuous targets: number of examples (N), number of descriptive attributes ($|A_d|$), and number of target attributes ($|A_t|$).

| no. | Dataset | Domain | N | $\lvert A_d \rvert$ | $\lvert A_t \rvert$ |
|---|---|---|---|---|---|
| 1 | **ADNI** | Medicine | 659 | 232 | 14 |
| 2 | **ATP 1D** | Economy | 337 | 441 | 6 |
| 3 | **ATP 7D** | Economy | 296 | 441 | 6 |
| 4 | **Forestry Kras** | Ecology | 60607 | 160 | 11 |
| 5 | **OES 10** | Economy | 403 | 298 | 16 |
| 6 | **OES 97** | Economy | 334 | 263 | 16 |
| 7 | **PPMI** | Medicine | 713 | 148 | 35 |
| 8 | **Prespa lake top 10** | Ecology | 218 | 16 | 10 |
| 9 | **RF 1** | Ecology | 9125 | 64 | 8 |
| 10 | **RF 2** | Ecology | 9125 | 576 | 8 |
| 11 | **Sales** | Economy | 639 | 401 | 12 |
| 12 | **SCM 1D** | Economy | 9893 | 280 | 16 |
| 13 | **SCM 20D** | Economy | 8966 | 61 | 16 |
| 14 | **Soil resilience** | Ecology | 26 | 7 | 8 |
| 15 | **Vegetation clustering** | Ecology | 29679 | 65 | 11 |
| 16 | **Vegetation condition** | Ecology | 16967 | 39 | 7 |
| 17 | **Water quality** | Ecology | 106 | 16 | 14 |

from the Landsat and LIDAR recordings of that unit. The goal is to predict eleven targets, corresponding to properties of the vegetation in the observed spatial unit.

**OES 10 and OES 97.** The Occupational Employment Survey datasets were obtained from the US Bureau of Labor Statistics (Spyromitros-Xioufis et al., 2016). The datasets contain data for the years 1997 (OES 97) and 2010 (OES 10). Each row represents an estimated number of full-time equivalent employees across many employment types for a specific metropolitan area. The input variables are a randomly sequenced subset of employment types (e.g. doctor, dentist, car repair technician, etc.) observed in at least 50% of the cities. The 16 targets for both years are randomly selected from the entire set of categories above the 50% threshold.

**PPMI.** This dataset was collected within the Parkinson's Progression Markers Initiative (PPMI). It contains the regions of interest (ROIs) of the patients' fMRIs, DaT (dopamine transporter) scans and the motor assessment scores (MDS-UPDRS). The time interval between the date of scoring and the date of fMRI scans is smaller than 6 months. The task is to predict all of the scores for the motor impairment assessments from the extracted ROIs and the DaT scans features (Marek et al., 2011).

**Prespa lake top 10.** This dataset contains measurements of the effects of the environmental conditions of Lake Prespa on diatom communities (Kocev, Naumoski, Mitreski, Krstić, & Džeroski, 2010). The data represents water samples taken from the lake during the EU project TRABOREMA in the period of one and a half year. The descriptive attributes of the samples include temperature, dissolved oxygen, conductivity, alkalinity, nitrogen compounds, Secchi depth, sulphur oxide ions, sodium, potassium, magnesium,

copper, manganese and zinc. The task is to predict the relative abundances of the top ten most abundant diatoms.

**RF 1 and RF 2.** These datasets concern the prediction of river network flows for 48h in the future (Spyromitros-Xioufis et al., 2016). They were obtained from the US National Weather Service and contain hourly sampled data of river flows at 8 sites in the Mississippi River network in the United States. Each row includes the most recent observation for each of the 8 sites as well as time-lagged observations. In RF1, each site contributes 8 attribute variables to facilitate prediction. The RF2 dataset extends the RF1 data by adding precipitation forecast information for each of the 8 sites. The two datasets contain over 1 year of hourly observations collected from September 2011 to September 2012.

**Sales.** This is a pre-processed version of the dataset used in Kaggle's "Online Product Sales" competition (Kaggle, 2008) that concerns the prediction of the online sales of consumer products. Each row in the dataset corresponds to a different product that is described by various product features as well as features of an advertising campaign. The target variables correspond to the monthly sales in the first 12 months after the product launches (one target for each month).

**SCM 1D and SCM 20D.** The Supply Chain Management datasets are derived from the Trading Agent Competition in Supply Chain Management (TAC SCM) tournament from 2010 (Spyromitros-Xioufis et al., 2016). Each row corresponds to an observation day in the tournament. The input variables in this domain are observed prices for a specific tournament day. In addition, 4 time-delayed observations are included for each observed product and component to facilitate some anticipation of trends going forward. The datasets contain 16 regression targets, where each target corresponds to the next day mean price (SCM 1D) or mean price for 20 days in the future (SCM 20D) for each product in the simulation.

**Soil resilience.** This dataset (Debeljak et al., 2009) deals with the biological and physical resilience of soil. It consists of attributes describing physical properties of the soil such as texture (sand, silt, clay), chemical properties (pH, C, N, soil organic matter), FAO soil classification and the goal is to predict the eight indicators of physical resistance and resilience soil properties such as resistance and recovery from compression, recovery from overburden stress, as well as heat and copper.

**Vegetation clustering.** This dataset concerns the prediction of different plant species in Victoria State, Australia (Gjorgjioski, Džeroski, & White, 2008). It comes from the Arthur Rylah Institute for Environmental Research, Department of Sustainability and Environment (DSE) in Victoria, Australia and is provided by the Arthur Rylah Institute for Environmental Research, Department of Sustainability and Environment (DSE).

**Vegetation condition.** This dataset concerns the prediction of the vegetation condition (Kocev et al., 2009). It comes from the Arthur Rylah Institute for Environmental Research, Department of Sustainability and Environment (DSE) in Victoria, Australia and is provided by the Arthur Rylah Institute for Environmental Research, Department of Sustainability and Environment (DSE).

**Water quality.** This dataset contains information about water quality of Slovenian rivers. The goal is to predict abundances of plants and animals based on the chemical properties of the water (Džeroski et al., 2000). The data covers the period from 1990 to 1995 and is provided by the Hydrometeorological Institute of Slovenia.

### 5.1.4 Experimental setup

We designed the experimental setup according to the experimental questions posed in Section 5.1.1. First, we describe all parameter settings of the ROS ensemble methods. We then outline the procedures for statistical analysis of the results.

We consider **three types of ensembles**: bagging and random forests of PCTs and extremely randomized PCTs. In order for Algorithm 4.4 to simulate these three ensemble methods, we set its parameters to the values given in Table 4.1. Following the recommendations from (Bauer & Kohavi, 1999), the trees in the ensembles are unpruned. Our experimental study considers different **ensemble sizes**, i.e., different numbers of base models (PCTs) in the ensemble, in order to investigate the saturation of ensembles and to select the saturation point.

First, we construct **ensembles without** ROS (Bag, RF, ET) that use the full output space for learning the base predictive models. This means that the list $\mathsf{G}$ contains $b$ sets, where each set contains all the attributes from the set $\mathsf{A}_t$ (target attributes), i.e., $\mathsf{G} = \{\mathsf{A}_t, \mathsf{A}_t, \ldots, \mathsf{A}_t, \}$, where $|\mathsf{G}| = b$.

The second part of our experiments is concerned with the proposed extension. We start with the parametrization of the GenerateSubspaces function (Algorithm 4.1), which takes as input $b$, $\mathsf{A}_t$, the sampling function $\theta(\mathsf{X}, v)$ (see Algorithm 4.2) and sampling percentage $v$. We consider four values for $v$ in the allowed range $(0.0, 1.0)$, namely $\frac{1}{\sqrt{|\mathsf{A}_t|}}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$. Additionally, we use two ensemble predictions aggregation functions: total averaging and subspace averaging. Table 5.2 summarizes the parameter values considered in our experiments.

The third part of our study focuses on the comparison of our ROS ensemble methods to baseline methods. To that end, we also train multi-target PCTs and ensembles of single-target PCTs on each of the 17 benchmark datasets. F-test pruning is applied to single multi-target PCTs. The F value is selected using internal 3-fold cross-validation. We build one ensemble for each target variable. Ensembles contain 100 base models and are built by using the same parameters as the original ensembles.

Table 5.2: Parameter values used to build ensembles with ROS.

| **Location of use** | **Parameter** | **Used values** |
|---|---|---|
| ROSEnsemble$(\mathsf{S}, \mathsf{A}_d, \mathsf{A}_t, \gamma, \delta, v)$ | $\gamma, \delta$ | see Table 4.1 |
| GenerateSubspaces$(b, \theta, \mathsf{A}_t, v)$ | $b$ | $10, 25, 50, 75, 100, 150, 250$ |
| $\theta(\mathsf{X}, v)$ | $v$ | $\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{1}{\sqrt{\lvert\mathsf{A}_t\rvert}}$ |
| making predictions | averaging function | total, subspace |

We **estimate the predictive performance** of the considered methods by using 10-fold cross-validation. All methods use the same folds. For **statistical evaluation** of the obtained results, we follow the recommendations from (J. Demšar, 2006). The Friedman test (M. Friedman, 1940), with the correction by (Iman & Davenport, 1980), is used to determine statistical significance. In order to detect statistically significant differences, we

calculate the critical distances (CD) by applying the (Nemenyi, 1963) or (Dunn, 1961) post-hoc statistical tests. Both post-hoc tests compute critical distance between the ranks of considered algorithms. The difference is that Nemenyi post-hoc test compares the relative performance of all considered methods (all vs. all), whereas Bonferroni-Dunn post-hoc test compares the performance of a single method to other methods (one vs. all). The results of these tests are presented with average rank diagrams (J. Demšar, 2006), where methods connected with a line have results that are not statistically significantly different. All statistical tests were conducted at the significance level $\alpha = 0.05$. Statistical tests have been calculated for two variants of the results: per dataset (using aRRMSE value for each dataset) and per target (using the RRMSE values for all targets of all datasets). We used the Bonferroni-Dunn (CD is shown as a dotted blue line) post-hoc test to present results in Section 5.2.4 and Nemenyi post-hoc test otherwise (CD is shown as a solid red line).

The experiments were executed on a heterogeneous computing infrastructure, i.e., the SLING grid, which can affect time-sensitive evaluations. To avoid having incomparable measurements of running times, we run time-sensitive experiments separately by using a single computational node.

## 5.2 Results and Discussion

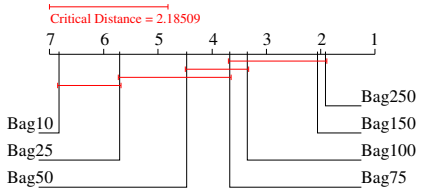
Here we present the results of our comprehensive experimental study. Considering a large number of datasets (17) and several ensemble methods, we present the results in terms of predictive performance (aRRMSE, overfitting score – OS), time complexity (learning and prediction time) and space complexity (model size). In the presentation of time complexity results, we focus on two datasets (*Forestry Kras* and *OES 10*) that have relatively large output spaces (11 and 16 targets, respectively). The selected datasets also differ in the number of examples: *Forestry Kras* has many whereas *OES 10* has few. For reference, all other results are available in Appendix A.

The presentation and discussion of the results follows the experimental questions from Section 5.1.1. First, we examine the convergence of original and ROS ensembles. Next, we focus on selecting the output space size. We experiment with four different output space sizes (see Table 5.2) that have consequently been used. This parameter is crucial because it introduces additional point of randomization in all three considered ensemble methods. In that sense, ROS can also be seen as a localization process: the constructed base models are tailored to a specific output subspace. We recommend values for this parameter for each ensemble learning method. Furthermore, we show the effects of changing the aggregation functions in our ensembles. Finally, we use the recommended parameters for ROS and provide an overall evaluation, by comparing the extended ensembles to the original ones. We compare the ROS methods to the baseline methods in terms of predictive power, running times and model sizes.

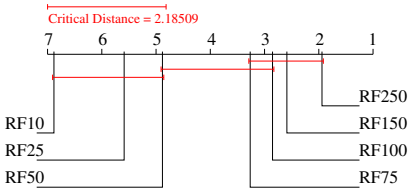
### 5.2.1 Ensemble convergence

The saturation points of the original ensembles (Bag, RF and ET) are located between 50 and 75 base models (Figure 5.1). These findings are in line with the work of (Kocev et al., 2013), where Bag and RF saturate with 50 base models and (Kocev & Ceci, 2015), RF and ET saturate with 100 and 75 base models, respectively. We attribute this difference to two factors: (i) we use a different and larger number of datasets, and (ii) the number of target attributes per dataset in our study is considerably larger. All in all, we consider the original ensembles with 75 base models saturated.

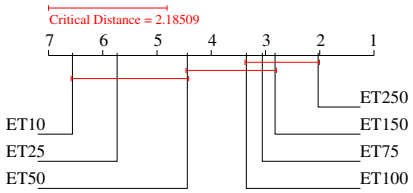
We next investigate the saturation of the ROS ensembles (denoted with -ROS postfix).




(a) Saturation of Bag




(b) Saturation of RF




(c) Saturation of ET

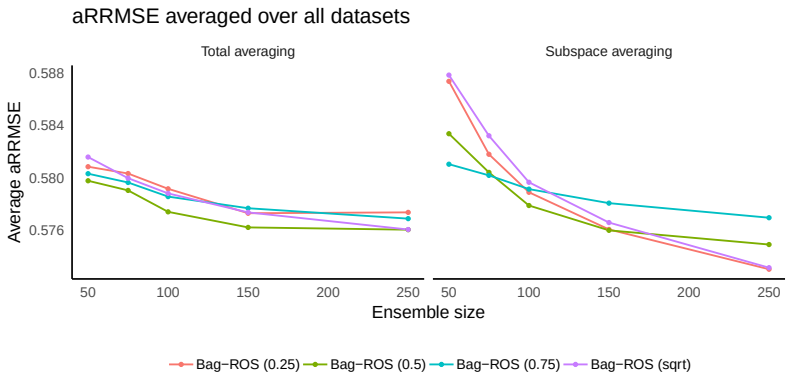
Figure 5.1: Saturation of original ensemble methods. The average rank diagrams compare the performance (aRRMSE) of ensembles with different size. Lower ranks are better. The saturation point is the lowest number of trees in the ensemble for which the performance is not significantly different than the best: This is 75 trees for Bag and RF and 50 trees for ET.

A subset of the results for ensembles with 50, 100, 150 and 250 models is reported in Figure 5.2 and illustrates the saturation of ROS ensembles for all three considered ensemble methods. Lines on the plots represent different output space sizes. Values in brackets indicate the value for the $v$ parameter. Left and right sides of the plots depict voting with total averaging and subspace averaging respectively. The $y$ axis shows aRRMSE values averaged over all considered datasets. The results show that Bag-ROS and ET-ROS ensembles saturate between 50 and 100 base models while RF-ROS ensembles saturate a bit later, between 75 and 100 base models. Figure 5.2 suggests that ROS ensembles saturate at a large number of base models when subspace averaging is used to aggregate the predictions of the base models. Performance in terms of aRRMSE and overfitting scores of all discussed ensembles with 100 trees (multi-target and single-target variants) and single multi-target regression trees is presented in Appendix A.3.

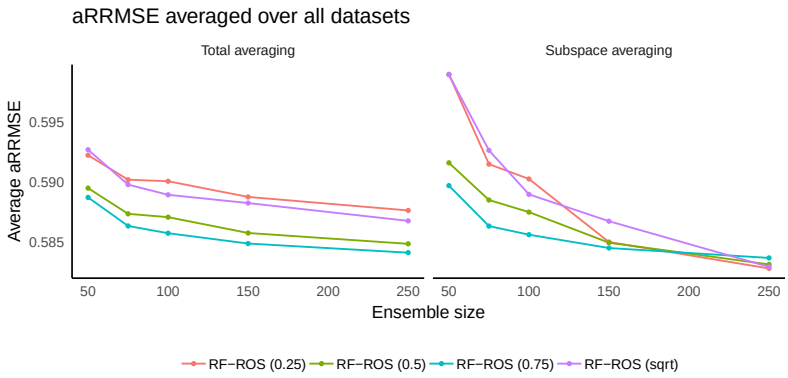
### 5.2.2 ROS parameter selection

This section describes the selection of the best performing output subspace size and aggregation function. The considered ensemble methods introduce different randomizations in their learning processes, so we cannot assume that ROS has the same influence on all three types of ensemble methods. Figure 5.2 also suggests that the choice of the aggregation function has a direct effect on performance. We therefore analyze the effect of output subspace size and aggregation function for each ensemble type separately.

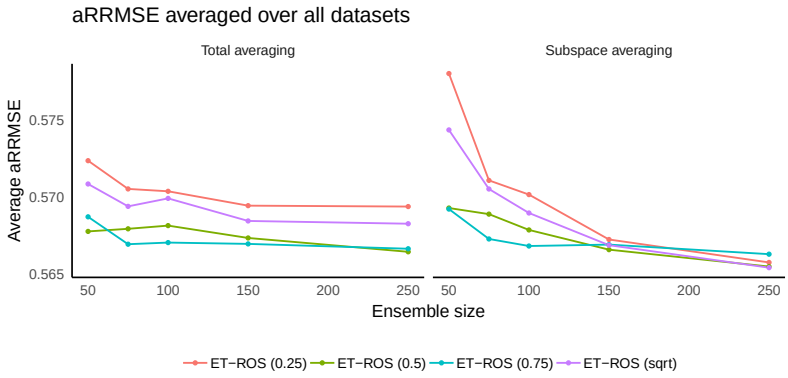
We selected candidate values for the ROS parameters based on the curves given in Figure 5.2. With both aggregation functions, candidate parameter values for subspace sizes are $v = \frac{1}{2}$ for Bag-ROS and $v = \frac{3}{4}$ for RF-ROS and ET-ROS. We selected these values because they exhibit the lowest aRRMSE averaged over all datasets used in this study. The averaged saturation curves in Figure 5.2 sometimes intertwine and make it




(a) Bagging of PCTs




(b) Random forests of PCTs




(c) Ensemble of extremely randomized PCTs

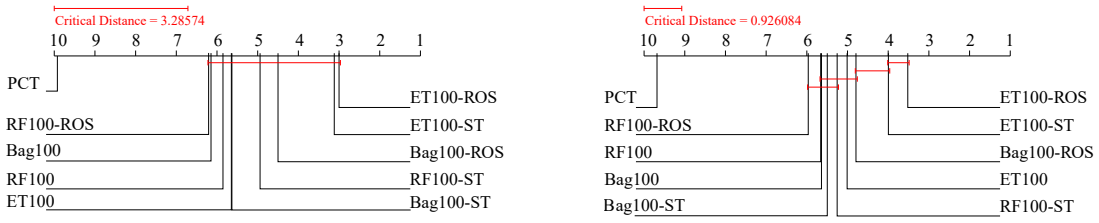
Figure 5.2: Saturation of all three ensemble methods extended with ROS. The different (color) lines on the plots represent different output space sizes. The values in the brackets after an ensemble method name indicate the value for the $v$ parameter. The left and right panels of the plots show results for voting with total averaging and subspace averaging, respectively. The $y$ axis shows aRRMSE values averaged over all 17 considered datasets.

difficult to make this decision. In those cases, we selected the parameter values based on the averaged performance of ensembles with 100 trees. Next, we performed a simple analysis by comparing the wins of the two considered aggregation functions using the

candidate output space size. For Bag-ROS, it turned out that total averaging had most wins, whereas subspace averaging was dominant for RF-ROS and ET-ROS. Our final parameter recommendation is therefore to use total averaging with $v = \frac{1}{2}$ for Bag-ROS and subspace averaging with $v = \frac{3}{4}$ for RF-ROS and ET-ROS.

### 5.2.3 Predictive performance and computational efficiency

Here, we compare the original ensemble methods (Bag, RF, ET) to the ones that use the ROS extension. In addition, ROS is also compared to multi-target PCTs and ensembles of single-target PCTs. We show the relative performance of the different methods by using the average rank diagrams shown in Figure 5.3.




(a) Per dataset (aRRMSE)

(b) Per target (RRMSE)

Figure 5.3: Overall average rank diagrams for predictive performance. Lower ranks are better.

Figure 5.3 depicts two average rank diagrams: one per dataset and one per target. The per dataset diagram is based on aRRMSE value, one per dataset. Both analyses show that ensembles statistically significantly outperform individual multi-target PCTs, i.e., multi-target PCTs perform significantly worse than the ensemble methods. The *per dataset analysis* shows no statistically significant difference in terms of predictive performance among the other methods. We can however note that Bag-ROS and ET-ROS outperform their original counterparts and ensembles of single-target PCTs. RF-ROS performs on par with the original bagging and random forest ensembles, but worse than the other ROS ensembles (Bag-ROS and ET-ROS). The best performing of all methods is ET-ROS.

The *per target analysis* detects two statistically significant differences in performance. First, with the exception of ET-ST, ET-ROS outperforms all other methods with statistical significance. Second, Bag-ROS outperforms RF-ROS, which performs worst of all ensemble methods. All original ensembles (Bag, RF and ET) show no statistically significant differences in performance. All in all, looking at the big picture, ROS ensembles generally perform better than their original counterparts, with the exception of random forests.

In Table 5.3, we show the predictive performance (aRRMSE) for two highlighted datasets: *Forestry Kras* and *OES 10*. The table contains results for the baseline ensembles (Bag, RF and ET) as well as the extended ROS ensembles, individual multi-target PCTs and ensembles of single-target PCTs. All ensembles contain 100 base models. The ensembles of single-target PCTs contain 100 base models per target.

For the *Forestry Kras* dataset, the proposed ROS methods do not have a notable effect on the predictive performance (aRMMSE) of the three ensemble methods. Similar findings are observed when calculating the overfitting score (OS): the ROS ensembles overfit the training data to the same extent as their original counterparts. Next, multi-target PCTs and ensembles of single-target PCTs have the worst predictive performance. The difference in predictive performance between ensembles of single-target PCTs and other ensembles

Table 5.3: Performance of ensembles and single trees on two datasets (*Forestry Kras* and *OES 10*) measured in terms of aRRMSE, overfitting score, average learning times, average per-instance prediction time and model complexity (total number of nodes).

| Dataset | Method | aRRMSE | OS | $\overline{LT}$ [s] | $\overline{PT}$ [$\mu s$] | Complexity |
|---|---|---|---|---|---|---|
| *Forestry Kras* | Bag | 0.55 | 0.476 | 394 | 272 | $3.22 \cdot 10^6$ |
| | Bag-ROS | 0.548 | 0.471 | 267 | 274 | $3.20 \cdot 10^6$ |
| | Bag-ST | 0.551 | 0.494 | 34500 | 3960 | $34.61 \cdot 10^6$ |
| | RF | 0.545 | 0.44 | 34.15 | 259 | $3.16 \cdot 10^6$ |
| | RF-ROS | 0.546 | 0.44 | 36.87 | 273 | $3.15 \cdot 10^6$ |
| | RF-ST | 0.546 | 0.457 | 2250 | 2300 | $17.13 \cdot 10^6$ |
| | ET | 0.557 | 0.575 | 450 | 264 | $3.67 \cdot 10^6$ |
| | ET-ROS | 0.557 | 0.579 | 281 | 274 | $3.67 \cdot 10^6$ |
| | ET-ST | 0.56 | 0.611 | 73780 | 3450 | $39.96 \cdot 10^6$ |
| | multi-target PCT | 0.61 | 0.169 | 76.68 | 2.39 | $2.59 \cdot 10^3$ |
| *OES 10* | Bag | 0.531 | 1.114 | 19 | 157 | $3.15 \cdot 10^4$ |
| | Bag-ROS | 0.527 | 1.052 | 17 | 159 | $3.14 \cdot 10^4$ |
| | Bag-ST | 0.487 | 1.556 | 412 | 3960 | $21.5 \cdot 10^4$ |
| | RF | 0.517 | 1.093 | 1.59 | 189 | $3.15 \cdot 10^4$ |
| | RF-ROS | 0.518 | 1.132 | 1.49 | 208 | $3.16 \cdot 10^4$ |
| | RF-ST | 0.492 | 1.525 | 69 | 4760 | $43.51 \cdot 10^4$ |
| | ET | 0.514 | 0.986 | 18.27 | 180 | $3.48 \cdot 10^4$ |
| | ET-ROS | 0.496 | 1.156 | 18.57 | 201 | $3.50 \cdot 10^4$ |
| | ET-ST | 0.467 | 2.654 | 480 | 4410 | $51.29 \cdot 10^4$ |
| | multi-target PCT | 0.616 | 0.704 | 0.80 | 2.81 | $0.45 \cdot 10^3$ |

is minimal. However, notable differences exist in terms of time needed for learning the model and making predictions. Namely, the multi-target ensembles have significantly lower learning and prediction times than the single-target ensembles. The ROS ensembles train the ensembles faster (for Bagging and Extra trees) but still in the same order of magnitude as the original methods. Not surprisingly, single multi-target PCTs have the shortest learning times at the cost of lowest predictive performance. Similar findings are observed when considering model complexity (measured as total number of nodes in all of the trees in an ensemble or in a multi-target PCT). Average prediction times per instance do not differ across the different approaches. This is expected since all base models are trees and no additional computation overhead is needed to calculate the predictions. Ensembles of single-target PCTs always have an order of magnitude higher learning and prediction times, as well as model complexity as a separate ensemble is learned for predicting each target.

For the *OES 10* dataset, improvements in predictive performance are present. The proposed ROS ensembles outperform their original counterparts. Furthermore, the original ensembles were outperformed by the ensembles of single-target PCTs. The predictive performance gain with ET-ROS w.r.t. ET is substantial. This is an interesting observation and suggests that ROS could lift predictive performance on smaller datasets with larger output spaces, especially for heavily randomized methods such as extra trees. One possible explanation is that the sampling of input variables in ET, coupled with the small number

of examples in the dataset and absence of bootstrapping, introduces a relatively high level of noise in the learning process. The ROS ensemble then actually reduces the effect of this noise at the level of individual base models by specializing them for a smaller output space. This can also explain the small gains for bagging and random forests with the ROS extension on this dataset, because the bootstrapping actually negatively impacts the overall ensemble performance. By inspecting the overfitting score, we note that ROS ensembles consistently exhibit a decreased score w.r.t. ensembles of single-target PCTs and perform comparably w.r.t. ensembles of multi-target PCTs. Learning and prediction times, as well as model complexity, follow similar patterns as for the *Forestry Kras* dataset.

### 5.2.4 Comparison with other output space transformation methods

In order to put ROS in the broader context of MTR methods with output space transformations, we compare the predictive performance of ROS ensembles and ensembles built with the competing methods proposed in (Joly et al., 2014) and (Tsoumakas et al., 2014). We have selected these specific methods because they all specialize individual models in the ensemble to a subset of target variables.

Joly et al. (2014) propose ensembles of multi-output regression trees, where each individual tree is built by using a projected output space. Gaussian, Rademacher, Hadamard and Achlioptas projections are used. The goal is to truncate the output space in order to reduce the number of calculations needed to find the best split, which is the main computational burden while building a decision tree. While learning the ensemble, each tree is given a different output space projection. They use two different ensemble methods: Random forests of multi-output trees and ensembles of extremely randomized multi-output trees. We dubbed their method Random projections and its two variants as RP-RF and RP-ET. Note that Random projections cannot handle nominal attributes and missing values. Hence, the nominal attributes have been converted to numeric scales and missing values have been imputed with the arithmetic mean of that feature.

Tsoumakas et al. (2014) propose an ensemble method called Random Linear Target Combinations for MTR (RLC). They construct new target variables via random linear combinations of existing ones. The data must be normalized in order for the linear combinations to make sense, i.e., to prevent targets on larger scales dominating over the ones with lower scales and thus deteriorating the learning process. The output space is transformed in such a way that each linear combination consists of $k$ original output features. Each combination is then considered for learning one ensemble member. The transformation of the output space matrix $\boldsymbol{Y}$ $(m \times q)$ is achieved via a coefficient matrix $\boldsymbol{C}$ of size $q \times r$ filled with random values uniformly chosen from $[0, 1]$. Columns of the matrix $\boldsymbol{C}$ represent coefficients of the linear combination of the target variables. By multiplying the two matrices, we get the transformed output space $\boldsymbol{Y'} = \boldsymbol{YC}$ $(m \times r)$ that is then used for training. A user-selected regression algorithm can then be applied on the transformed data.

**Data preprocessing.** In order to run the experiments, we had to preprocess the data. Random projections do not know how to handle nominal attributes and missing values. Nominal attributes have been converted to numeric scales. Missing values have been imputed with the arithmetic mean of that feature.

**Method parameters.** All ensembles contain 100 base models. ROS ensembles were parametrized as described in Section 4.1. **Random projections** variants (RP-RF, RP-ET) use $m = \log(|\mathsf{A}_t|)$ components in the projected output space where $\mathsf{A}_t$ is the set of target

attributes. In addition, Rademacher random projections were used for output space transformations. RP-RF used $k = \sqrt{|\mathsf{A}_d|}$ randomly chosen input features to calculate splits where $\mathsf{A}_d$ is the set of all input features. RP-ET used $k = |\mathsf{A}_d|$. Minimal number of allowed instances in a leaf node was set to 1 for both variants. The code for both variants of Random projections is available online[1]. RLC was parametrized to use gradient boosting with 4-terminal node regression tree as the base regressor with learning rate of 0.1 and 100 boosting iterations. Number of targets that participate in the random linear combinations was set to $k = 2$. RLC is implemented as part of the MULAN library[2](Tsoumakas, Spyromitros-Xioufis, Vilcek, & Vlahavas, 2011). All competing method were parametrized according to the recommendations of their respective authors.

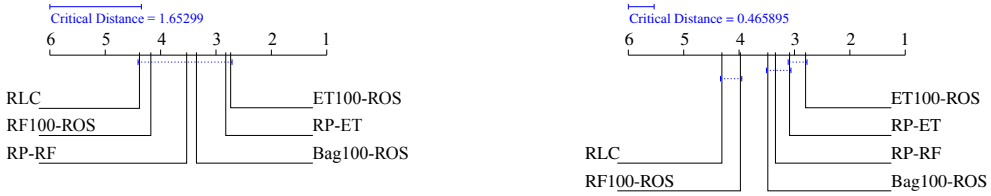
Table 5.4: Predictive performance of ROS ensembles, ensembles of Random projections variants (RP-RF, RP-ET) and RLC ensembles in terms of aRRMSE. Lower values mean better performance. Underlined numbers denote the best performing method on a given dataset.

| no. | Dataset | RP-RF | RP-ET | RLC | Bag-ROS | RF-ROS | ET-ROS |
|---|---|---|---|---|---|---|---|
| 1 | ADNI | 0.941 | 0.936 | 0.94 | 0.923 | 0.918 | 0.916 |
| 2 | ATP 1D | 0.435 | 0.427 | 0.415 | 0.38 | 0.412 | 0.374 |
| 3 | ATP 7D | 0.413 | 0.326 | 0.358 | 0.467 | 0.517 | 0.436 |
| 4 | Forestry Kras | 0.558 | 0.583 | 0.629 | 0.548 | 0.546 | 0.557 |
| 5 | OES 10 | 0.468 | 0.467 | 0.447 | 0.527 | 0.518 | 0.496 |
| 6 | OES 97 | 0.535 | 0.513 | 0.513 | 0.572 | 0.528 | 0.518 |
| 7 | PPMI | 0.73 | 0.757 | 0.767 | 0.739 | 0.75 | 0.746 |
| 8 | Prespa lake top 10 | 0.831 | 0.827 | 0.811 | 0.937 | 0.939 | 0.928 |
| 9 | RF 1 | 0.14 | 0.136 | 0.238 | 0.145 | 0.152 | 0.147 |
| 10 | RF 2 | 0.145 | 0.151 | 0.241 | 0.147 | 0.168 | 0.153 |
| 11 | Sales | 0.646 | 0.647 | 0.707 | 0.69 | 0.733 | 0.698 |
| 12 | SCM 1D | 0.311 | 0.285 | 0.376 | 0.305 | 0.31 | 0.281 |
| 13 | SCM 20D | 0.366 | 0.324 | 0.595 | 0.354 | 0.379 | 0.316 |
| 14 | Soil resilience | 0.847 | 0.845 | 0.761 | 0.872 | 0.872 | 0.873 |
| 15 | Vegetation clustering | 0.702 | 0.702 | 0.79 | 0.699 | 0.703 | 0.696 |
| 16 | Vegetation condition | 0.605 | 0.603 | 0.647 | 0.603 | 0.603 | 0.599 |
| 17 | Water quality | 0.902 | 0.901 | 0.91 | 0.899 | 0.899 | 0.894 |
| | Win count | 3 | 3 | 4 | 0 | 1 | 7 |

We present the results using average rank diagrams in Figure 5.4, while the complete experimental results are available in Appendix (A.1, A.2). Figure 5.4 depicts two average rank diagrams: one per dataset and one per target. The per dataset diagram is based on the aRRMSE values, one per dataset. The per target diagram is based on RRMSE values with multiple targets per dataset. The *per dataset analysis* shows no statistically significant differences between the predictive performances of the considered ensemble methods. We can however note that performances of ET-ROS and RP-ET ensembles seem to be on par (with a minimal advantage of the ET-ROS ensemble). The *per target analysis* detects two

[1]Random projections code is available at https://github.com/arjoly/random-output-trees.
[2]MULAN library is available at http://mulan.sourceforge.net

statistically significant differences. First, ET-ROS statistically significantly outperforms all other methods with the exception of RP-ET. Second, RLC and RF-ROS ensembles are on par and both are statistically significantly outperformed by the other methods. Third, Bag-ROS, RP-RF and RP-ET perform equally well. All in all, ET-ROS ensembles generally perform better than the other considered ensemble methods.




(a) Per dataset (aRRMSE)

(b) Per target (RRMSE)

Figure 5.4: Average rank diagrams showing the predictive performance of ROS (ET-ROS, Bag-ROS and RF-ROS), Random projections (RP-ET and RP-RF) and RLC ensembles. Lower ranks are better.

## 5.3 Case Study: Predicting the Power Consumption of the Mars Express Probe

Mars Express (MEX) is a spacecraft operated by the European Space Agency (ESA). It has been orbiting around the planet Mars since the end of 2003 and is scheduled to do so until (at least) 2022. MEX carries a payload of seven state-of-the-art scientific instruments that are used to record data for at least one Martian year (687 Earth days). The recorded data and spacecraft control parameters are transmitted from/to Earth using the on-board data relay system. The MEX mission facilitated some major discoveries, such as evidence of liquid water above and below the surface of the planet (Orosei et al., 2018), an ample amount of three-dimensional renders of its surface as well as the most complete map of the chemical composition of its atmosphere (Chicarro, Martin, & Trautner, 2004).

Science operations on MEX are possible when the scientific instruments have enough power to run. The spacecraft runs on electric power coming from the solar arrays. A certain amount of produced electricity is used to charge the batteries (used during eclipses). As is the case with all scientific equipment, instruments on MEX were designed to work in predefined temperature ranges. In space, temperature changes drastically and rapidly. To meet the temperature requirements of the installed instruments, MEX is equipped with a thermal subsystem, responsible for maintaining the temperature of the whole spacecraft within its operating range. The thermal subsystem must operate in a very efficient manner due to power limitations. This is a challenging task, especially when we take into account the power needed for scientific research. The produced power is first and foremost spent on running the spacecraft. If the remaining power allows for running the instruments, scientific work is possible. It is immediately obvious that poor decisions of the operators, malfunctions, or eclipses can substantially reduce the amount of possible scientific work.

Experiments must be carefully planned and optimized before they are sent to MEX. A crucial step of this process pertains to the estimation of required power. If the available power is overestimated, experimental plans will not be executed. In those cases, MEX cannot provide enough power. In contrast, if the available power is underestimated, the spacecraft is not fully utilized.

This case study is inspired by the Mars Express Power Challenge organized by the European Space Agency in 2016. The challenge was to predict the consumption of the electric current at 33 different thermal heaters in MEX. The data for 3 Martian years was provided as a training set, while the data from the fourth Martian year served as the testing/evaluation set. The raw data was composed of information about the spatial orientation and alignment of the probe with regards to the Sun, Mars and the Earth including eclipses (umbras), as well as of the MEX's flight dynamics and information about the activations and deactivations of subsystems.

### 5.3.1 Dataset

For the purpose of the Mars Express Power Challenge, telemetry data of MEX was made publicly available. The data consisted of electric current measurements of the 33 power consumers and context data consisting of five components:

- **SAA** (Solar Aspect Angles) data contain the angles between the Sun–MEX line and the axes of the MEX's coordinate system.
- **DMOP** (Detailed Mission Operations Plans) data contain the 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.
- **EVTF** (Miscellaneous Events) 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.
- **LTDATA** (Long Term Data) 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). Detailed description of the data is available in Boumghar, Lucas, and Donati (2018), Lucas and Boumghar (2017). The raw telemetry is not directly usable with the ML methods described in this dissertation. Therefore, a new dataset had to be constructed, containing carefully crafted features generated from the raw telemetry data. The newly-generated features include: (i) energy influx features, estimating the influx of solar energy based on solar angles and MEX position with respect to the Sun, (ii) historical energy influx features, representing energy influx estimations calculated in the previous point for several time intervals, (iii) DMOP and FTL features, encoding several events that were triggered in MEX. The original dataset has a resolution of 1 minute. In our experiments, we use the time resolution of 15 minutes. The resampling process of the train and test datasets is described in Petković et al. (2019). The train and test datasets contain 49118 and 16375 examples, respectively. Each example is described with 448 descriptive features.

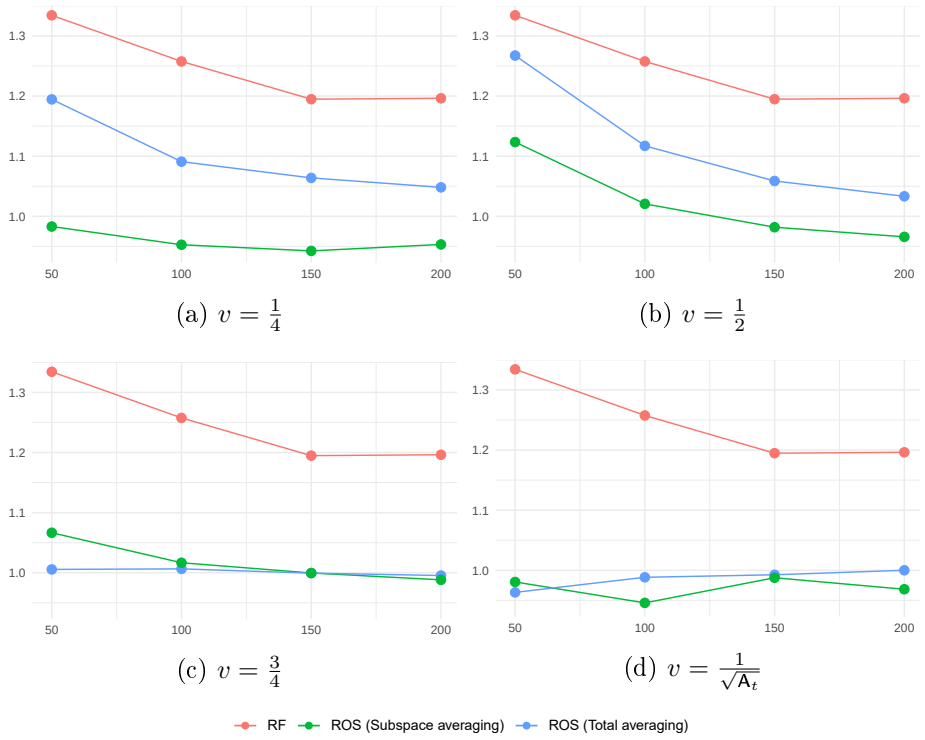
### 5.3.2 Experimental design

In this case study we focus on learning ensembles of PCTs for MTR. We will be using the features described in the previous section to predict the electric current of the 33 power consumers at 15-minute intervals.

The current state-of-the-art in MTR are random forest ensembles of PCTs (RF, Kocev et al. (2013)) that also performed well in the MEX Power Challenge (Breskvar, Kocev,

Levatić, et al., 2017). The previous chapters of this dissertation presented tree ensembles with ROS. In this chapter, we use ET-ROS, tree ensembles of extremely randomized PCTs whose models were shown to outperform the models of competing methods, including RF. We will compare the predictive performance of RF and ET-ROS in terms of aRRMSE (see Equation 5.1).

Both ensemble methods were instantiated to use minimally 8 examples in leaf nodes. RF uses $\frac{1}{4}$ of the descriptive space whereas ET-ROS uses all descriptive attributes. ROS ensembles were parametrized to use four different subspace sizes ($v \in (\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{1}{\sqrt{|\mathsf{A}_t|}})$) and two different prediction aggregation functions, namely total and subspace averaging.




(a) $v = \frac{1}{4}$ (b) $v = \frac{1}{2}$

(c) $v = \frac{3}{4}$ (d) $v = \frac{1}{\sqrt{\mathsf{A}_t}}$

Figure 5.5: Performance of RF and different variants of ET-ROS in terms of aRRMSE. Each subfigure presents the performance of ET-ROS models with a different ROS output subspace size ($v$), using one of the two possible prediction aggregation functions, as compared to RF. Horizontal axes represent ensemble sizes in terms of number of trees in the ensemble.

### 5.3.3 Results

The results of the comparison between the current state-of-the-art in MTR (RF) and ET-ROS variants are presented in Figure 5.5. Each subfigure represents the performance of ET-ROS models with a different ROS output subspace size ($v$), using one of the two possible prediction aggregation functions, compared to RF. The RF results are the same across all four subfigures, i.e., ROS subspace size does not apply to this method. Horizontal axes represent ensemble sizes in terms of number of trees in the ensemble. Vertical axes represent aRRMSE.

The first thing to notice is the saturation of all ensembles. The results suggest convergence at 150 base models in the ensembles. Next, results show that all variants of ET-ROS ensembles outperform RF ensembles. Moreover, ET-ROS ensembles with subspace aver-

aging perform better than the ET-ROS ensembles with total averaging, which is in line with the recommendation for this parameter in Section 5.2.2. The recommended ROS output space size ($v = \frac{3}{4}$) performs better than RF, but is in fact the worst performing ROS output space size. The optimal choice for this dataset is $v = \frac{1}{4}$, which actually had the worst average performance in the benchmark experiments (see Figure 5.2). This does not come as a surprise and confirms the mentioned volatility of this parameter. Ideally, this parameter would have been tuned for each dataset separately.

## 5.4 Summary

We summarize the main findings of the extensive experimental work presented in this chapter by answering the experimental questions posed in Section 5.1.1.

1. **How many base models do we need in ROS ensembles in order to reach the point of performance saturation?**
   The saturation point of the original PCT ensembles is between 50 and 75 base models. Bag-ROS and ET-ROS ensembles saturate between 50 and 100 base models. Especially RF-ROS ensembles saturate a bit later, at 75 to 100 base models learned. In the comparative analysis of performance, we consider ensembles with 100 base models (in order to make the comparison fair for all considered methods).

2. **What is the best value for the portion of target space to be used within such ensembles? Is this portion equal for all evaluated ensemble methods?**
   The most appropriate size of the portion of target space to be used varies with the ensemble method. The results suggest to use $v = \frac{1}{2}$ for Bag-ROS and $v = \frac{3}{4}$ for RF-ROS and ET-ROS. However, the best value of $v$ may be problem-dependent.

3. **Does it make sense to change the default aggregation function of the ensemble that uses the prediction for all targets? Can this improve predictive performance?**
   Changing the aggregation function changes the behaviour of the ROS ensembles. For Bag-ROS, it can even decrease the predictive performance, so we recommend using the standard aggregation function, i.e., total averaging. For RF-ROS and ET-ROS we recommend making predictions with subspace averaging.

4. **Considering predictive performance, how do ROS ensemble methods compare to the original ensemble methods?**
   Using ROS can improve the predictive performance of PCT ensembles. This is especially notable when using ET-ROS with small datasets with larger output spaces.

5. **Is ROS helpful in terms of time efficiency?**
   The observed learning times for ROS methods can be substantially lower than the ones of their original counterparts. This especially holds for large datasets. Prediction times, however, do not change.

6. **Do ROS models use less memory than the models trained with the original ensemble methods?**
   Ensemble models obtained with ROS have sizes comparable to the ensemble models produced by the original ensemble models.

7. **How the ROS models compare to other output transformation methods?**
   ET-ROS ensembles generally perform better than ensembles of other competing output transformation methods.

We also illustrate the practical usability of the proposed tree ensembles with ROS on the case study of the Mars Express Power Challenge dataset. The results show that ET-ROS yield better predictive performance than the state-of-the-art method (RF) on this dataset.

# Chapter 6

# Evaluation of Tree Ensembles with Random Output Selections for Multi-Label Classification

In this chapter, we address the task of multi-label classification (MLC), described in Section 2.2.2. We solve the task of MLC by learning ensembles of predictive clustering trees (PCTs). In particular, we learn random forest ensembles of PCTs that use ROS: the proposed ensemble extension method, described in Chapter 4.

The main focus of this study is to determine whether the proposed method can improve the predictive performance and shorten the learning times of the considered ensemble methods. An extensive empirical evaluation over a variety of benchmark datasets is performed in order to determine the effects of using ROS on predictive performance. In addition to this, we also perform an analysis with respect to time and space complexity. The results show that the proposed ensemble extension can yield better predictive performance, reduce learning time or both, without a considerable change in model size. The newly proposed aggregation function gives the best results when used with ensembles of extremely randomized PCTs.

We perform an extensive empirical evaluation on 13 benchmark datasets, which provides performance assessment for the original and extended ensemble method over 12 different evaluation measures. The study also includes parameter setting recommendations for the proposed method. Moreover, we compare the performance to other competing methods that also transform the output space. The details on this item are available in this chapter.

This chapter is composed of two sections. The first section provides details regarding the experimental design. We start by posing experimental questions. We also explain and describe the evaluation measures used in this chapter. Then, we describe the benchmark datasets used for the evaluation of the proposed method. Finally, we describe the experimental setup, specifically designed to answer the posed questions. The second section of this chapter contains results of the experiments and a comparison with the competing methods.

## 6.1 Experimental Design

To evaluate the performance of RF-ROS, we performed experiments for the considered task of MLC. This section presents: (i) the experimental questions posed, (ii) benchmark datasets, (iii) the experimental setup, and (iv) the evaluation measures used.

### 6.1.1 Experimental questions

We designed the experimental evaluation having the following research questions in mind:

1. What is the recommended label subspace size to use with RF-ROS ensembles?
2. Does it make sense to change the aggregation function, i.e., can subspace averaging improve the predictive performance of RF-ROS models?
3. Considering predictive performance, how do RF-ROS ensembles compare to other competing methods?

### 6.1.2 Evaluation measures

In order to determine the predictive performance of the induced models, we empirically evaluate them with 12 different measures that belong to two groups: example and label-based measures. Results in terms of different measures suggest the same conclusions. We therefore present results only in terms of Example-based $F_1$ (more is better) and Hamming loss (less is better). The results for remaining evaluation measures are in Appendix B.

We evaluate our models based on the measures presented in Madjarov, Kocev, et al. (2012). In the definitions below, $\mathcal{Y}_i$ denotes the set of true labels of example $x_i$ and $h(\mathbf{x_i})$ denotes the set of predicted labels for the same examples. All definitions refer to the multi-label setting. $N$ represents the number of examples in the dataset. $Q$ represents the number of all possible labels, i.e., $Q = |\mathcal{L}|$.

#### Example-based measures

**Hamming loss.** This measure evaluates how many times an example-label pair is misclassified, i.e., label not belonging to the example is predicted or a label belonging to the example is not predicted. The smaller the value of $\mathsf{HammingLoss}(h)$, the better the performance. The performance is perfect when $\mathsf{HammingLoss}(h) = 0$. This metric is defined as:

$$\mathsf{HammingLoss}(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Q}\left|h(\mathbf{x_i})\Delta\mathcal{Y}_i\right| \tag{6.1}$$

where $\Delta$ stands for the symmetric difference between two sets.

**Accuracy.** The accuracy for a single example $\mathbf{x_i}$ is defined by the Jaccard similarity coefficient between the label sets $h(\mathbf{x_i})$ and $\mathcal{Y}_i$, i.e., $\frac{|h(\mathbf{x_i})\bigcap\mathcal{Y}_i|}{|h(\mathbf{x_i})\bigcup\mathcal{Y}_i|}$. Accuracy over $N$ examples is averaged over all accuracies of single examples.

$$\mathsf{Accuracy}(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{|h(\mathbf{x_i})\bigcap\mathcal{Y}_i|}{|h(\mathbf{x_i})\bigcup\mathcal{Y}_i|} \tag{6.2}$$

**$F_1$ measure.** The $F_1$ measure is defined as the harmonic mean of precision and recall. **Precision** is defined as:

$$\mathsf{Precision}(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{|h(\mathbf{x_i})\bigcap\mathcal{Y}_i|}{|\mathcal{Y}_i|} \tag{6.3}$$

**Recall** is defined as:

$$\mathsf{Recall}(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{|h(\mathbf{x_i})\bigcap\mathcal{Y}_i|}{|h(\mathbf{x_i})|} \tag{6.4}$$

Finally, $\mathbf{F_1}$ is defined as:

$$\mathsf{F}_1 = \frac{1}{N}\sum_{i=1}^{N}\frac{2\cdot|h(\mathbf{x_i})\cap\mathcal{Y}_i|}{|h(\mathbf{x_i})|+|\mathcal{Y}_i|} \tag{6.5}$$

$\mathsf{F}_1$ is an example-based metric and its value is an average over all examples in the dataset. $\mathsf{F}_1$ reaches its best value at 1 and worst score at 0.

**Subset accuracy.** Subset accuracy, also called classification accuracy, is a very strict evaluation measure as it requires the predicted set of labels to be an exact match of the true set of labels. Here, $\mathbb{1}$ is an indicator function with the following behavior: $\mathbb{1}(true) = 1$ and $\mathbb{1}(false) = 0$. Subset accuracy is defined as:

$$\mathsf{SubsetAccuracy}(h) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(h(\mathbf{x_i}) = \mathcal{Y}_i) \tag{6.6}$$

### Label-based measures

**Macro-averaged measures.** Here, $\mathsf{TP}_j, \mathsf{FP}_j$ and $\mathsf{FN}_j$ denote the number of true positives, false positives and false negatives respectively. The index $j$ refers to the label $\lambda_j$, i.e., $\lambda_j \in \mathbb{L}$.
Macro-averaged precision is defined as:

$$\mathsf{Precision}_{macro} = \frac{1}{Q}\sum_{j=1}^{Q}\frac{\mathsf{TP}_j}{\mathsf{TP}_j+\mathsf{FP}_j} \tag{6.7}$$

Macro-averaged recall is defined as:

$$\mathsf{Recall}_{macro} = \frac{1}{Q}\sum_{j=1}^{Q}\frac{\mathsf{TP}_j}{\mathsf{TP}_j+\mathsf{FN}_j} \tag{6.8}$$

Macro-averaged $\mathrm{F}_1$ is defined as the harmonic mean of precision and recall, where the average is calculated on a per label basis and is then averaged across all labels. If $p_j$ and $r_j$ are the precision and recall for all $\lambda_j \in h(\mathbf{x_i})$ from $\lambda_j \in \mathcal{Y}_i$, the macro-averaged $\mathrm{F}_1$ is calculated as:

$$\mathsf{F}_{1macro} = \frac{1}{Q}\sum_{j=1}^{Q}\frac{2\cdot p_j\cdot r_j}{p_j+r_j} \tag{6.9}$$

**Micro-averaged measures.** Micro-averaged precision is defined as:

$$\mathsf{Precision}_{micro} = \frac{\sum_{j=1}^{Q}\mathsf{TP}_j}{\sum_{j=1}^{Q}\mathsf{TP}_j+\sum_{j=1}^{Q}\mathsf{FP}_j} \tag{6.10}$$

Micro-averaged recall is defined as:

$$\mathsf{Recall}_{micro} = \frac{\sum_{j=1}^{Q}\mathsf{TP}_j}{\sum_{j=1}^{Q}\mathsf{TP}_j+\sum_{j=1}^{Q}\mathsf{FN}_j} \tag{6.11}$$

Micro-averaged $\mathrm{F}_1$ is defined as the harmonic mean of micro-averaged precision and micro-averaged recall. The micro-averaged $\mathrm{F}_1$ is calculated as:

$$\mathsf{F}_{1micro} = \frac{2\cdot\mathsf{Precision}_{micro}\cdot\mathsf{Recall}_{micro}}{\mathsf{Precision}_{micro}+\mathsf{Recall}_{micro}} \tag{6.12}$$

### 6.1.3 Data description

To evaluate the proposed and existing methods, we use 13 publicly available benchmark datasets from four different domains: multimedia, biology, text and audio. Table 6.1 shows the main characteristics of the considered datasets. In order to have as general evaluation as possible, we use datasets of different sizes in terms of the number of instances, number of descriptive and number of target attributes. All datasets come predivided into train and test sets and we use them as such.

Table 6.1: Properties of the datasets: number of train/test examples ($\mathsf{N}_{train}/\mathsf{N}_{test}$), number of descriptive attributes ($|\mathsf{A}_d|$), and number of target labels ($|\mathsf{A}_t|$).

| no. | Dataset | Domain | $\mathsf{N}_{train}$ | $\mathsf{N}_{test}$ | $\lvert\mathsf{A}_d\rvert$ | $\lvert\mathsf{A}_t\rvert$ |
|---|---|---|---|---|---|---|
| 1 | **Bibtex** | Text | 4880 | 2515 | 1836 | 159 |
| 2 | **Birds** | Audio | 322 | 323 | 260 | 19 |
| 3 | **Bookmarks** | Text | 60000 | 27856 | 2150 | 208 |
| 4 | **Corel** | Multimedia | 4500 | 500 | 499 | 374 |
| 5 | **Delicious** | Text | 12920 | 3185 | 500 | 983 |
| 6 | **Emotions** | Multimedia | 391 | 202 | 72 | 6 |
| 7 | **Enron** | Text | 1123 | 579 | 1001 | 53 |
| 8 | **Genbase** | Biology | 463 | 199 | 1185 | 27 |
| 9 | **Mediamill** | Multimedia | 30993 | 12914 | 120 | 101 |
| 10 | **Medical** | Text | 645 | 333 | 1449 | 45 |
| 11 | **Scene** | Multimedia | 1211 | 1159 | 294 | 6 |
| 12 | **TMC 2007** | Text | 21519 | 7077 | 500 | 22 |
| 13 | **Yeast** | Biology | 1500 | 917 | 103 | 13 |

**Bibtex.** The data (Katakis, Tsoumakas, & Vlahavas, 2008) is based on the data from the ECML/PKDD 2008 discovery challenge. It contains bibtex entries from the BibSonomy social bookmark and publication sharing system, annotated with a subset of the tags assigned by users (e.g. statistics, quantum, data mining etc.). The title and abstract of entries were used to construct features using the boolean bag-of-words model.

**Birds.** The dataset contains audio recordings of vocalizing birds (Briggs et al., 2013). Each recording is 10 seconds long and can contain simultaneous vocalizations of several bird species. The goal is to detect the set of bird species that are present in the audio recording.

**Bookmarks.** The data (Katakis et al., 2008) is based on the data from the ECML/PKDD 2008 discovery challenge. It contains metadata of bookmark entries from the BibSonomy system such as the URL of the web page, an URL hash, a description of the web page, etc. The descriptive attributes represent terms in the document.

**Corel.** The data (Barnard et al., 2003) is based on 5000 Corel images. Images were segmented and then only regions larger than a threshold were clustered into 499 blobs using k-means, which are the features used to describe the image.

**Delicious.** This dataset contains textual data from the *Delicious* website, along with their tags such as academia, airline, algorithm, etc. (Tsoumakas et al., 2008). This dataset is a modified tagging problem where the label space was not defined prior to labeling and the size of the label space is greater than the size of the input space. Attributes are binary and represent the presence of a word in the document.

**Emotions.** This is a small dataset concerning classification of music into the emotions that it evokes (Trohidis, Tsoumakas, Kalliris, & Vlahavas, 2008). The emotions are described according to the Tellegen-Watson-Clark model of mood: amazed-surprised, happy-pleased, relaxing-clam, quiet-still, sad-lonely, and angry-aggressive.

**Enron.** This is a subset of the Enron email text corpus (Klimt & Yang, 2004). It is based on a collection of emails exchanged between the Enron Corporation employees, which were made available during a legal investigation. It contains emails that were categorized into 53 topic categories, such as company strategy, humor and legal advice.

**Genbase.** This is a dataset for classification of protein functions (Diplaris, Tsoumakas, Mitkas, & Vlahavas, 2005). Instances are protein chains represented by using a motif sequence vocabulary of fixed size. Each sequence is encoded as a binary array where each bit is 1 if the corresponding motif is present and 0 otherwise. Each label identifies the functional family of the sequence.

**Mediamill.** This is a multimedia dataset for generic video indexing (Snoek, Worring, Van Gemert, Geusebroek, & Smeulders, 2006). It was extracted from the TRECVID 2005/2006 benchmark. The dataset contains 85 hours of international broadcast news data categorized into 101 semantic concepts (e.g. car, golf, bird, etc.), and each video instance is represented as a numeric vector of 120 features including visual and textual information.

**Medical.** This dataset is based on the data made available during the Computational Medicine Center's 2007 Medical Natural Language Processing Challenge (Pestian et al., 2007). It consists of clinical free-text radiology reports labelled with ICD-9-CM disease codes.

**Scene.** This dataset contains information on images that were annotated with up to 6 concepts, such as beach, mountain, etc. (Boutell et al., 2004). Each image is described in terms of visual numeric features corresponding to spatial color moments in the LUV space.

**TMC 2007.** The SIAM Text Mining Competition (TMC) 2007 dataset is a subset of the Aviation Safety Reporting System (ASRS) dataset (Srivastava & Zane-Ulman, 2005). The dataset contains free text form reports on aviation safety that the flight crews submit after completion of each flight. The goal is to label the documents with respect to what types of problems they describe.

**Yeast.** This dataset contains features about microarray expressions and phylogenetic profiles for 2417 yeast genes (Elisseeff & Weston, 2002). Each gene is annotated with a subset of 14 functional categories (e.g. metabolism, energy, etc.) from the top level of the functional catalog (FunCat3).

### 6.1.4 Experimental setup

We designed the experiments to answer the experimental questions which we pose in Section 6.1.1. To evaluate the performance of the RF-ROS models, we generated ensembles with different output space sizes: $v \in (\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{1}{\sqrt{|\mathsf{A}_t|}}, \frac{\log|\mathsf{A}_t|}{|\mathsf{A}_t|})$. We also experimented with two prediction aggregation functions, namely total and subspace averaging.

In order to evaluate the predictive performance of RF-ROS ensembles in a broader context, we also compare their performance to the performance of: (i) Random forests of standard PCTs, dubbed RF (Kocev et al., 2013), (ii) Random $k$-Labelsets models, dubbed RAkEL (Tsoumakas & Vlahavas, 2007), and (iii) Random projections with Gaussian subspaces, dubbed RP-RF (Joly et al., 2014).

**Parameter selection.** RF and RF-ROS ensembles used 100 PCTs, which is the point at which the ensembles are typically saturated. The size of descriptive space randomly sampled was set to $|\mathsf{A}'_d| = \lfloor 0.1 \cdot |\mathsf{A}_d| \rfloor + 1$, following the suggestion from Kocev et al. (2013). The trees in ensembles were not pruned (Bauer & Kohavi, 1999).

RAkEL models were initialized with the $k$ parameter (size of labelset) set to $k = |\mathsf{A}_t|/2$. The number of models was set to $min(2 \cdot |\mathsf{A}_t|, 100)$. Support vector machine (SVM) classifier was selected as learning algorithm with a linear kernel and the complexity constant $C = 1$. All parameter settings were set according to the suggestions in Tsoumakas and Vlahavas (2007).

RP-RF ensembles were built with 100 decision trees in the ensemble. The number of Gaussian subspace components was set to $m = \log|\mathsf{A}_t|$. Minimal number of examples in a leaf node was set to $n_{min} = 1$. The size of descriptive space randomly sampled was set to $k = \sqrt{|\mathsf{A}_d|}$. All parameter settings were initialized according to the suggestions in Joly et al. (2014).

**Environments.** The compared methods are implemented in different environments. RAkEL is implemented in Java programming language and can be run from the MULAN library (Tsoumakas et al., 2011).[1] RF and RF-ROS are also implemented in Java and can be run from the CLUS software package. Random projections with Gaussian subspaces are implemented on top of scikit-learn: a library written in Python programming language [2]. Evaluation of listed methods was done by collecting predictions from all models and then calculating all evaluation measures separately in order to avoid mistakes due to implementation differences.

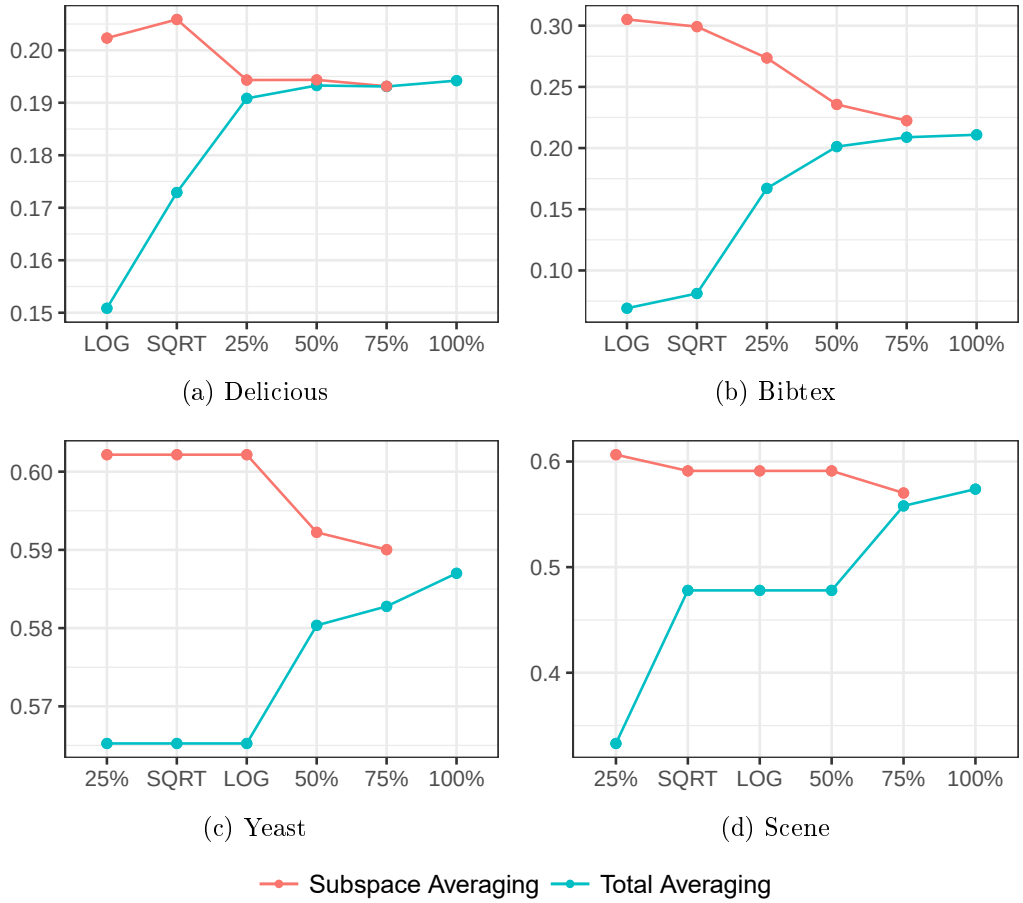
All methods were run on a computing infrastructure where each job had 60 GB of available memory and one week to produce a model for one dataset. The CPUs on the computing infrastructure are AMD Opteron 6134 @ 800 MHz or better, running Fedora Linux operating system.

**Statistical evaluation.** A statistical evaluation of the results was performed according to the guidelines in (J. Demšar, 2006). Friedman test with the correction was used to determine the statistical significance (M. Friedman, 1940; Iman & Davenport, 1980). Additional post-hoc test has been applied to detect statistically significant differences between considered methods (Nemenyi, 1963). Results are presented with average rank diagrams, where a line connects methods that do not differ with statistical significance (J. Demšar, 2006). All statistical tests were conducted at the significance level $\alpha = 0.05$ using three decimal places. Not all RAkEL models managed to produce results on all benchmarking

[1] MULAN library is available at http://mulan.sourceforge.net.

[2] Random projections code is available at https://github.com/arjoly/random-output-trees.

datasets. We did not want to discard the results of other methods on those datasets, so we gave RAkEL the worst possible score before the statistical evaluation (1 for Hamming loss and 0 for all other measures).




(a) Delicious (b) Bibtex

(c) Yeast (d) Scene

Figure 6.1: Example-based $F_1$ results for Delicious, Bibtex, Yeast and Scene datasets. Total averaging with 100% output space size represents the RF method without ROS.

## 6.2 Results

In this section, we present the experimental results. We divide the presentation into two parts. Section 6.2.1 focuses on parametrization of the proposed method whereas Section 6.2.2 shows how RF-ROS models perform against competition. We also revisit the experimental questions at the end of this section.

### 6.2.1 ROS parameter selection

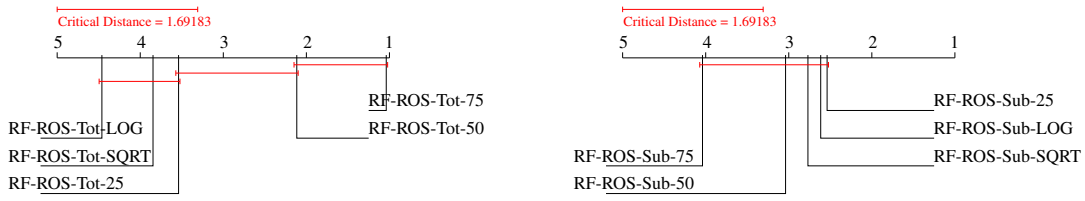
The proposed method has two degrees of freedom: subspace size and prediction aggregation function. We experimented with different values for both. Figure 6.1 shows the performance of RF-ROS on four datasets with various label and example counts. Plots also show a specific data point (total averaging, 100% target space, always the rightmost data point) which represents the performance of RF model on that dataset.

The results suggest that subspace averaging outperforms total averaging which is especially true for subset sizes below 50%. Moreover, considered aggregation functions exhibit inverse behavior w.r.t. the target space size. Total averaging performs better with larger

spaces while subset averaging benefits from smaller ones. When the target space size increases, both variants converge to similar performance as the original RF method. This behavior is expected because larger subset size leads to larger overlap between all target variables and subsets of them. Conversely, with smaller subset sizes, the overlap is smaller and the randomization effect is larger, hence the differences are larger.

We also observe that the performance of models with different aggregation functions converges at different rates. Although we observe convergence towards RF on all datasets, we speculate that the convergence rate is dataset dependent. For instance, on the Delicious dataset, both variants already converge with target subspace size of 25%. On Bibtex dataset, this number is a bit higher (50%) and on Yeast and Scene datasets even higher (75%). This suggests that smaller output space sizes should be favored for subspace averaging and large for total averaging. When the output space size approaches 100%, convergence toward the original ensemble performance is inevitable.

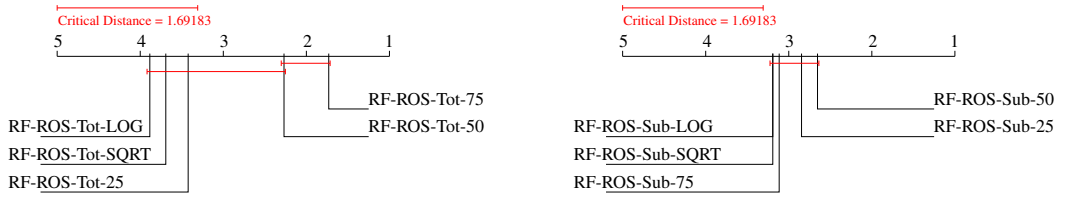
Figures 6.2 and 6.3 show average rank diagrams that confirm our speculations. Figures 6.2a and 6.3a show some statistically significant differences so we recommend the bigger subspace size ($v = \frac{3}{4}$) when using total averaging which is still computationally more efficient than using the whole output space. Figures 6.2b and 6.3b do not show any statistically significant differences between the considered RF-ROS variants. Nevertheless, we recommend the smallest evaluated subspace size ($v = \frac{\log |\mathsf{A}_t|}{|\mathsf{A}_t|}$) to be used with subspace averaging.




(a) Total averaging

(b) Subspace averaging

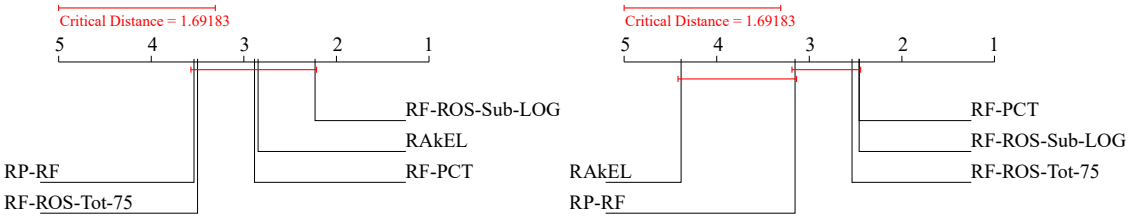
Figure 6.2: Average rank diagrams for RF-ROS in terms of Example-based $F_1$ measure. Lower ranks are better.




(a) Total averaging

(b) Subspace averaging

Figure 6.3: Average rank diagrams for RF-ROS in terms of Hamming loss measure. Lower ranks are better.




(a) Example-based $F_1$ (b) Hamming loss

Figure 6.4: Average rank diagrams for RF-ROS. Lower ranks are better.

### 6.2.2 Comparison with competing methods

We compared model performances of RF-ROS variants using the recommended parameters to RF, RAkEL and RP-RF ensembles. The average rank diagrams are shown in Figure 6.4. We generated them from results shown in Tables 6.2 and 6.2. The diagrams do not show any statistical significance in terms of Example-based $F_1$.

Table 6.2: The performance of considered methods in terms of Example-based $F_1$ (more is better). DNF (did not finish) denotes algorithms that did not produce results. Numbers in bold denote the best performance. Tot-75 denotes ROS ensembles with total averaging, using $v = \frac{3}{4}$. Sub-LOG denotes ROS ensembles with subspace averaging, using $v = \frac{\log |\mathsf{A}_t|}{|\mathsf{A}_t|}$.

| no. | Dataset | RAkEL | RP-RF | RF | RF-ROS | |
|---|---|---|---|---|---|---|
| | | | | | Tot-75 | Sub-LOG |
| 1 | **Bibtex** | DNF | 0.173 | 0.211 | 0.209 | **0.305** |
| 2 | **Birds** | **0.658** | 0.51 | 0.566 | 0.556 | 0.579 |
| 3 | **Bookmarks** | DNF | 0.2 | **0.206** | 0.203 | 0.175 |
| 4 | **Corel** | DNF | 0.018 | 0.007 | 0.009 | **0.089** |
| 5 | **Delicious** | DNF | **0.237** | 0.194 | 0.193 | 0.202 |
| 6 | **Emotions** | **0.637** | 0.534 | 0.574 | 0.582 | 0.588 |
| 7 | **Enron** | **0.562** | 0.508 | 0.527 | 0.518 | 0.559 |
| 8 | **Genbase** | **0.996** | 0.991 | 0.981 | 0.981 | 0.986 |
| 9 | **Mediamill** | DNF | 0.545 | **0.549** | 0.547 | 0.541 |
| 10 | **Medical** | **0.789** | 0.515 | 0.673 | 0.669 | 0.683 |
| 11 | **Scene** | **0.681** | 0.413 | 0.574 | 0.558 | 0.591 |
| 12 | **TMC 2007** | **0.81** | 0.992 | 0.908 | 0.902 | 0.926 |
| 13 | **Yeast** | **0.64** | 0.573 | 0.587 | 0.583 | 0.602 |

It is immediately visible that RAkEL performs very well. Although it did not finish on five datasets, it can still be considered as a serious competitor on datasets with smaller label spaces. However, its predictive performance comes with a big computational cost. This method is hindered by the fact that it uses label powersets and SVMs to generate models, which makes running times of RAkEL substantially longer than those of other methods considered in this study. RAkEL is not the clear winner w.r.t. the average rank diagrams because the method was penalized for not finishing as we described in Section 6.1.4. If we take RAkEL out of consideration, the average rank diagrams in Figure 6.4 suggest that the proposed method performs at least as good as the competition.

RF-ROS-Sub-LOG is ranked better than RF in terms of Example-based $F_1$ measure

Table 6.3: The performance of considered methods in terms of Hamming loss (less is better). DNF (did not finish) denotes algorithms that did not produce results. Numbers in bold denote the best performance. Tot-75 denotes ROS ensembles with total averaging, using $v = \frac{3}{4}$. Sub-LOG denotes ROS ensembles with subspace averaging, using $v = \frac{\log|\mathsf{A}_t|}{|\mathsf{A}_t|}$.

| no. | Dataset | RAkEL | RP-RF | RF | RF-ROS | |
|---|---|---|---|---|---|---|
| | | | | | Tot-75 | Sub-LOG |
| 1 | **Bibtex** | DNF | 0.014 | **0.013** | **0.013** | **0.013** |
| 2 | **Birds** | 0.05 | 0.048 | 0.044 | 0.044 | **0.043** |
| 3 | **Bookmarks** | DNF | **0.009** | **0.009** | **0.009** | **0.009** |
| 4 | **Corel** | DNF | **0.009** | **0.009** | **0.009** | 0.01 |
| 5 | **Delicious** | DNF | **0.018** | **0.018** | **0.018** | 0.021 |
| 6 | **Emotions** | 0.205 | 0.2 | 0.197 | **0.196** | 0.198 |
| 7 | **Enron** | 0.049 | 0.047 | 0.046 | 0.046 | **0.045** |
| 8 | **Genbase** | **0.001** | **0.001** | 0.002 | 0.002 | **0.001** |
| 9 | **Mediamill** | DNF | **0.03** | **0.03** | **0.03** | 0.032 |
| 10 | **Medical** | **0.01** | 0.016 | 0.013 | 0.013 | 0.012 |
| 11 | **Scene** | 0.098 | 0.111 | 0.09 | 0.093 | **0.088** |
| 12 | **TMC 2007** | 0.033 | **0.001** | 0.015 | 0.016 | 0.012 |
| 13 | **Yeast** | 0.2 | 0.199 | **0.198** | **0.198** | 0.199 |

and equally ranked in terms of Hamming loss measure. RF-ROS-Tot-75 also performs well in terms of Hamming loss measure but is ranked last w.r.t the Example-based $F_1$ measure. Moreover, we observe that RF-ROS-Sub-LOG is ranked better than RP-RF and RAkEL.

## 6.3 Summary

We summarize the main findings of the extensive experimental work presented in this chapter by answering the experimental questions posed in Section 6.1.1.

1. **What is the recommended label subspace size to use with** RF-ROS **ensembles?** We recommend that RF-ROS should be instantiated with $v = \frac{\log|\mathsf{A}_t|}{|\mathsf{A}_t|}$ and should use subset averaging. It could be beneficial to use a slightly larger label subspace size on datasets with larger label spaces (i.e., $\sqrt{|\mathsf{A}_t|} \leq v \cdot |\mathsf{A}_t| \leq \frac{|\mathsf{A}_t|}{2}$).

2. **Does it make sense to change the aggregation function, i.e., can subspace averaging improve the predictive performance of** RF-ROS **models?** The answer to the first question already partially answered the second one. Subspace averaging is preferred because total averaging seems to degrade the predictive performance of the models and it, with larger label subspace sizes, converges to the performance of the original method (RF), i.e., even if we do not use the optimal value for the subspace size, the model performance is lower-bounded by RF.

3. **Considering predictive performance, how do** RF-ROS **ensembles compare to other competing methods?** RF-ROS ensembles perform well compared to the competition, which especially holds for RF-ROS-Sub-LOG variant. The average rank diagrams show some advantage, but without statistical significance.

# Chapter 7

# Learning and Evaluation of Rule Ensembles with Random Output Selections for Multi-Target Regression

Rule and decision tree models are one of the most interpretable model types. With the rise of black-box models (e.g., deep learning), data scientists are facing a hard-to-overcome obstacle when they are tasked with explaining the predictions made by such models. There have been many attempts to unbox such models by using various approaches to provide descriptions for models that are otherwise not interpretable. The ability to adequately explain the predictions helps with user acceptance (e.g., in medicine) as well as conforms to legislation regarding fairness, equal opportunities, privacy, etc. Learning rule-based predictive models is therefore a natural choice, when the end result should be an interpretable model.

We propose a new rule-learning method for the task of MTR, called Fitted Rule Ensembles with Random Output Selections (FIRE-ROS) (Breskvar, Kocev, & Džeroski, under review). The method learns an ensemble of predictive clustering trees (PCTs) with random output selections which is then decomposed to individual predictive clustering rules. An optimization approach is used to find the best-performing rule set by changing the weights of individual rules. We show that using partially-predicting rules (i.e., rules that give predictions for only a subset of targets) can improve the overall predictive performance. An extensive empirical evaluation is performed over a variety of benchmark datasets to systematically determine whether the ROS extension improves the predictive performance of the models induced with the FIRE method.

We summarize the main contributions of this work as follows:

- An extension of a rule-learning approach for addressing the MTR task that selects random subsets of target attributes in order to generate candidate rules more focused on specific target attributes.
- A novel approach where rule ensembles can contain rules that only give partial predictions.
- An extensive empirical evaluation of the proposed method on 17 benchmark datasets, including the usage of three different approaches to build the candidate rules. The evaluation provides a performance assessment of the original and proposed rule-based method as well as individual multi-target regression trees and predictive clustering

rules. The analysis also includes parameter setting recommendations for the proposed method.

- A comparison with state-of-the-art and other interpretable MTR methods.

This chapter is organized in the following way. First, we describe the required changes to the FIRE method in order to integrate ROS. Next, we describe the experimental design, including experimental questions, evaluation measures and the actual experimental setup. This is followed by the description of the results. We conclude this chapter with the summary of the results.

## 7.1 Learning Rule Ensembles for Multi-Target Regression with FIRE-ROS

This section describes the integration of ROS tree ensembles into the Fitted Rule Ensemble (FIRE) for MTR rule learning system. FIRE method was proposed by Aho et al. (2012). The integration consists of two main steps. First, we replace the tree ensemble building method with the one that uses the ROS extension. Then, we modify the covariance calculations within the optimization function to take into account the fact that not all rules give predictions for all target attributes. We call such rules *partially-predicting* rules, as opposed to the original *fully-predicting* (predicting all target attributes) rules. Three types of pairwise covariance calculations are used: (i) between two rules and (ii) between two linear terms and (iii) between one rule and one linear term.

To integrate partially predicting rules, we need to modify the covariance calculation between two rules. As in the original FIRE method, we calculate the average covariance between two rules by averaging covariances of individual targets. While iterating over the target attributes, we check if both rules give a prediction for a specific target attribute. If they do not, we skip that target attribute. When calculating the average covariance, we average only over the target attributes that were predicted by both rules, i.e., we use the number of joint target attributes $|R_{r_1} \cap R_{r_2}|$, where $R_x$ represents the set of target attributes considered by rule $x$. If the two rules have a completely disjoint set of target attributes, the returned covariance is evaluated to zero.

**Size of candidate decision rules.** With tree ensembles, one should not restrict growing of individual base learners in order to obtain better predictive performance. Here, however, we focus also on the interpretability. If we consider a decision tree of depth 10, the rules that we would get from that tree would have 10 conditions. Although a manageable number of such decision rules is considerably more transparent than a tree ensemble, it borders on interpretability. Rules with many conditions are not easily comprehended by humans. A considerable cognitive effort is required to visualize how that many conditions actually partition the input space. Moreover, having many such rules represents an additional problem, since one also has to visualize the interplay of different rules that are triggered. We accept that there are use cases, where even more than 10 attributes would be perfectly acceptable for the domain experts but for the situations, where this is not desired, the size of the PCTs in the ensemble should be restricted by limiting their depth. This makes rules shorter and thus easier to process by a human. Individual PCTs in the ensemble are allowed to grow only to a certain depth. The depth is not fixed across all PCTs but is determined in a randomized fashion. The number of leaf nodes $t$ is determined for each PCT in the ensemble with a random variable: $t = 2 + \lfloor \gamma \rfloor$, where $\gamma$ is drawn from an

exponential distribution with probability

$$\Pr(\gamma) = \frac{e^{-\gamma/(\overline{L}-2)}}{\overline{L}-2},$$

where $\overline{L}$ is the average number of leaf nodes in all PCTs. The depth of a PCT can now be computed as $d = \lceil \log_2(\mathrm{t}) \rceil$, assuming the root node is at depth 0. $\overline{L}$ is the parameter of the algorithm. This tree pruning approach has been proposed by J. H. Friedman and Popescu (2008) and is already used by the original FIRE algorithm.

**Generating candidate rules.** As we have already previously stated, all PCTs in the ensemble are converted into rules. The individual rules must give predictions but these predictions need not be for all target attributes, as is the case with the method FIRE. We have learned PCTs that are able to give predictions for only a subset of all target attributes. Converting such PCTs to rules results in more target-oriented decision rules that are easier to explain and understand by a domain expert. Before we make the final decision on what kind of decision rules we would like to declare as candidates, we must first think about the way that we will optimize the weights of individual rules. The FIRE method always selects from rules that give predictions for all targets because it has nothing else to select from. Our initial experiments show that if we populate the pool of candidate rules with only partially-predicting rules, we end up with much more rules because we have to satisfy the requirement for a well performing model in terms of predictive performance. However, giving the optimization procedure the *ability to select* partially predicting rules without forcing it into doing so, is a good idea. Ideally, the optimization would be guided into producing an optimized rule set that contains fully-predicting decision rules (that predict all target attributes) which globally cover the target space and then possibly also partially-predicting rules (that predict only a subset of target attributes) which make local adaptations for specific target attributes – depending on the problem. This brings us to the final decision regarding which decision rules should be used in the candidate pool: we use fully-predicting as well as partially-predicting rules to populate the candidate pool of decision rules.

Before we continue with the description of the experimental design, we present an example of a FIRE-ROS rule model. Figure 7.1 depicts a rule model built on the same benchmark dataset as the rule model from Figure 3.6. This time we can see a difference in the learned rules. In addition to the fully-predicting rules, partially-predicting rules have also been selected into the final set. The letters NP denote the absence of prediction (NP = Not Predicted). As we have already explained, FIRE and FIRE-ROS models calculate the predictions by adding the predictions of individual rules and linear terms to the average rule. Therefore, in the case of FIRE-ROS models, we can easily substitute NP with the number zero.

## 7.2 Experimental Design

To evaluate the performance of the FIRE-ROS ensembles for MTR, we perform extensive experiments on various benchmark datasets. This section presents: (i) the experimental questions addressed, (ii) the evaluation measures used, and (iii) the experimental setup (including the parameter instantiations for the methods used in the experiments).

### 7.2.1 Experimental questions

In our experiments, we construct rule-based models by using the PCT ensembles for MTR to populate the initial pool of candidate rules. The goal of this work is to establish

$$(0.95, 0.73, 0.42, \ldots, 0.75, 0.74)$$

$$\text{rules} \begin{cases} +0.34 \cdot [\text{IF } O_2 > 3.52 \text{ AND } Cl \leq 0.38 \text{ AND } CO_2 > 0.27 \text{ THEN } (0.61, 0.41, \ldots, 1.13)] \\ -0.85 \cdot [\text{IF } K_2Cr_2O_7 > 0.43 \text{ AND } O_2Sat > 3.60 \text{ THEN } (0.6, \text{NP}, \ldots, 0.64)] \\ \ldots \\ +0.77 \cdot [\text{IF } K_2Cr_2O_7 \leq 0.43 \text{ AND } conduct \leq 2.46 \text{ THEN } (2.24, 3.3, \ldots, \text{NP}, \text{NP}, \ldots, 2.69)] \end{cases}$$

$$\text{linear terms} \begin{cases} +0.15 \cdot (0, 0, x_8, 0, \ldots, 0) \\ \ldots \\ +0.5 \cdot (0, \ldots, 0, x_1) \end{cases}$$

Figure 7.1: A part of the example rule ensemble model for predicting the abundance of animal and plant species in Slovenian rivers. The prediction vectors have been shortened in order to fit the model on the page. Due to space limitations, the majority of decision rules and linear terms has also been omitted. This rule ensemble contains rules learned with ROS ensemble extension. Moreover, the presented rule ensemble also contains partially-predicting rules.

whether output space specializations affect the resulting predictive models and how. We investigate the resulting models in terms of their **predictive performance**. We focus on (i) predictive performance of the proposed models w.r.t. that of models produced by the original approach (FIRE) and (ii) comparison to other MTR methods. In addition to predictive performance, we also take interest in the **interpretability** of the resulting models. Rule-based models have an inherent trait of being readable by humans, which results in better understandability of how the predictions are made. The experiments and their evaluation have been designed with the following research questions in mind:

1. What is the best value for the portion of target space to be used with the proposed ensembles? Are there any differences among the evaluated ensemble methods?

2. Are there differences in the ROS output space size with respect to the ROS tree ensembles?

3. Do partially predicting rules improve the predictive performance?

4. How do FIRE-ROS ensembles compare to the original FIRE ensembles in terms of predictive performance?

5. How do FIRE-ROS models compare to state-of-the-art MTR methods?

### 7.2.2 Evaluation measures

In order to understand the effects that FIRE-ROS has on the rule learning process, we first need to evaluate the models induced by the FIRE approach. In machine learning, empirical evaluation in terms of desired evaluation measures is most commonly used to achieve this goal. Below, we describe the measures used for assessing predictive power and interpretability.

**Predictive performance.** The predictive performance of a MTR model is assessed by using the average relative root mean squared error (aRRMSE), which averages the relative root mean squared errors (RRMSE) for the individual target variables. RRMSE is a relative measure calculated against the baseline model that predicts the arithmetic mean of all values of a given target in the learning set. Specifically, the value $\overline{y}_i$ in Equation 7.1 is the prediction of the baseline model for the $i^{th}$ target variable, while the value $\hat{y}_i^{(e)}$ represents the predicted value for the $i^{th}$ target variable of the example $e$.

$$\text{aRRMSE} = \frac{1}{t}\sum_{i=1}^{t}\text{RRMSE}_i = \frac{1}{t}\sum_{i=1}^{t}\sqrt{\frac{\sum_{e=1}^{N_{test}}(y_i^{(e)} - \hat{y}_i^{(e)})^2}{\sum_{e=1}^{N_{test}}(y_i^{(e)} - \overline{y}_i)^2}} \tag{7.1}$$

**Overfitting.** We also monitor how much our models overfit to the training data by calculating their relative decrease of performance on the testing data with respect to that on the training data. Smaller values mean less overfitting. We calculate the overfitting score with Equation 7.2.

$$\text{OS} = \frac{\text{aRRMSE}_{test} - \text{aRRMSE}_{train}}{\text{aRRMSE}_{train}} \tag{7.2}$$

**Interpretability.** The interpretability is measured in terms of sizes of the induced models. Specifically, we measure the number of rules and number of linear terms for rule-based models. For tree-based models, interpretability is measured by the total number of nodes in the tree model (intermediate and leaf nodes). In both cases smaller values are better. For other model types, interpretability is not evaluated.

**Data.** To evaluate the proposed method, we use 17 benchmark datasets that contain multiple continuous target attributes and are mainly from the domain of ecological modeling. The datasets have already been described in Chapter 5, Section 5.1.3. In order to have as general evaluation as possible, we use datasets of different sizes in terms of the number of instances, number of predictive and number of target attributes.

### 7.2.3 Experimental setup

We designed the experimental setup according to the experimental questions posed in Section 7.2.1. First, we describe all parameter settings of the FIRE-ROS method and then outline the procedures for statistical analysis of the results.

We **estimate the predictive performance** of the considered methods by using 10-fold cross-validation. All methods use the same folds. For **statistical evaluation** of the obtained results, we follow the recommendations from (J. Demšar, 2006). The Friedman test (M. Friedman, 1940), with the correction by (Iman & Davenport, 1980), is used to determine statistical significance. In order to detect statistically significant differences, we calculate the critical distances (CD) by applying the (Nemenyi, 1963) or (Dunn, 1961) post-hoc statistical tests. Both post-hoc tests compute critical distance between the ranks of considered algorithms. The difference is that Nemenyi post-hoc test compares the relative performance of all considered methods (all vs. all), whereas Bonferroni-Dunn post-hoc test compares the performance of a single method to other methods (one vs. all). The results of these tests are presented with average rank diagrams (J. Demšar, 2006), where methods connected with a line have results that are not statistically significantly different. All statistical tests were conducted at the significance level $\alpha = 0.05$. The lower the average rank, the better performance it signifies. Tests have been calculated for two variants of

the results: per dataset (using aRRMSE value for each dataset) and per target (using the RRMSE values for all targets of all datasets). In both cases, lower values are better. We used the Bonferroni-Dunn (CD is shown as a dotted blue line) post-hoc test to present results in Section 7.3.3 and Nemenyi post-hoc test otherwise (CD is shown as a solid red line).

**Methods.** We experiment with several methods that solve the MTR task. Here, we describe the parameters used to run the different methods. For FIRE-ROS, we consider three types of ensembles: bagging and random forests of PCTs and ensembles of Extra-PCTs. All ensembles consist of 100 base learners (PCTs or Extra-PCTs). Random forests and Extra-PCT ensembles use sampling of the descriptive space. The former use $\sqrt{|\mathsf{A}_d|}$ and the latter $|\mathsf{A}_d|$ (i.e., the $k$ parameter) attributes sampled from the set of all descriptive attributes ($\mathsf{A}_d$) when searching for the best split within a given node. These values are recommended by the authors of the two ensemble methods. Random forests and bagging ensembles use bootstrap replication whereas ensembles of Extra-PCTs do not. The original FIRE method was parametrized with the same parameters but was only used with the original random forest ensembles for MTR. FIRE-ROS experiments consider four values for the output space sizes: $\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \mathit{Random}$. With $v = \mathit{Random}$, the ROS ensembles use a different (randomly selected) subspace size for every PCT in the ensemble. All ensembles are allowed to grow to the average depth of 3. We consider two types of candidate rules sets: fully-predictive (F) and mixed (M). The former rule set type consists only of rules that always predict all target attributes. The latter rule set type also contains partially-predicting rules, which give predictions for only a subset of target attributes. Both methods use linear terms. Predictive clustering rules (PCRs) were executed with the default parameter values. We trained unordered rules using the multiplicative variant of the search heuristic. Covering weight was set to 0.1 with the covering weight threshold of 0.1. Maximal number of rules was set to 1000. Target attributes weight was set to 1.0. When using multi-target regression trees (PCTs), F-test pruning was applied. The threshold was computed using 3-fold cross-validation. Random projections with Random forests (RP-RF)[1] use $m = log(|\mathsf{A}_t|)$ components in the projected output space where $\mathsf{A}_t$ is the set of target attributes. In addition, Rademacher random projections were used for output space transformations. We used $k = |\mathsf{A}_d|$ randomly chosen input features to calculate splits. Minimal number of allowed instances in a leaf node was set to 1. Random Linear Combinations (RLC) algorithm[2] was parametrized to use gradient boosting with 4-terminal node regression tree as the base regressor with learning rate of 0.1 and 100 boosting iterations. Number of targets that participate in the random linear combinations was set to $k = 2$. Ensembles of Extra-PCTs with ROS (ET-ROS) use 100 trees with $k = |\mathsf{A}_d|$ randomly chosen input features to calculate splits. ROS was initialized with subspace size of $\frac{3}{4}$ and subspace averaging. All competing method were parametrized according to the recommendations of their respective authors.

## 7.3 Results and Discussion

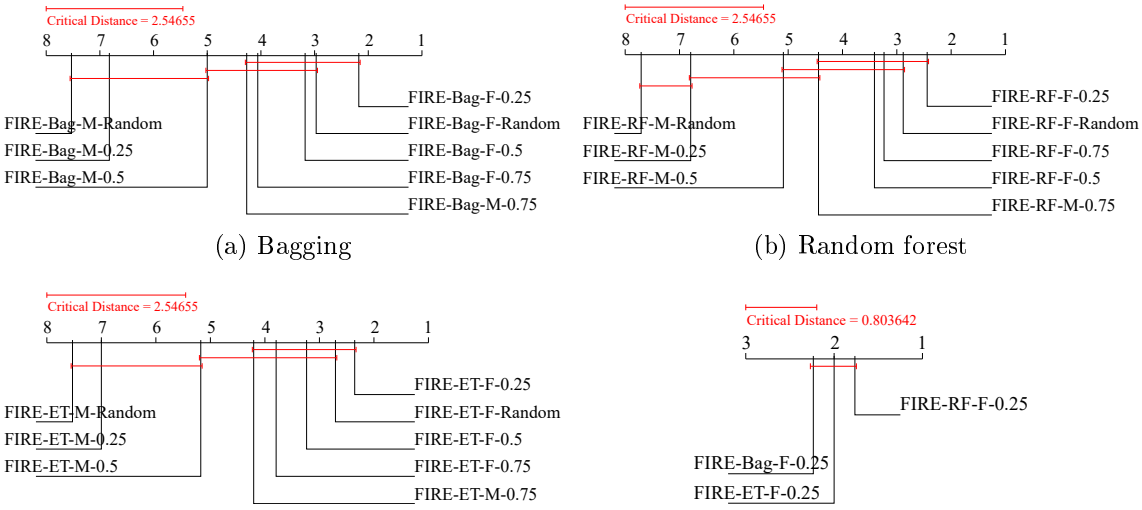
Here, we present results of our extensive experimental evaluation of the proposed and competing methods. Results are presented in terms of predictive performance (aRRMSE, overfitting score) and interpretability (model sizes). First, we present results regarding FIRE-ROS variants with different output space sizes and ensemble methods for generating candidate rules (Sections 7.3.1, 7.3.2). Next, we compare models induced with FIRE-ROS

[1] Random projections source code is available at https://github.com/arjoly/random-output-trees.
[2] Random linear combinations source code is available at http://mulan.sourceforge.net.

to the models induced by state-of-the-art MTR methods (Section 7.3.3). Then, we examine the overfitting score of FIRE-ROS (Section 7.3.4). Furthermore, we address the interpretability of learned models (Section 7.3.5). Finally, we summarize the results (Section 7.4). ROS models induced from only fully-predictive candidate rules are denoted with the letter *F*. Otherwise, ROS models are induced from mixed candidate rules and are denoted with the letter *M*.

### 7.3.1 FIRE-ROS parameter selection

In this section, we analyze the predictive performance of FIRE-ROS variants. Figure 7.2 depicts average predictive performance with the average rank diagrams for FIRE-ROS models. We used three different ensemble methods to generate the candidate rules and we analyze each of them separately. We have also experimented with two types of candidate rule sets: only fully-predictive (F) rules and mixed-predictivity (M) rules.




(a) Bagging

(b) Random forest

(c) Extra trees

(d) Best FIRE-ROS variants

Figure 7.2: Average rank diagrams comparing predictive performance of FIRE-ROS models in terms of aRRMSE. Lower ranks are better. *Bag*, *RF* and *ET* denote models where the candidate rules were generated using the bagging, random forests and extra tree ensembles, respectively. *M* and *F* stand for mixed and fully-predictive candidate rules, respectively. Labels 0.25, 0.5, 0.75 and Random denote various output space sizes. The best performance of the proposed method is, on average, achieved with fully-predictive rules originating from ROS ensembles that use $\frac{1}{4}$ of target attributes (i.e., label 0.25, denoting 25% of all target attributes).

We can immediately see that the performance of models induced from mixed rules is worse than that of fully-predictive rules (Figure 7.2). Models induced from mixed rules (M) exhibit the same pattern regarding the improvement/degradation of predictive performance with all three ensemble methods. Larger ROS subspace sizes perform better. Varying the ROS subspace size (i.e., Random) performs worst. Conversely, fully-predictive variants (F) exhibit a slightly different behavior, depending on the ensemble method used. Regardless of that, all three ensemble methods perform best when subspace size is set to $\frac{1}{4}$. We therefore conclude that the recommended parameter setting for FIRE-ROS ensembles is to use fully-predictive candidate rules generated from ROS tree ensemble with output space size of $\frac{1}{4}$.

Table 7.1: The best performing FIRE-ROS variants. F and M denote fully-predictive and mixed candidate rule sets. The numbers in brackets represent the output space size.

| Ensemble method | F ($\frac{1}{4}$) | F ($\frac{1}{2}$) | F ($\frac{3}{4}$) | F (Random) | M ($\frac{1}{4}$) | M ($\frac{1}{2}$) | M ($\frac{3}{4}$) | M (Random) | Total |
|---|---|---|---|---|---|---|---|---|---|
| Bagging | 10 | 0 | 1 | 2 | 0 | 3 | 1 | 0 | 17 |
| Random forest | 8 | 2 | 2 | 1 | 0 | 3 | 1 | 0 | 17 |
| Extra trees | 9 | 1 | 2 | 1 | 0 | 2 | 2 | 0 | 17 |
| Total | 27 | 3 | 5 | 4 | 0 | 8 | 4 | 0 | 51 |

Table 7.1 shows the best performing FIRE-ROS variants. The majority of the winning parameter settings are indeed fully-predictive rules (F) with output space size of 0.25. However, some datasets also benefit from using mixed candidate rules. It is interesting that in those cases, all three ensemble methods perform best with the same output space size of 0.5. Only one model parameter setting deviates from this: the *Soil resilience* dataset induced from candidate rules generated with the ensemble of extra trees, where the best output space size is 0.75. This does not come as a surprise. Ensembles of Extra-PCTs are heavily randomized and pruned (for the purpose of learning FIRE-ROS models). It is therefore likely the case that a too small output space size leads to the selection of suboptimal tests resulting in lower predictive performance. In fact, the exact same output space size ($\frac{3}{4}$) was also determined as optimal for ET-ROS decision tree ensembles in Breskvar et al. (2018a).

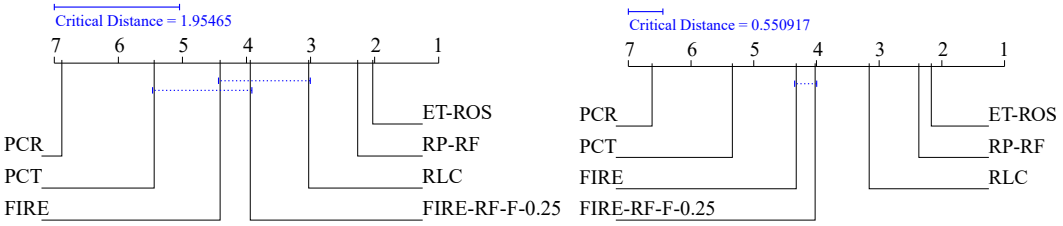
### 7.3.2 Rule set size restrictions for FIRE-ROS

In order to make FIRE-ROS models even more interpretable, we have carried out additional experiments, where we restricted the number of rules. We induced rule sets with maximal number of 30, 50 and 100 rules. The graphics are not included but the results show that generating more rules yields models with better predictive power. This is in line with the findings in (Aho et al., 2012). In addition to that, fully-predictive rules always outperform mixed candidate rules. We also investigated whether the recommended ROS subspace size remains the same as with the full rule ensembles. The best performing subspace size for random forest and Extra-PCTs is $\frac{1}{4}$ for all three reduced rule set sizes, which is the same as with the full rule ensembles. A small deviation was observed with bagging, where *Random* is the best choice, when restricting to only 30 or 50 rules. Per target analysis shows statistically significant differences.

### 7.3.3 Comparison to existing methods

In order to put FIRE-ROS in the broader context of MTR methods, we compare the best variant (we use the recommended parameter settings) to other MTR methods. Figure 7.3 shows the obtained results by means of average rank diagrams. All parameters are explained in detail in Section 7.2.3.

FIRE-ROS outperforms PCRs with statistical significance and PCTs and FIRE without statistical significance. All state-of-the-art MTR methods (ET-ROS, RP-RF, RLC) outperform all interpretable models. Differences are statistically significant only with ET-ROS and RP-RF. In per target analysis, FIRE-ROS additionally outperforms PCTs with statistical significance. All three state-of-the-art MTR methods outperform all interpretable models with statistical significance.

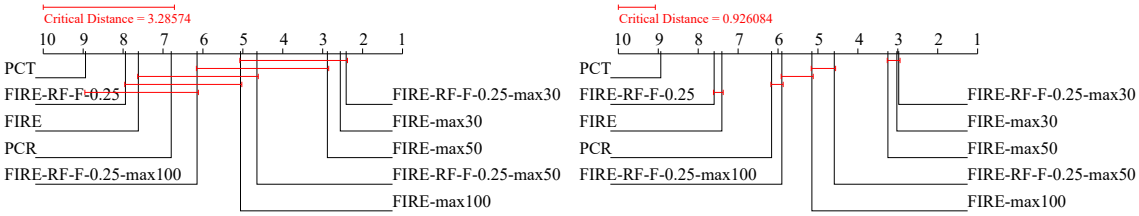



(a) Per dataset (b) Per target

Figure 7.3: Average rank diagrams comparing aRRMSE performance of FIRE-ROS with aRRMSE of the competing methods. Lower ranks are better. The proposed approach performs best in the group of interpretable methods. Methods that produce models that are not interpretable are considered state-of-the-art in MTR and, to no surprise, outperform all interpretable models.

### 7.3.4 Overfitting

Figure 7.4 depicts the overfitting comparison of interpretable models. Smaller models overfit less, which is an expected result. With statistical significance, PCT models overfit the most. If we ignore the size-restricted models, PCR models are least overfitted. Without statistical significance, the proposed models overfit marginally more than models generated with FIRE.



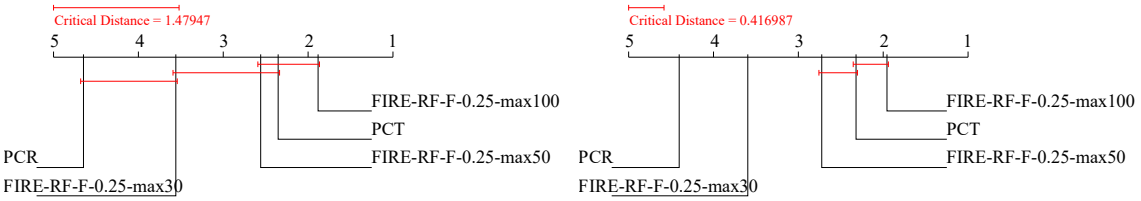

(a) Per dataset (b) Per target

Figure 7.4: Average rank diagrams comparing overfitting score of FIRE-ROS models compared to FIRE, PCRs and PCTs. Lower ranks are better.

### 7.3.5 Comparison of interpretabiliy

We compare model sizes of interpretable models generated with the following methods: PCRs, PCTs, FIRE and FIRE-ROS (with the recommended settings from Section 7.3.1). The graphics are not included but the results show that the smallest models are produced by PCRs and PCTs. FIRE and FIRE-ROS generate statistically significantly larger models. We also compare predictive power of smaller FIRE-ROS models (30, 50 and 100 rules) to that of PCTs and PCRs.

Figure 7.5 shows that FIRE-ROS achieves the best predictive performance while maintaining a small number of rules in the ensemble. FIRE-ROS with 50 and 100 rules performs better than PCRs with statistical significance. FIRE-ROS with 100 rules outperforms PCTs although not statistically significantly differently, while FIRE-ROS models with 50/100 rules perform worse/better than PCTs without statistical significance.




(a) Per dataset (b) Per target

Figure 7.5: Average rank diagrams comparing predictive performance of smaller FIRE-ROS models to PCRs and PCTs in terms of aRRMSE. Lower ranks are better.

## 7.4 Summary

We summarize the main findings of the extensive experimental work presented in this chapter by answering the experimental questions posed in Section 7.2.1.

1. **What is the best value for the portion of target space to be used with the proposed ensembles? Are there any differences among the evaluated ensemble methods?** The recommended subspace size is $\frac{1}{4}$. All three ensemble methods perform best with this setting.

2. **Are there differences in the ROS output space size with respect to the** ROS **tree ensembles?** Yes. ROS tree ensembles perform best with Extra-PCTs with subspace size of $\frac{3}{4}$. FIRE-ROS performs best with the same ensemble method but with a much smaller subset size of $\frac{1}{4}$. On datasets, where mixed candidate rules perform best, larger subspace sizes ($\frac{1}{2}$, $\frac{3}{4}$) should be considered.

3. **Do partially predicting rules improve the predictive performance?** On average, the best performing rule ensembles use only fully-predicting rules. However, we have shown that rules with partial predictions are beneficial for some problems and lead to better predictive performance.

4. **How do** FIRE-ROS **ensembles compare to the original** FIRE **ensembles in terms of predictive performance?** In terms of predictive performance, the proposed ensembles outperform the original FIRE ensembles, but the differences are not statistically significant.

5. **How do** FIRE-ROS **models compare to state-of-the-art MTR methods?** All state-of-the-art MTR methods outperform all interpretable models with statistically significant differences in performance. FIRE-ROS outperforms PCRs, PCTs and the original FIRE method. The per target analysis shows statistically significant improvement of FIRE-ROS over PCRs and PCTs.

# Chapter 8

# Conclusions

We propose a novel methodology for learning tree and rule ensembles for multi-target prediction (MTP). The proposed methods are an extension of the predictive clustering framework. The presented methodology builds ensembles by learning the individual base predictive models that focus on subsets of target variables.

The remainder of this chapter contains a summary of the scientific contributions of this dissertation, a discussion with respect to the goals and hypotheses listed in Chapter 1 and an outline of several directions for future work.

## 8.1 Contributions to Science

The work presented in this dissertation contributes to the field of supervised learning and in particular MTP. We introduce Random Output Selections (ROS), a novel ensemble learning methodology that learns individual ensemble members by focusing on different subsets of all available target variables. The subsets are sampled at random. The size of an individual randomly sampled subset is a user-specified parameter. The proposed methodology extends the ensemble methods implemented within the predictive clustering framework, which use predictive clustering trees (PCTs) and predictive clustering rules (PCRs) as base learners.

The algorithms for building ensembles required modifications. The ROS extension generates subsets of target attributes prior to learning the base predictive models. Each base predictive model is then learned by focusing on one of the generated subspaces. Consequently, the learning of base predictive models also required changes. In particular, the algorithm for the top-down induction of PCTs was modified in the part, where the algorithm searches for the best test to split the input data on. The introduced changes guide the search by focusing only on the restricted target space. The learning on subsets of target variables raised the question whether predictions of the base predictive models should be made only for the variables in the learning subset or for all target variables.

The empirical evaluation showed that the models obtained with the proposed approach have better predictive performance as compared to MTP ensemble models without using ROS. Moreover, tree ensembles with ROS can outperform the current state-of-the-art in multi-target regression tasks and perform comparably in multi-label classification tasks. Rule ensembles with ROS, on average, also outperform two interpretable model types: predictive clustering rules and predictive clustering trees. The ROS methodology has been used for solving multi-target regression (MTR) tasks (Breskvar et al., 2018a) as well as multi-label classification (MLC) tasks (Breskvar, Kocev, & Džeroski, 2017). In addition to the benchmark datasets, we also applied ROS to a dataset from a case study. In particular, we use the Mars Express Power Challenge dataset (Breskvar, Kocev, Levatić, et al., 2017)

and build models that predict 33 continuous target variables.

The ROS methodology was also applied in the context of learning rule ensembles. The existing method, called Fitted Rule Ensembles (FIRE) for MTR, internally uses a random forest ensemble of PCTs to generate candidate PCRs. An optimization procedure then selects a small subset of rules from this large pool of candidate rules. The integration of ROS with the FIRE algorithm was done in several steps. The first step was to learn ROS ensembles of PCTs instead of original ensembles. This step does not change the FIRE algorithm in any way as it just introduces a different set of candidate rules. The next step is optimization. Here, changes were required because an individual rule can predict either a subset or all of the target variables. An individual rule, obtained from a ROS ensemble, can be fully-predicting (predicting all target variables) or partially-predicting (predicting a subset of target variables). This is important, because the optimization procedure internally calculates covariances between the candidate rules. The candidate pool of rules contains both types and we let the optimization procedure decide which combination works best. The empirical evaluation showed that the models obtained with the proposed approach have better predictive performance as compared to the ensemble models without using ROS. The approach was evaluated on solving MTR tasks (Breskvar, Kocev, & Džeroski, 2018b, under review).

In sum, the major scientific contributions of the dissertation are as follows:

- A novel methodology for learning tree ensembles with Random Output Selections (ROS) in the context of the predictive clustering framework.
- A novel methodology for learning rule ensembles for MTR with ROS in the context of the predictive clustering framework.
- Extensive empirical evaluation of the developed methods across benchmark problems from various domains, including a comparison to state-of-the-art supervised approaches for multi-target prediction, which shows better or comparable performance of the proposed approaches.

## 8.2 Discussion

### Goals

The goals of this dissertation were divided into two categories: the design and implementation of new algorithms and evaluation of the implemented algorithms. However, the overall *main* goal of this dissertation was to contribute to the field of machine learning. In particular, bring novelty to the area of supervised learning for multi-target prediction. With the introduction and publication of the ROS tree ensemble extension, this goal was achieved.

**Design and implementation goals.** In more detail, we estimate that the group of design and implementation goals has been achieved in its entirety. The algorithm for TDI of PCTs was adequately extended and can now consider predefined target variables in the heuristic calculations. No changes were required in the prediction mechanism of a single PCT. The reason for this is that ROS works in context of ensemble models, i.e., ROS is an ensemble extension method. This means that ROS ensembles handle the aggregation of predictions of base predictive models and internally hold the information to make decisions on which attributes should or should not contribute to the overall prediction of the ensemble. The ensembles contain information about ROS subspaces of all base predictive models and can therefore make the appropriate calculations when votes need to

be omitted. In other words, PCTs continue to predict all target variables. The predictions for individual target variables are considered or discarded within the ensemble prediction aggregation function and depend on the user-defined input parameters. The last design goal pertains to the extension of the FIRE method, which is now able to use ROS ensembles to generate the candidate rules. In addition to that, the resulting rule sets can now contain fully-predicting and partially-predicting PCRs.

The ROS extension is implemented in the publicly available CLUS software. The three considered ensemble methods can now learn individual base predictive models by using only subsets of target variables. With tree ensembles, the predictions can be made by using total averaging or subspace averaging. The former prediction aggregation function calculates predictions based on all target variables whereas the latter only considers predefined subsets of them. With rule ensembles, partially-predicting rules are added to the pool of candidate rules. At that point, the weights of all rules are optimized and if the optimization procedure selects a partially-predicting rule into the final rule set, it will work without any modifications. The reason for this is that the model produced by the FIRE method is a simple function that makes per-component sums of numerical values in predicted vectors. If a partially predicting rule is fired, the prediction for each omitted target is the number zero, which (in the case of FIRE models) is the same as not predicting anything but simplifies the implementation.

**Evaluation goals.** The evaluation goals relate to the evaluation of the proposed ensemble models for MTR and MLC. Tree ensembles for MTR were evaluated by using bagging and random forests of PCTs and ensembles of extremely randomized PCTs. We observed degradation of model performance when using random forest of PCTs with ROS. The other two ensemble methods did benefit from using ROS and the resulting models outperformed their original counterparts. Furthermore, ensembles of extremely randomized PCTs with ROS and subspace averaging outperformed the competing methods that use transformations of the output space. Tree ensembles for MLC were evaluated by using random forests of PCTs with ROS and compared to methods that use transformations of the output space. Results show that the proposed models perform comparably to models produced by other methods.

The performance of rule ensembles for MTR was evaluated with all three considered ensemble methods. The original FIRE method uses random forest ensembles of PCTs to generate candidate rules. Results show that rule ensembles with ROS also perform best when using random forest ensembles of PCTs. This is in contrast to tree ensembles, where ensembles of extremely randomized PCTs perform best. FIRE models were also built by using candidate rules from bagging of PCTs and ensembles of extremely randomized PCTs. The ensembles without ROS were built in order to also compare the effect of ROS on bagging and extra trees ensembles.

### Hypotheses

We posed several hypotheses, which we will now address in light of the dissertation results. The first hypothesis states that the existing ensemble MTP methods within predictive clustering can be adapted to sample the output space, which has been confirmed and demonstrated in detail in this dissertation.

The second hypothesis relates to the predictive performance of ROS-extended methods. It was confirmed partially. More specifically, our results show that tree ensembles with ROS for MTR can outperform their original counterparts with the exception of random forest ensembles, where the predictive performance degrades. Moreover, tree ensembles with ROS outperform the state-of-the-art models for MTR in terms of predictive performance. The

predictive performance improvements of tree ensembles of extremely randomized PCTs with ROS for MLC are not entirely clear.

Rule ensembles (FIRE method) for MTR were initially implemented to use candidate rules obtained from an ensemble generated with the random forest of PCTs. We show that using candidate rules obtained from ROS extended random forest ensembles of PCTs outperforms the original approach without statistical significance. Other ensemble methods (bagging of PCTs and ensembles of extremely randomized PCTs) with or without using the ROS extension did not show promise in the rule learning setting.

The last hypothesis states that the optimal subset size of the output space depends on the domain. Results of using ROS were often statistically insignificant. However, by inspecting the performance on the individual datasets, substantial differences could be observed depending on the dataset. The optimal ROS output space size also differs across the three ensemble methods. The results suggest using $v = \frac{1}{2}$ for Bag-ROS and $v = \frac{3}{4}$ for RF-ROS and ET-ROS. Regarding the rule ensembles, all three ensemble methods perform best with $v = \frac{1}{2}$. This leads us to believe that the appropriate ROS subspace size and prediction aggregation function are dataset specific and would ideally be tunable parameters. This hypothesis is thus confirmed.

To summarize, ROS ensembles can have a positive effect on the model-learning process. In the case of tree ensembles, the ROS ensembles offer a state-of-the-art predictive performance for MTR and a performance comparable to that of other output transformation methods for MLC. Rule ensembles with ROS for MTR perform slightly better than state-of-the-art rule learning methods for MTR.

## 8.3 Further Work

The most immediate extension of the work presented in this dissertation is related to rule ensembles. In particular, Fitted rule ensembles (FIRE) were only extended to MTR. The extension towards solving MLC tasks is possible by changing the loss function used by the optimization function. RuleFit ensembles (J. H. Friedman & Popescu, 2008) use the squared-error ramp loss for solving classification problems. This loss function could be extended to MLC in a similar manner as was done in the case of MTR tasks. When learning rule ensembles for MTR, the data is normalized prior to the optimization. This step is unnecessary in case of MLC, because all target attributes (and therefore predictions of individual rules) are on the same scale. The result of this would be a novel method for learning rule ensembles for MLC within the predictive clustering paradigm. Continuing along this line, learning rule ensembles for MLC with ROS is immediately possible.

An interesting research direction would be an extension towards semi-supervised learning. Learning ensembles of semi-supervised PCTs is already possible within the predictive clustering framework (Levatić, Ceci, Kocev, & Džeroski, 2014). An extension towards semi-supervised PCTs with ROS is possible without modifications in any of the existing algorithms. The amount of missing values in the target space has a significant influence on the predictive performance of the resulting models. It would therefore make sense to adapt the function for generating ROS subspaces so that it takes into account weakly-populated target attributes. Obtaining semi-supervised rule ensembles for MTR is therefore also possible, because the FIRE algorithm internally uses ensembles of PCTs. After extending FIRE towards MLC, semi-supervised rule ensembles for MLC are also immediately possible.

As we have already hinted in the previous paragraph, the ROS subspaces could be generated in a smarter way. If we exclude the obvious choice of manually selecting the subspaces by either consulting with domain expert or leveraging other prior knowledge

regarding the problem at hand, there are several other ways that the ROS subspaces could be generated. The current approach generates target subspaces at random, which is not necessarily the best approach. The relations between target variables could be exploited in order to generate a smaller set of more sensible subspaces. One approach would be to transpose the dataset, cluster the target attributes and then learn ROS ensembles by using the obtained clusters of target attributes.

All proposed approaches could be extended towards hierarchical multi-label classification (HMLC) tasks. In HMLC tasks, examples can have more than one class (same as with MLC tasks), with the additional complexity of each class being a part of a hierarchical structure. When an example belongs to a target class, it automatically also belongs to all of its parent classes (superclasses). An HMLC hierarchy can be either tree-shaped or in the form of a directed acyclic graph. An example of a HMLC task with a tree-shaped hierarchy is annotation of X-ray images, where each target attribute represents a taxonomy of body parts that can be identified in an X-ray image. An example of a HMLC task with DAG hierarchy is gene function prediction, where each gene can have multiple functions and each function can have multiple parent functions/processes. This extension introduces an interesting problem of selecting the ROS subspaces. One can sample the actual hierarchy on a per-target basis, the bottom-most nodes in the tree/graph, or both. HMLC tasks also introduce an additional dimension of complexity, because the datasets are usually very sparse.

An extension towards solving hierarchical multi-target regression (HMTR) tasks is also possible. Here, hierarchies are introduced in the form of arbitrary aggregations of target attributes according to a predefined hierarchical structure imposed on them. As PCTs for HMTR already exist (Mileski, Džeroski, & Kocev, 2017), ensembles thereof are possible and consequently the ROS extension can be used. With or without ROS, HMTR can be implemented without major modifications to the existing algorithms for both, tree and rule ensembles.

For the tasks of MTR and HMLC, ensembles of PCTs are already being used (Petković, Džeroski, & Kocev, 2017, 2018) to obtain rankings of input features. ROS could improve these approaches, by considering subsets of the set of target attributes in the process of producing ranks. Another approach would be to use rule ensembles to obtain importances of input features, as described by J. H. Friedman and Popescu (2008). This approach would first have to be evaluated in the context of MTP tasks in general. If such rankings would show promise, an extension with ROS would be immediately possible.

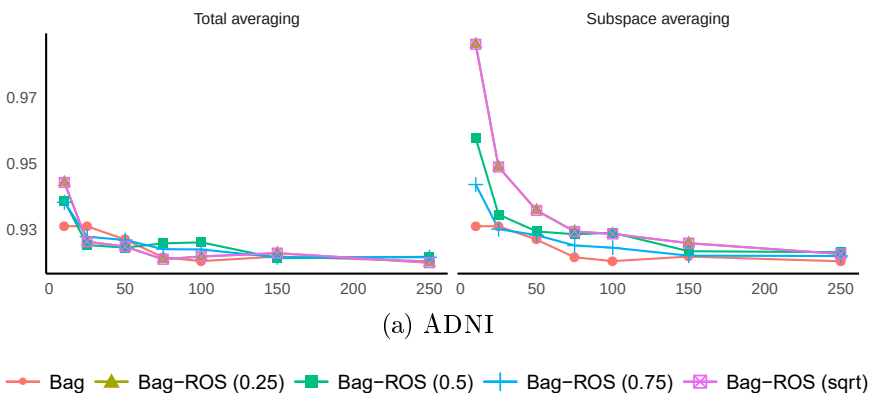
# Appendix A

# Additional Details on the Evaluation of Tree Ensembles with Random Output Selections for Multi-Target Regression

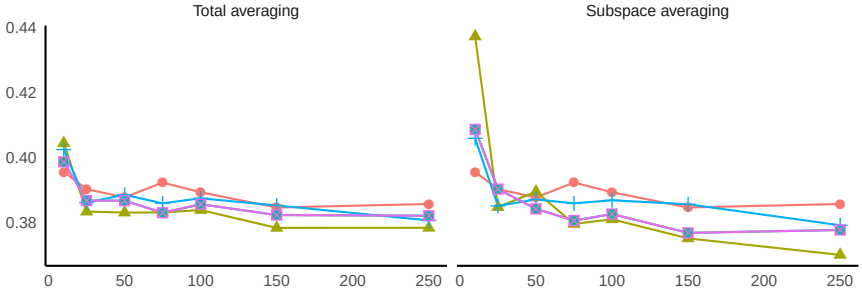
## A.1 Saturation Curves

Below we provide saturation curves of original and ROS ensembles in terms of aRRMSE. Each dataset is represented with two plots. The left and right plots in each figure represent saturation curves for ROS ensembles with total and subspace averaging, respectively. Horizontal and vertical axes denote the number of ensemble members and aRRMSE respectively. Each ensemble variant is represented with a different color as shown in the legends. Numbers in brackets represent different ROS subspace sizes.

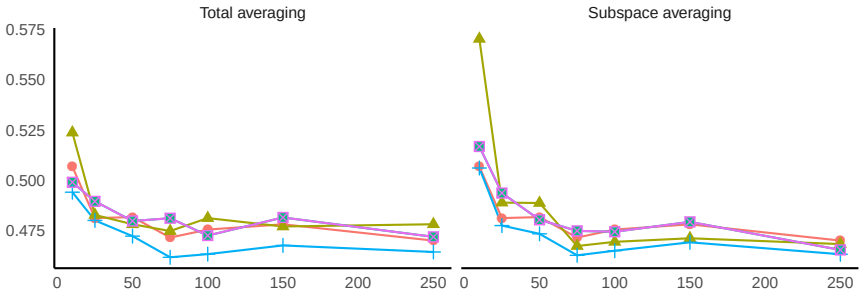
### A.1.1 Bagging ensembles



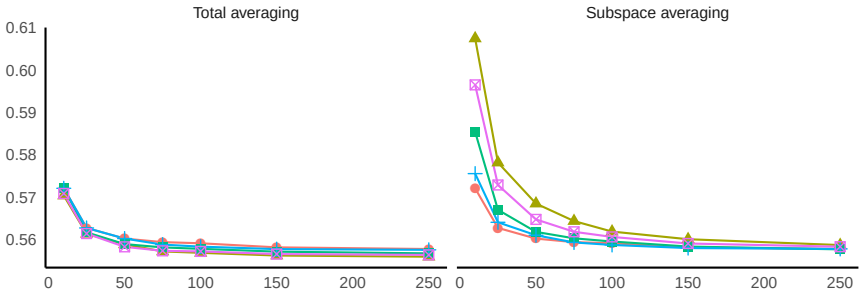

(a) ADNI



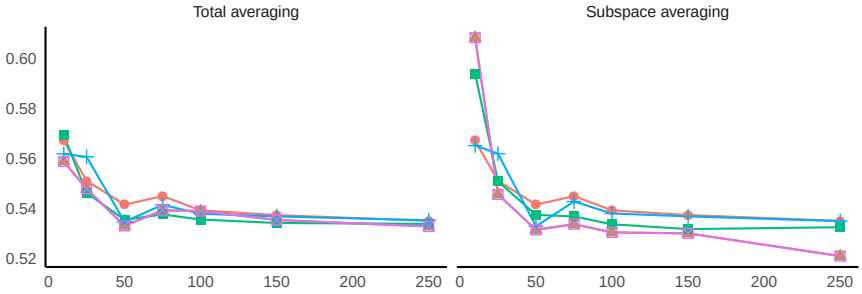

(b) ATP 1D



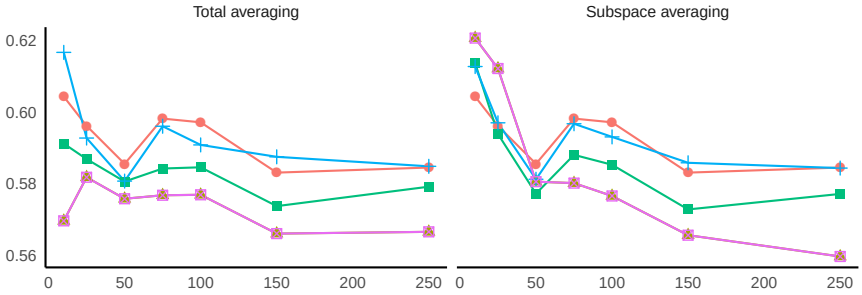

(c) ATP 7D



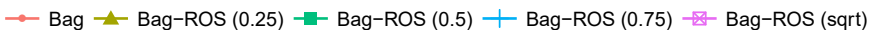

(d) Forestry Kras

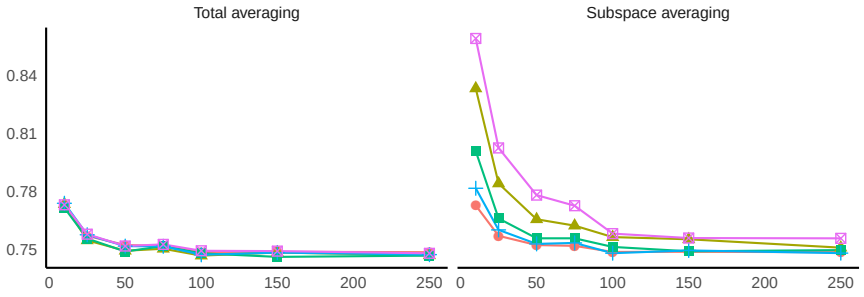



(e) OES 10

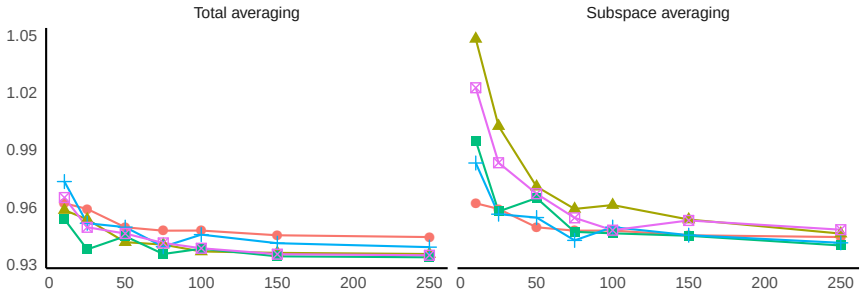



(f) OES 97

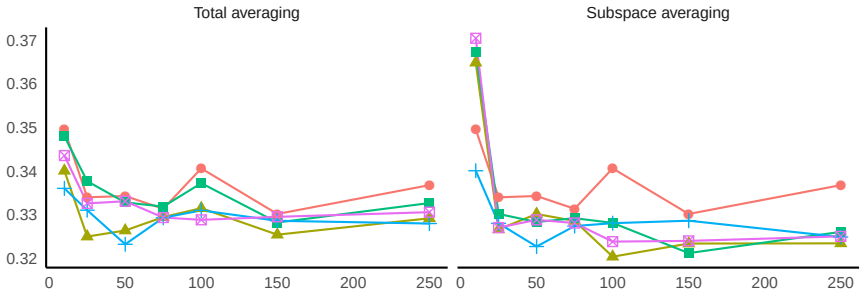




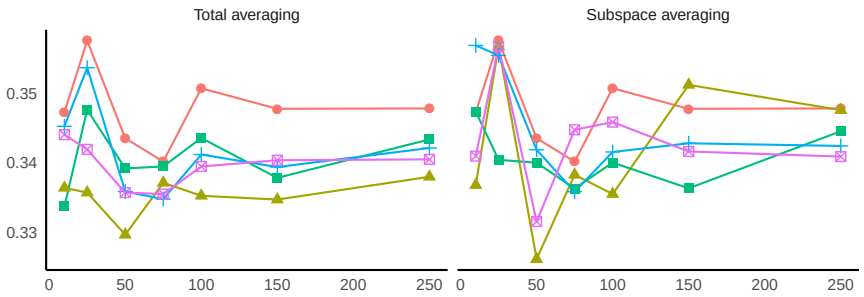

(g) PPMI



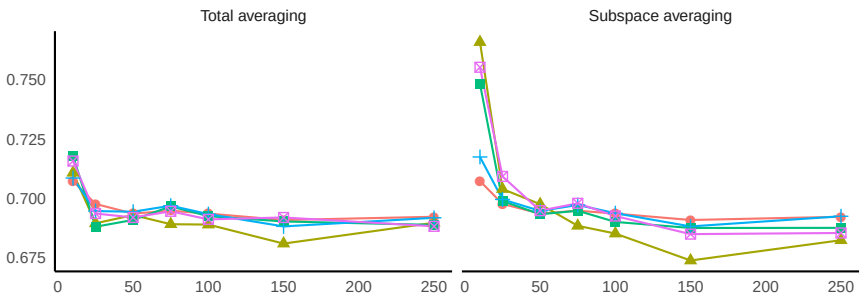

(h) Prespa lake top 10



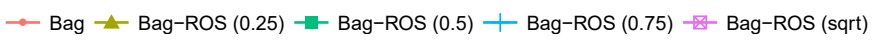

(i) RF 1

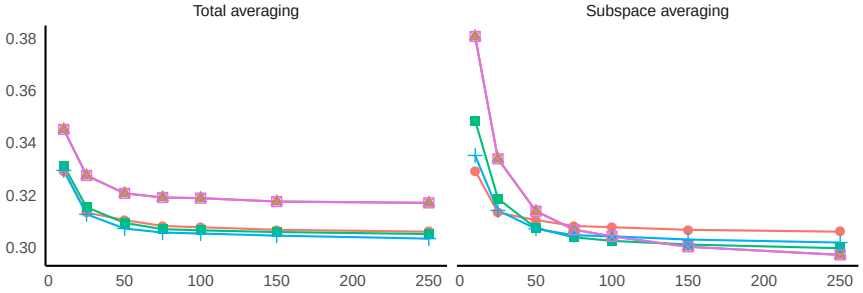



(j) RF 2

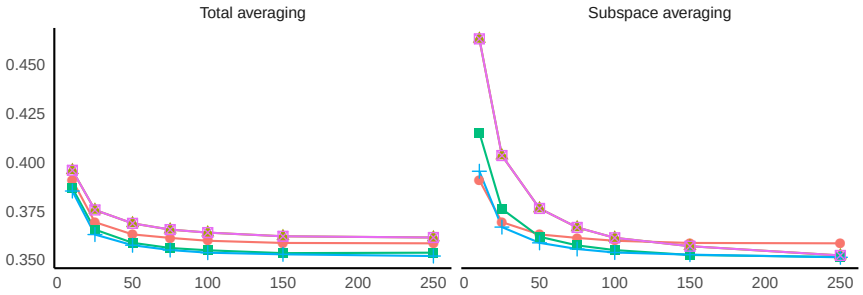



(k) Sales

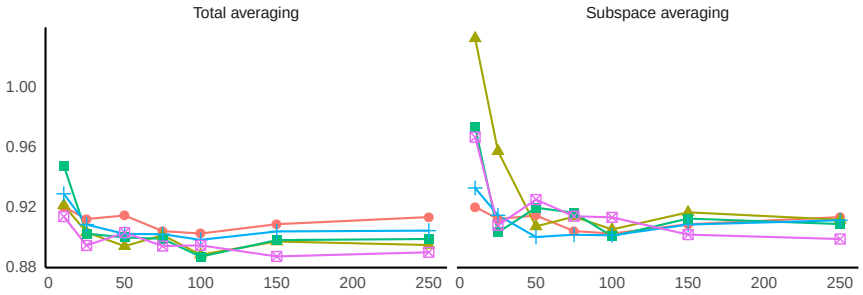




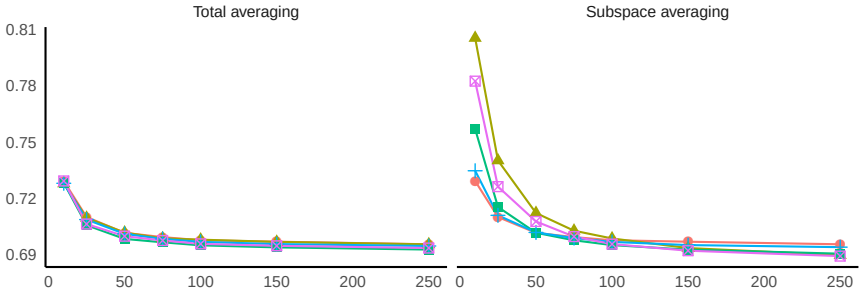

(l) SCM 1D



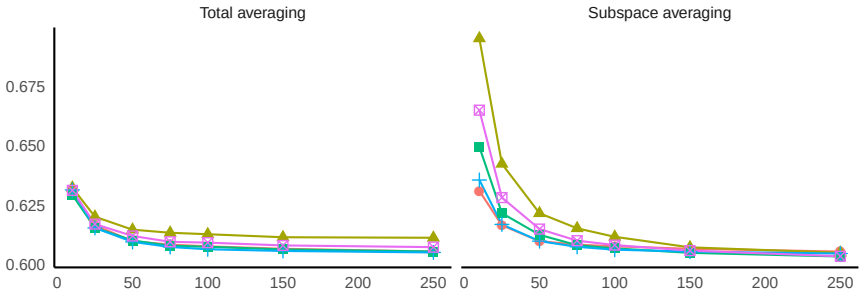

(m) SCM 20D



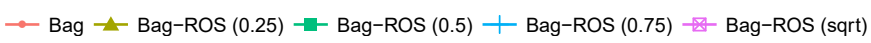

(n) Soil resilience

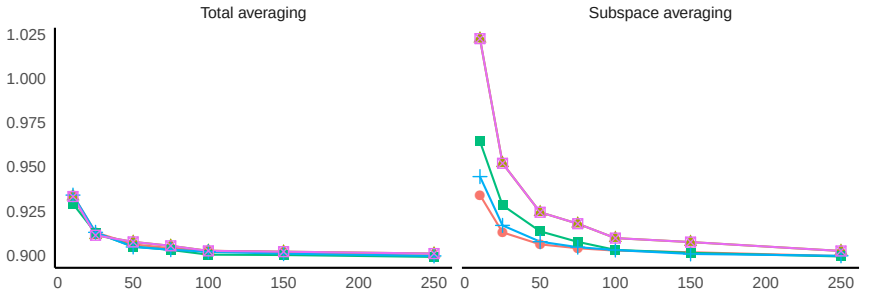



(o) Vegetation clustering




(p) Vegetation condition






(q) Water quality

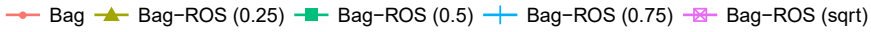
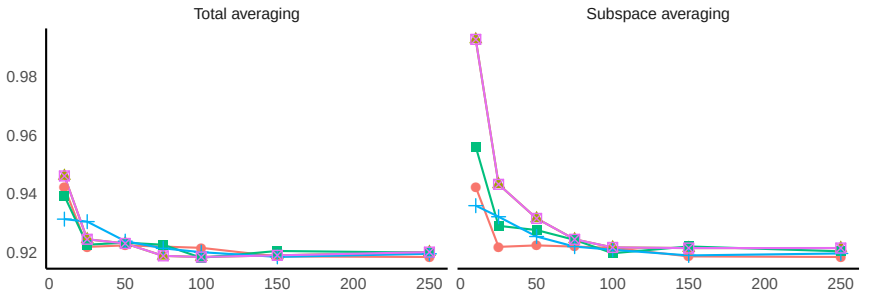
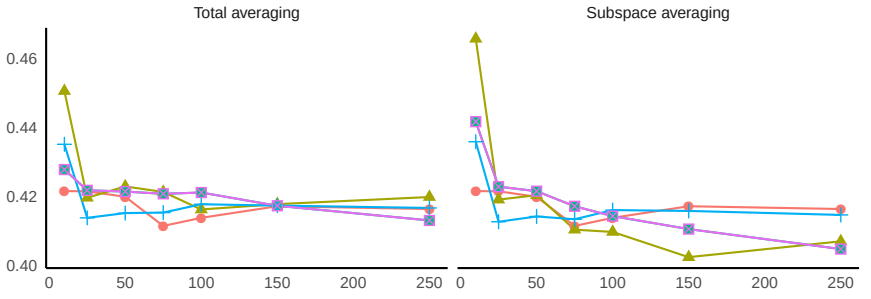



Figure A.1: Per-dataset saturation curves for Bag-ROS ensembles.

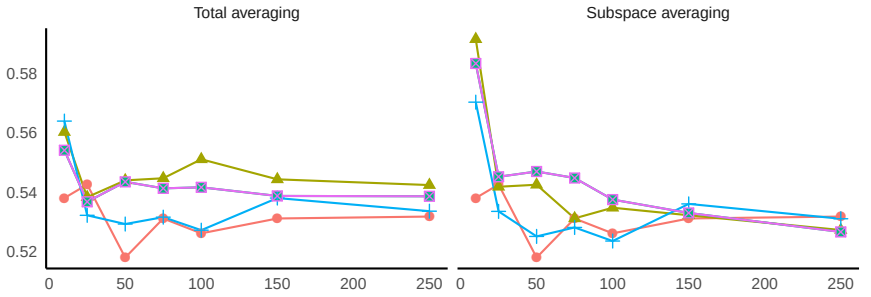
### A.1.2 Random forest ensembles



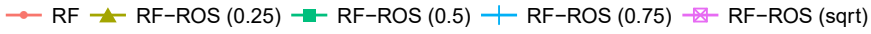

(a) ADNI

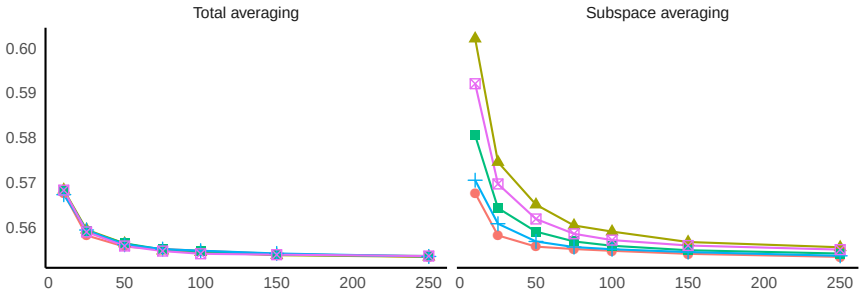



(b) ATP 1D

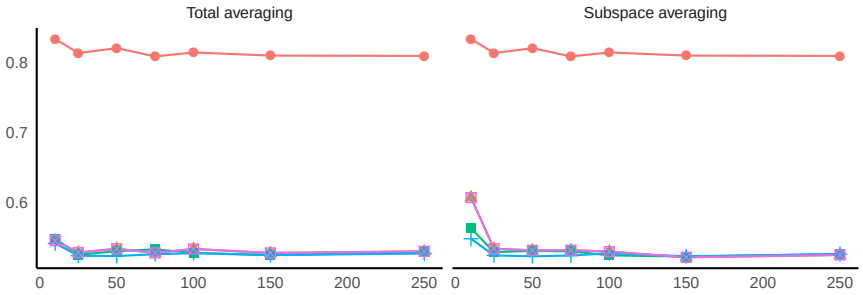



(c) ATP 7D

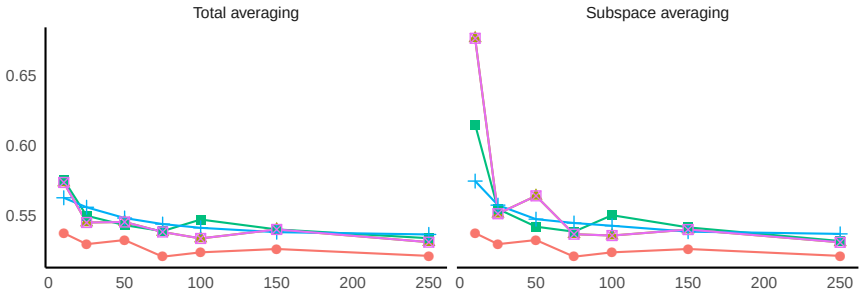




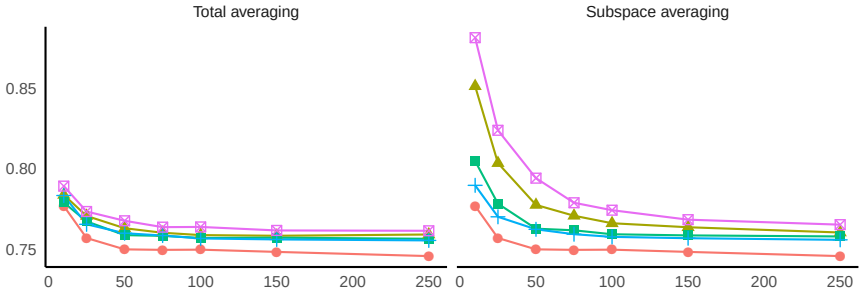

(d) Forestry Kras



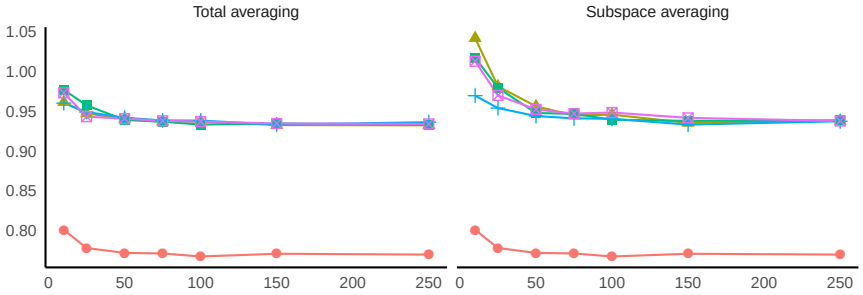

(e) OES 10



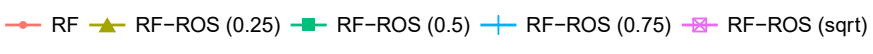

(f) OES 97

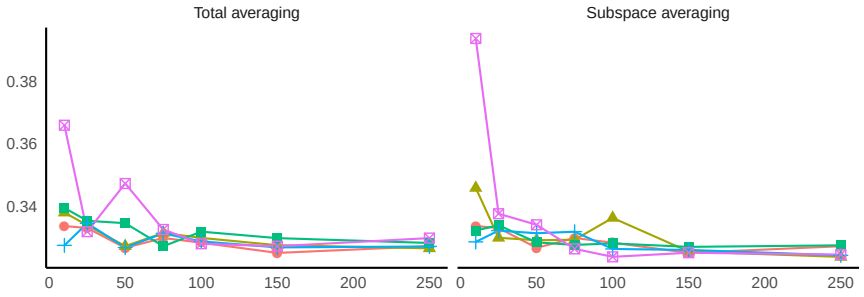



(g) PPMI

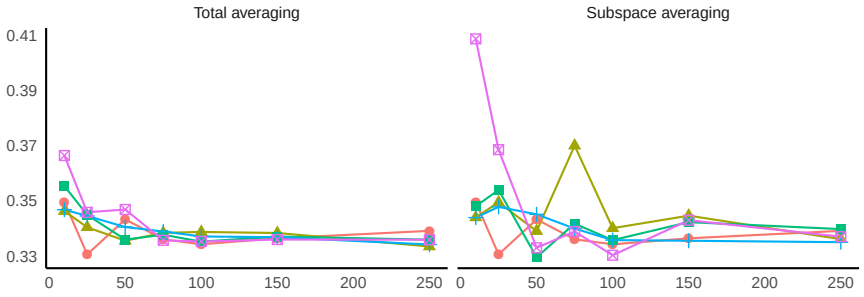



(h) Prespa lake top 10

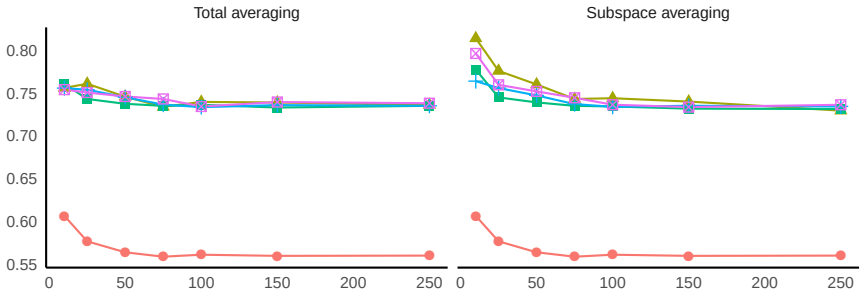




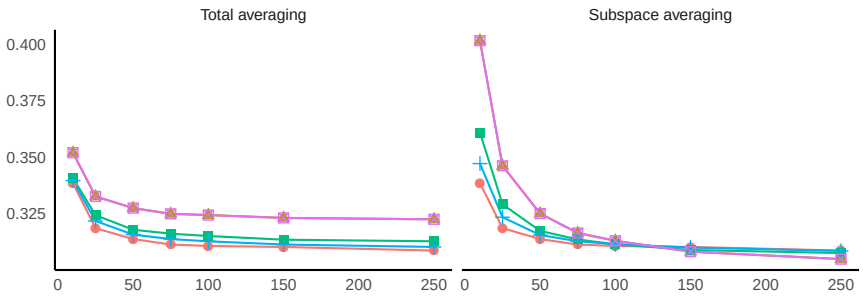

(i) RF 1



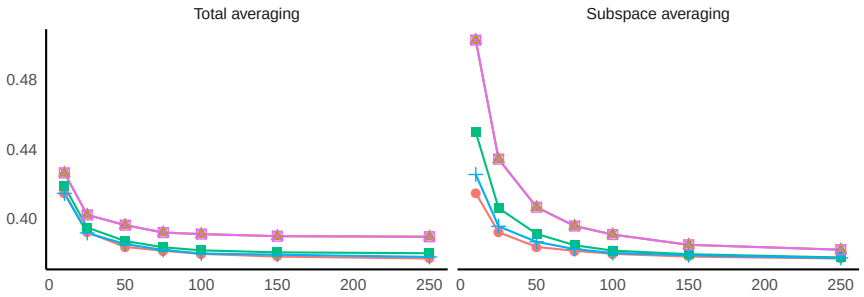

(j) RF 2



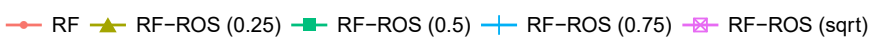

(k) Sales

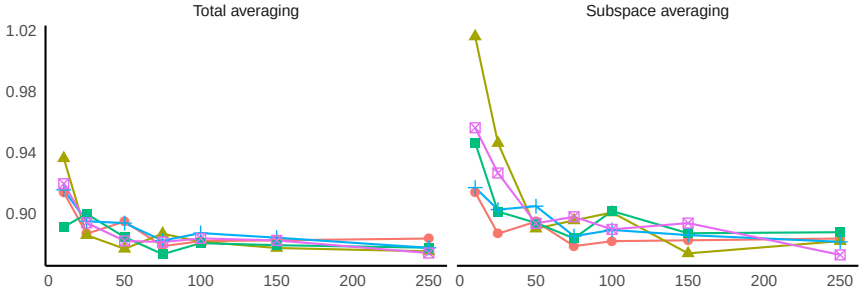



(l) SCM 1D

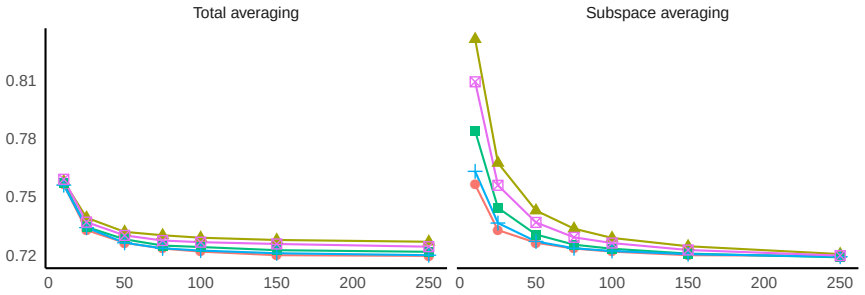



(m) SCM 20D

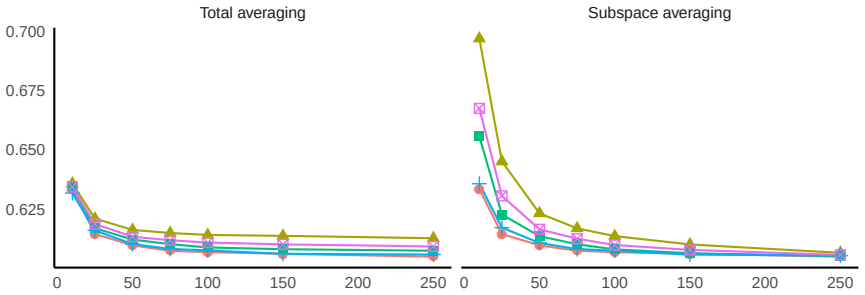




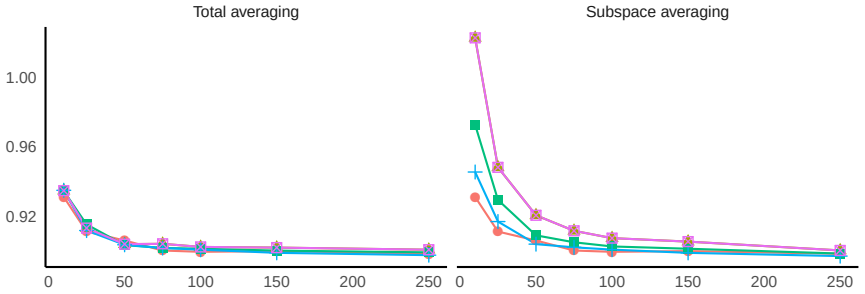

(n) Soil resilience




(o) Vegetation clustering




(p) Vegetation condition




(q) Water quality

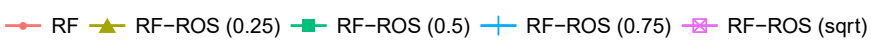
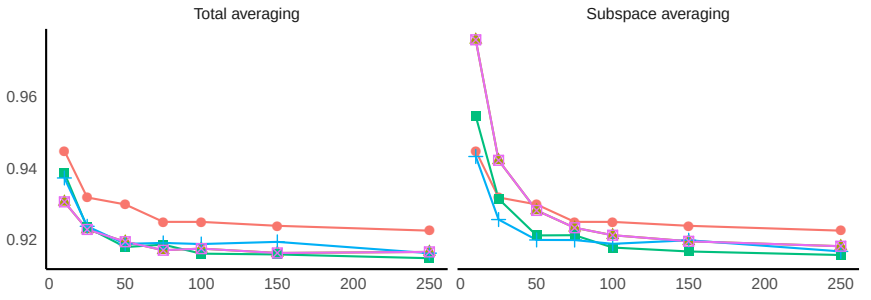
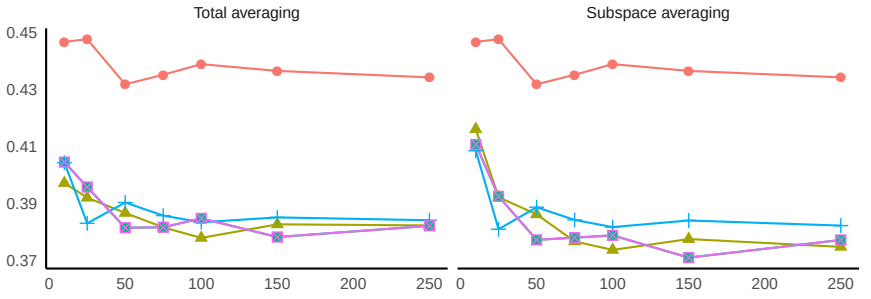



Figure A.2: Per-dataset saturation curves for RF-ROS ensembles.

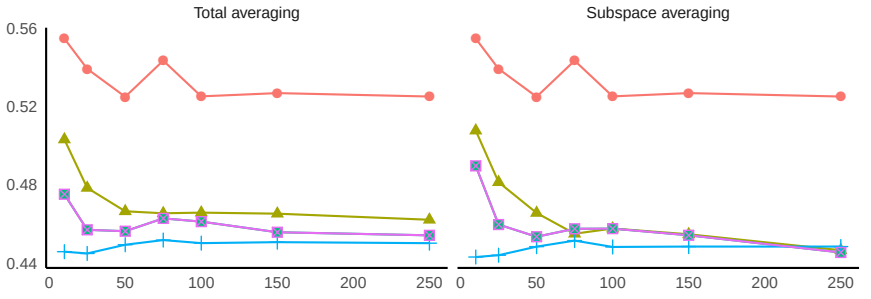
### A.1.3 Extremely randomized tree ensembles



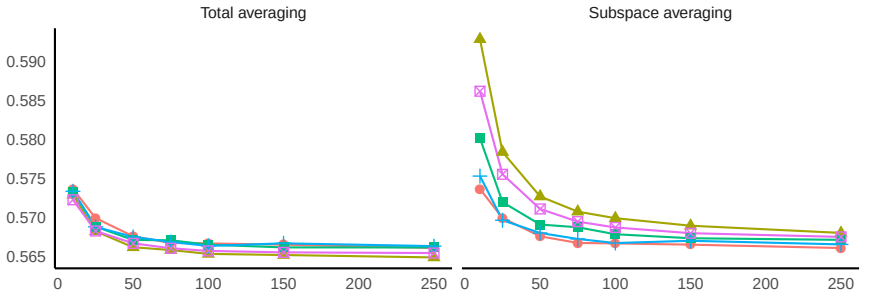

(a) ADNI



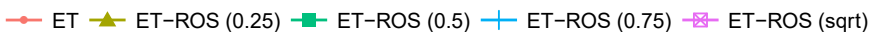

(b) ATP 1D

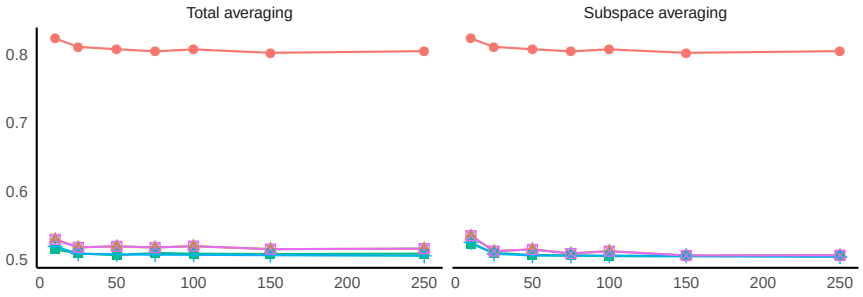



(c) ATP 7D

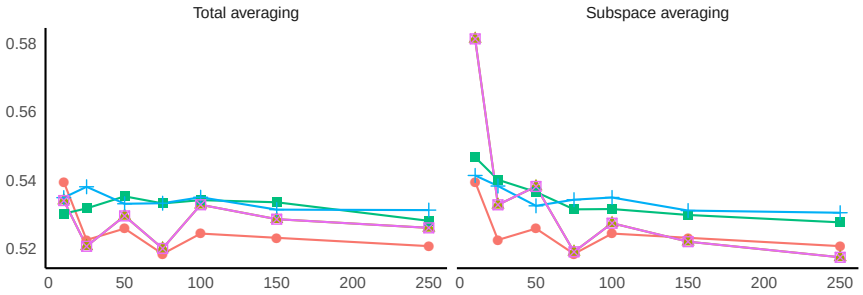



(d) Forestry Kras

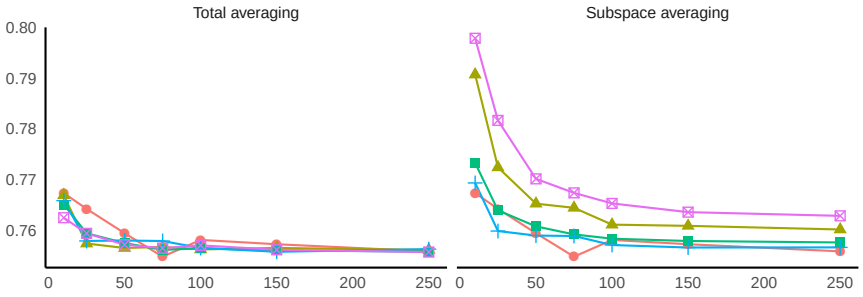




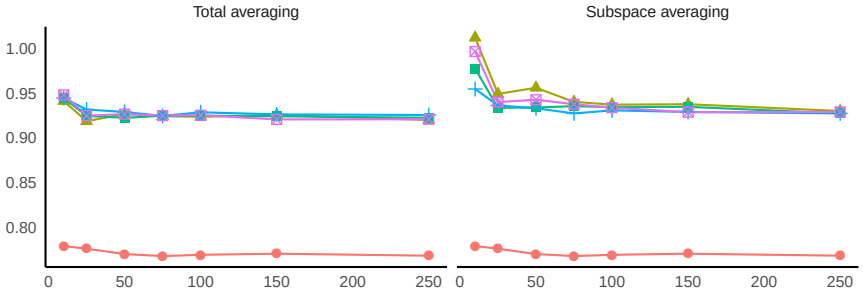

(e) OES 10



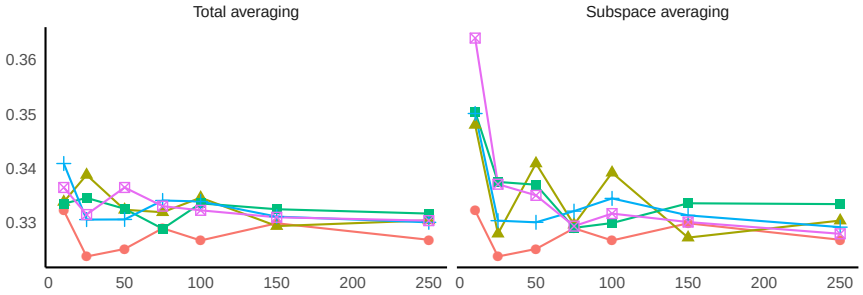

(f) OES 97



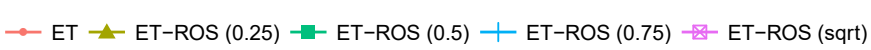

(g) PPMI

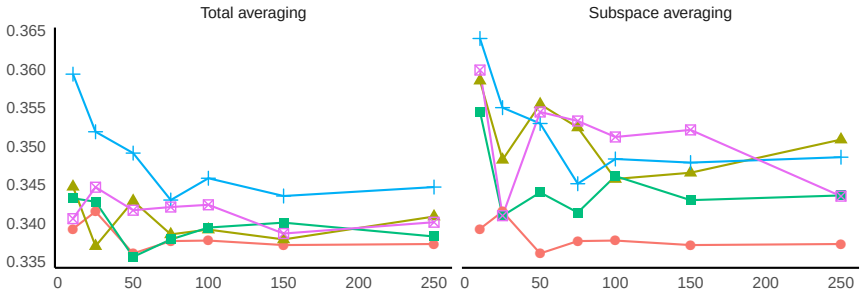



(h) Prespa lake top 10

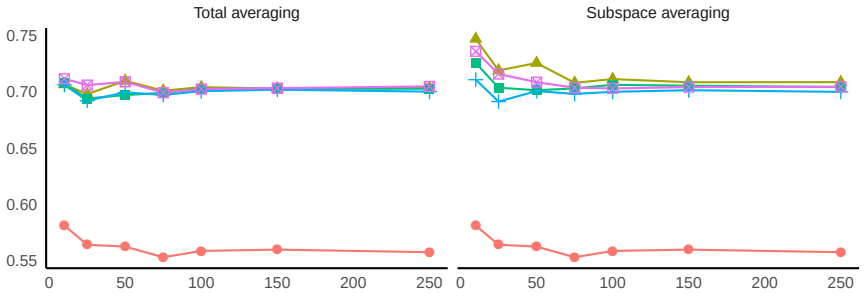



(i) RF 1

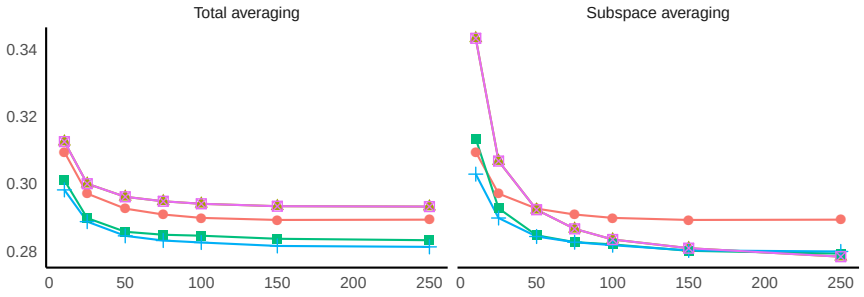




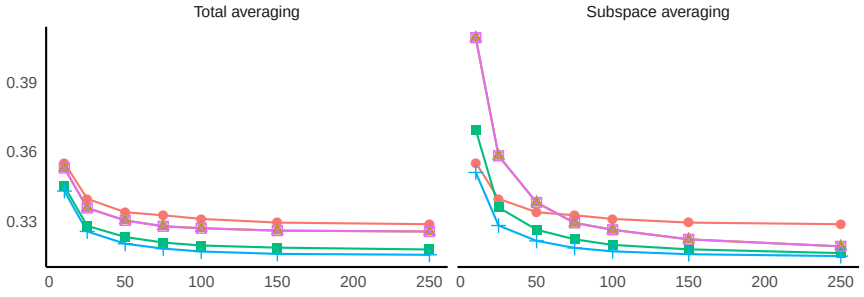

(j) RF 2



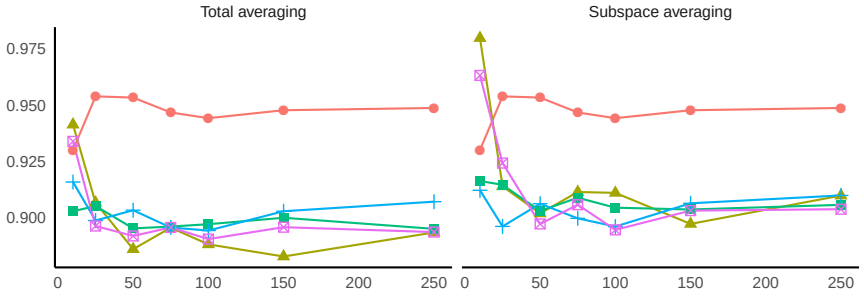

(k) Sales



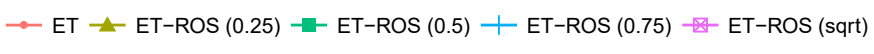

(l) SCM 1D

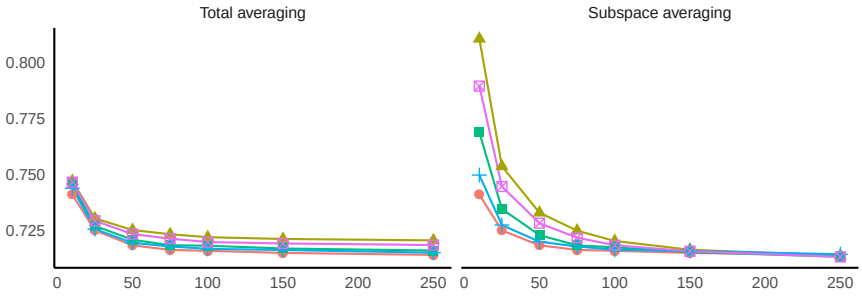



(m) SCM 20D

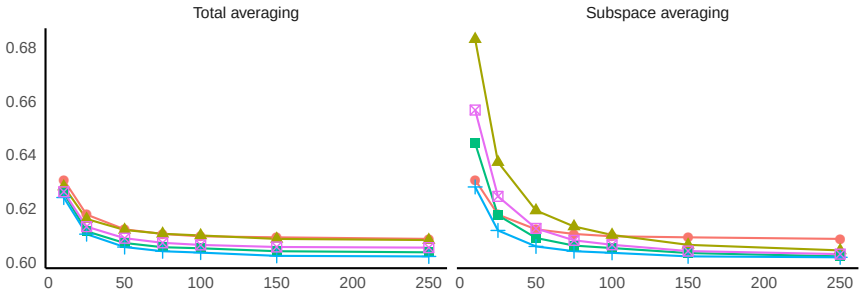



(n) Soil resilience

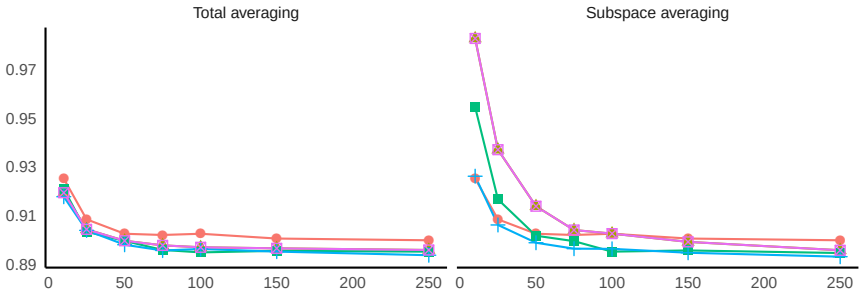





(o) Vegetation clustering




(p) Vegetation condition




(q) Water quality

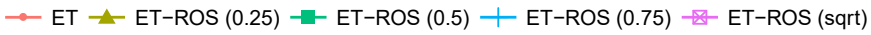
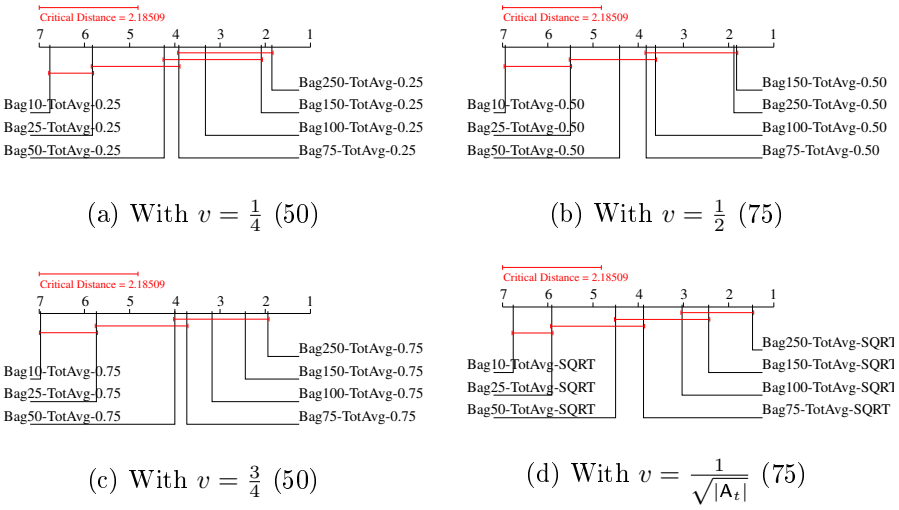



Figure A.3: Per-dataset saturation curves for ET-ROS ensembles.

## A.2 Average Rank Diagrams for Saturation of ROS Ensembles

Below we provide average rank diagrams for all considered ROS variants (Bag-ROS, RF-ROS, ET-ROS) for all considered values of the parameter $v \in \left\{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{1}{\sqrt{\mathsf{A}_t}}\right\}$ and the two types of prediction averaging functions. One average rank diagram corresponds to one choice of a combination of values of the above parameters, for which it compares ensembles of different sizes (10, 25, 50, 75, 100, 150, 250) base models. The saturation point is the lowest number of trees in the ensemble for which the performance is not significantly different than the best. Saturation points are shown in brackets next to value for parameter $v$. Lower ranks are better.

### A.2.1 Bag-ROS saturation.



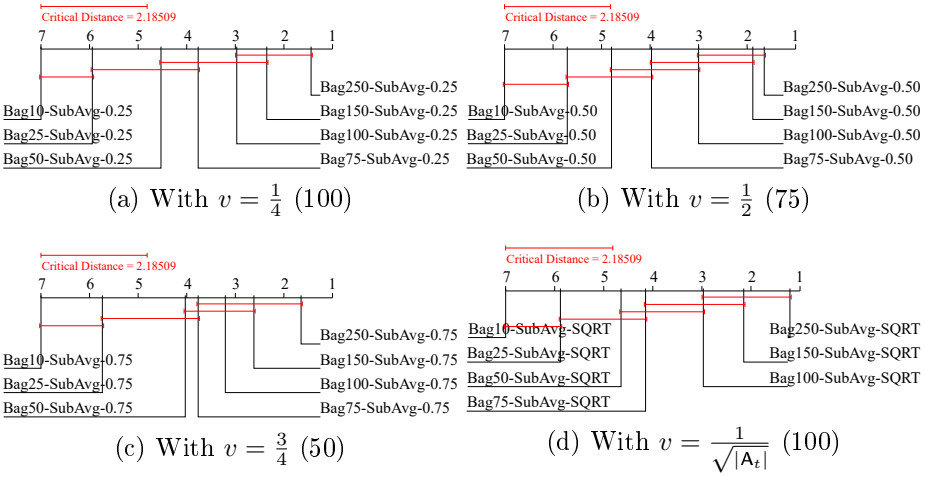

(a) With $v = \frac{1}{4}$ (50)

(b) With $v = \frac{1}{2}$ (75)

(c) With $v = \frac{3}{4}$ (50)

(d) With $v = \frac{1}{\sqrt{|A_t|}}$ (75)

Figure A.4: Bag-ROS saturation with total averaging.




(a) With $v = \frac{1}{4}$ (100)

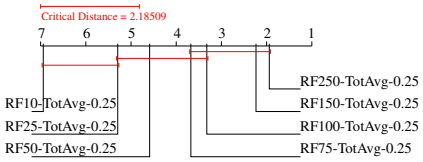
(b) With $v = \frac{1}{2}$ (75)

(c) With $v = \frac{3}{4}$ (50)

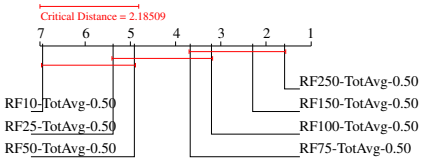
(d) With $v = \frac{1}{\sqrt{|A_t|}}$ (100)

Figure A.5: Bag-ROS saturation with subset averaging.

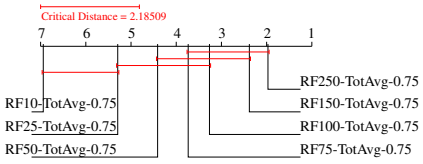
### A.2.2 RF-ROS saturation.



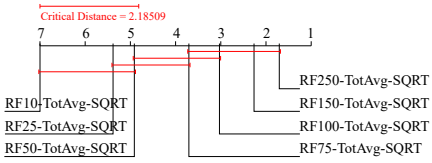

(a) With $v = \frac{1}{4}$ (75)




(b) With $v = \frac{1}{2}$ (75)

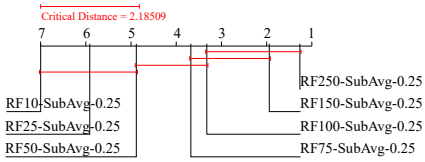



(c) With $v = \frac{3}{4}$ (75)

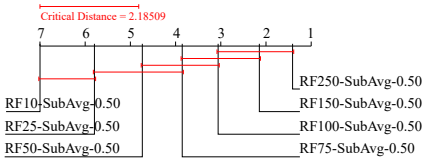



(d) With $v = \frac{1}{\sqrt{|\mathsf{A}_t|}}$ (75)

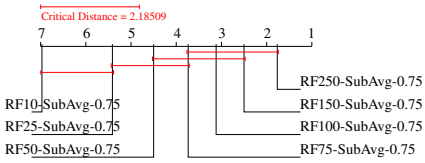
Figure A.6: RF-ROS saturation with total averaging.



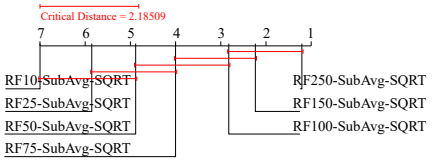

(a) With $v = \frac{1}{4}$ (100)




(b) With $v = \frac{1}{2}$ (100)



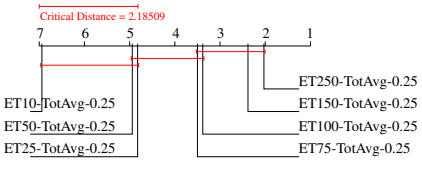

(c) With $v = \frac{3}{4}$ (75)



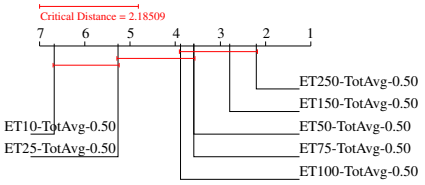

(d) With $v = \frac{1}{\sqrt{|\mathsf{A}_t|}}$ (100)

Figure A.7: RF-ROS saturation with subset averaging.

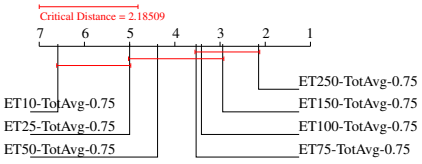
### A.2.3 ET-ROS saturation.



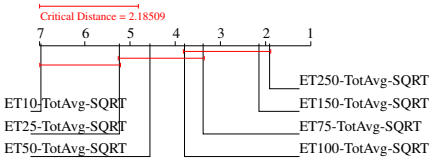

(a) With $v = \frac{1}{4}$ (75)




(b) With $v = \frac{1}{2}$ (50)

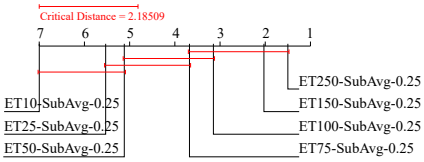



(c) With $v = \frac{3}{4}$ (75)

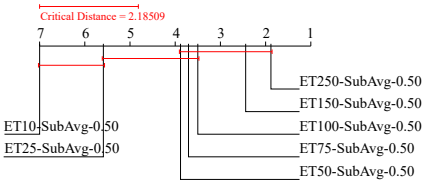



(d) With $v = \frac{1}{\sqrt{|A_t|}}$ (75)

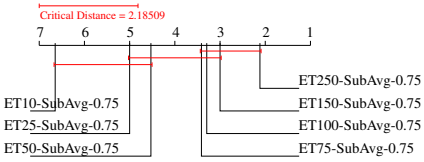
Figure A.8: ET-ROS saturation with total averaging.



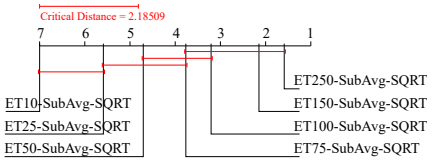

(a) With $v = \frac{1}{4}$ (75)




(b) With $v = \frac{1}{2}$ (50)




(c) With $v = \frac{3}{4}$ (75)




(d) With $v = \frac{1}{\sqrt{|A_t|}}$ (100)

Figure A.9: ET-ROS saturation with subset averaging.

## A.3 Predictive Performance Results in Tabular Form

### A.3.1 Bag-ROS aRRMSE and overfitting scores.

Table A.1: Predictive performance of bagging ensembles in terms of aRRMSE. PCT represents multi-target PCTs. Bag-ST represents ensembles of single-target PCTs. Bag are original bagging ensembles. Bag-ROS are the ROS variants. Ensembles contain 100 trees (ST ensembles contain 100 trees/target).

| Dataset | PCT | Bag-ST | Bag | Bag-ROS | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Total averaging | | | | Subspace averaging | | | |
| | | | | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ |
| ADNI | 1.086 | 0.923 | 0.917 | 0.919 | 0.923 | 0.921 | 0.919 | 0.926 | 0.926 | 0.921 | 0.926 |
| ATP 1D | 0.515 | 0.369 | 0.385 | 0.378 | 0.38 | 0.382 | 0.38 | 0.369 | 0.374 | 0.381 | 0.374 |
| ATP 7D | 0.585 | 0.486 | 0.473 | 0.474 | 0.467 | 0.461 | 0.467 | 0.458 | 0.464 | 0.462 | 0.464 |
| Forestry Kras | 0.61 | 0.551 | 0.55 | 0.548 | 0.548 | 0.549 | 0.548 | 0.553 | 0.55 | 0.549 | 0.551 |
| OES 10 | 0.616 | 0.487 | 0.531 | 0.531 | 0.527 | 0.53 | 0.531 | 0.522 | 0.526 | 0.53 | 0.522 |
| OES 97 | 0.704 | 0.52 | 0.585 | 0.565 | 0.572 | 0.578 | 0.565 | 0.566 | 0.573 | 0.581 | 0.566 |
| PPMI | 0.868 | 0.751 | 0.739 | 0.738 | 0.739 | 0.738 | 0.741 | 0.746 | 0.742 | 0.739 | 0.749 |
| Prespa lake top 10 | 0.995 | 0.968 | 0.946 | 0.935 | 0.937 | 0.944 | 0.937 | 0.959 | 0.944 | 0.948 | 0.946 |
| RF 1 | 0.19 | 0.153 | 0.156 | 0.148 | 0.145 | 0.144 | 0.143 | 0.14 | 0.142 | 0.143 | 0.141 |
| RF 2 | 0.199 | 0.157 | 0.16 | 0.15 | 0.147 | 0.147 | 0.147 | 0.145 | 0.146 | 0.148 | 0.148 |
| Sales | 0.867 | 0.681 | 0.691 | 0.687 | 0.69 | 0.691 | 0.689 | 0.682 | 0.688 | 0.692 | 0.69 |
| SCM 1D | 0.433 | 0.298 | 0.307 | 0.318 | 0.305 | 0.304 | 0.318 | 0.303 | 0.301 | 0.303 | 0.303 |
| SCM 20D | 0.527 | 0.363 | 0.359 | 0.363 | 0.354 | 0.353 | 0.363 | 0.36 | 0.354 | 0.353 | 0.36 |
| Soil resilience | 0.926 | 0.889 | 0.883 | 0.874 | 0.872 | 0.88 | 0.88 | 0.885 | 0.883 | 0.883 | 0.893 |
| Vegetation clustering | 0.809 | 0.699 | 0.706 | 0.702 | 0.699 | 0.702 | 0.699 | 0.702 | 0.699 | 0.702 | 0.7 |
| Vegetation condition | 0.658 | 0.602 | 0.603 | 0.608 | 0.603 | 0.602 | 0.605 | 0.607 | 0.602 | 0.602 | 0.604 |
| Water quality | 0.952 | 0.913 | 0.901 | 0.901 | 0.899 | 0.9 | 0.901 | 0.908 | 0.901 | 0.901 | 0.908 |

Table A.2: Predictive performance of bagging ensembles in terms of OS. PCT represents multi-target PCTs. Bag-ST represents ensembles of single-target PCTs. Bag are original bagging ensembles. Bag-ROS are the ROS variants. Ensembles contain 100 trees (ST ensembles contain 100 trees/target).

| Dataset | PCT | Bag-ST | Bag | Bag-ROS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Total averaging | | | | Subspace averaging | | | | |
| | | | | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ |
| ADNI | 0.959 | 0.832 | 0.593 | 0.555 | 0.587 | 0.601 | 0.555 | 0.659 | 0.639 | 0.617 | 0.659 |
| ATP 1D | 0.368 | 1.464 | 1.216 | 0.933 | 1.069 | 1.181 | 1.069 | 1.255 | 1.257 | 1.246 | 1.257 |
| ATP 7D | 2.131 | 1.511 | 1.307 | 1.052 | 1.173 | 1.369 | 1.173 | 1.464 | 1.444 | 1.407 | 1.444 |
| Forestry Kras | 0.169 | 0.494 | 0.476 | 0.464 | 0.471 | 0.474 | 0.467 | 0.476 | 0.475 | 0.475 | 0.476 |
| OES 10 | 0.704 | 1.556 | 1.114 | 1.034 | 1.052 | 1.108 | 1.034 | 1.238 | 1.152 | 1.117 | 1.238 |
| OES 97 | 0.745 | 1.427 | 0.973 | 0.839 | 0.933 | 0.958 | 0.839 | 1.055 | 1.038 | 1.013 | 1.055 |
| PPMI | 0.383 | 0.529 | 0.467 | 0.449 | 0.46 | 0.462 | 0.438 | 0.48 | 0.472 | 0.466 | 0.478 |
| Prespa lake top 10 | 0.226 | 1.182 | 0.988 | 0.745 | 0.857 | 0.927 | 0.803 | 1.078 | 1.038 | 0.982 | 1.042 |
| RF 1 | 1.085 | 1.034 | 0.748 | 0.545 | 0.65 | 0.685 | 0.584 | 0.539 | 0.679 | 0.656 | 0.751 |
| RF 2 | 1.219 | 1.046 | 0.806 | 0.532 | 0.654 | 0.699 | 0.608 | 0.568 | 0.683 | 0.695 | 0.8 |
| Sales | 0.792 | 1.128 | 0.781 | 0.667 | 0.728 | 0.763 | 0.698 | 0.875 | 0.83 | 0.814 | 0.867 |
| SCM 1D | 0.681 | 1.482 | 0.975 | 0.846 | 0.923 | 0.958 | 0.846 | 1.105 | 1.031 | 0.999 | 1.105 |
| SCM 20D | 0.595 | 1.469 | 1.054 | 0.961 | 1.011 | 1.032 | 0.961 | 1.138 | 1.096 | 1.061 | 1.138 |
| Soil resilience | 0.217 | 1.153 | 0.791 | 0.679 | 0.746 | 0.784 | 0.697 | 0.944 | 0.878 | 0.852 | 0.872 |
| Vegetation clustering | 0.085 | 1.179 | 1.018 | 0.839 | 0.932 | 0.993 | 0.878 | 1.06 | 1.036 | 1.023 | 1.05 |
| Vegetation condition | 0.075 | 1.352 | 1.061 | 0.628 | 0.903 | 1.021 | 0.8 | 1.205 | 1.136 | 1.08 | 1.172 |
| Water quality | 0.018 | 1.245 | 0.915 | 0.643 | 0.788 | 0.871 | 0.643 | 1.041 | 0.976 | 0.938 | 1.041 |

### A.3.2 RF-ROS aRRMSE and overfitting scores.

Table A.3: Predictive performance of random forest ensembles in terms of aRRMSE. PCT represents multi-target PCTs. RF-ST represents ensembles of single-target PCTs. RF are original random forest ensembles. RF-ROS are the ROS variants. Ensembles contain 100 trees (ST ensembles contain 100 trees/target).

| Dataset | PCT | RF-ST | RF | RF-ROS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Total averaging | | | | Subspace averaging | | | | |
| | | | | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ |
| ADNI | 1.086 | 0.92 | 0.919 | 0.916 | 0.916 | 0.917 | 0.916 | 0.919 | 0.917 | 0.918 | 0.919 |
| ATP 1D | 0.515 | 0.398 | 0.41 | 0.413 | 0.418 | 0.414 | 0.418 | 0.404 | 0.41 | 0.412 | 0.41 |
| ATP 7D | 0.585 | 0.518 | 0.521 | 0.545 | 0.535 | 0.521 | 0.535 | 0.528 | 0.53 | 0.517 | 0.53 |
| Forestry Kras | 0.61 | 0.546 | 0.545 | 0.545 | 0.545 | 0.545 | 0.545 | 0.55 | 0.547 | 0.546 | 0.548 |
| OES 10 | 0.616 | 0.492 | 0.517 | 0.525 | 0.519 | 0.518 | 0.525 | 0.521 | 0.515 | 0.518 | 0.521 |
| OES 97 | 0.704 | 0.491 | 0.519 | 0.52 | 0.532 | 0.526 | 0.52 | 0.523 | 0.534 | 0.528 | 0.523 |
| PPMI | 0.868 | 0.759 | 0.748 | 0.751 | 0.749 | 0.749 | 0.757 | 0.758 | 0.751 | 0.75 | 0.766 |
| Prespa lake top 10 | 0.995 | 0.942 | 0.937 | 0.936 | 0.932 | 0.937 | 0.936 | 0.944 | 0.938 | 0.939 | 0.947 |
| RF 1 | 0.19 | 0.144 | 0.154 | 0.157 | 0.155 | 0.154 | 0.154 | 0.155 | 0.152 | 0.152 | 0.15 |
| RF 2 | 0.199 | 0.161 | 0.167 | 0.173 | 0.169 | 0.169 | 0.17 | 0.17 | 0.168 | 0.168 | 0.166 |
| Sales | 0.867 | 0.721 | 0.735 | 0.738 | 0.735 | 0.732 | 0.733 | 0.743 | 0.733 | 0.733 | 0.735 |
| SCM 1D | 0.433 | 0.292 | 0.31 | 0.323 | 0.314 | 0.312 | 0.323 | 0.312 | 0.31 | 0.31 | 0.312 |
| SCM 20D | 0.527 | 0.378 | 0.378 | 0.39 | 0.38 | 0.379 | 0.39 | 0.39 | 0.38 | 0.379 | 0.39 |
| Soil resilience | 0.926 | 0.86 | 0.864 | 0.871 | 0.865 | 0.872 | 0.869 | 0.887 | 0.884 | 0.872 | 0.87 |
| Vegetation clustering | 0.809 | 0.696 | 0.703 | 0.711 | 0.705 | 0.703 | 0.708 | 0.709 | 0.704 | 0.703 | 0.706 |
| Vegetation condition | 0.658 | 0.601 | 0.602 | 0.61 | 0.604 | 0.603 | 0.606 | 0.609 | 0.603 | 0.603 | 0.605 |
| Water quality | 0.952 | 0.902 | 0.898 | 0.901 | 0.899 | 0.899 | 0.901 | 0.905 | 0.901 | 0.899 | 0.905 |

Table A.4: Predictive performance of random forest ensembles in terms of OS. PCT represents multi-target PCTs. RF-ST represents ensembles of single-target PCTs. RF are original random forest ensembles. RF-ROS are the ROS variants. Ensembles contain 100 trees (ST ensembles contain 100 trees/target).

| Dataset | PCT | RF-ST | RF | RF-ROS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Total averaging | | | | Subspace averaging | | | | |
| | | | | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ |
| ADNI | 0.959 | 0.645 | 0.481 | 0.447 | 0.462 | 0.479 | 0.447 | 0.514 | 0.495 | 0.49 | 0.514 |
| ATP 1D | 0.368 | 1.31 | 1.115 | 0.982 | 1.127 | 1.186 | 1.127 | 1.243 | 1.292 | 1.216 | 1.292 |
| ATP 7D | 2.131 | 1.411 | 1.177 | 1.002 | 1.176 | 1.205 | 1.176 | 1.242 | 1.309 | 1.225 | 1.309 |
| Forestry Kras | 0.169 | 0.457 | 0.44 | 0.434 | 0.438 | 0.44 | 0.435 | 0.443 | 0.44 | 0.44 | 0.441 |
| OES 10 | 0.704 | 1.525 | 1.093 | 1.059 | 1.074 | 1.108 | 1.059 | 1.276 | 1.105 | 1.132 | 1.276 |
| OES 97 | 0.745 | 1.291 | 0.88 | 0.804 | 0.877 | 0.871 | 0.804 | 1.009 | 0.974 | 0.918 | 1.009 |
| PPMI | 0.383 | 0.531 | 0.471 | 0.464 | 0.47 | 0.475 | 0.462 | 0.487 | 0.478 | 0.479 | 0.495 |
| Prespa lake top 10 | 0.226 | 1.071 | 0.917 | 0.718 | 0.782 | 0.859 | 0.764 | 0.913 | 0.926 | 0.902 | 0.941 |
| RF 1 | 1.085 | 0.678 | 0.511 | 0.493 | 0.521 | 0.552 | 0.479 | 1.005 | 0.515 | 0.53 | 0.614 |
| RF 2 | 1.219 | 0.53 | 0.479 | 0.427 | 0.488 | 0.507 | 0.462 | 0.718 | 0.508 | 0.501 | 0.524 |
| Sales | 0.792 | 0.563 | 0.458 | 0.462 | 0.468 | 0.475 | 0.437 | 0.556 | 0.503 | 0.484 | 0.496 |
| SCM 1D | 0.681 | 1.366 | 0.955 | 0.87 | 0.925 | 0.951 | 0.87 | 1.047 | 1 | 0.978 | 1.047 |
| SCM 20D | 0.595 | 1.356 | 1.034 | 0.958 | 0.997 | 1.015 | 0.958 | 1.061 | 1.045 | 1.036 | 1.061 |
| Soil resilience | 0.217 | 1.012 | 0.766 | 0.679 | 0.732 | 0.763 | 0.708 | 0.944 | 0.865 | 0.801 | 0.803 |
| Vegetation clustering | 0.085 | 1.029 | 0.885 | 0.767 | 0.827 | 0.866 | 0.794 | 0.922 | 0.898 | 0.887 | 0.91 |
| Vegetation condition | 0.075 | 1.241 | 0.982 | 0.654 | 0.866 | 0.951 | 0.787 | 1.097 | 1.036 | 0.997 | 1.065 |
| Water quality | 0.018 | 1.12 | 0.836 | 0.659 | 0.761 | 0.812 | 0.659 | 0.946 | 0.898 | 0.857 | 0.946 |

### A.3.3 ET-ROS aRRMSE and overfitting scores.

Table A.5: Predictive performance of extremely randomized tree ensembles in terms of aRRMSE. PCT represents multi-target PCTs. ET-ST represents ensembles of single-target PCTs. ET are original extra tree ensembles. ET-ROS are the ROS variants. Ensembles contain 100 trees (ST ensembles contain 100 trees/target).

| Dataset | PCT | ET-ST | ET | ET-ROS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Total averaging | | | | Subspace averaging | | | | |
| | | | | $v=\frac{1}{4}$ | $v=\frac{1}{2}$ | $v=\frac{3}{4}$ | $v=\frac{1}{\sqrt{\lvert A_t\rvert}}$ | $v=\frac{1}{4}$ | $v=\frac{1}{2}$ | $v=\frac{3}{4}$ | $v=\frac{1}{\sqrt{\lvert A_t\rvert}}$ |
| ADNI | 1.086 | 0.917 | 0.922 | 0.914 | 0.913 | 0.916 | 0.914 | 0.918 | 0.915 | 0.916 | 0.918 |
| ATP 1D | 0.515 | 0.365 | 0.435 | 0.372 | 0.378 | 0.376 | 0.378 | 0.364 | 0.37 | 0.374 | 0.37 |
| ATP 7D | 0.585 | 0.468 | 0.523 | 0.453 | 0.448 | 0.438 | 0.448 | 0.441 | 0.442 | 0.436 | 0.442 |
| Forestry Kras | 0.61 | 0.56 | 0.557 | 0.556 | 0.557 | 0.557 | 0.556 | 0.56 | 0.558 | 0.557 | 0.559 |
| OES 10 | 0.616 | 0.467 | 0.514 | 0.511 | 0.499 | 0.497 | 0.511 | 0.502 | 0.495 | 0.496 | 0.502 |
| OES 97 | 0.704 | 0.464 | 0.52 | 0.517 | 0.517 | 0.518 | 0.517 | 0.513 | 0.514 | 0.518 | 0.513 |
| PPMI | 0.868 | 0.759 | 0.743 | 0.745 | 0.745 | 0.745 | 0.746 | 0.75 | 0.747 | 0.746 | 0.754 |
| Prespa lake top 10 | 0.995 | 0.934 | 0.929 | 0.922 | 0.923 | 0.926 | 0.923 | 0.935 | 0.932 | 0.928 | 0.931 |
| RF 1 | 0.19 | 0.137 | 0.144 | 0.151 | 0.148 | 0.147 | 0.147 | 0.146 | 0.145 | 0.147 | 0.144 |
| RF 2 | 0.199 | 0.147 | 0.149 | 0.154 | 0.151 | 0.152 | 0.152 | 0.149 | 0.152 | 0.153 | 0.152 |
| Sales | 0.867 | 0.733 | 0.763 | 0.702 | 0.7 | 0.698 | 0.7 | 0.709 | 0.704 | 0.698 | 0.701 |
| SCM 1D | 0.433 | 0.274 | 0.289 | 0.293 | 0.284 | 0.282 | 0.293 | 0.283 | 0.281 | 0.281 | 0.283 |
| SCM 20D | 0.527 | 0.322 | 0.33 | 0.326 | 0.319 | 0.316 | 0.326 | 0.325 | 0.319 | 0.316 | 0.325 |
| Soil resilience | 0.926 | 0.884 | 0.929 | 0.869 | 0.877 | 0.873 | 0.871 | 0.884 | 0.88 | 0.873 | 0.87 |
| Vegetation clustering | 0.809 | 0.688 | 0.695 | 0.702 | 0.698 | 0.696 | 0.7 | 0.699 | 0.696 | 0.696 | 0.697 |
| Vegetation condition | 0.658 | 0.6 | 0.605 | 0.606 | 0.601 | 0.599 | 0.602 | 0.606 | 0.601 | 0.599 | 0.602 |
| Water quality | 0.952 | 0.896 | 0.9 | 0.895 | 0.893 | 0.894 | 0.895 | 0.9 | 0.893 | 0.894 | 0.9 |

Table A.6: Predictive performance of extremely randomized tree ensembles in terms of OS. PCT represents multi-target PCTs. ET-ST represents ensembles of single-target PCTs. ET are original extra tree ensembles. ET-ROS are the ROS variants. Ensembles contain 100 trees (ST ensembles contain 100 trees/target).

| Dataset | PCT | ET-ST | ET | ET-ROS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Total averaging | | | | Subspace averaging | | | | |
| | | | | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ | $v = \frac{1}{4}$ | $v = \frac{1}{2}$ | $v = \frac{3}{4}$ | $v = \frac{1}{\sqrt{\lvert A_t \rvert}}$ |
| ADNI | 0.959 | 1.274 | 1.022 | 0.708 | 0.746 | 0.765 | 0.708 | 0.923 | 0.844 | 0.793 | 0.923 |
| ATP 1D | 0.368 | 4 | 2.616 | 1.694 | 2.147 | 2.504 | 2.147 | 3.507 | 3.495 | 2.856 | 3.495 |
| ATP 7D | 2.131 | 5.397 | 2.766 | 2.186 | 2.679 | 3.176 | 2.679 | 4.713 | 4.188 | 3.481 | 4.188 |
| Forestry Kras | 0.169 | 0.611 | 0.575 | 0.567 | 0.575 | 0.576 | 0.57 | 0.595 | 0.584 | 0.579 | 0.589 |
| OES 10 | 0.704 | 2.654 | 0.986 | 1.254 | 1.156 | 1.104 | 1.254 | 1.908 | 1.344 | 1.156 | 1.908 |
| OES 97 | 0.745 | 3.31 | 0.827 | 1.13 | 1.195 | 1.217 | 1.13 | 2.028 | 1.609 | 1.39 | 2.028 |
| PPMI | 0.383 | 0.614 | 0.517 | 0.531 | 0.535 | 0.538 | 0.525 | 0.563 | 0.549 | 0.542 | 0.574 |
| Prespa lake top 10 | 0.226 | 1.99 | 1.419 | 1.018 | 1.167 | 1.356 | 1.095 | 1.651 | 1.548 | 1.489 | 1.57 |
| RF 1 | 1.085 | 0.763 | 0.648 | 0.539 | 0.563 | 0.6 | 0.538 | 0.765 | 0.661 | 0.643 | 0.623 |
| RF 2 | 1.219 | 0.605 | 0.572 | 0.51 | 0.516 | 0.556 | 0.566 | 0.554 | 0.541 | 0.581 | 0.601 |
| Sales | 0.792 | 1.32 | 1.006 | 0.783 | 0.833 | 0.85 | 0.804 | 1.047 | 0.948 | 0.894 | 0.972 |
| SCM 1D | 0.681 | 5.235 | 1.415 | 1.256 | 1.39 | 1.432 | 1.256 | 2.443 | 1.836 | 1.585 | 2.443 |
| SCM 20D | 0.595 | 4.439 | 1.524 | 1.387 | 1.469 | 1.503 | 1.387 | 2.242 | 1.807 | 1.623 | 2.242 |
| Soil resilience | 0.217 | 1.3 | 0.908 | 0.802 | 0.866 | 0.85 | 0.84 | 1.106 | 0.977 | 0.89 | 1.018 |
| Vegetation clustering | 0.085 | 2.562 | 1.785 | 1.293 | 1.558 | 1.73 | 1.404 | 2.159 | 1.953 | 1.839 | 2.071 |
| Vegetation condition | 0.075 | 4.321 | 1.935 | 1.166 | 1.616 | 1.886 | 1.421 | 3.143 | 2.451 | 2.108 | 2.739 |
| Water quality | 0.018 | 2.558 | 1.325 | 0.981 | 1.124 | 1.241 | 0.981 | 1.785 | 1.555 | 1.39 | 1.785 |

# Appendix B

# Additional Details on the Evaluation of Tree Ensembles with Random Output Selections for Multi-Label Classification

## B.1 Per-Dataset Predictive Performance Results

Below we provide figures that show the performance of RF-ROS on the 13 benchmark datasets in terms of 12 different evaluation measures. Each dataset is represented with one plot that contains two curves: one for each prediction aggregation type (see legend for color coding). Horizontal and vertical axes denote ROS subspace sizes and values of one of the 12 calculated evaluation measures, respectively. All points on the plots represent the performance of RF-ROS ensembles except for the right-most point on all plots (with subspace size 100%). Those points represent the original RF ensembles which always use the whole output space.

### B.1.1 Example-based measures

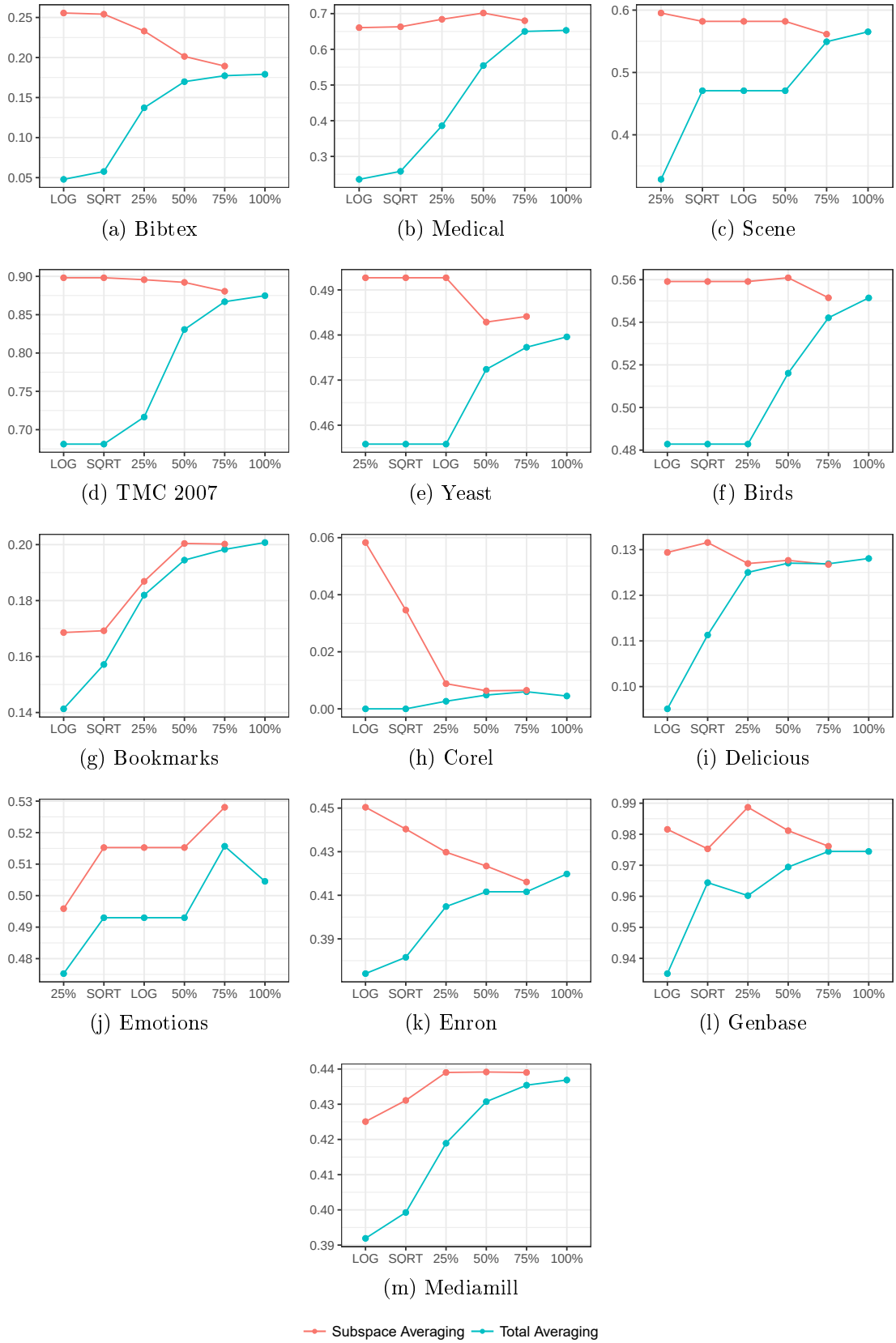



Figure B.1: RF-ROS performance in terms of example-based accuracy.

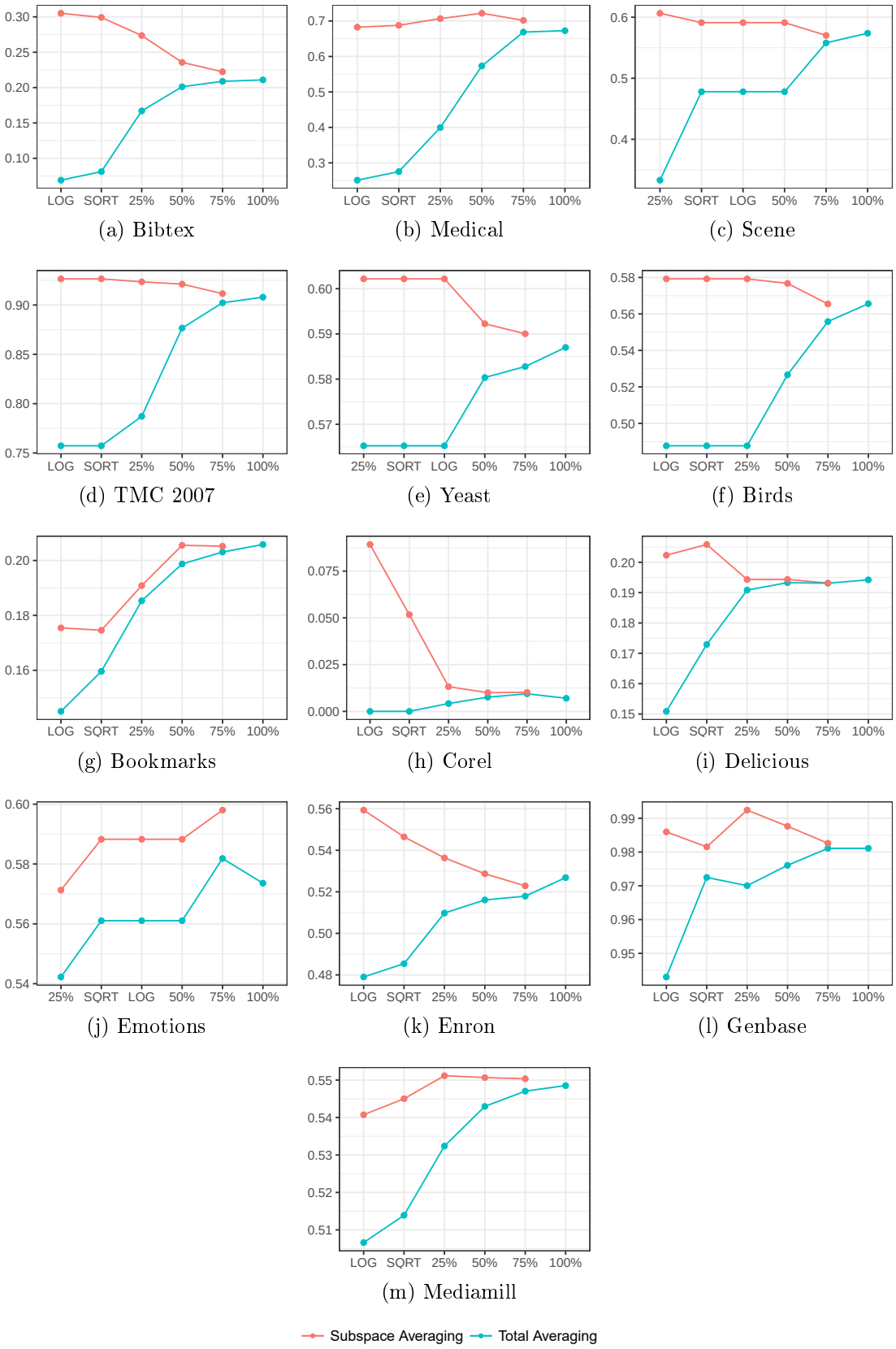



Figure B.2: RF-ROS performance in terms of example-based $F_1$.

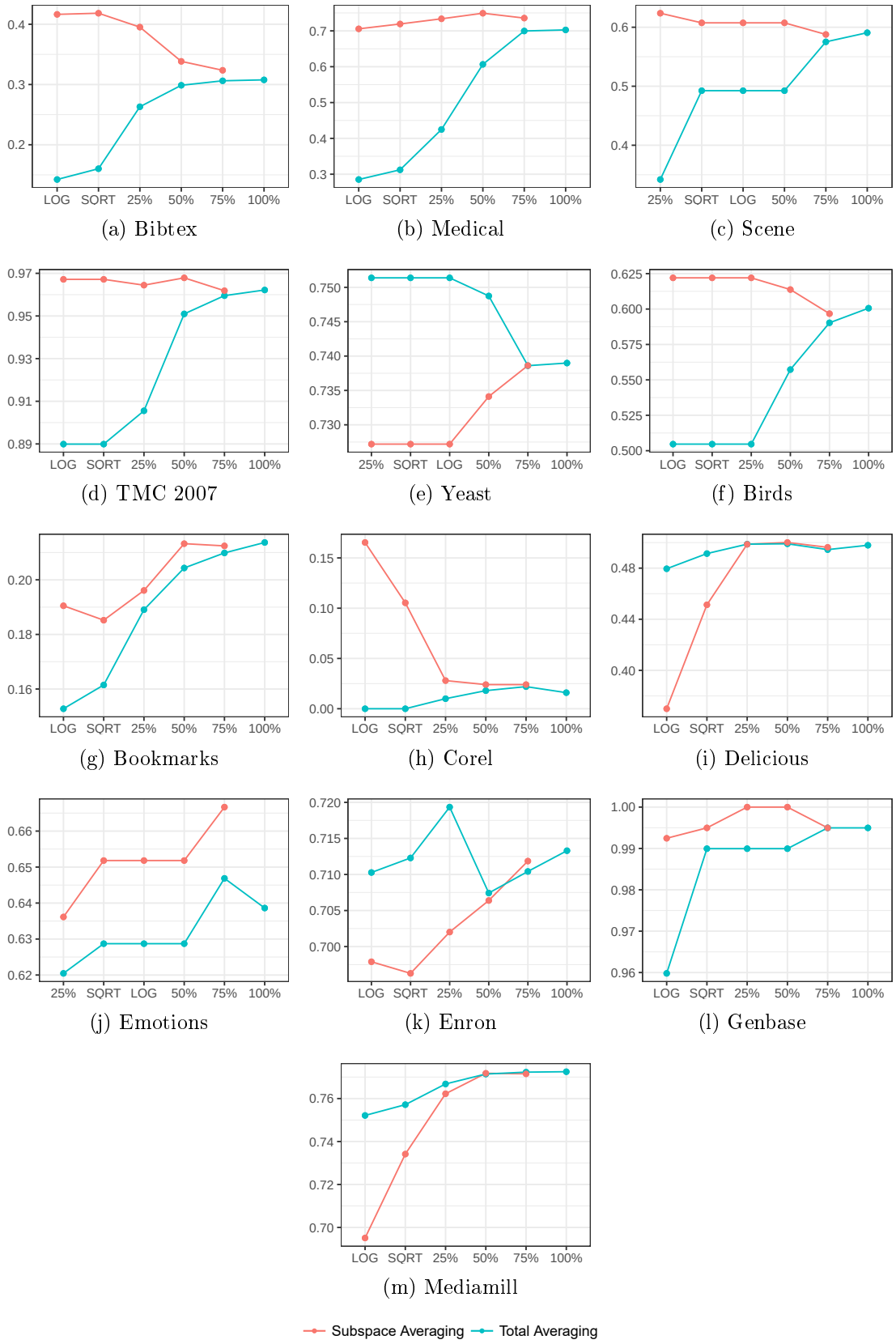
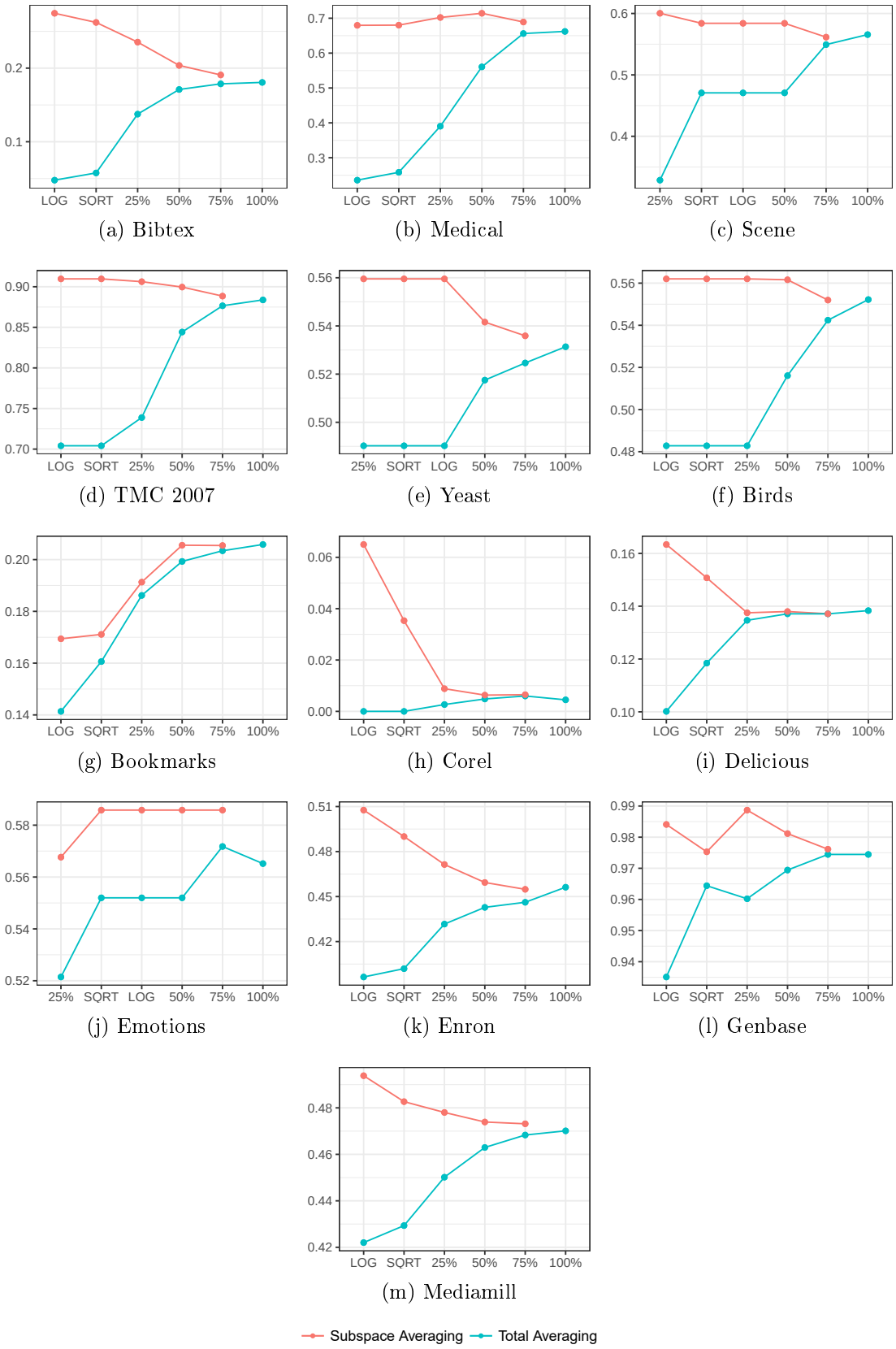



Figure B.3: RF-ROS performance in terms of example-based precision.




(a) Bibtex (b) Medical (c) Scene

(d) TMC 2007 (e) Yeast (f) Birds

(g) Bookmarks (h) Corel (i) Delicious

(j) Emotions (k) Enron (l) Genbase

(m) Mediamill

Figure B.4: RF-ROS performance in terms of example-based recall.

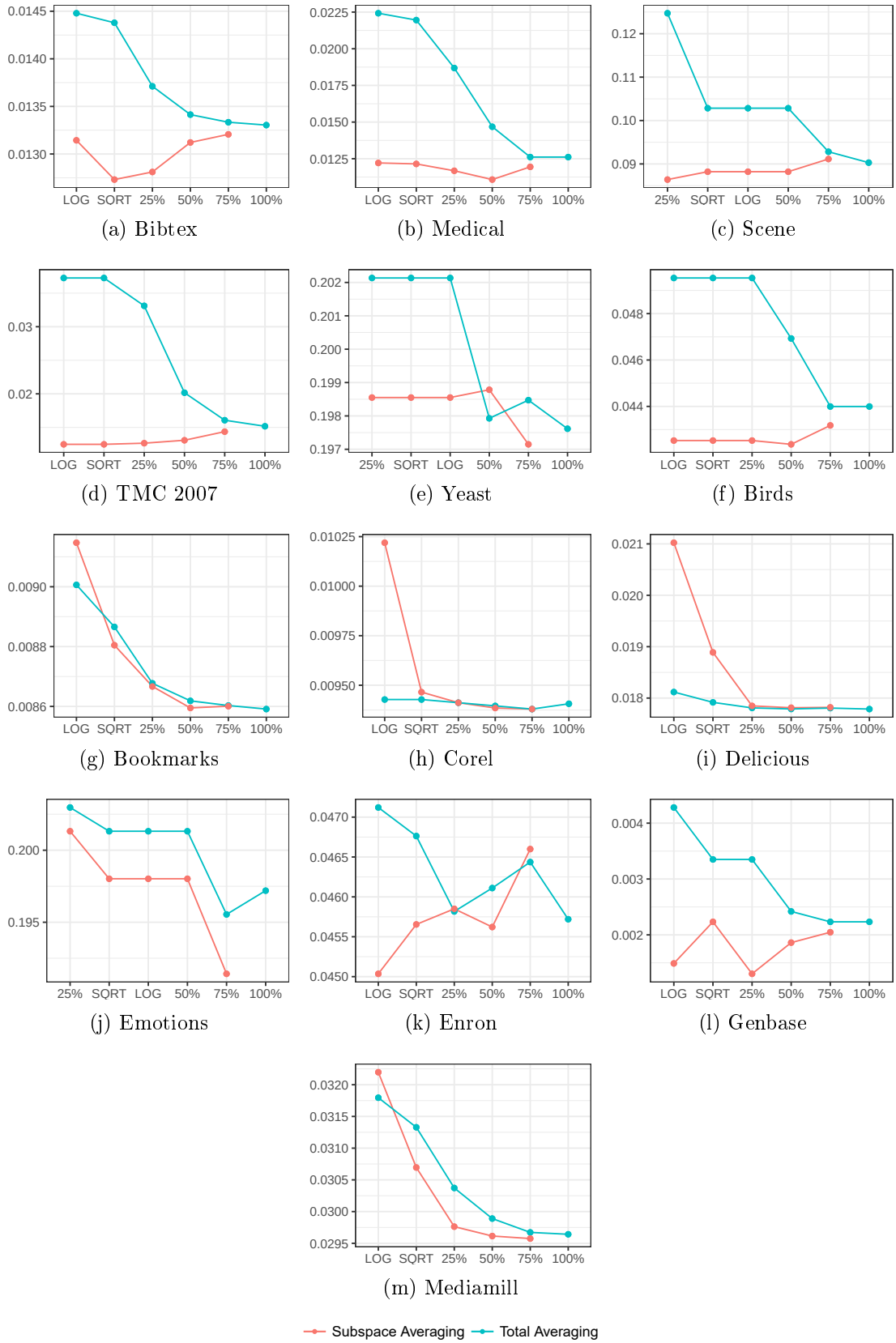
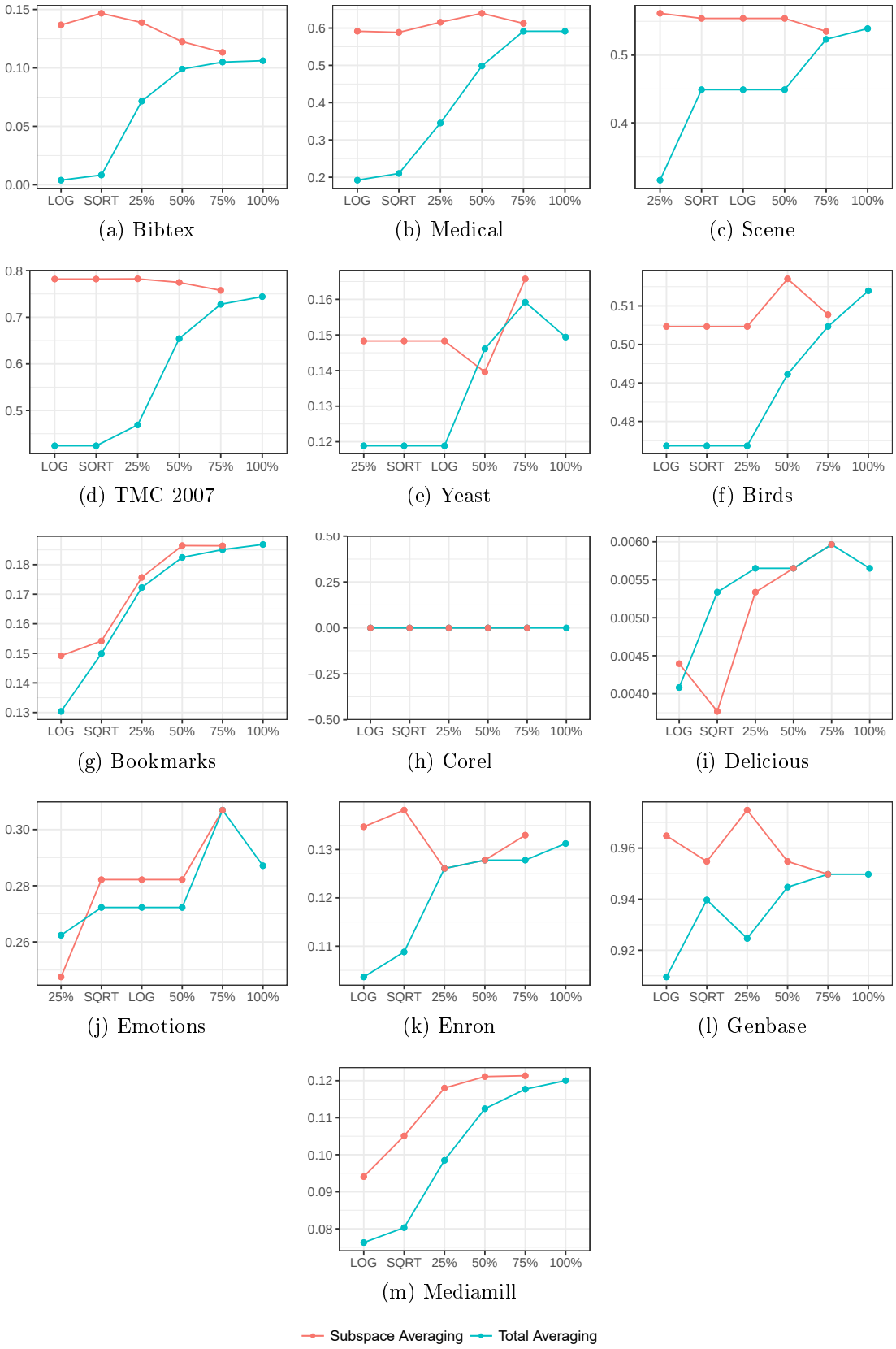



Figure B.5: RF-ROS performance in terms of hamming loss.




(a) Bibtex (b) Medical (c) Scene

(d) TMC 2007 (e) Yeast (f) Birds

(g) Bookmarks (h) Corel (i) Delicious

(j) Emotions (k) Enron (l) Genbase

(m) Mediamill

Figure B.6: RF-ROS performance in terms of subset accuracy.

### B.1.2 Label-based measures

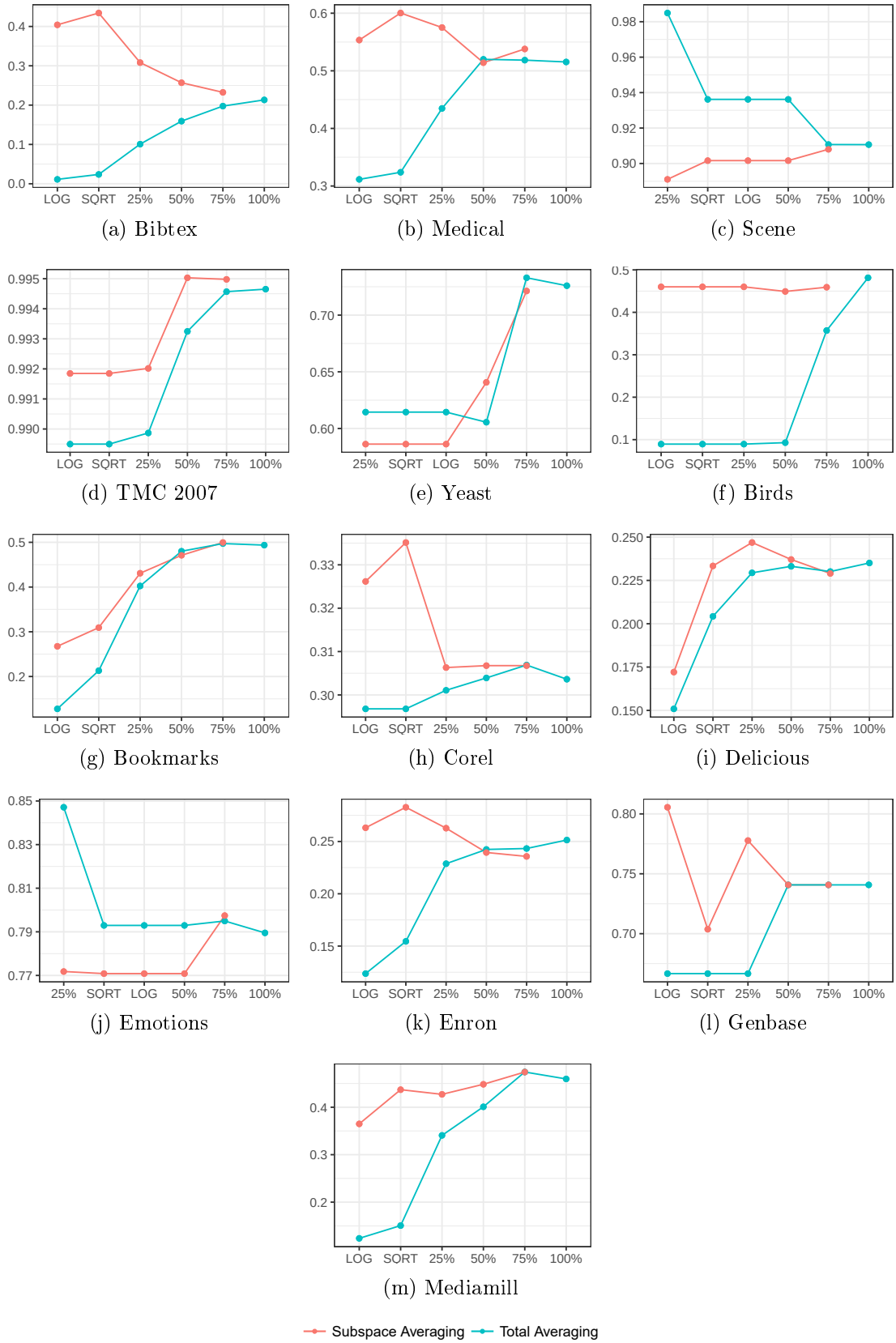



Figure B.7: RF-ROS performance in terms of macro-averaged precision.

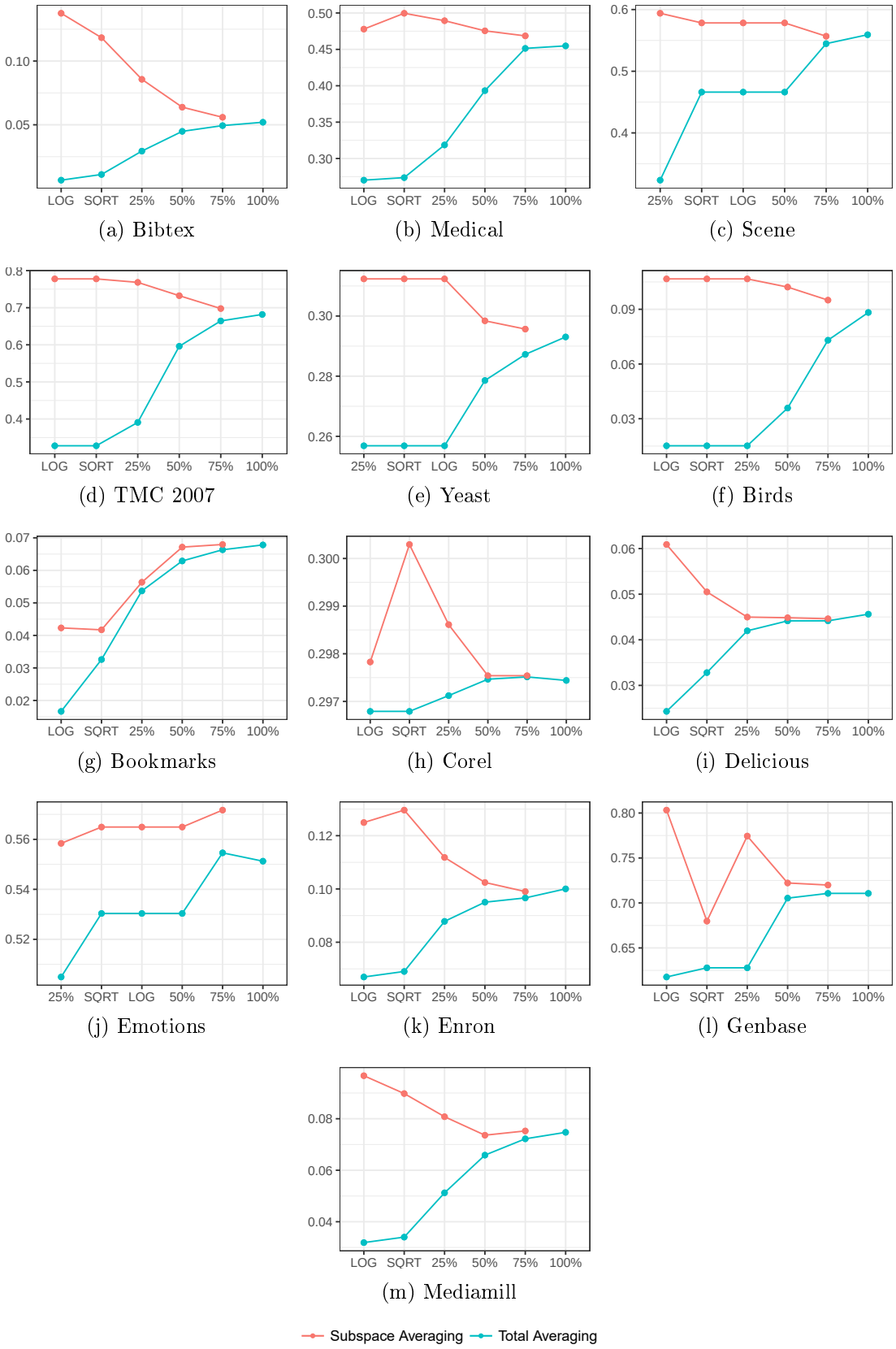



Figure B.8: RF-ROS performance in terms of macro-averaged recall.

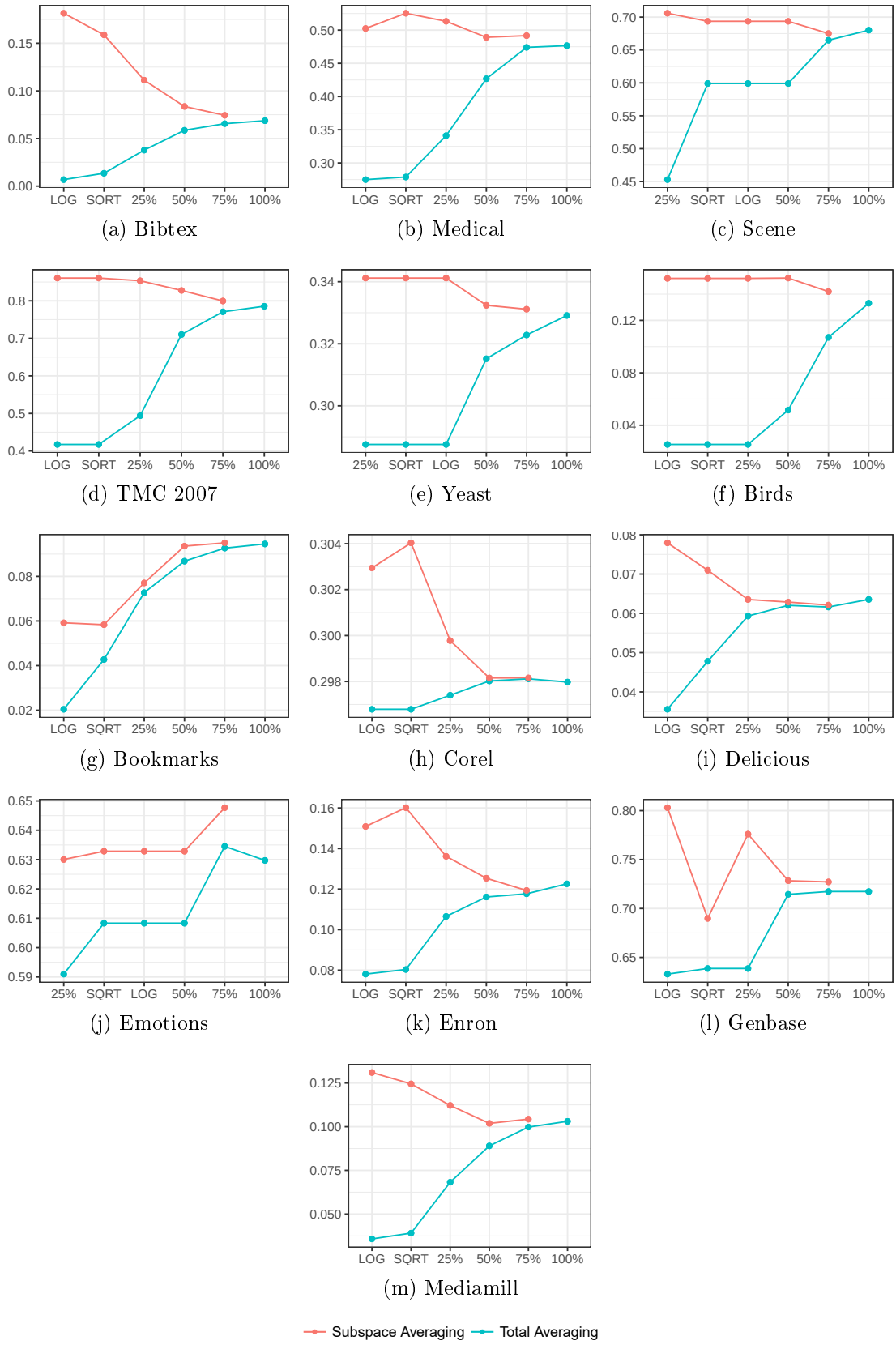
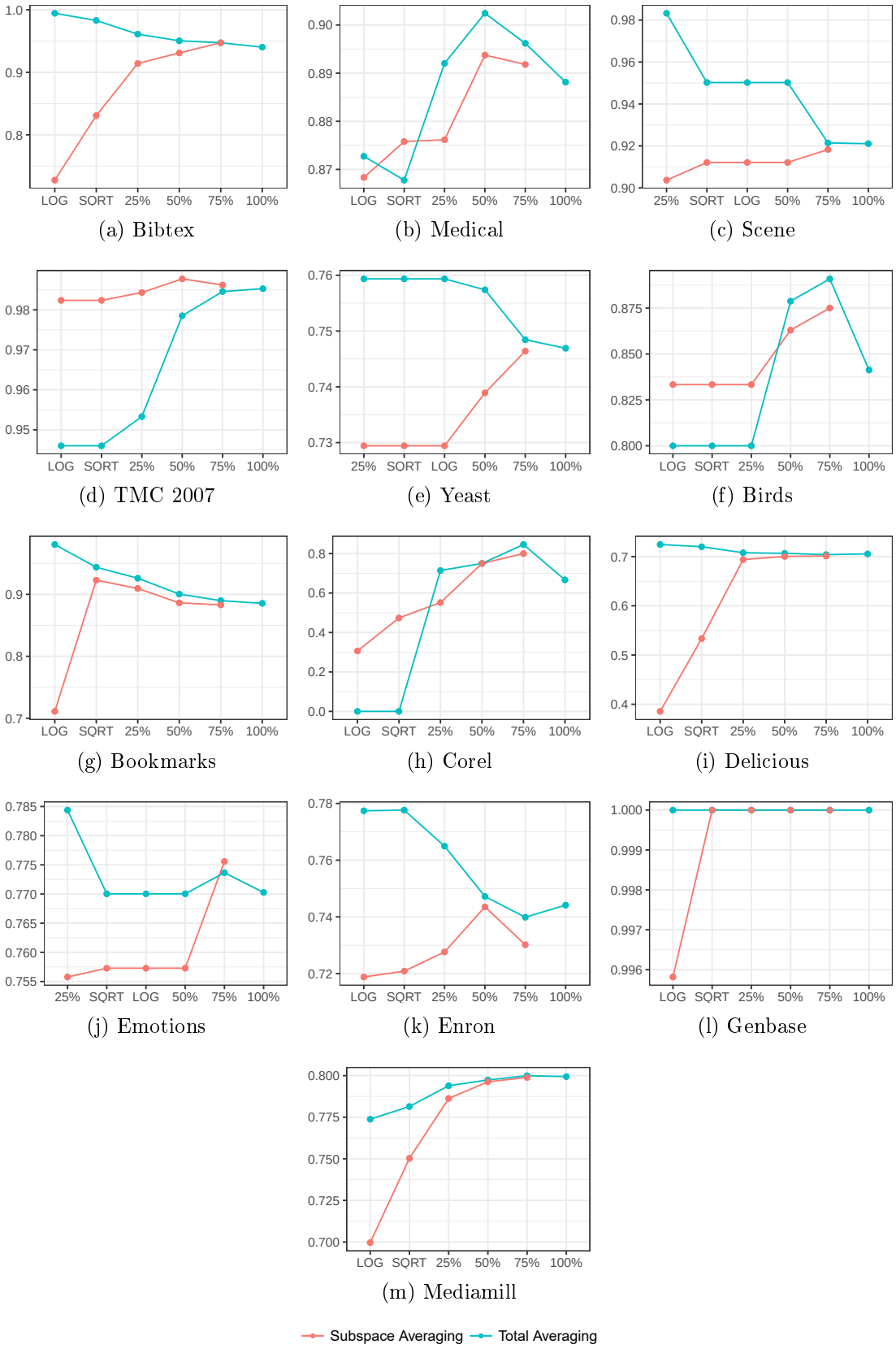



Figure B.9: RF-ROS performance in terms of macro-averaged $F_1$.




(a) Bibtex (b) Medical (c) Scene

(d) TMC 2007 (e) Yeast (f) Birds

(g) Bookmarks (h) Corel (i) Delicious

(j) Emotions (k) Enron (l) Genbase

(m) Mediamill

Figure B.10: RF-ROS performance in terms of micro-averaged precision.

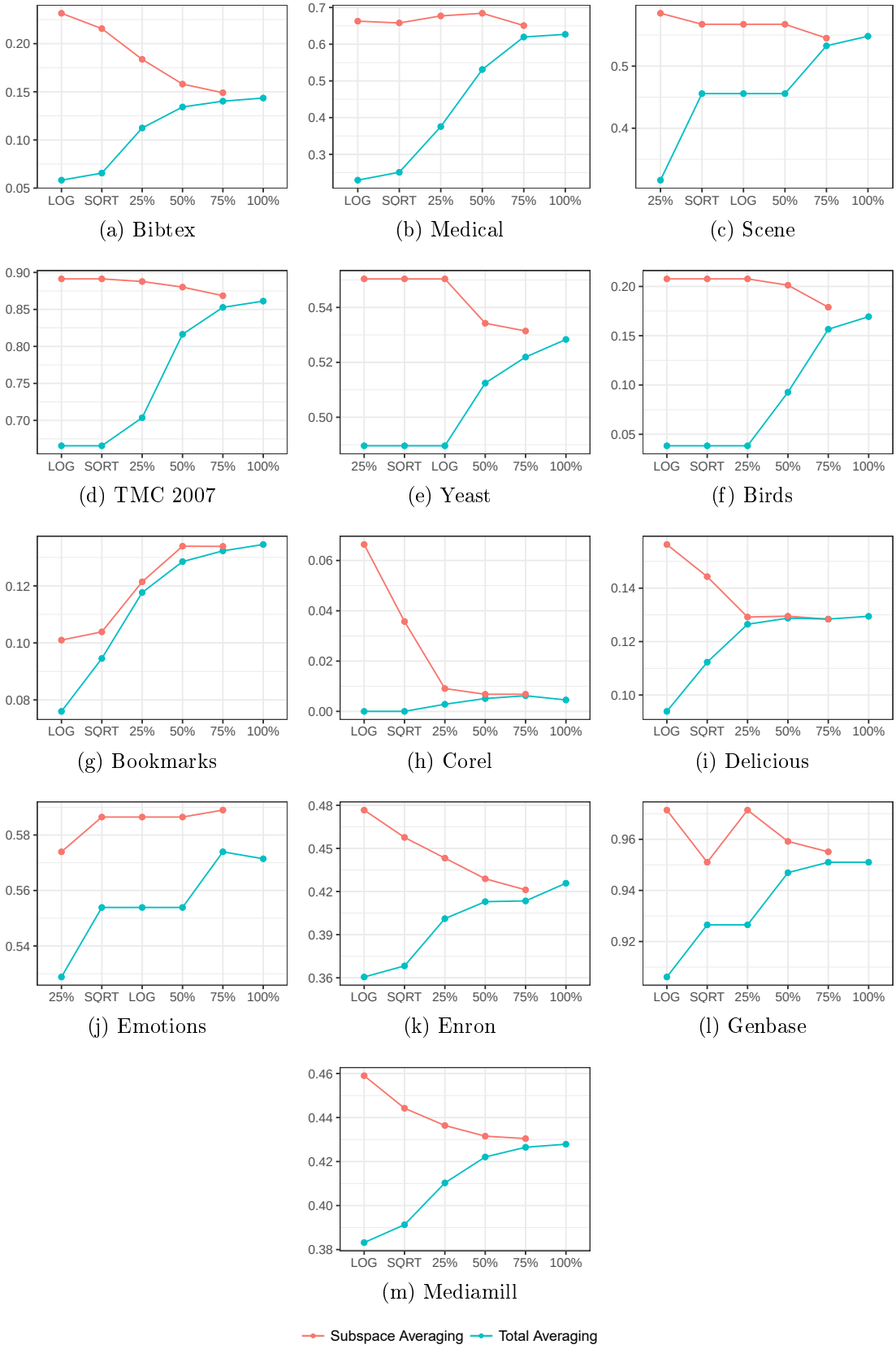



Figure B.11: RF-ROS performance in terms of micro-averaged recall.

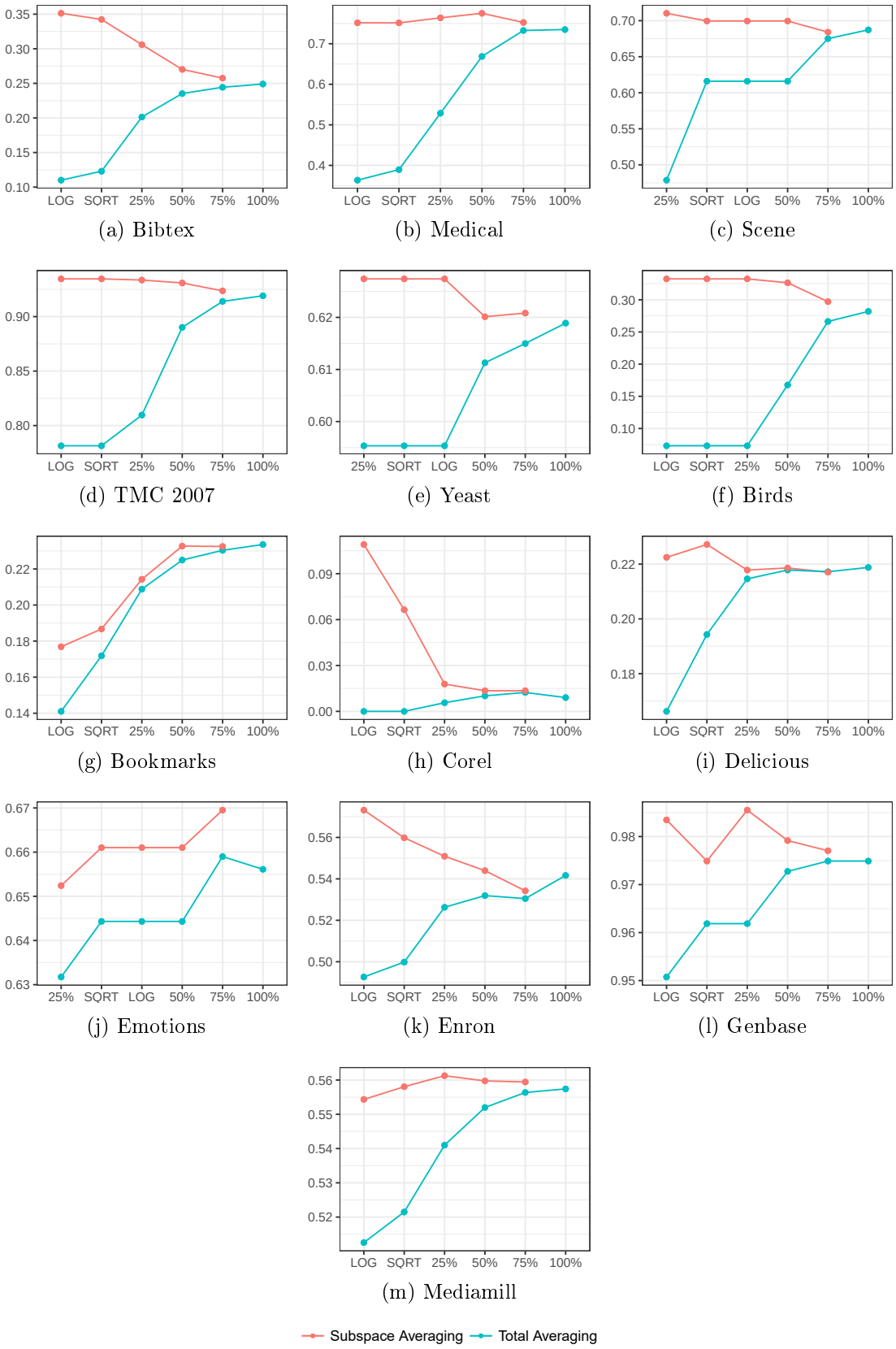



Figure B.12: RF-ROS performance in terms of micro-averaged $F_1$.

## B.2 Average Rank Diagrams for all Evaluation Measures

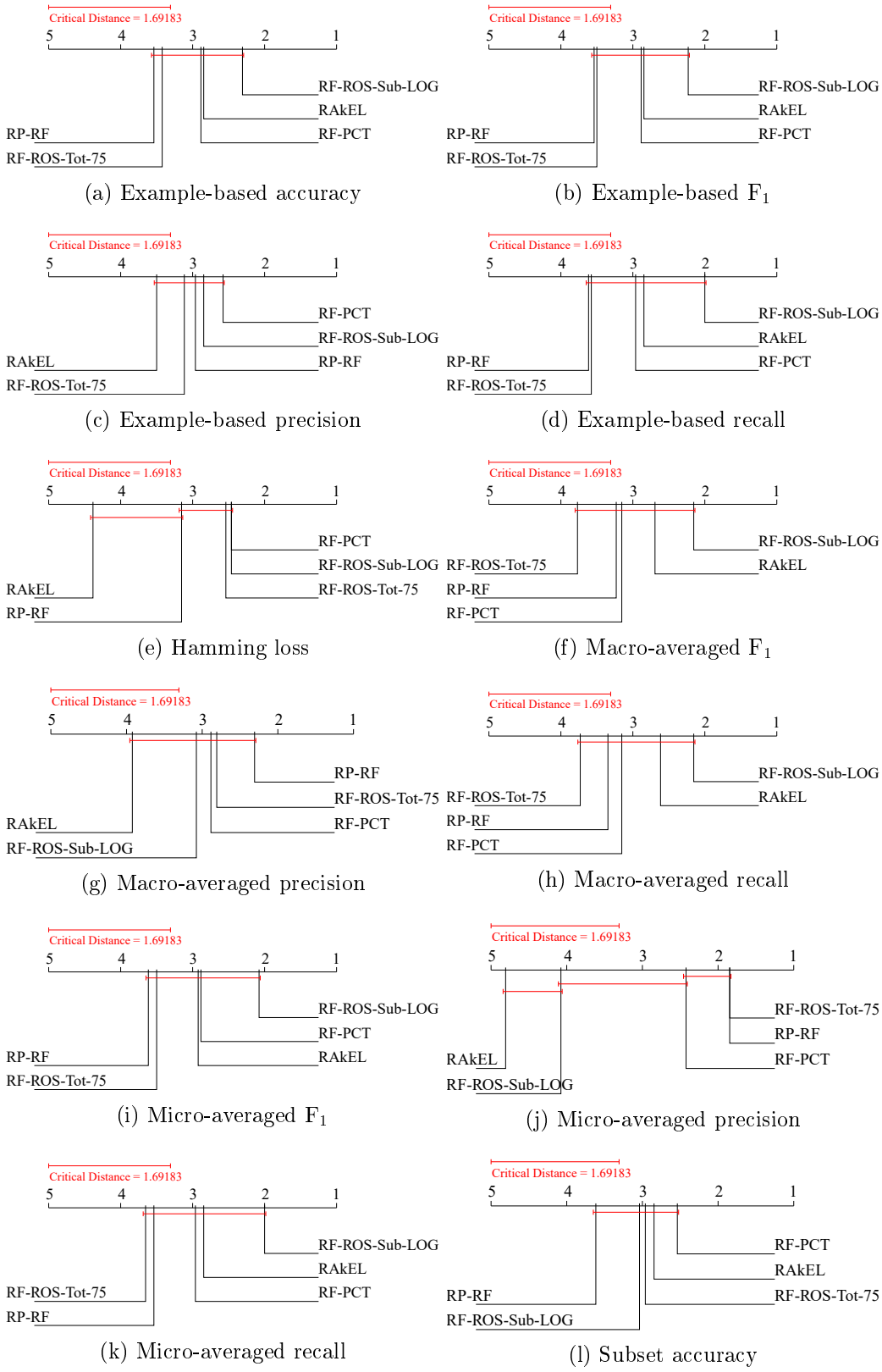


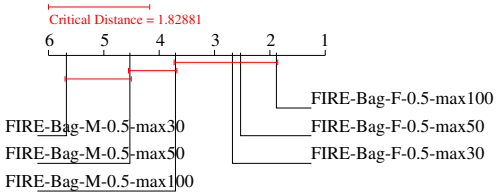

Figure B.13: Average rank diagrams for all evaluation measures. Lower ranks are better.

# Appendix C

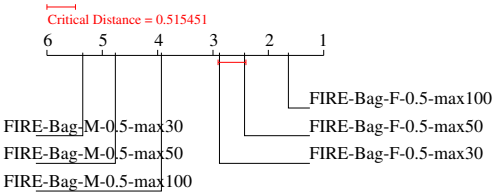
# Additional Details on the Evaluation of Rule Ensembles with Random Output Selections for Multi-Target Regression

## C.1 Average Rank Diagrams of Predictive Performance with Fixed ROS Subspace Size

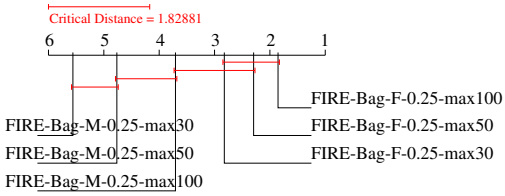
Below we provide figures that show the average rank diagrams of predictive performance of FIRE-ROS rule ensembles. The diagrams have been calculated on a per-dataset and per-target basis. For the per-dataset and per-target analyses, we used aRRMSE and RRMSE values, respectively. Each average rank diagram compares predictive performance of different-sized FIRE-ROS rule ensembles, while keeping the ROS subspace size the same for all sizes of ensembles. The value for the parameter $v$ is given in brackets in the caption of each average rank diagram. The individual ensemble in the average rank diagram is represented as such: FIRE-A-B-ROS-C-maxD, where A, B, C and D represent the ensemble method, rule type, ROS output space size and maximal number of rules, respectively. Ensemble methods are abbreviated as such: Bag (bagging of PCTs), RF (random forests of PCTs), ET (ensembles of extremely randomized PCTs). Rule types: F (fully-predicting rules) and M (mixed rules). ROS subspace size: 0.25 ($\frac{1}{4}$), 0.5 ($\frac{1}{2}$), 0.75 ($\frac{3}{4}$), Rnd (Random). Number of rules: 30, 50, 100 rules in the ensemble. Lower ranks are better.



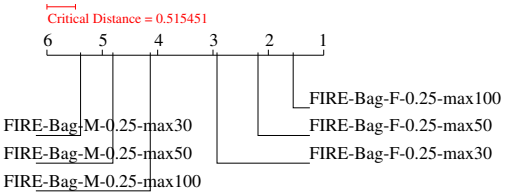

(a) Per dataset ($v = \frac{1}{2}$)



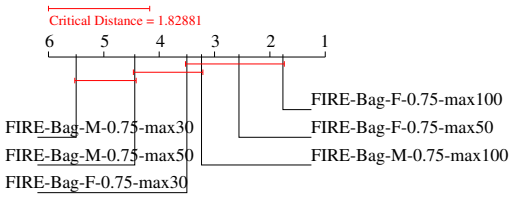

(b) Per target ($v = \frac{1}{2}$)



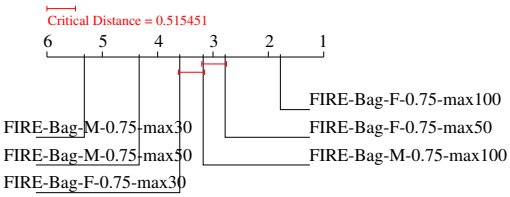

(c) Per dataset ($v = \frac{1}{4}$)



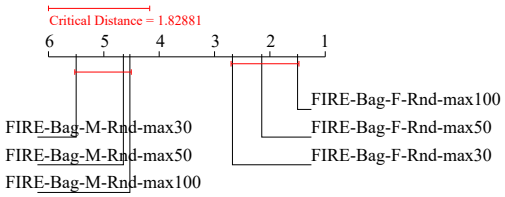

(d) Per target ($v = \frac{1}{4}$)



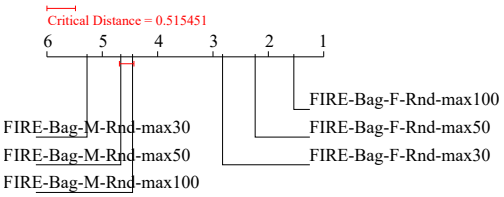

(e) Per dataset ($v = \frac{3}{4}$)




(f) Per target ($v = \frac{3}{4}$)

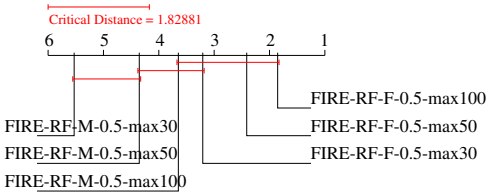



(g) Per dataset ($v =$ Random)

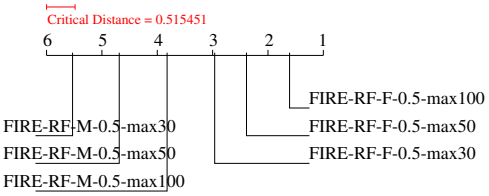



(h) Per target ($v =$ Random)

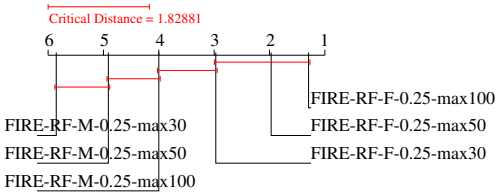
Figure C.1: FIRE-ROS performance in terms of aRRMSE (per dataset) and RRMSE (per target). The initial rules were generated with the bagging ensembles of PCTs.



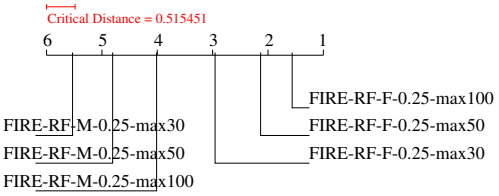

(a) Per dataset ($v = \frac{1}{2}$)



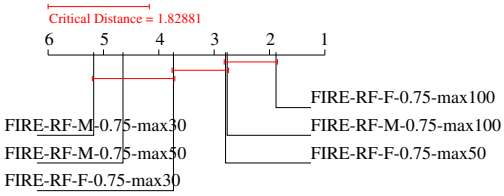

(b) Per target ($v = \frac{1}{2}$)



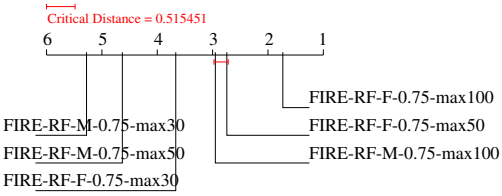

(c) Per dataset ($v = \frac{1}{4}$)



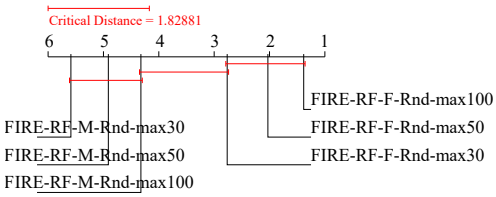

(d) Per target ($v = \frac{1}{4}$)



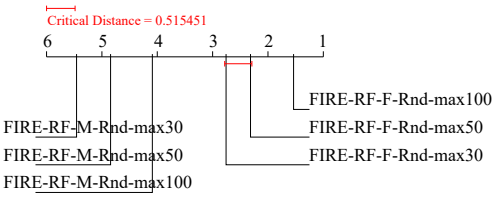

(e) Per dataset ($v = \frac{3}{4}$)




(f) Per target ($v = \frac{3}{4}$)

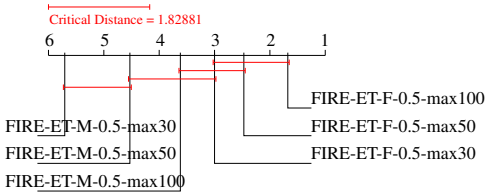



(g) Per dataset ($v =$ Random)

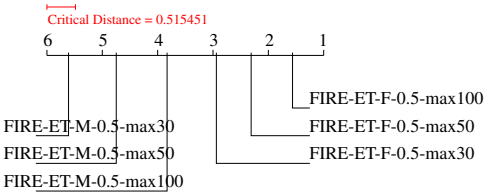



(h) Per target ($v =$ Random)

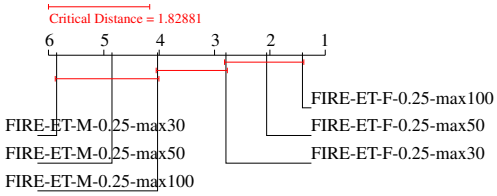
Figure C.2: FIRE-ROS performance in terms of aRRMSE (per dataset) and RRMSE (per target). The initial rules were generated with the random forest ensembles of PCTs.



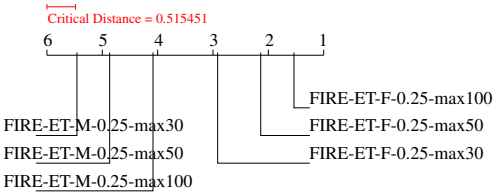

(a) Per dataset ($v = \frac{1}{2}$)



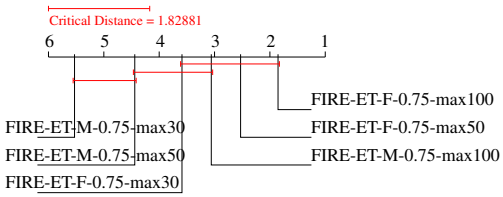

(b) Per target ($v = \frac{1}{2}$)



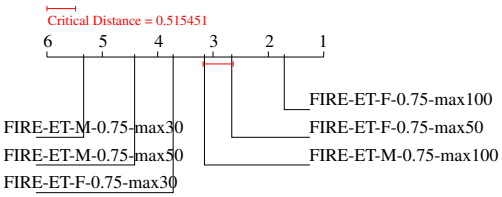

(c) Per dataset ($v = \frac{1}{4}$)



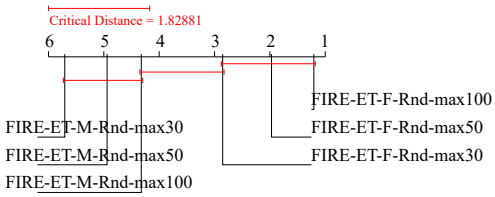

(d) Per target ($v = \frac{1}{4}$)



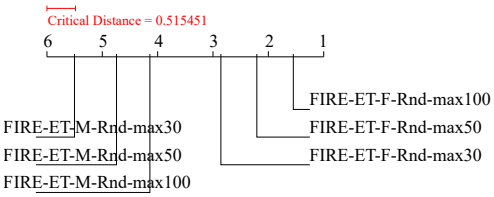

(e) Per dataset ($v = \frac{3}{4}$)




(f) Per target ($v = \frac{3}{4}$)




(g) Per dataset ($v =$ Random)




(h) Per target ($v =$ Random)

Figure C.3: FIRE-ROS performance in terms of aRRMSE (per dataset) and RRMSE (per target). The initial rules were generated with the ensembles of extremely randomized PCTs.

# References


Abraham, Z., Tan, P.-N., Winkler, J., Zhong, S., Liszewska, M., et al. (2013). Position preserving multi-output prediction. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2013)* (Vol. 8189, pp. 320–335). LNCS. doi:10.1007/978-3-642-40991-2_21

Aho, T., Ženko, B., & Džeroski, S. (2009). Rule ensembles for multi-target regression. In *Proceedings of the 9th IEEE International Conference on Data Mining* (pp. 21–30). IEEE. doi:10.1109/icdm.2009.16

Aho, T., Ženko, B., Džeroski, S., & Elomaa, T. (2012). Multi-Target Regression with Rule Ensembles. *Journal of Machine Learning Research*, *13*(Aug), 2367–2407.

Alali, A., & Kubat, M. (2015). PruDent: A pruned and confident stacking approach for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, *27*(9), 2480–2493. doi:10.1109/tkde.2015.2416731

Alvarez, M. A., Rosasco, L., Lawrence, N. D., et al. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, *4*(3), 195–266. doi:10.1561/2200000036

Appice, A., & Džeroski, S. (2007). Stepwise Induction of Multi-target Model Trees. In *Machine learning: ECML 2007* (Vol. 4701, pp. 502–509). LNCS. doi:10.1007/978-3-540-74958-5_46

Bakır, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (2007). *Predicting structured data*. Neural Information Processing. The MIT Press.

Barnard, K., Duygulu, P., Forsyth, D., Freitas, N. d., Blei, D. M., & Jordan, M. I. (2003). Matching words and pictures. *Journal of Machine Learning Research*, *3*(Feb), 1107–1135.

Bauer, E., & Kohavi, R. (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, *36*(1), 105–139.

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

Blockeel, H., & De Raedt, L. (1998). Top-down induction of first-order logical decision trees. *Artificial Intelligence*, *101*(1), 285–297. doi:10.1016/S0004-3702(98)00034-4

Blockeel, H., Raedt, L. D., & Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning* (pp. 55–63). Morgan Kaufmann.

Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A Survey on Multi-output Regression. *Wiley Int. Rev. Data Min. and Knowl. Disc. 5*(5), 216–233.

Boumghar, R., Lucas, L., & Donati, A. (2018). Machine Learning in Operations for the Mars Express Orbiter. In *15th international conference on space operations*. doi:10.2514/6.2018-2551

Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*(9), 1757–1771. doi:10.1016/j.patcog.2004.03.009

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140. doi:10.1023/a:1018054314350

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. doi:10.1023/a:1010933404324

Breiman, L., & Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *59*(1), 3–54. doi:10.1111/1467-9868.00054

Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Chapman & Hall/CRC.

Breskvar, M., Kocev, D., & Džeroski, S. (2017). Multi-label Classification Using Random Label Subset Selections. In *Proceedings of the 20th International Conference on Discovery Science* (pp. 108–115). Springer. doi:10.1007/978-3-319-67786-6_8

Breskvar, M., Kocev, D., & Džeroski, S. (2018a). Ensembles for multi-target regression with Random Output Selections. *Machine Learning*. doi:10.1007/s10994-018-5744-y

Breskvar, M., Kocev, D., & Džeroski, S. (2018b). Fitted rule ensembles for multi-target regression with Random Output Selections. In *ETAI 2018: Proceedings of abstracts of XIV international conference* (p. 25).

Breskvar, M., Kocev, D., & Džeroski, S. (under review). Multi-target regression rules with Random Output Selections. *Acta Polytechnica Hungarica*.

Breskvar, M., Kocev, D., Levatić, J., Osojnik, A., Petković, M., Simidjievski, N., ... Lucas, L. (2017). Predicting Thermal Power Consumption of the Mars Express Satellite with Machine Learning. In *Proceedings of the 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)* (pp. 88–93). IEEE. doi:10.1109/smc-it.2017.22

Breskvar, M., Ženko, B., & Džeroski, S. (2015). Relating Biological and Clinical Features of Alzheimer's Patients with Predictive Clustering Trees. In *Proceedings of the 18th International Information Society conference* (Vol. E, *15*, pp. 5–8).

Briggs, F., Huang, Y., Raich, R., Eftaxias, K., Lei, Z., Cukierski, W., ... Fern, X. Z., et al. (2013). The 9th annual MLSP competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *Machine learning for signal processing (mlsp), 2013 ieee international workshop on* (pp. 1–8). IEEE.

Brouwer, W. J., Kubicki, J. D., Sofo, J. O., & Giles, C. L. (2014). An investigation of machine learning methods applied to structure prediction in condensed matter. *arXiv preprint arXiv:1405.3564*. arXiv: 1405.3564

Brown, P. J., & Zidek, J. V. (1980). Adaptive multivariate ridge regression. *Annals of Statistics*, *8*(1), 64–74. doi:10.1214/aos/1176344891

Cai, F., & Cherkassky, V. (2009). SVM+ regression and multi-task learning. In *Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN 2009)* (pp. 418–424). doi:10.1109/ijcnn.2009.5178650

Chen, B., Li, W., Zhang, Y., & Hu, J. (2016). Enhancing multi-label classification based on local label constraints and classifier chains. In *Proceedings of the 7th International Joint Conference on Neural Networks* (pp. 1458–1463). doi:10.1109/ijcnn.2016.7727370

Chen, W.-J., Shao, Y.-H., Li, C.-N., & Deng, N.-Y. (2016). MLTSVM: A novel twin support vector machine to multi-label learning. *Pattern Recognition*, *52*, 61–74. doi:10.1016/j.patcog.2015.10.008

Cheng, W., & Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, *76*(2-3), 211–225. doi:10.1007/s10994-009-5127-5

Chicarro, A., Martin, P., & Trautner, R. (2004). The Mars Express mission: An overview. In *Mars express: The scientific payload* (Vol. 1240, pp. 3–13).

Chung, W., Kim, J., Lee, H., & Kim, E. (2015). General dimensional multiple-output support vector regressions and their multiple kernel learning. *IEEE Transactions on Cybernetics*, *45*(11), 2572–2584. doi:10.1109/tcyb.2014.2377016

Clare, A., & King, R. (2001). Knowledge discovery in multi-label phenotype data. In *Principles of Data Mining and Knowledge Discovery (PKDD 2001)* (Vol. 2168, pp. 42–53). LNCS. doi:10.1007/3-540-44794-6_4

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine learning*, *3*(4), 261–283.

Cover, T., & Hart, P. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27. doi:10.1109/tit.1967.1053964

De Comité, F., Gilleron, R., & Tommasi, M. (2003a). Learning multi-label alternating decision trees from texts and data. In *Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition* (pp. 35–49). Springer-Verlag Berlin, Heidelberg.

De Comité, F., Gilleron, R., & Tommasi, M. (2003b). Learning multi-label alternating decision trees from texts and data. In *Machine Learning and Data Mining in Pattern Recognition (MLDM 2003)* (Vol. 2734, pp. 35–49). LNCS. doi:10.1007/3-540-45065-3_4

De'Ath, G. (2002). Multivariate regression trees: A new technique for modeling species–environment relationships. *Ecology*, *83*(4), 1105–1117. doi:10.2307/3071917

Debeljak, M., Kocev, D., Towers, W., Jones, M., Griffiths, B., & Hallett, P. (2009). Potential of multi-objective models for risk-based mapping of the resilience characteristics of soils: Demonstration at a national level. *Soil Use and Management*, *25*(1), 66–77.

Demšar, D., Džeroski, S., Larsen, T., Struyf, J., Axelsen, J., Bruns-Pedersen, M., & Krogh, P. H. (2006). Using multi-objective classification to model communities of soil. *Ecological Modelling*, *191*(1), 131–143.

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

Diplaris, S., Tsoumakas, G., Mitkas, P. A., & Vlahavas, I. (2005). Protein classification with multiple algorithms. In *Panhellenic conference on informatics* (pp. 448–456). Springer.

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, *56*(293), 52–64.

Džeroski, S. (2007a). Towards a General Framework for Data Mining. In *Proceedings of the 5th international workshop on knowledge discovery in inductive databases kdid – lncs 4747* (pp. 259–300). Springer.

Džeroski, S. (2007b). Towards a general framework for data mining. In *Knowledge discovery in inductive databases: 5th international workshop, kdid 2006 berlin, germany, september 18, 2006 revised selected and invited papers* (pp. 259–300). Berlin, Heidelberg: Springer Berlin Heidelberg.

Džeroski, S., Demšar, D., & Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, *13*(1), 7–17.

Džeroski, S., Kobler, A., Gjorgjioski, V., & Panov, P. (2006). Using decision trees to predict forest stand height and canopy cover from LANSAT and LIDAR data. In *Managing environmental knowledge : Enviroinfo 2006 : Proceedings of the 20th interna-*

*tional conference on informatics for environmental protection* (pp. 125–133). Aachen: Shaker Verlag.

Elisseeff, A., & Weston, J. (2002). A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems 14 (NIPS 2001)* (pp. 681–687). NIPS Foundation.

Enrique Sucar, L., Bielza, C., Morales, E. F., Hernandez-Leal, P., Zaragoza, J. H., & Larrañaga, P. (2014). Multi-Label classification with Bayesian network-based chain classifiers. *Pattern Recognition Letters*, *41*, 14–22.

Freund, Y., & Mason, L. (1999). The alternating decision tree learning algorithm. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)* (pp. 124–133). doi:10.1.1.19.3673

Freund, Y., & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *Proceedings of the 13th international conference on machine learning - icml* (pp. 148–156). Morgan Kaufman.

Friedman, J. H., & Popescu, B. E. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 916–954. doi:10.1214/07-AOAS148

Friedman, J. H., Popescu, B. E. et al. (2003). Importance sampled learning ensembles. *Journal of Machine Learning Research*, *94305*, 1–32.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, *11*(1), 86–92. doi:10.1214/aoms/1177731944

Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, *73*(2), 133–153.

Gamberger, D., Ženko, B., Mitelpunkt, A., Shachar, N., & Lavrač, N. (2016). Clusters of male and female alzheimer's disease patients in the alzheimer's disease neuroimaging initiative (adni) database. *Brain Informatics*, *3*(3), 169–179.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42.

Gibaja, E., & Ventura, S. (2014). Multi-label learning: A review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *4*(6), 411–444.

Gillberg, J., Marttinen, P., Pirinen, M., Kangas, A. J., Soininen, P., Ali, M., ... Kaski, S. (2016). Multiple output regression with latent noise. *Journal of Machine Learning Research*, *17*(122), 1–35.

Gjorgjevikj, D., Madjarov, G., & Džeroski, S. (2013). Hybrid decision tree architecture utilizing local SVMs for efficient multi-label learning. *International Journal of Pattern Recognition and Artificial Intelligence*, *27*(7), 1351004. doi:10.1142/S021800141351004X

Gjorgjioski, V., Džeroski, S., & White, M. (2008). *Clustering analysis of vegetation data* (tech. rep. No. 10065). Jožef Stefan Institute.

Godbole, S., & Sarawagi, S. (2004). Discriminative Methods for Multi-labeled Classification. In H. Dai, R. Srikant, & C. Zhang (Eds.), *Advances in Knowledge Discovery and Data Mining* (pp. 22–30). Berlin, Heidelberg: Springer Berlin Heidelberg.

Gonçalves, E. C., Plastino, A., & Freitas, A. A. (2013). A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In *Proceeedings of the 2013 IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI 2013)* (pp. 469–476). doi:10.1109/ictai.2013.76

Gonçalves, T., & Quaresma, P. (2004). Using IR techniques to improve automated text classification. In *Natural Language Processing and Information Systems (NLDB 2004)* (Vol. 3136, pp. 374–379). LNCS. doi:10.1007/978-3-540-27779-8_34

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(10), 993–1001. doi:10.1109/34.58871

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67. doi:10.1080/00401706.1970.10488634

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. *Journal of Computational and Graphical Statistics*, *15*(3), 651–674. doi:10.1198/106186006x133933

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the Friedman statistic. *Communications in Statistics - Theory and Methods*, *9*(6), 571–595. doi:10.1080/03610928008827904

International Organization for Standardization. (2007). *Information technology – General-Purpose Datatypes (GPD)*. Standard. Retrieved from https://www.iso.org/standard/39479.html

Izenman, A. J. (1975). Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis*, *5*(2), 248–264.

Jayadeva, Khemchandani, R., & Chandra, S. (2007). Twin support vector machines for pattern classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(5), 905–910. doi:10.1109/tpami.2007.1068

Joly, A. (2017). *Exploiting random projections and sparsity with random forests and gradient boosting methods–Application to multi-label and multi-output learning, random forest model compression and leveraging input sparsity* (Doctoral dissertation, Faculty of Applied Sciences, Department of Electrical Engineering & Computer Science, University of Liège, Liège, Belgium).

Joly, A., Geurts, P., & Wehenkel, L. (2014). Random forests with random projections of the output space for high dimensional multi-label classification. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 607–622). Springer.

Kaggle. (2008). Kaggle competition: Online product sales. https://www.kaggle.com/c/online-sales/data. [Online; accessed 19-July-2017].

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. *ECML PKDD discovery challenge*, *75*.

Klimt, B., & Yang, Y. [Yiming]. (2004). The Enron Corpus: A New Dataset for Email Classification Research. In *Proceedings of the 18th European Conference on Machine Learning – LNCS 3201* (pp. 217–226). Springer.

Kocev, D. (2011). *Ensembles for Predicting Structured Outputs* (PhD Thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia).

Kocev, D., & Ceci, M. (2015). Ensembles of Extremely Randomized Trees for Multi-target Regression. In *Proceedings of the 18th International Conference on Discovery Science* (Vol. 9356, pp. 86–100). LNCS. doi:10.1007/978-3-319-24282-8_9

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., Naumoski, A., Mitreski, K., Krstić, S., & Džeroski, S. (2010). Learning habitat models for the diatom community in lake Prespa. *Ecological Modelling*, *221*(2), 330–337.

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, *46*(3), 817–833. doi:10.1016/j.patcog.2012.09.023

Kriegel, H.-P., Borgwardt, K., Kröger, P., Pryakhin, A., Schubert, M., & Zimek, A. (2007). Future trends in data mining. *Data Mining and Knowledge Discovery*, *15*, 87–97.

Kuncheva, L. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience.

Langley, P. (1996). *Elements of machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2014). Semi-supervised learning for multi-target regression. In *International Workshop on New Frontiers in Mining Complex Patterns* (pp. 3–18). Springer. doi:10.1007/978-3-319-17876-9_1

Levatic, J., Kocev, D., & Džeroski, S. (2014). The importance of the label hierarchy in hierarchical multi-label classification. *Journal of Intelligent Information Systems*, *45*, 247–271.

Liu, C., & Cao, L. (2015). A Coupled $k$-Nearest Neighbor Algorithm for Multi-label Classification. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2015)* (Vol. 9077, pp. 176–187). LNCS. doi:10.1007/978-3-319-18038-0_14

Loza Mencía, E., & Janssen, F. (2016). Learning rules for multi-label classification: A stacking and a separate-and-conquer approach. *Machine Learning*, *105*(1), 77–126.

Lucas, L., & Boumghar, R. (2017). Machine Learning for Spacecraft Operations Support-The Mars Express Power Challenge. In *Proceedings of the 6th international conference on space mission challenges for information technology (smc-it)* (pp. 82–87). IEEE. doi:10.1109/SMC-IT.2017.21

Madjarov, G., & Gjorgjevikj, D. (2012). Hybrid decision tree architecture utilizing local SVMs for multi-label classification. In *Hybrid Artificial Intelligent Systems (HAIS 2012)* (Vol. 7209, pp. 1–12). LNCS. doi:10.1007/978-3-642-28931-6_1

Madjarov, G., Gjorgjevikj, D., Dimitrovski, I., & Džeroski, S. (2016). The use of data-derived label hierarchies in multi-label classification. *Journal of Intelligent Information Systems*, *47*(1), 57–90. doi:10.1007/s10844-016-0405-8

Madjarov, G., Gjorgjevikj, D., & Džeroski, S. (2012). Two stage architecture for multi-label learning. *Pattern Recognition*, *45*(3), 1019–1034. doi:10.1016/j.patcog.2011.08.011

Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, *45*(9), 3084–3104. doi:10.1016/j.patcog.2012.03.004

Marek, K., Jennings, D., Lasch, S., Siderowf, A., Tanner, C., Simuni, T., ... Chowdhury, S., et al. (2011). The parkinson progression marker initiative (PPMI). *Progress in neurobiology*, *95*(4), 629–635. doi:10.1016/j.pneurobio.2011.09.005

Mileski, V., Džeroski, S., & Kocev, D. (2017). Predictive Clustering Trees for Hierarchical Multi-Target Regression. In *International Symposium on Intelligent Data Analysis* (pp. 223–234). Springer. doi:10.1007/978-3-319-68765-0_19

Mitchell, T. (1997). *Machine learning*. doi:10.1002/(SICI)1099-1689(199909)9:3<191::AID-STVR184>3.0.CO;2-E

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons* (Doctoral dissertation, Princeton University, Princeton, NY, USA).

Orosei, R., Lauro, S. E., Pettinelli, E., Cicchetti, A., Coradini, M., Cosciotti, B., ... Seu, R. (2018). Radar evidence of subglacial liquid water on Mars. *Science*, *361*, 490–493. doi:10.1126/science.aar7268

Al-Otaibi, R., Kull, M., & Flach, P. (2014). LaCova: A tree-based multi-label classifier using label covariance as splitting criterion. In *Proceedings of the 13th International Conference on Machine Learning and Applications (ICMLA 2014)* (pp. 74–79). doi:10.1109/icmla.2014.17

Al-Otaibi, R., Kull, M., & Flach, P. A. (2016). Declaratively Capturing Local Label Correlations with Multi-Label Trees. In *ECAI 2016* (Vol. 285, pp. 1467–1475). FAIA. doi:10.3233/978-1-61499-672-9-1467

Panov, P., Soldatova, L. N., & Džeroski, S. (2016). Generic ontology of datatypes. *Information Sciences*, *329*(Supplement C), 900–920. doi:10.1016/j.ins.2015.08.006

Pestian, J. P., Brew, C., Matykiewicz, D. J., Paweł and Hovermale, Johnson, N., Cohen, K. B., & 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* (pp. 97–104). BioNLP '07. doi:10.3115/1572392.1572411

Petković, M., Boumghar, R., Breskvar, M., Džeroski, S., Kocev, D., Levatić, J., ... Simidjievski, N. (2019). Machine learning for predicting thermal power consumption of the Mars Express Spacecraft. *IEEE Aerospace and Electronics Systems Magazine*.

Petković, M., Džeroski, S., & Kocev, D. (2017). Feature ranking for multi-target regression with tree ensemble methods. In *Proceedings of the 20th International Conference on Discovery Science* (Vol. 10558, pp. 171–185). LNCS. Springer. doi:10.1007/978-3-319-67786-6_13

Petković, M., Džeroski, S., & Kocev, D. (2018). Feature ranking for hierarchical multi-label classification with tree ensemble methods. In *Etai 2018: Proceedings of abstracts of xiv international conference* (p. 58).

Pugelj, M., & Džeroski, S. (2011). Predicting structured outputs k-nearest neighbours method. In *International Conference on Discovery Science* (pp. 262–276). Springer.

Read, J. (2008). A pruned problem transformation method for multi-label classification. In *Proceedings of the 6th New Zealand Computer Science Research Student Conference* (pp. 143–150).

Read, J., Martino, L., Olmos, P. M., & Luengo, D. (2015). Scalable multi-output label prediction: From classifier chains to classifier trellises. *Pattern Recognition*, *48*(6), 2096–2109. doi:10.1016/j.patcog.2015.01.004

Read, J., Pfahringer, B., & Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE International Conference on Data Mining* (pp. 995–1000). doi:10.1109/icdm.2008.74

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2009). Classifier Chains for Multi-label Classification. In *Machine learning and knowledge discovery in databases - lncs 5782* (pp. 254–269). Springer.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. doi:10.1038/323533a0

Russell, S., & Norvig, P. (2009). *Artificial intelligence: A modern approach* (3rd). Prentice Hall Press.

Sánchez-Fernández, M., de-Prado-Cumplido, M., Arenas-García, J., & Pérez-Cruz, F. (2004). SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE Transactions on Signal Processing*, *52*(8), 2298–2307. doi:10.1109/tsp.2004.831028

Schapire, R. E., & Singer, Y. (2000). BoosTexter: A boosting-based system for text categorization. *Machine Learning*, *39*(2-3), 135–168. doi:10.1023/a:1007649029923

Snoek, C. G., Worring, M., Van Gemert, J. C., Geusebroek, J.-M., & Smeulders, A. W. (2006). 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.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., & Vlahavas, I. (2012). Multi-label classification methods for multi-target regression. *arXiv preprint arXiv:1211.6581*, 1159–1168.

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. doi:10.1007/s10994-016-5546-z

Spyromitros, E., Tsoumakas, G., & Vlahavas, I. (2008). An empirical study of lazy multilabel classification algorithms. In *Artificial Intelligence: Theories, Models and Applications (SETN 2008)* (Vol. 5138, pp. 401–406). LNCS. doi:10.1007/978-3-540-87881-0_40

Srivastava, A. N., & Zane-Ulman, B. (2005). Discovering recurring anomalies in text reports regarding complex space systems. In *Aerospace conference, 2005 ieee* (pp. 3853–3862). IEEE. doi:10.1109/aero.2005.1559692

Struyf, J., & Džeroski, S. (2006). Constraint Based Induction of Multi-Objective Regression Trees. In *Proceedings of the 4th International Workshop on Knowledge Discovery in Inductive Databases KDID - LNCS 3933* (pp. 222–233). Springer.

Su, H., & Rousu, J. (2015). Multilabel classification through random graph ensembles. *Machine Learning*, *99*(2), 231–256. doi:10.1007/s10994-014-5465-9

Szymański, P., Kajdanowicz, T., & Kersting, K. (2016). How is a data-driven approach better than random choice in label space division for multi-label classification? *Entropy*, *18*(8), 282.

Trohidis, K., Tsoumakas, G., Kalliris, G., & Vlahavas, I. (2008). Multilabel Classification of Music into Emotions. In *Proceedings of the 9th international conference on music information retrieval (ismir 2008)* (pp. 325–330).

Tsoumakas, G., & Katakis, I. (2007). Multi label classification: An overview. *International Journal of Data Warehouse and Mining*, *3*(3), 1–13. doi:10.4018/jdwm.2007070101

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2008). Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *Proceedings of the ecml/pkdd workshop on mining multidimensional data* (pp. 30–44).

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2009). Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook* (pp. 667–685). Springer US.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., & Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, *12*, 2411–2414.

Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., & Vlahavas, I. (2014). Multi-target Regression via Random Linear Target Combinations. In *Machine Learning and Knowledge Discovery in Databases: ECML-PKDD 2014* (Vol. 8726, pp. 225–240). LNCS.

Tsoumakas, G., & Vlahavas, I. (2007). Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *Proceedings of the 18th european conference on machine learning* (pp. 406–417).

Van Wolputte, E., Korneva, E., & Blockeel, H. (2018). MERCS: multi-directional ensembles of regression and classification trees. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*.

Vazquez, E., & Walter, E. (2003). Multi-output suppport vector regression. *IFAC Proceedings Volumes*, *36*(16), 1783–1788. doi:10.1016/s1474-6670(17)35018-8

Wang, S., Wang, J., Wang, Z., & Ji, Q. (2014). Enhancing multi-label classification by modeling dependencies among labels. *Pattern Recognition*, *47*(10), 3405–3413. doi:10.1016/j.patcog.2014.04.009

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. doi:10.1186/1475-925X-5-51

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(1), 241–259. doi:10.1016/S0893-6080(05)80023-1

Wu, Q., Tan, M., Song, H., Chen, J., & Ng, M. K. (2016). ML-FOREST: A multi-label tree ensemble method for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, *28*(10), 2665–2680. doi:10.1109/tkde.2016.2581161

Wu, Q., Ye, Y., Zhang, H., Chow, T. W. S., & Ho, S.-S. (2015). ML-TREE: A tree-structure-based approach to multilabel learning. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(3), 430–443. doi:10.1109/tnnls.2014.2315296

Xu, J. (2014). Multi-label core vector machine with a zero label. *Pattern Recognition*, *47*(7), 2542–2557. doi:10.1016/j.patcog.2014.01.012

Xu, S., An, X., Qiao, X., Zhu, L., & Li, L. (2013). Multi-output least-squares support vector regression machines. *Pattern Recognition Letters*, *34*(9), 1078–1084. doi:10.1016/j.patrec.2013.01.015

Yang, Q., & Wu, X. (2006). 10 Challenging Problems in Data Mining Research. *International Journal of Information Technology and Decision Making*, *5*(4), 597–604. doi:10.1142/S0219622006002258

Yang, Y. [Yanyan], Chen, D., & Dong, Z. (2015). Novel Multi-output Support Vector Regression Model via Double Regularization. In *Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2015)* (pp. 2697–2701). doi:10.1109/smc.2015.471

Yeh, C.-K., Wu, W.-C., Ko, W.-J., & Wang, Y.-C. F. (2017). Learning Deep Latent Space for Multi-Label Classification. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence and the 29th Innovative Applications of Artificial Intelligence Conference (AAAI 2017)* (Vol. 4, pp. 2838–2844). AAAI.

Ženko, B. (2007). *Learning predictive clustering rules* (Doctoral dissertation, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia).

Zhang, M.-L. (2009). Ml-Rbf: RBF neural networks for multi-label learning. *Neural Processing Letters*, *29*(2), 61–74. doi:10.1007/s11063-009-9095-3

Zhang, M.-L., & Zhou, Z.-H. (2005). A $k$-nearest neighbor based algorithm for multi-label classification. In *Proceedings of the 2005 IEEE International Conference on Granular Computing (GrC 2005)* (Vol. 2, pp. 718–721). doi:10.1109/grc.2005.1547385

Zhang, M.-L., & Zhou, Z.-H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, *18*(10), 1338–1351. doi:10.1109/tkde.2006.162

Zhang, M.-L., & Zhou, Z.-H. (2007). ML-kNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, *40*(7), 2038–2048. doi:10.1016/j.patcog.2006.12.019

Zhang, M.-L., & Zhou, Z.-H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837. doi:10.1109/tkde.2013.39

Zhang, W., Liu, X., Ding, Y., & Shi, D. (2012). Multi-output LS-SVR machine in extended feature space. In *Proceedings of the 2012 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 2012)* (pp. 130–134). doi:10.1109/cimsa.2012.6269600

Zhou, T., Tao, D., & Wu, X. (2012). Compressed labeling on distilled labelsets for multi-label learning. *Machine Learning*, *88*(1-2), 69–126. doi:10.1007/s10994-011-5276-1

# Bibliography

## Publications Related to the Thesis

### Journal Articles

Breskvar, M., Kocev, D., & Džeroski, S. (2018a). Ensembles for multi-target regression with Random Output Selections. *Machine Learning*. doi:10.1007/s10994-018-5744-y

Breskvar, M., Kocev, D., & Džeroski, S. (under review). Multi-target regression rules with Random Output Selections. *Acta Polytechnica Hungarica*.

### Conference Papers

Breskvar, M., Kocev, D., & Džeroski, S. (2017). Multi-label Classification Using Random Label Subset Selections. In *Proceedings of the 20th International Conference on Discovery Science* (pp. 108–115). Springer. doi:10.1007/978-3-319-67786-6_8

Breskvar, M., Kocev, D., & Džeroski, S. (2018b). Fitted rule ensembles for multi-target regression with Random Output Selections. In *ETAI 2018: Proceedings of abstracts of XIV international conference* (p. 25).

## Other Publications

### Journal Articles

Horvat, S., Mahnič, A., Breskvar, M., Džeroski, S., & Rupnik, M. (2017). Evaluating the effect of Clostridium difficile conditioned medium on fecal microbiota community structure. *Scientific reports*, *7*(1), 16448.

Mahnič, A., Breskvar, M., Džeroski, S., Skok, P., & Rupnik, M. (under review). Bacterial and fungal gut microbiota in IBD and non-IBD hospitalized gastroenterological patients. *Microbiome*.

Petković, M., Boumghar, R., Breskvar, M., Džeroski, S., Kocev, D., Levatić, J., ... Simidjievski, N. (2019). Machine learning for predicting thermal power consumption of the Mars Express Spacecraft. *IEEE Aerospace and Electronics Systems Magazine*.

### Conference Papers

Breskvar, M., Kocev, D., Levatić, J., Osojnik, A., Petković, M., Simidjievski, N., ... Lucas, L. (2017). Predicting Thermal Power Consumption of the Mars Express Satellite with Machine Learning. In *Proceedings of the 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)* (pp. 88–93). IEEE. doi:10.1109/smc-it.2017.22

Breskvar, M., Ženko, B., & Džeroski, S. (2015). Relating Biological and Clinical Features of Alzheimer's Patients with Predictive Clustering Trees. In *Proceedings of the 18th International Information Society conference* (Vol. E, *15*, pp. 5–8).

# Biography

The author of this dissertation was born in Slovenia, where he finished his primary and secondary education. In 2003, he started his studies at the Faculty of Computer and Information Science at the University of Ljubljana, Slovenia. He defended his bachelor thesis entitled "Synchronization of data sources with Microsoft Sync framework" under the supervision of Asst. Prof. Dr. Matija Marolt in 2011.

In the fall of 2014, he enrolled in the "Information and Communication Technologies" PhD programme at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, under the supervision of Prof. Dr. Sašo Džeroski and co-supervision of Dr. Dragi Kocev. The author of this dissertation holds a young researcher grant for doctoral studies awarded by the Slovenian Research Agency. He works at the Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia, where he has been involved in the EU-funded FP7 project MAESTRA (Learning from Massive, Incompletely annotated, and Structured Data) and FP7 and H2020 project HBP (The Human Brain Project).

His research interests are in the area of machine learning, more specifically in the area of ensemble learning for structured output prediction. He has published several journal and conference papers and has presented his work at international conferences and project meetings.