STRUCTURED OUTPUT PREDICTION ON DATA STREAMS

Aljaž Osojnik

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

Supervisor: Prof. Dr. Sašo Džeroski, IPS and Jožef Stefan Institute, Ljubljana, Slovenia
Co-Supervisor: Asst. Prof. Dr. Panče Panov, Jožef Stefan Institute, Ljubljana, Slovenia

Evaluation Board:
Asst. Prof. Dr. Bernard Ženko, Chair, IPS and Jožef Stefan Institute, Ljubljana, Slovenia
Assoc. Prof. Dr. Zoran Bosnić, Member, Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia
Prof. Dr. Albert Bifet, Member, Institut Mines-Télécom, Télécom ParisTech, Université Paris-Saclay, Paris, France

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Aljaž Osojnik

STRUCTURED OUTPUT PREDICTION ON DATA STREAMS

Doctoral Dissertation

NAPOVEDOVANJE STRUKTURIRANIH VREDNOSTI NA PODATKOVNIH TOKOVIH

Doktorska disertacija

Supervisor: Prof. Dr. Sašo Džeroski

Co-Supervisor: Asst. Prof. Dr. Pančе Panov

Ljubljana, Slovenia, March 2017

For Mojca.
Without her, this would never have come to be.

Acknowledgments

> We are like dwarfs on the shoulders of giants, so that we can see more than they, and things at a greater distance, not by virtue of any sharpness of sight on our part, or any physical distinction, but because we are carried high and raised up by their giant size.
>
> — Bernard of Chartres

Bernard of Chartres' quote alludes to the notion that we discover and expand new ideas because the knowledge of discoveries made by the eponymous giants, discoverers of the past, is at our disposal. Nowhere is this more true than in the scientific pursuit.

This work was primarily made possible due to the mentorship of my supervisor Prof. Sašo Džeroski. Sašo, you provided the bedrock that this thesis was built on. You always know how to further guide and motivate my research and I look forward to collaborating with you in the future.

I thank my co-supervisor Asst. Prof. Panče Panov for always having time for my questions and for helping make my ramblings coherent. Panče, thank you for your patience with my writing and thank you for helping me refine the thesis from a rough draft to the final text, polished to a gleam.

Furthermore, I thank the members of the evaluation board, Asst. Prof. Bernard Ženko, Assoc. Prof. Zoran Bosnić and Prof. Albert Bifet for taking the time to read and evaluate my thesis. Your invaluable comments and suggestions have considerably improved this work.

This thesis is the final result of the support of the Slovenian Research Agency given through a Young Researcher grant. The work of this thesis was also partially supported 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).

This thesis is also a tribute to the teachers that perch us atop the giants' shoulders. Teachers, beyond those already mentioned above, whose immense ability to impart knowledge and inspire, inspired me to pursue an academic career and produce this work.

Aleš Sojar, my high-school biology teacher, instilled in me the rigor of the scientific method, be it through his thunderous lectures or the cubes of agar we experimented on.

Olga Arnuš always had an answer for all my high-school questions about mathematics, whether immediately or after consulting her library, and taught me that the wisest people, in particular, seek wisdom in the knowledge of others. It was her enthusiasm for mathematics that ultimately tipped my choice from studying computer science toward studying mathematics.

Prof. Andrej Bauer has taught me many things, the least of which were taught during his many lectures. His unbridled enthusiasm for teaching is extremely contagious, and has inspired me to pass my knowledge to others. And, incidentally, it was he who had introduced me to Prof. Džeroski.

However, scientific work is made possible not only by knowledge of our mentors and teachers, but also by the care and support of our family, friends and colleagues.

I am grateful to my officemates Bernard, Panče, Martin, and officemate in absentia Tea, for the endless discussions on topics both related and unrelated to this thesis. Special thanks go to Tea, who helped wrangle the LaTeX beast you see before you.

The colleagues in our research group have provided countless hours of discussion of the proper scientific methodology at the group meetings, and countless hours of entertainment during our many joint travels. Thank you, Dragi, Jurica, Jovan, Ljupčo, (the other) Martin, Matej, Nikola, Tomaž, and the rest of our group, and I apologize for my incessant questions during our group meetings!

A special thanks goes to Mili who continues to make my stay at the Jožef Stefan Institute a pleasant one and who always has just the right solution for each problem.

I am grateful to my sister Kaja and my brothers Benjamin and Jernej for their endless support. Growing up with you has brought me to this point and I am grateful that you were by my side on this journey. Melita and Jani, you were my first teachers that nurtured and encouraged my curiosity from an early age. I am eternally grateful for your unwavering support in whatever I do. Finally, thank you, Mojca, for always supporting me and pushing me to strive for higher and higher goals. Were it not for you, this thesis and many other wonderful things would likely never have come to be.

Abstract

In this thesis, we deal with structured output prediction (SOP) on data streams. SOP is concerned with learning predictive models that can predict structured outputs, i.e., outputs that are composed from multiple component values. On the other hand, data stream mining is concerned with learning in the online setting, where new data examples arrive at high frequencies and a predictive model is expected to operate in real-time.

Methods for SOP are now common in the classical batch data mining setting. However, they are fairly rare in the online setting, where the most addressed SOP tasks are multi-target regression (MTR) and multi-label classification (MLC). In MTR, we predict multiple real-valued targets, while in MLC we predict the presence or absence of predefined labels. Methods for online SOP tasks are typically specific to the SOP task that they address.

In this thesis, we introduce the incremental Structured Output Prediction Tree (iSOUP-Tree) family of methods that are designed to address multiple SOP tasks in the online learning setting. We introduce tree-based methods for online MTR: iSOUP-Tree, a method that uses the Hoeffding inequality and an intra-cluster variance reduction heuristic to grow the tree, iSOUP-OptionTree, a method that extends iSOUP-Tree toward option trees, online bagging of iSOUP-Trees and online random forests of iSOUP-Trees.

We continue by introducing the MLC via MTR problem transformation methodology that transforms an online MLC task into an online MTR task. We use the introduced methods for online MTR in combination with this methodology to address online MLC.

We then extend the iSOUP-Tree method to handle hierarchical prediction tasks, hierarchical MTR and hierarchical MLC where a hierarchy of the target variables is given. To this end, we adapt a batch method that introduces feature weights in the splitting heuristic of iSOUP-Tree where the weights are based on the target hierarchy.

We continue by extending iSOUP-Trees into the predictive clustering framework and introduce iSOUP predictive clustering trees (iSOUP-PCTs). We use iSOUP-PCTs for online semi-supervised MTR, where we learn from incompletely annotated examples as well as from completely annotated examples that are used in regular, supervised learning.

We also address the task of online feature ranking for SOP tasks. Feature ranking is concerned with identifying the input attributes that are most crucial for learning models with good predictive performance. We adapt the symbolic random forest feature ranking method from the batch to the online setting by using online random forests of iSOUP-Trees.

We evaluate the introduced methods through experiments suited to the SOP tasks we address. For online MTR, we find that bagging of iSOUP-Trees has the best performance, while random forests of iSOUP-Trees provide a good trade-off between predictive performance and consumption of computational resources. For online MLC, the compared methods are competitive with state-of-the-art methods. For the online hierarchical tasks, the introduced methods show promise, though further examination is warranted. In semi-supervised learning, iSOUP-PCTs perform well, particularly, when there are few labeled examples. For online feature ranking for SOP, the obtained results are unclear and warrant further examination.

Finally, we perform two case studies in which we explore how the introduced methods could be applied to real-world problems. The two case studies show that the introduced methods process data examples quickly and that they can also handle large numbers of input attributes and targets.

Povzetek

V disertaciji se ukvarjamo z napovedovanjem strukturiranih vrednosti (NSV) na podatkovnih tokovih. Pri NSV gre za učenje napovednih modelov, ki lahko napovedujejo vrednosti, strukturirane iz več preprostih komponent. Rudarjenje podatkovnih tokov pa obravnava učenje v sprotnem načinu, kjer novi podatkovni primeri prihajajo z visoko frekvenco in kjer napovedni modeli delujejo v realnem času.

Metode za NSV so v klasičnem paketnem podatkovnem rudarjenju pogoste, a le redke obravnavajo NSV v sprotnem podatkovnem rudarjenju. Najpogosteje obravnavani učni nalogi sta večciljna regresija (VCR) ter večoznačna klasifikacija (VOK). Pri VCR gre za napovedovanje več numeričnih vrednosti, medtem ko gre pri VOK za napovedovanje prisotnosti ali odsotnosti vnaprej definiranih značk. Tipično so metode za sprotno NSV zelo specifične za učno nalogo, ki jo obravnavajo.

V disertaciji vpeljemo družino metod, ki temeljijo na sprotnih drevesih za napovedovanje strukturiranih vrednosti (iSOUP-Tree) in naslavljajo več sprotnih učnih nalog NSV. Sprva vpeljemo drevesne metode za sprotno VCR, in sicer iSOUP-Tree, ki temelji na Hoeffdingovi neenakosti ter hevristiki, ki maksimizira zmanjšanje medgručne variance, iSOUP-OptionTree, ki razširi metodo iSOUP-Tree z uporabo opcijskih dreves, sprotno učenje iz samovzorcev z iSOUP-Tree drevesi ter naključni gozd iSOUP-Tree dreves.

V nadaljevanju vpeljemo metodologijo za pretvorbo iz VOK v VCR, ki nam omogoča uporabo metod za sprotno VCR za reševanje VOK.

Za naslavljanje učnih nalog napovedovanja hierarhičnih vrednosti metodo iSOUP-Tree razširimo z uporabo hierarhično utežene hevristike. Ta posameznim komponentam hierarhije dodeli uteži glede na njihovo lego v hierarhiji. To nam omogoči, da metodo iSOUP-Tree uporabimo za sprotni hierarhični različici nalog VCR ter VOK.

Poleg tega metodo iSOUP-Tree razširimo tudi v okviru napovednega razvrščanja. Napovedno razvrščevalna drevesa (iSOUP-PCT) uporabimo za reševanje naloge polnadzorovane VCR, kjer se učimo tudi iz delno neoznačenih primerov, poleg popolnoma označenih primerov, ki jih uporabljamo v običajnem, nadzorovanem učenju.

Vpeljemo tudi metode za sprotno rangiranje značilk za naloge NSV. Pri rangiranju značilk skušamo prepoznati značilke, ki so ključne za dobro napovedno zmogljivost naučenih modelov. Za reševanje te naloge prilagodimo metodo simboličnih naključnih gozdov iz paketnega načina v sprotni način učenja z uporabo naključnega gozda iSOUP-Tree dreves.

Vse vpeljane metode tudi eksperimentalno ovrednotimo. Pri sprotnem VCR se najbolje izkaže učenje iz samovzorcev z iSOUP-Tree drevesi, medtem ko naključni gozd iSOUP-Tree dreves doseže dober kompromis med napovedno točnostjo ter porabo računskih sredstev. V kontekstu sprotne VOK so vpeljane metode konkurenčne vrhunskim metodam. Pri hierarhičnih nalogah so rezultati vzpodbudni, vendar zahtevajo nadaljnjo eksperimentalno vrednotenje. Metoda iSOUP-PCT deluje izjemno dobro za polnadzorovano VCR, še posebej kadar je popolnoma označenih primerov malo. Rezultati eksperimentov za sprotno razvrščanje značilk so nejasni in zahtevajo nadaljnjo teoretično in eksperimentalno obravnavo.

Nazadnje smo preučili še dva možna primera uporabe vpeljanih metod v praktičnih okoljih. Primera kažeta, da vpeljane metode učinkovito obravnavajo podatkovne primere ter da so vpeljane metode primerne tudi, kadar imajo podatkovni primeri veliko tako odvisnih kot neodvisnih komponent.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

SOP	...	structured output prediction
MTR	...	multi-target regression
MLC	...	multi-label classification
HMTR	...	hierarchical multi-target regression
HMLC	...	hierarchical multi-label classification
DAG	...	directed acyclic graph
SSL	...	semi-supervised learning
PCT	...	predictive clustering tree
TDIDT	...	top-down induction of decision trees
MOA	...	Massive Online Analysis framework for online learning
PAC	...	probably approximately correct

Symbols

$\mathbb{N}$ ... The set of natural numbers, starting with 1.
$\mathbb{N}_k$ ... The set of the first k natural numbers, i.e., $\mathbb{N}_k = \{1, 2, \dots, k\}$.
$\mathbb{R}$ ... The set of real numbers.
$\mathbb{R}^+$ ... The set of positive real numbers.
A^c ... The complement of set A.
$\mathcal{P}(A)$ ... The powerset of set A, i.e., the set of all subsets of A.
$\mathcal{B}(A)$ ... The set of all bags of set A, i.e., the set of all "subsets with repetitions" of A.
A^B ... For sets A and B, A^B is the set of all mappings from B to A.
$\mathrm{P}(A)$ ... Probability of event A occurring.
$\mathrm{E}[\mathcal{X}]$ ... Expected value of random variable $\mathcal{X}$.

Chapter 1

Introduction

> This book was written using 100% recycled words.
>
> — Terry Pratchett

Artificial intelligence (Russell & Norvig, 2009) applications are becoming ubiquitous and range from self-driving cars to automated recognition of speech. *Machine learning* (Mitchell, 1997), as a branch of artificial intelligence, is primarily concerned with computer programs that can learn from experience, e.g., from interaction with available data, and then be applied to a variety of application domains. Machine learning methods have been especially successful at data analysis, more specifically, *data mining* (Witten, Frank, Hall, & Pal, 2016). In data mining, we are generally interested in models that generalize over the data that is provided. We wish to address several types of tasks, such as predicting some of the properties, which we call *targets*, of a data example from the remaining properties in the task of *predictive modeling*, grouping similar data examples together in the task of *clustering* or finding interesting patterns, such as association rules, in *pattern mining*. These tasks are by far the most studied and well understood.

Classical machine learning methods for predictive modeling which learn from the data in its entirety have been researched for the better part of the last century. These methods expect the data to come in the form of a table, where the rows represent individual data examples and the columns denote the different properties (called *attributes*) of the examples. The data examples are assumed to be generated by the same process, i.e., to be distributed according to a fixed probability distribution. The entirety of the training data is available at the start and throughout the learning process, with the assumption that the methods can fit the data into their working memory. After the learning process is completed, we assess whether the learned model has the desired properties, such as predictive performance. These methods operate in the *batch learning* mode, which relates to the fact that all of the (training) data examples are available in a single batch that is used for learning.

However, data sources of considerably higher complexities have recently become abundant, to which classical machine learning methods are not adapted. This increased complexity can be due to arbitrarily large amounts of data examples, an increase in the complexity of the individual data examples or the high frequency of incoming data examples. These types of tasks were recently grouped under the umbrella term “Big Data” (Marz & Warren, 2015). In general, Big Data tasks have one or more of the following (alliterative) aspects:

- *volume*, denoting the large quantity of data examples;

- *variety*, denoting the different types of data collected;
- *velocity*, denoting the high speed of generation of data;
- *variability*, denoting the potential of non-stationary data; and
- *veracity*, denoting the varying quality of the measured data.

Data stream mining (Aggarwal, 2007; Gama, 2010) is a subfield of data mining that analyzes high-frequency, practically unbounded data sources, i.e., deals with both the volume and velocity aspects of Big Data. The need for mining *data streams*, which are information-rich high-frequency data sources, has only increased as they become readily available for analysis. Data streams generally require real-time interaction, which is where classical data mining methods fail, since interaction with the model is only available after learning, i.e., after all the data examples have been processed. When mining data streams, we assume that the data stream is potentially infinite in size. In practice, this means that the learning process is ongoing as new data examples are always becoming available. Therefore, waiting for the model to fit all of the data examples is impossible.

To address data stream mining, machine learning methods which learn incrementally have been developed and have become the subject of considerable study. These methods operate in the *online learning* mode, meaning that they learn from data examples as they become available and are expected to operate in real- or near real-time, i.e., a model must be available for application at any point in time. To facilitate real-time operation, the learned generalizations should keep a low profile in terms of computational resources, which invariably requires (selective) forgetting of past data examples. Since there can be arbitrarily many examples, forgetting is crucial, as it allows the model to learn from a considerably larger amount of data under the constraints of computational resources of the underlying platform. Forgetting is also paramount to another aspect of data stream mining, *concept drift* detection (Gama, Žliobaite, Bifet, Pechenizkiy, & Bouchachia, 2014). Unlike the classical data mining scenario, there is no assumption that the data is generated by a single underlying distribution. In stream mining, the distribution can (and often does) change. A data stream mining model should be able to correctly identify the change in distribution and update the model to take into account the new, changed distribution.

Data has also become more complex in other ways. Most early machine learning methods for predictive modeling were designed to only predict one of the properties, a target, of an example from the rest. The target can be a nominal value and the predictive modeling task is called *classification* or a numeric value in which case the predictive task is called *regression* (Kononenko & Kukar, 2007). However, many modern data sources involve data where we predict not just one of the properties of each data example but several of them. For example, when an example represents a document, we wish to predict several categories into which it belongs. This kind of task is called *multi-label classification* (MLC) (Madjarov, Kocev, Gjorgjevikj, & Džeroski, 2012; M.-L. Zhang & Zhou, 2014; Gibaja & Ventura, 2015). When we are interested in predicting multiple numeric values, we talk about *multi-target regression* (MTR) (Struyf & Džeroski, 2006). In general, the targets that we predict are structured, e.g., from a simple vector in multi-target regression to a hierarchy of categories in hierarchical multi-label classification. We group these predictive modeling task under the umbrella term of *structured output prediction* (SOP; Bakir (2007))
.

Recently, there has been considerable research into SOP task in the batch setting, in particular, addressing multi-target regression, multi-label classification, hierarchical multi-label classification, hierarchical multi-target regression, time-series prediction and other tasks. However, few machine learning methods that address SOP tasks on data streams have been introduced. These methods are, most often, specifically tied to a particular SOP task, e.g., the method of Read, Bifet, Holmes, and Pfahringer (2012) is intended for multi-label classification and the method of Ikonomovska, Gama, and Džeroski (2011a) is intended for multi-target regression. This reduces the reusability of these methods and requires separate methods for addressing different SOP tasks.

In the following sections, we present the motivation for this thesis as well as the goals, hypotheses and methodology of the thesis. Afterwards, we summarize the contributions of the thesis and conclude the chapter with a description of the organization of the thesis.

1.1 Motivation

As we have discussed above, few methods address SOP tasks in an online learning setting. These methods are generally tailored to one specific SOP task and have a limited potential for reuse on other SOP tasks. Furthermore, several SOP tasks, such as hierarchical multi-label classification and hierarchical multi-target regression have, to the best of our knowledge, not been addressed in an online learning setting at all.

We wish to address SOP tasks in an online setting by developing a versatile method that can address several types of SOP tasks. To this end, we consider transforming tasks, such as multi-label classification, into multi-target regression tasks. This kind of method can then be extended also to hierarchical SOP tasks, such as hierarchical multi-target regression and hierarchical multi-label classification.

Similarly, *semi-supervised learning* SOP tasks have also not been addressed in an online setting. In these tasks, not all data examples are completely annotated. In particular, some of the examples do not have information about the values of the targets. These examples are called *unlabeled*, while the examples for which all target values are provided are called *labeled*. Semi-supervised machine learning methods utilize the unlabeled examples, in addition to the labeled examples, to achieve better predictive performance as compared to using just the labeled examples. In the batch setting, various methods for semi-supervised SOP tasks have been proposed (Brefeld & Scheffer, 2006; Altun, McAllester, & Belkin, 2006; Levatić, Ceci, Kocev, & Džeroski, 2017a, 2017b), however, semi-supervised SOP tasks have not yet been addressed in the online setting.

Another data mining task related to predictive modeling is *feature ranking*, which seeks to identify the attributes (independent variables) most important for the learning process. Feature ranking for SOP tasks has received some consideration in the batch setting. The recent work by Petković, Džeroski, and Kocev (2017) addresses feature ranking for the multi-target regression task. In the online setting, feature ranking for SOP tasks has, as far as we can tell, not been addressed at all.

1.2 Goals, Hypotheses and Methodology

Given the motivations above, the **main goal** of the thesis is to design and implement versatile methods for online structured output prediction that can be applied across multiple types of structured outputs. In particular, we will be dealing with stationary data streams in which the distribution governing the examples is not expected to change. Specifically, to achieve the main goal, we will:

Goal 1 Design and implement methods for online multi-target regression that will serve as a cornerstone on which methods for other types of structured outputs are designed.

Goal 2 Design and implement methods which address the online multi-label classification task through the utilization of multi-target regression methods.

Goal 3 Extend the methods for online multi-target regression and multi-label classification towards hierarchical multi-target regression and hierarchical multi-label classification.

Goal 4 Design and implement methods for online semi-supervised learning, which can utilize unlabeled examples in addition to the labeled ones.

Goal 5 Design and implement a method for online feature ranking, which can address feature ranking for a variety of structured output prediction tasks.

Goal 6 Experimentally evaluate and compare the proposed methods for SOP on data streams.

To achieve the main goal of the thesis and develop a method for online structured output prediction, which can address multiple types of structured outputs, we start with a method for online multi-target regression and use it as a basis for addressing other tasks. We proceed by transforming other types of structured outputs into vectors of numeric values, i.e., by transforming other SOP tasks into multi-target regression tasks. Thus, to use the online multi-target regression methods for other tasks, we first transform the task to the multi-target regression task, apply a MTR method and then transform the multi-target regression predictions back to the original structured output. The remaining goals follow incrementally from this method.

To achieve the goals of this thesis, we operate under the following hypotheses:

Hypothesis 1 Online tree-based methods for online multi-target regression, such as model trees (Ikonomovska, Gama, & Džeroski, 2011a), option trees and ensembles of trees (Ikonomovska et al., 2015), are good candidates for extension towards other types of structured outputs.

Hypothesis 2 Online multi-label classification tasks can be transformed into online multi-target regression tasks, by using problem transformation methods.

Hypothesis 3 Methods that address hierarchical prediction tasks in the batch learning setting (Vens et al., 2008; Mileski et al., 2017) can be modified and applied to the online learning setting.

Hypothesis 4 The predictive clustering framework (Blockeel & De Raedt, 1998), which enables semi-supervised learning in the batch setting (Levatić et al., 2017b), can be adapted to address semi-supervised tasks in an online setting.

Hypothesis 5 Methods for batch feature ranking based on predictive models that are agnostic of the type of structured output (Petković et al., 2017) can also be employed in the online setting.

To achieve the goals outlined above and to test the validity of our hypotheses, we use the following methodology:

Online tree-based methods for MTR. Tree-based methods are very popular in both the batch and online settings for a variety of tasks (Kocev, Vens, Struyf, & Džeroski, 2013; Levatić et al., 2017b; Read et al., 2012; Ikonomovska et al., 2015). Their popularity stems from their learning and prediction speed, interpretability of the learned models and excellent predictive performance. We utilize the tree-based online multi-target method as a cornerstone for addressing other online SOP tasks. In particular, we re-implement and correct several shortcomings of the preliminary FIMT-MT method introduced by Ikonomovska, Gama, and Džeroski (2011a) and name it incremental Structured Output Prediction tree (iSOUP-Tree). The name already indicates that the method will be able to address several types of structured outputs.

Problem transformation from MLC to MTR. Many methods which address multi-label classification use problem transformation methods in both the batch and online learning settings. These methods transform a multi-label classification task into several simpler non-structured classification tasks (Read, Pfahringer, Holmes, & Frank, 2011; Read, Pfahringer, & Holmes, 2008; Tsoumakas & Vlahavas, 2007). We take a different path: instead of decomposing a multi-label classification task into simpler classification tasks, we transform it into a multi-target regression task of similar complexity and solve the tasks by using the developed methods for MTR.

Adaptation of batch SOP methods to an online learning setting. As tree-based methods have been studied extensively for various SOP tasks in the batch setting, we seek to adapt some of these methods and include them into our online tree-based methods. In particular, we adapt the methods for hierarchical multi-label classification (Vens et al., 2008) and hierarchical multi-target regression (Mileski et al., 2017) for the online variants of these SOP tasks.

Furthermore, we extend our tree-based methods toward the predictive clustering framework (Blockeel & De Raedt, 1998). This allows us to use them in a semi-supervised learning setting, by using a method adapted from predictive clustering trees in the batch setting (Levatić et al., 2017b).

Finally, we utilize a symbolic random forest method for feature ranking method, recently introduced by Petković et al. (2017). Given our ability to address several SOP tasks, this feature ranking method also extends to all of the tasks.

Evaluation. To experimentally evaluate the introduced methods, we use task appropriate evaluation procedures and measures. In particular, we use the predictive sequential evaluation method (Dawid, 1984) for evaluating online learners. In addition to evaluating the predictive performance of the introduced methods, we also look at their consumption of computational resources, in particular in terms of memory consumption and processing time.

Implementation. All of the introduced methods are implemented in the Massive Online Analysis (MOA) framework[1] (Bifet, Holmes, Kirkby, & Pfahringer, 2010). MOA is an open source and freely available platform implemented in Java.

1.3 Contributions

In this thesis, we propose several methods for online multi-target regression, introduce the MLC via MTR problem transformation methodology, and adapt several methods from the batch setting for the tasks of online hierarchical multi-label classification, online hierarchical multi-target regression, online semi-supervised learning and online feature ranking.

[1] URL: https://moa.cms.waikato.ac.nz/ (accessed 2018/01/22)

Several of these methods have been published in conference and journal publications (Osojnik, Panov, & Džeroski, 2015a, 2015b, 2016, 2017a, 2017b). These papers are listed in the Bibliography section. We summarize the contributions of this thesis as follows:

Contribution 1 *The iSOUP-Tree family of methods for online multi-target regression.*

We introduce several methods for online multi-target regression. These range from single-tree methods like iSOUP-Tree, in particular, its regression and model tree variants, the option tree extension of iSOUP-OptionTree, as well as online ensemble methods like bagging and random forests (Oza & Russel, 2001; Oza, 2005) using iSOUP-Trees as base learners. These methods utilize the Hoeffding bound to grow the tree incrementally when (probabilistically) sufficient evidence is observed. To facilitate the calculation of the requisite heuristics without access to the observed data examples, these methods use extended binary search trees to store and update the needed statistics.

Contribution 2 *The online multi-label classification via online multi-target regression problem transformation methodology.*

Problem transformation methods are particularly common when addressing the task of multi-label classification. However, the multi-label classification via multi-target regression problem transformation methodology that we introduce transforms a complex task (MLC) into a different complex task (MTR). This allows for the use of methods for multi-target regression to address multi-label classification. In particular, this methodology is not specifically tied to the use of the introduced tree-based methods for MTR, but can be used in conjunction with any other MTR method.

Contribution 3 *Extension of the iSOUP-Tree method towards hierarchical prediction tasks.*

We extend the regular iSOUP-Tree toward hierarchical prediction tasks, such as hierarchical multi-target regression and hierarchical multi-label classification, by modifying its growth heuristic. In particular, we weigh different targets or labels differently, based on their position in the hierarchy. This puts different degrees of emphasis on the different targets during the growth of the trees.

Contribution 4 *Extension of iSOUP-Tree into the predictive clustering framework and towards online semi-supervised learning.*

We introduce the iSOUP-PCT method, which is an extension of the iSOUP-Tree method into the predictive clustering framework. Regular trees only consider the homogeneity of the targets when growing the tree, while in the predictive clustering framework we additionally consider the homogeneity of the input attributes as well. For the iSOUP-PCT method, we adapt the extended binary search tree structure to also include the statistics of the input attributes in addition to the statistics of the target attributes. This does, however, incur a considerable cost in terms of computational resources.

Nevertheless, the increased use of computational resources is justified in some cases. In particular, we can use iSOUP-PCTs to address semi-supervised learning tasks and use unlabeled examples in addition to the labeled ones.

Furthermore, for online semi-supervised learning, we additionally modify the initialization of the predictive models of each leaf in the tree, by utilizing the statistics that are needed for the calculation of the growth heuristic. In this way, we can extract additional information from the labeled examples which is particularly valuable when the labeled examples occur infrequently.

Contribution 5 *The online symbolic random forest method for feature ranking.*

In online feature ranking, we must produce importance scores for each feature, i.e., descriptive attribute. To calculate the scores, and consequently determine a ranking, we first grow a random forest of randomized iSOUP-Trees. The scores for each attribute are then calculated based on its appearance in the internal nodes of the individual trees in the ensemble. Each appearance of an attribute contributes to its score, proportionally to how close to the root of the tree it is positioned.

Contribution 6 *Empirical evaluation of the proposed methods.*

We perform empirical evaluation of all of the proposed methods. We use appropriate evaluation procedures and measures for all of the addressed SOP tasks.

For the multi-target regression task, we show that ensemble methods perform the best of all of the introduced methods, but use considerably more computational resources. We also observe a trade-off between the model complexity and its predictive performance.

We compare the introduced methods in combination with the MLC via MTR methodology to a state-of-the-art online tree-based method for multi-label classification. While the results on example-based and label-based evaluation measures are mixed, the introduced methods tend to outperform the competitors in terms of the ranking-based measures. As the ranking-based evaluation measures take into account relative confidences in the predictions of the models, we conclude that the introduced methods have a lot of merit for online multi-label classification.

We also investigate how the use of the hierarchy in hierarchical prediction tasks impacts the predictive performance for the targets at the lowest level in the hierarchy. In hierarchical multi-target regression, using a bottom-weighted heuristic to grow the tree improves the predictive performance over the non-hierarchical method. In hierarchical multi-label classification, the results are not as clear-cut. In some cases, the bottom-weighted method performs best, in some cases the top-weighted methods (that gives larger emphasis to the targets higher up in the hierarchy) performs best, while in some cases the non-hierarchical method outperforms both of the hierarchical methods.

For semi-supervised multi-target regression, we show that the use of semi-supervised iSOUP-PCTs improves the predictive performance over just using the labeled examples for learning. This improvement is larger when the ratio of labeled to unlabeled examples is lower.

Finally, we compare the online feature rankings obtained by the symbolic random forest feature ranking method to those obtained by the corresponding batch feature ranking method. In most cases, there is considerable overlap in the top parts of the ranking, however, this is not always the case. This prompts us to discuss whether it is reasonable to expect the rankings from the batch and online methods to overlap exactly.

Contribution 7 *Case studies in two practically relevant domains.*

In addition to the empirical experimental evaluation, we discuss two case studies of online multi-target regression applied to two relevant domains.

The first case study deals with the prediction of electrical power consumption for the Mars Express satellite. Given the orientation of the satellite in relation to the Sun, we observe features which determine the thermal influx over the different faces of the satellite. We apply the introduced methods for multi-target regression and discuss the potential for practical use of the obtained results.

Similarly, we address a case study which concerns the forecasting of photo-voltaic power generation. Here, we are tasked with predicting the power generation of several photo-voltaic power-plants based on historical power generation data, current weather conditions and weather forecasts, as well as the geographical position of the power-plants. Due to the large number of both input attributes and targets, we also consider this case study as a "stress-test" of the introduced methods.

1.4 Organization of the Thesis

In this introductory chapter, we have presented the general context of the thesis, in particular how it relates to other topics, such as machine learning, data mining, data stream mining and structured output prediction. We have outlined the thesis' goals and the hypotheses we operate under. Finally, we have listed the main scientific contributions of the thesis. Here, we present the general structure of the remainder of the thesis.

Chapter 2 formally defines terms pertaining to the thesis, e.g., data example, dataset, data mining task, online learning setting, etc. In particular, we define the structured output prediction tasks that are addressed in the following chapters.

Chapter 3 gives a detailed overview of related work. In particular, we consider related work in the field of online learning, batch methods for structured output prediction, and online methods for structured output prediction.

Chapter 4 introduces the iSOUP-Tree family of methods for online multi-target regression. We continue with a description of the MLC via MTR problem transformation methodology. We proceed by introducing the adaptation of iSOUP-Tree for online hierarchical prediction and by introducing the iSOUP-PCT, which extends iSOUP-Tree into the predictive clustering framework and can address online semi-supervised learning. Finally, we conclude the chapter by introducing the online symbolic random forests method for feature ranking.

Chapter 5 describes the evaluation methodology we use for the experimental evaluation. In particular, we define the types of evaluation procedures used in the online setting, as well as task-appropriate evaluation measures for each considered task.

Chapter 6 presents the experimental design that we use to evaluate the introduced methods. It is subdivided into several sections that deal with different aspects of the individual experiments, grouped per task, such as experimental question, datasets, etc.

Chapter 7 details and discusses the experimental results. It is structured to follow Chapter 6.

Chapter 8 presents the two case studies, i.e., the Mars Express case study and the photo-voltaic power forecasting case study.

Chapter 9 concludes the thesis with a brief summary of the scientific contributions of the thesis, a discussion on the achievement of thesis' goals and confirmation of hypotheses, and an outline of several avenues for further work.

Chapter 2

Data Mining Tasks on Data Streams

> You can have data without information, but you cannot have information without data.
>
> — Daniel Keys Moran

Machine learning is an application and research field that is concerned with computer programs that learn from experience (Mitchell, 1997). Machine learning methods have been applied to a multitude of application domains, such as computer vision, robotics, and, of course, data mining. The focus of this thesis is to introduce and evaluate a family of machine learning methods that can be used to address multiple tasks. For this purpose, we need to understand several concepts from the broader area of data mining. In this chapter, we define the concepts of datasets, data mining tasks and learning settings. The chapter details the various tasks that one might encounter in a data mining scenario, e.g., clustering, predictive modeling, pattern mining, and specifically defines the various structured output prediction tasks that are addressed in this thesis.

This chapter is divided into three sections. The first section is concerned with the data, specifically, with individual data examples and the way they are combined into collections, which can then be used as inputs to learning methods. In the second section, we introduce a general framework for defining data mining task groups. In the third section, we differentiate between the batch and online learning settings. We conclude with an overview of the data mining tasks, which will be addressed by the machine learning methods we introduce in this thesis.

2.1 Data Examples, Data Types and Datasets

A *data example*, or *example*, for short, represents the record of an object or a measurement. It is described in terms of one or more properties, which can take various types of values. To properly define the structure of the example we use the generic ontology of data types (Panov, Soldatova, & Džeroski, 2016) based on the ISO/IEC 11404 standard of general-purpose data types (International Organization for Standardization, 2007). In the ontology, data types are initially divided into primitive and generated, and are further refined by their properties, such as numeric or non-numeric, ordered or non-ordered, etc. For a brief overview of the introduced data types see Table 2.1. Notably, in this thesis we present only a subset of data types that are relevant in the context of this thesis, however, there are other primitive and generated data types that are described in the work of Panov et al.

(2016)[1].

The most common primitive data types in data mining are the boolean data type, that takes a true ($\top$) or false ($\bot$) value, the discrete data type, which takes one value out off a set of predefined possible values, and the real data type, which takes values from the set of real numbers $\mathbb{R}$. Formally, discrete is a family of data types, as each choice of possible values, e.g., discrete(A, B, C), defines a separate data type. For example, boolean is equivalent to discrete($\top$, $\bot$). In this thesis, we refer to discrete as a data type, inferring that the set of choices had already been determined.

The boolean and discrete data types are unordered and only have a finite number of possible values, while real can take an infinitely many possible values, which are ordered using the standard ordering of real numbers. On the other end of the spectrum we have another important primitive data type, the data type void, which takes only one value. We usually denote its value by ?, as in data mining we most commonly use the void data type to represent missing values. However, to properly define a data type, which can either take a value or have its value be missing, we need to first define generated data types.

Generated data types are explicitly defined in relation to one or more (primitive or generated) *component* data types. The tuple data type represents an ordered list of data types, e.g., tuple(boolean, real) represents a pair of a boolean value and a real number. An example of this data type might be tuple($\top$, 3.14). Notably, a tuple will always have a set number of values.

Other generated data types, like set, bag or sequence, have a variable size. The bag data type represents a collection of some data type, e.g., set(real) represents a collection of real numbers. We specifically use the word "collection" in place of "set", as the bag data type allows its values to appear multiple times. Conversely, the set data type is a bag which does not allow the repetition of its values, i.e., it represents a set in the mathematical sense. Unlike set and bag, which are both unordered, the sequence data type is an ordered data type, since we have an ordering of its values. These data types are examples of *aggregate data types*, which are characterized by having their value composed from the values of their component data types.

The final generated data type we will specifically describe in this thesis is the choice data type. The choice data type represents that a value can be taken from any of the provided data types, e.g, both 3.14 and $\top$ are examples of values for the type choice(boolean, real). The choice data type is predominantly used for quantifying the possibility of missing values. The choice data type is not an aggregate data type, as its value is not composed of many component values, but the value it takes can be of more than one type. For example, let us consider a numeric value that can, in some examples, be missing. To define this kind of data type we use choice(real, void), and an example of this type might take a real value, e.g., 3.14, or it can be missing, i.e., has value ?.

Using the ontological definitions it is fairly easy to mathematically construct the value space of all possible examples of a given data type: real examples are members of $\mathbb{R}$, boolean are members of $\{\top, \bot\}$, while discrete examples have values from the set of their possible options. The value space of a tuple is the Cartesian product of its components, e.g, the corresponding space of tuple(boolean, real) is $\{\top, \bot\} \times \mathbb{R}$. To obtain the space of a set data type, we take the powerset of the dependent data type, e.g., for set(discrete(A,B,C)) we get $\mathcal{P}(\{\mathtt{A}, \mathtt{B}, \mathtt{C}\})$. To obtain the space of a choice data type, we take the disjoint union[2] of the corresponding spaces, e.g., choice(boolean, real) yields $\{\top, \bot\} \cup^* \mathbb{R}$. The case of missing data (when one of the components of choice

[1] In the following paragraphs we use the mono-space font to refer both to the entities of the ontology as well as the individual examples of a given data type.

[2] $A \cup^* B = A^* \cup B^* = A \times \{0\} \cup B \times \{1\}$

Data type	**Values**	**Notes**
		Primitive
`boolean`	$\top, \bot$	equivalent to `discrete(`$\top$`, `$\bot$`)`
`discrete(`$\mathcal{L}$`)`	$\lambda \in \mathcal{L}$	
`real`	$x \in \mathbb{R}$	
`void`	`?`	represents missing values
		Generated
`tuple(`T_1`, `T_2`)`	$(x_1, x_2) \in T_1 \times T_2$	tuple of values
`set(`T_1`)`	$S \in \mathcal{P}(T_1)$	example is a set of values, i.e., it can contain multiple elements
`choice(`T_1`, `T_2`)`	$x \in T_1 \cup^* T_2$	commonly used in conjunction with `void` to represent missing values

Table 2.1: Overview of relevant data types. T_1 and T_2 are arbitrary data types.

is `void`), translates to adding a representation of a missing value `?` to the space of the other component. For example, for the `choice(real, void)` data type we get the value space of $\mathbb{R} \cup^* \{\texttt{?}\}$. For a given data type, we can thus define the value space of all possible examples, i.e., the *domain*, which we denote by X.

Now that we have defined the format of the individual example, we can proceed to the definition of the dataset. A *dataset* $\mathcal{D}$ is a collection of examples of a certain data type, and, unlike the name would suggest, datasets are most commonly not sets, but rather bags (sets with repetitions) of examples or sequences of examples (as is case in learning from data streams)[3].

To properly define the concept of a dataset, we must thus provide the data type of its examples as well as the examples themselves. For brevity, we refer to both the data type and the corresponding examples as $\mathcal{D}$ based on the context. Given that a dataset contains homogeneous examples, i.e., examples of the same data type, we can look at the dataset as a sample from the domain X. A basic assumption of data mining is that the individual examples of the dataset $\mathcal{D}$ are similar, i.e., that they are generated by the same underlying distribution D over the domain X. Thus, $\mathcal{D}$ can be seen as a sample of distribution D in space X.

Most machine learning methods expect datasets to be presented in a particular representation. Most commonly, we represent a dataset as a table, where columns correspond to different components of the data type, while the rows denote individual examples. This representation is often called *dense*, as it explicitly provides values for all of the components, even when a presumed default value can be assumed, and could thus be omitted. In a *sparse* representation only the values which are different from the presumed default value and their corresponding index are provided. The representations are convertible among themselves, given the knowledge of the default values.

As is evident from Figure 2.1, which shows a sample dataset in both dense and sparse representation, the choice between the representations is a trade-off. If only few values are

[3] We use angular brackets when referring to datasets to emphasize that the collection structure is not necessarily a set, i.e., we use $\mathcal{D} = \langle x_1, x_2, x_3, \dots \rangle$. If we are talking with a specific dataset, which is a set or a sequence, we use the regular mathematical notation, i.e., $\mathcal{D} = \{x_1, x_2, x_3, \dots\}$ and $\mathcal{D} = (x_1, x_2, x_3, \dots)$, respectively.

(a)

		set(discrete(A, B, C))		
real (1)	real (2)	boolean-A	boolean-B	boolean-C
1.7	0.0	1	0	1
42.0	16.2	0	0	1
0.0	0.0	0	1	0

(b)

```
{   1:  1.7,             3:  1,         5:  1  }
{   1:  42.0,  2:  16.2,                 5:  1  }
{                                4:  1         }
```

Figure 2.1: Dense (a) and sparse (b) representation of a sample dataset of data type `tuple(real, real, set(discrete(A, B, C))`. In the sparse representation, all values that are not provided are assumed to be 0.

different from the default values, the use of a sparse representation considerably reduces the length of an example, however, when most values are provided, the sparse representation is longer due to explicitly defining the column indexes in addition to the values. For clarity, we will present data from here on in a tabular, dense representation.

From construction of the domain X for a `tuple` data type, it is evident that its individual components easily map into columns of a table. However, achieving a tabular representation of a `set` data type is more complicated, especially when the dependent data type is infinite, though this case is rare. In the case of a finite dependent data type, we can enumerate each of the possible values as its own `boolean` data type and column and use $\top$ when the value is present, and $\bot$ when it is not. Furthermore, we also often use numerical values to represent `boolean` components, i.e., 1 to represent $\top$ and 0 to represent $\bot$, as is seen in Figure 2.1.

In the context of using machine learning to address data mining tasks, a dataset is given as an input to the machine learning method to "solve" the data mining task. The type of result produced by a machine learning method depends on the task at hand. These are described in the following section.

2.2 Data Mining Tasks

According to Džeroski (2006), data mining tasks are divided into several groups, particularly in relation to what is their result. These are probability distribution estimation, (probabilistic) predictive modeling, clustering and pattern discovery. Probability distribution estimation is (in short) the estimation of the underlying probability distribution D of a dataset $\mathcal{D}$. In predictive modeling, the domain X is divided into two components, the descriptive and target components. The goal is then to produce a predictive model, which can predict the target component based on the values of the descriptive component. Clustering is the task of assigning each example in $\mathcal{D}$ to one of several possible clusters – subsets of the dataset which are internally similar, but externally different. Finally, pattern discovery is concerned with constructing a set of possible patterns $\Pi(\mathcal{D})$ from the dataset and subsequently determining which of them are interesting according to some predefined criteria.

Before the tasks are properly defined, we offer a few general remarks. Most notably, we omit a detailed description of probability distribution estimation, as, methodologically, this thesis focuses particularly on the task of predictive modeling, with a brief consideration of pattern discovery. Clustering is defined to offer a better understanding of some of the introduced methods. Additionally, when we refer to a result of a data mining task in general, i.e., independently of the task, we adopt the terminology used by Džeroski, Goethals, and Panov (2010, Chapter 1) and use the term *generalization*. This term is motivated by observing that estimations of probability distributions, predictive models, clusterings and discovered patterns in one way or another generalize over the provided data examples.

2.2.1 Predictive modeling

In predictive modeling each example is composed of two components, specifically, a *descriptive* and a *class* or *target* component. In a way, examples are assumed to be of a tuple data type, i.e., `tuple(`*descriptive*`,` *target*`)`, where *descriptive* and *target* are arbitrary complex data types. In predictive modeling, the domain of the descriptive data type is called the *input* or *descriptive space* and is denoted by X.[4] Conversely, the domain of the target data type is called the *output* or *target space* and is denoted by Y. Each example is represented as a pair (x, y), where $x \in X$ and $y \in Y$. We refer to the components of the descriptive data type as *attributes* or *features*. Attributes of `boolean` and `discrete` data types are called *nominal* attributes, while `real` attributes are called *numeric* attributes. Throughout the thesis, we will always refer to the input space as X and N will be its dimension, i.e., the number of attributes, while Y will be the output space of dimension M.

The goal of predictive modeling is to learn a predictive model, which best captures the dependencies in the provided dataset $\mathcal{D}$. A *predictive model* is thus a result of applying a learning method to a dataset, which can predict the target component of an example based on its input component. Here, we make the assumption that for each observed pair (x, y), the value y is in some way dependent on the value of x. This means that a predictive model m is a *predictive function* $m :: X \to Y$, i.e., $m \in Y^X$.

To measure how well a predictive model fits the data, we use various task-specific evaluation measures, which measure how well the prediction $m(x) =: \hat{y}(x)$ approximates the actual value y for some examples (x, y) from the given dataset. Specific evaluation measures which are used for the relevant predictive modeling tasks in this thesis are described in Chapter 5.

From a machine learning standpoint, it is important to consider a model m as more than just a predictive function, as the model might contain more information than just how to predict the output value based on the input value. Džeroski (2006) refers to this as the dichotomy of models or patterns. In some cases the model can contain no additional information, e.g., in many statistical models, such as least-squares regression, the learned model is exactly a function from X to Y. In the case of models learned by machine learning methods, the models contain additional structural information. This is especially important in online learning, as we will see below, where learning from additional examples can change the structural information but not impact the predictive function.

[4] We use the same notation both for the domain as well as the input space, as we rarely discuss the entire domain in the context of predictive modeling. When we do, however, we denote the domain by $D = X \times Y$.

Notably, when both the input space X and output space Y are finite, there is only a finite amount of different functions that map from X to Y. However, there are many learning methods which produce models in completely distinct formalisms, e.g., decision trees, neural networks or simply functions, as is often the case in statistical learning[5]. These formalisms refer to the structural component(s) of a model. In the above, finite, case, models in different formalisms will inevitably induce the same predictive function when learned from the same data examples. In the extreme, a method which randomly selects a predictive function can achieve the same quality of fit as a well reasoned learning method. In this sense, a model is more than just a predictive function, as it operates on some assumptions, which, through the learning process, guide the induction of its predictive function.

Clearly, when we look at a model as a predictive function, it is a function which maps from X to Y. However, this raises the question of where does the model, with all of its structural information, "live" – to what set does *it* belong? An imagined set of models would then contain *all* possible models, regardless of their formalism and the validity of their assumptions. While this kind of a set of models could technically exist, it would be impossible to describe. To this end, we generally limit ourselves to a set of models which share a formalism. In this context, we call the model formalism the *model language*, and the set of all possible models expressed in a model language is usually called the *hypothesis space* $\mathcal{H}$. Notably, the model language is not considered to be a part of a predictive modeling task, rather its selection is based on the desired properties of the predictive model, such as interpretability, greediness, etc.

To formally define a predictive modeling task, we must thus provide a dataset $\mathcal{D}$ and an evaluation measure $\mathcal{M} :: Y^X \to \mathbb{R}$, which the learned model seeks to optimize, i.e., either minimize or maximize. In plain terms, this means we wish to learn a predictive model which is, for example, as accurate as possible. However, as this thesis deals with an introduction of new machine learning methods which address various predictive modeling tasks, we are not specifically interested in a particular dataset and a particular evaluation measure.

Comparing different machine learning methods in the scope of predictive modeling is generally done as a "gauntlet," where the methods compete over a selection of datasets for a given evaluation measure. The *predictive performance* of a model m with regards to a measure $\mathcal{M}$ on dataset $\mathcal{D}$ refers to calculating the measure $\mathcal{M}$ with predictions provided by the model m.

In the case of predictive modeling tasks where there are many evaluation measures, e.g., in (single-target) classification or multi-label classification (see Section 5.2.2), it can occur that optimizing all of the evaluation measures at the same time is impossible, as can be seen in classification by observing the trade-off between the precision and recall measures (Kononenko & Kukar, 2007). This is further discussed in Chapter 5.

As we have discussed above, when we say that a dataset $\mathcal{D}$ must be provided, we specifically refer to *both* the data type definition as well as the examples themselves. Predictive modeling tasks are divided along two main dimensions, which are both encoded in the target data type. The first dimension refers to the components of the target data type, i.e., when the target data type is composed of or is itself a `boolean` or `discrete` data type, the task is called classification. Similarly, when the target data type has components or is itself a `real` data type, the task is called regression. For a more extensive taxonomy of tasks refer to Panov et al. (2016).

The second dimension is concerned with the presence of missing data, i.e., whether

[5] In statistical learning, predictive models are functions of a prescribed form whose parameters are fit using the available data examples, e.g., partial least squares regression or logistic regression.

there is some component of the target data type of the form `choice(`*data type*`, void)`. When this is not the case, the predictive modeling task is said to be *supervised*. If the target data type is `choice(`*data type*`, void)`, the task is called a *semi-supervised* task, e.g., semi-supervised classification or semi-supervised regression. If the target data type is structured and each of its components can be missing individually, we refer to the task as *partially-labeled*, e.g., partially-labeled multi-target regression or partially-labeled multi-label classification.

To summarize, we formulate a predictive modeling task as:

Given:

- An input (or descriptive) space X constructed from the input data type, most commonly a `tuple` aggregate of primitive data types. The space X is then constructed as $X_1 \times X_2 \times \cdots \times X_N$ and is spanned by N descriptive attributes.
- An output (or target) space Y constructed from the target data type, which is spanned by one or more target variables.
- A collection of examples, $\mathcal{D} = \{e_i = (x_i, y_i) \mid x_i \in X,\ y_i \in Y\}$, where each example e_i is comprised of an element of the input space x_i and an element of the output space y_i. Some components of y_i, or even the entire y_i, can be missing in the case of semi-supervised or partially-labeled tasks.
- An evaluation measure $\mathcal{M} : Y^X \to \mathbb{R}$, which measures how well a model performs.

Find: A predictive model m trained on $\mathcal{D}$ that optimizes $\mathcal{M}$.

Notably, $\mathcal{D}$ is the collection of examples that are used to learn the predictive model m. In the batch learning setting, this is called the training set, while in online learning setting all examples from $\mathcal{D}$ are generally used both for learning and for evaluation. Furthermore, we define $\mathcal{M}$ as generally as possible allowing it to encode not only the particular measure of predictive performance, such as those described in Chapter 5, but also the general evaluation procedure. In particular, it can describe the evaluation procedure which uses a training and testing set commonly used in the batch learning setting, as well as the holdout and predictive sequential evaluation approaches in used in online learning, which are described in Section 5.1. Notably, many methods, including the ones introduced in Chapter 4, optimize a heuristic function throughout the learning process. This heuristic function is generally not the same as the evaluation measure $\mathcal{M}$.

The remainder of this section is structured as follows. First, we define the supervised predictive modeling tasks, starting from the fundamental tasks, defined on primitive target data types, classification and regression, and proceeding to structured output prediction tasks, such as multi-target regression and multi-label classification. The latter are described in individual sections, where we provide, in addition to the task definition, a verbose description and some examples. In the following section, we describe semi-supervised predictive modeling tasks.

Throughout the following task definitions we look at two illustrative problems, one from the natural sciences and another from engineering, and explain how they can be formalized as a predictive modeling task. The problems we will investigate are habitat modeling and fault detection. In the former, we are interested which organisms (usually animals) live in a certain habitat, based on the habitat's properties. In the latter, we are interested in predicting when a fault will occur in a system, e.g., in a wind turbine due to mechanical stress.

(a)

A_1	A_2	A_3	A_4	T
15.7	0.00	1	tall	red
42.0	0.27	0	short	blue
23.9	0.81	0	tall	green

(b)

A_1	A_2	A_3	A_4	T
15.7	0.00	1	tall	61
42.0	0.27	0	short	33
23.9	0.81	0	tall	27

Table 2.2: Sample datasets for classification (a) and regression (b). A_1 and A_2 are `real` attributes. A_3 is a `boolean` attribute, represented in the $\{0, 1\}$ notation. A_4 is a `discrete(tall, short)` attribute. In (a), T is a `discrete(red, blue, green)` target. In (b), T is a `real` target.

2.2.1.1 Classification and regression

The most fundamental and most commonly addressed supervised predictive tasks are classification and regression. These tasks are *single-target* tasks, i.e., for each data example a single value must be predicted and Y is one dimensional. In other words, the target data type is primitive, i.e., `boolean`, `discrete` or `real`.

Classification is the task of predicting a single nominal value based on the values of the descriptive attributes, i.e., the target data type is finite. In classification, individual examples belong to a single *class*. The values we use to represent classes are called *labels*, i.e., an example is of a given class, if its y value is the corresponding label. In a way, a label is a succinct representation of the class, e.g., the word *bird* can be seen as a label for the concept of birds.

The simplest form of classification is *binary classification*[6], where the target data type is `boolean`, and usually represents the presence of some phenomenon function, i.e., value $\top \in Y$ means the phenomenon is observed, while $\bot \in Y$ denotes its absence[7]. When there are more than two classes, i.e., when the data type is `discrete`, the task is called *multi-class classification*. For classification, we most commonly use accuracy, precision, recall, F^1 or similar evaluation measures (Kononenko & Kukar, 2007). A sample dataset that can be used in a classification task is shown in Table 2.2a.

Regression, on the other hand, is the task of predicting a continuous value based on the values of the descriptive attributes, i.e., Y is a continuous space, usually $\mathbb{R}$ or some interval $[a, b] \subseteq \mathbb{R}$. This is the case when the target data type is `real`. For regression, we commonly use another set of evaluation measures, such as mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), relative root mean squared error (RRMSE), and so on (Witten et al., 2016). A sample dataset that can be used in a regression task is shown in Table 2.2b.

Illustrative examples. Let us observe our sample real-world problems through the lens of classification and regression. The simplest form of habitat modeling is determining whether a specific organism can live in a given environment, i.e., based on the description of the environment we predict $\top$ if the organism can live in the environment and $\bot$ if it can not. In this case, we have formalized the habitat modeling problem as a binary classification task. Let us broaden our interest and ask instead what is the population of the organism

[6] Historically, this task is also known under the name *concept learning*, though, in concept learning a greater emphasis is placed on modeling the positive examples, i.e., those labeled by $\top$.

[7] These values are usually represented in the $\{0, 1\}$ representation as discussed earlier.

in a given environment, i.e., we predict the population based on the description of the habitat. Thus we have formalized the task as regression. The multi-class formalization of habitat modeling is slightly different. Let us say that the different classes correspond to different organisms. Then, the task might be to predict which of the available organisms is best suited to the given habitat.

In the fault detection scenario, the classification task can be predicting whether a given example represents a faulty wind turbine or not, while a regression task might be predicting the probability of failure or the cost of repairs for a given wind turbine. In a multi-class classification task, the turbine might be segmented into several components and the task would be to predict which of the components (if any) is faulty.

The above examples illuminate two things. The first is that predictive modeling tasks are related and can often be transformed from one to the other. In the first example, we extended binary classification to regression, i.e., $\perp$ (the organism is not present) is equivalent to 0 (no population), while $\top$ (the organism is present) is a more descriptive representation as it provides additional information about the population. The reverse is also possible, starting from the regression task, we can define some thresholds for the population which serve as delineations between low, medium or high population. These are then the classes of a multi-class classification task. *Problem transformation* approaches are often used to address a predictive modeling task using a method that cannot be directly applied to it, but to a different, related predictive modeling task.

The second thing we notice in the above examples is that multi-class classification is very "one-dimensional", i.e., in the case of a faulty wind turbine with multiple faulty components a predictive model can only predict that one of the components is faulty. To address this shortcoming of multi-class classification, we must look at complex target spaces, i.e., more complex data types on the target side.

When the target data type is an aggregate data type, composed from one or more component data types, we are dealing with *multi-target* or, more generally, *structured output prediction* (SOP) tasks. Structured-output prediction tasks are more complex, because for each example a more complex data type must be predicted, specifically, each of its component data types must be predicted. In this scenario, we refer to all components of the target data type as targets. Furthermore, when we consider SOP tasks, the target data type can also take structures more complex than a `tuple` or `set`, e.g., it can be a one- or multi-dimensional time-series (Slavkov & Džeroski, 2010) or it can be structured hierarchically (Vens et al., 2008). Several structured output prediction tasks are defined below.

2.2.1.2 Multi-target regression

Multi-target regression (MTR) is the task of predicting multiple continuous values. The target data type is a tuple of two or more reals, i.e., `tuple(real, real, ...)` and the target space is $\mathbb{R}^N$, where N is the number of the tuple's components. A sample dataset for use in a multi-target regression task is shown in Table 2.3. Regular, single-target, regression can then be seen as a special case of multi-target regression.

It is important to note that when we deal with a multi-target regression task, the targets are assumed to be equally important. If there are any differences in the importance of the targets, they can generally be encoded in the evaluation measure $\mathcal{M}$.

Note that, when evaluating and comparing machine learning methods for multi-target regression, we tend to observe measures which take into account normalized values over the different targets. This ensures that the evaluation is not biased towards, e.g., targets which have larger mean values. However, in a practical example of a multi-target regression

A_1	A_2	A_3	A_4	T_1	T_2	T_3
15.7	0.00	1	tall	61	0.746	594.3
42.0	0.27	0	short	33	0.288	414.4
23.9	0.81	0	tall	27	0.031	280.0

Table 2.3: Sample dataset for multi-target regression. Attributes A_i are as before. T_1, T_2 and T_3 are `real` targets.

task, where the actual evaluation measure $\mathcal{M}$ is known, there is no need for normalized measures, as the relative importances of the features are encoded in $\mathcal{M}$.

On the topic of evaluation measures, in multi-target regression we tend to use similar measures to single-target regression. The measures are often aggregated over the targets, although we must be careful when aggregating non-normalized measures. For more details on multi-target regression measures, see Section 5.2.1.

Illustrative examples. The sample problems are easily extended from the single-target regression scenario. Instead of predicting the population of a single organism in a given habitat, we can now predict the populations of several organisms in the habitat. Similarly, in the fault detection scenario, we can divide the turbine into components and try to model the probability of faults in individual components instead of the turbine as a whole.

2.2.1.3 Multi-label classification

In binary or multi-class classification, only one of the possible classes needs to be predicted. The task of *multi-label classification* (MLC) generalizes over the task of multi-class classification and each example can now belong to zero, one or more classes, i.e., it can be assigned a combination (subset) of all possible classes or labels (Iman & Davenport, 1980). In multi-label classification, the target data type is `set(discrete(`$\mathcal{L}$`))` over some labelset $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots, \lambda_K\}$ and the target space Y is the powerset of the labelset $\mathcal{L}$, i.e., $Y = \mathcal{P}(\mathcal{L})$. Explicitly, we refer to the individual labels as $\lambda_k \in \mathcal{L}$, while members of the target space, actual values and predictions, are referred to as *actual* or *real* and *predicted labelsets* and denoted by $y \in Y$ and $\hat{y} \in Y$, respectively. In multi-class classification, each label λ_k is directly a member of the target space.

As we have discussed earlier, we most commonly represent this kind of data type, i.e., `set(discrete(`$\mathcal{L}$`))` as columns, one corresponding to each label. If an example takes the value 1 for a given column, the corresponding label is present, while if it takes the value 0, the label is not present. This representation also shows us that multi-label classification is equivalent to multi-target binary classification, i.e., a task where the target data type is `tuple(boolean, boolean, ...)`. For this reason, we sometimes refer to the labels as individual targets similar to the `real` targets in multi-target regression, especially, when we discuss multi-label classification in the broader context of structured output prediction. See Table 2.4 for a sample dataset that can be used for a multi-label classification task.

Illustrative examples. In the habitat modeling example, multi-label classification naturally extends binary classification. Instead of being interested in whether the habitat can support a given species, we are interested whether it can support multiple species. Similarly, in the fault detection scenario, the labels correspond to different components that are at risk of failure.

A_1	A_2	A_3	A_4	T_{red}	T_{blue}	T_{green}
15.7	0.00	1	tall	1	0	1
42.0	0.27	0	short	0	0	1
23.9	0.81	0	tall	1	1	0

Table 2.4: Sample dataset for multi-label classification. Attributes A_i are as before. T_{red}, T_{blue} and T_{green} are components of a `set(discrete(red, blue, green))` target.

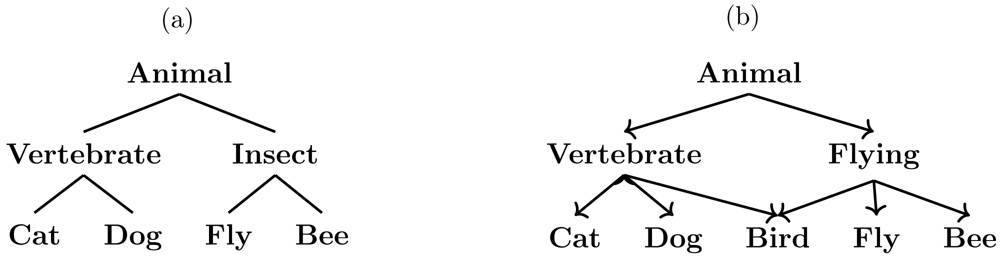


Figure 2.2: Sample (a) tree hierarchy and (b) DAG hierarchy of animals.

Similarly as with regression and binary classification, it is possible to transform a multi-label classification task into a multi-target regression task (for details see Section 4.3).

As with multi-target regression, all labels are assumed to be equally important, which is evident from the evaluation measures that are generally used in multi-label classification. The evaluation measures for multi-label classification are described in detail in Section 5.2.2.

2.2.1.4 Hierarchical prediction tasks

We define two hierarchical prediction tasks that extend the tasks defined above, hierarchical multi-label classification (HMLC) and hierarchical multi-target regression (HMTR). In both, the individual labels or targets are arranged into a hierarchy.

The hierarchy is generally represented by a graph and we distinguish between tree hierarchies and directed acyclic graph (DAG) hierarchies. In a *tree hierarchy*, each label/target only has one parent, while in a *DAG hierarchy* a label/target can have multiple parents. For example, in Figure 2.2a each animal only has one parent, vertebrate or insect, while in Figure 2.2b, bird is both a vertebrate and a flying animal.

We start with the definition of hierarchical multi-label classification (HMLC), as it is, among the two hierarchical tasks, the one that has received more attention in the literature (Silla & Freitas, 2011). A problem we encounter is that none of the data types we defined in Section 2.1 are able to encode a hierarchy. The simplest way to represent a multi-label classification hierarchy with the available data types is to use nested `boolean`s and `tuple`s. For example, the nested data type of the sample hierarchy shown in Figure 2.2a is shown in Figure 2.3a.

A label is called a *leaf label* if it has no children and is represented as a `boolean`, otherwise it is a `tuple(boolean,` *children*`)`, where the `boolean` corresponds to the label itself and *children* is a `tuple` of the node's children defined in the same way[8].

[8] To encode directed acyclic graphs, one must perform some additional notational gymnastics, which we

(a)

$$
\begin{array}{l}
\texttt{tuple(}\underbrace{\texttt{boolean}}_{\texttt{animal}}\texttt{,}\\
\qquad\texttt{tuple(}\\
\qquad\qquad\texttt{tuple(}\underbrace{\texttt{boolean}}_{\texttt{vertebrate}}\texttt{,}\\
\qquad\qquad\qquad\texttt{tuple(}\underbrace{\texttt{boolean}}_{\texttt{cat}}\texttt{, }\underbrace{\texttt{boolean}}_{\texttt{dog}}\texttt{),}\\
\qquad\qquad\texttt{),}\\
\qquad\qquad\texttt{tuple(}\underbrace{\texttt{boolean}}_{\texttt{insect}}\texttt{,}\\
\qquad\qquad\qquad\texttt{tuple(}\underbrace{\texttt{boolean}}_{\texttt{fly}}\texttt{, }\underbrace{\texttt{boolean}}_{\texttt{bee}}\texttt{)}\\
\qquad\qquad\texttt{)}\\
\qquad\texttt{)}\\
\texttt{)}
\end{array}
$$

(b)

$$
\texttt{tuple(}\underbrace{\texttt{boolean}}_{\texttt{animal}}\texttt{, }\underbrace{\texttt{boolean}}_{\texttt{vertebrate}}\texttt{, }\underbrace{\texttt{boolean}}_{\texttt{cat}}\texttt{, }\underbrace{\texttt{boolean}}_{\texttt{dog}}\texttt{, }\underbrace{\texttt{boolean}}_{\texttt{insect}}\texttt{, }\underbrace{\texttt{boolean}}_{\texttt{fly}}\texttt{, }\underbrace{\texttt{boolean}}_{\texttt{bee}}\texttt{)}
$$

Figure 2.3: The (a) nested data type and (b) flat data type of the hierarchy from Figure 2.2a.

The above example clearly shows that this representation is somewhat convoluted. When we store data in a table, we can use a flat representation, in which the labels are listed as they would be in a multi-label classification, i.e., for the hierarchy in Figure 2.2a we would use the data type in Figure 2.3b instead.

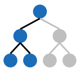
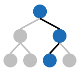
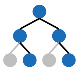
Hierarchical multi-label classification is the predictive modeling task, where the target is defined by a hierarchical `boolean` data type, defined as above. An additional property of hierarchical multi-label classification is the *hierarchy constraint.* The hierarchy constraint is satisfied when, for each label that is present in an example, all its ancestors are also present. This ensures that labels lower in the hierarchy are refinements of their ancestors, e.g., in Figure 2.2a a cat is a vertebrate which is a type of animal.

Illustrative examples. The sample problems are naturally extended using hierarchies. We can extend the habitat modeling problem to taxonomically cover the animal kingdom, utilizing the fact that taxonomically similar animals tend to have similar habitat needs. On the other hand, in the fault detection problem, we can group different components according to their proximity, to capture whether a particular part of the machine is faulty, i.e., whether the effects of components which are individually not yet faulty, combine to yield a faulty machine.

Hierarchical multi-target regression (HMTR) is a hierarchical variant of multi-target regression. Similar to the task of hierarchical multi-label classification, the `real` targets are arranged in a hierarchy. To represent the `real` hierarchical data type, we use the same nested `tuple` structure shown in Figure 2.3a, with the exception of `boolean` data types which are replaced by `real` data types. A sample dataset that can be used for the task of

omit for brevity.

(a)

A_1	A_2	A_3	A_4	$T_{\text{hierarchical}}$
15.7	0.00	1	tall	
42.0	0.27	0	short	
23.9	0.81	0	tall	

(b)

A_1	A_2	A_3	A_4	T_{animal}	$T_{\text{vertebrate}}$	T_{cat}	T_{dog}	T_{insect}	T_{fly}	T_{bee}
15.7	0.00	1	tall	1	1	1	1	0	0	0
42.0	0.27	0	short	1	0	0	0	1	1	0
23.9	0.81	0	tall	1	1	1	0	1	0	0

Table 2.5: (a) Hierarchical and (b) flat representations of a dataset for hierarchical multi-label classification. Attributes A_i are as before. In (a), $T_{\text{hierarchical}}$ represents the entire hierarchy, where blue nodes are present, while gray nodes are absent. In (b), T_{animal}, $T_{\text{vertebrate}}$, T_{cat}, etc., are individual labels in the hierarchy.

hierarchical multi-target regression is shown in Figure 2.6, again using the hierarchy from Figure 2.2a.

In hierarchical multi-target regression, the hierarchy constraint is not as straightforward as in hierarchical multi-label classification. In HMLC, the value of a non-leaf label, if not individually defined, is determined by the values of its children. If any of its children are present, according to the hierarchy constraint so must be the observed label. In other words, the label is present if we take the disjunction of its children's labels, i.e.,

$$\text{label } \lambda \text{ is present} \iff \bigvee_{\lambda' \text{ is a child of } \lambda} \big(\text{label } \lambda' \text{ is present}\big)\,.$$

Thus, the observed label's presence is an aggregate of the presences of its children. This prompts us to define the hierarchy constraint in hierarchical multi-target regression in a similar way. A non-leaf target is then assumed to have a value that is an aggregate of its children's values. The aggregate can be a sum, minimum, maximum, etc. However, due to the targets being continuous, the enforcement of the hierarchy constraint is not as simple as in HMLC. Instead of expecting the aggregate values to be matched exactly, we instead expect the predictions to be as close to the aggregate values as possible according to the measure $\mathcal{M}$.

Illustrative examples. The sample problems follow the same conversions as in hierarchical multi-label classification. In habitat modeling, each target might correspond to a taxonomic class and its value might be the population of all representatives of that class. On the other hand, in fault detection, we can use the same spatial hierarchy as in hierarchical multi-label classification to obtain hierarchically ordered components of the wind turbine. Each target then corresponds to a probability of failure of any components lower than it in the hierarchy.

A_1	A_2	A_3	A_4	T_{animal}	$T_{\text{vertebrate}}$	T_{cat}	T_{dog}	T_{insect}	T_{fly}	T_{bee}
15.7	0.00	1	tall	458	16	16	0	442	430	12
42.0	0.27	0	short	291	39	0	39	252	71	181
23.9	0.81	0	tall	42	22	5	17	20	13	7

Table 2.6: Sample dataset for hierarchical multi-target regression. Attributes A_i are as before. T_{animal}, $T_{\text{vertebrate}}$, T_{cat}, etc., are individual `real` targets in the hierarchy. The non-leaf targets are aggregated by summing.

2.2.1.5 Semi-supervised prediction tasks

In many application settings, recording attribute values is considerably cheaper than recording the values of the targets. Sometimes, this is due to the need for a human being to actually provide the proper target values. For example, in sentiment prediction, where tweets are labeled as negative, neutral or positive, a person must manually label each tweet.

These circumstances often lead to a situation where there is a large body of records for which the attributes are known, but the targets are not. These examples are called *unlabeled examples*. Given their abundance and low cost of recording, it is prudent to utilize these examples to the maximum possible extent. Towards this end, we speak about the task of semi-supervised learning, i.e., we define semi-supervised predictive modeling data mining tasks. In these tasks, the target component of each example may or may not be present, and approaches which address semi-supervised predictive modeling are expected to utilize all of the examples, regardless whether they are labeled or not.

Formally, if we have a supervised predictive modeling task, where the target data type is T, we can produce a semi-supervised predictive modeling task, by replacing the target data type with `choice(`T`, void)`. A sample dataset that can be used in the semi-supervised multi-target regression task is shown in Table 2.7. Note, that when the example is unlabeled, we say that all of the targets are equal to `?`, even though, technically, the example's entire target is only one `?`.

For the evaluation of semi-supervised predictive modeling tasks, we use similar measures as we do in the corresponding supervised predictive modeling tasks. Most of the latter are only able to be calculated on labeled examples. For more details see Section 5.3.

We can see that the degree of supervision is about how much information is available about the targets. In supervised tasks, all of the values are present, while in semi-supervised tasks, the target value can be missing. In the unsupervised task of clustering, which we will define below, there is no information about the targets. In fact there are no targets to begin with, as we are not looking for a dependence between the attributes and the targets, the task is to group similar examples together.

There is, however, another family of predictive modeling tasks, which fall between unsupervised and semi-supervised tasks, but it exists as separate from semi-supervised learning only in the case of structured output prediction. These are the *partially-labeled* predictive modeling tasks. Instead of an example having a missing value of the entire target, in partially-labeled tasks each individual target can be missing.

For example, if we start with a multi-target regression task with three targets, the corresponding target data type is `tuple(real, real, real)`. In semi-supervised multi-target regression, the target data type is `choice(tuple(real, real, real), void)`. However, in partially-labeled multi-target regression, each target component can be missing individ-

A_1	A_2	A_3	A_4	T_1	T_2	T_3
15.7	0.00	1	tall	61	0.746	594.3
42.0	0.27	0	short	?	?	?
23.9	0.81	0	tall	27	0.031	280.0

Table 2.7: Sample dataset for semi-supervised multi-target regression. Attributes A_i are as before. T_1, T_2 and T_3 are `real` targets. The second example is unlabeled.

A_1	A_2	A_3	A_4	T_1	T_2	T_3
15.7	0.00	1	tall	61	0.746	594.3
42.0	0.27	0	short	33	?	?
23.9	0.81	0	tall	?	0.031	280.0

Table 2.8: Sample dataset for partially-labeled multi-target regression. Attributes A_i are as before. T_1, T_2 and T_3 are `real` targets. The second and third examples are partially-labeled.

ually, yielding a target data type of `tuple(choice(real, void), choice(real, void), choice(real, void))`. A sample dataset that can be used for the task of partially-labeled multi-target regression is shown in Table 2.8.

Trivially, when there is only one target, the definitions of semi-supervised tasks and partially-labeled tasks coincide. It is also easy to see, that partially-labeled tasks generalize over semi-supervised tasks, in that each example that is missing its entire target value is missing also all of its components.

2.2.2 Clustering

In *clustering* we wish to group similar examples of a dataset together into *clusters*. In addition to a high degree of similarity among the members of a cluster (high intra-cluster similarity), we generally wish that the clusters are different among themselves (low inter-cluster similarity). However, we do not know in advance which clusters we will find in a dataset, and, generally, we are not even sure of their number. Practically, though, we often intentionally fix the number of clusters in advance as many methods find the best clustering with a prescribed number of clusters.

Formally, in clustering we learn a mapping $c :: X \to \mathbb{N}_k$ or $c :: X \to \mathbb{N}$, where $\mathbb{N}_k$ and $\mathbb{N}$ are the sets of the first k natural numbers and the set of all natural numbers, respectively, depending if we prescribe the number of clusters in advance or let the clustering method determine the number of clusters, respectively. Then, examples $x_1, x_2 \in X$ belong to the same cluster if $c(x_1) = c(x_2)$. The values that c maps examples into are called the enumerations of the clusters. Specifically, we do not treat the enumerations as ordered, but only as unique cluster identifiers. In this way, clusters are defined by their examples and not by the number that is used to represent them[9]. Notably, the mapping into $\mathbb{N}_k$ or $\mathbb{N}$ is external, i.e., it maps to a set that is not defined by $\mathcal{D}$, unlike in predictive modeling where the predictive function maps from one subspace of $\mathcal{D}$ into another.

[9] A clustering therefore does not change if we permute the values it maps into.

The previous definition defines the most basic task of clustering, however, we sometimes deal with clustering where we search for *overlapping clusters.* In this task, each example belongs to one *or more* clusters. Thus, we learn a mapping $c_o :: X \to \mathcal{P}(\mathbb{N}_k)$ or $c_o :: X \to \mathcal{P}(\mathbb{N})$. Furthermore, some methods produce *hierarchical clusterings*, where the clusters are arranged hierarchically. This means that if a cluster k is a subcluster of cluster j, this implies that for each $x \in X$ in cluster k, x also belongs to cluster j. Additionally, if an example belongs to cluster k, it belongs to at most one of it subclusters. Naturally, a hierarchical clustering is an overlapping clustering, but not each overlapping clustering is necessarily hierarchical.

2.2.3 Pattern mining

Pattern mining is different from predictive modeling and clustering in the sense that the mapping we are trying to learn is not necessarily linked to the individual examples in the dataset, i.e., we are not necessarily learning a mapping from the domain (or descriptive space) X to another set. Instead, a set of interesting *patterns* is first generated from the dataset, optionally restrained by some constraints, after which each of the patterns is evaluated. The type of patterns we generate is called the *pattern language* $\mathbb{L}$; for example, we can generate association rules from sets of items (Agrawal, Imieliński, & Swami, 1993), or we can generate patterns that correspond to individual attributes in the case of feature ranking.

Formally, we will denote the set of interesting patterns from some language $\mathbb{L}$ on a dataset $\mathcal{D}$ as $\Pi_{\mathbb{L}}(\mathcal{D})$. Notably, the patterns might refer to the individual examples, the attributes of the dataset or potentially both. Pattern mining can then be formalized as the task of learning a mapping $p :: \Pi_{\mathbb{L}}(\mathcal{D}) \to S$, where S is some comparison set. Based on what we choose as S, we can differentiate several pattern mining subtasks:

- When $S = \{\top, \bot\}$, we name the subtask *pattern selection.* In essence, we are choosing which of the patterns are important (when $p(\pi) = \top$, for some $\pi \in \Pi_{\mathbb{L}}(\mathcal{D})$) and which are not;
- When $S = \mathbb{R}$, the subtask is called *pattern importance.* Here, each pattern is scored using a mapping p and we assume that patterns with higher scores are more important;
- Sometimes we wish to rank the patterns without explicitly calculating the scores. In that case, $S = \mathbb{N}_{|\Pi_{\mathbb{L}}(\mathcal{D})|}$ and p maps each pattern to its rank. Then, patterns with a lower rank are more important. This is the task of *pattern ranking.* In addition to defining S as above, we also desire that p is bijective, i.e., that each pattern maps into a distinct rank. However, this is not a strict requirement, as sometimes we allow multiple patterns to occupy the same rank.

It is fairly easy to obtain a pattern ranking p_r from pattern importance scores p_i by ordering the patterns π according to their scores $p_i(\pi)$ and noting their ranks according to this ordering. Similarly, producing a pattern selection p_{sel} from a pattern importance scores p_i is also not difficult. A pattern π is mapped to $\top$ if its score $p_i(\pi)$ is greater than some threshold τ, i.e., $p_{\text{sel}}(\pi) = \top \Leftrightarrow p_i(\pi) > \tau$. Finally, a pattern selection p_{sel} can also be obtained from a pattern ranking p_r by selecting the top k patterns. Notably, producing a feature importance scores from a feature ranking is meaningless. In pattern scoring we are explicitly interested in the differences of the scores of the individual patterns, while in pattern ranking, we are only interested in determining only whether a pattern is better than one or more others, but not by how much.

In the context of this thesis, we will address one group of pattern mining tasks. These are *feature importance*, *ranking* and *selection*, though we will generally refer to all of the tasks as feature ranking as is common in the literature[10]. In the context of feature ranking for predictive modeling, we follow the same division of the domain into the descriptive space X and the target space Y[11]. The language we use in feature ranking is the feature language $\mathbb{F}$, i.e., the descriptive attributes, which are components of the descriptive space X. The individual attributes are then scored, ranked or selected based on how important they are for a potential predictive model. An attribute for which a change in the value often means a change in the target y is considered to be important, while an attribute for which a change does not often impact y is not.

Suppose that the descriptive space X is composed of N descriptive attributes, i.e., $X = X_1 \times X_2 \times \cdots \times X_N$, and we have a dataset $\mathcal{D} = \left\langle \left(\left(x_1^\ell, x_2^\ell, \dots, x_N^\ell \right), y^\ell \right) \mid \ell = 1, 2, 3, \dots \right\rangle$, where ℓ indexes the examples. The patterns in the feature language $\mathbb{F}$ are then

$$\Pi_{\mathbb{F}}(\mathcal{D}) = \left\{ \pi_i = \left\langle \left(x_i^\ell, y^\ell \right) \mid \ell = 1, 2, 3, \dots \right\rangle \mid i = 1, 2, \dots, N \right\}.$$

If $\mathcal{D}$ is a set or a bag, then so is each π_i. If $\mathcal{D}$ is a sequence, each π_i is a sequence, i.e., $\pi_i = ((x_i^1, y^1), (x_i^2, y^2), (x_i^3, y^3), \dots)$. Note that we retain each attribute's index i so it is easier to distinguish between the components of the actual representation of the dataset (e.g., columns of a table).

It is important to note that, while the mapping we wish to find only maps from patterns into an evaluation set, the approaches that address the feature ranking task have access to the entire dataset. This means that some approaches for feature ranking also take into account how each attribute impacts the target values in combination with other features. These types of approaches are called *multivariate* approaches for feature ranking. On the other hand, approaches that only consider values of the attribute in question when producing its evaluation are called *univariate* (or myopic) approaches.

Feature ranking is especially useful when we have a lot of attributes and we wish to find out which really impact the targets. This can be beneficial for a multitude of reasons. Predictive models which are based on a fewer number of attributes are more interpretable than those that take into account a larger amount. Additionally, learning predictive models based on fewer attributes as well as using them requires fewer resources, i.e., the models are learned and used faster with a lower requirement for memory. Furthermore, recording some of the attributes might be more expensive than others. Feature ranking allows us to decide whether this expense is worth it.

Feature ranking also allows us to identify redundant or dependent attributes. An attribute is redundant when the same information is encoded in one or more other attributes, while an attribute whose importance depends on the presence of other attributes is called a dependent attribute.

Sometimes, we also perform feature ranking despite not being directly interested in the predictive modeling aspect, i.e., we might not be interested in actually learning a predictive model. In cases like these, we are more interested which attributes, representing physical quantities, are more important to a system we are observing. This gives a more qualitative view of the system, without the need for building a predictive model.

[10] Throughout this thesis, we predominantly use the term *attribute* instead of *feature*, when describing the descriptive properties of each example. However, we will still use the terms *feature importance*, *feature ranking* and *feature selection*, as they are established terms in the literature.

[11] Notably, there also exists the related task of feature ranking for unsupervised data mining tasks.

2.3 Classical Data Mining and Data Stream Mining

Orthogonal to the data mining task we are addressing, we are also concerned with the learning setting. We commonly differentiate between two learning settings, the *classical* or *batch learning* setting and the *online learning* setting, sometimes also referred to as the *stream mining* setting. First, we discuss the batch learning setting and batch methods that are generally used to address tasks in this setting. We continue by discussing the online learning setting and contrast incremental methods that are used primarily in the online learning setting with batch methods.

2.3.1 The batch learning setting and batch methods

In the batch learning setting, the entire dataset is available at the start of the learning process and the order of the examples in the dataset is generally assumed not to have an impact on the learning process. Consequently, in batch learning the dataset is most commonly a bag of examples. Additionally, in batch learning all of the examples in the dataset $\mathcal{D}$ are assumed to be sampled from the same underlying probability distribution D. As such, any learned generalization is assumed to be applicable for new, future examples. Notably, until the learning process is complete we are unable to use the generalization, e.g., in predictive modeling, we cannot make predictions until all of the data examples have been processed, i.e., the model is fully learned.

A *batch learning method* for a given data mining task is therefore a mapping $\mathbb{M}$ that maps a dataset $\mathcal{D}$ into a task-appropriate generalization. A given approach produces generalizations of a specific formalism, e.g., a tree-based learning approach will produce a decision tree. The set of all possible generalizations that a method $\mathbb{M}$ can produce (i.e., can map into) is called the *hypothesis space* $\mathcal{H}_{\mathbb{M}}$. Then $\mathbb{M} :: \mathbb{D} \rightarrow \mathcal{H}_{\mathbb{M}}$, where $\mathbb{D}$ is the space of all possible datasets. In essence, a batch learning method is a mapping that takes as input a dataset $\mathcal{D}$ (from $\mathbb{D}$) and produces a generalization, e.g., a predictive model m (from $\mathcal{H}_{\mathbb{M}}$), i.e., $\mathbb{M}(\mathcal{D}) = m$, where $\mathcal{D}$ and m are the collection of examples and predictive model, respectively, as earlier. As the name implies, batch learning methods are the de facto methods used (but not the only ones) in the batch learning setting.

2.3.2 The online learning setting and incremental methods

In online learning, the entire dataset is not available at the start of the learning process. Instead, we are learning from a *data stream*, an ordered sequence of examples which become available throughout the learning process. This also means that there is an inherent time component to online learning, i.e., one example arrives either before or after another example. The datasets in online learning are naturally ordered sequences of examples.

A data stream can theoretically be arbitrarily long, and methods that are applied in the online learning setting are expected to allow for this possibility. Consequently, the size of even a finite subsample of the data stream cannot be expected to fit into the processing memory of any given machine on which the learning is occurring. To avoid the need for an arbitrarily large memory storage, each example is processed only once at the time of its arrival. Afterwards, it is generally discarded, although a select number of examples may be archived for further use.

Furthermore, we expect any generalization to be applicable in real-time or near real-time, i.e., at any point throughout the learning process the current generalization can be applied in whatever way is appropriate for the data mining task at hand. The generalization is expected to be as up to date as possible, i.e., it should incorporate information from all relevant examples available up to this point in time. To allow the most current generalization to be kept updated, the processing of each individual example should therefore be quick. Ideally, an online learning method should completely process an example before the next one arrives.

Above, we stated that information from all relevant examples should be incorporated to keep the generalization as up to date as possible. In particular, in online learning not all examples are necessarily relevant as the underlying distribution D governing the examples can change. Thus, examples that were recorded prior to the change in distribution are not necessarily relevant any longer. When the distribution changes, we talk about *concept drift* (Gama, 2010), where the *concept* refers to the distribution itself or to whatever patterns we have defined on it, i.e., the concept is what a generalization is aiming to approximate. Approaches for online learning therefore do not assume that the distribution D is stationary, but that it can change. Towards this end approaches for stream mining utilize *change detection* mechanisms, such as the Page-Hinckley test (Mouss, Mouss, Mouss, & Sefouhi, 2004) or adaptive windows (Bifet & Gavaldà, 2009).

To summarize, learning from data streams is subject to several constraints:

- The examples arrive one-by-one in a fixed order.
- There can be an arbitrarily large number of examples.
- Each example is processed only once. Once the example is processed it is either discarded or, in select cases, archived.
- The underlying distribution D which governs the examples is not necessarily stationary.

To address several of the above constraints most online learning methods utilize incremental as well as decremental updating (Cauwenberghs & Poggio, 2001; Gama, 2010). *Incremental updating* refers to updating the current generalization by utilizing information available from new examples. For example, this includes updating statistics, as well as "growing" the current generalization. *Decremental updating*, on the other hand, allows for forgetting, i.e., discarding of information, when updating the current generalization. This is done to keep the generalization as relevant to the current time point as possible, i.e., favoring measurements from recent examples over those of older examples. Decremental updating also allows a learning method to lower the resource footprint of the current generalization, with regards to memory consumption or response time.

Addressing concept drift is a balance between incremental and decremental learning. On the one hand, when a change in the underlying distribution is detected, the learning method must learn the new concept as soon as possible, while on the other hand, it must also discard parts of the generalization which correspond to the old concept.

The definition of a method in the online learning setting is not as straightforward as in the batch setting, due to the need for generalization updating. In fact, a method cannot produce a single final generalization. To properly define a learning method in the online learning setting, we must define a *starting* or *default generalization* h_0 and an *update* or *refinement operator* u. The update operator takes a generalization and one or more example and produces a new, update generalization. In essence, the update operator *is* the method, as it completely defines how the current generalization changes with incoming examples.

If the update operator takes exactly one instance in addition to the current generalization, the method is called *instance incremental*, while if it takes one or more, i.e., a bag of examples, the method is *batch-incremental*. Therefore, the update operator is a mapping of the type $u :: \mathcal{H}_{\mathbb{M}} \times X \rightarrow \mathcal{H}_{\mathbb{M}}$ or $u :: \mathcal{H}_{\mathbb{M}} \times \mathcal{B}(X) \rightarrow \mathcal{H}_{\mathbb{M}}$, where X is the

Algorithm 2.1: Incremental learning on a dataset $\mathcal{D}$ in the batch learning setting.

```
Input: Dataset $\mathcal{D}$, incremental method $(\mathsf{h}_0, u)$
Output: Generalization $\mathsf{h}$ learned from entire dataset
$\mathsf{h} \leftarrow \mathsf{h}_0$;
while $\mathcal{D}$ has more examples do
    $\mathsf{e} \leftarrow \mathsf{NextExample}(\mathcal{D})$;
    $\mathsf{h} \leftarrow u(\mathsf{h}, \mathsf{e})$;
end
return $\mathsf{h}$
```

Algorithm 2.2: Batch learning on a dataset $\mathcal{D}$ in the online learning setting (update procedure).

```
Input: A new example $\mathsf{e}$, current generalization $\mathsf{h}$, batch method $\mathsf{M}$, window
       length $l$
Output: Updated generalization $\mathsf{h}$
$\mathsf{h.W.append(e)}$;
if $|\,\mathsf{h.W}\,| = l$ then
    $\mathsf{h} \leftarrow \mathsf{M(h.W)}$;
    $\mathsf{h.W} \leftarrow []$;
end
return $\mathsf{h}$
```

(complete) domain of a dataset $\mathcal{D}^{12}$, $\mathcal{B}(X)$ is the set of all bags of examples from X. In the case of an instance-incremental approach, the generalization after n examples is $u(\ldots(u(u(h_0, x_1), x_2), \ldots), x_n)$.

Note that an instance- or batch-incremental learning method can also be used to address data mining tasks in the batch learning setting, as is shown in Algorithm 2.1. For techniques of applying incremental methods to batch learning, see the work of Gallant (1986), Littlestone (1989), or Helmbold and Warmuth (1995). In particular, some of the currently most popular methods for predictive modeling in the batch learning setting, e.g., neural networks (Rosenblatt, 1958), are instance-incremental. If using an incremental method for machine learning in the batch setting, it is highly desirable for the method to be as agnostic of the ordering of the examples in a dataset as possible, i.e., the final generalization is expected to be similar, even if the order of examples is altered.

Conversely, a batch learning method can be used for incremental learning. Each incoming example is stored in a *window of examples.* Once enough examples accumulate in the window, we use them to learn a generalization using a batch method. That generalization then becomes the current generalization until the next one is learned. Afterwards, we empty the window and repeat the process for new incoming examples. On the first window, we use a placeholder generalization. This process is shown in Algorithm 2.2. This can also be seen as a batch-incremental approach. This approach does have some drawbacks, most notably, its use of resources is not evenly distributed, but peaks when the window is full and a new generalization is learned.

Incremental methods generally require significantly larger amounts of data than batch methods to achieve comparable performance of the learned models. This is expected, as batch methods have access to the entire dataset at once, much more information is available about each example, most notably, its relation to the other examples in the dataset. On the

[12] In the case of predictive modeling, X is replaced by $X \times Y$.

Addressed data mining task	Relevant sections
Online multi-target regression	Methods (4.2), evaluation (5.2.1), experiments (6.1)
Online multi-label classification	Methods (4.3), evaluation (5.2.2), experiments (6.2)
Online hierarchical prediction	Methods (4.4), evaluation (5.2.3), experiments (6.3)
Online semi-supervised multi-target regression	Methods (4.5), evaluation (5.3), experiments (6.4)
Online feature ranking for multi-target regression	Methods (4.6), evaluation (5.4), experiments (6.5)

Table 2.9: Overview of addressed data mining tasks and relevant sections (in parentheses).

other hand, incremental methods attempt to extract as much information as possible from each individual example, while also attempting to reconstruct the concept that generated it step by step. This also means that applying an incremental method in the batch learning scenario is not advised, unless the size of the dataset is substantial.

2.4 Addressed Tasks

Now that we have defined the requisite concepts and terminology, we list the data mining tasks that we address. They are all structured output prediction tasks in the online learning setting, with one exception. We address the following tasks (summarized in Table 2.9):

- **Online multi-target regression.** We introduce the iSOUP-Tree family of methods for learning model and regression trees, option trees and ensembles of trees for multi-target regression. The methods are described in Section 4.2. Appropriate evaluation measures are described in Section 5.2.1, while the experimental setup for MTR is described in Section 6.1.

- **Online multi-label classification.** We introduce the multi-label classification via multi-target regression problem transformation methodology that lets us utilize multi-target regressors for the task of multi-label classification. The methodology is described in Section 4.3. We use it in conjunction with the above-defined iSOUP-Tree methods for MTR in experiments described in Section 6.2. A plethora of measures of predictive performance for multi-label classification are described in Section 5.2.2.

- **Online hierarchical prediction.** We address both hierarchical multi-target regression and hierarchical multi-label classification, by modifying the splitting heuristic of the methods for the matching non-hierarchical tasks. We also propose an expansion of the MLC via MTR methodology for application to HMLC. The modified methods are introduced in Section 4.4, while the evaluation of hierarchical models and the corresponding experiments are described in Sections 5.2.3 and 6.3.

- **Online semi-supervised multi-target regression.** We use a modification of the iSOUP-Tree in the predictive clustering framework, iSOUP-PCT, to address online semi-supervised multi-target regression. While this method is experimentally evaluated specifically for the multi-target regression task, see Section 6.4, it can also be easily applied to other online SOP tasks. We discuss the appropriate evaluation methodology for semi-supervised tasks in the online learning setting in Section 5.3.

- **Online feature ranking for multi-target regression.** While not itself a predictive modeling task, feature ranking is closely intertwined with it. Here, we present an extension of a batch random forest-based method for feature ranking for various predictive modeling tasks, towards the online learning setting. The approach is described in Section 4.6, with the accompanying evaluation approaches in Section 5.4 and experiments in Section 6.5.

An aspect that we do not directly address in this thesis is the detection of concept drift and adaptation to the detected changes. Currently there are only a few online change detection methods in the SOP setting, however, they are specific for the online multi-label classification task (Spyromitros-Xioufis, 2011; Shi, Wen, Feng, & Zhao, 2014; Shi, Xue, Wen, & Cai, 2014). There are currently no directly applicable approaches for the detection of concept drift for the online multi-target regression task. Thus we leave the problem of change detection in the online structured output prediction setting for future consideration, while focusing on methods that generalize as best as possible on stationary data streams.

Chapter 3

Related Work

> To hear, one must be silent.
>
> — Ursula K. Le Guin

In this chapter, we present the related work that pertains to this thesis. We discuss the related work in three areas: data stream mining, batch structured output prediction and structured output prediction on data streams.

In the context of data stream mining, we present methods for online classification and online regression, methods for online drift detection, methods for online semi-supervised learning and methods for online feature ranking. The related work from the area of structured output prediction in the batch setting is divided into related work pertaining to each individual task, related work for semi-supervised structured output prediction and feature ranking for structured output prediction. Finally, we provide an overview of existing methods for structured output prediction on data streams. We conclude the chapter with closing remarks regarding the state of online methods for structured output predictions and by detailing the relevant earlier works that this thesis builds upon.

3.1 State of the Art in Single-Target Data Stream Mining

Unlike classical data mining, data stream mining has only become an actively investigated research area in the last three decades. Notably, incremental machine learning algorithms, such as neural networks (Rosenblatt, 1958) had been introduced much earlier.

Most of the early work in incremental learning is focused on the binary classification task. The first theoretical description and analysis of incremental algorithms as we describe them in this thesis, i.e., incrementally updating the hypothesis with new examples, is due to Maass (1991), who give theoretical results for binary classification. However, as we will see later, incremental methods had been introduced earlier. Alternative definitions, where an incremental learner is always given counter examples, i.e., examples on which it is wrong (Angluin, 1988), or where an incremental learner only receives feedback on positive examples (Helmbold, Littlestone, & Long, 1992) were also introduced.

A similar task to data stream mining has also been addressed in the field of computational intelligence under the name *evolving fuzzy systems* (Angelov, Lughofer, & Zhou, 2008; Angelov, Filev, & Kasabov, 2010). The major difference stems from the use of *fuzzy* sets, where, instead of being present or absent, members (attribute values) have degrees of presence.

Due to the nature of data stream mining, in particular, due to the inability to observe the entire dataset at once, many data stream mining algorithms utilize probabilistic estimates to learn. A particular class of these estimates falls into the category of *probably approximately correct* (PAC) learning introduced by Valiant (1984). Probably approximately correct learning constrains learning by defining a maximum probability δ of making errors of magnitude ε. This allows us to define a desired occurrence rate of errors of large magnitudes. Thus, many machine learning methods are based on probabilistic bounds and inequalities which enable PAC learning, such as the Chernoff bound (Chernoff, 1952), Hoeffding inequality (Hoeffding, 1963) and McDiarmid's inequality (McDiarmid, 1989). These are most commonly employed by methods that grow the model, i.e., where the model is an expanding data structure.

3.1.1 Methods for single-target classification and regression

Several methods for classification and regression that have been first introduced in the batch setting are naturally extendable to the online setting. Most notably, the naïve Bayes (Hand & Yu, 2001) method for classification and the perceptron (Rosenblatt, 1958) method for classification and regression are often used in combination with other online methods to increase their performance (Holmes, Kirkby, & Pfahringer, 2005; Bifet, Holmes, Pfahringer, & Frank, 2010; Ikonomovska, Gama, & Džeroski, 2011a).

Early methods

Most early work in the field of incremental learning addressed the binary classification task and used particularly rudimentary models, e.g., Widmer and Kubat (1996) use conjunctive descriptions on the input attributes for the description of positive, negative and potential patterns[1] in the FLORA framework that they introduce. This framework uses dynamically adjusted windows to learn the above descriptions and can store old concepts and reuse them.

Early tree-based methods include work by Schlimmer and Fisher (1986) and by Ichihashi, Shirai, Nagasaka, and Miyoshi (1996), which adapted the ID3 (Quinlan, 1986) batch decision induction algorithm to an online setting. Utgoff (1994) introduced the improved tree induction (ITI), with several constraints on the learning, particularly, that the incrementally learned tree should be the same as a tree learned in the batch setting and that tree updating should be computationally cheaper than relearning the tree. Black and Hickey (1999) and, later, Hickey and Black (2001) used a time-stamp attribute to modify the window of observed examples, rather than rely on a predefined size. They implemented the CD3, CD4 and CD5 variants of ID3 (Quinlan, 1986) for online classification.

Tree-based methods

As evidenced by the early methods, learning decision trees is particularly popular in the online setting. Most modern online-tree based algorithms are based on Hoeffding trees (Domingos & Hulten, 2000), which utilize the Hoeffding inequality to achieve PAC growth of the tree. Early improvements of Hoeffding trees allowed them to detect and adapt to concept drift (Hulten, Spencer, & Domingos, 2001) and learn from numeric attributes (Gama, Rocha, & Medas, 2003). Later, Holmes, Richard, and Pfahringer (2005) addressed the problem of split candidate ties in the tree growing procedure. Then, the focus shifted to model trees, decision trees that contain simple models in the leaves, and Hoeffding trees that utilize a naïve Bayes classifiers (Holmes, Kirkby, & Pfahringer, 2005) and perceptrons (Bifet, Holmes, Pfahringer, & Frank, 2010) were introduced.

[1] Note, that in their case they consider binary classification on exclusively nominal input attributes.

Potts and Sammut (2005) introduced incremental model trees which grow based on a heuristic which is optimized for placing linear regressors in the leaves of the tree. Ikonomovska, Gama, and Džeroski (2011a) introduced the fast induction of model trees with drift detection (FIMT-DD) for online regression based on the Hoeffding inequality and the variance reduction splitting heuristic. Later, Ikonomovska, Gama, Ženko, and Džeroski (2011) extended the FIMT-DD method by using option trees in the ORTO method, as well as using online bagging and online random forests (Ikonomovska et al., 2015). Similarly, Verbeeck and Blockeel (2015) introduced iRetis, which is an adaption of the batch model tree algorithm RETIS (Karalič, 1992), for online regression.

Kernel-based methods

Littlestone (1988) introduced a method based on linear separability only on the attributes which are relevant to the target class. Later, Yi Li and Long (2000) adapted support vector machines (Vapnik & Kotz, 1982) to the online setting, while Gentile (2001) introduced a method, which approximates the maximal margin hyperplane according to a selected $p \geq 2$ norm.

Instance-based methods

Salganicoff (1993) explore several ways of forgetting older examples in the case of the k nearest neighbors method in an online setting. Crammer and Singer (2003b) determine one characteristic input vector per class, then predict the class o the vector which is closest to an example vector, while Crammer, Kandola, and Singer (2004) later extend their work toward the explicit selection and discarding of instances. Recently, Shaker and Hüllermeier (2012) introduced an instance-based system for both online classification and online regression (IBLStreams).

Network-based methods

Last (2002) introduced the online information network (OLIN) method based on relearning an info-fuzzy network, which is a structure learned similar to decision trees, except the model is a directed network instead. OLIN periodically re-learns its model based on a dynamically adjusted window of examples. The window adjusts based on the observed amount of concept drift.

Ensemble methods

Oza (2005) adapted the classical ensemble methods of bagging, boosting and random forest (Breiman, 1996, 2001) to the online learning setting. The many methods that utilize ensembles for concept drift detection will be described below.

Reliability estimation of regression models

Some attention has also been devoted to estimating the reliability of the predictions of an arbitrary general regressor. Rodrigues, Bosnić, Gama, and Kononenko (2012) summarize their earlier work (Rodrigues, Gama, & Bosnić, 2008; Bosnić, Rodrigues, Kononenko, & Gama, 2011) in which they defined five measures for online empirical reliability estimation which were extended from the batch setting (Bosnić & Kononenko, 2008a, 2008b), based on a similarity kNN-based error, local bias and variance, and bagging bias and variance. They also present results on using these estimates to correct the predictions of the regressors, which yields improvements in predictive performance. Bosnić et al. (2014) use this methodology, as well as apply the methods of Štrumbelj and Kononenko (2010) to explain arbitrary online models for regression.

3.1.2 Detecting concept drift

In this section, we describe the general terminology pertaining to the detection of the concept drift. Additionally, we present the work that has been done on general change detection, i.e., methods that can be used in conjunction with classifiers or regressors. Note that many of the methods described above have specific drift detection mechanisms.

Tsymbal (2004) presented an early review of the methods for detecting concept drift, as well as properly defined many terms used by the community. Recent surveys by Gama et al. (2014) and Khamassi, Sayed-Mouchaweh, Hammami, and Ghédira (2016) detail modern approaches for detecting concept drift.

Concept drift is generally described along several dimensions. One dimension is whether the concept occurs *abruptly* or *gradually*, while another is whether the drift occurs over the entire input space, i.e., it is *global*, or present only on a part of the input space, in which case it is *local*. Furthermore, we differentiate between *real* and *virtual* concept drift. In *real* concept drift, the relationship between the input attributes and the targets changes, i.e., a linear relationship might change into a quadratic one. On the other hand, in *virtual* concept drift the relationship between the attributes and the targets remains the same, but the distribution governing the values of the attributes changes. For example, after virtual concept drift occurs, certain rare values of a nominal attribute might become common and vice versa. For an overview of the various types of concept drifts, see the surveys stated earlier. Note that most methods for concept drift detection are adapted for specific types of drifts, whether this is stated implicitly or explicitly.

We start our review of related work by mentioning some theoretical results. Most notably Basseville and Nikiforov (1993) present a thorough statistical analysis of abrupt changes, while Helmbold and Long (1994) provide theoretical results and error bounds for drifting concepts for the task of online classification. Recently, Webb, Hyde, Cao, Nguyen, and Petitjean (2016) proposed a quantitative measure of concept drift, however, it is only applicable to the online classification task with exclusively nominal input attributes resulting in a finite space of predictive models. This makes extensions towards other tasks, e.g., online regression, seem particularly difficult.

Below, we describe a selection of drift detection methods that can be used in conjunction with any given method for the task at hand. Notably, there are many methods for detecting concept drift that are specific to the underlying model, e.g., for support vector machines (Klinkenberg & Joachims, 2000; Klinkenberg, 2004). However, due to the existence of many model formalisms, we focus on generally applicable methods for drift detection.

As we will see, many drift detection methods are based on windows of examples, whether of static or dynamic size (Widmer & Kubat, 1996; Dong et al., 2003; Fan, 2004). Klinkenberg and Renz (1998) observe standard sample errors, in particular accuracy, precision and recall to for changes. Gama, Medas, Castillo, and Rodrigues (2004) introduce the drift detection methods (DDM), also known as statistical process control (SPC; Gama (2010)), which observes the overall error-rate of a general classifier as a Bernoulli trial approximated by a normalized distribution. It analyzes confidence intervals to dynamically adjust the time window. Notably, the approach does not achieve good results on gradual drifts and is specifically targeted toward model relearning instead of model adaptation. Baena-García et al. (2006) extend DDM toward detection of gradual drifts and, later, Shuo Wang et al. (2013) extend DDM for imbalanced data, by observing the recall measure of a minority class. Ditzler and Polikar (2011) introduced a batch-incremental change detector similar to DDM, which uses the symmetric and bounded Hellinger distance in place of the error rate.

Dries and Rückert (2009) use uniform convergence bounds from computation learning theory to detect drift. The approach of Harel, Mannor, El-Yaniv, and Crammer (2014) is based on permutation of an example's attribute values, i.e., resampling. As such, it can be applied to both classification and regression.

Dasu, Krishnan, Venkatasubramanian, and Yi (2006) use the Kullback-Leibler divergence measure from the field of information theory to determine whether two observed distributions are different. They observe past and present data distributions and, when they differ, they detect concept drift. Similarly, Sobhani and Beigy (2011) look at past and present distributions, however, they observe the nearest neighbors of an example on consecutive batches, as the approach is batch incremental. When the labels differ between batches, they have support for detecting concept drift. Ross, Adams, Tasoulis, and Hand (2012) use an exponentially weighted moving average to monitor the misclassification rate and, when it rises, detects concept drift. P. M. Gonçalves and de Barros (2013) introduced a method for detecting concept drifts which is able to identify recurring concepts, i.e., it saves past models and, when drift occurs, checks whether the concept has already been learned and reuses the corresponding model.

Recently, Jaka Demšar and Bosnić (2018) have used the interactions-based method for explanation (Štrumbelj, Kononenko, & Šikonja, 2009; Štrumbelj & Kononenko, 2010) to quantitatively describe what each input attribute contributed to the predicted label. They then applied the Page-Hinckley test or the SPC process to these descriptions to detect whether they remain stationary.

Many ensemble methods have been introduced to specifically address the problem of drift detection (Minku, White, & Yao, 2010; Minku & Yao, 2012; Stanley, 2003). In particular, Bifet and Gavaldà (2009) combined the online bagging ensemble approach with a concept drift detection mechanism based on adaptive windows (ADWIN) that relearns individual ensemble members if their predictions start to degrade. Furthermore, Gomes et al. (2017) combined the ADWIN change detector with the random forest methodology, using two differently sensitive change detectors for each ensemble model. The first, easily triggered change detector signals when a new alternate model needs to be learned, while the second, more specific change detector decides when the alternate model replaces the original model in the ensemble. Furthermore, Elwell and Polikar (2011) introduced the Learn++.NSE method, which is tested in combination with perceptrons, support vector machines and naïve Bayes classifier. Later, Ditzler and Polikar (2013) extended Learn++.NSE to Learn++.CDS using the synthetic minority class oversampling technique (SMOTE; Chawla, Bowyer, Hall, and Kegelmeyer (2002)) as well as introduced Learn++.NIE, which replaces the Learn++.NSE's raw classification accuracy with the weighted recall measure and the SMOTE technique with a variation of bagging that is sensitive to the relative appearance rate of each class. Finally, Brzezinski and Stefanowski (2014) introduced the accuracy weighted ensemble in which the weight of each base model is adjusted based on its accuracy on previous batches.

3.1.3 Methods for online semi-supervised learning

The first incremental method that is able to utilize unlabeled examples is the multi-view hidden Markov perceptron (Brefeld, Büscher, & Scheffer, 2005). Furao, Sakurai, Kamiya, and Hasegawa (2007) presented a method for semi-supervised learning based on a self-organizing incremental neural network, which was later extended by Shen, Yu, Sakurai, and Hasegawa (2011).

Additionally, Goldberg, Li, and Zhu (2008) proposed a manifold regularization method for semi-supervised learning that is based on a combination of convex programming with stochastic gradient descent. Later, Goldberg, Zhu, Furger, and Xu (2011) introduced OASIS, a Bayesian learning framework for semi-supervised learning. Recently, Sousa and Gama (2017) introduced a method for online semi-supervised single-target regression using AMRules (Duarte, Gama, & Bifet, 2016) based on the co-training approach to semi-supervised learning.

Online semi-supervised classification has also been used to tackle image tracking in an online setting, e.g., by Grabner, Leistner, and Bischof (2008) and Zeisl, Leistner, Saffari, and Bischof (2010).

3.1.4 Methods for online feature ranking

Feature ranking is not often addressed as a standalone task in the online setting. Usually, it is done under the name of *feature weighing* as part of a method for classification that weighs the input attributes, such as the work of Salzberg (1991), Goodman and Yih (2006), Crammer, Dredze, and Pereira (2012), Teo, Globerson, Roweis, and Smola (2008) or Dekel, Shamir, and Xiao (2010). Perkins, Lacker, and Theiler (2003) introduced the grafting method that combines multiple types of regularization to estimate the importances of attributes and uses a logistic function of the binomial negative loss function to calculate the probabilities of the class presence.

Several methods that specifically address online feature ranking have been proposed. In their works, Katakis, Tsoumakas, and Vlahavas (2005) and Katakis, Tsoumakas, and Vlahavas (2006) introduce a feature-based classifier that uses a system for incremental feature selection (IFS) and explore how IFS impacts the predictive performance of simple online classification methods, such as, e.g., naïve Bayes. Another method that specifically addresses online feature ranking is I-RELIEF (Y. Sun, 2007), which stands for iterative RELIEF, and is an adaptation of the Relief (Kira & Rendell, 1992) method for batch feature ranking to the online learning setting. Both of these methods operate in the online predictive modeling scenario. On the other hand, Yoon, Yang, and Shahabi (2005) introduced a method for online feature selection that is unsupervised, i.e., it is not directly tied to the predictive scenario. Their method utilizes the CLeVer method for principal component analysis. Recently, Duarte and Gama (2017) introduced methods for online feature ranking specifically designed for methods that use the Hoeffding inequality and used them with AMRules (Duarte et al., 2016).

Other examples of online feature ranking come from related fields, such as the work of Collins, Liu, and Leordeanu (2005) who perform feature selection for online object tracking for computer vision and the work of (W. Jiang, Er, Dai, & Gu, 2006) for the task of online image retrieval.

3.2 State of the Art in Batch Structured Output Prediction

Methods for structured output prediction can generally be divided into two types, *local* and *global* methods (Bakir, 2007). Local methods decompose the structured output into (multiple) primitive outputs and then learn separate models for predicting each of them. They then combine the predictions of the local models into a structured output prediction. On the other hand, global methods predict the entire structured output with a single global model. Global methods can model the dependencies between different primitive components of the structured output, though some local models also attempt to do the same, e.g., classifier chains (Read, Pfahringer, Holmes, & Frank, 2009) for multi-label classification and regressor chains (Spyromitros-Xioufis, Tsoumakas, Groves, & Vlahavas, 2016) for multi-target regression.

In this section, we give an overview of methods for batch structured output prediction, focusing on the tasks of multi-target regression, multi-label classification, hierarchical prediction (HMTR and HMLC), semi-supervised SOP and feature ranking in SOP.

3.2.1 Methods for batch multi-target regression

The task of multi-target regression has received considerable attention in the batch setting. We present several groups of different approaches for multi-target regression based on the model representations they use. These include problem transformation methods, tree-based methods, rule-based methods, kernel-based methods, Gaussian-process-based methods, instance-based methods, statistical and linear regression-based methods, spline-based methods and neural networks.

Problem transformation methods

Spyromitros-Xioufis et al. (2016) introduced methods for multi-target regression called stacked single-target and ensemble of regressor chains by adapting similar approaches from multi-label classification (Godbole & Sarawagi, 2004; Read et al., 2009). Regressor stacking employs local single-target models for each target, then learns an additional meta-model for each target. The meta-model learns from the predictions of other local models as input attributes in addition to the regular input attributes. These meta-models are designed to model the dependencies between the targets. On the other hand, in regressor chains, the targets are arranged in a chain, i.e., they are ordered in some way. A model for a specific target then learns from the input attributes as well as predictions of models for targets earlier in the regressor chain.

Tree-based methods

Many tree-based methods have been proposed for multi-target regression in the batch setting. De'Ath (2002) proposed one of the earliest tree-based methods for multi-target regression, where they modified the CART method (Breiman, Friedman, Olshen, & Stone, 1984) for multiple targets.

Hothorn, Hornik, and Zeileis (2006) proposed conditional inference trees, which embed tree-structured regression models into a framework for conditional inference. They use statistical tests to facilitate the growth of the tree.

Struyf and Džeroski (2006) extended the work of Blockeel and De Raedt (1998) and implemented predictive clustering trees for multi-target regression. The predictive clustering framework considers not only homogeneity of the target(s) when evaluation potential split candidates, but also the homogeneity of the input attributes. The particular work of Struyf and Džeroski (2006) allowed a user to provide constraints on the size and/or accuracy of the learned models. Their work was later extended by Appice and Džeroski (2007) to learn model trees, i.e., trees where each leaf has a corresponding linear model. Predictive clustering trees have been shown to have state-of-the-art performance when used as base models for ensemble methods, such as bagging and random forest (Kocev et al., 2013).

Tree-based methods for multi-target regression have also appeared in the popular machine learning framework scikit-learn (Pedregosa et al., 2011). In particular, scikit-learn implements a multi-output decision tree for regression, as well as ensembles of such trees.

Recently, D'Ambrosio, Aria, Iorio, and Siciliano (2017) proposed a method that can be applied for multi-target regression, however, it is designed to address more general tasks in which a target might not be single values but, for example, intervals or probabilistic estimates of multiple values.

Rule-based methods

Rule-based methods for multi-target regression are rarer than their tree-based relatives. Aho, Ženko, and Džeroski (2009) introduced the rule-based FIRE method, which learns a random forest of PCTs, then decomposes it into a large set of rules and then through an optimization procedure selects a concise set of best performing rules. Their method achieves comparable predictive performance as a multi-target regression PCT. Aho, Ženko, Džeroski, and Elomaa (2012) later extended FIRE toward using linear models in each rule.

Kernel-based methods

Like tree-based methods, kernel-based methods are particularly popular for addressing multi-target regression (Vazquez & Walter, 2003; Sánchez-Fernández, de-Prado-Cumplido, Arenas-García, & Pérez-Cruz, 2004; Cai & Cherkassky, 2009; W. Zhang, Liu, Ding, & Shi, 2012; S. Xu, An, Qiao, Zhu, & Li, 2013; Brouwer, Kubicki, Sofo, & Giles, 2014; Chung, Kim, Lee, & Kim, 2015; Yang, Chen, & Dong, 2015). In particular, support vector machines are the formalism of choice when using kernel-based methods. The most contended field of research is how to accurately model the dependencies of the targets using support vector machines. For example, Vazquez and Walter (2003) adapted the Kringing method to exploit correlations between the targets, while Yang et al. (2015) used an additional regularization term to capture the effect of correlations between the targets.

Gaussian process-based methods

Rasmussen and Williams (2006) proposed a method for Gaussian process regression. In their work, they use a kernel to define a covariance of a prior distribution over the target functions. This allows them to define likelihood functions based on the training examples. To choose the proper hyperparameters of the kernel, they use gradient ascent on the likelihood functions.

Instance-based methods

Pugelj and Džeroski (2011) extended the k nearest neighbors method (kNN, Cover and Hart (1967)) towards structured outputs in particular to multi-target regression as well as multi-label classification. They recognize the need for arbitrary distance measures in addition to the regular Euclidean distance.

Statistical and linear regression-based methods

One of the earliest methods for multi-target regression is due to Brown and Zidek (1980). They adapted the standard single-value ridge regression (Hoerl & Kennard, 1970) for multivariate ridge regression. Breiman and Friedman (1997) proposed the Curds & Whey method, where they combined canonical correlation with multivariate regression. This allowed them model target dependencies.

Mevik and Wehrens (2007) implemented the principal component regression and partial least squares regression, which can be used for multi-target regression.

Recently, Abraham, Tan, Winkler, Zhong, Liszewska, et al. (2013) present a position-regularized, multi-output prediction framework, which couples regressors with geometric quantile mapping that preserves the relationships among the targets.

The reduced rank regression framework (Anderson & Rubin, 1949; Anderson, 1951) addresses regression and multi-target regression by reducing the size (rank) of the data matrix to a small number of latent factors. This is particularly applicable to problems where the number of attributes exceeds by far the number of examples. Valente, Ginsburg, and Engelhardt (2015) extended the reduced rank regression framework in the R3-XBRRR and R3-BERRRI methods, which are non-parametric reduced rank Bayesian methods that utilize an Indian buffet process prior. This non-parametric prior allows for automatic inference of the number of relevant attributes, i.e., the rank, in the reduced rank procedure. Later, Gillberg et al. (2016) introduced a Bayesian reduced rank regression method that particularly addresses structured noise in multi-target regression.

Lasso regression (Tibshirani, 1996) is a method for regression that combines attribute selection and regularization to produce accurate and interpretable models. J. H. Friedman, Hastie, and Tibshirani (2010) among other tasks consider multinomial regression and use lasso and ridge regularization, as well as a combination of the two, called elastic-net.

Spline-based methods

Multivariate adaptive regression splines (MARS), introduced by J. H. Friedman (1991, 1993), and its recent implementation Earth (Milborrow, 2017), construct the relation between the input attributes and the target from a predetermined set of basis functions and coefficients. The input space is partitioned into regions based on the provided data, and each region is modeled using a separate regression equation learned from the corresponding data examples.

Neural networks

Neural networks (Rosenblatt, 1958) can naturally address multi-output problems like multi-target regression by using multiple nodes in the output layers. The most commonly used method for training neural networks is the backpropagation algorithm introduced by Riedmiller and Braun (1993). Ensembles of neural networks have also been proposed (Hansen & Salamon, 1990).

Recently, deep neural networks have attracted attention. Deep neural networks are neural networks that contain more than one hidden layer. Network architectures like deep belief networks (Hinton, Osindero, & Teh, 2006) or sum product networks (Poon & Domingos, 2011) can be used to address multi-target regression.

3.2.2 Methods for batch multi-label classification

In this section, we present methods for multi-label classification in the batch setting. Multi-label classification has been extensively addressed in the batch setting, with the first approach, which is due to Iman and Davenport (1980), already being several decades old. As before, we group them according to their common properties and formalisms. These include problem transformation methods, tree-based methods, rule-based methods, kernel-based methods, Bayesian methods, instance-based methods, neural methods and ensemble methods.

Problem transformation methods

In the batch setting, the problem transformation approach is commonly used to tackle the task of multi-label classification. Problem transformation methods are usually used as basic methods to compare to, and are used in combination with off-the-shelf base algorithms.

Binary relevance. The most common problem transformation approach, called *binary relevance*, transforms a multi-label task into several binary classification tasks, one for each of the possible labels (Read et al., 2011). As such, we consider it a local method. Binary relevance models are often criticized due to their inability to account for label correlations (Zhou, Tao, & Wu, 2012). However, addition of new labels is trivial, whereas in other methods, this might require considerable effort.

Classifier chains. Classifier chains (Godbole & Sarawagi, 2004; Read et al., 2011; Sucar et al., 2014; B. Chen, Li, Zhang, & Hu, 2016) are similar to the binary relevance approach, however, instead of each label being modeled independently, the classifiers are linked into a chain. Each classifier in the chain receives the predictions of the preceding classifiers as input attributes. Alali and Kubat (2015) introduced the PruDent method, which uses two layers of classifiers. The first layer consists of binary relevance classifiers, while the second layer gets as input the predictions of the models in the first layer. To address the scalability issues of classifier chains, Read, Martino, Olmos, and Luengo (2015) proposed classifier trellises, which do not model complete chains, and can be efficiently used for multi-label classification tasks with thousands of labels.

Label powerset. Another common problem transformation approach is the *label combination* or *label powerset* method, where each subset of the labelset is considered as an atomic label for a multi-class classification problem (Boutell, Luo, Shen, & Brown, 2004; Read et al., 2008; Tsoumakas & Vlahavas, 2007). If we start with a multi-label classification task with a labelset of $\mathcal{L}$, we transform it into a multi-class classification task with a labelset $\mathcal{L}' = 2^{\mathcal{L}}$. Notably, label powerset is not a local method as it does not predict individual components of the structured output, rather, it encodes all components into a multi-class classification task. This approach suffers from intractability, as the number of labels in the labelset is exponential in the size of the original labelset, and is thus only suitable for tasks with a low number of labels (Cheng & Hüllermeier, 2009).

To address this shortcoming, several approaches have been proposed. Read (2008) proposed pruning the powerset, while Tsoumakas, Katakis, and Vlahavas (2008) proposed an approach where first the labels are hierarchically arranged, then classifiers are learned that predict the labelsets in each node of the hierarchy.

Pairwise approaches. In *pairwise classification* a binary model is learned for each possible pair of labels (Fürnkranz, Hüllermeier, Mencía, & Brinker, 2008). Each label pair is trained using modified instances, where examples are labeled as positive if they are labeled with the first label of the pair, and negative if they are labeled with the second label of the pair. A prediction is then obtained by majority vote over all pairs. For larger problems, i.e., when there are a lot of labels, the method becomes intractable because of model complexity. Recently, Madjarov and Gjorgjevikj (2012) introduced a two-stage approach, where each example is first classified using binary relevance models, and afterwards using pairwise models.

Tree-based methods

As in multi-target regression, tree-based methods are very popular also for multi-label classification. For example, Clare and King (2001) used a modified definition of entropy for multi-label data and extended the C4.5 algorithm (Quinlan, 1993) to learn multi-label trees for the prediction of gene expression data.

The predictive clustering framework (Blockeel & De Raedt, 1998) is also directly applicable for multi-label classification, if appropriate variance and prototype functions are selected.

Q. Wu, Ye, Zhang, Chow, and Ho (2015) introduced a method, where SVMs are trained in each node in a one-versus-all setting and are then used to evaluate potential splits. On the other hand, 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. Modifying the tree growing procedure to explicitly capture local label dependencies, i.e., label dependencies on the subspace that the node represents, has also been proposed (Al-Otaibi, Kull, & Flach, 2014; Al-Otaibi, Kull, & Flach, 2016).

Recently, Breskvar, Kocev, and Džeroski (2017) introduced a method based on predictive clustering trees that randomly selects label subsets and learns models that predict only the labels in the selected label subset.

Rule-based methods

The method of De Comité, Gilleron, and Tommasi (2003) learns rules for multi-label classification, though they are organized in such a way that we can look at their method also as a tree-based method. Ženko and Džeroski (2008) introduced predictive clustering rules, which can be applied to multi-label classification. Later, Mencía and Janssen (2016) introduced two rule-based methods, one based on stacking and the other based on a separate-and-conquer approach, which learn single-label rules. While these rules predict a single label, their antecedents can contain conditions on the predictions of other labels. This enables the modeling of label dependencies.

Kernel-based methods

Support vector machines have often been used in combination with the binary relevance approach to address multi-label classification (Boutell et al., 2004; T. Gonçalves & Quaresma, 2004; E. C. Gonçalves, Plastino, & Freitas, 2013; Elisseeff & Weston, 2002; J. Xu, 2014; W.-J. Chen, Shao, Li, & Deng, 2016; Jayadeva, Khemchandani, & Chandra, 2007). The introduced methods generally extend the support vector machine framework, for example, Jayadeva et al. (2007) implemented a support vector machine method for multi-label classification that uses non-parallel separating hyperplanes.

Bayesian methods

Shangfei Wang, Wang, Wang, and Ji (2014) introduced a method that models inter-label dependencies using a Bayesian network. Individual nodes represent labels, while the edges with conditional probabilities encode label dependencies. The learned Bayesian networks can also be applied to other methods for multi-label classification to improve their performance.

Instance-based methods

M.-L. Zhang and Zhou (2005) introduced ML-kNN, which is an adaptation of the k nearest neighbors method (kNN, Cover and Hart (1967)) for multi-label classification. C. Liu and Cao (2015) later extended the ML-kNN by using a coupled similarity measure, which does not assume an independence of labels and examples. Spyromitros, Tsoumakas, and Vlahavas (2008) used kNN in combination with the binary relevance problem transformation method, while the approach of Pugelj and Džeroski (2011), which we already mentioned in the context of multi-target regression, is also applicable to multi-label classification. Cheng and Hüllermeier (2009) combined the kNN method with an additional step in which they use the labels of the nearby instances as inputs to a logistic model.

Neural networks

M.-L. Zhang and Zhou (2006) adapted the back propagation algorithm (Rumelhart, Hinton, & Williams, 1986) for multi-label classification, by considering an error function ranks the present labels of an example higher than absent labels. Furthermore, M.-L. Zhang (2009) adapted the radial basis function neural network for multi-label classification. Finally, Yeh, Wu, Ko, and Wang (2017) proposed deep neural networks, i.e., neural networks with more than one hidden layer, for multi-label classification.

Ensemble approaches

Schapire and Singer (2000) extended the boosting ensemble method for multi-label classification in two methods, AdaBoost.MH and AdaBoost.MR. Alternating decision trees (Freund & Mason, 1999) were adapted by De Comité et al. (2003) for multi-label classification. As discussed before, Kocev et al. (2013) used bagging and random forests of PCTs, which can be applied to multi-label classification as well as multi-target regression. Su and Rousu (2015) proposed a network-ensemble combination method, in which a graph is defined over the labels to model their interdependencies. The ensemble members then jointly label the graph for each example. Recently, Q. Wu, Tan, Song, Chen, and Ng (2016) introduced a modified bagging method that automatically tracks, which labels are relevant for the examples as they are used in the tree learning procedure.

Comparison of multi-label classification methods

Several empirical comparisons of methods for multi-label classification have been conducted (Q. Wu et al., 2016; Madjarov, Kocev, et al., 2012). As we will discuss later in Chapter 5, a method cannot optimize all of the evaluation measures in a multi-label classification context, which is confirmed by these studies.

In terms of predictive performance, the best methods are classifier chains (Read et al., 2011), HOMER (Tsoumakas et al., 2008), the two-step architecture introduced by Madjarov, Gjorgjevikj, and Džeroski (2012), ensembles of PCTs (Kocev et al., 2013), and ML-FOREST (Q. Wu et al., 2016). When additionally considering also the efficiency of the model, ensembles of PCTs and HOMER outperform all other methods.

3.2.3 Methods for batch hierarchical prediction

Hierarchical prediction problems are found in many application domains, most notably in text classification (Rousu, Saunders, Szedmak, & Shawe-Taylor, 2006), functional genomics (Barutcuoglu, Schapire, & Troyanskaya, 2006) and object recognition (Stenger, Thayananthan, Torr, & Cipolla, 2007). Historically, the only hierarchical task that was considered was hierarchical multi-label classification, however, recently, Mileski et al. (2017) introduced the task of hierarchical multi-target regression.

Batch hierarchical multi-target regression

The hierarchical multi-target regression task has only been introduced recently by Mileski et al. (2017). They address hierarchical multi-target regression by using predictive clustering trees adapted from methods for hierarchical multi-label classification (Vens et al., 2008).

Batch hierarchical multi-label classification

Silla and Freitas (2011) present an extensive survey of methods for hierarchical multi-label classification. Levatić et al. (2017b) investigated the importance of the hierarchy in hierarchical multi-label classification, and its single-target variant hierarchical single classification. Recently, Madjarov, Gjorgjevikj, Dimitrovski, and Džeroski (2016) explored automatically generated data-driven hierarchies that help improve predictive performance for multi-label classifiers.

Tree-based methods. Cerri, Pappa, Carvalho, and Freitas (2015) present a survey of tree-based methods for hierarchical multi-label classification, and suggest precision-recall curves (Davis & Goadrich, 2006) and hierarchical precision and recall (Kiritchenko, Matwin, Nock, & Famili, 2006) as the evaluation metrics of choice for this setting.

Clare and King (2003) extended their earlier work for multi-label classification (Clare & King, 2001) for hierarchically arranged labels, penalizing errors higher up in the hierarchy more severely.

Blockeel, Bruynooghe, Džeroski, Ramon, and Struyf (2002) and Blockeel, Schietgat, Struyf, Džeroski, and Clare (2006) use predictive clustering trees (Blockeel & De Raedt, 1998) to address hierarchical multi-label classification, while Vens et al. (2008) extend their work to also be applicable for DAG hierarchies. Kocev et al. (2013) showed that ensembles of PCTs for hierarchical multi-label classification significantly outperform single PCTs.

Kernel-based methods. Another group of methods that is popular for hierarchical multi-label classification are kernel-based methods, in particular, support vector machines (Obozinski, Lanckriet, Grant, Jordan, & Noble, 2008; Barutcuoglu et al., 2006; Guan et al., 2008; Valentini, 2011). These methods all require a second step to ensure that the hierarchy constraint is satisfied, while Rousu et al. (2006) ensure this directly.

Network-based methods. Methods in the application area of protein function prediction often utilize protein-protein interaction networks and functional association networks (Y. Chen & Xu, 2004; Mostafavi, Ray, Warde-Farley, Grouios, & Morris, 2008; Tian et al., 2008). Stojanova, Ceci, Malerba, and Džeroski (2013) implemented PCTs for hierarchical multi-label classification in a network context and explored auto-correlation in this setting.

Other methods. Several methods that cannot be as easily arranged into method groups have also been proposed. Kiritchenko et al. (2006) expanded labelsets to make them consistent with a provided class hierarchy and then applied multi-class classification methods. Silla Jr and Freitas (2009) adapted the naïve Bayes approach for hierarchical multi-label classification. Otero, Freitas, and Johnson (2010) and Cerri, Barros, and de Carvalho (2012) used search heuristic to learn rules for hierarchical multi-label classification. Later, Cerri, Barros, and De Carvalho (2014) represented the label hierarchy as an interconnected network of neural networks which feed into each other according to the hierarchy. Alaydie, Reddy, and Fotouhi (2012) used a boosting-like method, by utilizing the hierarchy to select the training sets for each classifier in the boosting ensemble. Barros, Cerri, Freitas, and de Carvalho (2013) proposed a method based on probabilistic clustering with the expectation-maximization algorithm. Bi and Kwok (2015) developed a method for learning a Bayes-optimal classifier, based on a hierarchically aware loss function. Z. Sun, Zhao, Cao, and Hao (2017) transform the hierarchical multi-label prediction task into a task of optimal path prediction and used network solvers to train models.

3.2.4 Methods for batch semi-supervised structured output prediction

Most work in semi-supervised learning deals with the prediction of primitive targets, i.e., with classification and regression (Chapelle, Schlkopf, & Zien, 2010). However, it has also received attention in the scope of other SOP tasks, such as sequence labeling (Suzuki, Fujino, & Isozaki, 2007; Brefeld et al., 2005; Dhillon, Sellamanickam, & Selvaraj, 2011; Chang, Ratinov, & Roth, 2012; Dhillon, Keerthi, Bellare, Chapelle, & Sellamanickam, 2012), parse tree prediction (Brefeld & Scheffer, 2006) and time-series classification (Kim, 2013).

We present a brief overview of semi-supervised methods for structured output prediction, dividing the work into two categories: methods that address a specific SOP task and general methods that are applicable to multiple SOP tasks.

We also make a distinction between inductive and transductive methods for semi-supervised learning (Chapelle et al., 2010). Inductive methods learn a predictive model as we have described in Section 2.2.1.5. On the other hand, transductive methods are concerned only with the prediction of labels of specific unlabeled examples, and thus do not require the explicit learning of a model.

Semi-supervised methods for specific data mining tasks

Ceci (2008) introduced a transductive graph-based method for hierarchical text categorization. Santos and Canuto (2014) introduced a method based on self-training for hierarchical multi-label classification. Self-training methods use their own predictions of the unlabeled examples as training values in specific circumstances. Y. Zhang and Yeung (2009) and Cardona, Álvarez, and Orozco (2015) introduced methods based on Gaussian processes for semi-supervised multi-task learning, a task related to multi-target regression. Navaratnam, Fitzgibbon, and Cipolla (2007) also introduced a Gaussian process-based method for semi-supervised multi-target regression, though their approach is aimed specifically at applications in computer vision. Levatić et al. (2017a) introduced a semi-supervised method for multi-target regression based on self-training of random forests of predictive clustering trees. L. Wu and Zhang (2013) introduced an inductive approach for multi-label classification based on relearning of a support vector machine both on labeled and unlabeled examples. Other inductive methods for multi-label classification include semi-supervised boosting (Zhao & Zhai, 2015), semi-supervised kNN (de Lucena & Prudencio, 2015), co-training (M. Xu, Sun, & Jiang, 2014) and semi-supervised binary relevance (Švec, 2014).

A bevy of transductive methods for multi-label classification have been introduced (G. Chen, Song, Wang, & Zhang, 2008; Zha, Mei, Wang, Wang, & Hua, 2009; J. Wang, Zhao, Wu, & Hua, 2011; Guo & Schuurmans, 2012; Kong, Ng, & Zhou, 2013; Y. Wang, Pei, Lin, Zhang, & Zhang, 2014; B. Wang & Tsotsos, 2016). As these methods use a different labeling setup, we do not specifically describe them, only mention that these approaches are often computationally expensive (W. Liu, Wang, & Chang, 2012).

Generally applicable semi-supervised methods

Altun et al. (2006) and Yujia Li and Zemel (2014) introduced SVM-like methods based on the maximum-margin approach that can address multiple SOP tasks. Brefeld and Scheffer (2006) used SVMs in a co-training approach and applied the principle of maximal consensus between independent hypotheses. Zien, Brefeld, and Scheffer (2007) introduced transductive SVMs for structured output prediction tasks.

Conditional random fields (Lafferty, McCallum, & Pereira, 2001) are a method commonly used for nominal structured output prediction tasks, e.g., sequence labeling. This approach has been extended toward the semi-supervised setting (Y. Wang, Haffari, Wang, & Mori, 2009; Subramanya, Petrov, & Pereira, 2010; Dhillon et al., 2011).

Recently, F. Jiang, Jia, Sheng, and LeMieux (2016) and Du (2017) utilized the kNN method for several types of nominal structured output prediction tasks, such as multi-label classification, hierarchical multi-label classification and sequence learning.

Sellamanickam, Tiwari, and Selvaraj (2012) used probabilistic entropy-based models with label distribution regularization to address hierarchical and non-hierarchical multi-label classification. Furthermore, Suzuki et al. (2007) introduced a hybrid discriminative-generative method, that uses generated labels for unlabeled data examples.

Gönen and Kaski (2014) and Brouard, Szafranski, and d'Alché-Buc (2016) proposed methods that can be applied to both nominal and continuous structured output prediction tasks, with kernelized Bayesian matrix factorization and input output kernel regression, respectively.

Finally, Levatić et al. (2017b) introduced an inductive method for semi-supervised structured output prediction based on predictive clustering trees. The unlabeled examples are considered when the tree is grown by impacting the splitting heuristic which takes into account the homogeneity of the input attributes in addition to the homogeneity of the output attributes.

3.2.5 Methods for batch feature ranking for structured output prediction

As far as we can tell, only the work of Petković et al. (2017) has addressed feature ranking for any SOP task in the batch setting. They introduce and adapt several techniques based on the tree-based ensembles.

3.3 State of the Art in Online Structured Output Prediction

In this section, we present the related work on the topic of online methods for structured output prediction. We present a review of existing online methods for multi-target regression and multi-label classification. As far as we were able to discern, there is currently no related work in the context of online hierarchical learning, online semi-supervised structured output learning or online feature ranking for structured output prediction.

Most modern methods for structured output prediction are implemented in the Java-based Massive Online Analysis (MOA) framework[2], which includes online methods for classification, regression, multi-label classification, multi-target regression, clustering and concept drift detection (Bifet, Holmes, Kirkby, & Pfahringer, 2010).

3.3.1 Existing methods for online multi-target regression

In the online setting, some attention has been given to multi-target regression. Namely, Ikonomovska, Gama, and Džeroski (2011b) extended the FIMT-DD method to the multi-target regression setting in the FIMT-MT method (Ikonomovska, Gama, & Džeroski, 2011a). Recently, Duarte and Gama (2015) implemented a rule-based learning method for multi-target regression based on the Hoeffding bound.

3.3.2 Existing methods for online multi-label classification

With the exception of the work of Crammer and Singer (2003a), which introduced a local incremental multi-label classifier that learns one perceptron for each label, online multi-

[2] URL: http://moa.cms.waikato.ac.nz/, accessed on (2018/01/22)

label classification has only recently started attracting attention.

Most of the methods for online multi-label classification utilize some manner of problem transformation, for example, Qu, Zhang, Zhu, and Qiu (2009) introduced a batch-incremental method that trains stacked binary relevance classifiers. Read et al. (2012) proposed the use of multi-label Hoeffding trees with pruned sets in the leaves and used them in combination with the ADWIN bagging (Bifet, Holmes, Pfahringer, Kirkby, & Gavaldà, 2009) ensemble method.

Recently, Spyromitros-Xioufis (2011) introduced a parameterized windowing technique for dealing with concept drift in multi-label data in an online context. Next, Shi, Wen, et al. (2014) proposed an efficient and effective method to detect concept drift based on label grouping and entropy for multi-label data, where the labels are grouped by using clustering and association rules. This allowed for an effective detection of concept drift which takes into account label dependence. Later, Shi, Xue, et al. (2014) proposed an efficient class incremental learning algorithm, which dynamically recognizes some new frequent label combinations. Finally, Sousa and Gama (2018) recently introduced a method for online multi-label classification based on AMRules (Duarte et al., 2016).

3.4 Critical Summary of Related Work Relevant to the Thesis

From the preceding description of related work, we see that SOP tasks have received considerable attention in the classical batch learning setting. In the online setting, however, only multi-target regression and multi-label classification have been addressed so far. Currently there are no methods for online hierarchical prediction, for both hierarchical multi-label classification and hierarchical multi-target regression tasks. Online semi-supervised learning has only been addressed in the non-structured tasks of classification and regression, while there are no methods that deal with online SSL for structured output prediction. The body of work for online feature ranking is similar to that of online SSL, in that there are methods for online feature ranking for classification and regression, but none for online feature ranking for structured output prediction.

Many of the approaches mentioned earlier in this chapter serve as foundations of this thesis. In particular, we introduce iSOUP-Tree which extends FIMT-DD (Ikonomovska, Gama, & Džeroski, 2011b) for online regression and its initial multi-target regression adaptation FIMT-MT (Ikonomovska, Gama, & Džeroski, 2011a). Additionally, we extend iSOUP-Tree toward iSOUP-OptionTree, similarly to how ORTO (Ikonomovska et al., 2015) extends FIMT-DD. We also use online ensemble methods, in particular online bagging (Oza & Russel, 2001) and online random forests (Oza, 2005).

Furthermore, we adapt the approaches of Vens et al. (2008) and Mileski et al. (2017) for hierarchical multi-label classification and hierarchical multi-target regression, respectively, to the online setting. We utilize the predictive clustering framework (Blockeel, 1998) in the online setting and apply online predictive clustering trees for semi-supervised multi-target regression, as proposed by Levatić et al. (2017b) in the batch learning setting. Finally, we adapt the approach of Petković et al. (2017) that uses symbolic random forests to calculate feature importances for SOP tasks. In place of a random forest of predictive clustering trees, we instead use an ensemble of online randomized iSOUP-Trees.

Chapter 4

Methods for Structured Output Prediction on Data Streams

> With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.
>
> — John von Neumann

As we have seen in Chapter 3, there are many methods for structured output prediction that learn distinctly different models, i.e., they use different model formalisms, in the batch learning setting. In particular, tree-based methods are popular across a wide variety of predictive modeling tasks due to their favorable properties: they can be learned quickly, they produce interpretable models, and they achieve state-of-the-art predictive performance, in particular when combined with ensemble methods such as bagging or random forests.

In the online scenario, however, there are far fewer methods for structured output prediction tasks and the few that do exists are generally restricted to one particular structured predictive modeling task. In the context of this thesis, we introduce a family of machine learning methods that can, in combination with problem transformation approaches, address several types of structured output prediction tasks in the online learning setting.

We start this chapter with an introduction into tree-based predictive models, where we describe how predictions are made with trees and how trees are learned. Afterward, we introduce the iSOUP-Tree family of methods for online multi-target regression: iSOUP-Tree in both its regression and model tree variants, iSOUP-RegressionTree and iSOUP-ModelTree, iSOUP-OptionTree and online bagging and random forests of iSOUP-Trees. We continue by introducing the online multi-label classification by online multi-target regression problem transformation methodology, which enables the use of methods for online multi-target regression to address online multi-label classification. Later, we present three extensions of the iSOUP-Tree family of methods that were inspired by batch tree-based methods. In particular, we present methods that can be used to address online hierarchical prediction, online semi-supervised learning and online feature ranking. Notably, these three extensions are independent of each other, meaning that, for example, we can use a semi-supervised tree in conjunction with the hierarchical extension to address online semi-supervised hierarchical multi-label classification.

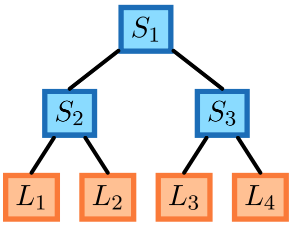


Figure 4.1: A sample decision tree. Blue S_i nodes are split nodes, orange L_j nodes are leaf nodes, S_1 is the root node.

4.1 Introduction to Tree-Based Predictive Models

The methods we introduce in this thesis are all tree-based. That means that they produce models that are either decision trees or ensembles of decision trees. A *decision tree* is a predictive model that is described with a tree structure comprised of different types of nodes, as shown in Figure 4.1. Each tree has a special node called the *root node* (node S_1 in Figure 4.1), which denotes where the tree "starts". Decision trees that can be used in classification are called *classification trees*, and similarly, decision trees for regression are called *regression trees* (Breiman et al., 1984).

In a decision tree, we differentiate between two types of nodes, *split nodes* and *leaf* nodes. Split nodes act as guides for examples, i.e., they direct them toward the appropriate leaves. Each split node contains an attribute test, which can be evaluated by applying the value of the appropriate attribute to it. Note, that most commonly, the initial tree model starts as a single leaf node and is generally split during the learning process, resulting in the root node being a split node. Leaf nodes, on the other hand, provide a predictive function. This function can be obtained directly from the examples that reached the leaf, such as the majority class in classification or average value in regression, or it can be the result of using a simple model learned from these examples.

Split nodes most commonly contain tests of the following form:

- *for nominal attributes:* "Is the value of attribute A_i of the example equal to value v?", which we represent as $x_i = v$,
- *for numeric attributes:* "Is the value of attribute A_i of the example smaller or equal to value v?", compactly represented as $x_i \leq v$,

where x_i represents the value of the attribute A_i of a passing example and v is one of the possible values of attribute A_i.

To calculate a prediction for a data example e using a decision tree, we must first traverse the example to the appropriate leaf node. We start by applying the test to the example at the root node, i.e., to its attribute values, and if the test evaluates as true, we move to the left child, while if the test evaluates as false, we move to the right child. We continue this process recursively, until we encounter a leaf node. This process is called *tree traversal* or *traversal of example e*.

Once we have traversed an example to a leaf, we make a prediction based on the examples stored in the leaf. For example, in regression, a prediction might be the average value of the target(s) of the examples stored in the leaf, while in classification, this might be the majority class, i.e., the most common class, of the stored examples. Alternatively, each leaf might have a (simple) predictive model, such as a naïve Bayes classifier (Hand & Yu, 2001) in classification trees or a perceptron (Rosenblatt, 1958) in regression trees.

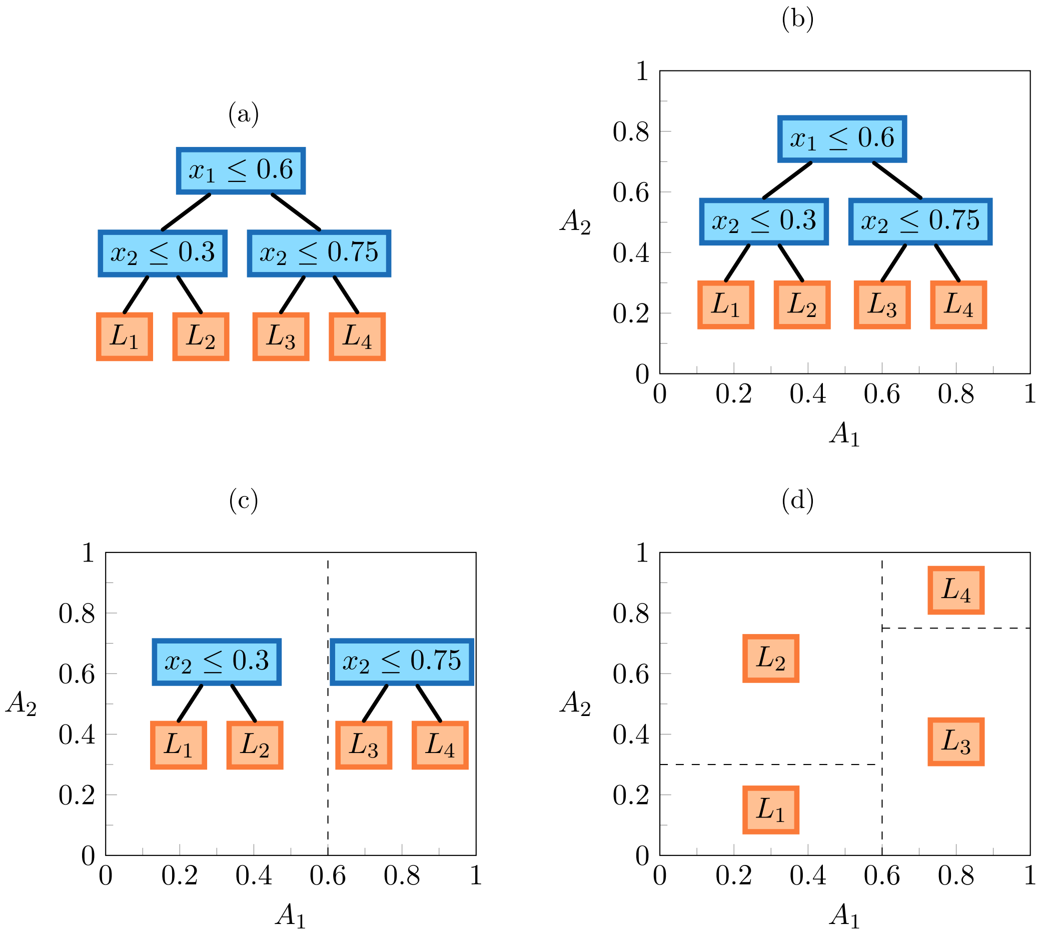


Figure 4.2: An example of a (a) decision tree and its partitioning of the input space at various levels in the tree: (b) at the root node (depth 0), (c) at depth 1 and (d) at depth 2.

These simple models are learned from the examples that the leaf has stored. Trees that utilize additional, nontrivial, models in the leaves are called *model trees*.

A decision tree naturally defines a partition on the input attribute space. To illustrate this, let us observe how a split node which is at the root of the tree separates the input space. For example, let us consider an input space which is comprised of two real valued attributes A_1 and A_2 that take values only on the $[0, 1]$ interval and the decision tree pictured in Figure 4.2a. The entire tree applies to the entirety of the input space as shown in Figure 4.2b. The root of the tree is a split node that has the test $x_1 \leq 0.6$. Each of the split node's two subtrees, i.e., its left and right child, are then applied to only a part of the input space, as shown in Figure 4.2c. If we apply the same logic to the subtrees and subspaces defined above, we get the final partitioning of the input space, where a leaf node is applied to each subspace, as shown in Figure 4.2d.

Another concept important for decision trees, and also other tree structures, e.g., label hierarchies, is the concept of *depth* of a node. We define the depth of a node as the number of nodes that are placed before it, that is, the number of nodes on the path from the node to the root of the tree. The root node has a depth of 0, its children a depth of 1, their children a depth of 2 and so on.

By far the most popular way of learning is to use the above line of thought and re-

Algorithm 4.1: Top-down induction of decision trees (TDIDT).

```
Procedure: TDIDT
Input: Dataset $\mathcal{D}$ with $N$ descriptive attributes, measure of purity $\mathcal{M}$, stoping
       criteria Stop
Output: A decision tree T
if stopping criteria Stop holds for $\mathcal{D}$ then
    let T become a new leaf created from $\mathcal{D}$;
else
    find the attribute $A_i$, which most increases the purity $\mathcal{M}$;
    divide $\mathcal{D}$ into $m$ subsets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$, according to the values of attribute
     $A_i$;
    for $j \leftarrow 1$ to $m$ do
        let $\mathsf{T}_j$ be the decision tree we get by applying TDIDT($\mathcal{D}_j$, $\mathcal{M}$, Stop);
    end
    create a new split node according to $A_i$ with all $\mathsf{T}_j$ as its children;
end
return T
```

cursively partition the input space. This is most commonly referred to as the *top-down induction of decision trees* (TDIDT) introduced by Breiman et al. (1984). The TDIDT algorithm is shown in Algorithm 4.1. Top-down induction refers to the fact that we start at the top of the tree, i.e., the root node, which covers the entire input space as we have seen above. Most tree-based methods are based on the TDIDT algorithm, from the earliest algorithms, such as CART (Breiman et al., 1984), ID3 (Quinlan, 1986), ASSISTANT (Cestnik, Kononenko, & Bratko, 1987), M5 (Quinlan, 1992), RETIS (Karalič, 1992), C4.5 (Quinlan, 1993), to current stat-of-the art methods, such as predictive clustering trees (Blockeel & De Raedt, 1998).

The TDIDT algorithm always selects the split with the best purity (for the current subspace of the input space). This selection might not be optimal and might cause the TDIDT algorithm to learn a local optimum instead of the global one. This is why the TDIDT algorithm (and its direct modifications) are called *greedy* or *myopic*. The algorithm only ever selects the best split (greed), without consideration for what might happen at later splits lower in the tree (shortsightedness, myopia).

The TDIDT algorithm was originally introduced for the classification task, where purity was measured in terms of, e.g., information gain and Gini-index (Kononenko & Kukar, 2007). To adapt the TDIDT algorithm for regression, purity was measured in terms of, e.g., variance reduction. The *variance reduction* of a split that splits a sample of examples S into subsamples $S_\top$ and $S_\bot$ is defined as

$$\mathrm{VR} = \frac{1}{\mathrm{Var}(S)} \left(\mathrm{Var}(S) - \frac{|S_\top|}{|S|} \mathrm{Var}(S_\top) - \frac{|S_\bot|}{|S|} \mathrm{Var}(S_\bot) \right), \tag{4.1}$$

where Var(A) is the variance of the target values of examples in set A.

The astute reader will, however, notice that the TDIDT algorithm requires access to the entire dataset to calculate the splitting heuristic. This means that the TDIDT procedure is not directly applicable to an online learning scenario. Schlimmer and Fisher (1986) were one of the first to propose an incrementally updated tree induction procedure for the binary classification task based on the ID3 (Quinlan, 1986) method. They incrementally updated statistics, which are then used to calculate the heuristic (in their case the information score) and potentially split a leaf node or replace a split node that has become obsolete.

In their method, the splitting statistic is checked after every example. However, with the advent of problems with more and more input attributes, this became unfeasible.

Hence, the evaluation of possible splits was deferred to only occur once a certain predefined number of examples have been observed in the particular leaf. This kind of approach was used, e.g., by Domingos and Hulten (2000) in Hoeffding trees and by Ikonomovska, Gama, and Džeroski (2011b) in FIMT-DD.

4.2 Methods for Online Multi-Target Regression

The global approach, where methods learn models that predict all of the targets at once, has been shown to yield good predictive performance in the case of tree-based methods for structured output prediction in the batch setting. This has motivated us to introduce new global tree-based methods for data streams. The methods defined below are all instance incremental methods for online multi-target regression. As we have described in Chapter 2, this means that we provide a starting hypothesis h_0 and the appropriate update operator u.

In the following sections, we first define the iSOUP-Tree method, following with the iSOUP-OptionTree method, before focusing on the online ensemble methods of online bagging and online random forest. Notably, all of these methods utilize the global approach to multi-target regression. Finally, we also consider a local approach, which harnesses multiple applications of the single-target FIMT-DD method to address the multi-target regression task.

A summary of the introduced methods and their parameters is shown in Table 4.1. All of the methods described below were implemented in the Massive Online Analysis (MOA) framework for online learning (Bifet, Holmes, Kirkby, & Pfahringer, 2010).

4.2.1 The iSOUP-Tree method

The incremental Structured Output Prediction Tree method (iSOUP-Tree) is a tree-based instance-incremental method for online multi-target regression[1]. It is based on FIMT-DD (Ikonomovska, Gama, & Džeroski, 2011b) method for online single-target regression and FIMT-MT (Ikonomovska, Gama, & Džeroski, 2011a) method for online multi-target regression. It expands on these methods in several ways, which we discuss in the sections below.

In order to define an instance-incremental method, we must first define the starting hypothesis, that is the initial predictive model. In the case of iSOUP-Tree, the initial model is an empty leaf node, i.e., a leaf node that has observed no examples so far. The update operator, which describes how the model learns and fully describes the method, is described in Algorithm 4.2.

From Algorithm 4.2 we can see that the updating procedure has many different components that need to be explained in detail. In the following sections, we first explain how the statistics of each node are kept and updated. We continue with a look at how splitting occurs, specifically, in regard to the calculation of the splitting heuristic and the application of the Hoeffding inequality. Afterwards, we examine how predictions are made using iSOUP-Trees, delving deeper into the leaf models.

[1] iSOUP-Trees, similar to the approach of Struyf and Džeroski (2006) in the batch learning scenario, also utilize the predictive clustering framework (Blockeel & De Raedt, 1998) which we later use to address online semi-supervised learning. However, in this thesis, we use the term *predictive clustering* to explicitly describe methods that consider the homogeneity of the descriptive attributes in addition to the homogeneity of the targets in the learning process.

Table 4.1: Summary of introduced methods. **P** is the parameter designation, **DV** is its default value.

<table>
<tr><th colspan="4">iSOUP-Tree</th></tr>
<tr><td colspan="4">It uses the Hoeffding inequality to assess whether there is support for splitting a leaf. Each leaf is checked for splitting only at set intervals of observed examples. It learns model or regression trees.</td></tr>
<tr><th>P</th><th>DV</th><th>Range</th><th>Description</th></tr>
<tr><td>GP</td><td>200</td><td>$\mathbb{N}$</td><td>Grace period</td></tr>
<tr><td>δ</td><td>10^{-7}</td><td>$[0, 1]$</td><td>Split confidence</td></tr>
<tr><td>τ</td><td>0.05</td><td>$[0, 1]$</td><td>Tie breaking threshold</td></tr>
<tr><td>LP</td><td>M</td><td>M, R</td><td>Leaf predictor; M – adaptive model, R – mean regressor</td></tr>
<tr><td>η_0</td><td>0.2</td><td>$[0, 1]$</td><td>Initial learning rate; only used if LP = M</td></tr>
<tr><td>η_Δ</td><td>0.001</td><td>$[0, 1]$</td><td>Learning rate decay; only used if LP = M</td></tr>
<tr><th colspan="4">iSOUP-OptionTree (iSOUP-OT)</th></tr>
<tr><td colspan="4">Extension of iSOUP-Tree utilizing option nodes to address myopia of the greedy tree learning procedure. It inherits all parameters of iSOUP-Tree.</td></tr>
<tr><th>P</th><th>DV</th><th>Range</th><th>Description</th></tr>
<tr><td>β</td><td>0.95</td><td>$[0, 1]$</td><td>Option decay rate</td></tr>
<tr><td>$d_{\max}$</td><td>2</td><td>$\mathbb{N}$</td><td>Maximum option node depth</td></tr>
<tr><th colspan="4">Online Bagging (iSOUP-Bag)</th></tr>
<tr><td colspan="4">Ensemble method for online learning utilizing bootstrap aggregates to achieve base model diversity. It uses iSOUP-Trees as base models.</td></tr>
<tr><th>P</th><th>DV</th><th>Range</th><th>Description</th></tr>
<tr><td>$\mathcal{N}$</td><td>100</td><td>$\mathbb{N}$</td><td>Size of ensemble</td></tr>
<tr><th colspan="4">Online Random Forest (iSOUP-RF)</th></tr>
<tr><td colspan="4">Ensemble method that utilizes individually randomized iSOUP-Trees to achieve high diversity among base models. It extends online bagging.</td></tr>
<tr><th>P</th><th>DV</th><th>Range</th><th>Description</th></tr>
<tr><td>NA</td><td>$\sqrt{N}$</td><td>$\sqrt{N}$, $\log N$</td><td>Number of observed attributes in base models</td></tr>
<tr><th colspan="4">Local FIMT-DD</th></tr>
<tr><td colspan="4">Local method that learns a collection of M single-target FIMT-DD models. Each model predicts one of the targets. Same parameters as iSOUP-Tree.</td></tr>
</table>

Algorithm 4.2: The iSOUP-Tree update operator.

```
Input: current iSOUP-Tree T, new example e
Output: updated iSOUP-Tree T
currentNode ← T.rootNode;
while currentNode is a split node do          // traverse example e to a leaf
    direction ← currentNode.directionForExample(e);
    currentNode ← currentNode.children(direction);
end
currentNode.updateStatistics(e);
currentNode.updateModel(e);
if currentNode.examplesSeen mod GP = 0 then
    foreach attribute $A_i$ do
        $\mathcal{S}_i, h_i$ ← BestSplitForAttribute($A_i$);
    end
    $\mathcal{S}_{\mathbb{1}}, h_{\mathbb{1}}$ ← BestSplit($\{(\mathcal{S}_1, h_1), \ldots, (\mathcal{S}_N, h_N)\}$);
    $\mathcal{S}_{\mathbb{2}}, h_{\mathbb{2}}$ ← BestSplit($\{(\mathcal{S}_1, h_1), \ldots, (\mathcal{S}_N, h_N)\} \setminus \{(\mathcal{S}_{\mathbb{1}}, h_{\mathbb{1}})\}$);
    $\varepsilon$ ← HoeffdingBound($\delta$, leaf.observedExamples);
    if $\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}} < 1 - \varepsilon$ or $\varepsilon \leq \tau$ then          // split currentNode
        splitNode ← SplitNode($\mathcal{S}_{\mathbb{1}}$);
        splitNode.left ← LeafNode(currentNode);
        splitNode.right ← LeafNode(currentNode);
        replace currentNode with splitNode;
    end
end
return T
```

4.2.1.1 Splitting heuristic and split selection

Once enough examples have been observed (but not stored directly) in a leaf node, we check whether we have sufficient statistical support to split the leaf. From the records of the values of attributes, we calculate all possible (binary) splits and evaluate them according to a heuristic function. As above, the splits on numeric input attributes take the form $A \leq c$, for some numeric value c of attribute A. The splits on nominal attributes take the form $A = n$ for some discrete value n of attribute A.

To split a leaf node, we wish to find the best possible split, given the data examples that have reached the leaf. For each input attribute, we enumerate all possible candidate splits. We denote a split candidate that splits the examples based on the attribute values of attribute A_i as $\mathcal{S} : x_i \leq v$, where v is a value of attribute A_i, i.e., $v \in X_i$.

In the case of a numeric attribute, we do not consider all the possible split points, as there are infinitely many of them. Instead, we consider only the values of the attribute that have been recorded thus far, as these values are exactly the values where the value of the heuristic function can change. For nominal attributes we consider each split into one class versus all other classes.

In a single-target scenario, several heuristics for the evaluation of split candidates have been proposed. For example, Potts and Sammut (2005) and Verbeeck and Blockeel (2015) adapt splitting heuristics that are designed for use with model trees that use linear regressors in the leaves. On the other hand, Ikonomovska, Gama, and Džeroski (2011b) use variance reduction to evaluate the potential splits. Variance reduction is generally calculated faster and is better suited for using a mean regressor in the leaves. As we will show in the following sections, we will use models in the leaves of iSOUP-Trees, however, these will be *a combination* of a linear model and a mean regressor. Furthermore, in some applications we will explicitly use regression trees. To this end we adopt the variance reduction-based approach, as shown below.

We evaluate all split candidates $\mathcal{S}$ using a multi-target variant of the variance reduction described in Equation 4.1. In particular, we use the *intra-cluster variance reduction* (ICVR) heuristic, defined as

$$\mathrm{ICVR}(\mathcal{S}) = \frac{1}{M} \sum_{j=1}^{M} \frac{1}{\mathrm{Var}^j(S)} \left(\mathrm{Var}^j(S) - \frac{|S_\top|}{|S|} \mathrm{Var}^j(S_\top) - \frac{|S_\perp|}{|S|} \mathrm{Var}^j(S_\perp) \right), \tag{4.2}$$

to evaluate a split candidate $\mathcal{S}$, where j indexes the target variables, S is the set of the accumulated examples in the given leaf, $S_\top$ and $S_\perp$ are the post-split subsets of S that contain examples for which the considered candidate split test is evaluated either as true or false, respectively. Var^j is the variance of the j-th target, i.e.,

$$\mathrm{Var}^j(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} \left(y_i^j - \overline{y}^j \right)^2. \tag{4.3}$$

ICVR calculates what proportion of the variance of the target values is reduced when applying the candidate split's test to the set of examples S. In essence, it calculates whether the splitting subsets $S_\top$ and $S_\perp$ are more homogeneous than S with regard to the distribution of the target values. Hence, higher values of ICVR are desirable.

Note that we do not actually store examples of S, $S_\top$ and $S_\perp$. We furthermore describe how we maintain and update the necessary statistics to calculate the variances of sets S, $S_\top$, and $S_\perp$.

For each input attribute A_i, we record the best split candidate $\mathcal{S}_i : x_i \leq v_i$ that has the highest value of the splitting heuristic among all split candidates for this attribute. Let us denote[2] the heuristic value of the overall best split candidate $\mathcal{S}_{\mathbb{1}}$ as $h_{\mathbb{1}}$. This is the best candidate split on an attribute A_i that has the highest value of the splitting heuristic over the best candidate splits of other attributes. Similarly, we denote the heuristic value of the overall second best split $\mathcal{S}_{\mathbb{2}}$ as $h_{\mathbb{2}}$. Let us observe the ratio $\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}}$, which falls in the $[0, 1]$ interval by definition. As more examples become available, we observe the following sequence

$$\ldots, \frac{h_{\mathbb{2}}(k)}{h_{\mathbb{1}}(k)}, \frac{h_{\mathbb{2}}(k+1)}{h_{\mathbb{1}}(k+1)}, \frac{h_{\mathbb{2}}(k+2)}{h_{\mathbb{1}}(k+2)}, \ldots$$

where k denotes the number of examples considered in the calculation of $h_{\mathbb{1}}(k)$ and $h_{\mathbb{2}}(k)$.

Let us consider the ratio $\frac{h_{\mathbb{2}}(k)}{h_{\mathbb{1}}(k)}$ as a random variable $\mathcal{X}_k$. With each incoming example the ratio we record is a sample x_k from the distribution $\mathcal{X}_k$. When we have recorded enough samples, we can compute the observed average as $\overline{x} = \frac{1}{|S|} \left(x_1 + x_2 + \cdots + x_{|S|} \right)$. $\overline{x}$ is then a sample from the random variable $\overline{\mathcal{X}} = \frac{1}{|S|} \left(\mathcal{X}_1 + \mathcal{X}_2 + \cdots + \mathcal{X}_{|S|} \right)$.

We now want to estimate the distance between $\overline{x}$ and the expected value of the average random variable $\mathrm{E}\left[\overline{\mathcal{X}}\right]$. If the observed average is close to the actual average and the observed average is under 1 by definition, then the actual average might also fall below 1. This then implies that the split $\mathcal{S}_{\mathbb{1}}$ is in fact better than the second best split and we can split the node with confidence.

However, the actual average $\mathrm{E}\left[\overline{\mathcal{X}}\right]$ is not known, as we have only observed a finite sample S. To that end we use the Hoeffding inequality Hoeffding (1963), which allows us

[2] We use the $\mathcal{S}_{\mathbb{1}}$ notation to specify that this is the overall best split candidate and not to confuse it with $\mathcal{S}_1$, which is the best split candidate on attribute A_1.

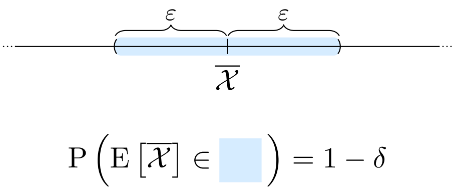

ε
ε
$\overline{\mathcal{X}}$
$\mathrm{P}\left(\mathrm{E}\left[\overline{\mathcal{X}}\right] \in \quad\right) = 1 - \delta$

Figure 4.3: A visual representation of the Hoeffding inequality.

to estimate the probability that the observed average $\overline{x}$ lies less than ε from the actual average $\mathrm{E}\left[\overline{\mathcal{X}}\right]$.

Theorem 4.1 (Hoeffding inequality). *Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be independent random variables. Suppose that each $\mathcal{X}_i$, $i = 1, \ldots, n$ is bounded, i.e., $\mathrm{P}(\mathcal{X}_i \in [a_i, b_i]) = 1$. Let $\overline{\mathcal{X}} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{X}_i$ be a random variable with expected value $\mathrm{E}\left[\overline{\mathcal{X}}\right]$. Then*

$$\mathrm{P}\left(\overline{\mathcal{X}} - \mathrm{E}\left[\overline{\mathcal{X}}\right] > \varepsilon\right) \leq e^{-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}} \tag{4.4}$$

and

$$\mathrm{P}\left(\left|\overline{\mathcal{X}} - \mathrm{E}\left[\overline{\mathcal{X}}\right]\right| > \varepsilon\right) \leq 2e^{-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}}. \tag{4.5}$$

While all of the prerequisites of the Hoeffding inequality are not met, i.e., the individual ratios are not independently distributed, as shown by Rutkowski, Pietruczuk, Duda, and Jaworski (2013), the use of the Hoeffding inequality still produces good empirical results (Ikonomovska et al., 2015).

In applying the Hoeffding inequality to the situation at hand, we note that n is equal to the size of the sample $|S|$ and that each $\overline{\mathcal{X}}$ takes values from $[0, 1]$. Therefore, the denominator of the exponent in Equation 4.5 is equal to $|S|$. Thus, we get a simplified variant of the Hoeffding inequality

$$\mathrm{P}\left(\left|\overline{\mathcal{X}} - \mathrm{E}\left[\overline{\mathcal{X}}\right]\right| > \varepsilon\right) \leq 2e^{-2|S|\varepsilon^2} =: \delta. \tag{4.6}$$

If, as above, we define the right hand side of the inequality as δ, we get an (ε, δ) approximation of the difference between an observed value of the average random variable $\overline{\mathcal{X}}$ and its expected value $\mathrm{E}\left[\overline{\mathcal{X}}\right]$. Thus, the Hoeffding inequality facilitates the probably approximately correct (PAC) learning (Valiant, 1984). A visual representation of the Hoeffding inequality is shown in Figure 4.3.

We can further express ε in terms of δ as

$$\varepsilon = \sqrt{\frac{1}{2|S|} \ln \frac{2}{\delta}}. \tag{4.7}$$

We specify the *splitting confidence* δ as a parameter of the iSOUP-Tree method (with default value of 0.0000001), while ε is calculated from δ and $|S|$.

Plugging our observation $\overline{x}$ of the $\overline{\mathcal{X}}$ random variable into Equation 4.6, we get $E[\overline{\mathcal{X}}] \in [\overline{x} - \varepsilon, \overline{x} + \varepsilon]$ with probability $1 - \delta$. It follows that, if $\overline{x} + \varepsilon < 1$, then $E(\overline{\mathcal{X}}) < 1$. Finally, this implies that $\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}} < 1$ (with probability $1 - \delta$), i.e., we have support to choose the best split $\mathcal{S}_{\mathbb{1}}$. This case, as well as the case when $\overline{x} + \varepsilon > 1$, are visually represented in Figure 4.4. In this case, we use the best split candidate $\mathcal{S}_{\mathbb{1}}$ to split the leaf. This is done by replacing the observed leaf node with a new split node with split $\mathcal{S}_{\mathbb{1}}$, under which we initialize two new leaf nodes. If $\overline{x} + \varepsilon \geq 1$, we cannot differentiate between the two best split, i.e., we do not have sufficient support for splitting the leaf node. In this case, the leaf node waits for more examples to accumulate and the procedure is repeated.

We use $\frac{h_{\mathbb{2}}(|S|)}{h_{\mathbb{1}}(|S|)}$ as an approximation of $\overline{x}$, as is commonly done by methods that utilize the Hoeffding inequality (Domingos & Hulten, 2000; Ikonomovska, Gama, & Džeroski, 2011b).

As we have stated before, the splitting criteria are only checked when enough examples have accumulated. Specifically, we check a given leaf every time a multiple of GP examples have accumulated. GP stands for *grace period* and is a parameter of the iSOUP-Tree method. We recommend a value of GP = 200, i.e., we check for splits in each leaf when 200, 400, 600, ... examples have accumulated. From Equation 4.6, it is clear that ε decreases as the number of accumulated examples increases. For example, at 200, 400 and 600 observed examples, $\varepsilon \approx 0.21$, 0.14 and 0.11, respectively, with a default δ value of 0.0000001. This means that as more examples accumulate, the ratio of $h_{\mathbb{2}}$ and $h_{\mathbb{1}}$ can be higher and still result in a split.

One drawback of using the Hoeffding inequality occurs when the heuristic scores of the two best splits $h_{\mathbb{1}}$ and $h_{\mathbb{2}}$ are very close to each other[3]. Their ratio $\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}}$ is then very close to 1. From the above, we see that the condition for splitting amounts to checking whether $\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}} + \varepsilon < 1$. Thus, when $\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}}$ is close to 1, even a very small ε value will prevent the split. In this case the two best splits are *tied* and we do not have sufficient support to differentiate them and, by design, we do not split the leaf. As the number of observed examples increases, ε falls towards 0. However, waiting to split means that we are not partitioning the input space, which is the core principle behind tree-based learning. To this end, we only allow ε to decrease so far. Once ε falls below a certain threshold τ, we no longer wait for additional evidence of the difference between best and second best split candidates and instead split the node using the best split candidate. This procedure is called *tie breaking.*

τ is a parameter of the iSOUP-Tree method and we commonly use a value of 0.05. To calculate how many examples must accumulate for tie breaking to occur, we must solve Equation 4.7 for $|S|$:

$$|S| = \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}.$$

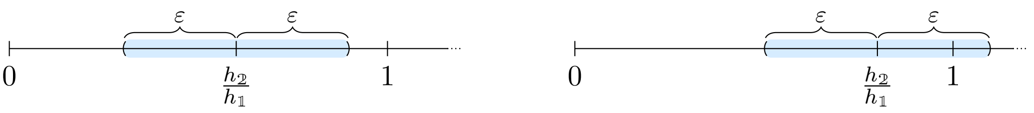
For the recommended value of $\delta = 0.0000001$, GP = 200 and $\tau = 0.05$, this means that we break ties when at least 6725 examples have accumulated, i.e., when evaluating split candidates at $6800 = 34\,\mathrm{GP}$ accumulated examples.

4.2.1.2 Maintaining and calculating the statistics in the tree nodes

As we have seen above, we need to be able to calculate the variances of various (sub)sets of examples which reach a certain leaf. In the batch setting, this is straightforward, as we have access to all examples. In the online setting, we cannot store all of the examples due its inherent constraints. Additionally, we also cannot calculate measures as easily and must employ different incrementally updated mechanisms for calculating statistics commonly used in an online learning scenario (Gama, 2010).

Let us begin with the incremental calculation of the average. If we have already calculated an average $\overline{x}$ of some number of measurements, and a new measurement x arrives, we can calculate the new average $\overline{x}'$ if we know how many measurements (k) were in the

[3] An extreme version of this scenario occurs if we encounter a dataset with repeated attributes. If the best split candidate is on a repeated attribute, the second best candidate will have the exact same heuristic value. Thus, the ratio of their heuristics will always be equal to 1.



ε
ε
0
$\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}}$
1
ε
ε
0
$\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}}$
1

(a) (b)

Figure 4.4: Application of the Hoeffding inequality to the ratio of the heuristics of the best and second best candidate splits. (a) When $\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}} < 1 - \varepsilon$ the expected heuristic ratio $\mathrm{E}\left[\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}}\right]$ is smaller than 1 with probability $1 - \delta$, thus split $\mathcal{S}_{\mathbb{1}}$ is expected to be better than split $\mathcal{S}_{\mathbb{2}}$. (b) When $\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}} > 1 - \varepsilon$ the expected heuristic ratio $\mathrm{E}\left[\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}}\right]$ is not smaller than 1 with probability $1 - \delta$. Splits $\mathcal{S}_{\mathbb{1}}$ and $\mathcal{S}_{\mathbb{2}}$ are similarly discriminative.

original average, i.e.,

$$\overline{x}' = \frac{k\overline{x} + x}{k+1},$$

where k is the number of measurements in the original average. As we will see below, it is more prudent to calculate the average in a slightly different way: instead of storing the current average, we can store the number of measurements k and the sum of all measurements Σ. Then, when we require the average, we calculate it on the fly as

$$\overline{x} = \frac{x_1 + x_2 + \cdots + x_k}{k} = \frac{\Sigma}{k}.$$

When a new measurement arrives, we add its value to Σ and increment the value of k by one.

Now we can continue with the incremental calculation of the variance of a sample of measurements. We rearrange the definition of variance of a sample of k measurements $x_1, x_2, \ldots, x_k$, to achieve a similar form as with the average:

$$\begin{aligned}
\mathrm{Var}\{x_1, x_2, \ldots, x_k\} &= \frac{1}{k}\sum_{i=1}^{k}(x_i - \overline{x})^2 = \frac{1}{k}\sum_{i=1}^{k}\left(x_i - \frac{\Sigma}{k}\right)^2 = \\
&= \frac{1}{k}\sum_{i=1}^{k}\left(x_i^2 - 2x_i\frac{\Sigma}{k} + \left(\frac{\Sigma}{k}\right)^2\right) = \\
&= \frac{1}{k}\left(\sum_{i=1}^{k}x_i^2 - \frac{2}{k}(\Sigma)^2 + \frac{1}{k^2}\sum_{i=1}^{k}(\Sigma)^2\right) = \\
&= \frac{1}{k}\left(\sum_{i=1}^{k}x_i^2 - \frac{1}{k}(\Sigma)^2\right) = \frac{1}{k}\left(\Sigma^2 - \frac{1}{k}(\Sigma)^2\right).
\end{aligned}$$

In the last line of the above equation, we have defined Σ^2 as the sum of squares of all measurements. Clearly, Σ^2 can also be easily updated with new measurements.

Therefore, to calculate the average or the variance of a sample of measurements, we need to keep an updated count of the examples k, the sum of measurements Σ and the sum of squares of measurements Σ^2.

To calculate the intra-cluster variance reduction of a split for a given attribute, we must have access to the pre- and post-split variances. To calculate the pre-split variance, we require the above statistics to calculate the variances of all of the targets. Hence, in each leaf node we keep an updated count of examples k, a sum vector Σ and a sum of squares vector Σ^2, which contain the sum of target values and the sum of squares of target values, respectively.

Nominal attributes. To calculate the post-split variances of an attribute value, we require k, Σ and Σ^2 for both split subsets. For a nominal attribute, this is straightforward, as the corresponding space of split candidates is finite. For each possible value v of the nominal attribute A, we keep the k_v, Σ_v and ${\Sigma^2}_v$ statistics, which are the examples' statistics as above, which have value v of attribute A. To obtain the other split statistics k'_v, Σ'_v and ${\Sigma^2}'_v$ of the remaining splitting subset, i.e., the subset of all examples which have a value other than v of attribute A, we subtract the value's statistics from those of the node, i.e.,

$$\begin{aligned} k'_v &= k - k_v, \\ \Sigma'_v &= \Sigma - \Sigma_v, \\ {\Sigma^2}'_v &= \Sigma^2 - {\Sigma^2}_v. \end{aligned} \tag{4.8}$$

If we know k_v, Σ_v, ${\Sigma^2}_v$ and k, Σ, Σ^2 we can calculate the intra-cluster variance reduction (ICVR) directly as

$$\mathrm{ICVR}(k_v, \Sigma_v, {\Sigma^2}_v, k, \Sigma, \Sigma^2) = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{\mathrm{Var}^j(k, \Sigma, \Sigma^2)} \cdot \\ \left(\mathrm{Var}^j(k, \Sigma, \Sigma^2) - \frac{k_v}{k} \mathrm{Var}^j(k_v, \Sigma_v, {\Sigma^2}_v) - \frac{k - k_v}{k} \mathrm{Var}^j(k - k_v, \Sigma - \Sigma_v, \Sigma^2 - {\Sigma^2}_v) \right), \tag{4.9}$$

however, as k, Σ, Σ^2 are independent of the attribute A and its value v, for brevity, we omit them from the arguments of ICVR.

Therefore, to calculate the ICVR of a nominal attribute A, we keep updated splitting statistics k_v, Σ_v and ${\Sigma^2}_v$ for all possible values v of attribute A. To maintain updated statistics for a nominal attribute, we use $\mathcal{O}(nM)$ processing time and memory, where n is the number of distinct values of the attribute, and M is the number of targets (as Σ and Σ^2 are M-dimensional vectors).

Numeric attributes. On the other hand, for a numeric attribute, the corresponding space of split candidates is infinite. However, we limit ourselves only to attribute values that have been recorded so far. As stated above, these are the only values at which the value of splitting heuristic can change. Therefore, checking all potential splits results in a time complexity of $\mathcal{O}(n)$ where n is the number of recorded values. However, we evaluate potential splits only at set intervals, while new values can be recorded with each example. To this end we wish to keep our statistics stored in a data structure, into which we can quickly insert new values and which can be quickly updated.

Extended binary search tree. To this end, we use an extended binary search tree (E-BST) data structure to store requisite values and statistics (Ikonomovska, Gama, & Džeroski, 2011b). The E-BST holds values v_i of an attribute A which have been observed so far. In each node, we maintain partial statistics k_i, Σ_i and ${\Sigma^2}_i$. We explain how to calculate the complete statistics for v_i below, after we describe how E-BST observes new examples.

Let us assume we have an E-BST and a new example e arrives, which has a value v of attribute A and the target vector y. Starting at the root node, we employ a modified binary search tree method for insertion.

At the current node, we check whether $v \leq v_i$, where v_i is the value in the current node. If it is, we update the statistics k_i, Σ_i and ${\Sigma^2}_i$, by adding 1, y and y^2 to the statistics, respectively. From here on out, we use the shorthand $y^2 = (y_1^2, y_2^2, \ldots, y_M^2)$, when y is a

Algorithm 4.3: E-BST observing an example.

```
Procedure: UpdateEBST
Input: E-BST node N, value of observed attribute $v$, target vector $y$
if $v \leq$ N.value then
    N.addToStatistics(1, $y$, $y^2$);
    if $v <$ N.value then
        if N.left is empty then
            create new node N.left with value $v$ and initialize it with $(1, y, y^2)$;
        else
            UpdateEBST(N.left, $v$, $y$);
        end
    end
else
    if N.right is empty then
        create new node N.right with value $v$ and initialize it with $(1, y, y^2)$;
    else
        UpdateEBST(N.right, $v$, $y$);
    end
end
```

vector. If value v is strictly less than v_i, then we continue with the left child of the current node. If there is no left child, we create a new node with the value v and initialize its statistics to 1, y and y^2. Otherwise, we continue recursively on the left child.

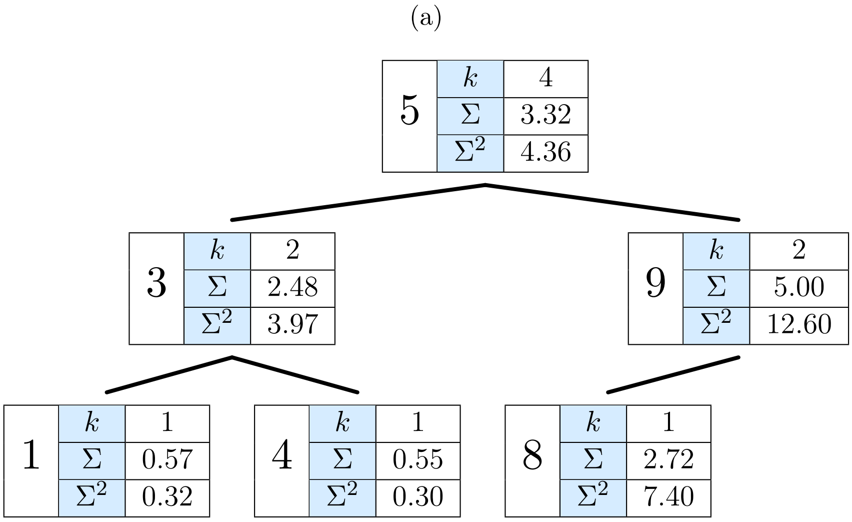
If the value v was greater than the value in the current node v_i, then we continue on the right child. Again, if the right child does not exist, we create a new node with the value v and statistics as above. On the other hand, if the right child exists, we continue recursively on it. This process is summarized in Algorithm 4.3. As the observing procedure is only ever called recursively at most once for each example, its time complexity is on average $\mathcal{O}(\log n)$ and at worst $\mathcal{O}(n)$, where n is the number of values observed so far.

At first glance, one might think that these partial statistics are not sufficient to calculate the post-split variance. If we look at a sample E-BST shown in Figure 4.5, we clearly see that 9 has a partial k value of 2, even though all of the examples have values lower than 9. In fact, the partial statistics are accurate only for nodes on the path, starting at the root and always moving left.

To calculate the complete statistics, we need another set of auxiliary statistics. Here, we show how the complete k can be calculated for a node using the auxiliary statistic $\underline{k}$. The complete values of Σ and Σ^2 are calculated in the same way.

At the beginning the auxiliary $\underline{k}$ is set to 0. We start at the root of the E-BST, i.e., at the 5 node. To get the complete k for node 5, we compute the sum of the auxiliary $\underline{k}$ and the partial k, i.e., in our case 4. This is the complete k for the node 5, resulting from examples $(5, 0.29)$, $(3, 1.91)$, $(1, 0.57)$ and $(4, 0.55)$. When we move to the left child of 5, i.e., to the node 3, we leave the auxiliary $\underline{k}$ unmodified. Then, its complete k for node 3 is 2. When we move to its left child, node 1, we again do not modify $\underline{k}$. We get a complete k of 1 for node 1.

However, when we move from node 3 to node 4, i.e., when we move to a node's right child, we add increase $\underline{k}$ by the partial k of node 3. Hence, when we are calculating the complete k of node 4, we get $2 + 1 = 3$ which corresponds to examples $(3, 1.91)$, $(1, 0.57)$ and $(4, 0.55)$. Similarly, when we move from node 5 to node 9 we add the partial k of node 5 to $\underline{k}$. Notably, in node 5 the auxiliary $\underline{k}$ was equal to 0. Thus, when we calculate the



(b)

A	T
5	0.29
9	2.28
3	1.91
1	0.57
8	2.72
4	0.55

Figure 4.5: (a) A sample extended binary search tree and (b) the values it encodes. For brevity, we show the case with only one target, hence, Σ and Σ^2 are single values instead of vectors of values.

complete k of node 9, we get 2 plus the auxiliary $\underline{k}$, which is currently equal to 4. As 9 is the largest attribute value in the table in Figure 4.5 which contains 6 attribute–target value pairs, we used to grow the E-BST, the calculated complete k of 6 is correct. Moving to node 8 we do not modify $\underline{k}$, resulting in the correct calculation of the value of the complete $k = 5$.

We only require one set of complete statistics when we are evaluating a particular potential split value v, so we interweave the calculation of the auxiliary $\underline{\textit{statistics}}$ into that procedure. This way, we keep updated complete statistics only for the value v we are currently evaluating, reducing the memory footprint of the method. The auxiliary $\underline{\textit{statistics}}$ are initialized to $(0, 0, 0)$ in the original, top-level call which starts at the root of the E-BST. We start with calculating the variances of the left path, for which the partial statistics are complete and auxiliary statistics are unnecessary. We update $\underline{\textit{statistics}}$ and maintain the current best split candidate as we search through the tree in-order depth-first, as shown in Algorithm 4.4[4]. As earlier, in Equation 4.9, we omit the statistics of all examples from the arguments of ICVR for brevity.

4.2.1.3 Leaf models: iSOUP-RegressionTree and iSOUP-ModelTree

With regards to using models in the leaves, we define two variants of the iSOUP-Tree method, one that learns regression trees (iSOUP-RegressionTree) and one that learns model trees (iSOUP-ModelTree). When we refer to the iSOUP-Tree method, we generally refer to the model tree method unless we state otherwise.

The iSOUP-RegressionTree only uses the statistics of the targets to calculate their average values, which are then used as predictions. Conveniently, we already store the statistics k and Σ that are used for the evaluation of potential splits, which we use to calculate the average values, and thus, require no further memory.

[4] The programmatically inclined reader will notice that we appear to be creating multiple instances of $\underline{\textit{statistics}}$. In practice this does not happen, as the $\underline{\textit{statistics}}$ are stored as a property of a shared object and updated when transitioning from node to node. This helps reduce memory consumption.

Algorithm 4.4: Finding the best split, according to the ICVR reduction statistic.

```
Procedure: FindBestSplit
Input: E-BST node E, auxiliary statistics statistics = $(k, \Sigma, \Sigma^2)$
Output: The best split $(v, \mathsf{merit})$
currentCandidate ← (None, $-\infty$);
if E.left exists then
    currentCandidate ← FindBestSplit(E.left, statistics);
end
merit ← ICVR (statistics + E.statistics);
if merit > currentCandidate.merit then
    currentCandidate ← (E.value, merit);
end
if E.right exists then
    rightCandidate ← FindBestSplit(E.right, statistics + E.statistics);
    if rightCandidate.merit > currentCandidate.merit then
        currentCandidate ← rightCandidate;
    end
end
return currentCandidate
```

On the other hand, the iSOUP-ModelTree method learns an adaptive multi-target model in each of its leaves, which combines a multi-target perceptron and a multi-target mean predictor, i.e., the predictor used by iSOUP-Regression tree.

Multi-target perceptron. The multi-target perceptron produces the prediction vector[5] as

$$\hat{\mathbf{y}}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}, \tag{4.10}$$

where we consider the prediction $\hat{\mathbf{y}}$ and $\mathbf{b}$ as vectors of length M, $\mathbf{W}$ is a $M \times N$ matrix and $\mathbf{x}$ is the vector of attribute values of length N. If we limit ourselves to the j-th target, we get

$$\hat{y}^j = \mathbf{w}_j\mathbf{x} + b_j,$$

where $\mathbf{w}_j$ is the j-th row of $\mathbf{W}$ and b_j is the j-th component of $\mathbf{b}$. In essence, the multi-target perceptron models each y^j as a linear function of the input parameters. In the above, b_j (and by extension $\mathbf{b}$) is needed to expand the space of possible mappings to include all linear functions, not just those that intercept the y^j-axis (or axes) at 0.

To facilitate an easier description of learning, we transform Equation 4.10 slightly, to

[5] For this part we adopt the standard notation used in linear algebra, i.e., lower-case variables written in bold font are vectors, lower-case variables of regular font weight are scalars, while uppercase variables written in bold font are matrices.

get

$$
\begin{aligned}
\hat{\mathbf{y}}(\mathbf{x}) &= \mathbf{W}\mathbf{x} + \mathbf{b} = \\
&= \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_M \end{bmatrix} \mathbf{x} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix} = \\
&= \begin{bmatrix} \mathbf{w}_1\mathbf{x}^\intercal + b_1 \\ \mathbf{w}_2\mathbf{x}^\intercal + b_2 \\ \vdots \\ \mathbf{w}_M\mathbf{x}^\intercal + b_M \end{bmatrix} = \\
&= \begin{bmatrix} [\mathbf{w}_1, b_1][\mathbf{x}, 1]^\intercal \\ [\mathbf{w}_2, b_2][\mathbf{x}, 1]^\intercal \\ \vdots \\ [\mathbf{w}_M, b_M][\mathbf{x}, 1]^\intercal \end{bmatrix} = \\
&= \mathbf{W}^*\mathbf{x}^*,
\end{aligned}
$$

where $\mathbf{W}^* = [\mathbf{W}, \mathbf{b}]$ is the matrix $\mathbf{W}$ augmented with column $\mathbf{b}$ and $\mathbf{x}^* = [\mathbf{x}, 1]$ is vector $\mathbf{x}$ augmented with an additional value 1^6. From here onward, when we refer to $\mathbf{W}$ or x, we are in fact referring to the augmented matrix $\mathbf{W}^*$ or the augmented vector x^*, respectively.

As we wish to utilize a multi-target perceptron as an online predictive model, we must know how to update it with new examples. To this end we use the Widrow–Hoff additive rule (Widrow & Hoff, 1960) also called the *delta rule*. Its reasoning and application are as follows.

We can measure the error of an output of a neural network with M outputs (in our case, the multi-target perceptron) as

$$
\mathrm{E} = \frac{1}{2}\sum_{j=1}^{M}(\hat{y}^j - y^j)^2,
$$

where $\hat{y}^j$ are components of the prediction $\hat{\mathbf{y}}$ made by the perceptron. We want to modify the weight matrix $\mathbf{W}$ in the opposite direction of the gradient of the error, since the gradient points into the direction of the growth of the error. The gradient of the error is defined as

$$
\begin{aligned}
\nabla \mathrm{E} &= \left(\frac{\partial}{\partial y^1}\mathrm{E}, \dots, \frac{\partial}{\partial y^M}\mathrm{E}\right) = \\
&= \left(\frac{\partial}{\partial y^1}\left(\frac{1}{2}\sum_{j=1}^{M}(\hat{y}^j - y^j)^2\right), \dots, \frac{\partial}{\partial y^M}\left(\frac{1}{2}\sum_{j=1}^{M}(\hat{y}^j - y^j)^2\right)\right) = \\
&= \left(\hat{y}^1 - y^1, \dots, \hat{y}^M - y^M\right) = \\
&= \hat{\mathbf{y}} - \mathbf{y}.
\end{aligned}
$$

Therefore, in the point $\mathbf{x}$ we want to move in the direction of $-\nabla \mathrm{E}$. To update $\mathbf{W}$ in a way that reflects this move, we add $-\nabla \mathrm{E}\,\mathbf{x}^\intercal = (\mathbf{y} - \hat{\mathbf{y}})\mathbf{x}^\intercal$ to it. However, since the gradient

[6] In fact, we have embedded the input space $X = \mathbb{R}^N$ into $\mathbb{R}^{N+1}$ as an affine subspace $X \times \{1\}$, in which transformations of the form $\mathbf{W}\mathbf{x} + \mathbf{b}$ are linear transformations.

is a local quantity, i.e., it changes if we move too far away, we only make a small "step" in the opposite direction of the gradient. Thus, we modify the weight matrix $\mathbf{W}$ as

$$\mathbf{W} \leftarrow \mathbf{W} + \eta(\mathbf{y} - \hat{\mathbf{y}})\mathbf{x}^{\mathsf{T}}, \tag{4.11}$$

where $\eta \in [0, 1]$ is the *learning rate.* When we learn model trees with the iSOUP-Tree method, we provide the learning rate as follows. At the perceptron's initialization the learning rate is set to the *initial learning rate* η_0 which is a parameter of the iSOUP-Tree method (with a default value of 0.2). After each incoming example, we modify the learning rate as

$$\eta = \frac{\eta_0}{1 + n \cdot \eta_\Delta},$$

where η_Δ is the *learning rate decay factor*, which is also a parameter to the model tree variant of the iSOUP-Tree method (with a default value of 0.01). This modification of the learning rate ensures that when the model has just been initialized it learns faster to facilitate faster adaptation to the new concept. With the increasing number of observed examples, the perceptron is getting closer and closer to approximating the concept as best as possible, and thus, large changes in the weight matrix are undesirable.

To ensure that the potential impact of all attributes is equal, we normalize[7] the input vector $\mathbf{x}$ before we apply the matrix $\mathbf{W}$ to it. Each input value x_i is normalized according to the standard normalization procedure, i.e., we subtract the mean value of the corresponding observed attribute values $\overline{x}$, then divide the difference by the standard deviation σ. Conveniently, the requisite statistics required to calculate these values are already recorded, as they are required for the split evaluation procedure. The normalized value of x_i is thus given by

$$x'_i = \frac{x_i - \overline{x}}{\sigma}.$$

Ikonomovska, Gama, and Džeroski (2011b) suggest a normalization in which the difference of the value and the mean is instead divided with 3σ. However, this has the exact effect as dividing the initial learning rate by 3. Therefore, modifying the normalization procedure produces the same results as modifying the learning rate. Thus, we use the standard normalization procedure and allow a user to modify the learning rate, which already appears as a parameter of the iSOUP-Tree model.

In addition to providing the update operator of an incremental method, we must also provide the initial hypothesis. We initialize the multi-target perceptron in one of two ways, based on the leaf node we are initializing the perceptron in:

- if the node is the root node of the iSOUP-Tree, the matrix $\mathbf{W}$ is initialized randomly; in particular, each element $w_{i,j}$ is uniformly randomly selected from the interval $[-1, 1]$,

- if the node was created as a result of the splitting of another node, i.e., it is a child of a split node, we record the weight matrix $\mathbf{W}$ of the leaf node that was split and use it as the initial weight matrix of the new node.

The latter initialization procedure ensures that there are no sudden jumps in the predictions after a leaf node is split. Importantly, though, only the weight matrix is reused, i.e., the learning rate is reset for the new nodes, so that they can model the new (sub)concepts faster.

[7] We normalize $\mathbf{x}$ before we augment it with the extra 1. As the value is always set to the same value, it does not need to be normalized.

Calculating the predictions in iSOUP-ModelTree. However, the multi-target perceptron is only one part of the model that is utilized as the model in iSOUP-ModelTree leaves. As discussed above, the other part is the mean regressor. In some cases, the mean regressor is a better predictor, e.g., if the concept in the part of the input space that corresponds to the leaf is a sine wave between -1 and 1. In this case, its error will never be larger than 2. On the other hand, the perceptron model will try to fit a linear function to the sine wave and spectacularly fail. Its errors can in this case be arbitrarily large.

To address this situation, we adopt the approach of Duarte et al. (2016). In a single-target scenario, they monitor the errors of the mean regressor and the perceptron, and use the predictions of the model that is currently better. This approach is called the *adaptive model*. Duarte et al. (2016) have shown that the adaptive model outperforms both the mean regressor as well as the perceptron.

We expand this methodology for the multi-target scenario as follows. For each target y^j, we monitor the errors of the mean regressor and the perceptron. We do this by recording their *fading mean absolute error*, which is defined as

$$\mathrm{fMAE}^j(e_n) = \frac{\sum_{i=1}^{n} 0.95^{n-i} |\hat{y}_i^j - y_i^j|}{\sum_{i=1}^{n} 0.95^{n-i}},$$

where e_n is the n-th observed example and $\hat{y}_i^j$ and y_i^j are the predicted and real values of the i-th example. The value 0.95 is called the *fading factor*. As it is exponentiated with larger numbers for older examples, it decreases the importance of older errors. In essence, it keeps the estimate of the error current. For example, the contribution of the first example e_1 to the fading error is equal to $0.95^{n-1}|\hat{y}_1^j - y_1^j|$, while the contribution of the latest example e_n is $|\hat{y}_1^j - y_1^j|$. Thus, we can say that the numerator in the above definition is the fading sum of mean absolute errors. Similarly, the denominator is the fading count of examples.

When a prediction must be made by the adaptive model, we compare the fading errors of the mean regressor and the perceptron. This comparison is carried out on a target-by-target basis. Then, for the j-th target, we select the prediction of the model which currently has the lower fading mean absolute error, i.e., the prediction $\hat{y}$ is defined per target as

$$\hat{y}^j = \begin{cases} \hat{y}^j_{\text{perceptron}} & \text{, if } \mathrm{fMAE}^j_{\text{perceptron}} \leq \mathrm{fMAE}^j_{\text{mean}}. \\ \hat{y}^j_{\text{mean}} & \text{, otherwise.} \end{cases}$$

When the fading errors are equal, we prefer the perceptron's prediction, though due to the nature of numerical calculations this situation is unlikely to ever occur.

The final prediction $\hat{y}$ can thus contain both predictions of the mean regressor as well as those of the perceptron. This is by design, as different targets can have radically different behaviors. If we return to the sine wave example, by adding a target which behaves as a linear function of the input attributes, we can clearly see that either model will not be able to predict both targets with a low error.

4.2.2 The iSOUP-OptionTree method

While the TDIDT algorithm is not directly applicable in a streaming setting, the proposed iSOUP-Tree method emulates the algorithm over time as more examples accumulate. Consequently, similarly to TDIDT approach in the batch setting, the proposed method suffers from myopia. One way of potentially addressing this problem is to use option trees in this setting.

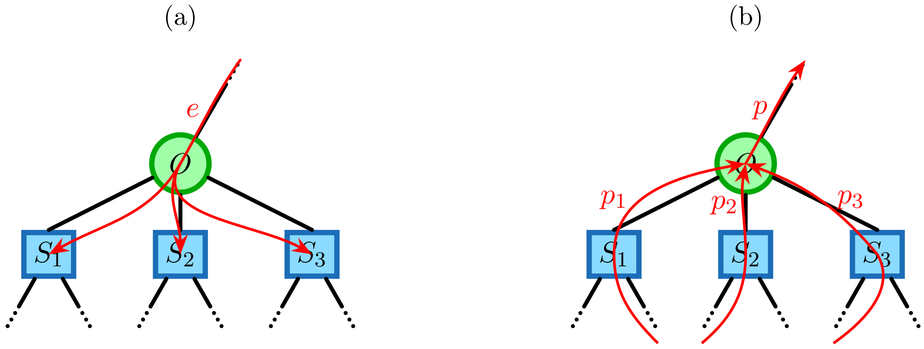


Figure 4.6: (a) Traversing an example e through an option node and (b) aggregating the options' predictions p_1, p_2 and p_3 into one prediction p.

4.2.2.1 Option trees

An option tree is an extension of a regular decision (or regression tree) that introduces an additional type of node in the tree structure, i.e., the *option node*. This was first introduced (in the batch setting) by Buntine (1992) and later expanded by Kohavi and Kunz (1997) and Osojnik, Džeroski, and Kocev (2016). In the streaming setting, option trees have been applied to the tasks of classification (Pfahringer, Holmes, & Kirkby, 2007) and regression (Ikonomovska et al., 2015).

The myopia of the TDIDT algorithm results from only ever selecting the best split at the time, even though this choice might not be optimal. In the batch case, this may be due to sampling artifacts or noise. In the streaming setting, this can be caused by insufficient statistical evidence for the selection of a given split.

Option trees address this shortsightedness by selecting multiple candidate splits when certain conditions are met. Specifically, an option node is introduced when we do not have enough heuristic-based support to split the leaf. In that case, instead of a split node, an option node is created from a leaf of the tree. Each of its children, called options, is a split node with the split corresponding to one of the selected candidate splits. Each of the split nodes is further split into leaf nodes as is the case in regular trees. When enough support is present, a single split is constructed, much in the same way as in a regular tree.

Following from the more complex learning procedure, using an option tree for learning and prediction is also more complex. In a regular tree, each example reaches exactly one leaf. In an option tree, however, an option node can cause an example to reach multiple leaves. Specifically, when an example passes through an option node, it does not choose only one of the paths as in a split node. Instead, the example is copied once for each option. Each copy is then traversed down the option nodes, as shown in Figure 4.6a and Algorithm 4.5. Therefore, for learning, the example affects all of the options (and their associated subtrees) of an option node.

In this way, an example can reach more than just one leaf. If more than one leaf is reached, this means that the example passed through at least one option node. Each of the leaves that were reached by an example produces a prediction which is then aggregated through the option node in a bottom-up manner. Specifically, an option node will receive one prediction p_i from each of its options, as seen in Figure 4.6b. The prediction p of the option node is then an aggregation of this set of predictions. In particular, we use

Algorithm 4.5: Traversal in an option tree.

```
Procedure: TraverseOpt
Input: option tree T, example e
Output: set of reached leaves
node ← T.rootNode;
if node is a leaf node then
    return {node}
else if node is a split node then
    direction ← node.direction(e);
    return TraverseOpt(node.children(direction), e)
else                                              // node is an option node
    leaves ← {};
    for i ← 1 to node.numChildren do
        leaves ← leaves ∪ TraverseOpt(node.children(i), e);
    end
    return leaves
end
```

averaging to obtain the prediction of the option node, i.e.,

$$p = \frac{1}{n} \sum_{i=1}^{n} p_i,$$

where n is the number of options. This prediction is then passed further up, where it may be further aggregated in an option node at a higher level.

The property that each example can reach multiple leaves is shared with tree ensembles, as we will see in the following sections. There, however, an example reaches multiple leaves in different (base) models, while in an option tree the leaves are part of the same option tree.

We usually consider an option tree as a single tree, however, it can also be interpreted as a compact representation of a tree ensemble. To generate the ensemble of the *embedded trees*, we start recursively from the root node and move in a top-down fashion. Each time we encounter an option node we copy the tree above (and in “parallel”) for each of the options and replace the option node with only the option, i.e., one of the split nodes. This produces one tree for each option while removing the option node in question. We repeat this procedure on all the generated trees until we are left with no option nodes. This is illustrated in Figure 4.7. For this reason, we sometimes refer to option trees as pseudo-ensembles.

4.2.2.2 Extending iSOUP-Tree to utilize option nodes

The **iSOUP-OptionTree** method is the option tree extension of the iSOUP-Tree method for the task of multi-target regression. The extension is done in a similar way as the Online Regression Trees with Options (ORTO) method extends the FIMT-DD method for the single-target regression task (Ikonomovska et al., 2015). The pseudocode of the update operator of the iSOUP-OptionTree method is shown in Algorithm 4.6. The starting hypothesis is again a single leaf node.

As in the case of iSOUP-Trees, the Hoeffding inequality is used to grow the tree. In addition to it being used to split leaf nodes into split nodes, it is also used as a criterion when to introduce an option node. If, when evaluating splits, we encounter the case where

Algorithm 4.6: The iSOUP-OptionTree update operator.

```
Input: current iSOUP-OptionTree T, new example e
Output: updated iSOUP-OptionTree T
leaves ← TraverseOpt(T, e);
foreach leaf ∈ leaves do
    leaf.updateStatistics(e);
    leaf.updateModel(e);
    if leaf.examplesSeen mod GP = 0 then
        foreach attribute $A_i$ do
            $\mathcal{S}_i, h_i$ ← BestSplitForAttribute($A_i$);
        end
        sort split candidates $(\mathcal{S}_i, h_i)$ according to their decreasing heuristic scores;
        $\mathcal{S}_{\mathbb{1}}, h_{\mathbb{1}}$ ← best split according to the heuristic score;
        $\mathcal{S}_{\mathbb{2}}, h_{\mathbb{2}}$ ← second best split according to the heuristic score;
        $\varepsilon$ ← HoeffdingBound($\delta$, leaf.observedExamples);
        if $\frac{h_{\mathbb{2}}}{h_{\mathbb{1}}} < 1 - \varepsilon$ or $\varepsilon \leq \tau$ then
            splitNode ← SplitNode($\mathcal{S}_{\mathbb{1}}$);
            splitNode.left ← LeafNode(leaf);
            splitNode.right ← LeafNode(leaf);
            replace leaf with splitNode;
        else if depth(leaf) $\leq d_{\max}$ and $N \cdot \beta^{\mathrm{depth(leaf)}} \geq 1$ then
            optionNode ← OptionNode();
            $\mathbb{i}$ ← 1;
            while $\mathbb{i} - 1 \leq \lfloor N \cdot \beta^{\mathrm{depth(leaf)}} \rfloor$ and $\mathbb{i} \leq 5$ do
                if $\frac{h_{\mathbb{i}}}{h_{\mathbb{1}}} > 1 - \varepsilon$ then
                    splitNode ← SplitNode($\mathcal{S}_{\mathbb{i}}$);
                    splitNode.left ← LeafNode(leaf);
                    splitNode.right ← LeafNode(leaf);
                    optionNode.addOption(SplitNode(splitNode));
                    $\mathbb{i} \leftarrow \mathbb{i} + 1$;
                end
            end
            replace leaf with optionNode;
        end
    end
end
return T
```

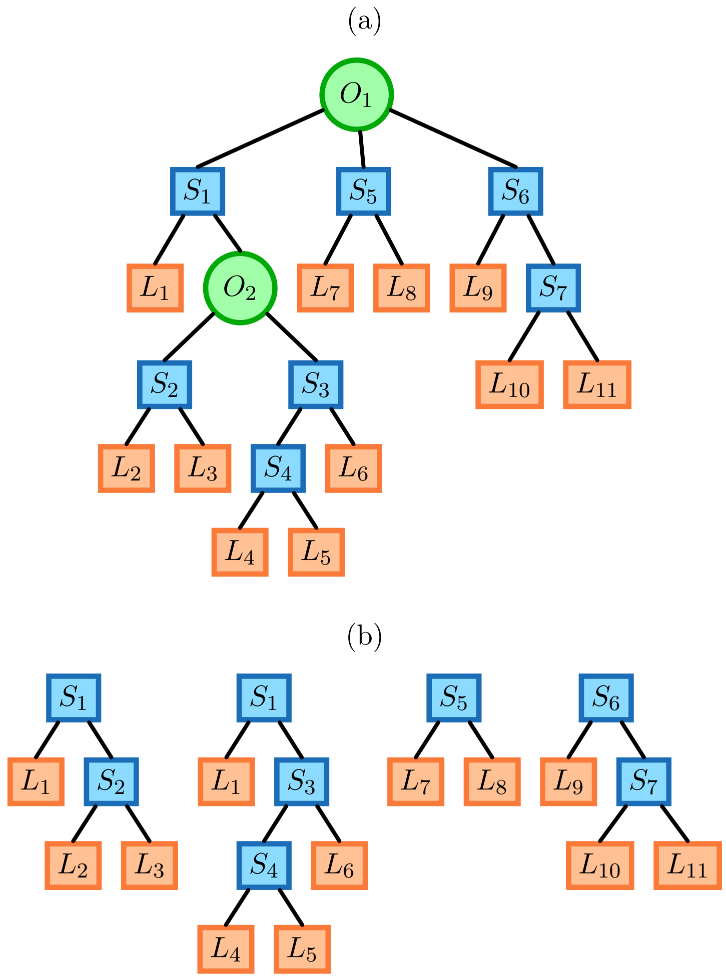


Figure 4.7: An option tree (a) and its embedded trees (b). Green nodes O_i are option nodes, blue nodes S_j are split nodes and orange nodes L_k are leaf nodes.

$\overline{x} + \varepsilon < 1$, a split node is grown as in a regular tree. However, if $\overline{x} + \varepsilon > 1$, we do not have enough evidence to split the node according to the best split, i.e., we do not have enough evidence to differentiate between the best and second best splits. Therefore, we introduce an option node, with options for which the following holds

$$\frac{h_{\mathbb{i}}}{h_{\mathbb{1}}} > 1 - \varepsilon,$$

where $\mathbb{i}$ enumerates all of the input attributes in descending order of the corresponding heuristic scores. The reasoning behind this is the following: all splits for which $\overline{x} + \varepsilon < 1 \sim \frac{h_{\mathbb{i}}}{h_{\mathbb{1}}} + \varepsilon < 1$ does not hold, are approximately equally discriminative, i.e., they are about as discriminative as the best split. The fact that the condition is not met is interpreted as the lack of evidence towards discarding of these splits. This concisely determines the candidates for the options in an option node.

However, we do not (necessarily) select all of the options, i.e., candidate splits. Kohavi and Kunz (1997) have suggested that option nodes are best induced higher in the tree, where they affect more of the data examples. To this end we select only a portion of the candidate splits, in order to decrease the heuristic score, until $N \cdot \beta^{\mathrm{depth}(L)}$ are selected in addition to the best split candidate, $\beta \in (0, 1]$ is the option decay factor and $\mathrm{depth}(L)$ is the depth of the leaf node L we are replacing with the option node.

The parameter β regulates the rate of induction of option nodes as we descend lower into the tree. As usual, the root note has a depth of 0. Notably, we do not count option nodes when we are calculating the depth of a node and their depth is the same as their children's, as option nodes are used to represent several "parallel" split nodes. The construction of these parallel splits (contained within the option nodes) is the mechanism through which option trees attempt to address the myopia of the greedy tree building procedure. However, selecting too many options can lead to the combinatorial explosion of the size of the tree. Therefore, only up to 5 candidates are ever selected, and option nodes are never induced above level $d_{\max}$.

As a consequence of the above, each node above level $d_{\max}$ is (generally) tested for splitting only once, as one of two possibilities happen. Either there is enough evidence to split the node, or there are at least two candidate splits, the best split and the second best split, which satisfy the condition of the option split candidates. The only case where this does not happen, is when $\beta^{\mathrm{depth}(L)}$ falls below $\frac{1}{N}$. In that case, the leaf node can go through multiple split evaluations. This means that the option tree grows faster than a regular tree.

Let us consider a complete option node, that is, an option node which has the complete 5 options. Let us further assume that there are no option nodes further down in the tree. Since we have 5 options and no options lower in the tree, we have a total of 5 embedded trees. Now, let us consider the number of its embedded trees, when we add two such nodes under a split node, i.e., each leaf of a split node was extended into a complete option node. To construct an embedded tree we can now choose one of the 5 options when the test is satisfied and one of 5 options when it is not. This results in 25 different embedded trees. It can be inferred, that to calculate the number of embedded trees for an option node, we need to sum up the number of embedded trees for each of its options. In a split node, however, we multiply the numbers of embedded trees of the subtree that satisfy the split and of the subtree that does not, to obtain the total number of embedded trees.

Given the construction constraints described above, we know that option nodes with up to 5 options can appear only on the first 3 levels, i.e., levels 0, 1 and 2 ($d_{\max}$). If we now consider a complete standard option tree[8], we calculate a maximum of $(5^2 \cdot 5)^2 \cdot 5 = 5^7 = 78125$ embedded trees. However, many of these trees overlap to a large extent.

Notably, a given example will not traverse the entire option tree. For example, in Figure 4.7, if an example traverses through S_1 into its left child L_1, the same result would happen in both the first and second embedded tree. The example is "agnostic" of any option nodes in the right child of S_1. Therefore, in a complete option tree, a given example will visit only up to $5^3 = 125$ leaves, as it will only traverse down one side of the tree in each split node.

In other words, there are a maximum of 125 different predictions that would be aggregated in order to obtain the final prediction of a complete option tree. This can be compared to a single tree, where only 1 prediction will be made for each example, or to a tree ensemble, where n prediction would be made, where n is the size of the ensemble, and then aggregated to produce the final prediction.

4.2.3 Ensembles of iSOUP-Trees

Ensemble methods utilize an ensemble, i.e., a collection, of *base models*. When all the base models are of the same type, the ensemble is *homogeneous*, while otherwise it is *heterogeneous*. Ensemble methods are popular since they generally produce great results

[8] An option tree with the recommended parameter values.

in terms of predictive performance (Aho et al., 2009; Kocev et al., 2013; Bifet et al., 2009).

Ensemble methods strive to learn *diverse* base models. To achieve diversity several approaches are commonly employed. For example, in bagging (Breiman, 1996) the diversity is a result of resampling of the training data examples. Each base model then learns from a different sample of the dataset, which produces diverse models. On the other hand, in random forests (Breiman, 2001) the base models are randomized at each step of learning to only consider a subset of the input attributes instead of all of them.

For each example, each base model of an ensemble produces a prediction. These predictions are then aggregated to produce the ensemble's prediction. This is how the ensembles achieve good performance. By aggregating the predictions of diverse base models that make errors on different parts of the input space, ensembles produce errors with lower bias. Most commonly, we aggregate the predictions by averaging, weighted averaging or using the median prediction.

In this thesis, we focus on online bagging and online random forests and we use iSOUP-Trees as base models.

4.2.3.1 Online bagging: iSOUP-Bag

A popular method for learning ensembles in the batch setting is *bootstrap aggregation — bagging* (Breiman, 1996). To introduce variability among the constituents of the ensemble, each base model is learned on a bootstrap replicate of the original data. Each bootstrap replicate is in fact a resampling with replacement of the original data.

The use of the batch approach for constructing bagging ensembles is not feasible in the streaming setting, as the data is not fully available at any given point in time. To address this problem Oza and Russel (2001) have introduced the online bagging procedure, which maintains several properties from the batch approach. Notably, a given data example in a data set of n examples can appear multiple times in a bootstrap replicate. The probability that it does not appear can easily be calculated as

$$\mathrm{P}(\text{given example does not appear}) = 1 - \left(1 - \frac{1}{n}\right)^n .$$

If $n \to \infty$, this converges to $1 - \frac{1}{e}$. Similarly, when $n \to \infty$, we can expect $\frac{1}{e}$ of all examples to be repeated in each bootstrap replicate. The number of repetitions of a given example in a bootstrap replicate is distributed according to the binomial distribution $\mathrm{B}(n, \frac{1}{n})$. When $n \to \infty$, this distribution tends to the Poisson distribution with $\lambda = 1$, i.e., to Poisson(1).

This motivates the following use of online bagging and using our developed iSOUP-Trees as base models. This method is called **iSOUP-Bag**. For each incoming data example and each base model in the ensemble, we sample the number of repetitions k according to the Poisson(1) distribution for that example-model pair. The selected base model then learns from the current data example k times, which introduces variety in the base models.

4.2.3.2 Online random forest: iSOUP-RF

Oza and Russel (2001) also introduced an extension of the random forest methodology (Breiman, 2001) for data streams. Where online bagging is agnostic to the selection of the base method, the random forest requires an adaptation of the base tree-based model method.

In order to develop an online random forest method for our setting, we modified the proposed iSOUP-tree method, which is used as a base method, in the following way. Whenever a leaf node is constructed, i.e., at the beginning of the learning procedure as the initial

Algorithm 4.7: Online bagging update operator.

```
Input: current ensemble E, incoming example e
Output: updated ensemble E
foreach model M ∈ E do
    $k \leftarrow \mathsf{Poisson}(1)$;
    if $k > 0$ then
        for $i \leftarrow 1$ to $k$ do
            M.update(e);
        end
    end
end
return E
```

hypothesis or when a leaf node is split into two new leaf nodes, a subset of the input attributes is randomly selected. The statistics are then recorded only for the selected input attributes. Consequently, only the selected attributes can be selected as splits. The method constructed this way is named **iSOUP-RF**.

The random forest methodology greatly alleviates the stress on the consumption of computational resources, as only a portion of the statistics are stored, thus requiring less memory. Additionally, fewer splits are considered when splitting a leaf, making this approach considerably faster.

There are several suggested values for the portion of input attributes to be considered in a given leaf. The most common are $\sqrt{N}$, $\lceil \log N \rceil + 1$ and $\alpha \cdot N$, where N is the total number of attributes and $\alpha \in (0, 1)$. In addition to the randomization of the base model, the online random forest procedure includes the online bagging procedure, i.e., we randomly sample the attribute space as well as the dataset.

4.2.4 The local FIMT-DD method

We also introduce a *local approach* for online multi-target regression based on the single-target FIMT-DD method (Ikonomovska, Gama, & Džeroski, 2011b). Here, for each target, we learn one FIMT-DD model. The predictions of these single-target models are then combined to produce the final, multi-target prediction.

As iSOUP-Tree is based on FIMT-DD, the two methods operate in a very similar way. Thus, we can apply the single-target variant of the adaptive model (Duarte et al., 2016) to FIMT-DD. This combination is referred to as the **local FIMT-DD** method.

4.3 Online Multi Label-Classification via Online Multi-Target Regression

As we have stated earlier, we design iSOUP-Tree to be applicable to multiple online structured output prediction tasks. Thus, we use the already introduced methods for online multi-target regression to address other types of online SOP tasks. In particular, we address online multi-label classification by transforming an online multi-label classification task into an online multi-target regression task and apply the methods introduced earlier.

$$\begin{tikzcd}
 & \textbf{Target space} & & \textbf{Example} \\
\textbf{MLC} & y \subseteq \mathcal{L} = \{\lambda_1, \dots, \lambda_n\} \arrow[dd] & & y = \{\lambda_1, \lambda_3, \lambda_4\} \arrow[dd] \\
 & & \textit{transformation} & \\
\textbf{MTR} & y \in \mathbb{R}^M & & y = (1, 0, 1, 1, \dots)
\end{tikzcd}$$

Figure 4.8: Transforming a multi-label classification task to a multi-target regression task.

4.3.1 Problem transformation methodology

The problem transformation methods described in Section 3.2.2 generally transform a multi-label classification task into one, or several, binary or multi-class classification tasks. In this thesis, we take a different approach and transform a classification task into a regression task. The simplest example of a transformation of this type is to transform a binary classification task into a regression task. For example, if we have a binary target with labels *yes* and *no*, we would consider a numeric target to which we would assign a numeric value of 0 if the binary label is *no* and 1 if the binary label is *yes*.

In the same way, we can approach the multi-class classification task. Specifically, if the multi-class target variable is ordinal, i.e., the class labels have a meaningful ordering, we can assign the numeric values from 0 to $n-1$ to each of the corresponding n labels. This makes sense, since if the labels are ordered, a misclassification of a label into a “nearby” label is better than a misclassification into a “distant” label. However, if the variable is not ordinal, this makes less sense, as any given label is not in a strict relationship with other labels.

In that case, an approach similar to that introduced by Frank, Wang, Inglis, Holmes, and Witten (1998) to address multi-class classification using regression can be used. In their case, they produced several versions of the observed data, one version per class in the multi-class classification task. For each class, its version of the data featured a derived binary classification target, which corresponded to the presence of the class. Consequently, for each class a model tree regressor was learned. For a given example, the prediction of each of the trees was calculated, after which the example was classified into the class with the highest corresponding (numeric) tree prediction. This approach produces one regressor per target, however, with the use of methods for multi-target regression, this can be reduced to one (multi-target) regressor for all of the targets.

4.3.2 Transforming multi-label classification to multi-target regression

To address the multi-label classification task using regression, we transform it into a multi-target regression task (see Figure 4.8). This procedure is performed in two steps: first, we take the viewpoint that the multi-label classification target is composed of several binary classification variables, just as in the binary relevance method described in Section 3.2.2. However, instead of training one classifier for each of the binary variables, we further transform the values of the binary targets into numeric values. For a given example, a numeric target corresponding to a given label has a value of 1 if the label is present, and a value of 0 if the label is not present. This exactly coincides with the common representation of multi-label classification targets, i.e., of targets of type `set(discrete(`$\mathcal{L}$`))`, as we described in Chapter 2.

For example, we transform a multi-label classification task with three labels $\mathcal{L} = \{\text{red}, \text{blue}, \text{green}\}$ into a multi-target regression task with three numeric target variables

$$\begin{tikzcd}
& \textbf{Target space} & & \textbf{Example} \\
\textbf{MTR} & \hat{y} \in \mathbb{R}^M \arrow[d] & & \hat{y} = (0.98, 0.21, 0.59, 0.88, \dots) \arrow[d] \\
& & \textit{thresholding} & \\
\textbf{MLC} & \hat{y} \subseteq \mathcal{L} & & \hat{y} = \{\lambda_1, \lambda_3, \lambda_4\}
\end{tikzcd}$$

Figure 4.9: Transforming a multi-target regression prediction into a multi-label classification prediction.

y^{red}, y^{blue}, $y^{\text{green}} \in \mathbb{R}$. If an example is labeled with red and green, but not blue, the corresponding numeric targets will have values $y^{\text{red}} = 1, y^{\text{blue}} = 0$, and $y^{\text{green}} = 1$.

Once we have learned a multi-target regressor, we use it to make predictions. Since we are using a regressor, it is possible and probable that a prediction for a given example and label will not result in a value of exactly 0 or 1 for each of the targets. For this purpose, we use thresholding to transform a multi-target regression prediction back into a multi-label one (see Figure 4.9). Namely, we construct the multi-label prediction in such a way that it contains labels of which the corresponding numeric values are over a certain threshold, i.e., in our case, the labels selected are those with corresponding numeric values over the *classification threshold* of $\tau = 0.5$. It is clear, however, that choosing a different threshold leads to different predictions.

In the batch setting, thresholding can be performed in the pre- and post-processing phases. However, in the streaming setting it needs to be done in real time. Specifically, the process of thresholding occurs at two times. The first thresholding occurs when the multi-target regressor has produced a multi-target prediction, which must then be converted into a multi-label prediction. The second thresholding occurs when we are updating the regressor, i.e., when the regressor is learning. Most streaming regressors are heavily dependent on the values of the target variables in the learning process, so the examples must be converted into the numeric representation that the multi-target regressor can utilize.

The problem of thresholding is not only problematic in the scope of the MLC via MTR methodology, but also when addressing the multi-label classification task with other approaches. In general, multi-label classification models produce predictions which are interpreted as probability estimations for each of the labels and, thus, the thresholding problem is a fundamental part of multi-label classification.

Notably, the MLC via MTR problem transformation methodology is not applicable online in the online learning setting, but can also be applied in the batch learning setting.

4.3.3 Methods for online multi-label classification

We select several of the methods we introduced for online multi-target regression to apply to the online multi-label classification task. In particular, we select two single-tree methods, iSOUP-RegressionTree (under the name **RT**) due to the concerns raised earlier and iSOUP-ModelTree (under the name **MT**) as it achieves the good predictive performance. Additionally, we also select bagging of regression trees (**E$_\mathbf{B}$RT**) and model trees (**E$_\mathbf{B}$MT**), as these tend to behave very well in terms of predictive performance.

4.4 Methods for Online Hierarchical Prediction

In order to address online hierarchical prediction, we utilize a weighted splitting heuristic, which has been used in the batch setting for hierarchical multi-label classification by Blockeel et al. (2006) and Vens et al. (2008) and for hierarchical multi-target regression by Mileski (2017). The weighted heuristic assigns a weight to each target or label, based on its location in the hierarchy. For this extension of iSOUP-Tree, we specify how the approach is used for hierarchical multi-label classification and hierarchical multi-target regression.

The weighted ICVR heuristic is calculated as before in Equation 4.2, however instead of using regular variance defined in Equation 4.3, we use the weighted variance

$$\mathrm{wVar}^j(S) = w_j \frac{\sum_{i=1}^{|S|}(y_i^j - \overline{y}^j)}{|S|}$$

where w_j is the weight of the j-th target, which is calculated as

$$w_j = w_0^{\mathrm{depth}(j)}$$

where $w_0 \in \mathbb{R}^+$ is the weight of the root node and $\mathrm{depth}(j)$ is the average depth of the j-th target over all paths from the root to it in the hierarchy. In the case of a tree hierarchy, this $\mathrm{depth}(j)$ coincides with the standard definition of depth. When the weight of the root node is greater than 1, i.e., $w_0 < 1$, a larger emphasis is placed on the variances of the targets/labels higher in the hierarchy, i.e., nodes which are closer to the root of the hierarchy. This aims to address the fact that a wrong prediction higher in the hierarchy is more detrimental than a mistake lower in the hierarchy, i.e., wrongly predicting a high-level concept results in wrong predictions of the lower-level concepts due to the hierarchy constraint.

On the other hand, when $w_0 > 1$, variances of the targets/labels deeper in the hierarchy, in particular of leaf targets/labels, are emphasized. This directly prioritizes splits which reduce the variances of the leaf targets/labels first and foremost. The variances of non-leaf targets/labels are then used as a discrimination factor for splits with similar variances in the leaf targets/leaves.

We use the weighted heuristic directly to address hierarchical multi-target regression, while in addressing hierarchical multi-label classification we combine it with the MLC via MTR methodology.

4.5 Methods for Online Semi-Supervised Multi-Target Regression: SSL-iSOUP-PCT

To address online semi-supervised structured output prediction tasks we utilize the predictive clustering framework (Blockeel & De Raedt, 1998; Struyf & Džeroski, 2006). Levatić et al. (2017b) utilized this approach to great success in addressing semi-supervised structured output prediction in the batch setting and we adapt their approach to the online setting.

4.5.1 Predictive clustering trees

Predictive clustering trees (PCTs) are a state-of-the-art method for structured output prediction in the batch setting for a wide selection of output structures (Struyf & Džeroski, 2006; Vens et al., 2008; Slavkov & Džeroski, 2010; Kocev et al., 2013). They utilize the predictive clustering framework (Blockeel & De Raedt, 1998), which connects the tasks

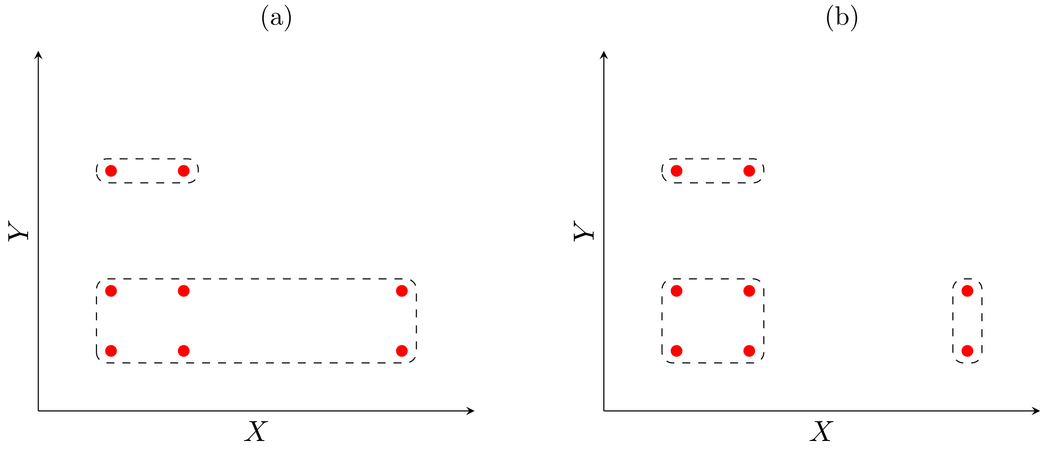


Figure 4.10: The regular splitting heuristic only seeks to improve the homogeneity in the target space Y (a). The predictive clustering heuristic additionally seeks to improve homogeneity in the input space X (b). Figure adapted from Blockeel (1998).

of predictive modeling and clustering. Briefly said, in predictive clustering all attributes are seen as part of the domain for clustering and targets for predictive modeling, as well. PCTs work under the assumption that grouping similar examples together, i.e., clustering, can improve the predictive performance. They use a modified splitting heuristic that takes into account not only the homogeneity of the targets, but also the homogeneity of input attributes. An example can be seen in Figure 4.10.

To take into account the homogeneity of the input attributes in the splitting heuristic we modify the standard intra-cluster variance reduction definition (Levatić et al., 2017b) to include the variance of the input attributes

$$\mathrm{ICVR} = w \cdot \frac{1}{M} \sum_{j=1}^{M} \frac{1}{\mathrm{Var}^j(S)} \left(\mathrm{Var}^j(S) - \frac{|S_\top|}{|S|} \mathrm{Var}^j(S_\top) - \frac{|S_\perp|}{|S|} \mathrm{Var}^j(S_\perp) \right) + \\ + (1-w) \cdot \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\mathrm{Var}_i(S)} \left(\mathrm{Var}_i(S) - \frac{|S_\top|}{|S|} \mathrm{Var}_i(S_\top) - \frac{|S_\perp|}{|S|} \mathrm{Var}_i(S_\perp) \right),$$

where $w \in [0,1]$ is the *level of supervision* and $\mathrm{Var}_i(S)$ is the variance of the values of the i-th input attribute over set S. When $w = 1$, this definition coincides with the regular definition of intra-cluster variance reduction. When $w = 0$, we consider the input attributes exclusively, when estimating the homogeneity of the different split subsets. In essence, we are grouping the examples according to their similarity, instead of trying to minimize an evaluation measure, which is equivalent to solving the clustering data mining tasks. In the batch learning approach, the w parameter is chosen based on internal cross-validation that determines the appropriate w for each individual dataset. In the online scenario, this kind of procedure is unfeasible. Thus, we select a midpoint value of $w = 0.5$.

4.5.2 Adapting SSL PCTs to the online setting

When we look at the original definition, all of the quantities require that we know the values of the targets. Therefore it is not possible to utilize any unlabeled examples. We use a methodology similar to Levatić et al. (2017b) and use the expanded definition of the ICVR heuristic to obtain information from the unlabeled examples. This is the defining characteristic of the **iSOUP-PCT** method.

5	k	2		$\tilde{k}$	4	
	Σ	3.32	10.19	$\tilde{\Sigma}$	15.05	24.05
	Σ^2	5.56	75.93	$\tilde{\Sigma^2}$	58.45	196.82

Figure 4.11: A sample node of a modified E-BST used for semi-supervised learning, which measures the statistics for input attributes $(\tilde{k}, \tilde{\Sigma}, \tilde{\Sigma^2})$ in addition to the statistics of the targets (k, Σ, Σ^2). In this example, 2 attributes and 2 targets are present.

To facilitate the calculation of the expanded ICVR, we modify the E-BST data structure described in Section 4.2.1 to record the statistics of the input attributes in addition to the statistics of the targets, as shown in Figure 4.11. We record the statistics (including the counts) of the attributes and targets separately, as both labeled and unlabeled examples contribute to the statistics of the input attributes $(\tilde{k}, \tilde{\Sigma}, \tilde{\Sigma^2})$, while only labeled examples contribute toward the target statistics (k, Σ, Σ^2). The update procedure is extended in the same manner as the individual nodes, i.e., by also providing the value of the input attributes. Notably, the update procedure also works when only the input attributes are given, by updating only the corresponding statistics.

This modification, however, incurs a heavy cost on the consumption of resources. In a regular E-BST, we use $\mathcal{O}(n \cdot M)$ units of memory to record the statistics, while in the modified E-BST, we use $\mathcal{O}(n(N + M))$ units of memory, where n is the number of unique attribute values recorded in the tree. Similarly, in the regular E-BST the updating procedure had (on average) a time complexity of $\mathcal{O}(\log n \cdot M)$, where as the update and insertion in a modified E-BST has a time complexity $\mathcal{O}(\log n \cdot (N + M))$. Given that we keep one E-BST for each attribute, this increases the complexity of the memory consumption from $\mathcal{O}(n_{\max} NM)$ to $\mathcal{O}(n_{\max}(N + M)N)$ and the time complexity of updating the trees from $\mathcal{O}(\log n_{\max} \cdot NM)$ to $\mathcal{O}(\log n_{\max} \cdot (N + M)N)$, where $n_{\max} = \max\{\text{number of distinct values of attribute } A\}$ over all attributes A. Therefore, in problems where there are strict constraints on either learning time or memory consumption, we must be careful in applying iSOUP-PCTs, especially, when the number of input attributes is large, as both complexities are quadratic in the number of attributes.

However, this issue does not appear in the batch case. As all examples are available throughout the learning process, there is no additional need to record any kind of statistics as they can be calculated on the fly. This has propelled semi-supervised PCTs to great performances for SSL data mining tasks (Levatić et al., 2017b).

We apply another adaptation in the initialization of new leaf models, when splitting a leaf. In particular for semi-supervised learning, we do not recommend the use of the adaptive perceptron. As we have shown above, when a leaf node is split, the current linear coefficients of the perceptron are copied to the initial perceptrons of the new leaves. However, as the perceptron updating procedure only works on labeled examples, in streams with a low ratio of labeled examples, it takes a while for the two perceptrons to produce substantially different predictions.

To this end we use the mean regressor in the leaves. However, it too, requires labeled examples to start accurately modeling the different input subspaces. To facilitate faster learning, we harness the splitting procedure. To evaluate various possible splits, we calculate the pre- and post-split variances. In particular we have access to the splitting heuristics of both splitting subsets, i.e., k, Σ, Σ^2 and $k', \Sigma', \Sigma^{2\prime}$. The split we select, is by definition the one that produces the most homogeneous splitting subsets. For each leaf corresponding to a splitting subset, we set the appropriate counts and sums used by the mean predictor to k and Σ (or k' and Σ', as is appropriate). These mean predictors utilize the past (labeled) examples to the fullest extent, from the moment of the splitting of a leaf.

Notably, we can use an iSOUP-PCT as a supervised method for online semi-supervised multi-target regression. When we are specifically using iSOUP-PCTs as semi-supervised methods, i.e., when they learn from unlabeled examples in addition to the labeled ones, we call the method **SSL-iSOUP-PCT**. Furthermore, it is straightforward to combine the iSOUP-PCT method with the MLC via MTR problem transformation methodology to address online semi-supervised multi-label classification, or even with the extensions toward hierarchical prediction.

4.6 Methods for Online Feature Ranking with Symbolic Random Forests

To address online feature ranking using the iSOUP-Tree family of methods, we adapt the symbolic random forest feature ranking method recently introduced by Petković et al. (2017) in the batch setting to the online learning setting. In particular, Petković et al. (2017) have introduced several feature ranking methods for multi-target regression based on ensembles of PCTs. They introduce the symbolic ranking method, which calculates the feature importances directly from the structure of ensemble members, by employing the Genie3 ranking method, which calculates the feature importances based on the heuristic scores produced by the split nodes in the ensemble members. The authors also introduce the symbolic random forest feature ranking method, which calculates the scores of the attributes by looking at the out-of-bag errors, i.e., errors on examples that were not used for learning.

Of these three methods, only the symbolic random forest ranking method is directly applicable to the online learning scenario. To calculate the Genie3 feature importances, we need access to splitting heuristic scores, which are easily accessed in the batch scenario. In online learning, the heuristic score of a split was calculated only on a small sample of the data, and is only partially indicative of the score on the entire dataset.

The random forest method permutes the values of out-of-bag examples for each tree and observes how the error changes from the original, unpermuted example. This requires the permutation of many example values, after which many predictions must be calculated to estimate the error. While this approach could technically be applied to online learning, it would incur high consumption of computational resources, particularly in processing time.

Symbolic random forest feature ranking, however, calculates the feature importances using only the structure of the members of the ensemble. Particularly, no predictions are needed, which significantly reduces the operational time of the method. To calculate the total feature importance score of an attribute A, we first calculate the feature importance score of A for a given ensemble member T, which is defined as

$$\mathrm{I}(A, T) = \sum_{\mathcal{N} \in T(A)} w^{\mathrm{depth}(\mathcal{N})},$$

where w is a predefined weight and $T(A)$ is the set of all split nodes of tree T which have splits on attribute A. The total feature importance of attribute A is then

$$\mathrm{I}(A, E) = \frac{1}{|E|} \sum_{T \in E} \mathrm{I}(A, T) = \frac{1}{|E|} \sum_{T \in E} \sum_{\mathcal{N} \in T(A)} w^{\mathrm{depth}(\mathcal{N})},$$

where E is the ensemble of trees. We adapt this method to online learning, as the calculation of the scores is quick, since it requires only the traversal of each tree in the ensemble.

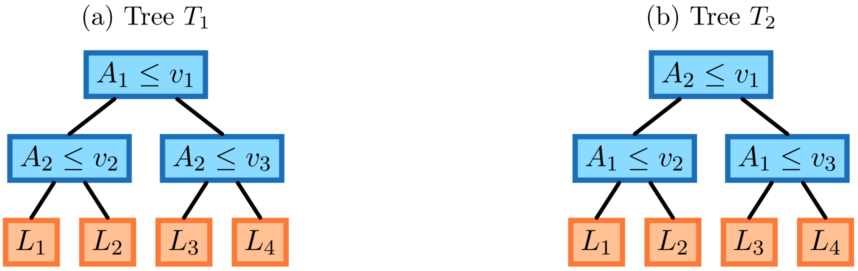


Figure 4.12: Sample trees T_1 and T_2 that motivate the selection of the weight parameter w.

To produce diverse ensemble members, we use the symbolic feature ranking with a random forest of iSOUP-Trees, as defined above. Note that for the task of online feature ranking, no leaf models are ever necessary, as there is no need for any predictions.

What remains is the choice of the weight factor w. When considering its possible values, we note that $w < 1$ gives higher scores to attributes which appear closer to the root, and, consequently, affect the larger parts of the input space. To settle on a particular value of w, we observe the following example. Consider a leaf in which two best attributes A_1 and A_2 have the exact same heuristic score. In the first case, we split on the first attribute and likewise in the second case, we split on the second attribute. Afterwards, in the first case we split both leaves on A_2, and vice versa (see Figures 4.12a and 4.12b, respectively).

In both cases, all example traversal paths include splits on A_1 and on A_2. This implies that A_1 and A_2 should have equal importances, as they affect the same sets of examples. Under this assumption it follows that

$$\begin{aligned} \mathrm{I}(A_1, T_1) &= \mathrm{I}(A_1, T_2) \\ w^d &= w^{d+1} + w^{d+1} \\ 1 &= 2w \\ 0.5 &= w, \end{aligned}$$

where d is the depth of the initial leaf that was split twice. Hence, we choose $w = 0.5$.

Chapter 5

Evaluation of Online Structured Output Prediction Methods

> If you torture the data long enough,
> it will confess to anything.
>
> — Darrell Huff

In this chapter, we define the evaluation approaches and measures that we use to evaluate the methods introduced in this thesis and the models they learn. We start with an overview of evaluation approaches on data streams, which are considerably different from those in the batch setting. We continue with a summary of measures of predictive performance that are used for the tasks of multi-target regression and multi-label classification, and briefly comment on the evaluation of predictive performance of hierarchical learners. In addition, we define how we measure the use of computational resources, in terms of memory usage and learning time. Furthermore, we discuss the differences between the batch and online evaluation procedures for the task of semi-supervised learning. Afterwards, we look at the intricate problem of evaluating feature importances in an online setting. Finally, we look at a commonly used procedure for estimation of statistical significance of the obtained evaluations of predictive performance and resource use.

5.1 Evaluation Approaches on Data Streams

In the batch learning setting, the most common approach for evaluating the predictive performance of a learning method is to split the available dataset into a training set and a testing set, after which a predictive model is learned on the training set and evaluated on the testing set. This means that predictions are made only on the testing set and the evaluation measures are calculated from these predictions. In this way, we obtain an unbiased estimate of the predictive performance, as we are evaluating the model only on examples which it has not learned from.

In the online setting, there is no clear distinction between training and testing phases and evaluation approaches are designed to be continuous to keep pace with the online learning procedure. Due to the real-time nature of online learning, we are interested in how the model performs throughout the learning procedure, at each time-point. Hence, we use each example to both test and train the model. To avoid introducing bias into the evaluation procedure, we always test each model on an example before we learn from it.

The most commonly used evaluation approaches in online learning are the *holdout* approach and *predictive sequential* (*prequential*) approach (Dawid, 1984).

Algorithm 5.1: Holdout evaluation for online learning.

```
Data: Data stream $\mathcal{D}$, updatable model M, $l$ window length
W ← [];                                           // W - window
P ← [];                            // P - vector of predictions
while $\mathcal{D}$ has more examples do
    e ← NextExample($\mathcal{D}$);
    P.append(M(e));
    W.append(e);
    if $|W| = l$ then
        calculate and record evaluation measure(s) using predictions P;
        foreach example e ∈ W do
            M.update(e);
        end
        W ← [];
        P ← [];
    end
end
```

Algorithm 5.2: Prequential evaluation for online learning.

```
Data: Data stream $\mathcal{D}$, updatable model M
P ← [];
while $\mathcal{D}$ has more examples do
    e ← NextExample($\mathcal{D}$);
    P.append(M(e));
    M.update(e);
end
calculate and record evaluation measures from P;
```

Holdout evaluation. In the holdout approach, each incoming example is first used for testing, i.e., we use the model to calculate a prediction. Afterward, we store it in a *window*, which represents a bag of examples. When the window reaches a certain predetermined number of examples (window length), we calculate and report the value of the evaluation measure on the current window. Afterwards, all of the examples are used, in order of arrival, to train the model. Finally, the window is emptied and the process is repeated, as long as there are available examples in the data stream. A summary of the procedure is presented in Algorithm 5.1.

The holdout approach is in a way similar to the evaluation approach used in the batch setting. The model that is used to make predictions is the same on the entire window, as it is never updated until the window gets full. The batch approach can be seen as a holdout scenario, where there is only ever one window, which is exactly the size of the test set.

Prequential evaluation. Prequential evaluation (Dawid, 1984) uses an interleaved test-then-train approach. An example is used for training immediately after it was used to record the prediction, as shown in Algorithm 5.2. It can be seen as an extreme case of holdout evaluation, one in which the window size is equal to 1.

In holdout evaluation, toward the end of the window, the model is getting “stale”, i.e., it is still the same model that has been learned only on the examples that appeared before the current window. On the other hand, in prequential evaluation, the model is always "fresh", as it has already learned from *all* the examples, which are available up to a certain point in time.

Discussion. Given the above, one might posit the question of why holdout evaluation is used at all, given that it produces a pessimistic evaluation of performance. However, we must consider that in online learning there is a need for real-time response. Even if the example arrives for training immediately after its prediction was calculated, the training procedure still takes time and might not have completed, when a new example arrives and a prediction must again be calculated. Prequential evaluation then produces the most optimistic evaluations of predictive performance, as it assumes that the model has had enough time to update before the arrival of the next example.

Holdout evaluation is more appropriate in a scenario where the predictive model is used as a tool for long-term planning. In this kind of scenario, predictions might be made for a future time interval, with the intent of assisting an operator. The operator might then consider several execution plans that impact the expected state of the system, which can not be exactly predicted, due to various reasons, e.g., stochasticity of the system or high probability of unplanned-for events. In this case, the pessimistic evaluation can be more appropriate than an optimistic one.

Consequently, either evaluation approach can be appropriate in a particular real-world application. In this thesis, we use the prequential approach to observe the performance of the methods in the most optimistic scenario.

Another, independent, facet of measuring predictive performance of online learners is the selection of evaluation measures. Often, we use *fading* evaluation measures, such as the fading mean absolute error introduced in Section 4.2.1.3. These types of measures place greater emphasis on more recent examples and less emphasis on older examples. These measures capture the current predictive performance of a model, which is particularly important when we expect concept drift. As we do not address concept drift in this thesis, we use regular, non-fading measures.

5.2 Measures of Predictive Performance for Structured Output Prediction Tasks

As we will show below, most measures of predictive performance for structured output prediction tasks are adapted from similar measures, used in primitive output prediction, e.g., in regression or classification. This is similar to the way approaches for structured output prediction are adapted from approaches for primitive output prediction. However, some of the measures are unique to the structured output setting, e.g., ranking-based measures in the case of multi-label classification.

In the following sections, we overview the measures of predictive performance, which are commonly used to evaluate methods and models for structured output prediction tasks. We start with measures for multi-target regression and continue with the measures for multi-label classification[1]. Finally, we briefly discuss the performance evaluation of hierarchical models.

In the following definitions, we use extensive notation that defines many symbols in the appropriate settings. Table 5.1 summarizes the notation in terms of the symbols that we use below.

[1] For each introduced evaluation measure, we mark it with $\downarrow$ if lower values are desired and with $\uparrow$ if higher values are desired.

Table 5.1: Notation used in the definitions of measures of predictive performance.

Symbol	Definition
X	Input space
N	Dimension of input space, number of input attributes
x	Particular input attribute values of an example, $x \in X$
Y	Output space
M	Dimension of output space, number of targets
y	Actual target value(s) of an example, $y \in Y$
$\hat{y}$	A prediction of target value(s), $\hat{y} \in Y$
$\hat{y}^j$	Prediction of j-th target, $j = 1, \dots, M$
$\mathcal{S}$	A data sample, i.e., a bag of examples
n	Number of examples in data sample $\mathcal{S}$
	Multi-target regression
y_i	Actual target value(s) of i-th example in $\mathcal{S}$, $i = 1, \dots, n$
$\hat{y}_i$	Predicted target value(s) of i-th example in $\mathcal{S}$, $i = 1, \dots, n$
$\overline{y}^j$	Average value of j-th target in $\mathcal{S}$, $j = 1, \dots, M$
	Multi-label classification
$\mathcal{L}$	The labelset, $Y = \mathcal{P}(\mathcal{L})$
M	Number of labels, $M = \lvert\mathcal{L}\rvert$; also dimension of output space, as above
λ_j	The j-th label, $j = 1, \dots, M$
τ	The classification threshold
y	True set of labels of an example
$\hat{z}$	Unthresholded prediction for an example, $\hat{z} \in \mathbb{R}^M$
$\hat{y}$	Thresholded prediction for an example, also considered as a set of labels, $\hat{y} \in Y$
$\hat{y}^\lambda$	Prediction for individual label λ of $\hat{y}$
$\lvert \cdot \rvert$	Size of contained (label)set

5.2.1 Performance evaluation for multi-target regression

In single-target regression many measures have been adapted from statistics (Witten et al., 2016). These range from mean absolute error (MAE), root mean squared error (RMSE)

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\hat{y} - y|, \;\; \mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\hat{y} - y)^2}$$

that operate on the same scale as the values, to relative measures which compare to the mean of the sample, such as the relative mean absolute error (RMAE) and the relative root mean squared error (RRMSE)

$$\mathrm{RMAE} = \frac{\sum_{i=1}^{n} |\hat{y} - y|}{\sum_{i=1}^{n} |\overline{y} - y|}, \;\; \mathrm{RRMSE} = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y} - y)^2}{\sum_{i=1}^{n} (\overline{y} - y)^2}}. \tag{5.1}$$

In both groups of measures, the perfect predictive model achieves an error of 0. However, in the first group of measures, the actual value of the error gives us very little information, if we do not know the sample. For example, a MAE with value 10 is very good on a sample with values $1000, 1100, 1200$, while it is extremely bad on a sample with values $1, 2, 3$.

Let us now consider the relative measures, in the particular case of the single-target mean predictor. The mean predictor predicts the mean of the entire sample for each example it encounters. However, following the definition of both the relative measures in Equation 5.1, we quickly see that both measures are equal to 1. Therefore, a value of a relative measure of lower than 1 implies that the evaluated model is better than the mean predictor, while values over 1 imply that the model is worse than the mean predictor. Note that because we generally calculate evaluation measures on examples which have not (yet) been used for learning, these observations might not be exactly true in all scenarios.

When we move into a multi-target scenario, we can always calculate any single-target measure on the predictions of the individual targets. However, this is especially cumbersome when there are many targets. Additionally, we wish to produce a single-value evaluation, as comparing multiple values is considerably more difficult.

A common approach to combining single-target evaluations into a multi-target evaluation is averaging. While this obfuscates some of the fine-grained detail we can achieve at looking at the performance evaluations on individual targets, it is a necessary step for pairwise comparison of methods and models. We must, however, be especially vigilant, as averaging values which are defined over different ranges is problematic, if we have no prior knowledge of the relative importances of the targets. When evaluating machine learning methods for data mining, we must be especially careful as we want the methods to be compared fairly.

Let us examine an example where we average the mean absolute errors of three targets, each of which are defined on a separate interval, defined by the target values. Suppose that we have two methods A and B, which we compare on targets y^1, y^2 and y^3. Method A is evaluated according to the targets as $(10.0, 10.0, 10.0)$, while method B is evaluated as $(20.0, 5.0, 5.0)$. However, given the previous discussion, we cannot tell which of the methods is better. The first target might take values of 15, 20, 25 or 5, 10, 15. Averaging all of the target evaluations only compounds this problem.

On the other hand, this kind of evaluation can also be appropriate. If we know that all three targets, e.g., measure a number of various sold items, then we know they measure a similar physical quantity and that they may be equally important. In this case, an average of mean absolute errors is more appropriate than an average of a relative measure.

In the scenario when comparing multiple competing machine learning methods, we often make comparisons over many datasets. In this case, we cannot take into account any knowledge of the dataset, in particular knowledge about the relative target importances. The goal is to produce evaluations of predictive performance that are as comparable among themselves as possible, i.e., we want them to be comparable across datasets. Therefore, in this scenario, we recommend the use of relative evaluation measures.

Hence, to evaluate the predictive performance of predictive models for multi-target regression, we define the *average relative mean absolute error* ($\overline{\mathrm{RMAE}}$) measure on an evaluation sample $\mathcal{S}$ as

$$\overline{\mathrm{RMAE}}(\mathcal{S}) = \frac{1}{M} \sum_{j=1}^{M} \frac{\sum_{i=1}^{n} \left| y_j^i - \hat{y}_i^j \right|}{\sum_{i=1}^{n} \left| y_i^j - \overline{y}^j(i) \right|},$$

where y_i^j is a true value of the target j for example i, $\hat{y}_i^j$ are the predictions of the evaluated model and $\overline{y}^j(i)$ is the value predicted by the j-th mean regressor for the i-th example. As we will be using the prequential evaluation approach, each prediction $\hat{y}_i$ will have been made using knowledge of only prior examples. We use the mean regressor in place of the actual mean of the sample, to approximate the same setting. To avoid the problem for the first example, which is not well defined, we take the average to be the prediction of the mean regressor based on all prior examples. Finally, the $\overline{\text{RMAE}}$ of a perfect regressor is 0, and lower values of the error are desired (↓).

5.2.2 Performance evaluation for multi-label classification

When evaluating the performance of a multi-label classifier, we use a set of measures employed in recent surveys and experimental comparisons of different multi-label algorithms in the batch setting (Madjarov, Kocev, et al., 2012; M.-L. Zhang & Zhou, 2014; Gibaja & Ventura, 2015). The measures are grouped into three categories: example-based measures (accuracy, F^1, Hamming score), label-based measures (macro-averaged precision, recall, F^1, as well as micro-averaged precision, recall, F^1) and ranking-based measures (average precision, ranking loss, logarithmic loss). This yields a total of 12 measures of predictive performance.

From the above, it is clear that in the MLC setting the performance can be investigated along a wide variety of measures. Example-based measures evaluate the quality of classification on a per-example basis, i.e., how good is the classification over different examples, while label-based measures evaluate the quality of the classification on a per-label basis, i.e., how good is the classification over different labels. Ranking-based measures evaluate the classification based on the ordering of the labels according to their presence, e.g., a classification is evaluated more positively if the present labels are ranked higher, often without regard to the thresholding procedure.

In particular, example-based and label-based measures are calculated based on the comparison of the predicted labels with the actual, ground truth labels. On one hand, example-based measures depend on the average difference of the actual and predicted sets of labels over the complete set of data examples from the evaluation set. On the other hand, label-based measures assess the performance for each label separately and then average the performance over all labels. Multi-label classification models often make predictions as numerical values for each of the labels. The label is then predicted as present if the numerical value exceeds a predefined threshold τ. This means that both example-based and label-based measures are directly dependent on the choice of the parameter τ. Ranking-based evaluation measures, however, compare the predicted ranking of the labels with the ground truth ranking and do not necessarily depend on the choice of the threshold parameter.

Furthermore, it becomes clear that any given method for multi-label classification is not able to optimize all of these measures at once, as they measure vastly different quantities. When comparing multiple multi-label classification methods, it is common that some methods optimize some measures better, while other methods optimize other measures better.

In the following definitions, n is the number of examples in the evaluation sample $\mathcal{S}$, $\mathcal{L}$ is the set of all labels and $M = |\mathcal{L}|$ is the number of labels. As noted earlier, $\hat{y}_i$ and y_i represent the predicted labels and actual labels of the i-th example, respectively. $\hat{y}$ and y refer to a prediction made by a method on some specific (non-indexed) example and its actual labelset, respectively, and are used to define loss functions. $\hat{z}_i^\lambda$ refers to the unthresholded predicted score for the i-th example in $\mathcal{S}$ and the label λ, while $\hat{y}_i^\lambda$ is already thresholded, i.e.,

$$\hat{y}_i^\lambda = \begin{cases} 1 & ; \text{if } \hat{z}_i^j \geq \tau \\ 0 & ; \text{otherwise} \end{cases} .$$

The example-based and label-based measures are calculated using $\hat{y}$, while the ranking-based measures are calculated using $\hat{z}$.

5.2.2.1 Example-based measures

Accuracy. The accuracy for an example with a predicted labelset $\hat{y}$ and a real labelset y is defined as the Jaccard similarity coefficient between them, i.e., $\frac{|\hat{y} \cap y|}{|\hat{y} \cup y|}$. The *accuracy* over a sample $\mathcal{S}$ of size n is the averaged accuracy over all examples:

$$\text{Accuracy}(\mathcal{S}) = \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{y}_i \cap y_i|}{|\hat{y}_i \cup y_i|}.$$

The higher the accuracy (↑) of a model the better is its predictive performance.

F^1 measure. The F^1 measure for MLC is the natural extension of F^1 measure used in regular classification, however, we can approach it from either the example-based or label-based perspective. The general formula for calculating F^1 is the usual harmonic mean of the precision and recall

$$\text{F}^1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

If we want to calculate F^1 from the example-based perspective, we define *example-based precision* (↑) and *example-based recall* (↑) as follows:

$$\text{Precision}_{\text{ex}}(\mathcal{S}) = \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{y}_i \cap y_i|}{|y_i|} \qquad \text{Recall}_{\text{ex}}(\mathcal{S}) = \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{y}_i \cap y_i|}{|\hat{y}_i|},$$

resulting in the final definition for *example-based F^1* (↑) measure

$${\text{F}^1}_{\text{ex}}(\mathcal{S}) = \frac{2 \cdot \text{Precision}_{\text{ex}} \cdot \text{Recall}_{\text{ex}}}{\text{Precision}_{\text{ex}} + \text{Recall}_{\text{ex}}}.$$

The definitions of the label-based F^1 measures are found below.

Hamming loss. The *Hamming loss* measures how many times an example-label pair is misclassified. Specifically, each label that is either predicted but not actual, or vice versa, carries a penalty to this score. The Hamming loss of a single example is the number of such misclassified labels divided by the number of all labels, i.e., $\frac{1}{M}|\hat{y} \triangle y|$ where $\hat{y} \triangle y = (\hat{y} \cup y) \setminus (\hat{y} \cap y)$ is the symmetric difference of the sets $\hat{y}$ and y. The Hamming loss of a sample is the averaged Hamming loss over all examples:

$$\text{HammingLoss}(\mathcal{S}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{M} |\hat{y}_i \triangle y_i|.$$

The Hamming loss of a perfect model, i.e., a model that makes exclusively correct predictions, is zero and the lower the Hamming loss, the better the predictive performance of a model. However, we generally report the Hamming loss as the *Hamming score* (↑), i.e., $\text{HammingScore}(\mathcal{S}) = 1 - \text{HammingLoss}(\mathcal{S})$.

5.2.2.2 Label-based measures

To define many of the label-based measures, we expand definitions of measures from single-target classification, i.e., the true positive (TP), false positive (FP), true negative (TN) and false negative (FN) rates are each defined on a per-label basis as follows:

$$\begin{aligned}
\text{TP}^{\lambda}(\mathcal{S}) &= |\{\hat{y}_i \mid \lambda \in y_i \wedge \lambda \in \hat{y}_i, 1 \leq i \leq n\}| \\
\text{FP}^{\lambda}(\mathcal{S}) &= |\{\hat{y}_i \mid \lambda \notin y_i \wedge \lambda \in \hat{y}_i, 1 \leq i \leq n\}| \\
\text{TN}^{\lambda}(\mathcal{S}) &= |\{\hat{y}_i \mid \lambda \notin y_i \wedge \lambda \notin \hat{y}_i, 1 \leq i \leq n\}| \\
\text{FN}^{\lambda}(\mathcal{S}) &= |\{\hat{y}_i \mid \lambda \in y_i \wedge \lambda \notin \hat{y}_i, 1 \leq i \leq n\}|,
\end{aligned}$$

where $\lambda \in \mathcal{L}$ is a label. This further allows us to extend the definitions of precision, recall and F^1 in a label-based manner. However, we have two choices how to combine the contributions of each label, *macro-* and *micro-averaging.* In macro-averaging, we compute each measure per label and then average the measures over all of the labels, while in micro-averaging, we first sum up TP, FP, TN and FN rates and use those to calculate the measures.

Macro-averaged measures. The *macro-averaged precision* (↑) and *recall* (↑) are defined as follows:

$$\begin{aligned}\mathrm{Precision}_{\mathrm{macro}}(\mathcal{S}) &= \frac{1}{M}\sum_{\lambda\in\mathcal{L}}\frac{\mathrm{TP}^{\lambda}}{\mathrm{TP}^{\lambda}+\mathrm{FP}^{\lambda}}\\ \mathrm{Recall}_{\mathrm{macro}}(\mathcal{S}) &= \frac{1}{M}\sum_{\lambda\in\mathcal{L}}\frac{\mathrm{TP}^{\lambda}}{\mathrm{TP}^{\lambda}+\mathrm{FN}^{\lambda}}\end{aligned}$$

To define the *macro-averaged F^1* measure (↑), we further extend the definition by substituting the precision and recall formulas with their forms in terms of TP^{λ}, FP^{λ}, TN^{λ} and FN^{λ}. This results in the following formula:

$$\mathrm{F^1}_{\mathrm{macro}}(\mathcal{S}) = \frac{1}{M}\sum_{\lambda\in\mathcal{L}}\frac{2\cdot\dfrac{\mathrm{TP}^{\lambda}}{\mathrm{TP}^{\lambda}+\mathrm{FP}^{\lambda}}\cdot\dfrac{\mathrm{TP}^{\lambda}}{\mathrm{TP}^{\lambda}+\mathrm{FN}^{\lambda}}}{\dfrac{\mathrm{TP}^{\lambda}}{\mathrm{TP}^{\lambda}+\mathrm{FP}^{\lambda}}+\dfrac{\mathrm{TP}^{\lambda}}{\mathrm{TP}^{\lambda}+\mathrm{FN}^{\lambda}}} = \frac{1}{M}\sum_{\lambda\in\mathcal{L}}\frac{2\,\mathrm{TP}^{\lambda}}{2\,\mathrm{TP}^{\lambda}+\mathrm{FP}^{\lambda}+\mathrm{FN}^{\lambda}}.$$

Micro-averaged measures. The following micro-averaged measures (↑) are all obtained using the procedure outlined above:

$$\begin{aligned}\mathrm{Precision}_{\mathrm{micro}}(\mathcal{S}) &= \frac{\sum_{\lambda\in\mathcal{L}}\mathrm{TP}^{\lambda}}{\sum_{\lambda\in\mathcal{L}}\mathrm{TP}^{\lambda}+\sum_{\lambda\in\mathcal{L}}\mathrm{FP}^{\lambda}},\\ \mathrm{Recall}_{\mathrm{micro}}(\mathcal{S}) &= \frac{\sum_{\lambda\in\mathcal{L}}\mathrm{TP}^{\lambda}}{\sum_{\lambda\in\mathcal{L}}\mathrm{TP}^{\lambda}+\sum_{\lambda\in\mathcal{L}}\mathrm{FN}^{\lambda}},\\ \mathrm{F^1}_{\mathrm{micro}}(\mathcal{S}) &= \frac{2\cdot\mathrm{Precision}_{\mathrm{micro}}\cdot\mathrm{Recall}_{\mathrm{micro}}}{\mathrm{Precision}_{\mathrm{micro}}+\mathrm{Recall}_{\mathrm{micro}}}.\end{aligned}$$

5.2.2.3 Ranking-based measures

Since thresholding has a significant impact on performance measures and the process of determining the optimal threshold is non-trivial, we also use measures that are independent of the chosen threshold. These measures include ranking loss, logarithmic loss and average precision.

Ranking loss. The *ranking loss* (RankLoss, ↓) measure is defined as

$$\mathrm{RankLoss}(\mathcal{S}) = \frac{1}{n}\sum_{i=1}^{n}\frac{|D_i|}{|y_i||y_i^{\mathsf{c}}|},$$

where $\mathcal{S}$ is an evaluation sample and $y_i^{\mathsf{c}} = \mathcal{L}\setminus y_i$ is the complement of y_i in $\mathcal{L}$, $D_i = \{(\lambda_1,\lambda_2) \mid \hat{z}_i^{\lambda_1} \le \hat{z}_i^{\lambda_2}, (\lambda_1,\lambda_2) \in y_i \times y_i^{\mathsf{c}}\}$. Note, that only the actual values y_i in the above definition are labelsets, while $\hat{z}^{\lambda}$ are predicted scores. Ranking loss essentially measures how well the labels are ordered by score, i.e., the loss is low when the labels that are not present have lower scores than the present labels. Consequently, lower values of ranking loss indicate better performance.

Logarithmic loss. Another ranking-based measure is the *logarithmic loss* (LogLoss, ↓) (Read et al., 2011). When calculating logarithmic loss, each labeling error is graded according to the confidence of the prediction, i.e., low confidence errors result in logarithmically smaller penalties than high confidence errors. Specifically, for an evaluation sample $\mathcal{S}$, it is calculated as

$$\mathrm{LogLoss}(\mathcal{S}) = \frac{1}{nM} \sum_{i=1}^{n} \sum_{\lambda \in \mathcal{L}} \min\Big(-\text{log-loss}(\hat{z}_i^{\lambda}, \mathbb{1}_{\lambda}(y)), \log n \Big), \tag{5.2}$$

where $\mathbb{1}_{\lambda}$ is the indicator function of label λ, i.e., $\mathbb{1}_{\lambda}(y) = 1$ if $\lambda \in y$ and $\mathbb{1}_{\lambda}(y) = 0$ if $\lambda \notin y$ and where $\text{log-loss}(\hat{z}, z)$ is defined as

$$\text{log-loss}(\hat{z}, z) = z \log \hat{z} + (1 - z) \log (1 - \hat{z})$$

Note that, when either of the logarithms encounters an argument with value 0, i.e., when $\hat{z} = 0$ or $\hat{z} = 1$, we take its value to be $-\infty$. To avoid over-penalizing an example, in Equation 5.2 the $-$log-loss, which can take a value of ∞, is thresholded to produce a maximum penalty of $\log n$. This thresholding also prevents a small poorly predicted labelset from greatly distorting the overall error. Therefore, LogLoss always falls on the $[0, \log n]$ interval for a sample $\mathcal{S}$, with 0 being the logarithmic loss of the perfect classifier. Hence, lower values are desired.

Notably, using the MLC via MTR approach, as discussed in Section 4.3, might yield predictions $\hat{z}^{\lambda}$ that do not lie on the $[0, 1]$ interval and fall out of the domain of the logarithm function. To that end, we replace all $\hat{z}^{\lambda}$ with $\min\{0, \max\{\hat{z}, 1\}\}$.

Average precision. Let us define $\mathrm{rank}(\hat{z}, \lambda)$ as the ranking of the label λ according to the prediction $\hat{z}$, i.e.,

$$\mathrm{rank}(\hat{z}, \lambda) = 1 + \Big| \Big\{ \lambda' \mid \hat{z}^{\lambda} < \hat{z}^{\lambda'}, \lambda' \in \mathcal{L} \Big\} \Big| .$$

If a given label λ has the highest predicted value in $\hat{z}_i$ of all the labels, it's rank will be 1, as no other label has a higher predicted value. The *average precision* (AvgPrecision, ↑) measure considers the average fraction of labels ranked above an actually present label $\lambda \in y_i$ in a given example. More specifically, for a sample $\mathcal{S}$, the average precision is defined as:

$$\mathrm{AvgPrecision}(\mathcal{S}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|y_i|} \sum_{\lambda \in y_i} \frac{|\mathcal{L}_i^{\lambda}|}{\mathrm{rank}(\hat{z}_i, \lambda)},$$

where $\mathcal{L}_i^{\lambda} = \{\lambda' \mid \mathrm{rank}(\hat{z}_i, \lambda') \leq \mathrm{rank}(\hat{z}_i, \lambda), \lambda' \in y_i\}$, i.e., the set of all labels ranked lower than λ in $\hat{z}_i$. The perfect average precision value is 1 and higher values are desired.

5.2.3 Performance evaluation for hierarchical prediction tasks

There are several approaches for evaluating the predictive performance of hierarchical prediction models. They differ in what they are trying to measure, and are different for different hierarchical tasks.

For example, in the HMTR task we may be interested in seeing how well a model can predict the value in the root node of the hierarchy, i.e., the totally aggregated value. On the other hand, in HMLC this kind of evaluation is not at all interesting, as the root is always present in an example due to the hierarchy constraint.

In this thesis, we wish to observe whether the addition of the hierarchy can improve the prediction in the leaves. We can think of the hierarchy as a tool to improve the predictive performance and we are evaluating whether it can improve the performance and what is its effectiveness in this regard.

To this end, we evaluate the predictive performance of the leaves. We use measures of predictive performance, which are appropriate for the corresponding non-hierarchical tasks, i.e., multi-target regression or multi-label classification.

5.3 Evaluation of Semi-Supervised Methods

In batch semi-supervised learning, the main goal is to utilize the cheap and abundant collected data, which is not labeled, to improve the predictive performance of a predictive model. Due to the unlabeled data examples, we can also not calculate any evaluation measures for them. However, when we evaluate methods for semi-supervised predictive modeling tasks, we often artificially unlabel examples. In this case, our main interest is the predictive performance of the model by using only the remaining labeled examples, even though, in this setting it would be possible to calculate the predictive performance on the unlabeled examples as well.

In an online semi-supervised learning setting, we view the unlabeled examples differently. As we have seen in Section 5.1, the prequential evaluation approach closely follows the natural streaming process with examples arriving unlabeled, after which the predictive model produces their predictions. Later on in the stream, the examples may arrive labeled and get used for the purpose of learning.

To compare machine learning methods for online semi-supervised predictive modeling, we use the same artificial unlabeling procedure as in the batch setting (Levatić, 2017). However, we record the predictions on the unlabeled examples, as well as using the predictions of both labeled and unlabeled examples to calculate the evaluation measures. Thus, we use measures appropriate for the corresponding supervised task.

To evaluate how well the various machine learning methods utilize the unlabeled data, we generate variants of the observed datasets, where each example retains its label according to a predetermined probability. For example, in one variant the probability of retaining the label might be 0.1, 0.2 or 0.5. This yields variants of the datasets where approximately 10%, 20% and 50% of all examples are labeled, respectively, while the rest are unlabeled. We then compare the predictive performances of (semi-supervised) methods on the different variants of the datasets to see how sensitive they are to the ratio of labeled and unlabeled examples.

5.4 Evaluation of Feature Importance Scores

Evaluating feature importance scores is particularly difficult, so we generally evaluate the rankings induced by the scores. In the batch setting, feature rankings can be evaluated by using forward feature addition and backward feature addition (Slavkov, 2012).

In forward feature addition, a predictive model is learned using only the top k attributes, for all $k = 1, \ldots, N$. The predictive performance of the different models is then plotted against the corresponding values of k. It is expected that the predictive performance of the induced models of a good feature ranking grows quickly at low k, then plateaus when irrelevant attributes are added at higher k.

Reverse feature addition works in a similar way, but in an opposite direction. At $k = 1$ we induce a model on just the lowest ranked attribute, and at each k we consider the bottom k attributes. Finally, when $k = N$ we induce a model on all attributes. Again, we plot the predictive performances of the models against k, however, in this scenario the predictive performances of a good feature ranking remain low until k rises and then grows rapidly with the best attributes are added.

Using this procedure for evaluating feature rankings in an online learning setting is less useful. A large part of the appeal of the above evaluation approach is that it allows for a quick qualitative analysis, by examining the shapes of the curves. In an online learning setting, this would be lost if we are to calculate the forward feature addition and backward feature removal curves for each example. Additionally, for each example we would have to retrain N predictive models, which is unfeasible from an experimental design standpoint, as the experiments would take inordinately long.

With this in mind, we can compare two rankings in an online scenario. Measures, such as the Jaccard similarity (Jaccard, 1912) and Canberra distance (Lance & Williams, 1967) can be quickly calculated from two feature rankings[2].

Let R_1 and R_2 be two feature rankings, and let R_1^k and R_2^k be their top k attributes, respectively. The *Jaccard similarity* of R_1 and R_2 is then a plot of

$$\mathrm{J}(n) = \frac{|R_1^n \cap R_2^n|}{|R_1^n \cup R_2^n|},$$

against n from 1 to N. The Jaccard similarity measures the proportion of the attributes in both the top n of R_1 and top n of R_2 to all of top n attributes in either ranking. However, it is prone to large fluctuations, especially at lower k values. For example, two rankings that share the top ranked attribute but differ in the attribute ranked second, will have a Jaccard similarity of 1 at $k = 1$ and $\frac{1}{3}$ at $k = 2$. This happens whenever two attributes are added, one to each respective ranking, which were not shared in the top $k - 1$ and are still not shared in the top k attributes. Then, the denominator increases by 2, while the numerator remains the same. Conversely, when both adding attributes that were already included in the other ranking, i.e., adding one of the top $k - 1$ ranked attributes of R_1 to R_2 and vice versa, the numerator increases by 2 while the denominator remains the same. Notably, the Jaccard similarity also places equal weight on attributes with high and low ranks.

The *Canberra distance* gives greater weight to top ranked attributes. Additionally, it aggregates over all attributes, and is reported as a single value. The Canberra distance between ranking R_1 and R_2 is

$$\mathrm{C}(R_1, R_2) = \sum_{i=1}^{N} \frac{|R_1(i) - R_2(i)|}{R_1(i) + R_2(i)},$$

where $R_j(i)$ is the rank of the i-th attribute of feature ranking R_j for $j \in \{1, 2\}$. We can easily and quickly calculate the Canberra distance between two rankings for each example in a data stream.

Given that we will be inducing rankings from feature importance scores, we must also consider what happens when the rankings do not discriminate between some attributes. A natural response is to assign the same ranking to the attributes that are not discriminated.

[2] As we are comparing two feature rankings and not comparing predictions against a known ground truth, we interpret the ↑ and ↓ symbols differently. When we use ↑, we signify that higher values of the measure indicate that the two rankings are similar. Conversely, when we use ↓, lower values of the measure denote similar rankings.

For example, when a ranking ranks one attribute over the rest, then there are two attributes that are indistinguishable, and finally there is a fourth attribute, we might assign them ranks 1, 2, 2 and 4, respectively. However, this would particularly affect the calculation of the Jaccard similarity as we expect that exactly k attributes are in the top k attributes. To this end, when calculating the Jaccard similarity, we assign unique ranks within a set of indistinguishable attribute by ordering them by the order of their appearance in the dataset. While this introduces some randomness into the calculation of the Jaccard similarity, as we have no control over the order of the attributes, most commonly the attributes which are indistinguishable are in the bottom portion of the ranking on which we place less emphasis.

In the case of the Canberra distance, we instead assign ranks to the indistinguishable attributes equal to their average rank, as assigning them the lowest rank results in too optimistic estimates.

5.5 Efficiency Evaluation

In this thesis, we evaluate the computational efficiency of a method on two dimensions. The first dimension is how fast the method can learn and make predictions on the provided dataset, i.e., a method's *processing time*. The second dimension is how much memory a method consumes to do so.

The most common way of evaluating the speed of learning is to observe how much time has elapsed since the method started learning to the current example. In the extreme, when we look how much time has passed to the last example in a dataset, we look at the total time of learning. Learning time is usually measured in seconds.

However, we can also look at learning speed in a different way. In Chapter 8, we are interested in how much time a specific learning approach requires to process an example. This is particularly interesting in a streaming setting, where the models generally become more complex throughout the learning process and with that increasing the processing time of a single example. In this case, we also measure the processing time for an example in seconds.

Finally, we observe the memory consumption of the different methods. Here, we use the tools available in the MOA framework (Bifet, Holmes, Kirkby, & Pfahringer, 2010) to directly measure the amount of memory in megabytes that the model uses. Another common approach, used especially for tree-based methods (Kocev et al., 2013), is to count the number of leaves in the model. However, given that we have introduced methods that use different leaf models, this approach is not appropriate in our setting.

5.6 Tests of Statistical Significance

To assess whether the differences in evaluation between observed methods are statistically significant, we use a statistical testing procedure recommended by Janez Demšar (2006). In particular, we use the Friedman test (M. Friedman, 1940) with post-hoc Nemenyi analysis (Nemenyi, 1963).

The Friedman test is a non-parametric statistical test that detects the significance of the differences between competitors based on multiple test attempts. In our case, the competitors are the competing machine learning methods, while test attempts correspond to their evaluations on different datasets. For each dataset, the competing methods are ranked in order from the best to the worst evaluation. In the case of a tie, all tying competitors are assigned the appropriate average ranks.

If we denote the rank of the j-th competitor (out of a total of k competitors) on the i-th dataset (out of a total N datasets) as r_i^j, the Friedman test compares the competitors' average ranks $R_j = \frac{1}{N}\sum_{i=1}^{k} r_i^j$. The null-hypothesis is that the competitors are equivalent and, consequently, so are their ranks R_j. The Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right]$$

is then distributed according to $\chi^2(k-1)$, i.e., according to the χ^2 distribution with $k-1$ degrees of freedom. However, Iman and Davenport (1980) have shown that the Friedman statistic is overly conservative and suggested a corrected statistic

$$\mathrm{F}_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2},$$

which is distributed according to $\mathrm{F}(k-1,(k-1)(N-1))$, i.e., according to Fisher–Snedecor distribution, which is used for the F-test, with parameters $k-1$ and $(k-1)(N-1)$.

When we can reject the null-hypothesis, i.e., when F_F is greater than the appropriate critical value (we use a significance level of $\alpha = 0.05$), we proceed with post-hoc tests. We use the Nemenyi test (Nemenyi, 1963), which states that two competitors are statistically significantly different if their corresponding average ranks differ by at least the *critical distance*

$$\mathrm{CD} = q_\alpha\sqrt{\frac{k(k+1)}{6N}},$$

where the critical values q_α are based on the studentized range statistic divided by $\sqrt{2}$. Again, we use a significance level of $\alpha = 0.05$.

This allows us to plot the average ranks of the competitors on a line from 1 to k, and any pair of competitors that is apart by further than the critical distance, defined as above, are statistically significantly different from each other for the evaluation measure we used.

Chapter 6

Experimental Design

A good plan isn't one where someone wins, it's where nobody thinks they've lost.

— Terry Pratchett

To experimentally evaluate the methods for online structured output prediction introduced in Chapter 4, we designed several experiments and for each experiment defined the corresponding experimental setup. We start with an overview of experimental design for the evaluation of the iSOUP-Tree family of methods for the online multi-target regression task. We continue with the design of the experiments for the online multi-label classification task, specifically, to evaluate how the MLC via MTR methodology performs in concert with the introduced methods. In the later sections of the chapter, we propose several preliminary experiments, in which we attempt to utilize the iSOUP-Tree-based methods for other data mining tasks described in Sections 4.4, 4.5 and 4.6. These experiments are not as comprehensive and rigorous as the experiments for multi-target regression and multi-label classification as is reflected in their experimental designs. In particular, we present experiments designed to address online hierarchical predictive modeling tasks, online semi-supervised multi-target regression and, finally, online feature ranking for structured output prediction.

Each section first presents the experimental questions we wish to address with the particular experiment and a consideration of which methods we will experimentally evaluate. Later, we continue with a description of the evaluation methodology, specifically defining what kind of an evaluation approach is used, as well as which evaluation measures are observed. Finally, we introduce the datasets that are used for each experiment.

6.1 Experimental Evaluation of Online Multi-Target Regression Methods

In this set of experiments we observe how well the introduced iSOUP-Tree family of methods addresses the online multi-target regression task. Particularly, we are interested in how methods that utilize the global approach, which address the structured output prediction task using a single model, compare to the local approach, which uses a problem transformation methodology to learn multiple models, one for each target, and uses them in concert to address multi-target regression.

6.1.1 Experimental questions

The first experimental question we explore is how the introduced methods for online multi-target regression compare among themselves in terms of predictive performance. We are particularly interested in the difference in predictive performance of the local FIMT-DD method and the global iSOUP-Tree method.

We are also interested in making specific comparisons between some of the introduced methods. In particular, we are interested in how the option tree compares to both a single tree and to ensembles of trees, as option trees show aspects of both models.

In a single-target regression study, Ikonomovska et al. (2015) have shown no particular differences in predictive performance between a basic model tree method and the bagging method (therein referred to as FIMT-DD and OBag, respectively). Additionally, the random forest methodology produced worse results than a single tree, while the option tree variant (ORTO) of FIMT-DD outperformed all of the other methods. Here, we investigate whether similar conclusions can be drawn for the multi-target regression task. To that end, we study the differences in predictive performance between the following methods the iSOUP-Tree, the iSOUP-OptionTree, the bagging of iSOUP-Trees and the random forest of iSOUP-Trees methods.

Another key aspect of the streaming setting are the potential constraints on the available resources. Therefore, we are interested in the differences in consumption of resources between the different methods. While the resource consumption of the bagging method in comparison to a single tree is extrapolated trivially, other comparisons are more meaningful. In particular, we are interested in the trade-off between resource consumption and predictive performance which occurs in larger, more complex models, i.e., in option trees, bagging and random forests (as compared to the basic single tree approach). Knowing about this trade-off is especially important when we select a method for a given real-world application.

As described above, we compare the local FIMT-DD method (**Local**), the **iSOUP-Tree** method, the iSOUP-OptionTree method (**iSOUP-OT**), the bagging of iSOUP-Trees (**iSOUP-Bag**) and random forest of iSOUP-Trees (**iSOUP-RF**) methods. All of these methods use the default parameter values provided in Table 4.1.

6.1.2 Experimental setup and evaluation methodology

We use the prequential approach to evaluate the online multi-target regression experiments. Each example is given to each of the methods to produce a prediction, which is then recorded to calculate the evaluation measures. Afterwards, the example is given to the method as a learning example.

We use the average relative mean error ($\overline{\text{RMAE}}$) defined in Section 5.2.1 to compare the performances of the observed methods. To determine the method with the highest predictive performance, we calculate the evaluation measure over the entire dataset, and then compare the values. To determine the statistical significance of the obtained results we use the Friedman test with Nemenyi post-hoc analysis, described in Section 5.6.

When comparing the different methods, we are also interested in the progression of the evaluation measure for the whole data stream. Since reporting evaluation measures for data streams can be volatile if reported on an instance by instance basis, due to, e.g., the sampling of different parts of the input space, we calculate the evaluation measures on windows of 1000 examples. This allows us to plot the progression of the predictive performance with respect to the number of examples that have been processed.

We evaluate the consumption of computational resources of the observed methods by looking at the amount of time it takes a certain method to process, i.e., make predictions on

Table 6.1: Datasets used in the online multi-target regression experiments. M – number of targets.

Dataset	No. of examples	Attributes	M
Bicycles	17379	12 numeric	3
EUNITE03	8064	29 numeric	5
Forestry Kras	60607	160 numeric	11
Forestry Slivnica	6218	150 numeric	2
RF1	9005	64 numeric	8
RF2	7679	575 numeric	8
SCM1d	9803	280 numeric	16
SCM20d	8966	61 numeric	16

and learn from, the entire dataset as well as how much memory it used doing so. As above, we are interested in the progression of the consumption of computational resources as more and more examples become available. To this end, we measure the time required and the current memory consumption at each 1000 examples. Again, we plot these measurements not only to observe the absolute measurements, but also the trends which govern the resource consumption. In particular, we will observe the rate of growth of time and memory use in different methods.

6.1.3 Datasets

For the online multi-target regression experiments, we have selected 8 datasets, based on their size, looking for diversity in the number of input and target attributes. We consider the datasets under the assumption of no concept drift, given that these datasets are usually considered as batch benchmark datasets. A summary of the datasets and their properties is shown in Table 6.1.

The *Bicycles* dataset is concerned with the prediction of demand for rental bicycles on an hour-by-hour basis (Fanaee-T & Gama, 2013). The 3 targets represent the number of casual (non-registered) users, the number of registered users and the total number of users for a given hour, respectively.

The *EUNITE03*[1] dataset was used for the competition at the 3rd European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems in 2003. The data describes a complex process of continuous manufacturing of glass products, i.e., the input attributes describe various influences (which can or can not be changed by an operator), while the 5 targets describe the glass quality.

The data in the *Forestry Kras* dataset was derived from multi-spectral multi-temporal Landsat satellite images and 3D LiDAR recordings of a part of the Kras region in Slovenia (Stojanova, Panov, Gjorgjioski, Kobler, & Džeroski, 2010). Each example corresponds to a spatial unit, i.e., an area of 25 by 25 meters. For each example, the input attributes and targets were derived from the LandSat and LiDAR recordings of the spatial unit. For specifics on the data preparation procedure, see Stojanova et al. (2010). The task is to predict 11 targets, which correspond to properties of the vegetation in the observed spatial unit.

The *Forestry Slivnica* dataset was, as in the previous case, constructed from multi-spectral multi-temporal Landsat satellite images and 3D LiDAR recordings of a part of

[1] URL: http://www.eunite.org/eunite/news/Summary%20Competition.pdf (accessed 2018/01/22)

the Slivnica region in Slovenia (Stojanova, 2009). In this dataset, the task is to predict only 2 target variables: vegetation height and canopy cover.

The river flow datasets, *RF1* and *RF2*, concern the prediction of river network flows for 48 hours at 8 locations on the Mississippi River network (Spyromitros-Xioufis, Groves, Tsoumakas, & Vlahavas, 2012). Each data example comprises observations for each of the 8 locations at a given time point, as well as time-lagged observations from 6, 12, 18, 24, 36, 48 and 60 hours in the past. In RF1, each location contributes 8 input attributes, for a total of 64 input attributes and 8 target variables. The RF2 dataset extends RF1 with the precipitation forecast information for each of the 8 locations and 19 other meteorological sites. Specifically, the precipitation forecast for 6 hour windows up to 48 hours in the future is added, which nets a total of 280 input attributes.

The *SCM1d* and *SCM20d* are datasets derived form the Trading Agent Competition in Supply Chain Management (TAC SCM) conducted in July 2010. The preparation (preprocessing) of the datasets is described by Spyromitros-Xioufis et al. (2012). The data examples correspond to daily updates in a tournament – there are 220 days in each game and 18 games per tournament. The 16 targets are the predictions of the next day and the 20 day mean price for each of the 16 products in the simulation, for the SCM1d and SCM20d datasets, respectively.

The Bicycles dataset is available at the UCI Machine Learning Repository[2] and the RF1, RF2, SCM1d and SCM20d datasets are available at the Mulan multi-target regression dataset repository[3]. The examples with missing values (on some input attributes) in the RF1 and RF2 datasets were removed, so the resulting datasets were somewhat smaller than reported in the repository.

6.2 Experimental Evaluation of Online Multi-Label Classification via Online Multi-Target Regression

The main purpose of the multi-label classification via multi-target regression experiments in an online learning scenario is to see how the problem transformation methodology performs in comparison to dedicated state-of-the-art methods (Read et al., 2012) for online multi-label classification. We also explore how well different iSOUP-Tree based methods perform in the online MLC via online MTR methodology for a variety of multi-label classification evaluation measures.

6.2.1 Experimental questions

For the evaluation of the online MLC via online MTR methodology, we address several lines of inquiry. For the following comparisons, we select a subset of the introduced methods for online multi-target regression. We select model and regression iSOUP-Trees as representatives of single-tree methods, and bagging of regression and model iSOUP-Trees as representatives of ensemble methods. Notably, the local FIMT-DD-based method can not be applied to this task, as all of the datasets consist of exclusively nominal attributes from which the local FIMT-DD method is unable to learn.

First, we investigate whether the use of model trees with the adaptive models in the leaves improves predictive performance over regression trees, due to the concerns raised in Section 4.3. Specifically, we address the concern that using model trees can lead to predictions that fall outside of the $[0, 1]$ interval. This could impact the predictive performance.

[2] URL: https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset (accessed 2018/01/22)

[3] URL: http://mulan.sourceforge.net/datasets-mtr.html, (accessed 2018/01/22)

Second, we evaluate the performance of the introduced single-tree multi-target regression methods, in the context of the MLC via MTR methodology in comparison to the Hoeffding tree with pruned sets (HT_PS) (Read et al., 2012). The latter is a direct single-tree competitor, which does not utilize the MLC via MTR methodology. This allows us to investigate how viable is the MLC via MTR methodology for online multi-label classification.

Furthermore, we compare all of the introduced methods, iSOUP regression trees, iSOUP model trees, as well as ensemble-based approaches, bagging of iSOUP trees and random forests of iSOUP trees, to determine how the methods rank both in terms of predictive performance with regard to a selection of multi-label classification evaluation methods.

In addition to the predictive performance, we also observe the methods' efficiency in terms of consumption of computational resources to determine what, if any, trade-offs between predictive performance and consumption of computational resources can be made when using the different methods.

We compare two variants of iSOUP-Tree method, one that learns regression trees (**RT**) and one that learns model trees (**MT**), bagging of both regression iSOUP-Trees ($\mathbf{E_B RT}$) and model iSOUP-Trees ($\mathbf{E_B MT}$), as well as a single Hoeffding tree with pruned sets ($\mathbf{HT_{PS}}$) and ADWIN bagging of Hoeffding trees with pruned sets ($\mathbf{E_A HT_{PS}}$). The first four methods use the default parameter values provided in Table 4.1, while the last two use parameter values suggested by Read et al. (2012).

6.2.2 Datasets

In our experiments, we use a selection of multi-label classification datasets, summarized in Table 6.2. Below, we briefly describe each dataset.

The *20 newsgroups* is a dataset constructed by Read (2010), from the classic 20 newsgroups data (Lang, 2008). The original dataset is a compilation of around 20,000 posts to 20 Usenet newsgroups ranging from rec.sports to politics.guns, which represent part of a newsgroup hierarchy (where, for example, rec and politics are internal nodes). The dataset contains about 1000 posts per newsgroup, while the groups themselves serve as labels.

The *Enron* dataset (Read, 2010) is a collection of labeled emails, labeled with a hierarchical set of categories, where top level categories were “coarse genre”, “included/forwarded information”, “primary topics” and “emotional tone”. For the purpose of the multi-label classification, we take as labels all of the leaves of the hierarchy.

The *IMDB* dataset was collected by Read (2010) and is constructed from text summaries of movie plots from the Internet Movie Database. Each example represents one movie and is labeled with the relevant genres.

The *Ohsumed* dataset (Hersh, Buckley, Leone, & Hickam, 1994) is a subset of the MEDLINE database[4]. Individual examples correspond to peer-reviewed medical articles and labeled with the appearing disease categories.

The *Slashdot* dataset was collected from the http://slashdot.org news portal and its attributes are words that appear in the article blurbs. The labels are news categories, e.g., Linux, technology, science, etc.

The *TMC* dataset (Srivastava & Zane-Ulman, 2005) was used in the SIAM 2007 Text Mining Competition and consists of human-generated aviation safety reports, which describe potential problems that occurred during flights. The examples are labeled with various problems that are described in the reports. In particular, we are using the reduced

[4] URL: https://www.nlm.nih.gov/bsd/pmresources.html (accessed 2018/01/22)

Table 6.2: Datasets used in the multi-label classification experiments. M – number of labels, ϕ_{LC} – average number of labels per instance.)

Dataset	**No. of examples**	**Attributes**	**M**	ϕ_{LC}
20NG	19300	1001 binary	20	1.1
Enron	1702	1001 binary	53	3.4
IMDB	120919	1001 binary	28	2.0
Ohsumed	13929	1002 binary	23	1.7
Slashdot	3782	1079 binary	22	1.2
TMC	28596	500 binary	22	2.2

version of the dataset by Tsoumakas and Vlahavas (2007), which limits the dataset to only the 500 most frequent attributes.

With the exception of the TMC dataset, all datasets are available at the MEKA project page[5]. The TMC dataset is available at the Mulan data repository[6].

6.2.3 Experimental setup

The experimental setup we use is designed to be a streaming setting analog to the commonly used batch multi-label classification experimental setup, used by, e.g., Madjarov, Kocev, et al. (2012) and Read et al. (2009), and is very similar to the setup used by Read et al. (2012).

As we have discussed in Section 5.2.2, in the multi-label classification task it is impossible to optimize all of the evaluation measures at once. Because of this, we use a set of different evaluation measures for these experiments. We observe the *example-based* measures accuracy, F^1 and Hamming score measures, the *label-based* measures macro- and micro-averaged precision, recall and F^1 measures, and, finally, we observe the *ranking-based* measures ranking loss, logarithmic loss and average precision.

For the online multi-label classification, we also use the prequential evaluation approach as for online multi-target regression. Again, for each example, first a prediction is made and collected, and afterwards, the same example is used to update the model. Once predictions for all the examples are collected, they are thresholded to calculate the label-based and example-based measures on the entire dataset, while the ranking measures are calculated using the unthresholded predictions. In the experiments, we use a classification threshold of $\tau = 0.5$. The recorded measurements are therefore calculated using the obtained predictions over the entire dataset.

Additionally, we measured the total time and memory used to learn and make predictions. This allows us to observe under which resource constraints the methods could possibly operate.

To assess whether the overall differences in the predictive performance and resource consumption across all compared methods are statistically significant for a given evaluation measure, we again employ the corrected Friedman test with the post-hoc Nemenyi test.

[5] URL: https://sourceforge.net/projects/meka/files/Datasets/ (accessed 2018/01/22)

[6] URL: http://mulan.sourceforge.net/datasets-mlc.html (accessed 2018/01/22)

6.3 Experimental Evaluation of Online Hierarchical Prediction with iSOUP-Trees

In the experiments for hierarchical tasks, we consider both the task of online multi-target regression and online multi-label classification. More specifically, we are exploring if and how the hierarchically adjusted splitting heuristic based on the work of Vens et al. (2008) affects the predictive performance of the iSOUP-Tree method.

Experimental scenarios. In hierarchical prediction tasks, we are often most interested in the predictions of the leaf labels/targets. In these experiments, we examine how the hierarchy and the adapted methods affect the predictive performance in the leaf labels/targets. To this end we define three scenarios. The first scenario serves as a control group. We remove all of the non-leaf labels/targets in the observed datasets to only include the leaf labels/targets. In essence, this yields an online multi-label classification or online multi-target regression task that we can address with a regular iSOUP-Tree. We name this scenario as the "**leaves-only**" scenario. In the second scenario, we use a **bottom-weighted** hierarchical iSOUP-Tree, as described in Section 4.4, by selecting a root node weight of $w_0 = 2$. This method places greater emphasis on the homogeneity of the leaf targets/labels when selecting splits. In the third scenario, we use a **top-weighted** hierarchical iSOUP-Tree, which places emphasis on the targets/labels that are closer to the root node of the hierarchy, by selecting a root node weight of $w_0 = 0.5$.

Experimental setup. To obtain the predictions of all the models we use the prequential evaluation approach. As we are most interested in the predictive performance in the leaves, we calculate the evaluation measures only on the leaf labels/targets.

For hierarchical multi-target regression, we look at the $\overline{\text{RMAE}}$ evaluation measure. For hierarchical multi-label classification, we look at the same evaluation measures as in the multi-label classification experiments, i.e., the *example-based* accuracy, F^1 and Hamming score measures, the *label-based* macro- and micro-averaged precision, recall and F^1 measures, and, finally, we observe the *ranking-based* ranking loss, logarithmic loss and average precision. Note that we plot the multi-label classification evaluations differently than $\overline{\text{RMAE}}$. In calculating $\overline{\text{RMAE}}$, we are comparing the predictions of the observed methods to an online average regressor, thus, we plot the progression of the $\overline{\text{RMAE}}$ up to the current example. On the other hand, the multi-label classification measures do not compare directly to a baseline classifier. Hence, we calculate the measures on the predictions on the last 1000 examples. This means that plots that show MLC measures are prone to considerable fluctuations, while the plots of $\overline{\text{RMAE}}$ are smoother.

Datasets. In the hierarchical multi-target setting, we use two datasets for our experimental evaluation, as shown in Table 6.3a. The first is a modified *Bicycles* dataset, in which we arrange the targets into a very simple hierarchy, where the registered users and unregistered users targets are children of the total users target. The second dataset is the *Mars Express* dataset, where the task is to predict the power consumption on 33 sites in the Mars Express satellite (for more information see Chapter 8). The targets have been arranged into a hierarchy according to their physical location within the satellite (Mileski, 2017). While the hierarchies of the hierarchical multi-label classification datasets are too large to be presented here, the hierarchies of the Bicycles and the Mars Express dataset are shown in Figure 6.1.

There are only several hierarchical multi-label classification datasets that are of appropriate size for online analysis (see Table 6.3b). The *ImageCLEF2007A* and *Image-*

Table 6.3: Datasets used in the (a) hierarchical multi-label classification and the (b) hierarchical multi-target regression experiments. $\mathbf{M_L}$ is the number of leaf labels/targets, while $\mathbf{M_H}$ is the total number of labels/targets in the hierarchy.

(a)

Dataset	**No. of examples**	**Attributes**	$\mathbf{M_L}$	$\mathbf{M_H}$
Bicycles	17379	12 numeric	2	3
Mars Express	100000	42 numeric	33	48

(b)

Dataset	**No. of examples**	**Attributes**	$\mathbf{M_L}$	$\mathbf{M_H}$
ImageCLEF07A	11006	80 numeric	63	97
ImageCLEF07D	11006	80 numeric	26	47

CLEF2007D datasets were prepared by Dimitrovski, Kocev, Loskovska, and Džeroski (2011) from images provided of the ImageCLEF 2007 Medical Annotation Task, which was part of the Cross Language Evaluation Forum in 2007. Each of the 11000 examples represents one medical image annotated with an IRMA (image retrieval in medical applications, Lehmann, Schubert, Keysers, Kohnen, and Wein (2003)) code. IRMA codes combine four dimensions of annotation: *technical*, describing the imaging modality, *directional*, describing the body orientation, *anatomical*, describing the anatomical area examined and *biological*, describing the biological system examined. Each axis is hierarchically arranged from broad descriptions at the top, to more detailed descriptions at the bottom of the hierarchy. An image might then be annotated as an x-ray – fluoroscopy – analog, coronal – anteroposterior – supine, abdomen – middle abdomen – peri navel region, gastrointestinal systems – stomach image.

The attributes of the ImageCLEF datasets describe edge histograms calculated from the images, while the IRMA code of each image serves as the hierarchical target. The *A* and *D* variants of the dataset differ in the level of completeness of the IRMA hierarchy. In the *D* dataset the hierarchy contains only 47 nodes, of which 26 are leaves, while in the *A* dataset there are 97 nodes in the hierarchy, 63 of which are leaf nodes. Notably, the examples in these datasets are ordered by their target values, thus we shuffled examples prior to using the datasets.

6.4 Experimental Evaluation of Online Semi-Supervised Multi-Target Regression with iSOUP-PCTs

In the semi-supervised experiments, we look at how the predictive clustering paradigm (Blockeel, 1998) can be applied in an online learning scenario in a similar way as it was applied in the batch scenario by Levatić et al. (2017b). Specifically, we address the task of online semi-supervised multi-target regression, though the method could also be applied to other online tasks.

Experimental scenario and datasets. We compare the predictive performance of three methods, a regular **iSOUP-Tree** and two variants of iSOUP-PCT, to a baseline method Oracle-iSOUP-Tree. The first variant of **iSOUP-PCT** is used as a supervised

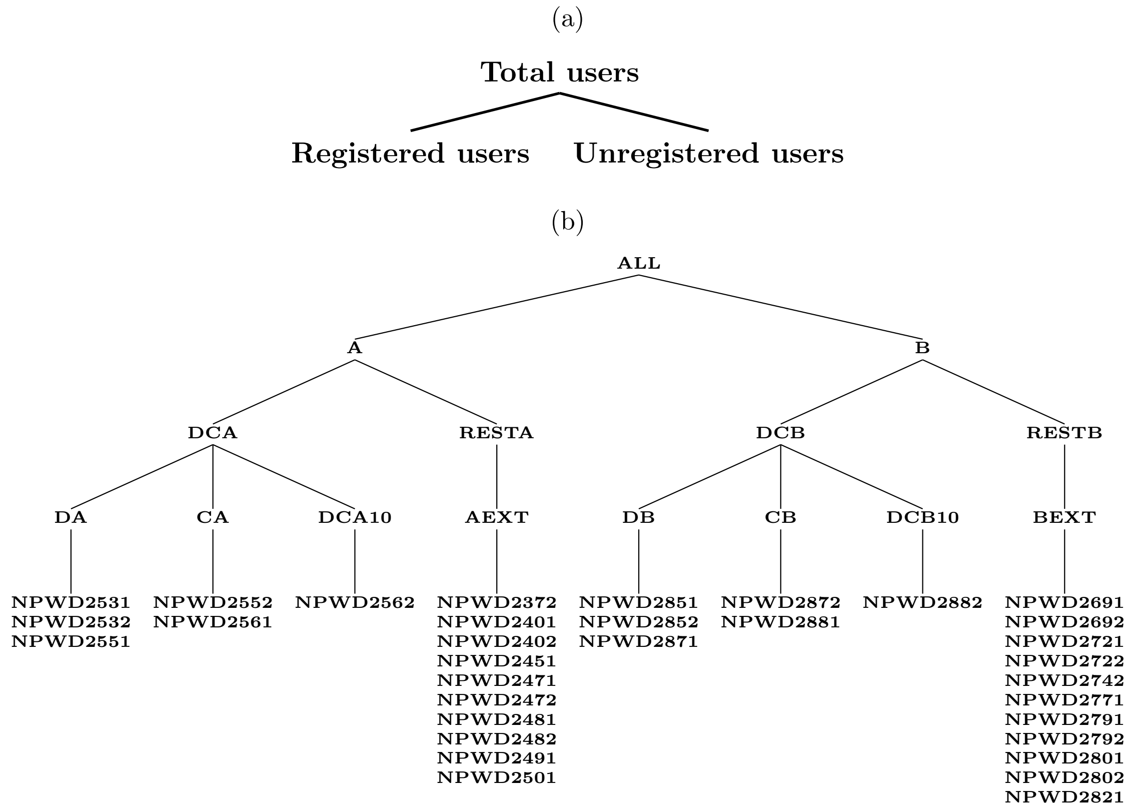


Figure 6.1: The hierarchy of the (a) Bicycles and the (b) Mars Express dataset.

learner, i.e., it learns only from labeled examples. The second variant, which we denote by **SSL-iSOUP-PCT**, is a semi-supervised learner that learns from all examples, both labeled and unlabeled. The **Oracle-iSOUP-Tree** method is an "oracle" method, which means that it learns from all examples as though they were labeled, i.e., it also has access to the target values even for the unlabeled examples. The oracle trees show a practical limit of how much can be learned in a semi-supervised scenario, if the semi-supervised method is able to utilize the unlabeled examples to the same extent as the labeled ones. Notably, for fairness of comparison all the methods learn regression trees and not model trees, as we discussed in Section 4.5.

In addition to the semi-supervised experiments, here we also compare iSOUP-Tree and iSOUP-PCT methods. In the batch setting, PCTs have equal or better performance as regular tree models. We wish to see if this occurs in the online setting as well. If the PCTs perform better, it does come at an increased cost of resource consumption, as we have discussed in Section 4.5.

Additionally, we are also interested in how the predictive performance depends on the ratio of labeled examples. This shows us whether at some ratios it is even advisable to use semi-supervised methods. We observe three labeling ratios in our experiments, generated by the following probabilities of labeling, 0.1, 0.2 and 0.5. This yields three variants of each of the multi-target regression datasets shown in Table 6.1. Specifically, for a given *labeling ratio* κ, we unlabel each example, except for the first two, with probability $1 - \kappa$. The first two examples are always labeled to allow the predictive models to properly initialize. To account for the randomness of the unlabeling procedure, we repeat each experiment for a given dataset and a given labeling ratio 10 times.

Experimental setup. We compare the methods with less rigor than in the previous multi-target regression and multi-label classification experiments. Same as in the previous experiments, here we utilize the prequential evaluation approach. Unlike the batch setting, where we are predominantly interested in the performance on labeled examples, we treat both the (artificially) unlabeled and the labeled examples equally. Therefore, we calculate the $\overline{\text{RMAE}}$ evaluation measure for each example on each of the repeated dataset, and record their average value once every 1000 examples, i.e., we are averaging over the last 1000 examples in each of the 10 repetitions of a dataset. When calculating the $\overline{\text{RMAE}}$, we are comparing to the oracle mean regressor, i.e., the average values we divide by are calculated on averages of all target values, even on the examples that were artificially unlabeled. We present the obtained $\overline{\text{RMAE}}$ values as plots to qualitatively compare the observed methods.

6.5 Experimental Evaluation of Online Feature Ranking with Symbolic Random Forests

To address the feature ranking task in an online learning setting, we use the **symbolic random forest** method, which utilizes a random forest of iSOUP-Trees. We calculate the feature importances from the internal structure of the trees in the ensemble, where an attribute that appears in a split node closer to the tree root is given a higher score than an attribute that appears in a split node that appears closer to the leaves.

As we have discussed in Section 5.4, it is untenable to objectively evaluate the quality of feature importance scores. It is however possible to compare multiple sets of feature importance scores among themselves using the Jaccard similarity and the Canberra distance.

In this setting, we compare the feature importances obtained in the online learning setting to the ones obtained in the batch setting. As a comparison, we use the implementation of the symbolic random forest feature ranking method implemented in the Clus software[7]. The members of this random forest are randomized predictive clustering trees, similar to the randomized iSOUP-Trees in the online random forest.

We compare the two methods along two dimensions. First, we look how the Canberra distance evolves when more and more examples are processed in comparison to the batch feature ranking. For each example in the dataset, we calculate the Canberra distance between the current feature ranking induced from feature importances obtained from the online random forest and the ranking induced from the final feature importances obtained by the batch random forest. In particular, we are interested whether the online feature ranking converges toward the batch feature ranking. The other dimension we observe is the Jaccard similarity of the final online feature importance and the batch feature importance. The Jaccard similarity allows us to see what portions of the attributes overlap.

This experimental setup is very preliminary and we first intend to qualitatively explore and discuss how the feature ranking task is different between the online and the batch learning settings.

[7] URL: https://sourceforge.net/projects/clus/ (accessed 2018/01/22)

Chapter 7

Results and Discussion

Reality is frequently inaccurate.

— Douglas Adams

In this chapter, we present and discuss the experimental results per task in the same order that we described the experimental designs in Chapter 6. We start with the experimental results and discussion pertaining to the evaluation of the online multi-target regression methods and continue with the results that regard the online multi-label classification methods. Afterwards, we present and discuss the results of the evaluation of the online hierarchical prediction methods. We continue with a discussion of the online semi-supervised multi-target regression experiments. Finally, we conclude with a comparison of the batch and online approaches for the online feature ranking task.

7.1 Results of Experimental Evaluation of Online Multi-Target Regression Methods

In this section, we present the experimental results for the multi-target regression experiments and use them to answer the corresponding experimental questions that we defined in Section 6.1. We start by examining the predictive performance of the models and continue with a review of the consumption of computational resources. Afterwards, we discuss how we can achieve a trade-off between the use of additional computational resources and better predictive performance.

7.1.1 Predictive performance

The results of the experiments in terms of $\overline{\text{RMAE}}$ are presented in Table 7.1. We note that all of the $\overline{\text{RMAE}}$ results are below 1, which means that each of the observed tree-based methods performs better than the mean regressor, i.e., a regressor that always predicts the mean of the observed target values.

When comparing an option tree (iSOUP-OT) with a regular iSOUP-Tree, we notice that using option trees produces slightly better results in terms of $\overline{\text{RMAE}}$ than regular trees. This can be observed in the majority of the datasets (excluding EUNITE03, RF1 and SCM20d datasets). However, the differences between regular and option trees on these datasets are relatively small. On the other hand, neither regular nor option trees outperform the local method in performance, which is a strong baseline for the online multi-target regression task.

Table 7.1: Predictive performance in terms of $\overline{\text{RMAE}}$ ($\downarrow$) on the multi-target regression datasets. The table contains the values of $\overline{\text{RMAE}}$ (and the ranks) of each method on each of the datasets.

	Local	**iSOUP-Tree**	**iSOUP-OT**	**iSOUP-Bag**	**iSOUP-RF**
Bicycles	0.4717 [3]	0.5257 [4]	0.4039 [1]	0.4144 [2]	0.6408 [5]
EUNITE03	0.8199 [4]	0.7014 [2]	0.7504 [3]	0.5976 [1]	0.8916 [5]
RF1	0.1946 [5]	0.1861 [3]	0.1923 [4]	0.1832 [2]	0.1761 [1]
RF2	0.3834 [2]	0.5814 [5]	0.5454 [4]	0.5246 [3]	0.3711 [1]
F. Kras	0.6190 [4]	0.6461 [5]	0.5988 [3]	0.4766 [1]	0.4838 [2]
F. Slivnica	0.6397 [2]	0.7417 [5]	0.7028 [4]	0.6043 [1]	0.6569 [3]
SCM1d	0.3866 [1]	0.5360 [5]	0.5026 [4]	0.4084 [2]	0.4765 [3]
SCM20d	0.5283 [5]	0.3890 [3]	0.3903 [4]	0.2931 [2]	0.2825 [1]
Avg. rank	**3.25**	**4.0**	**3.375**	**1.75**	**2.625**

Option trees do not compare favorably to ensemble methods in terms of predictive performance. Specifically, they lose on almost all datasets to bagging (iSOUP-Bag) and online random forests (iSOUP-RF), though in the latter comparison the difference is lower. Both bagging and random forests are also notably better than the local approach. Overall, bagging of iSOUP-Trees outperforms all of the competitors, with random forests coming in second. These results are not unexpected, as these types of ensembles generally improve in predictive performance as was seen in the batch learning scenario (Kocev et al., 2013), though, curiously, in that case random forests outperform bagging. Both iSOUP-Tree and iSOUP-OT perform worse than the local FIMT-DD-based approach.

The results of the Friedman test and Nemenyi post-hoc analysis are summarized in the average rank diagram shown in Figure 7.1. We interpret average rank diagrams as follows. The methods are listed in order from highest average (worst) rank to lowest average (best) rank. Any methods that are further apart than the critical distance, which is displayed in the top of the diagram as a red line, have statistically significantly different performances. On the other hand, differences between methods which are connected with a red line, the length of which corresponds to the critical distance, cannot be confirmed as statistically significant, i.e., they are considered equivalent until further statistical evaluation.

Based on the critical distance diagram, we determine that the predictive performances of iSOUP-Tree and bagging of iSOUP-Trees are significantly different. We need further evidence to statistically confirm or deny any of the other observed differences. We also note that the local approach is hard to beat using a single tree approach.

7.1.2 Efficiency

The results in terms of the use of computational resources are shown in Figure 7.2 and 7.3. For brevity, only the results on four of the selected datasets (Bicycles, Forestry Slivnica, RF1 and SCM20d) are presented here, while the remaining results can be found in Figures A.2 and A.1 in Appendix A. The results on the remaining datasets are very similar. Additionally, each dataset is represented on two plots, as the scales for the single tree and the local approaches are drastically different from those of the ensemble methods. We first discuss the consumption of memory and continue with a discussion of the processing times.

When comparing the memory use of the local approach and the single iSOUP-Tree, we can see that (even when the number of targets is low) iSOUP-Tree uses less memory than

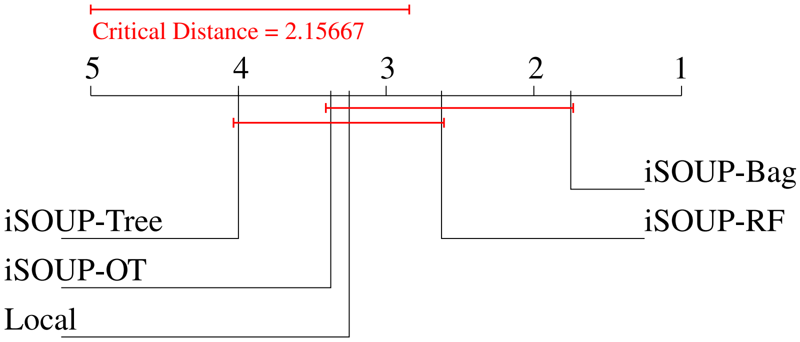


Figure 7.1: Average rank diagrams of $\overline{\text{RMAE}}$.

the local FIMT-DD approach. This is also the case on the Forestry Slivnica dataset, which only has 2 targets, and as such represents the simplest multi-target regression problem.

Option trees use more memory than both the single tree and the local approach. However, the growth rate of memory usage shows that they do not reach a size (e.g., in terms of the number of leaves) comparable to the size of the models produced by bagging. As compared to the 100 trees in the bagging ensemble, the option tree is composed of a smaller number of nodes.

As expected, random forests use considerably less memory than bagging. Their memory use is more similar to that of option trees, however, it still remains higher than that of the option trees.

When considering the memory consumption results, we notice several steep decreases in the amount of used memory. This occurs when multiple leaves are split in the observed time period. Whenever a leaf is split, the statistics the method was storing to calculate the splitting heuristics are forgotten, which releases sizable chunks of memory.

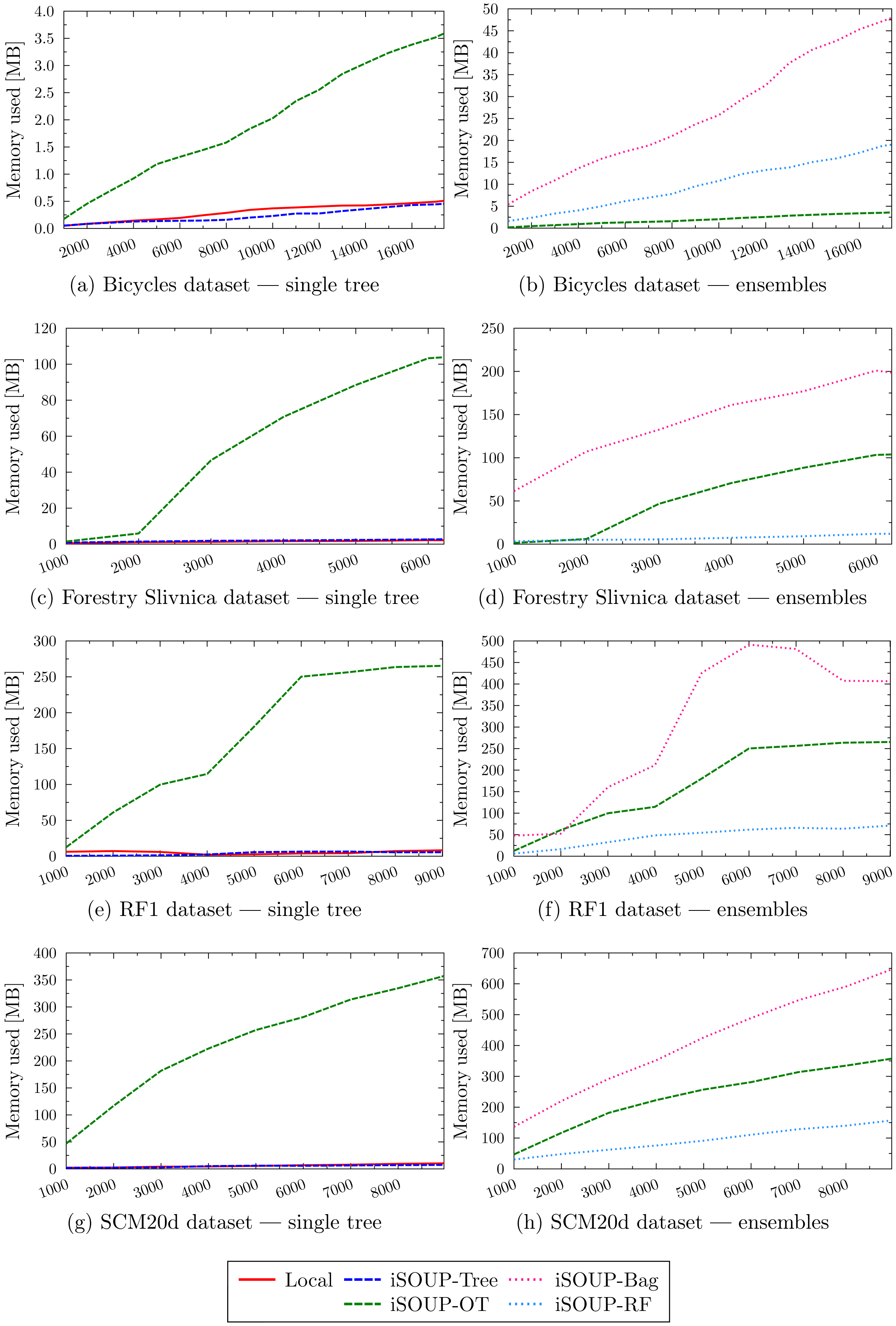
With respect to processing time shown in Figure 7.3, we again note that the iSOUP-Tree method generally outperforms (i.e., is quicker than) the local approach. As expected, option trees are slower than the local approach. The results of the ensemble methods in terms of processing time entirely follow our observations on the memory consumption.

7.1.3 Discussion

In the streaming setting, our desire for better predictive performance is often held back by strict constraints on the available computational resources. The introduced methods for online multi-target regression have a different trade-off between their predictive performance and their consumption of computational resources.

Specifically, the iSOUP-Tree method is the quickest and the least memory-intensive method. However, it also produces comparatively the worst results. Option trees use more memory and time and on average achieve better results than iSOUP-Tree, but the additional use of resources might be better spent on using the local approach, which produces better results with less memory and time. However, if the number of targets increased drastically, e.g., 10- or even 100-fold, the trade-off would most likely go in favor of the option trees.

Option trees never really grow to the size of the ensembles, which also explains their lower predictive performance. iSOUP-Bag, the winner in terms of predictive performance, uses about 100-times more computational resources than a single iSOUP-Tree. If the computational resources are unbounded, bagging is the clear method of choice. However, if the computational resources are constrained, random forests provide an excellent trade-



(a) Bicycles dataset — single tree

(b) Bicycles dataset — ensembles

(c) Forestry Slivnica dataset — single tree

(d) Forestry Slivnica dataset — ensembles

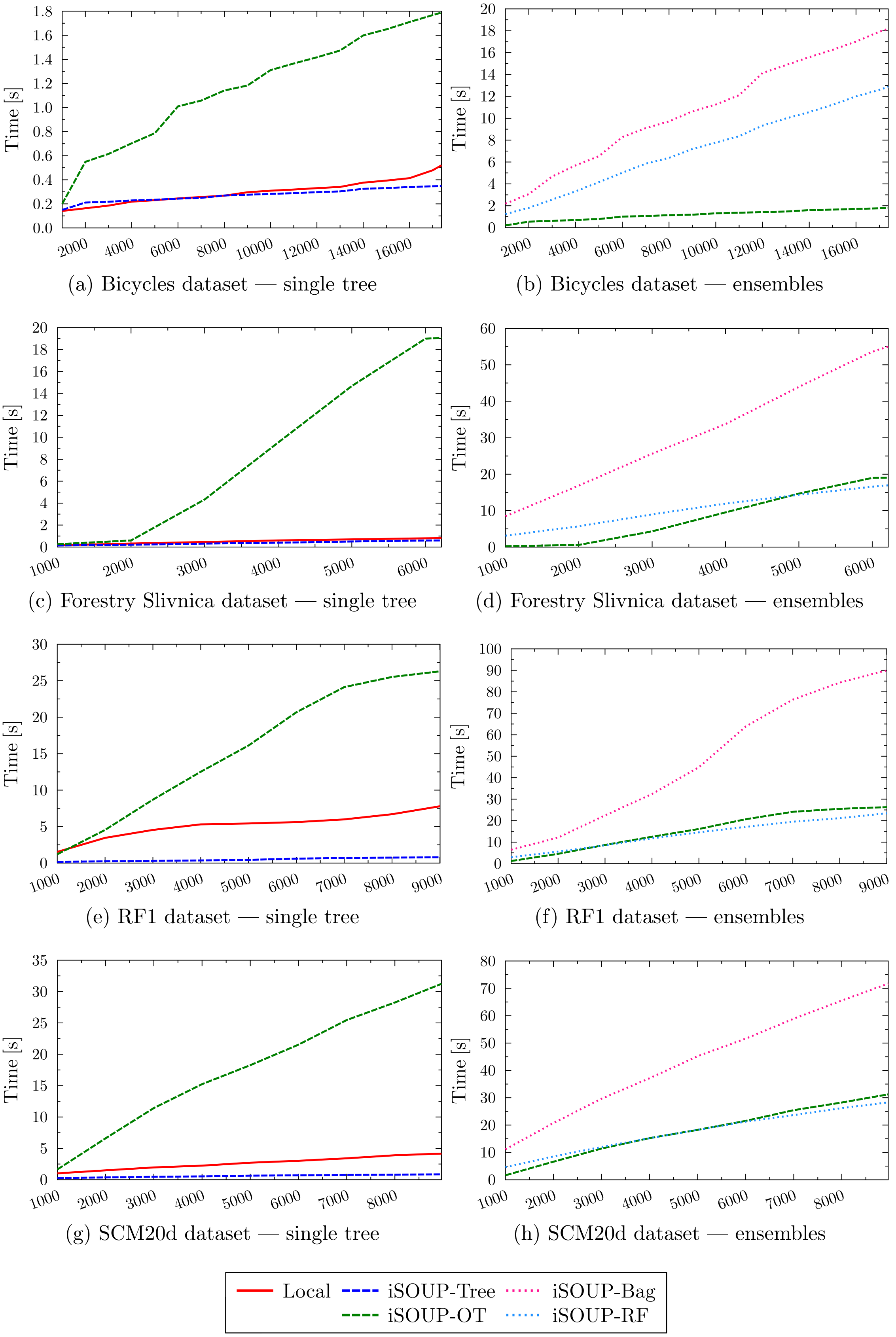
(e) RF1 dataset — single tree

(f) RF1 dataset — ensembles

(g) SCM20d dataset — single tree

(h) SCM20d dataset — ensembles

Figure 7.2: The memory consumption (↓) of the introduced methods. Horizontal axes show the numbers of processed examples.



(a) Bicycles dataset — single tree

(b) Bicycles dataset — ensembles

(c) Forestry Slivnica dataset — single tree

(d) Forestry Slivnica dataset — ensembles

(e) RF1 dataset — single tree

(f) RF1 dataset — ensembles

(g) SCM20d dataset — single tree

(h) SCM20d dataset — ensembles

Figure 7.3: The time consumption (↓) of the introduced methods. Horizontal axes show the numbers of processed examples.

off between the predictive performance and the use of computational resources. Specifically, they produce good results in terms of predictive performance, while consuming considerably fewer computational resources than the bagging method.

Many of our findings run in opposition to the results and findings of Ikonomovska et al. (2015) in the single-target scenario. There, both bagging and random forests performed equally to or worse than a single FIMT-DD tree, while option trees (ORTO) clearly outperformed the regular FIMT-DD tree.

7.2 Results of Experimental Evaluation of Online Multi-Label Classification via Online Multi-Target Regression

In this section, we present the results on the experimental evaluation of online MLC task by using MTR methods. For ease of discussion, we group the results of the experimental evaluation by the type of evaluation measure. Within each group of evaluation measures, we discuss how the results pertain to the experimental questions. Then, we continue with a discussion of the complete set of results.

7.2.1 Predictive performance: example-based measures

The values and rankings on the example-based measures (accuracy, $\mathrm{F^1}_{\mathrm{ex}}$ and Hamming score) are presented in Table 7.2, while the results of the Friedman-Nemenyi tests of statistical significance are presented in Figure 7.4 in the form of average rank diagrams.

The comparison of iSOUP model and regression trees shows us that the average rank of model trees is higher than the average rank of regression trees in all example-based measures, even though the difference is not statistically significant. The results on individual datasets in terms of the Hamming score are nearly identical, while model trees are slightly better on the accuracy and $\mathrm{F^1}_{\mathrm{ex}}$ measures. Even when regression trees beat model trees on a particular dataset, the difference in performance is much smaller on the datasets where the model trees achieve better performance.

The results of the comparison between the single-tree methods on the example-based evaluation measures are not entirely clear-cut. For both accuracy and $\mathrm{F^1}_{\mathrm{ex}}$, the average rank of the $\mathrm{HT_{PS}}$ method is higher than the average ranks of model and regression trees, but the difference is not statistically significant. The $\mathrm{HT_{PS}}$ method has worse performance on the Enron and TMC datasets, where regression and model trees both outperform $\mathrm{HT_{PS}}$. However, the results on the Hamming score show that the average ranks of both iSOUP model and regression trees are considerably higher than the rank of the $\mathrm{HT_{PS}}$ method. In this case, the differences in performance between model trees and the $\mathrm{HT_{PS}}$ method are statistically significant.

In terms of the accuracy and $\mathrm{F^1}_{\mathrm{ex}}$, we again observe varied results. We notice that, on some datasets, a group of methods' results that are orders of magnitude better than results of the other methods, i.e., $\mathrm{HT_{PS}}$ and $\mathrm{E_AHT_{PS}}$ on the Slashdot dataset, MT, RT, $\mathrm{E_BMT}$ and $\mathrm{E_BRT}$ on the Enron dataset, and $\mathrm{E_AHT_{PS}}$ on the IMDB dataset. We found no statistically significant differences in the performance for both the accuracy measure (Figure 7.4a) as well as the $\mathrm{F^1}_{\mathrm{ex}}$ measure (Figure 7.4b). On the other hand, the results in terms of the Hamming score are much clearer. MT, RT, $\mathrm{E_BMT}$ and $\mathrm{E_BRT}$ have a higher average rank than $\mathrm{HT_{PS}}$ and $\mathrm{E_AHT_{PS}}$. However, according to the Friedman-Nemenyi post-hoc test, only $\mathrm{HT_{PS}}$ is significantly worse than MT, $\mathrm{E_BRT}$ and $\mathrm{E_BMT}$ (Figure 7.4c).

Table 7.2: Predictive performance in terms of example-based measures (↑). Each table contains the values of the measure (and the rank) of each method on each dataset.

(a) Accuracy

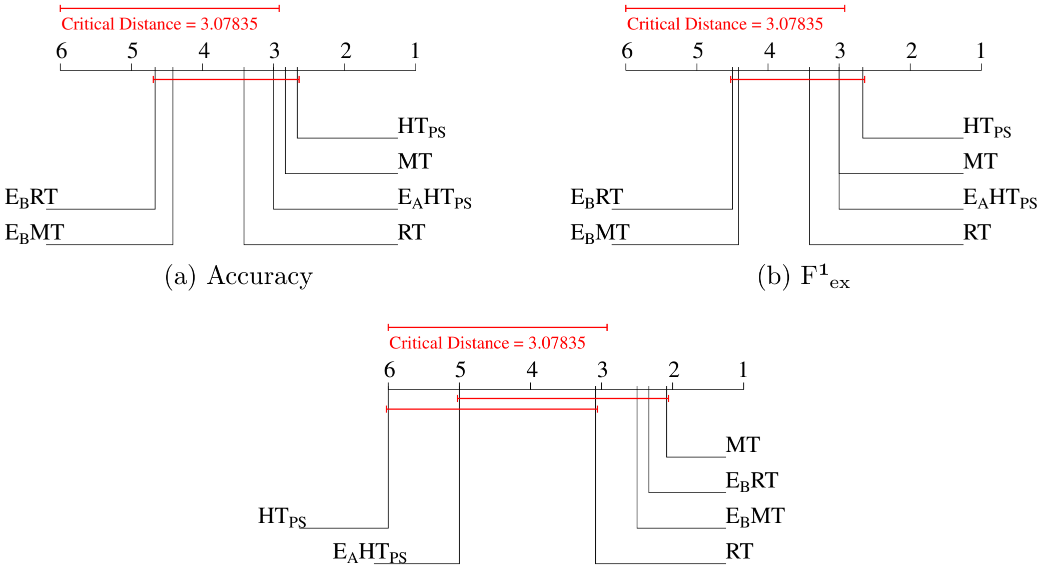
	MT	**RT**	$\mathbf{E_{B}RT}$	$\mathbf{E_{B}MT}$	$\mathbf{HT_{PS}}$	$\mathbf{E_{A}HT_{PS}}$
20NG	0.1142 [4]	0.1174 [3]	0.0682 [5]	0.0648 [6]	0.3182 [1]	0.2773 [2]
Enron	0.2438 [1]	0.1797 [4]	0.1887 [3]	0.2379 [2]	0.0022 [5]	0.0010 [6]
IMDB	0.0187 [3]	0.0026 [5]	0.0007 [6]	0.0031 [4]	0.0435 [2]	0.1955 [1]
Ohsumed	0.1563 [4]	0.1611 [3]	0.1143 [5]	0.1035 [6]	0.3178 [1]	0.2980 [2]
Slashdot	0.0049 [3]	0.0003 [4]	0.0000 [6]	0.0003 [4]	0.1393 [2]	0.1452 [1]
TMC	0.3448 [2]	0.3479 [1]	0.3439 [3]	0.3317 [4]	0.0112 [5]	0.0094 [6]
Avg. rank	**2.83**	**3.33**	**4.67**	**4.33**	**2.67**	**3.00**

(b) $\mathrm{F^{1}}_{\mathrm{ex}}$

	MT	**RT**	$\mathbf{E_{B}RT}$	$\mathbf{E_{B}MT}$	$\mathbf{HT_{PS}}$	$\mathbf{E_{A}HT_{PS}}$
20NG	0.1146 [4]	0.1177 [3]	0.0683 [5]	0.0649 [6]	0.3205 [1]	0.2804 [2]
Enron	0.3296 [1]	0.2411 [4]	0.2530 [3]	0.3221 [2]	0.0039 [5]	0.0015 [6]
IMDB	0.0227 [3]	0.0031 [5]	0.0008 [6]	0.0037 [4]	0.0597 [2]	0.2469 [1]
Ohsumed	0.1767 [4]	0.1829 [3]	0.1280 [5]	0.1156 [6]	0.3612 [1]	0.3382 [2]
Slashdot	0.0049 [3]	0.0003 [4]	0.0000 [6]	0.0003 [4]	0.1455 [2]	0.1493 [1]
TMC	0.4303 [3]	0.4335 [1]	0.4307 [2]	0.4175 [4]	0.0163 [5]	0.0138 [6]
Avg. rank	**3.00**	**3.33**	**4.50**	**4.33**	**2.67**	**3.00**

(c) Hamming score

	MT	**RT**	$\mathbf{E_{B}RT}$	$\mathbf{E_{B}MT}$	$\mathbf{HT_{PS}}$	$\mathbf{E_{A}HT_{PS}}$
20NG	0.9523 [1]	0.9522 [2]	0.9512 [3]	0.9511 [4]	0.9311 [6]	0.9432 [5]
Enron	0.9416 [2]	0.9375 [4]	0.9381 [3]	0.9419 [1]	0.9250 [6]	0.9350 [5]
IMDB	0.9282 [4]	0.9284 [3]	0.9286 [2]	0.9286 [1]	0.8886 [6]	0.9151 [5]
Ohsumed	0.9344 [1]	0.9341 [2]	0.9330 [3]	0.9326 [4]	0.9224 [6]	0.9249 [5]
Slashdot	0.9461 [4]	0.9461 [3]	0.9463 [1]	0.9463 [1]	0.9154 [6]	0.9233 [5]
TMC	0.9154 [1]	0.9146 [4]	0.9151 [2]	0.9149 [3]	0.8483 [6]	0.8503 [5]
Avg. rank	**2.17**	**3.00**	**2.33**	**2.33**	**6.00**	**5.00**



(a) Accuracy

(b) $F^1{}_{ex}$

(c) Hamming score

Figure 7.4: Average rank diagrams for the example-based measures.

7.2.2 Predictive performance: label-based measures

The values of the label-based performance measure (Precision$_{macro}$, Recall$_{macro}$, $F^1{}_{macro}$, Precision$_{micro}$, Recall$_{micro}$ and $F^1{}_{micro}$) and the corresponding method ranks are presented in Tables 7.3 and 7.4. The results of the Friedman-Nemenyi post-hoc significance tests are presented in Figure 7.5.

On all macro-averaged label-based evaluation measures, model trees achieve better or equal results as the results of regression trees. When regression trees outperform model trees on some datasets, e.g., all of the macro-averaged measures on the Ohsumed dataset, the differences are relatively small. On the other hand, when the model trees outperform regression trees, e.g., all macro-averaged measures on the IMDB dataset, the differences are considerably larger. For all three measures, the difference in average ranks of the methods is not statistically significant.

The results on the micro-averaged measures are similar. Model trees have highe a average rank than regression trees in terms of Precision$_{micro}$, though the differences are not statistically significant. The results on Recall$_{micro}$ and $F^1{}_{micro}$ are more scattered, with model still mostly having higher average rank than regression trees. Again, however, the differences in performance when model trees win are considerably larger than when regression trees outperform them.

When comparing the single tree methods, we find that the results on two of the datasets, Enron and TMC, deviate from the rest. Noticeably, on the remaining datasets HT$_{PS}$ outperforms model and regression trees on all label-based measures, with the exception of Precision$_{macro}$ and Precision$_{micro}$, while on the Enron and TMC datasets regression and model trees outperform HT$_{PS}$ on all label-based evaluation measures. Additionally, the results for Precision$_{macro}$ and Precision$_{micro}$ show that single-tree iSOUP-Tree methods also outperform HT$_{PS}$ on the remaining datasets.

The comparison of all of the methods in terms of each of the label-based evaluation measures is not straightforward. Bagging methods (excluding E$_A$HT$_{PS}$) perform relatively badly according to Recall$_{macro}$, Recall$_{micro}$, $F^1{}_{macro}$ and $F^1{}_{micro}$, as can be seen from the

Table 7.3: Predictive performance in terms of macro-averaged label-based measures (↑). Each table contains the values of the measure (and the rank) of each method on each dataset.

(a) Precision$_{\text{macro}}$ measure.

	MT	**RT**	**E$_{\mathbf{B}}$RT**	**E$_{\mathbf{B}}$MT**	**HT$_{\mathbf{PS}}$**	**E$_{\mathbf{A}}$HT$_{\mathbf{PS}}$**
20NG	0.5527 [1]	0.3873 [4]	0.3351 [5]	0.4279 [3]	0.1944 [6]	0.4427 [2]
Enron	0.0679 [1]	0.0341 [6]	0.0427 [5]	0.0588 [3]	0.0643 [2]	0.0474 [4]
IMDB	0.2306 [2]	0.1452 [3]	0.0576 [5]	0.2824 [1]	0.0392 [6]	0.1157 [4]
Ohsumed	0.2872 [2]	0.2946 [1]	0.2788 [3]	0.2612 [4]	0.1653 [6]	0.1907 [5]
Slashdot	0.1347 [2]	0.0152 [5]	0.0000 [6]	0.0227 [4]	0.1311 [3]	0.1842 [1]
TMC	0.3321 [1]	0.3135 [3]	0.3185 [2]	0.2380 [4]	0.0081 [6]	0.0083 [5]
Avg. rank	**1.50**	**3.67**	**4.33**	**3.17**	**4.83**	**3.50**

(b) Recall$_{\text{macro}}$ measure.

	MT	**RT**	**E$_{\mathbf{B}}$RT**	**E$_{\mathbf{B}}$MT**	**HT$_{\mathbf{PS}}$**	**E$_{\mathbf{A}}$HT$_{\mathbf{PS}}$**
20NG	0.1127 [4]	0.1156 [3]	0.0667 [5]	0.0635 [6]	0.3156 [1]	0.2787 [2]
Enron	0.0319 [1]	0.0193 [4]	0.0205 [3]	0.0299 [2]	0.0096 [5]	0.0019 [6]
IMDB	0.0060 [3]	0.0012 [4]	0.0002 [6]	0.0010 [5]	0.0411 [2]	0.0557 [1]
Ohsumed	0.0840 [4]	0.0884 [3]	0.0495 [5]	0.0406 [6]	0.1623 [1]	0.1433 [2]
Slashdot	0.0021 [3]	0.0001 [4]	0.0000 [6]	0.0001 [4]	0.0861 [1]	0.0586 [2]
TMC	0.1237 [2]	0.1332 [1]	0.1070 [3]	0.0981 [4]	0.0413 [5]	0.0388 [6]
Avg. rank	**2.83**	**3.17**	**4.67**	**4.50**	**2.50**	**3.17**

(c) F$^{\mathbf{1}}{}_{\text{macro}}$ measure.

	MT	**RT**	**E$_{\mathbf{B}}$RT**	**E$_{\mathbf{B}}$MT**	**HT$_{\mathbf{PS}}$**	**E$_{\mathbf{A}}$HT$_{\mathbf{PS}}$**
20NG	0.1619 [4]	0.1630 [3]	0.1047 [5]	0.0999 [6]	0.2287 [2]	0.2717 [1]
Enron	0.0364 [1]	0.0199 [4]	0.0217 [3]	0.0340 [2]	0.0127 [5]	0.0032 [6]
IMDB	0.0113 [3]	0.0023 [4]	0.0004 [6]	0.0019 [5]	0.0239 [2]	0.0540 [1]
Ohsumed	0.1210 [4]	0.1269 [3]	0.0745 [5]	0.0617 [6]	0.1586 [1]	0.1523 [2]
Slashdot	0.0041 [3]	0.0002 [5]	0.0000 [6]	0.0002 [4]	0.0627 [1]	0.0487 [2]
TMC	0.1503 [2]	0.1605 [1]	0.1228 [3]	0.1110 [4]	0.0064 [5]	0.0041 [6]
Avg. rank	**2.83**	**3.33**	**4.67**	**4.50**	**2.67**	**3.00**

Table 7.4: Predictive performance in terms of micro-averaged label-based measures (↑). Each table contains the values of the measure (and the rank) of each method on each dataset.

(a) $\text{Precision}_{\text{micro}}$ measure.

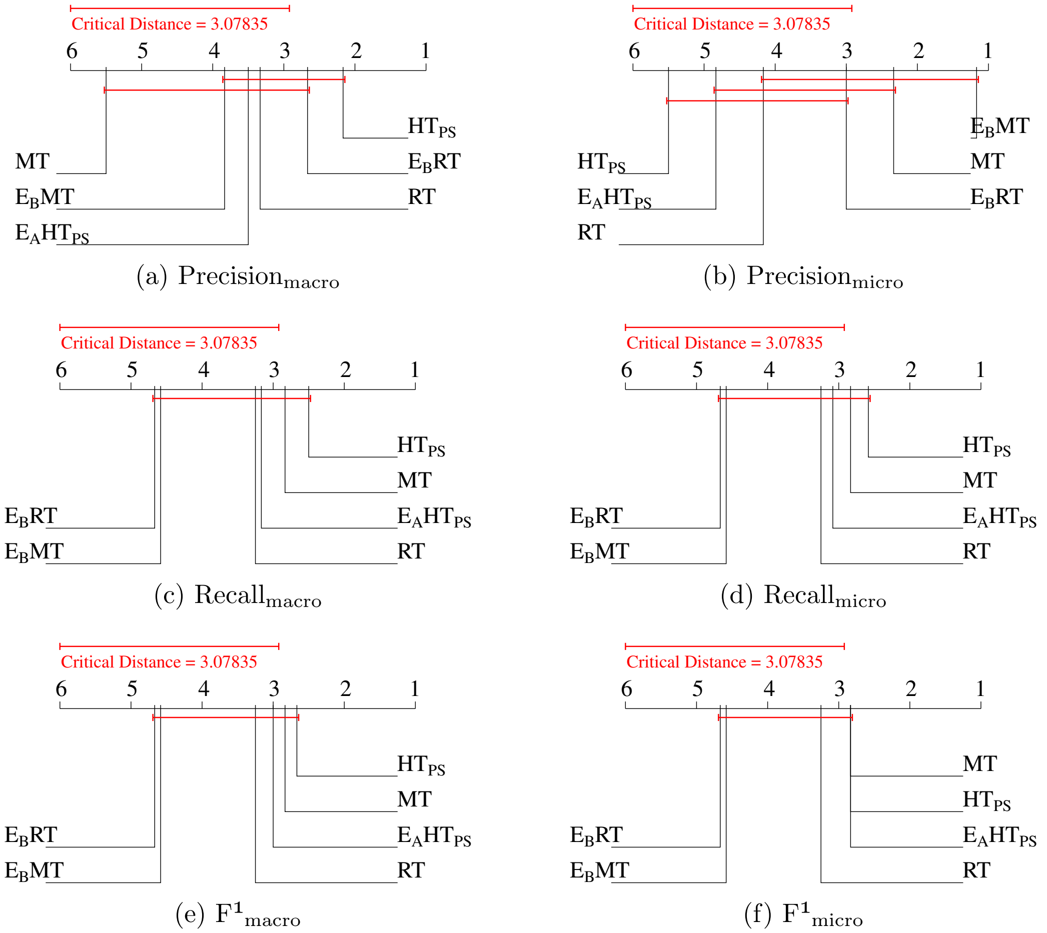
	MT	**RT**	**$\mathbf{E_B}$RT**	**$\mathbf{E_B}$MT**	**$\mathbf{HT_{PS}}$**	**$\mathbf{E_A HT_{PS}}$**
20NG	0.7408 [3]	0.7189 [4]	0.8227 [2]	0.8270 [1]	0.3253 [6]	0.4218 [5]
Enron	0.6108 [2]	0.5363 [4]	0.5539 [3]	0.6249 [1]	0.0664 [6]	0.0863 [5]
IMDB	0.4411 [3]	0.3746 [4]	0.5242 [2]	0.5864 [1]	0.0844 [6]	0.3461 [5]
Ohsumed	0.7561 [3]	0.7216 [4]	0.8086 [2]	0.8189 [1]	0.4453 [6]	0.4677 [5]
Slashdot	0.3220 [1]	0.0556 [5]	0.0000 [6]	0.2500 [2]	0.1587 [4]	0.1923 [3]
TMC	0.6427 [2]	0.6263 [4]	0.6394 [3]	0.6481 [1]	0.0280 [5]	0.0248 [6]
Avg. rank	**2.33**	**4.17**	**3.00**	**1.17**	**5.50**	**4.83**

(b) $\text{Recall}_{\text{micro}}$ measure.

	MT	**RT**	**$\mathbf{E_B}$RT**	**$\mathbf{E_B}$MT**	**$\mathbf{HT_{PS}}$**	**$\mathbf{E_A HT_{PS}}$**
20NG	0.1123 [4]	0.1151 [3]	0.0666 [5]	0.0633 [6]	0.3161 [1]	0.2786 [2]
Enron	0.2330 [1]	0.1424 [4]	0.1520 [3]	0.2214 [2]	0.0136 [5]	0.0021 [6]
IMDB	0.0182 [3]	0.0029 [4]	0.0006 [6]	0.0028 [5]	0.0568 [2]	0.2123 [1]
Ohsumed	0.1374 [4]	0.1439 [3]	0.0965 [5]	0.0865 [6]	0.2957 [1]	0.2762 [2]
Slashdot	0.0043 [3]	0.0002 [4]	0.0000 [6]	0.0002 [4]	0.1341 [1]	0.1341 [1]
TMC	0.3644 [2]	0.3803 [1]	0.3643 [3]	0.3428 [4]	0.0149 [5]	0.0126 [6]
Avg. rank	**2.83**	**3.17**	**4.67**	**4.50**	**2.50**	**3.00**

(c) $\text{F}^1_{\text{micro}}$ measure.

	MT	**RT**	**$\mathbf{E_B}$RT**	**$\mathbf{E_B}$MT**	**$\mathbf{HT_{PS}}$**	**$\mathbf{E_A HT_{PS}}$**
20NG	0.1950 [4]	0.1985 [3]	0.1232 [5]	0.1176 [6]	0.3207 [2]	0.3356 [1]
Enron	0.3374 [1]	0.2251 [4]	0.2385 [3]	0.3270 [2]	0.0225 [5]	0.0041 [6]
IMDB	0.0350 [3]	0.0057 [4]	0.0012 [6]	0.0056 [5]	0.0679 [2]	0.2632 [1]
Ohsumed	0.2325 [4]	0.2399 [3]	0.1724 [5]	0.1564 [6]	0.3554 [1]	0.3473 [2]
Slashdot	0.0084 [3]	0.0004 [5]	0.0000 [6]	0.0004 [4]	0.1454 [2]	0.1580 [1]
TMC	0.4651 [2]	0.4732 [1]	0.4642 [3]	0.4484 [4]	0.0195 [5]	0.0167 [6]
Avg. rank	**2.83**	**3.33**	**4.67**	**4.50**	**2.83**	**2.83**



(a) $\text{Precision}_{\text{macro}}$

(b) $\text{Precision}_{\text{micro}}$

(c) $\text{Recall}_{\text{macro}}$

(d) $\text{Recall}_{\text{micro}}$

(e) $\text{F}^1{}_{\text{macro}}$

(f) $\text{F}^1{}_{\text{micro}}$

Figure 7.5: Average rank diagrams for the label-based measures.

average rank diagrams in Figure 7.5.

Interestingly, bagging of model trees performs very well in terms of $\text{Precision}_{\text{micro}}$, where it statistically significantly outperforms both HT_{PS} and $\text{E}_{\text{A}}\text{HT}_{\text{PS}}$. Additionally, model trees also significantly outperform HT_{PS}. On the other hand, we only have enough evidence to conclude that HT_{PS} significantly outperforms model trees in terms of $\text{Precision}_{\text{macro}}$. We found no other statistically significant differences in method ranks on any of the remaining label-based measures.

7.2.3 Predictive performance: ranking-based measures

The values of the evaluation measures and the corresponding rankings on the ranking-based measures (ranking loss, logarithmic loss and average precision) are presented in Table 7.5, while the results of the Friedman-Nemenyi significance tests are presented in Figure 7.6.

The differences between the results of model and regression trees on the ranking-based evaluation measures are very small. There is variation in which type of tree performs better over the different measures. The average rank of regression trees is slightly higher than that of model trees for ranking loss, while the opposite is true for logarithmic loss and average precision. Both iSOUP regression and model trees outperform HT_{PS} in terms of ranking loss and logarithmic loss (and the difference in performance is statistically significant). In terms of average precision, their results are very close with each of the

Table 7.5: Predictive performance in terms of ranking-based measures (ranking loss ↓, logarithmic loss ↓, average precision ↑). Each table contains the values of the measure (and the rank) of each method on each dataset.

(a) Ranking loss

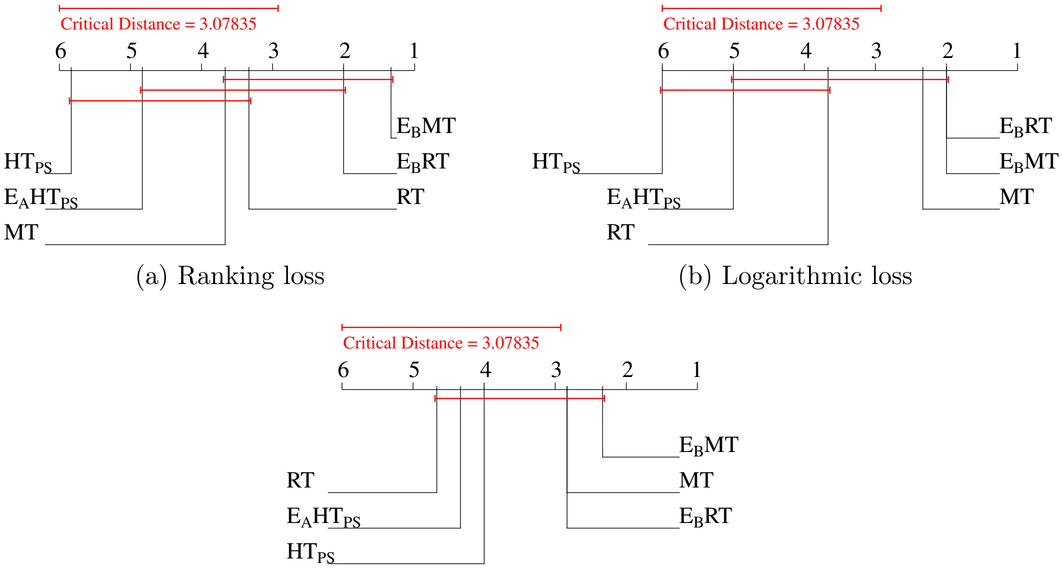
	MT	**RT**	**$\mathbf{E_B}$RT**	**$\mathbf{E_B}$MT**	**$\mathbf{HT_{PS}}$**	**$\mathbf{E_A HT_{PS}}$**
20NG	0.2672 [4]	0.2768 [5]	0.2272 [2]	0.2271 [1]	0.3852 [6]	0.2545 [3]
Enron	0.1208 [4]	0.1181 [2]	0.1183 [3]	0.1165 [1]	0.3474 [6]	0.3430 [5]
IMDB	0.1878 [4]	0.1737 [3]	0.1708 [2]	0.1705 [1]	0.5912 [6]	0.2615 [5]
Ohsumed	0.2254 [4]	0.2163 [3]	0.2024 [1]	0.2110 [2]	0.3752 [6]	0.3102 [5]
Slashdot	0.2202 [2]	0.2216 [4]	0.2206 [3]	0.2185 [1]	0.4801 [6]	0.3760 [5]
TMC	0.1220 [4]	0.1158 [3]	0.1012 [1]	0.1132 [2]	0.4688 [5]	0.4820 [6]
Avg. rank	**3.67**	**3.33**	**2.00**	**1.33**	**5.83**	**4.83**

(b) Logarithmic loss

	MT	**RT**	**$\mathbf{E_B}$RT**	**$\mathbf{E_B}$MT**	**$\mathbf{HT_{PS}}$**	**$\mathbf{E_A HT_{PS}}$**
20NG	0.1771 [3]	0.1785 [4]	0.1648 [1]	0.1671 [2]	0.6799 [6]	0.2923 [5]
Enron	0.1565 [1]	0.1697 [4]	0.1693 [3]	0.1574 [2]	0.5582 [6]	0.4736 [5]
IMDB	0.2089 [1]	0.2147 [4]	0.2104 [3]	0.2101 [2]	1.3033 [6]	0.4726 [5]
Ohsumed	0.2293 [4]	0.2279 [3]	0.2103 [1]	0.2148 [2]	0.7401 [6]	0.4860 [5]
Slashdot	0.1824 [1]	0.1857 [4]	0.1839 [3]	0.1834 [2]	0.6972 [6]	0.4575 [5]
TMC	0.2328 [4]	0.2290 [3]	0.2128 [1]	0.2212 [2]	1.5564 [6]	1.3226 [5]
Avg. rank	**2.33**	**3.67**	**2.00**	**2.00**	**6.00**	**5.00**

(c) Average precision

	MT	**RT**	**$\mathbf{E_B}$RT**	**$\mathbf{E_B}$MT**	**$\mathbf{HT_{PS}}$**	**$\mathbf{E_A HT_{PS}}$**
20NG	0.1793 [5]	0.1755 [6]	0.1900 [2]	0.1928 [1]	0.1831 [3]	0.1816 [4]
Enron	0.1131 [1]	0.1023 [4]	0.1024 [3]	0.1125 [2]	0.0739 [6]	0.0750 [5]
IMDB	0.1986 [2]	0.1901 [5]	0.1907 [4]	0.1972 [3]	0.2385 [1]	0.1758 [6]
Ohsumed	0.1846 [2]	0.1806 [4]	0.1836 [3]	0.1848 [1]	0.1674 [6]	0.1748 [5]
Slashdot	0.1586 [3]	0.1529 [6]	0.1585 [4]	0.1565 [5]	0.1871 [2]	0.1951 [1]
TMC	0.1992 [4]	0.2001 [3]	0.2121 [1]	0.2016 [2]	0.1698 [6]	0.1708 [5]
Avg. rank	**2.83**	**4.67**	**2.83**	**2.33**	**4.00**	**4.33**



(a) Ranking loss

(b) Logarithmic loss

(c) Average precision

Figure 7.6: Average rank diagrams for the ranking-based measures.

methods outperforming the rest on some of the datasets.

Finally, the ranking diagram for the algorithms in terms of ranking loss shows that bagging with model trees generally performs best on all of the datasets, followed by bagging of regression trees, regression and model trees, and finally $\mathrm{E_AHT_{PS}}$ and $\mathrm{HT_{PS}}$. In terms of statistical significance, bagging of model trees is better than $\mathrm{HT_{PS}}$ and $\mathrm{E_AHT_{PS}}$, and bagging of regression trees is better than $\mathrm{HT_{PS}}$, as can be seen from Figure 7.6a. The results in terms of logarithmic loss are very similar. Here, bagging of model trees, bagging of regression trees, as well as single model trees, statistically significantly outperform $\mathrm{HT_{PS}}$ (Figure 7.6b). The results for average precision are mixed, with different methods taking first and last rank on different datasets (Figure 7.6c). Hence, no statistically significant differences were observed.

7.2.4 Efficiency

The values and rankings on the efficiency measures (memory and time use) are presented in Table 7.6. The results of the Friedman-Nemenyi significance tests are presented in Figure 7.7.

The expected result of model trees using both more memory and time is evident from Table 7.6. While the difference in memory use is relatively small, the time used is increased by about 10%–20% when using model trees.

$\mathrm{HT_{PS}}$ uses considerably less memory when compared to model and regression trees. In terms of time use, it performs better than iSOUP single trees on some datasets, while it uses considerably more time on others. The differences in time are most likely due to the pruned sets procedure, as the base tree learning steps are similar in all single-tree methods.

Overall, regression trees are the quickest method, while $\mathrm{HT_{PS}}$ are to be the least memory intensive. We observe several statistically significant differences. Specifically, $\mathrm{HT_{PS}}$ uses less memory than bagging of regression trees and bagging of model trees, regression trees and $\mathrm{E_AHT_{PS}}$ use less memory than bagging of model trees, regression trees are quicker than bagging of model trees and $\mathrm{E_AHT_{PS}}$, and both MT and $\mathrm{HT_{PS}}$ are quicker

Table 7.6: Efficiency results: Memory and time consumption (↓). Each table contains the values of the efficiency measure (and the rank) of each method on each dataset.

(a) Memory [MB]

	MT	**RT**	$\mathbf{E_{B}RT}$	$\mathbf{E_{B}MT}$	$\mathbf{HT_{PS}}$	$\mathbf{E_{A}HT_{PS}}$
20NG	101.61 [4]	101.44 [3]	970.35 [5]	972.03 [6]	3.43 [1]	22.46 [2]
Enron	10.20 [3]	9.77 [2]	99.92 [5]	104.20 [6]	6.56 [1]	56.82 [4]
IMDB	382.33 [4]	382.08 [3]	3575.54 [5]	3578.03 [6]	22.53 [1]	39.31 [2]
Ohsumed	191.29 [4]	191.09 [3]	1368.17 [5]	1370.13 [6]	3.63 [1]	32.89 [2]
Slashdot	15.45 [3]	15.26 [2]	140.02 [5]	141.94 [6]	3.72 [1]	26.03 [4]
TMC	36.75 [4]	36.65 [3]	304.34 [5]	305.31 [6]	2.30 [1]	18.62 [2]
Avg. rank	3.67	**2.67**	**5.00**	**6.00**	**1.00**	**2.67**

(b) Time [s]

	MT	**RT**	$\mathbf{E_{B}RT}$	$\mathbf{E_{B}MT}$	$\mathbf{HT_{PS}}$	$\mathbf{E_{A}HT_{PS}}$
20NG	33.0 [2]	28.6 [1]	196.2 [4]	230.1 [5]	51.9 [3]	723.2 [6]
Enron	7.7 [3]	6.6 [2]	41.0 [5]	48.6 [6]	1.9 [1]	13.1 [4]
IMDB	277.1 [2]	250.9 [1]	2434.2 [4]	2971.1 [5]	344.5 [3]	4723.5 [6]
Ohsumed	27.9 [2]	25.3 [1]	168.7 [4]	195.3 [5]	39.7 [3]	632.7 [6]
Slashdot	7.1 [2]	5.8 [1]	41.5 [4]	48.6 [5]	9.0 [3]	78.3 [6]
TMC	26.1 [3]	22.4 [2]	150.5 [4]	188.4 [5]	14.5 [1]	259.1 [6]
Avg. rank	**2.33**	**1.33**	**4.17**	**5.17**	**2.33**	**5.67**

than $\mathrm{E_{A}HT_{PS}}$ as shown in Figure 7.7.

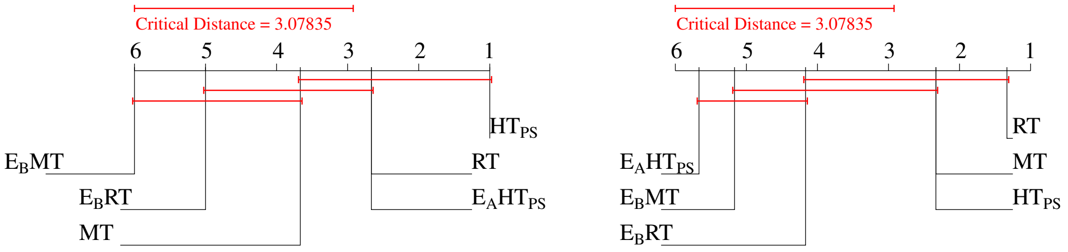
7.2.5 Discussion

In the discussion of the experiments in the context of online multi-label classification by using online multi-target regression methods, we start with a few general remarks, then proceed to address each of the experimental questions in turn.

First, we emphasize again that both example-based and label-based measures depend on the selected threshold. While threshold selection is far from a trivial task in the multi-label classification scenario and its difficulty is further compounded in the online learning setting, we expect that a better selection of threshold would increase the performance of any multi-label classifier. Whether the selection is best done on a method-by-method basis, dataset-by-dataset basis or even for each method-dataset pair requires substantial further investigation. Thus, we place a stronger emphasis on the threshold-independent ranking-based evaluation measures.

Regarding the first experimental question, whether model trees with the adaptive perceptron outperform regression trees, based on the above results, we conclude that some improvement can be gained by using model trees. However, we also note that using model trees increases the use of resources, which can be a limiting factor when choosing a method for a particular data mining problem. While the increase in memory use is relatively small, model tree construction consumes about 10-20% more time as compared to regression trees.

Continuing to the second question of how iSOUP single model and regression trees compare to Hoeffding trees with pruned sets, we observe that, on the example-based and



(a) Memory consumption

(b) Time consumption

Figure 7.7: Average rank diagrams for the efficiency measures.

label-based evaluation measures, all of the trees have a similar performance. The only exception is the Hamming score. However, based on the observations made on the ranking-based methods, we conclude that both model and regression iSOUP trees either perform as well as or outperform Hoeffding trees with pruned sets.

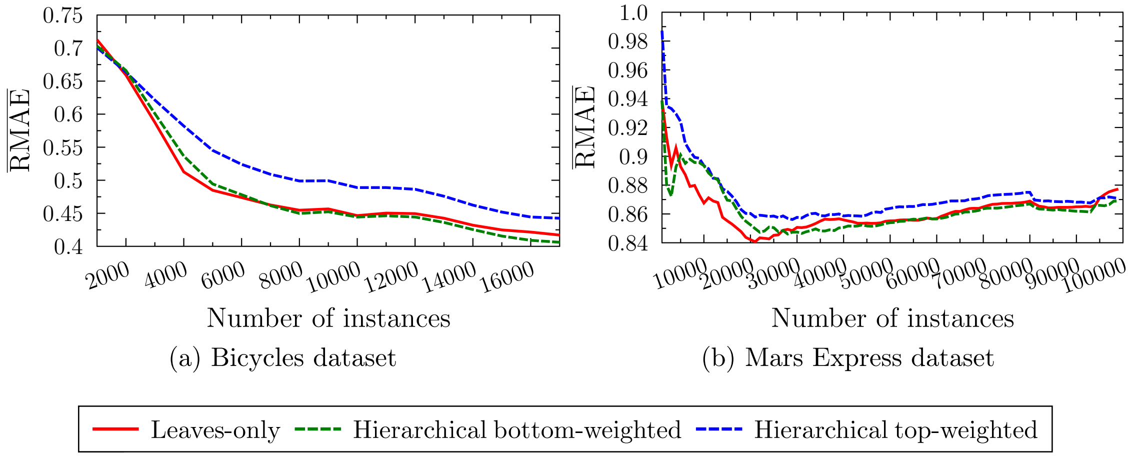
Looking at the results of the comparisons of both single tree and ensemble methods along the various evaluation measures, we find that different methods achieve the best results for different evaluation measures. This is to be expected, as the different learning procedures inevitably optimize different quantities. This in turn influences the performance evaluation for a given evaluation measure, depending on the similarity of the optimized quantity and the evaluation measure.

We recorded some statistically significant differences, most notably for the ranking-based ranking loss and logarithmic loss measures. The bagging ensemble of iSOUP model trees outperformed both single tree and ensembles of Hoeffding trees with pruned sets for the first measure and only single Hoeffding trees with pruned sets for the second. These significant differences are very important, as the ranking-based measures in question are threshold independent, while measures like precision and recall can be traded off by setting different thresholds.

Finally, in terms of the use of computational resources, the results show that, in general, the additional use of computational resources by the more complex methods, like model trees or ensembles of models, contributes to predictive better performance. This justifies the use of more complex models. However, we must be aware that there are considerable costs to using more complex models. This is especially relevant for particular data mining applications, where such models might be expected to operate on low-memory or computationally slow devices.

7.3 Results of Experimental Evaluation of Online Hierarchical Prediction with iSOUP-Trees

In this section, we present the experimental results that pertain to the experimental scenarios of online hierarchical multi-target regression and hierarchical multi-label classification. We start with the experimental results for the former, and continue with the results for the latter task. We conclude with a brief discussion of the experimental questions and the observed results.



(a) Bicycles dataset

(b) Mars Express dataset

Figure 7.8: Progression of $\overline{\text{RMAE}}$ (↓) on the hierarchical multi-target regression datasets.

7.3.1 Results for online hierarchical multi-target regression

The results of applying hierarchical and leaves-only variants of iSOUP-Tree to the two hierarchical multi-target regression datasets described in Section 6.3 are shown in Figure 7.8. On both datasets, the bottom-weighted hierarchical models outperform the top-weighted models. The bottom-weighted models start out slightly worse than the leaves-only tree, however, at some point the bottom-weighted models reach and even slightly beat its predictive performance. On the other hand, top-weighted models are worse than the leaves-only model, with the exception of a few intervals in either dataset where their predictive performances are comparable. Both the difference between the leaves-only model and the bottom-weighted model and the difference between the leaves-only model and the top-weighted models, might be the consequence of a slower growth of the model trees.

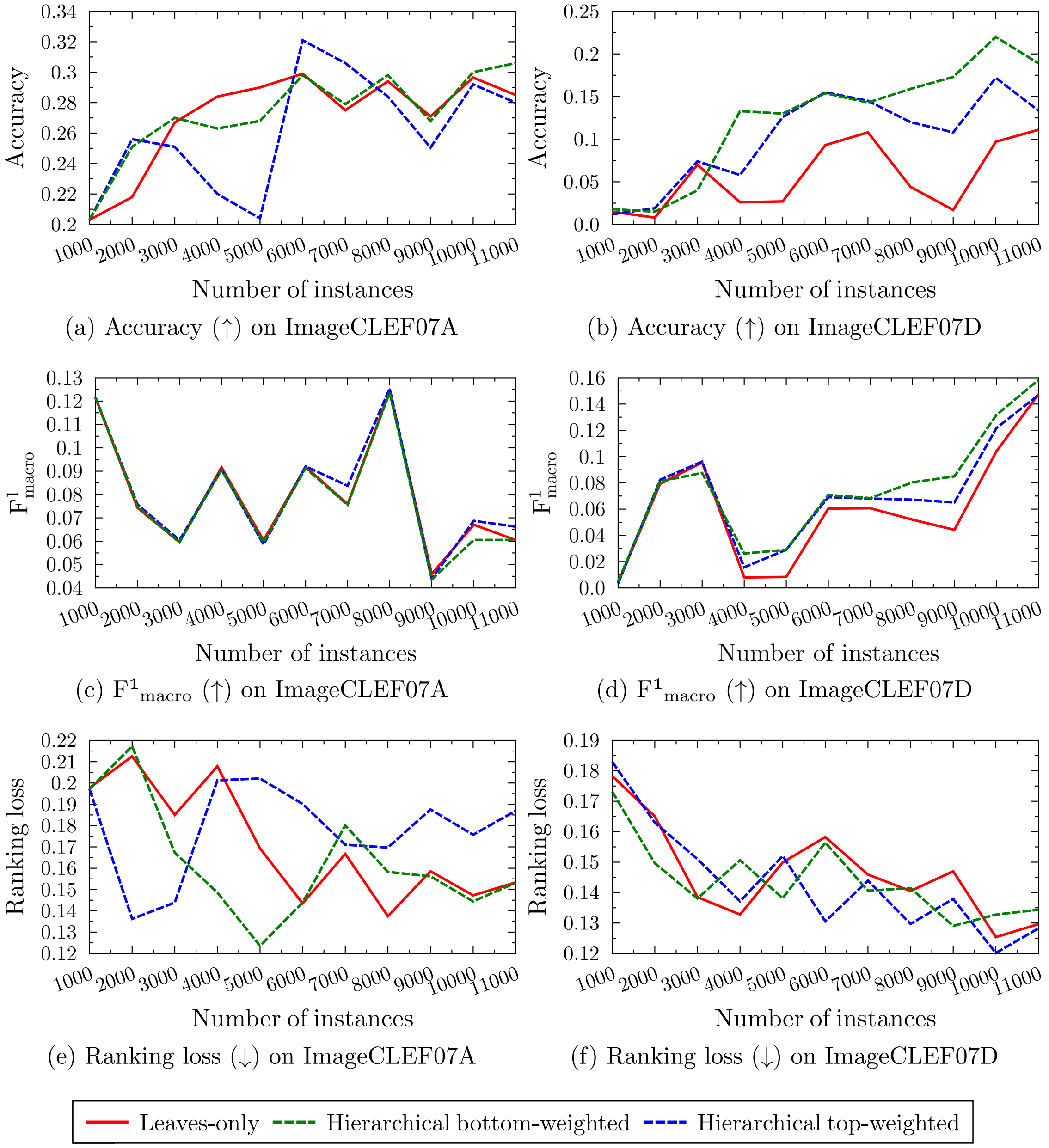
7.3.2 Results for online hierarchical multi-label classification

The experimental results for online hierarchical multi-label classification are more extensive than those for hierarchical multi-target regression. We select one measure of each category (example-, label- and ranking based measures), as discussed in Section 5.2.2 for comparison. The results of the comparison are shown in Figure 7.9. In this section, we present results on the performance of the accuracy, $\mathrm{F}^{\mathbf{1}}{}_{\mathrm{macro}}$ and ranking loss. The remainder of the results on the example-based (Figure A.3), label-based (Figures A.4 and A.5) and ranking-based (Figure A.6) measures are presented in Appendix A[1].

On the A variant of the ImageCLEF07 dataset, the results are not clear cut. The bottom-weighted model's accuracy generally mirrors that of the leaves-only model, while the top-weighted model's accuracy is more erratic, sometimes producing better predictions, sometimes worse. Notably, in general, accuracies on the A dataset are higher than those on the D dataset. On the other hand, on the ImageCLEF07D dataset, both the bottom-weighted and the top-weighted iSOUP-tree models outperform the leaves-only model in terms of accuracy (Figure 7.9b). Furthermore, the bottom-weighted models perform slightly better than the top-weighted models.

The results in terms of the $\mathrm{F}^{\mathbf{1}}{}_{\mathrm{macro}}$ measure are similar on the D variant of the ImageCLEF07 dataset. Both hierarchical models outperform the leaves-only model, while the

[1] Note again, that in the figures, the plotted values represent the values of the selected evaluation measures for the last 1000 examples. This results in more volatile measurements than in the hierarchical multi-target regression results.



(a) Accuracy (↑) on ImageCLEF07A

(b) Accuracy (↑) on ImageCLEF07D

(c) F^1_{macro} (↑) on ImageCLEF07A

(d) F^1_{macro} (↑) on ImageCLEF07D

(e) Ranking loss (↓) on ImageCLEF07A

(f) Ranking loss (↓) on ImageCLEF07D

Figure 7.9: The predictive performance results on the hierarchical multi-label classification datasets.

bottom-weighted model slightly outperforms the top-weighted model in terms of $\mathrm{F^1}_{\mathrm{macro}}$. On the A dataset though, values of $\mathrm{F^1}_{\mathrm{macro}}$ at a given time point are very close, with the top-weighted model slightly outperforming the leaves-only model and the bottom-weighted one. Interestingly, in both datasets, the observed values of $\mathrm{F^1}_{\mathrm{macro}}$ are very low.

Finally, in terms of the ranking loss measure, we see that the ranking loss is quite low in both variants of the ImageCLEF07 dataset. Given the results on the other measures, this suggests that using $\tau = 0.5$ as the classification threshold was not the optimal choice. The ranking losses on the A dataset show considerable fluctuations, while on the D dataset ranking losses are lower to a slight extent, as well as more stable. Overall, it is hard to determine a winning method in terms of the ranking loss on either variant of the dataset.

7.3.3 Discussion

Online hierarchical multi-target regression. On the hierarchical multi-target datasets, we can clearly see a difference between the top- and bottom-weighted hierarchical methods. In these cases, the bottom-weighted method appears superior in terms of the predictive performance. However, we must consider that we have chosen an evaluation procedure that is solely focused on the predictive performance in the leaf targets/labels. Intuitively, it seems that putting a larger weight on the variance of the leaves of the target hierarchy, as in the bottom-weighted iSOUP-Tree, we are selecting splits which first and foremost reduce the variance of the leaf targets. If we were to use a different evaluation methodology, where we would consider hierarchical evaluation measures in which the "cost" of an error higher up in the hierarchy is higher than for targets lower in the hierarchy, the top-weighted models might outperform the bottom-weighted ones.

With regard to the question of whether the use of the target hierarchy improves predictive performance, from these experiments we cannot conclusively say that this is the case. In our results, bottom-weighted models do eventually outperform the leaves-only ones. However, we must also consider whether adding additional targets to a multi-target regression problem inhibits the growth of the models. As we are averaging more and more individual variances in the calculation of the ICVR heuristic, we might encounter the effects of the central limit theorem, which states that the normalized sum of independent random variables tends toward a normal distribution[2]. In particular, as we average more values, the resulting heuristics of the split candidates will be distributed closer and closer to the normal distribution, with a prescribed mean.

Combined with the Hoeffding inequality, this would mean that the ratio of the heuristics of the best and second best split candidates becomes closer to 1, which in turn means that more examples have to accumulate to provide sufficient evidence for a split. This results in slower growth of the trees that consider additional targets. The effect gets more and more pronounced as more targets are added. To address this problem, we might explore the use of option trees, as they try to address the shortcoming of the Hoeffding inequality-based approaches.

Online Hierarchical multi-label classification. The largest problem of the comparison of the different versions of the iSOUP-Tree algorithms for online hierarchical multi-label classification is that the datasets which we compared them on are in essence not streaming datasets. Most importantly, we had to shuffle the original datasets to undo the ordering of the target values. The results shown here are one of the many possible results that can be obtained by shuffling the original datasets.

[2] Here, we specifically refer to the Lyapunov central limit theorem.

In order to evaluate the introduced methods further, we plan to explore how they perform on real streaming datasets. Unfortunately, hierarchical datasets that are snapshots of actual hierarchical data streams are still not readily available. Hence, pursing the development of methods for online hierarchical multi-label classification further on requires significant efforts in data collection and preparation of the collected data.

While the results on the example-based and label-based evaluation measures are inconclusive with regards to which of the methods is best suited to the online tasks, the results on the ranking-based measures are encouraging. Both ranking losses and logarithmic losses of the hierarchical iSOUP-Tree models were, at various points in the data set, higher than those of the leaves-only model. Therefore, a different selection of the classification threshold might possibly increase the performance on the other measures.

7.4 Results of Experimental Evaluation of Online Semi-Supervised Multi-Target Regression with iSOUP-PCTs

In the following sections, we present and discuss the results of the experimental evaluation of the iSOUP-PCT method for semi-supervised multi-target regression. The iSOUP-PCT method is compared to two baseline methods, supervised iSOUP-Tree and supervised iSOUP-PCT, as well as to an oracle method Oracle-iSOUP-Tree, which has access to all labeled examples.

7.4.1 Predictive performance

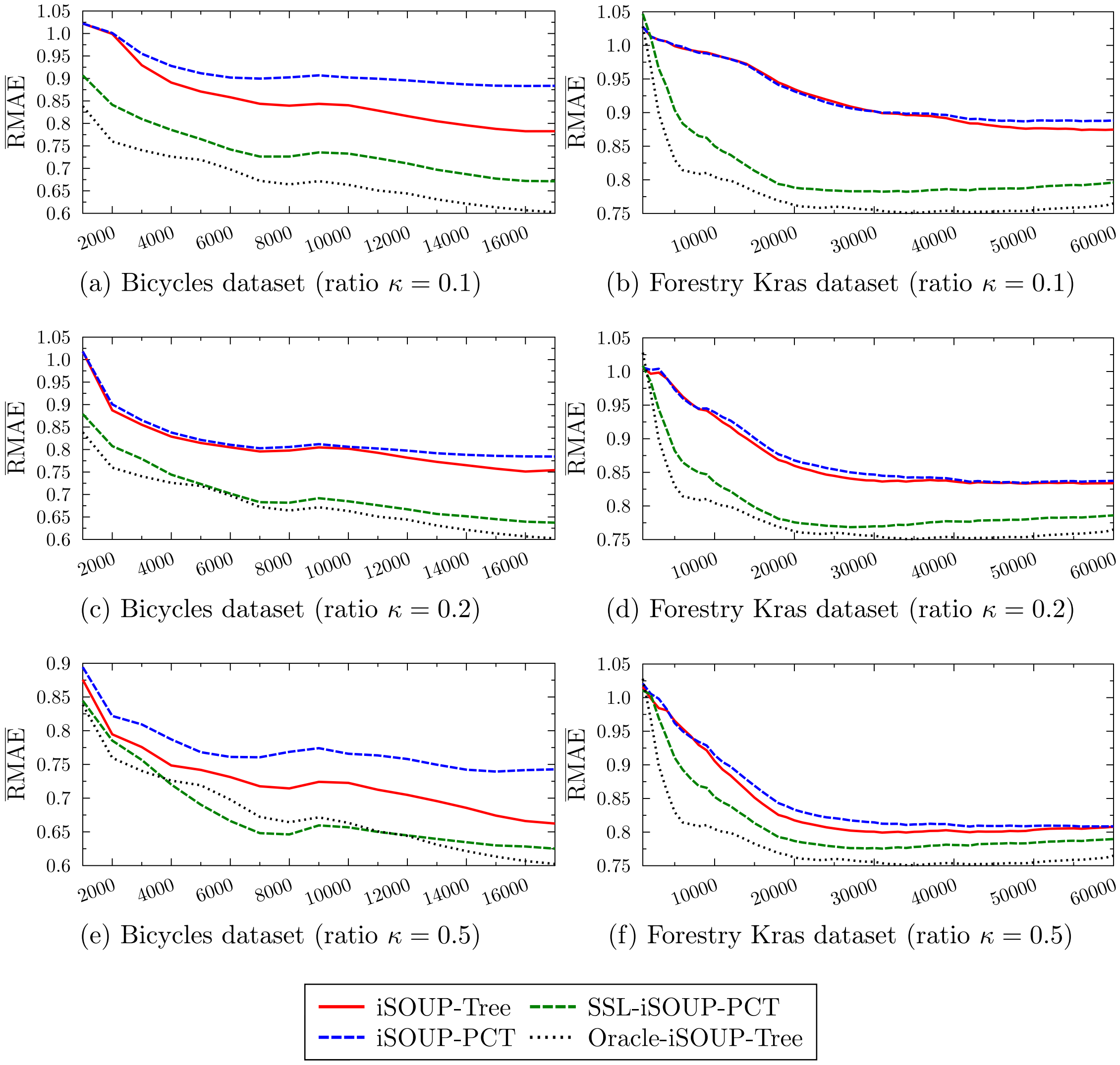
The results of the online semi-supervised multi-target regression experiments are shown in Figures 7.10 and 7.11. The results on the remaining datasets are omitted for brevity, though they can be found in Figures A.7 and A.8.

On all datasets and all labeling ratios κ, with the exception of $\kappa = 0.5$ on the Forestry Slivnica dataset, the semi-supervised iSOUP-PCT achieves better predictive performance than both the unsupervised iSOUP-Tree and iSOUP-PCT. As expected, on most datasets the SSL-iSOUP-Tree is worse than the Oracle-iSOUP-Tree, i.e., the regression tree that learns from the entire dataset as though it was labeled. Curiously, on the RF1, RF2 and SCM20d dataset the SSL-iSOUP-PCT actually performs better than the oracle model. On the RF2 dataset even the iSOUP-Tree and iSOUP-PCT outperform the oracle tree. This very likely means that mean regressors are not good as leaf models on the RF1 and RF2 dataset, as learning from more examples decreases the performance.

The results are unclear regarding the experimental comparison of iSOUP-Trees and iSOUP-PCTs. The two methods generally produce models that have a comparable predictive performance. iSOUP-PCTs are generally slightly worse than iSOUP-Tree, though, conversely, in some cases iSOUP-PCTs outperform the iSOUP-Tree to a small degree.

As we expected, the differences between the semi-supervised models and regular, supervised models lessen as we go from low to high labeling ratios. As the supervised models see more and more examples, the benefits of using the semi-supervised iSOUP-PCT method are reduced.

Finally, the actual values of the $\overline{\text{RMAE}}$ error are quite high, compared to the results from Section 7.1.1. Obviously, one factor that reduces the predictive performance is the reduced number of labeled examples. However, we must also consider that we are using regression trees in place of model trees. Thus, even the oracle trees, which see all examples as labeled, perform worse than above. This further confirms our earlier findings that model trees considerably improve over regression trees in terms of predictive performance.



(a) Bicycles dataset (ratio $\kappa = 0.1$)

(b) Forestry Kras dataset (ratio $\kappa = 0.1$)

(c) Bicycles dataset (ratio $\kappa = 0.2$)

(d) Forestry Kras dataset (ratio $\kappa = 0.2$)

(e) Bicycles dataset (ratio $\kappa = 0.5$)

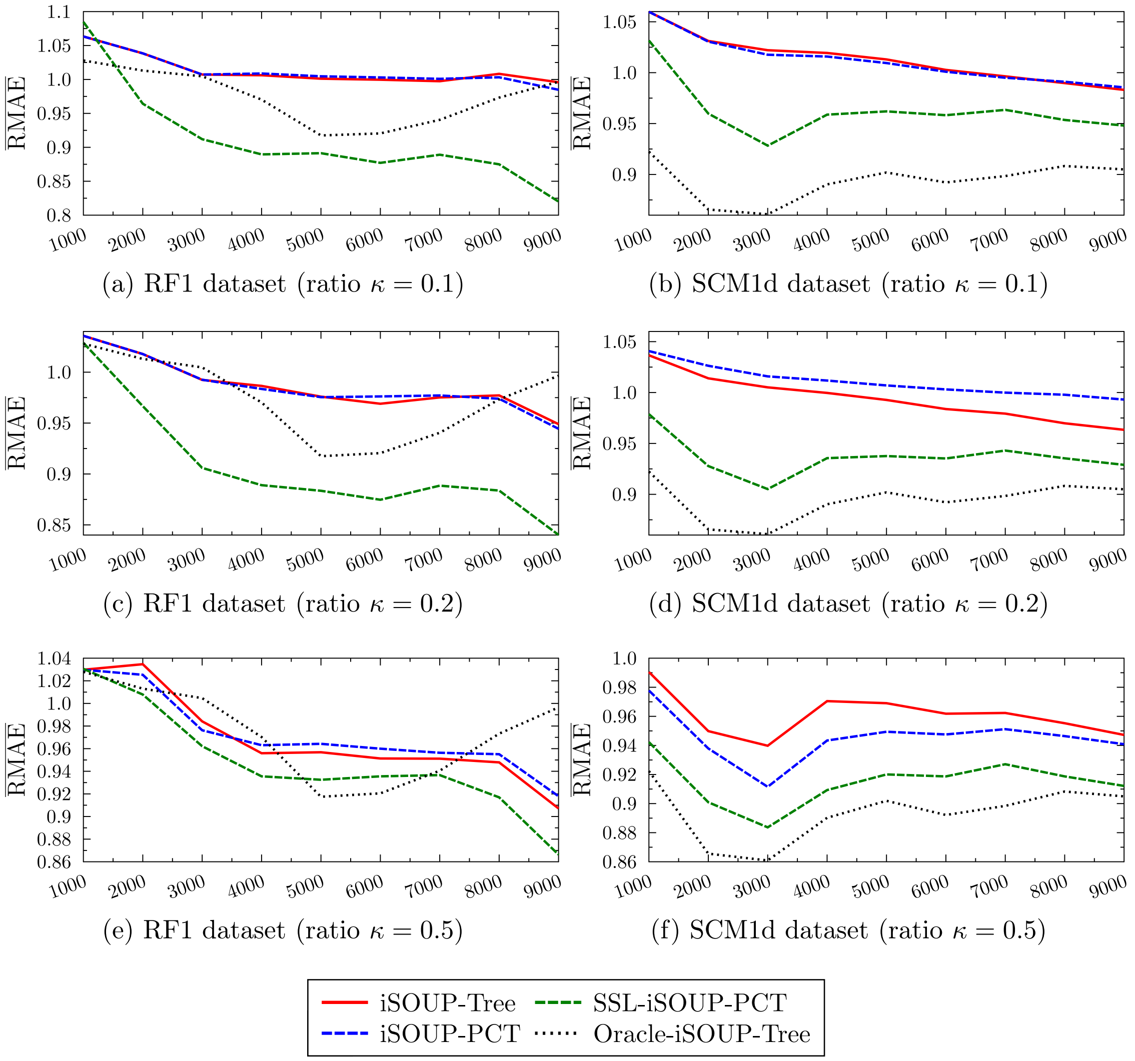
(f) Forestry Kras dataset (ratio $\kappa = 0.5$)

Figure 7.10: The predictive performance results in terms of $\overline{\text{RMAE}}$ (↓) on the Bicycles and Forestry Kras datasets in an online semi-supervised scenario.

7.4.2 Discussion

An important aspect of using iSOUP-PCTs is the trade-off between the use of additional resources and the achieved predictive performance. In particular, the improvement in predictive performance might come at such a high increase in memory usage or processing time to make it unfeasible. Thus, we should consider approaches which reduce the memory consumption. Using an approach similar to random forests, we could observe only a random subset of input attributes in each leaf.

iSOUP-PCTs might also suffer from slower growth than regular iSOUP-Trees for reasons already outlined above. In the PCT heuristic score, we are averaging additional variance reductions, resulting in split candidates which have closer heuristic scores. While the semi-supervised iSOUP-PCTs get more examples to calculate the heuristic values, the supervised iSOUP-PCTs receive the same labeled examples as iSOUP-Trees and generally perform slightly worse. The benefit of the semi-supervised approach thus appears to entirely stem from the fact that it can utilize unlabeled examples. While using the PCT heuristic achieves that utilization, it also introduces the above problem.



(a) RF1 dataset (ratio $\kappa = 0.1$)

(b) SCM1d dataset (ratio $\kappa = 0.1$)

(c) RF1 dataset (ratio $\kappa = 0.2$)

(d) SCM1d dataset (ratio $\kappa = 0.2$)

(e) RF1 dataset (ratio $\kappa = 0.5$)

(f) SCM1d dataset (ratio $\kappa = 0.5$)

Figure 7.11: The predictive performance results in terms of $\overline{\mathrm{RMAE}}$ (↓) on the RF1 and SCM1d datasets in an online semi-supervised scenario.

Interestingly, this is not a problem that occurs in the batch learning scenario, similarly, to how in the batch scenario there is no need for additional computational resources. There, the heuristic scores are directly compared to one another, instead of using a probabilistic bound on their ratios. This means that even a minuscule difference in the best and second best split candidates will result in a split. In the streaming scenario, a minuscule difference in the ratios is a very undesirable result as it stops the growth of the tree until either the Hoeffding inequality gets satisfied with additional examples or the tie-breaking mechanism selects one of the splits.

In the results, we also saw a significant reduction of the predictive performance of the oracle tree in comparison with the earlier results in the semi-supervised learning scenario. However, as outlined in Section 4.5, model trees are not always appropriate for this task. In particular, when we are learning from streams which have a very low labeling ratio, linear models do not have enough learning examples to converge to a good fit of the data.

7.5 Results of Experimental Evaluation of Online Feature Ranking with Symbolic Random Forests

In this section, we present the comparison of the batch and online variants of the symbolic random forest methods for online feature ranking. We observe the comparison in terms of two metrics, the Canberra distance and Jaccard similarity.

7.5.1 Comparison of feature rankings in online and batch settings

The results of the feature ranking experiments are shown in Figure 7.12 for the Canberra distance and in Figure 7.13 for the Jaccard distance. On all datasets the Canberra distance increases quickly at the start of the dataset. This is due to the fact that until the trees in the random forest grow, the induced ranking ranks all of the attributes the same. At about 200 examples, the first splits occur in the random forest and the online feature ranking starts to differentiate between the attributes. The distance on the first few hundred examples is thus an artifact of the comparison procedure. We also note that the differences in the Canberra distance entirely reflect the changes in the online ranking as the batch ranking remains unchanged throughout the entire process.

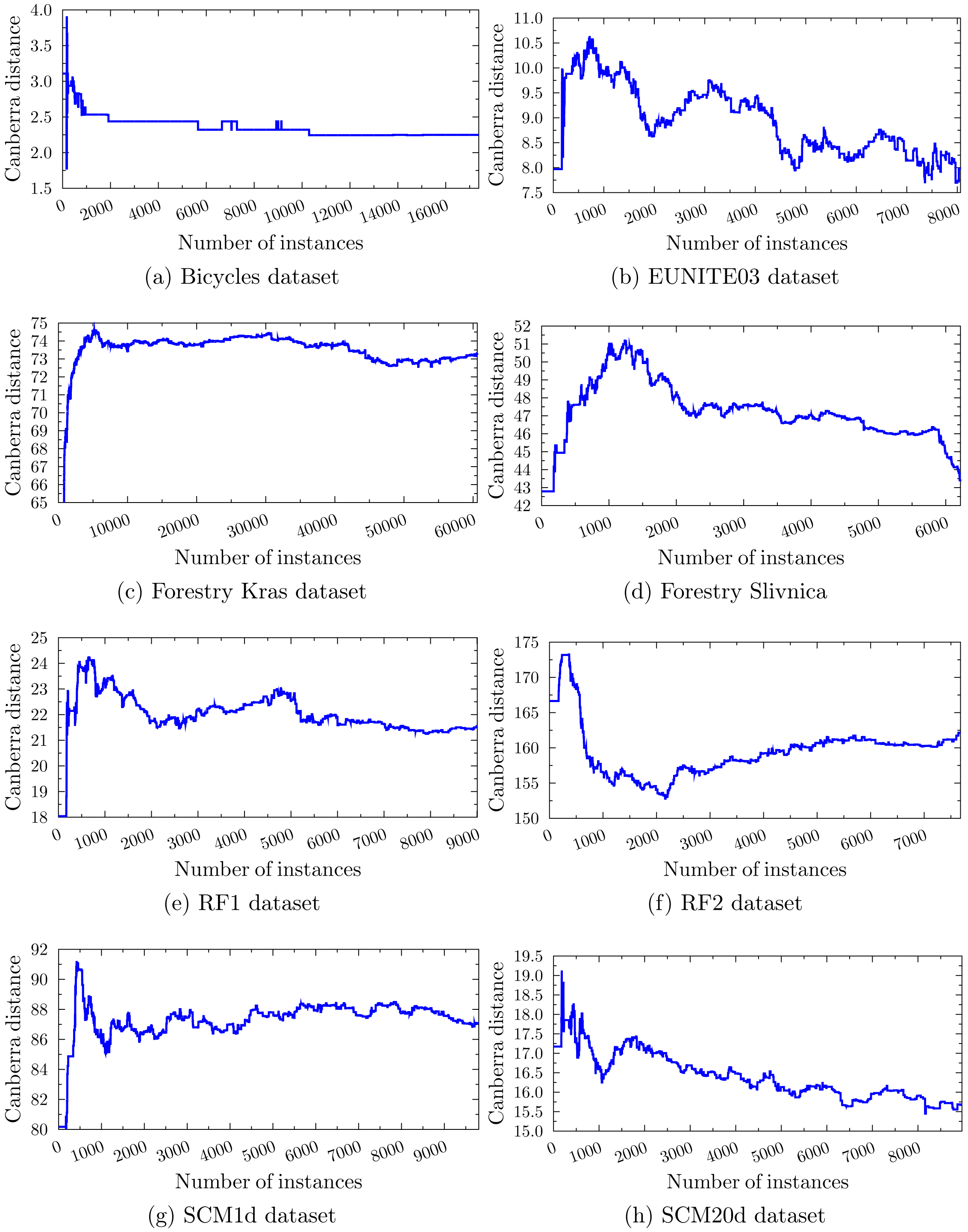
The results on the Canberra distance vary from dataset to dataset. On the Bicycles dataset the distance between the feature rankings obtained by the online and batch random forests converges fairly quickly, and only slight differences are obtained by learning from additional examples. On the EUNITE03, Forestry Slivnica and SCM20d datasets the distance behaves similarly. After the initial jump, the distance generally gradually decreases with additional examples, though it shows local increases. On the Forestry Kras, RF1, RF2 and SCM1d datasets the distance quickly settles on a distance, and in the case of the RF2 dataset even starts to increase with additional examples.

We can attempt to explain some of these findings through an analysis of the results of the Jaccard similarity. On the Bicycles dataset the rankings are fairly similar, e.g., of the top 5 attributes in either ranking, four of them are the same. We also see that the rankings agree on the top ranked attribute. Notably, the Bicycles is the smallest dataset among the observed datasets.

While the rankings do not agree on the top few attributes on the EUNITE03, Forestry Slivnica and RF2 datasets, they achieve a relatively high degree of agreement around the top 15, 70 and 90 attributes, respectively. Notably, on the Forestry Slivnica and RF2 datasets the online symbolic approach is not able to differentiate between a sizable portion of the bottom ranked attributes, while the same is not true of the batch approach. Consequently, for the purposes of the Jaccard similarity in the online ranking several of the bottom ranked attributes are ranked in order of appearance in the dataset, skewing the plots of the Jaccard similarity.

Of the observed rankings the ones on the Forestry Kras and RF1 datasets are the most different. In these cases, the Jaccard similarity grows very slowly. We also notice a pattern of linear growth on the Forestry Kras dataset from around top 120 attributes onward. This means that each attribute has already been included in either ranking, and thus the denominator of the Jaccard similarity has reached its maximum value. From then on, the numerator increases in equal steps, resulting in linear growth of the Jaccard similarity. The same happens also on the RF1 dataset, though it occurs at much higher values of k, at around $k = 65$.

Finally, on the SCM1d and SCM20d the rankings agree on the top ranked attribute, but not on the next several. On the SCM1d dataset a high level of agreement requires the inclusion of quite a large number of attributes, while on the SCM20d dataset a high level of agreement is reached relatively quickly at around top 10 attributes.



(a) Bicycles dataset

(b) EUNITE03 dataset

(c) Forestry Kras dataset

(d) Forestry Slivnica

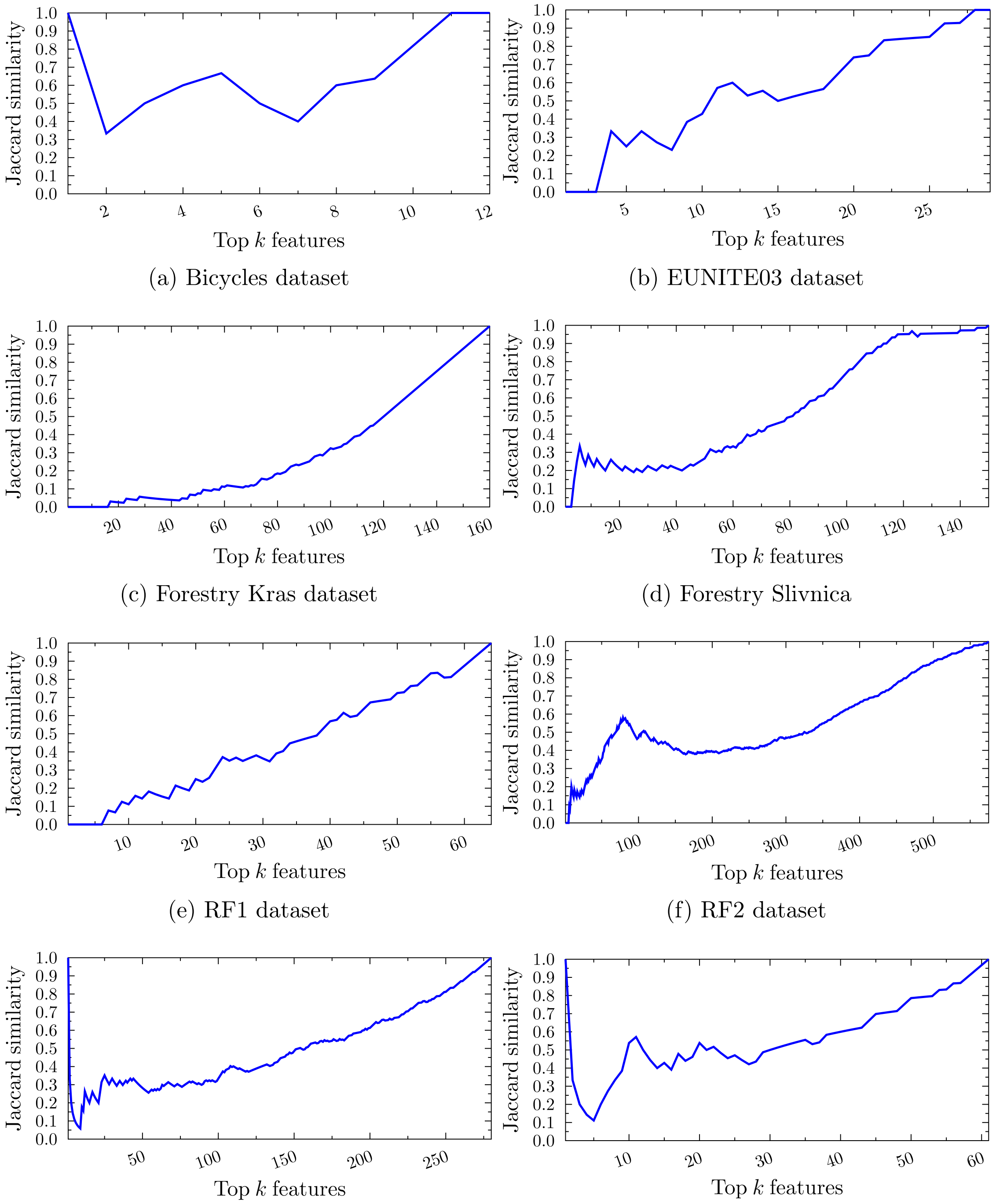
(e) RF1 dataset

(f) RF2 dataset

(g) SCM1d dataset

(h) SCM20d dataset

Figure 7.12: The Canberra distances (↓) between the feature rankings learned in batch and online settings.



(a) Bicycles dataset

(b) EUNITE03 dataset

(c) Forestry Kras dataset

(d) Forestry Slivnica

(e) RF1 dataset

(f) RF2 dataset

(g) SCM1d dataset

(h) SCM20d dataset

Figure 7.13: The Jaccard similarity (↑) between the final feature rankings learned in batch and online settings.

7.5.2 Discussion

Our experimental results show considerable promise for the online symbolic random forest feature ranking method. While the differences between the batch and online rankings are quite considerable on some datasets, this may be due to the fact that the attributes *actually* have different importances for the learning process.

Let us consider a attribute which for the first half of a dataset has value 0 and for the second half has value 1. At the point where the attribute's value changes the underlying concept changes from D_1 to D_2. When a batch feature ranking approach observes the dataset it sees it in its entirety, thus it is able to determine that the concepts prior and after the change in the attribute value are considerably different and consequently it heavily emphasizes the importance of the attribute.

An online feature ranking approach is only able to detect changes in the feature importance "locally," i.e., in a short period before and after the value changes. Notably, this is not directly related to the problem of concept drift detection, but is a consequence of the incremental updating of the random forest. When the value of the attribute is 0, the online approach is ranking the attributes according to the concept D_1. However, given that the observed attribute does ever change during this time frame it is never detected as important. Once the concept does change to D_2, the approach can try to adapt to the new concept, however, the attribute will again not be determined to be important, as it will again be constant from the point of the change onward.

Given that we are working under an assumption of stationarity, i.e., we assume no concept drift, and that we are generally working on batch benchmark datasets, we might repeat our experiments with reordered datasets. This could have a considerable impact on the online feature rankings as features that are important at the start of the learning process are awarded higher scores due to the inherent nature of the random forest learning process.

The applicability of the feature rankings in the online setting is also not as clear cut as in the batch learning scenario. There, feature ranking can severely reduce the cost of data collection and use of computational resources, by eliminating unimportant attributes. In the streaming setting, eliminating an attribute is not advisable, as their importance might change due to concept drift.

Thus, in the online setting, feature importances and rankings could be used in a different way. In particular, significant changes in the rankings mean that different attributes have become important for the predictive modeling process. This implies that the concept has changed. Hence, we could attempt to use feature importance and ranking approaches as detectors of concept drift. Furthermore, looking at the pre- and post-concept drift importances can also at the very least quantitatively describe how the concept has changed.

Chapter 8

Case Studies

> It was an exceedingly odd thing to see an elephant in a spacesuit.
>
> — Alastair Reynolds

In addition to evaluating the proposed method on benchmark datasets, as described in Chapters 6 and 7, in this thesis we consider two case studies that show how the developed methods can be applied in application domains. In the first, we address the task of predicting the power consumption of the European Space Agency's Mars Express probe, currently in orbit around the planet Mars. In the second, we address the task of predicting the photo-voltaic power generation across the United States from the data provided by the National Renewable Energy Laboratory (NREL). For these two case studies, we provide some insights into the practicality of applying predictive modeling in a data streaming setting, in addition to performing the performance comparisons described earlier (Chapters 6 and 7).

8.1 Predicting the Power Consumption of the Mars Express Probe

Mars Express (MEX), a space probe operated by the European Space Agency (ESA), has been orbiting Mars since the end of 2003. Its scientific payload consists of seven instruments that provide global coverage of Mars' surface, subsurface and atmosphere (Chicarro, Martin, & Trautner, 2004). The instruments and on-board equipment of MEX have to be kept within their operating temperature ranges, which range from –180°C for some equipment to standard room temperature for others. To maintain operating temperatures, the orbiter is equipped with an autonomous thermal system composed of heaters and passive coolers.

MEX is powered by electricity provided either by its solar arrays or batteries, when the arrays are in shadow. The thermal system, together with the platform units, consumes a significant amount of the available power, while the remaining power is used for science operations. The power consumption of the thermal system changes through time, depending on various external and internal factors, such as exposure of the orbiter to Sun or heat generated by the on-board equipment units. Predicting the power consumption of the thermal system is a non-trivial but crucial task, which allows the optimization of science operations of MEX.

To predict MEX's power consumption, its operators currently use a manually constructed model based on simplified physical models, expert knowledge and experience.

However, due to aging of the probe and decaying capacity of its batteries, power is a precious resource and every little bit saved in the thermal subsystem can be used for science acquisitions.

The Mars Express case study is derived from the Mars Express Power Challenge organized by the European Space Agency in 2016 (Breskvar, Kocev, Levatić, et al., 2017). The task of the Mars Express Power challenge was to predict the electric current at 33 different thermal heaters in the MEX probe for each operating hour, corresponding to the power consumption of the thermal regulation system. 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 probe's flight dynamics and information about the activations and deactivations of its internal systems. This data is also expected to suffer from concept drift, as the probe's different systems degrade in time resulting in different thermal properties.

8.1.1 Dataset

The attributes used in this dataset are the apparent influx of solar energy for each of the six sides of the approximate cuboid probe, as well as the influx of energy to its solar panels. The attributes take into account the solar angle of incidence to a particular surface of the probe, the position of the probe in relation to the Sun and the solar constant at a given time point that takes into account the distance to the Sun. Specifically, the attributes are calculated as

$$\texttt{feat}(t_i) = \int_{t_i}^{t_{i+1}} \mathrm{A_E}(t)\, \mathrm{c}(t)\, \mathrm{U}(t) \mathrm{d}t,$$

where $\texttt{feat} \in \{\texttt{front}, \texttt{back}, \texttt{left}, \texttt{right}, \texttt{up}, \texttt{down}, \texttt{panels}\}$, t_i and t_{i+1} are the time of the i-th and $(i+1)$-th measurement, respectively, $\mathrm{A_E}(t)$ is the apparent area of the given surface, $\mathrm{c}(t)$ is the solar constant and $\mathrm{U}(t)$ is the umbra coefficient. $\mathrm{A_E}(t)$ is calculated as

$$\mathrm{A_E} = \mathrm{A} \max\{\cos \alpha(t), 0\},$$

where $\alpha(t)$ is the angle of incidence for the given surface at time t and A is the area of the surface. However, since the values of the attributes are always compared only relative to the values of the same attribute and never to the values of the other attributes, we can, without loss of generality for the learning process, assume that $\mathrm{A} = 1$. The umbra coefficient conveys whether the probe is partially or completely in the Mars' shadow (or in the shadow of one of Mars' two moons, Phobos and Deimos), i.e., $\mathrm{U}(t) \equiv 0$ when the probe is fully in Mars' umbra (shadow) and $\mathrm{U}(t) \equiv 0.5$ when the probe is in Mars' penumbra (partial shadow). Otherwise, $\mathrm{U}(t) \equiv 1$. Additionally, the dataset also contains the sums of the above attributes, calculated as

$$\texttt{feat-sum}_\sigma(t_i) = \sum_{k=i-1}^{i-\sigma} \texttt{feat}(t_k),$$

for $\sigma \in \{4, 16, 32, 64, 128\}$ describing the probe's history. This yields a total of $7 + 7 \cdot 5 = 42$ continuous descriptive attributes.

We calculate the values of the attributes for each minute of operation, producing a total of around 2.6 million examples. Note that the competition called for hourly predictions. Our dataset is constructed at the one minute granularity to match the actual recordings of the target variables, which were recorded at a rate of about one measurement per minute. In the competition setting, the predictions for 60 consecutive minutes would then be aggregated to obtain the hourly predictions. For the purposes of this thesis, we only use the first 100k (of 2.6 million) examples, as the experimental setup, specifically, the measurement of the memory consumption[1] makes the learning on the entire dataset unfeasible.

8.1.2 Data mining task

In this case study, we are concerned specifically with how well the different methods for online multi-target regression (**Local**, **iSOUP-Tree**, **iSOUP-OT**, **iSOUP-Bag** and **iSOUP-RF**, defined as in Section 4.2) cope with the requirement of producing real-time minute-by-minute predictions. In this case, re-learning a model, e.g., a decision tree, from all data for each new example quickly becomes infeasible. This occurs as soon as we reach the point where the learning process lasts more than the one minute time window in which a prediction must be made.

8.1.3 Results and discussion

The results of the comparison of the different online multi-target regression methods for the Mars Express dataset are presented in Figure 8.1. In addition to the measures used in Section 6.1.2, we also show the average time spent processing an example. Specifically, this time includes the time to make the prediction for the example, as well as the time it takes to learn from the example, i.e., the time to update the model. The processing time naturally increases due to the tree growing larger and larger as the model is updated.

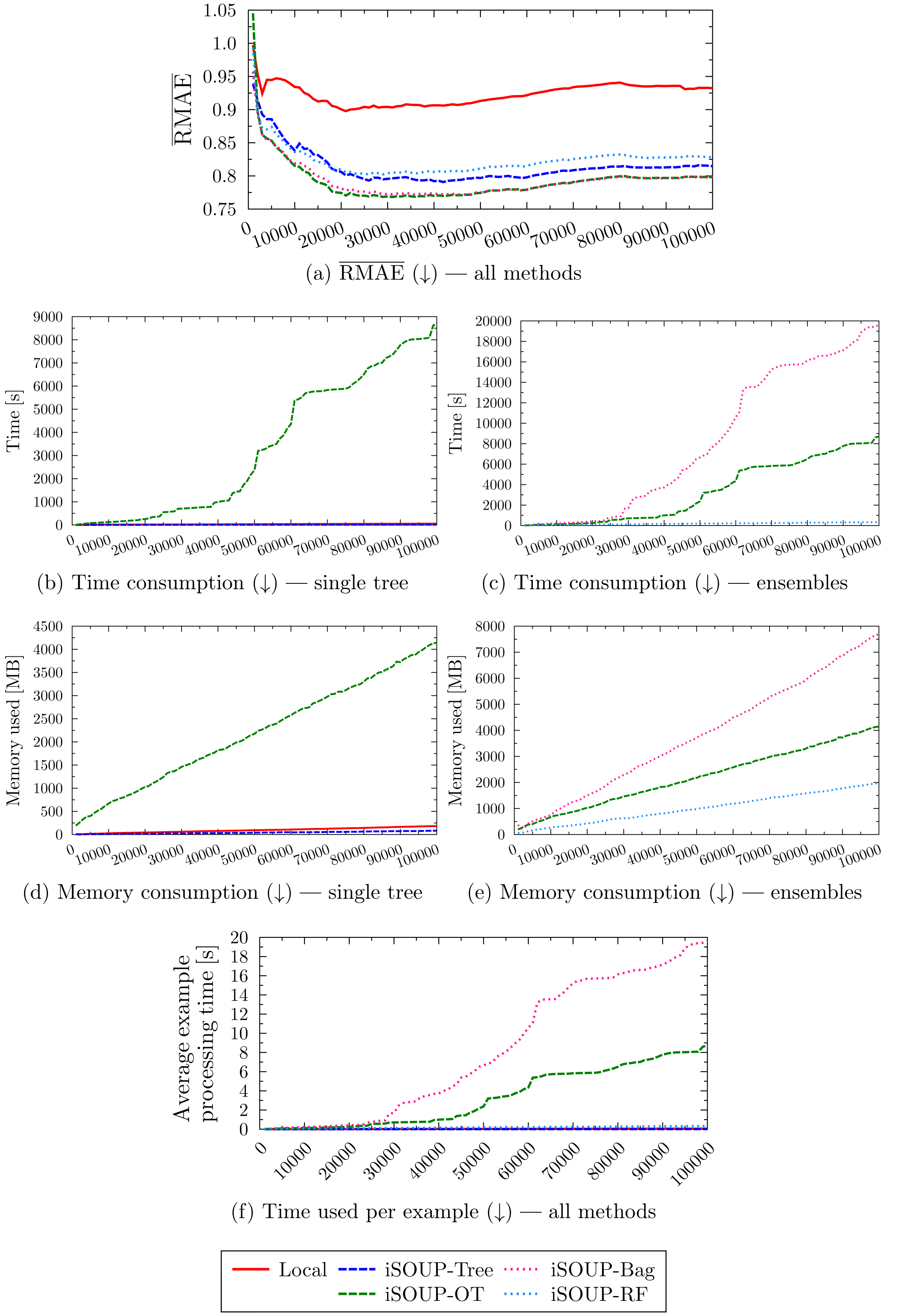
The local, FIMT-DD based, approach has the worst performance of all of the observed methods. Given the proximity of the different elements of the heating apparatus, it is reasonable to expect that the targets (electric currents/power consumption at these elements) are quite correlated, which explains the better performance of the multi-target models. Of the observed multi-target iSOUP-Tree based models, iSOUP option trees and bagging of iSOUP-Trees achieve comparable results, with iSOUP option trees notably outperforming bagging in terms of efficiency.

All of the models use on average less than a minute to process one example, even at the end of the dataset. This would theoretically allow for the models to be used in real-time to predict the electric currents within the Mars Express probe. We expect the processing time to be in a linear relation to the depth of the tree, i.e., $\mathcal{O}(\log n)$ where n is the number of processed examples. Note, however, that the results presented are for the first 100k (of the 2.6 million) examples. If the processing time were to exceed one minute, a less complex model would have to be used to reduce the processing time. There are two obvious alternatives to address this problem, i.e., to learn a less complex model like a single tree which has a very low example processing time, or to reduce the complexity of the ensemble/option tree, by discarding some members of the ensemble/options, respectively.

8.2 Predicting Photo-Voltaic Power Generation

Renewable energy has recently become an important strategic sector in local and global communities due to the need to reduce pollution. However, managing and maintaining renewable energy sources presents many challenges, such as grid integration, load balancing and energy trading. The energy output of many renewable energy sources, in particular, of photo-voltaic power-plants, varies greatly and is prone to frequent interruptions.

[1] The calculation of memory consumption takes upwards of 80% of the total experimental time.



(a) $\overline{\text{RMAE}}$ (↓) — all methods

(b) Time consumption (↓) — single tree

(c) Time consumption (↓) — ensembles

(d) Memory consumption (↓) — single tree

(e) Memory consumption (↓) — ensembles

(f) Time used per example (↓) — all methods

Figure 8.1: The experimental results of the Mars Express case study. The horizontal axes show the numbers of processed examples.

To this end predicting photo-voltaic power generation has become an urgent task for the energy sector. This task has also been addressed by utilizing machine learning and data mining techniques (Bacher, Madsen, & Nielsen, 2009; Sharma, Sharma, Irwin, & Shenoy, 2011; Rashkovska, Novljan, Smolnikar, Mohorčič, & Fortuna, 2015). In particular, Ceci, Corizzo, Fumarola, Malerba, and Rashkovska (2016) address the task of predictive modeling of photo-voltaic energy production. They use data provided by the National Renewable Energy Laboratory (NREL), which describes a collection of simulated photo-voltaic power-plants, to model photo-voltaic energy generation based on historical data, current weather, weather forecasts and power-plant location.

8.2.1 Dataset

The raw data provided by NREL contains power production forecasts of about 6,000 simulated photo-voltaic power-plants. We use the version of the dataset used by Ceci et al. (2016) as a starting point. This dataset has been narrowed down to 48 representative plants and the associated measurements and forecasts. For this version of the dataset the following 24 attributes are provided for each of the 48 plants:

- historical data on power production,
- current weather information,
- weather forecasts from numerical weather predictions (NWP) models and
- geographic coordinates of the plant.

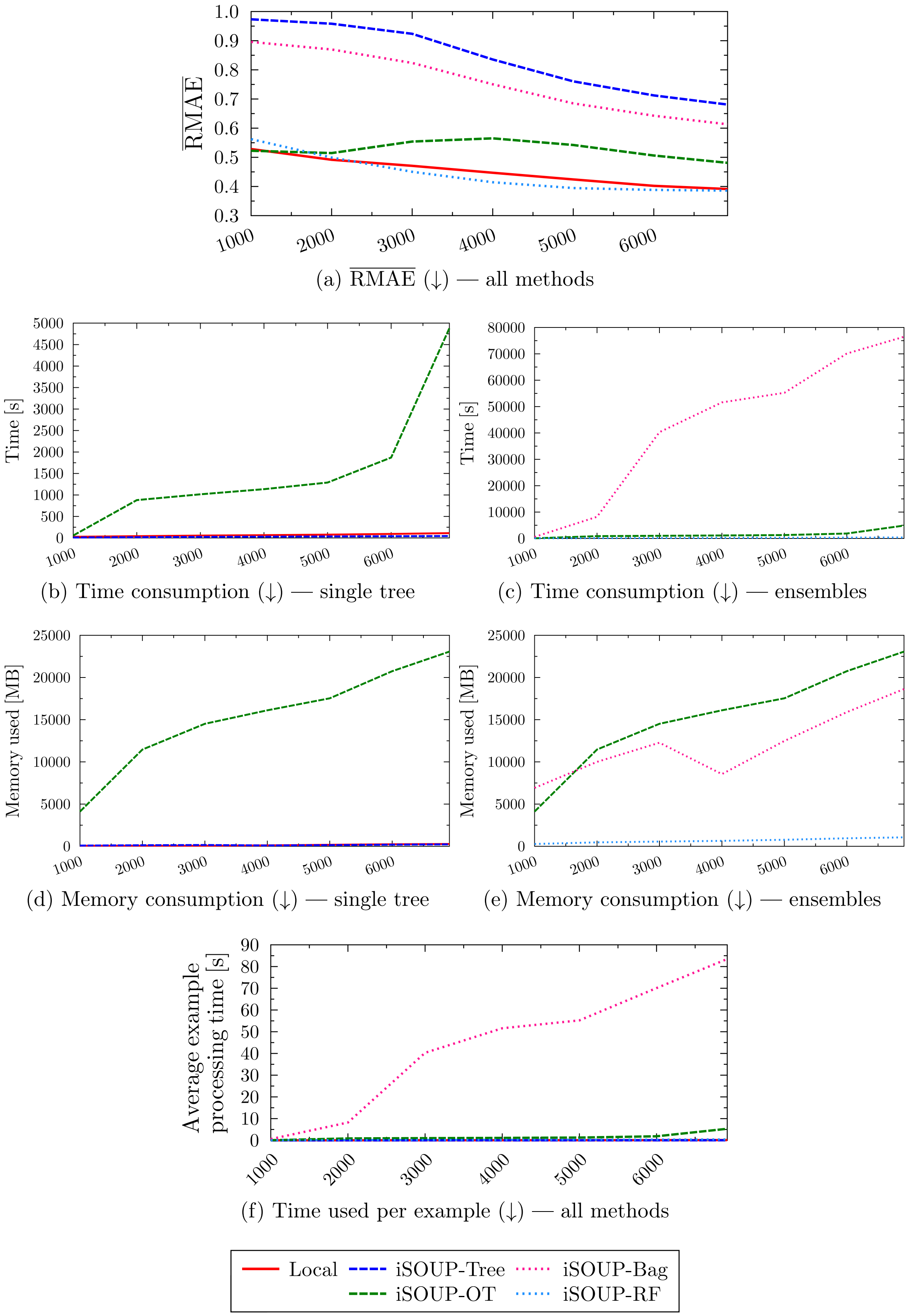
The attributes are recorded at hourly intervals, for each hour between the hours of 2:00 and 20:00.

Unlike Ceci et al. (2016), who produce predictions for each plant separately in an effort to explore the effects of spatial and temporal correlation, we join all of the data in a single time point, i.e., for a given hour. This reduces the number of examples from around 280,000 (one per time point per site) to about 7,000 (one per time point for all sites). Examples joined in this way have a large number of attributes, as well as targets. Namely, each example is composed of 1152 attributes and 48 targets.

In terms of attributes, this dataset has more than twice the number of attributes of any other dataset in our experiments. In terms of targets, its number of targets it is three times greater than the highest number of targets of the datasets used in multi-target regression experiments defined in Section 6.1.3. This means that each example contains a large quantity of information.

8.2.2 Data mining task

We use the NREL dataset as a “stress test” in terms of the size of each individual example for the methods for online multi-target regression (**Local**, **iSOUP-Tree**, **iSOUP-OT**, **iSOUP-Bag** and **iSOUP-RF**) defined in Section 4.2. We are interested in how this impacts the time and memory consumption, as well as the average example processing time of the different methods. With the predictions being made at hourly intervals, there is less pressure to achieve a low processing time. Given the relatively large distances between the different photo-voltaic power-plant sites, we also expect the targets to be less correlated than in the Mars Express case study, which should favor the local approach.



(a) $\overline{\text{RMAE}}$ (↓) — all methods

(b) Time consumption (↓) — single tree

(c) Time consumption (↓) — ensembles

(d) Memory consumption (↓) — single tree

(e) Memory consumption (↓) — ensembles

(f) Time used per example (↓) — all methods

Figure 8.2: The experimental results of the photo-voltaic power generation case study. The horizontal axes show the numbers of processed examples.

8.2.3 Results and discussion

The results for the NREL case study are presented in Figure 8.2. As expected, the local approach performs very well (i.e., the best) in this setting. Random forests of iSOUP-Trees achieve a comparable performance, which is likely the result of the large number of attributes. Option trees are better than bagging, while the single iSOUP-Tree is worse than both of these.

The local approach, as well as the iSOUP-Tree and random forest of iSOUP-Trees process the dataset rather quickly, unlike option trees which require around 90 minutes, while bagging takes more than 22 hours to process the whole dataset. Similarly, option trees and bagging use considerably larger amounts of memory, reaching more than 20 GB of memory use at the end of the dataset.

Since the predictions are made at hourly intervals and all models use far less than an hour to process an example, the models could also be used alongside a human domain expert, which could analyze the models' predictions and potentially improve the predictive performance. In a purely automated setting, it would be advisable to use either the local approach or the random forest of iSOUP-Trees, given their better performance, their higher speed and their lower memory usage.

Chapter 9

Conclusions and Further Work

> There is no real ending. It's just the place where you stop the story.
>
> — Frank Herbert

In this thesis, we deal with structured output prediction on data streams. We introduced methods that can address online multi-target regression, online multi-label classification, online hierarchical multi-target regression and online hierarchical multi-label classification. We also showed how these methods can be adapted and extended to allow for learning from unlabeled examples in the context of semi-supervised learning tasks. Furthermore, we addressed the online feature ranking task for structured output prediction.

In our experimental evaluation, we showed that all of the introduced methods have merit. In particular, bagging of iSOUP-Trees proved to be the method of choice in terms of predictive performance. The methods for online hierarchical prediction, online semi-supervised learning and online feature ranking that were adapted from a batch scenario also show promising results, but require some additional experimental evaluation as the experiments presented in this thesis were initial efforts. Finally, we also showed how our methods can be applied to the tasks of predicting the electric power consumption of the Mars Express probe and for predicting photo-voltaic power generation.

In the remainder of this chapter, we first summarize the scientific contributions of this thesis to science, in particular to the fields of machine learning and data (stream) mining. We follow up with a discussion of the goals and hypotheses outlined in Chapter 1 in the context of our contributions and results. We conclude with an outline of several avenues for further work.

9.1 Contributions to Science

In this thesis, we have contributed to the field of data stream mining a family of online methods for structured output prediction and performed an experimental evaluation of the proposed methods, as well as applied the developed methods in two case studies. First, we introduced the iSOUP-Tree family of methods for online multi-target regression (Osojnik et al., 2017b). Next, we introduced the MLC via MTR problem transformation methodology and shown that using it in conjunction with the iSOUP-Tree methods yields results comparable to the state-of-the-art methods (Osojnik et al., 2017a). Furthermore, we have introduced methods for online hierarchical prediction tasks, which had not been addressed in the online learning setting prior to this thesis. In addition, we have also adapted the predictive clustering framework to the online setting and used online predictive clustering trees to address online semi-supervised learning. Finally, we have adapted the symbolic random forest-based method for feature ranking to the online setting, which allows us to perform online feature ranking on any of the tasks addressed in this thesis.

In the following sections, we first summarize our contributions that relate to the introduction of novel methods for online structured output prediction tasks. We continue by providing an overview of the experimental results. Finally, we present the results of the two case studies.

9.1.1 Methods for structured output prediction on data streams

The **methods for online multi-target regression** are based on the online single-target regression method FIMT-DD (Ikonomovska, Gama, & Džeroski, 2011b) and its preliminary multi-target extension FIMT-MT (Ikonomovska, Gama, & Džeroski, 2011a). The iSOUP-Tree method that we introduce improves upon these methods by allowing for nominal input attributes and by introducing better models in the leaves. In addition, we extend the iSOUP-Tree method with the capability to introduce option nodes, which address the myopia of the tree induction procedure.

The **multi-label classification via multi-target regression problem transformation methodology** allows us to transform an online multi-label classification task into an online multi-target regression task. After we apply an online multi-target regression model to obtain numeric predictions, we transform them back into multi-label classification predictions. We have combined the MLC via MTR methodology with the above methods for online multi-target regression to obtain methods for online multi-label classification.

To **address hierarchical prediction tasks**, in particular, hierarchical multi-target regression and hierarchical multi-label classification, we have adapted the approaches of Vens et al. (2008) and Mileski et al. (2017) to the online setting. We appropriately modify the tree splitting heuristic. The adaptation towards HMTR is straightforward, while the adaptation towards HMLC combines HMTR with a hierarchical variant of the MLC via MTR methodology.

Prior to our work, the predictive clustering framework (Blockeel & De Raedt, 1998) had not been used in an online learning setting. We have introduced a predictive clustering variant of iSOUP-Tree, called iSOUP-PCT. In addition to modifying the splitting heuristic, we have expanded the data structure that stores the required statistics for the calculation of the heuristic. We use this method to **address online semi-supervised learning** in a similar fashion to the approach of Levatić et al. (2017b) taken in the batch setting. Furthermore, we have adapted the initialization of the leaf model upon splitting to utilize as much information from the labeled examples as possible.

We have also **adapted the symbolic random forest method for feature ranking** introduced by Petković et al. (2017), which can be used to address feature ranking in the batch setting for different types of structured outputs. Instead of random forests of PCTs, we use random forests of randomized iSOUP-Trees to determine the scores of the individual attributes. Each occurrence of an attribute in a split node in a tree in the random forest increases the attribute's score proportionally to the distance of the split node from the root of the tree.

9.1.2 Experimental evaluation of methods for structured output prediction on data streams

We have performed an **experimental evaluation of the introduced methods** using evaluation methodologies that are appropriate for the online learning setting and evaluation measures appropriate for the considered data mining tasks. In our comparison of multi-target regression methods, we have found that bagging ensembles of iSOUP-Trees exhibit the best predictive performance, followed by random forests of iSOUP-Trees. However, these ensemble methods consume considerably more computational resources than single-tree methods. To address this issue, we have investigated the trade-off between predictive performance and use of computational resources.

In the comparison of online multi-label classification methods, we have compared single iSOUP-Trees and bagging of iSOUP-Trees in conjunction with the MLC via MTR methodology and the state-of-the-art method for online multi-label classification proposed by Read et al. (2012). While many of the experimental results did not yield clear-cut conclusions, bagging of iSOUP-Trees performed best on the threshold independent ranking-based measures. This implies that with a better threshold selection procedure, the predictive performance could be further increased also for the other evaluation measures.

When evaluating the methods for hierarchical tasks, we have looked at how two variants of the hierarchical iSOUP-Tree compare to a non-hierarchical iSOUP-Tree that predicts only the leaves. Within the splitting heuristic, the bottom-weighted hierarchical variant puts a larger emphasis on the targets/labels lower in the hierarchy, while the top-weighted variant does the opposite, i.e., it places larger emphasis on targets closer to the root of the hierarchy. We have explored whether the addition of the hierarchy improves the predictive performance over considering just the leaves. In the case of HMTR, the bottom-weighted method performed similarly to the non-hierarchical method, improving it slightly overall. The top-weighted method performed worse than both of the other methods. For the evaluation on HMLC data sets, the results are not as clear-cut as for HMTR. On some measures, the bottom-weighted iSOUP-Tree outperformed the other two methods, while on other the top-weighted method performed better.

The evaluation results of iSOUP-PCTs for online semi-supervised learning gave comparatively straightforward conclusions. In almost all cases, the usage of additional unlabeled examples for learning propelled the semi-supervised iSOUP-PCTs to a better performance as compared to the baseline supervised methods. The increase in performance was more pronounced when the ratio of labeled to unlabeled examples was lower.

Finally, we evaluated the online symbolic random forest-based feature ranking method by comparing the rankings it produced to the rankings produced by the corresponding batch learning method. While the produced rankings did not agree on the ranking completely, similar attributes were ranked close to the top. Additionally, we note that even though the rankings were different, this does not necessarily point to bad performance, but rather to the rankings capturing different aspects of the learning problem. While the batch feature ranking measures the importances of attributes over the entire dataset, the online feature ranking measures the importance of attributes “locally,” i.e., attributes which are important at a given time point in the data stream.

9.1.3 Case studies

In addition to the experimental evaluation on benchmark datasets we **conducted two case studies** in practical application domains. The first deals with the prediction of electric power consumption for the Mars Express satellite, while the second deals with forecasting the power generation for photo-voltaic power-plants. In these case studies, we have shown that the introduced methods are appropriate for practical use in these domains.

In the Mars Express case study, we have explored how the introduced methods for online multi-target regression could be used for predicting the electrical power consumption of the autonomous thermal subsystem of the Mars Express space probe. We have shown that even after learning from a large amount of data examples, the methods are still able to learn and make predictions in the desired time frames.

In the second case study, we have explored how well the methods for online multi-target regression cope with tasks that have a large number of input attributes and targets. We learned to predict the energy generated by photo-voltaic power-plants based on historic information about power production, current weather conditions, weather forecasts and the power-plants' locations. The methods processed the data quickly enough even in the presence of large numbers of input attributes and targets.

9.2 Discussion

Let us now consider whether we have achieved the goals of the thesis and provided evidence for our hypotheses. The primary goal of the thesis was to introduce versatile methods for online structured output prediction that can be applied to a variety of online structured output prediction tasks. As we have implemented a family of methods that address online multi-target regression, online multi-label classification, online hierarchical multi-target regression and online multi-label classification, the primary goal has been thus achieved. Consequently, the subordinate goals that deal individually with the design and implementation of the methods for particular online tasks were also achieved, i.e., the second goal for multi-target regression, the third for multi-label classification and the fourth for hierarchical prediction tasks.

The fourth goal, which is concerned with the design and implementation of methods for online semi-supervised learning for structured output prediction tasks, has also been achieved. While we have applied iSOUP-PCT only to online multi-target regression, it can also be easily applied to the other online semi-supervised SOP tasks. We must, however, be aware of the fact that this method incurs considerably higher use of computational resources than the other introduced methods.

While we have designed and implemented a method for online feature ranking for SOP tasks, the experimental evaluation has not led to conclusive results. Further experimental examination is required in order to have stronger conclusions regarding this task, thus, we consider the sixth goal has been only partially achieved.

With regard to the final goal that considers the experimental evaluation of the introduced methods, we conclude that it has been achieved. The introduced methods were compared on a wide selection of experimental questions, appropriate available datasets and evaluation measures. While further experimental evaluation is always desirable, in particular, evaluation of methods adapted from the batch setting, the experimental evaluation in this thesis is extensive enough. In addition to the experimental evaluation, we also conducted two case studies which showcase how the introduced methods might be used in actual application domains and scenarios.

With respect to the first three hypotheses introduced in Chapter 1, we were able to adapt online tree-based methods for MTR to other types of SOP (MLC, HMTR, HMLC). We were also able to achieve good experimental results with the transformation of the various structured output prediction tasks to the task of online multi-target regression. In particular, we were able to transform multi-label classification into multi-target regression and hierarchical multi-label classification into hierarchical multi-target regression.

Concerning the last two hypotheses, the adaptations of batch approaches for semi-supervised learning and feature ranking were also fairly successful. The adaptation of the semi-supervised predictive clustering trees was especially successful, with the obvious caveat of additional consumption of computational resources. The success of the adaptation of the feature ranking method was less clear at least partially due to the lack of well defined success criteria and standard evaluation methodology for feature ranking. However, given that there are no other methods we can compare to, as ours is the only feature ranking method that can address structured outputs in an online setting (to the best of our knowledge), we can still consider the adaptation to be successful.

9.3 Further Work

We organize the further work into three categories: methodological further work, task-specific further work and further work on applications.

Methodological further work. The most obvious avenue for further work we plan to address is the adaption of change detection mechanisms for online-multi target regression. Here we see several starting points. The first is to adapt single-target change detectors, such as the Page-Hinckley test (Mouss et al., 2004) or the adaptive windowing approach (Bifet & Gavaldà, 2009) to online multi-target regression. Here, the naïve approach of averaging the error signals over the targets is not the best starting point, but, another starting point is not immediately evident and would need to be identified. A second approach is to adapt change detectors for online multi-label classification from the works of Spyromitros-Xioufis (2011) or Shi, Wen, et al. (2014) to the online multi-regression tasks. These methods are, however, highly specific to the task of online multi-label classification, and, hence, adapting them to other tasks might require considerable effort. A third possible approach in change detection for online multi-target regression might be to combine the targets into groups which behave similarly, then use one change detector for each group of targets.

We will also modify the E-BST structure that maintains the statistics in iSOUP-Tree leaves to utilize a self-balancing tree algorithm, such as AVL trees or red-black trees (Sedgewick & Wayne, 2011). In streaming applications, in particular, we often encounter attributes which are monotonous in regard to time. This means that regular binary search trees that observe these attributes are severely unbalanced. However, we must be particularly aware that we are only storing partial statistics that we must properly update when performing the rotations which balance the tree.

The experiments we performed for the evaluation of online hierarchical prediction, online semi-supervised multi-target regression and online feature ranking were all initial efforts. We will perform more rigorous experiments that further evaluate the introduced methods. In particular, we will compare them with state-of-the-art methods appropriate for the tasks, where such methods exist.

Another avenue for further work that we will clearly follow is combining several of the introduced methods, e.g., addressing online semi-supervised multi-label classification or online feature ranking for hierarchical multi-target regression. This is particularly straightforward as the introduced methods are generally compatible with one another. The most care probably needs to be taken with the combination of the adaptations for hierarchical prediction tasks and semi-supervised learning, as both of these adaptations modify the tree splitting heuristic.

While we have addressed several online structured output prediction tasks in the scope of this thesis, there are other SOP tasks that have not been considered in the online setting. We will consider further extending the iSOUP-Tree family of methods to the tasks of, e.g., time-series prediction (Slavkov & Džeroski, 2010) or sequence labeling (Brefeld et al., 2005).

Task-specific further work. We have only observed how well the single iSOUP-Tree methods and bagging methods perform in the multi-label classification via multi-target regression scenario. Additionally, we plan to apply other methods for online multi-target regression to the task of online multi-label classification via online multi-target regression. In particular, we will consider applying iSOUP-OptionTrees and random forests of iSOUP-Trees.

In using the introduced methods for multi-label classification, we have stated earlier in the thesis that a better choice of the classification threshold would yield better performance. To this end, we plan to explore methods for automatic threshold selection that can be applied in conjunction with the introduced methods for MLC via MTR.

Furthermore, we will consider the selection of the initial weight of the root of the hierarchy in hierarchical prediction tasks in greater detail. While this parameter can be optimized as part of the learning procedure in the batch learning scenario, this approach can not be directly adapted to the online learning setting. In addition, in this thesis we have only considered four hierarchical datasets, two for hierarchical multi-target regression and two for hierarchical multi-label classification. Further extending the experimental evaluation to other datasets, and in particular to datasets for hierarchical multi-label classifications generated in a streaming setting, is paramount for further development of the hierarchical methods. In all of the observed hierarchical datasets, the hierarchy at hand was a tree. We will further examine how the proposed methods perform on datasets where the hierarchy is a DAG.

Similarly, selecting the appropriate level of supervision in semi-supervised learning is also done automatically in the batch scenario through the use of an internal cross-validation procedure which selects the level of supervision from a predetermined set of potential values. This process occurs before the learning commences and is as such not applicable to the online scenario. Ideally, the selection procedure, which would determine the level of supervision, would run in parallel to the model and would also be adaptable to any potential changes in the concept. We plan to explore and implement potential mechanisms that could learn (and automatically adjust) the level of supervision in this scenario.

Furthermore, we intend to design/consider simple models that can utilize unlabeled examples to learn. This would greatly benefit the semi-supervised methods, as these kinds of models could be used as leaf models in model trees. Neither the mean regressor, nor the (multi-target) perceptron, nor the adaptive model are able to use the unlabeled examples in any way.

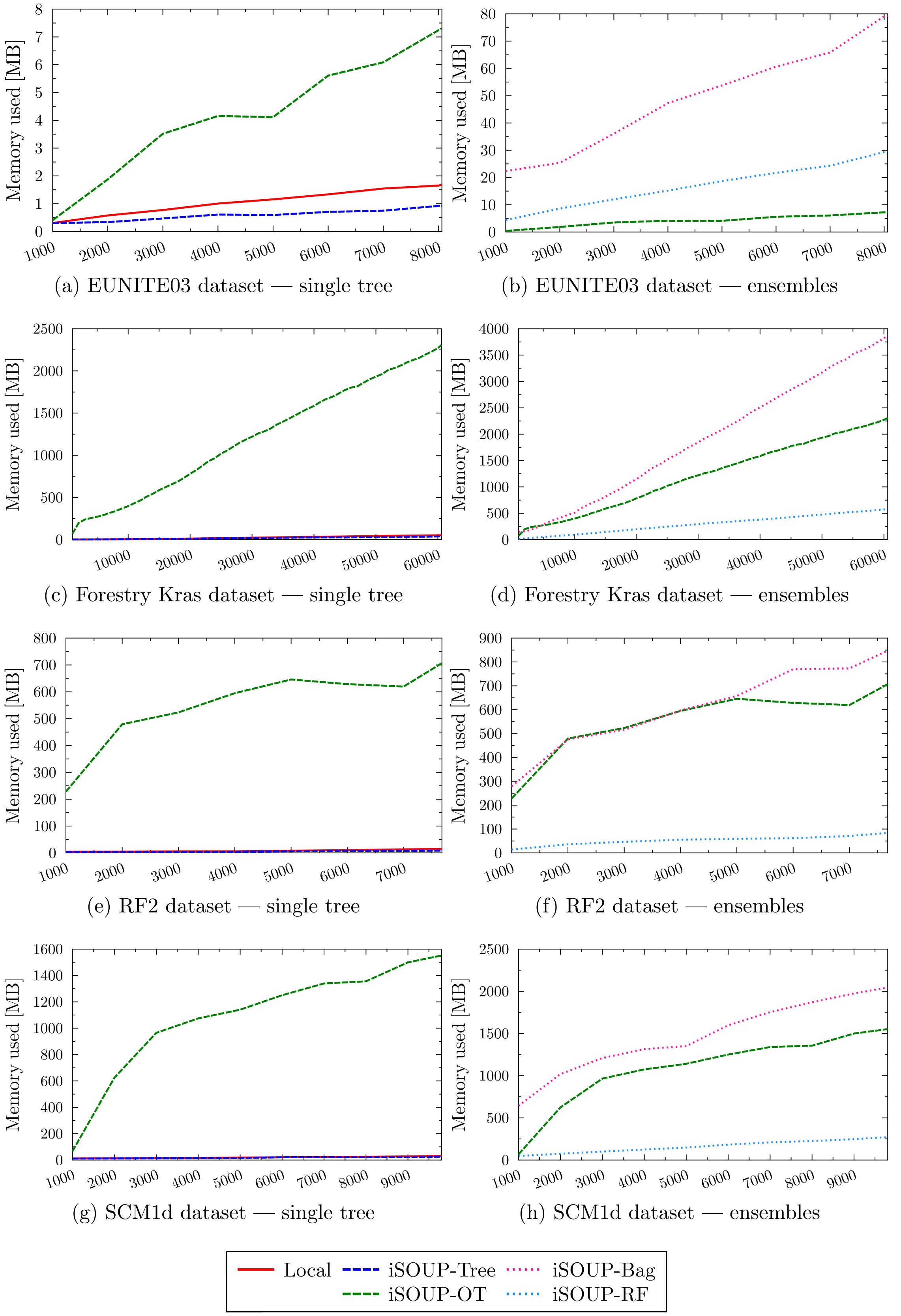
Further work in applications. While we have shown two case studies in the online multi-target regression scenario, the introduced methods can be applied to a variety of online structured output prediction tasks. In particular, we plan to apply our methods to tasks of online analysis of text and multimedia data, such as sentiment detection, automatic annotation of text, video or audio. Other tasks we plant to consider include fault forecasting and detection, service availability forecasting, fraud detection and others.

Appendix A

Additional Plots

This appendix holds additional plots that were omitted for brevity in the main body of the thesis.

A.1 Additional Plots for the Efficiency Evaluation of Multi-Target Regression Methods



(a) EUNITE03 dataset — single tree

(b) EUNITE03 dataset — ensembles

(c) Forestry Kras dataset — single tree

(d) Forestry Kras dataset — ensembles

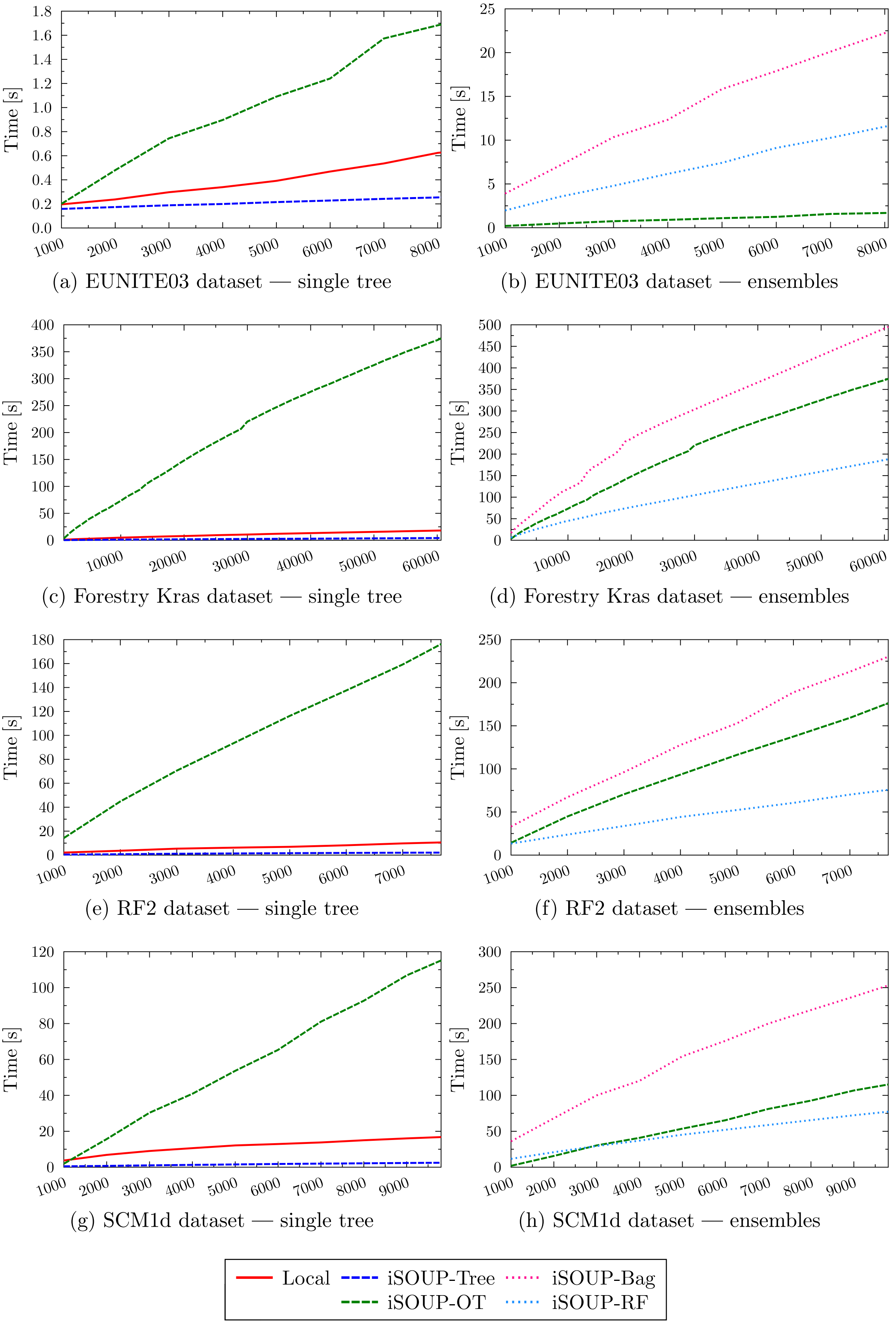
(e) RF2 dataset — single tree

(f) RF2 dataset — ensembles

(g) SCM1d dataset — single tree

(h) SCM1d dataset — ensembles

Figure A.1: Additional results in terms of the memory consumption (↓) of the observed methods. Horizontal axes show the numbers of processed examples.



(a) EUNITE03 dataset — single tree

(b) EUNITE03 dataset — ensembles

(c) Forestry Kras dataset — single tree

(d) Forestry Kras dataset — ensembles

(e) RF2 dataset — single tree

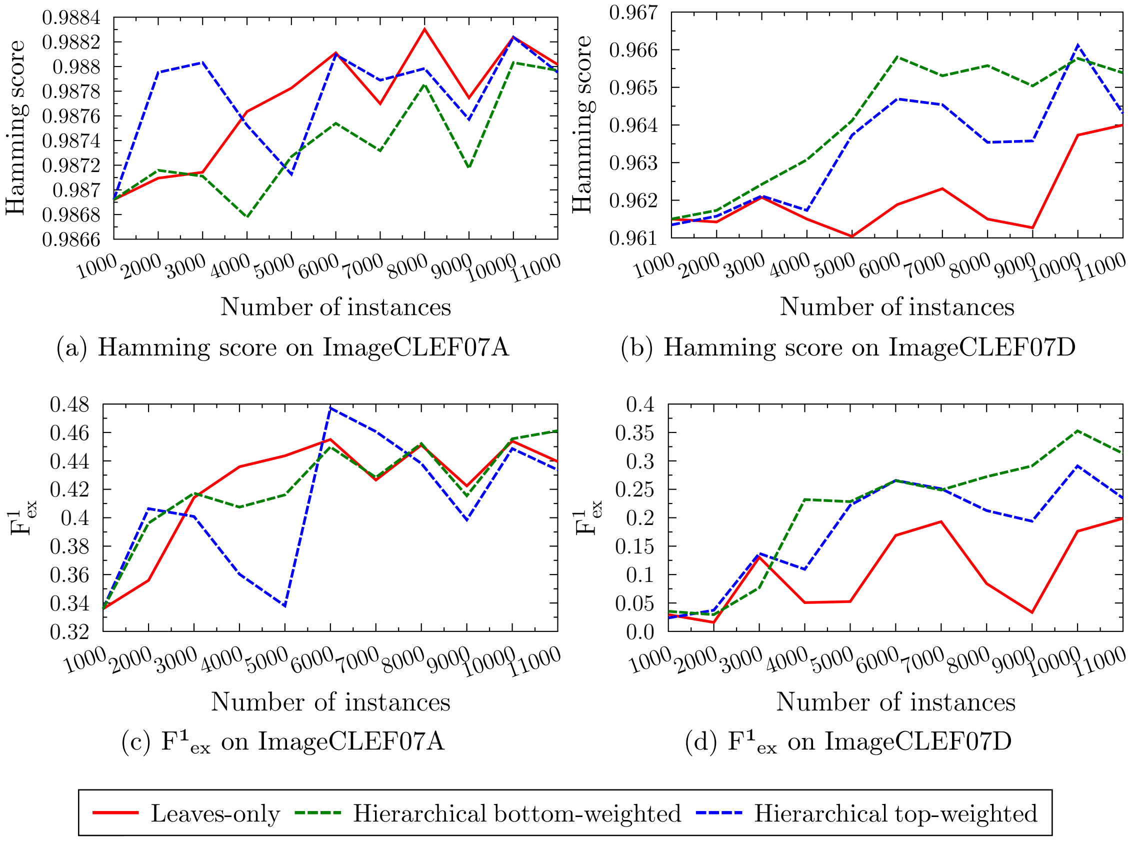
(f) RF2 dataset — ensembles

(g) SCM1d dataset — single tree

(h) SCM1d dataset — ensembles

Figure A.2: Additional results in terms of the time consumption (↓) of the observed methods. Horizontal axes show the numbers of processed examples.

A.2 Additional Plots for Hierarchical Multi-Label Classification Experiments



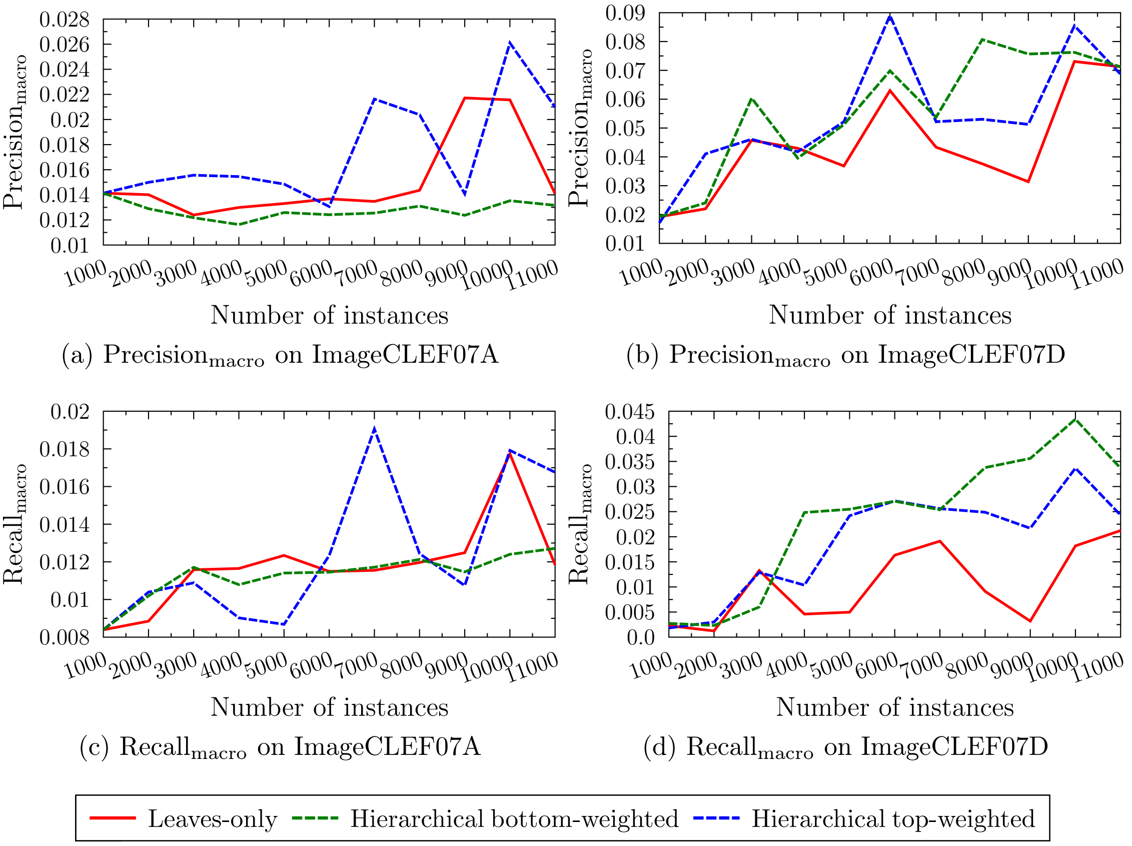
(a) Hamming score on ImageCLEF07A

(b) Hamming score on ImageCLEF07D

(c) $F^1{}_{ex}$ on ImageCLEF07A

(d) $F^1{}_{ex}$ on ImageCLEF07D

Figure A.3: Additional results on example-based measures (↑) for the hierarchical multi-label classification datasets.



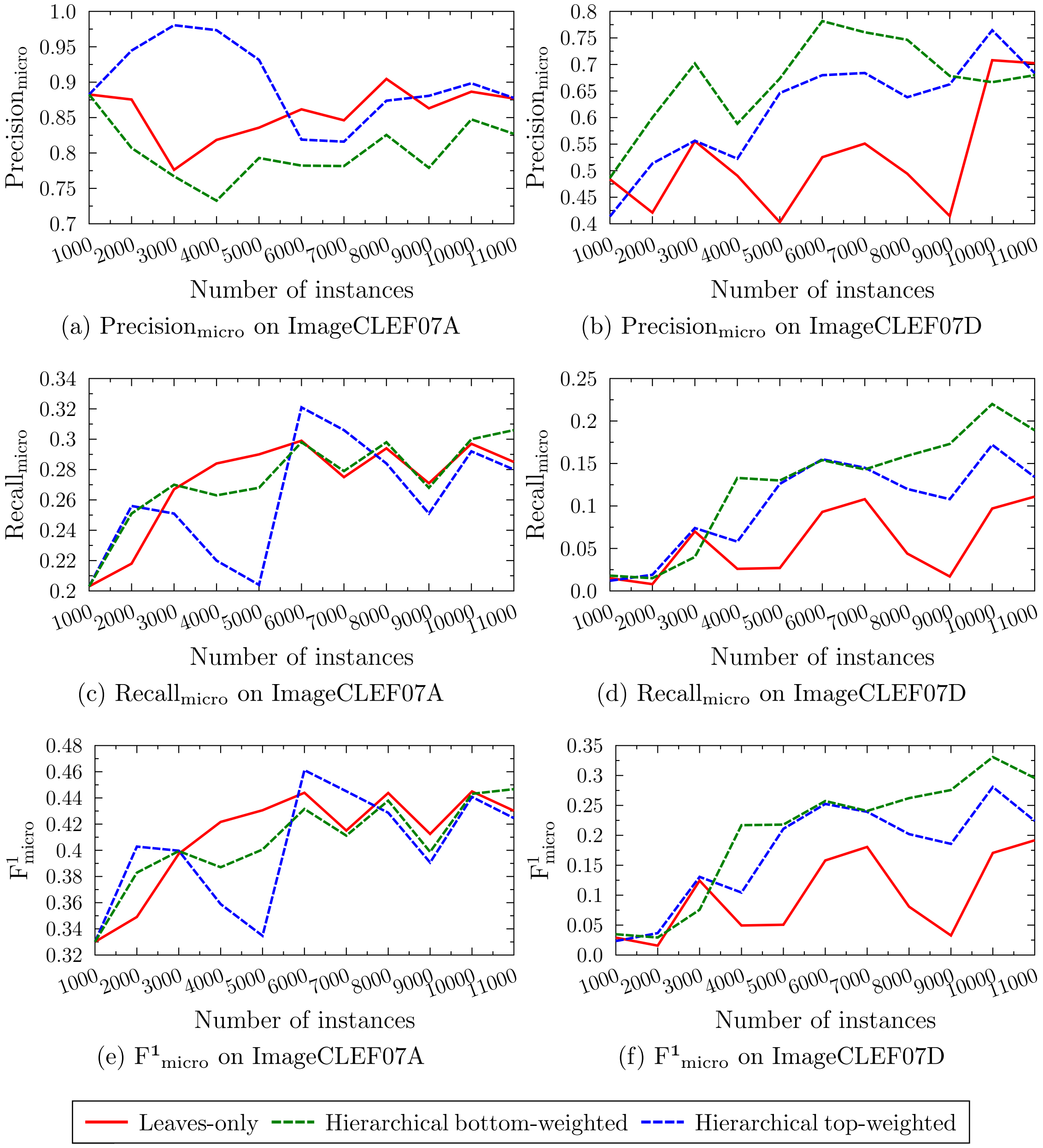
(a) Precision$_{\text{macro}}$ on ImageCLEF07A

(b) Precision$_{\text{macro}}$ on ImageCLEF07D

(c) Recall$_{\text{macro}}$ on ImageCLEF07A

(d) Recall$_{\text{macro}}$ on ImageCLEF07D

Figure A.4: Additional results on macro-averaged label-based measures (↑) for the hierarchical multi-label classification datasets.



(a) Precision$_{\text{micro}}$ on ImageCLEF07A

(b) Precision$_{\text{micro}}$ on ImageCLEF07D

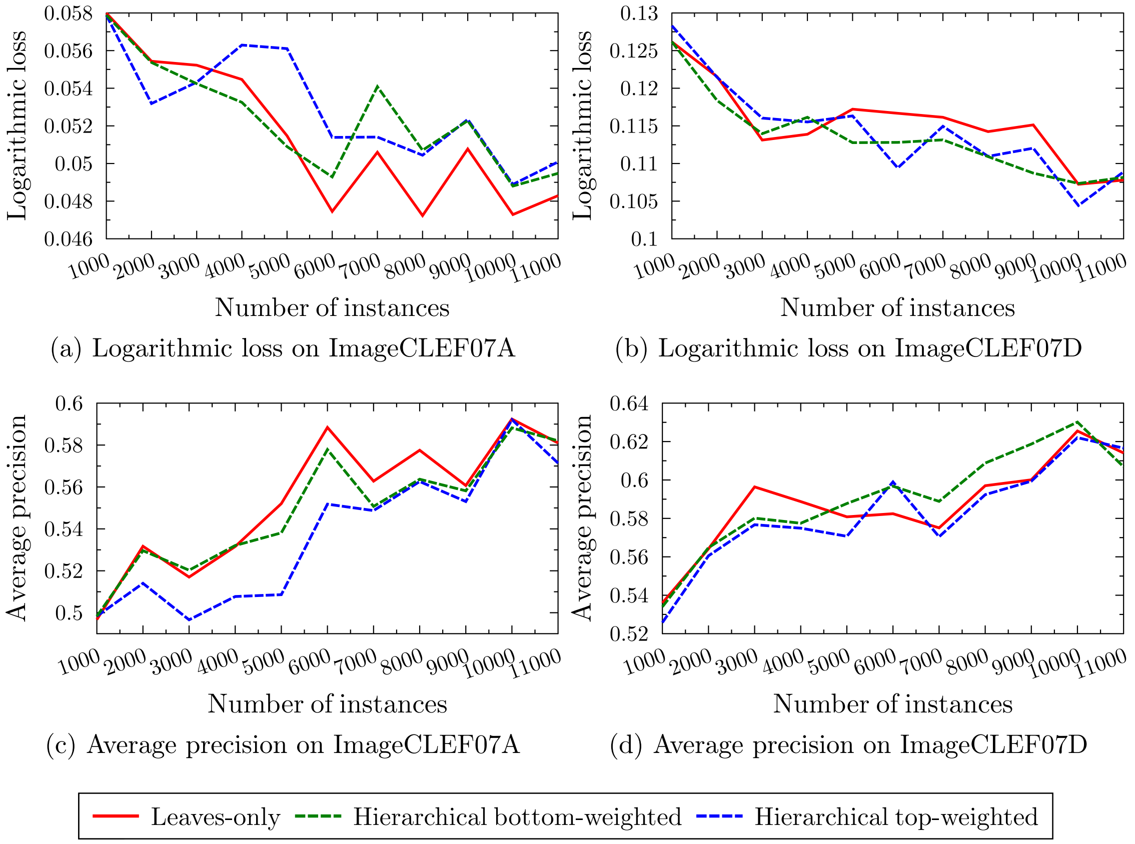
(c) Recall$_{\text{micro}}$ on ImageCLEF07A

(d) Recall$_{\text{micro}}$ on ImageCLEF07D

(e) F$^{1}_{\text{micro}}$ on ImageCLEF07A

(f) F$^{1}_{\text{micro}}$ on ImageCLEF07D

Figure A.5: Additional results on micro-averaged label-based measures (↑) for the hierarchical multi-label classification datasets.



(a) Logarithmic loss on ImageCLEF07A

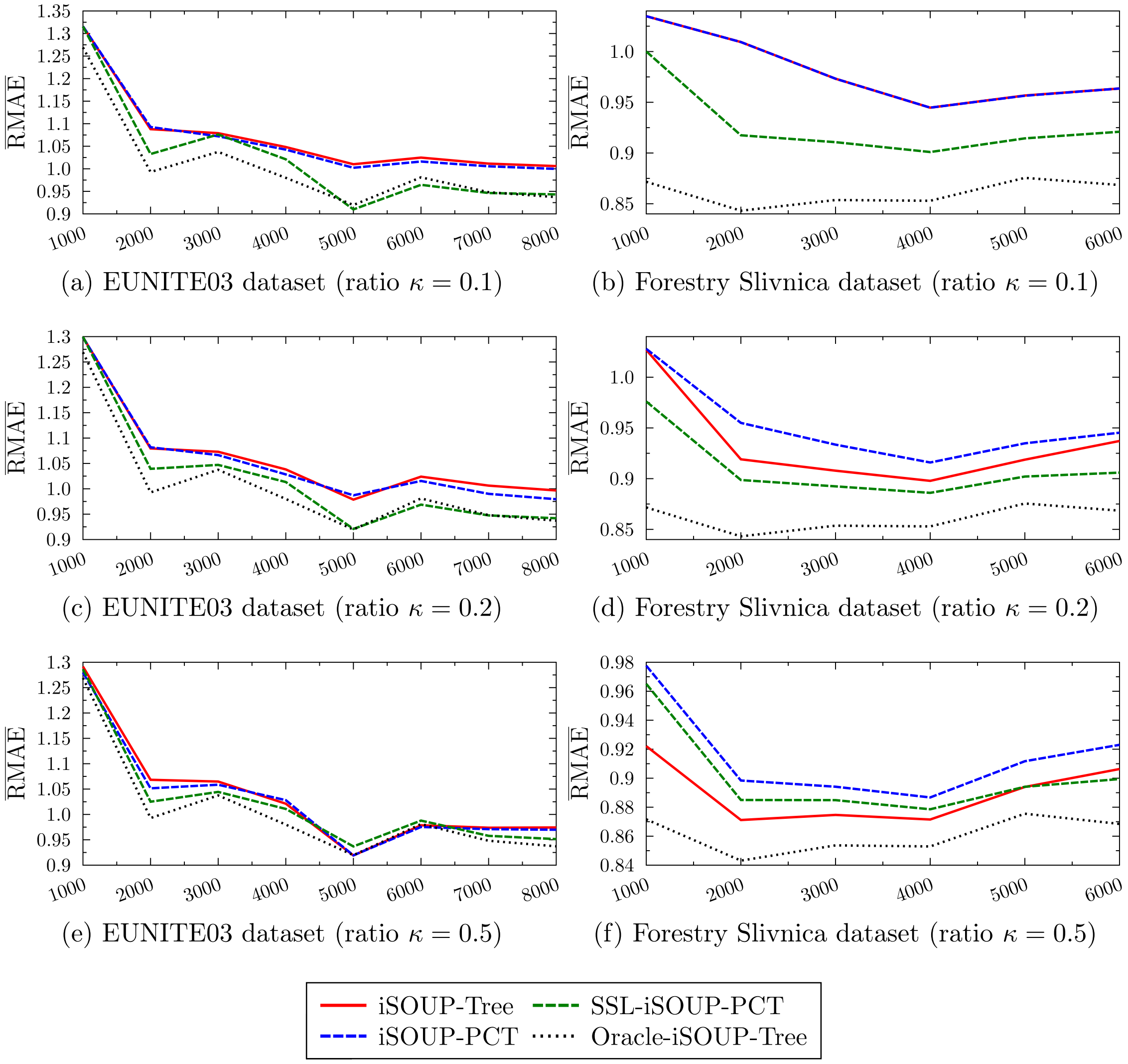
(b) Logarithmic loss on ImageCLEF07D

(c) Average precision on ImageCLEF07A

(d) Average precision on ImageCLEF07D

Figure A.6: Additional results on ranking-based measures (logarithmic loss ↓, average precision ↑) for the hierarchical multi-label classification datasets.

A.3 Additional Plots for Semi-Supervised Multi-Target Regression Experiments



(a) EUNITE03 dataset (ratio $\kappa = 0.1$)

(b) Forestry Slivnica dataset (ratio $\kappa = 0.1$)

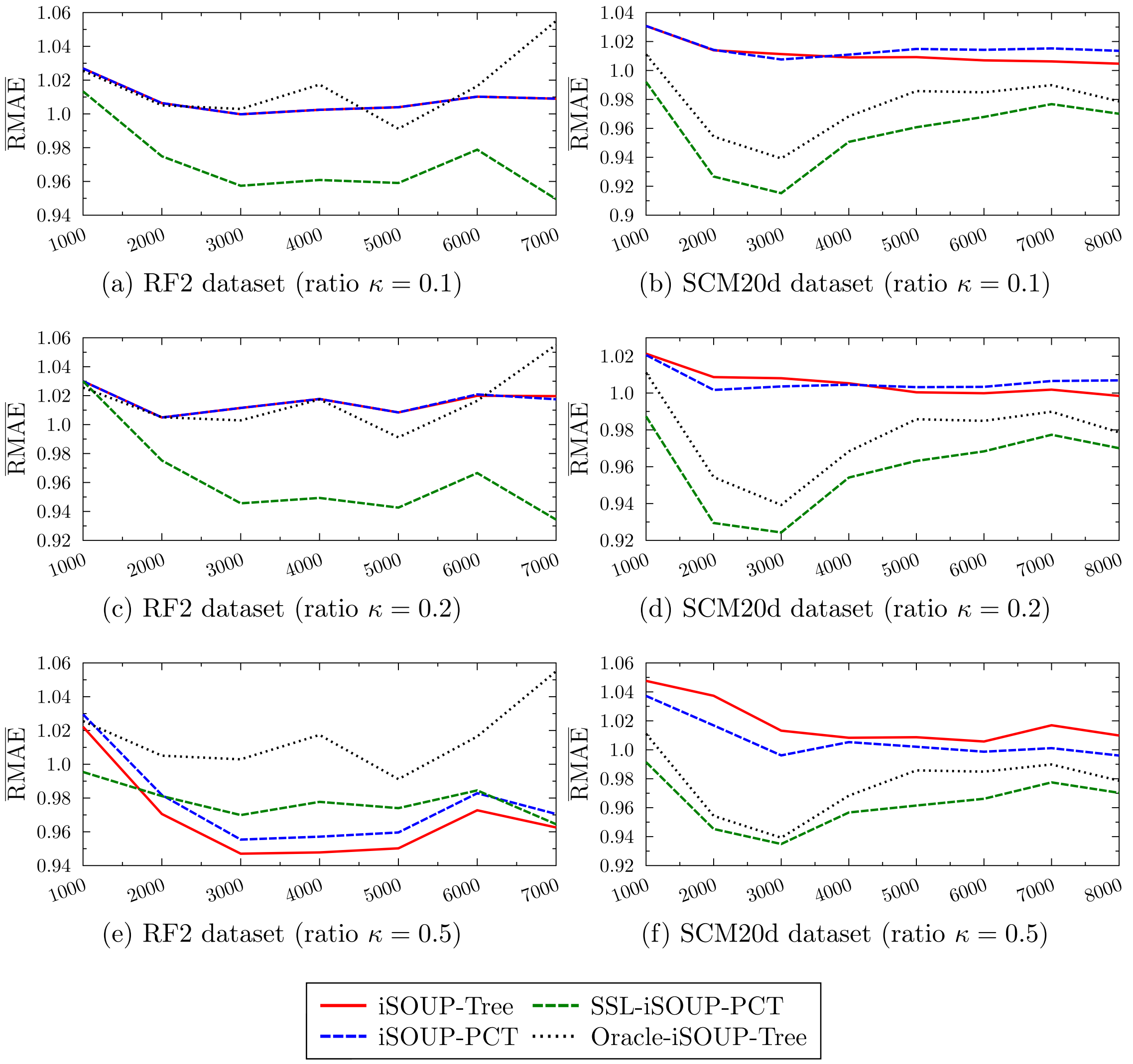
(c) EUNITE03 dataset (ratio $\kappa = 0.2$)

(d) Forestry Slivnica dataset (ratio $\kappa = 0.2$)

(e) EUNITE03 dataset (ratio $\kappa = 0.5$)

(f) Forestry Slivnica dataset (ratio $\kappa = 0.5$)

Figure A.7: The results in terms of $\overline{\text{RMAE}}$ (↓) on the EUNITE03 and Forestry Slivnica datasets in the semi-supervised scenario.



(a) RF2 dataset (ratio $\kappa = 0.1$)

(b) SCM20d dataset (ratio $\kappa = 0.1$)

(c) RF2 dataset (ratio $\kappa = 0.2$)

(d) SCM20d dataset (ratio $\kappa = 0.2$)

(e) RF2 dataset (ratio $\kappa = 0.5$)

(f) SCM20d dataset (ratio $\kappa = 0.5$)

Figure A.8: The results in terms of $\overline{\text{RMAE}}$ (↓) on the RF2 and SCM20d datasets in the semi-supervised scenario.

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. Springer. doi:10.1007/978-3-642-40991-2_21

Aggarwal, C. C. (Ed.). (2007). *Data streams: Models and algorithms*. Springer. Advances in Database Systems. doi:10.1007/978-0-387-47534-9

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, *22*(2), 207–216. doi:10.1145/170036.170072

Aho, T., Ženko, B., & Džeroski, S. (2009). Rule ensembles for multi-target regression. In *Proceedings of the Ninth IEEE International Conference on Data Mining (ICDM 2009)* (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

Alaydie, N., Reddy, C. K., & Fotouhi, F. (2012). Exploiting label dependency for hierarchical multi-label classification. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2012)* (Vol. 7301, pp. 294–305). LNCS. Springer. doi:10.1007/978-3-642-30217-6_25

Altun, Y., McAllester, D., & Belkin, M. (2006). Maximum margin semi-supervised learning for structured variables. In *Advances in Neural Information Processing Systems 18 (NIPS 2005)* (pp. 33–40). NIPS Foundation.

Anderson, T. W. (1951). Estimating linear restrictions on regression coefficients for multivariate normal distributions. *Annals of Mathematical Statistics*, 327–351. doi:10.1214/aoms/1177729580

Anderson, T. W. & Rubin, H. (1949). Estimation of the parameters of a single equation in a complete system of stochastic equations. *Annals of Mathematical Statistics*, 46–63. doi:10.1214/aoms/1177730090

Angelov, P., Filev, D. P., & Kasabov, N. (2010). *Evolving intelligent systems: Methodology and applications*. John Wiley & Sons.

Angelov, P., Lughofer, E., & Zhou, X. (2008). Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets and Systems*, *159*(23), 3160–3182. doi:10.1016/j.fss.2008.06.019

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*(4), 319–342. doi:10.1023/a:1022821128753

Appice, A. & Džeroski, S. (2007). Stepwise induction of multi-target model trees. In *Machine Learning: ECML 2007* (Vol. 4701, pp. 502–509). LNCS. Springer. doi:10.1007/978-3-540-74958-5_46

Bacher, P., Madsen, H., & Nielsen, H. A. (2009). Online short-term solar power forecasting. *Solar Energy*, *83*(10), 1772–1783. doi:10.1016/j.solener.2009.05.016

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method. In *Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams (IWKDDS 2006)* (pp. 77–86).

Bakir, G. (2007). *Predicting structured data*. MIT Press.

Barros, R. C., Cerri, R., Freitas, A. A., & de Carvalho, A. C. P. L. F. (2013). Probabilistic clustering for hierarchical multi-label classification of protein functions. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2013)* (Vol. 8189, pp. 385–400). LNCS. Springer. doi:10.1007/978-3-642-40991-2_25

Barutcuoglu, Z., Schapire, R. E., & Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, *22*(7), 830–836. doi:10.1093/bioinformatics/btk048

Basseville, M. & Nikiforov, I. V. (1993). *Detection of abrupt changes: Theory and application*. Prentice Hall.

Bi, W. & Kwok, J. T. (2015). Bayes-optimal hierarchical multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, *27*(11), 2907–2918. doi:10.1109/tkde.2015.2441707

Bifet, A. & Gavaldà, R. (2009). Adaptive learning from evolving data streams. In *Advances in Intelligent Data Analysis VIII (IDA 2009)* (Vol. 5772, pp. 249–260). LNCS. doi:10.1007/978-3-642-03915-7_22

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive online analysis. *Journal of Machine Learning Research*, *11*(May), 1601–1604.

Bifet, A., Holmes, G., Pfahringer, B., & Frank, E. (2010). Fast perceptron decision tree learning from evolving data streams. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2010)* (Vol. 6119, pp. 299–310). LNCS. Springer. doi:10.1007/978-3-642-13672-6_30

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009)* (pp. 139–148). ACM. doi:10.1145/1557019.1557041

Black, M. & Hickey, R. J. (1999). Maintaining the performance of a learned classifier under concept drift. *Intelligent Data Analysis*, *3*(6), 453–474. doi:10.1016/s1088-467x(99)00033-5

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

Blockeel, H., Bruynooghe, M., Džeroski, S., Ramon, J., & Struyf, J. (2002). Hierarchical multi-classification. In *Proceedings of the Workshop on Multi-Relational Data Mining (KDD 2002)* (pp. 21–35).

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., Schietgat, L., Struyf, J., Džeroski, S., & Clare, A. (2006). Decision trees for hierarchical multilabel classification: A case study in functional genomics. In *Knowledge Discovery in Databases: PKDD 2006* (Vol. 4213, pp. 18–29). LNCS. Springer. doi:10.1007/11871637_7

Bosnić, Z., Demšar, J., Kešpret, G., Rodrigues, P. P., Gama, J., & Kononenko, I. (2014). Enhancing data stream predictions with reliability estimators and explanation. *Engineering Applications of Artificial Intelligence*, *34*(Supplement C), 178–192. doi:10.1016/j.engappai.2014.06.001

Bosnić, Z. & Kononenko, I. (2008a). Comparison of approaches for estimating reliability of individual regression predictions. *Data & Knowledge Engineering*, *67*(3), 504–516. doi:10.1016/j.datak.2008.08.001

Bosnić, Z. & Kononenko, I. (2008b). Estimation of individual prediction reliability using the local sensitivity analysis. *Applied Intelligence*, *29*(3), 187–203. doi:10.1007/s10489-007-0084-9

Bosnić, Z., Rodrigues, P. P., Kononenko, I., & Gama, J. (2011). Correcting streaming predictions of an electricity load forecast system using a prediction reliability estimate. In *Man-Machine Interactions 2* (Vol. 103, pp. 343–350). AISC. Springer. doi:10.1007/978-3-642-23169-8_37

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

Brefeld, U., Büscher, C., & Scheffer, T. (2005). Multi-view discriminative sequential learning. In *Machine Learning: ECML 2005* (Vol. 3720, pp. 60–71). LNCS. Springer. doi:10.1007/11564096_11

Brefeld, U. & Scheffer, T. (2006). Semi-supervised learning for structured output variables. In *Proceedings of the 23rd International Conference on Machine learning (ICML 2006)* (pp. 145–152). ACM. doi:10.1145/1143844.1143863

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. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Chapman & Hall.

Breskvar, M., Kocev, D., & Džeroski, S. (2017). Multi-label classification using random label subset selections. In *Discovery Science (DS 2017)* (Vol. 10558, pp. 108–115). LNCS. Springer. doi:10.1007/978-3-319-67786-6_8

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 2017)* (pp. 88–93). doi:10.1109/smc-it.2017.22

Brouard, C., Szafranski, M., & d'Alché-Buc, F. (2016). Input output kernel regression: Supervised and semi-supervised structured output prediction with operator-valued kernels. *Journal of Machine Learning Research*, *17*(176), 1–48.

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: 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

Brzezinski, D. & Stefanowski, J. (2014). Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 81–94. doi:10.1109/tnnls.2013.2251352

Buntine, W. (1992). Learning classification trees. *Statistics and Computing*, *2*(2), 63–73. doi:10.1007/bf01889584

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). IEEE. doi:10.1109/ijcnn.2009.5178650

Cardona, H. D. V., Álvarez, M. A., & Orozco, Á. A. (2015). Convolved multi-output Gaussian processes for semi-supervised learning. In *Image Analysis and Processing — ICIAP 2015* (Vol. 9279, pp. 109–118). LNCS. Springer. doi:10.1007/978-3-319-23231-7_10

Cauwenberghs, G. & Poggio, T. (2001). Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 13 (NIPS 2000)* (pp. 409–415). NIPS Foundation.

Ceci, M. (2008). Hierarchical text categorization in a transductive setting. In *Proceedings of the 2008 IEEE International Conference on Data Mining Workshop (ICDMW 2008)* (pp. 184–191). IEEE. doi:10.1109/icdmw.2008.126

Ceci, M., Corizzo, R., Fumarola, F., Malerba, D., & Rashkovska, A. (2016). Predictive modeling of PV energy production: How to set up the learning task for a better prediction? *IEEE Transactions on Industrial Informatics*, *13*(3), 956–966. doi:10.1109/tii.2016.2604758

Cerri, R., Barros, R. C., & De Carvalho, A. C. (2014). Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, *80*(1), 39–56. doi:10.1016/j.jcss.2013.03.007

Cerri, R., Barros, R. C., & de Carvalho, A. C. (2012). A genetic algorithm for hierarchical multi-label classification. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC 2012)* (pp. 250–255). ACM. doi:10.1145/2245276.2245325

Cerri, R., Pappa, G. L., Carvalho, A. C. P., & Freitas, A. A. (2015). An extensive evaluation of decision tree-based hierarchical multilabel classification methods and performance measures. *Computational Intelligence*, *31*(1), 1–46. doi:10.1111/coin.12011

Cestnik, B., Kononenko, I., & Bratko, I. (1987). ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. In *Progress in Machine Learning – Proceedings of the Second European Working Session on Learning (ESWL 1987)* (pp. 31–45). Sigma Press.

Chang, M.-W., Ratinov, L., & Roth, D. (2012). Structured learning with constrained conditional models. *Machine Learning*, *88*(3), 399–431. doi:10.1007/s10994-012-5296-5

Chapelle, O., Schlkopf, B., & Zien, A. (2010). *Semi-supervised learning*. MIT Press. doi:10.7551/mitpress/9780262033589.001.0001

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. doi:10.1613/jair.953

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 2016 International Joint Conference on Neural Networks (IJCNN 2016)* (pp. 1458–1463). IEEE. doi:10.1109/ijcnn.2016.7727370

Chen, G., Song, Y., Wang, F., & Zhang, C. (2008). Semi-supervised multi-label learning by solving a Sylvester equation. In *Proceedings of the 2008 SIAM International Conference on Data Mining* (pp. 410–419). SIAM. doi:10.1137/1.9781611972788.37

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

Chen, Y. & Xu, D. (2004). Global protein function annotation through mining genome-scale data in yeast Saccharomyces cerevisiae. *Nucleic Acids Research*, *32*(21), 6414–6424. doi:10.1093/nar/gkh978

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

Chernoff, H. (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, *23*(4), 493–507. doi:10.1214/aoms/1177729330

Chicarro, A., Martin, P., & Trautner, R. (2004). The Mars Express mission: An overview. In *Mars Express: The Scientific Payload* (Vol. 1240, pp. 3–13). ESA Publications Division.

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. Springer. doi:10.1007/3-540-44794-6_4

Clare, A. & King, R. D. (2003). Predicting gene function in Saccharomyces cerevisiae. *Bioinformatics*, *19*(suppl_2), ii42–ii49. doi:10.1093/bioinformatics/btg1058

Collins, R. T., Liu, Y., & Leordeanu, M. (2005). Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(10), 1631–1643. doi:10.1109/tpami.2005.205

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

Crammer, K., Dredze, M., & Pereira, F. (2012). Confidence-weighted linear classification for text categorization. *Journal of Machine Learning Research*, *13*(Jun), 1891–1926.

Crammer, K., Kandola, J., & Singer, Y. (2004). Online classification on a budget. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)* (pp. 225–232). NIPS Foundation.

Crammer, K. & Singer, Y. (2003a). A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, *3*(Feb), 1025–1058.

Crammer, K. & Singer, Y. (2003b). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, *3*(Jan), 951–991.

D'Ambrosio, A., Aria, M., Iorio, C., & Siciliano, R. (2017). Regression trees for multivalued numerical response variables. *Expert Systems with Applications*, *69*, 21–28. doi:10.1016/j.eswa.2016.10.021

Dasu, T., Krishnan, S., Venkatasubramanian, S., & Yi, K. (2006). An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proceedings of the 38th Symposium on the Interface of Statistics, Computing Science, and Applications 2006 (Interface 2006)*. Interface Foundation of North America.

Davis, J. & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine learning (ICML 2006)* (pp. 233–240). ACM. doi:10.1145/1143844.1143874

Dawid, A. P. (1984). Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, *147*(2), 278–292. doi:10.2307/2981683

De Comité, F., Gilleron, R., & Tommasi, M. (2003). 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. Springer. doi:10.1007/3-540-45065-3_4 de Lucena, D. C. & Prudencio, R. B. (2015). Semi-supervised multi-label k-nearest neighbors classification algorithms. In *Proceedings of the 2015 Brazilian Conference on Intelligent Systems (BRACIS 2015)* (pp. 49–54). IEEE. doi:10.1109/bracis.2015.26

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

Dekel, O., Shamir, O., & Xiao, L. (2010). Learning to classify with missing and corrupted features. *Machine Learning*, *81*(2), 149–178. doi:10.1007/s10994-009-5124-8

Demšar, J. [Jaka] & Bosnić, Z. (2018). Detecting concept drift in data streams using model explanation. *Expert Systems with Applications*, *92*(Supplement C), 546–559. doi:10.1016/j.eswa.2017.10.003

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

Dhillon, P. S., Keerthi, S., Bellare, K., Chapelle, O., & Sellamanickam, S. (2012). Deterministic annealing for semi-supervised structured output learning. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2012)* (Vol. 22, pp. 299–307). PMLR. PMLR.

Dhillon, P. S., Sellamanickam, S., & Selvaraj, S. K. (2011). Semi-supervised multi-task learning of structured prediction models for web information extraction. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM 2011)* (pp. 957–966). ACM. doi:10.1145/2063576.2063713

Dimitrovski, I., Kocev, D., Loskovska, S., & Džeroski, S. (2011). Hierarchical annotation of medical images. *Pattern Recognition*, *44*(10-11), 2436–2449. doi:10.1016/j.patcog.2011.03.026

Ditzler, G. & Polikar, R. (2011). Hellinger distance based drift detection for nonstationary environments. In *Proceedings of the 2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE 2011)* (pp. 41–48). IEEE. doi:10.1109/cidue.2011.5948491

Ditzler, G. & Polikar, R. (2013). Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *25*(10), 2283–2301. doi:10.1109/tkde.2012.136

Domingos, P. & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000)* (pp. 71–80). doi:10.1145/347090.347107

Dong, G., Han, J., Lakshmanan, L. V., Pei, J., Wang, H., & Yu, P. S. (2003). Online mining of changes from data streams: Research problems and preliminary results. In *Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams (MDPS 2003)* (pp. 739–747).

Dries, A. & Rückert, U. (2009). Adaptive concept drift detection. *Statistical Analysis and Data Mining*, *2*(5-6), 311–327. doi:10.1002/sam.10054

Du, X. (2017). Semi-supervised learning of local structured output predictors. *Neurocomputing*, *220*, 151–159. doi:10.1016/j.neucom.2016.02.086

Duarte, J. & Gama, J. (2015). Multi-target regression from high-speed data streams with adaptive model rules. In *Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA 2015)* (pp. 1–10). doi:10.1109/dsaa.2015.7344900

Duarte, J. & Gama, J. (2017). Feature ranking in Hoeffding algorithms for regression. In *Proceedings of the Symposium on Applied Computing (SAC 2017)* (pp. 836–841). ACM. doi:10.1145/3019612.3019670

Duarte, J., Gama, J., & Bifet, A. (2016). Adaptive model rules from high-speed data streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *10*(3), 30. doi:10.1145/2829955

Džeroski, S. (2006). Towards a general framework for data mining. In *Knowledge Discovery in Inductive Databases (KDID 2006)* (Vol. 4747, pp. 259–300). LNCS. Springer. doi:10.1007/978-3-540-75549-4_16

Džeroski, S., Goethals, B., & Panov, P. (Eds.). (2010). *Inductive databases and constraint-based data mining*. Springer. doi:10.1007/978-1-4419-7738-0

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.

Elwell, R. & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, *22*(10), 1517–1531. doi:10.1109/tnn.2011.2160459

Fan, W. (2004). StreamMiner: A classifier ensemble-based engine to mine concept-drifting data streams. In *Proceedings of the Thirtieth International Conference on Very Large Databases (VLDB 2004)* (pp. 1257–1260). Morgan Kaufmann. doi:10.1016/B978-012088469-8/50121-2

Fanaee-T, H. & Gama, J. (2013). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, *2*(2-3), 113–127. doi:10.1007/s13748-013-0040-3

Frank, E., Wang, Y., Inglis, S., Holmes, G., & Witten, I. H. (1998). Using model trees for classification. *Machine Learning*, *32*(1). doi:10.1023/A:1007421302149

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). ACM.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, *19*(1), 1–67. doi:10.1214/aos/1176347963

Friedman, J. H. (1993). *Fast MARS* (Tech. Rep. No. LCS 110). Department of Statistics, Stanford University.

Friedman, J. H., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1–22. doi:10.18637/jss.v033.i01

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

Furao, S., Sakurai, K., Kamiya, Y., & Hasegawa, O. (2007). An online semi-supervised active learning algorithm with self-organiing incremental neural network. In *Proceedings of the 2007 International Joint Conference on Neural Networks (IJCNN 2007)* (pp. 1139–1144). IEEE. doi:10.1109/ijcnn.2007.4371118

Fürnkranz, J., Hüllermeier, E., Mencía, E. L., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, *73*(2), 133–153. doi:10.1007/s10994-008-5064-8

Gallant, S. I. (1986). Optimal linear discriminants. In *Proceeding of the Eighth International Conference on Pattern Recognition* (pp. 849–852). IEEE.

Gama, J. (2010). *Knowledge discovery from data streams*. CRC Press.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence – SBIA 2004* (Vol. 3171, pp. 286–295). LNCS. Springer. doi:10.1007/978-3-540-28645-5_29

Gama, J., Rocha, R., & Medas, P. (2003). Accurate decision trees for mining high-speed data streams. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003)* (pp. 523–528). ACM. doi:10.1145/956750.956813

Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, *46*(4), 44:1–44:37. doi:10.1145/2523813

Gentile, C. (2001). A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, *2*(Dec), 213–242.

Gibaja, E. & Ventura, S. (2015). A tutorial on multilabel learning. *ACM Computing Surveys (CSUR)*, *47*(3), Article no. 52. doi:10.1145/2716262

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.

Godbole, S. & Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2004)* (Vol. 3056, pp. 22–30). LNCS. Springer. doi:10.1007/978-3-540-24775-3_5

Goldberg, A. B., Li, M., & Zhu, X. (2008). Online manifold regularization: A new learning setting and empirical study. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2008)* (Vol. 5211, pp. 393–407). LNCS. Springer. doi:10.1007/978-3-540-87479-9_44

Goldberg, A. B., Zhu, X., Furger, A., & Xu, J.-M. (2011). OASIS: Online active semi-supervised learning. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (pp. 362–367). AAAI.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., … Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, *106*(9-10), 1469–1495. doi:10.1007/s10994-017-5642-8

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). IEEE. doi:10.1109/ictai.2013.76

Gonçalves, P. M. & de Barros, R. S. M. (2013). RCD: A recurring concept drift framework. *Pattern Recognit1ion Letters*, *34*(9), 1018–1025. doi:10.1016/j.patrec.2013.02.005

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. Springer. doi:10.1007/978-3-540-27779-8_34

Gönen, M. & Kaski, S. (2014). Kernelized Bayesian matrix factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(10), 2047–2060. doi:10.1109/tpami.2014.2313125

Goodman, J. & Yih, S. W. (2006). Online discriminative spam filter training. In *Proceedings of the 3rd Conference on Email and Anti-Spam (CAES 2006)*. CAES.

Grabner, H., Leistner, C., & Bischof, H. (2008). Semi-supervised on-line boosting for robust tracking. In *Computer Vision – ECCV 2008* (Vol. 5302, pp. 234–247). LNCS. Springer. doi:10.1007/978-3-540-88682-2_19

Guan, Y., Myers, C. L., Hess, D. C., Barutcuoglu, Z., Caudy, A. A., & Troyanskaya, O. G. (2008). Predicting gene function in a hierarchical context with an ensemble of classifiers. *Genome Biology*, *9*(Supplement 1), S3. doi:10.1186/gb-2008-9-s1-s3

Guo, Y. & Schuurmans, D. (2012). Semi-supervised multi-label classification. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2012)* (Vol. 7524, pp. 355–370). LNCS. Springer. doi:10.1007/978-3-642-33486-3_23

Hand, D. J. & Yu, K. (2001). Idiot's Bayes – not so stupid after all? *International Statistical Review*, *69*(3), 385–398. doi:10.2307/1403452

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

Harel, M., Mannor, S., El-Yaniv, R., & Crammer, K. (2014). Concept drift detection through resampling. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)* (Vol. 32, pp. 1009–1017). PMLR. PMLR.

Helmbold, D. P., Littlestone, N., & Long, P. M. (1992). Apple tasting and nearly one-sided learning. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS 1992)* (pp. 493–502). IEEE. doi:10.1109/sfcs.1992.267802

Helmbold, D. P. & Long, P. M. (1994). Tracking drifting concepts by minimizing disagreements. *Machine Learning*, *14*(1), 27–45. doi:10.1007/bf00993161

Helmbold, D. P. & Warmuth, M. K. (1995). On weak learning. *Journal of Computer and System Sciences*, *50*(3), 551–573. doi:10.1006/jcss.1995.1044

Hersh, W., Buckley, C., Leone, T. J., & Hickam, D. (1994). OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1994)* (pp. 192–201). Springer. doi:10.1007/978-1-4471-2099-5_20

Hickey, R. J. & Black, M. M. (2001). Refined time stamps for concept drift detection during mining for classification rules. In *Temporal, Spatial, and Spatio-Temporal Data Mining* (Vol. 2007, pp. 20–30). LNCS. Springer. doi:10.1007/3-540-45244-3_3

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554. doi:10.1162/neco.2006.18.7.1527

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*(301), 13–30. doi:10.2307/2282952

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

Holmes, G., Kirkby, R., & Pfahringer, B. (2005). Stress-testing Hoeffding trees. In *Knowledge discovery in databases: Pkdd 2005* (Vol. 3721, pp. 495–502). LNCS. Springer. doi:10.1007/11564126_50

Holmes, G., Richard, K., & Pfahringer, B. (2005). Tie-breaking in Hoeffding trees. In *Proceedings of the Second International Workshop on Knowledge Discovery from Data Streams (IWKDDS 2005).*

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

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001)* (pp. 97–106). ACM. doi:10.1145/502512.502529

Ichihashi, H., Shirai, T., Nagasaka, K., & Miyoshi, T. (1996). Neuro-fuzzy ID3: A method of inducing fuzzy decision trees with linear programming for maximizing entropy and an algebraic method for incremental learning. *Fuzzy Sets and Systems*, *81*(1), 157–167. doi:10.1016/0165-0114(95)00247-2

Ikonomovska, E., Gama, J., & Džeroski, S. (2011a). Incremental multi-target model trees for data streams. In *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC 2011)* (pp. 988–993). ACM. doi:10.1145/1982185.1982402

Ikonomovska, E., Gama, J., & Džeroski, S. (2011b). Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, *23*(1), 128–168. doi:10.1007/s10618-010-0201-y

Ikonomovska, E., Gama, J., & Džeroski, S. (2015). Online tree-based ensembles and option trees for regression on evolving data streams. *Neurocomputing*, *150*(Part B), 458–470. doi:10.1016/j.neucom.2014.04.076

Ikonomovska, E., Gama, J., Ženko, B., & Džeroski, S. (2011). Speeding-up Hoeffding-based regression trees with options. In *Proceedings of the Twenty-eighth International Conference on Machine Learning (ICML 2011)* (pp. 537–552). International Machine Learning Society.

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. (). *Information technology – General-Purpose Datatypes (GPD)* (Standard No. ISO/IEC 11404:2007). Retrieved from https://www.iso.org/standard/39479.html

Jaccard, P. (1912). The distribution of the flora in the Alpine zone. *New Phytologist*, *11*(2), 37–50. doi:10.1111/j.1469-8137.1912.tb05611.x

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

Jiang, F., Jia, L., Sheng, X., & LeMieux, R. (2016). Manifold regularization in structured output space for semi-supervised structured output prediction. *Neural Computing and Applications*, *27*(8), 2605–2614. doi:10.1007/s00521-015-2029-2

Jiang, W., Er, G., Dai, Q., & Gu, J. (2006). Similarity-based online feature selection in content-based image retrieval. *IEEE Transactions on Image Processing*, *15*(3), 702–712. doi:10.1109/tip.2005.863105

Karalič, A. (1992). Employing linear regression in regression tree leaves. In *Proceedings of the 10th European Conference on Artificial intelligence (ECAI 1992)* (pp. 440–441). John Wiley & Sons.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2005). On the utility of incremental feature selection for the classification of textual data streams. In *Advances in informatics (pci 2005)* (Vol. 3746, pp. 338–348). LNCS. Springer. doi:10.1007/11573036_32

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2006). Dynamic feature space and incremental feature selection for the classification of textual data streams. In *Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams (IWKDDS 2006)* (pp. 107–116). Springer.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2016). Discussion and review on evolving data streams and concept drift adapting. *Evolving Systems*, 1–23. doi:10.1007/s12530-016-9168-2

Kim, M. (2013). Semi-supervised learning of hidden conditional random fields for time-series classification. *Neurocomputing*, *119*, 339–349. doi:10.1016/j.neucom.2013.03.024

Kira, K. & Rendell, L. A. (1992). A practical approach to feature selection. In *Proceedings of the Ninth International Workshop on Machine learning (ML 1992)* (pp. 249–256). Morgan Kaufmann. doi:10.1016/B978-1-55860-247-2.50037-1

Kiritchenko, S., Matwin, S., Nock, R., & Famili, A. F. (2006). Learning and evaluation in the presence of class hierarchies: Application to text categorization. In *Advances in Artificial Intelligence (AI 2006)* (Vol. 4103, pp. 395–406). LNCS. Springer. doi:10.1007/11766247_34

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, *8*(3), 281–300.

Klinkenberg, R. & Joachims, T. (2000). Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)* (pp. 487–494). Morgan Kaufmann.

Klinkenberg, R. & Renz, I. (1998). *Adaptive information filtering: Learning in the presence of concept drifts*. AAAI.

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

Kohavi, R. & Kunz, C. (1997). Option decision trees with majority votes. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997)* (pp. 161–169). Morgan Kaufmann.

Kong, X., Ng, M. K., & Zhou, Z.-H. (2013). Transductive multilabel learning via label set propagation. *IEEE Transactions on Knowledge and Data Engineering*, *25*(3), 704–719. doi:10.1109/tkde.2011.141

Kononenko, I. & Kukar, M. (2007). *Machine learning and data mining: Introduction to principles and algorithms*. Horwood Publishing.

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)* (pp. 282–289). Morgan Kaufmann.

Lance, G. N. & Williams, W. T. (1967). Mixed-data classificatory programs I – Agglomerative systems. *Australian Computer Journal*, *1*(1), 15–20.

Lang, K. (2008). The 20 newsgroups dataset. Retrieved November 16, 2017, from http://qwone.com/~jason/20Newsgroups/

Last, M. (2002). Online classification of nonstationary data streams. *Intelligent Data Analysis*, *6*(2), 129–147.

Lehmann, T. M., Schubert, H., Keysers, D., Kohnen, M., & Wein, B. B. (2003). The IRMA code for unique classification of medical images. In *Medical Imaging 2003: PACS and Integrated Medical Information Systems – Design and Evaluation* (Vol. 5033, pp. 440–451). Proceedings of SPIE. SPIE. doi:10.1117/12.480677

Levatić, J. (2017). *Semi-supervised learning for structured output prediction* (Doctoral dissertation, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia).

Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2017a). Self-training for multi-target regression with tree ensembles. *Knowledge-Based Systems*, *123*, 41–60. doi:10.1016/j.knosys.2017.02.014

Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2017b). Semi-supervised classification trees. *Journal of Intelligent Information Systems*, *49*(3), 461–486. doi:10.1007/s10844-017-0457-4

Li, Y. [Yi] & Long, P. M. (2000). The relaxed online maximum margin algorithm. *Machine Learning*, *46*(1-3), 361–387. doi:10.1023/a:1012435301888

Li, Y. [Yujia] & Zemel, R. (2014). High order regularization for semi-supervised learning of structured output problems. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)* (Vol. 32, *2*, pp. 1368–1376). PMLR. PMLR.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*(4), 285–318. doi:10.1023/a:1022869011914

Littlestone, N. (1989). From on-line to batch learning. In *Proceedings of the Second Annual Workshop on Computational Learning Theory (COLT 1989)* (pp. 269–284). Morgan Kaufmann.

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. Springer. doi:10.1007/978-3-319-18038-0_14

Liu, W., Wang, J., & Chang, S.-F. (2012). Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, *100*(9), 2624–2638. doi:10.1109/jproc.2012.2197809

Maass, W. (1991). On-line learning with an oblivious environment and the power of randomization. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT 1991)* (pp. 167–178). Morgan Kaufmann.

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. Springer. 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

Marz, N. & Warren, J. (2015). *Big data: Principles and best practices of scalable realtime data systems*. Manning Publications.

McDiarmid, C. (1989). On the method of bounded differences. *Surveys in Combinatorics*, *141*, 148–188. doi:10.1017/cbo9781107359949.008

Mencía, E. L. & Janssen, F. (2016). Learning rules for multi-label classification: A stacking and a separate-and-conquer approach. *Machine Learning*, *105*(1), 77–126. doi:10.1007/s10994-016-5552-1

Mevik, B.-H. & Wehrens, R. (2007). The pls package: Principal component and partial least squares regression in R. *Journal of Statistical Software*, *18*(2), 1–23. doi:10.18637/jss.v018.i02

Milborrow, S. (2017). *Earth: Multivariate adaptive regression spline models*. R package version 4.6.0.

Mileski, V. (2017). *Tree methods for hierarchical multi-target regression* (Master's thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia).

Mileski, V., Džeroski, S., & Kocev, D. (2017). Predictive clustering trees for hierarchical multi-target regression. In *Advances in Intelligent Data Analysis XVI (IDA 2017)* (Vol. 10584, pp. 223–234). LNCS. Springer. doi:10.1007/978-3-319-68765-0_19

Minku, L. L., White, A. P., & Yao, X. (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, *22*(5), 730–742. doi:10.1109/tkde.2009.156

Minku, L. L. & Yao, X. (2012). DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, *24*(4), 619–633. doi:10.1109/tkde.2011.58

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Mostafavi, S., Ray, D., Warde-Farley, D., Grouios, C., & Morris, Q. (2008). GeneMANIA: A real-time multiple association network integration algorithm for predicting gene function. *Genome Biology*, *9*(Supplement 1), S4. doi:10.1186/gb-2008-9-s1-s4

Mouss, H., Mouss, D., Mouss, N., & Sefouhi, L. (2004). Test of Page-Hinckley, an approach for fault detection in an agro-alimentary production system. In *Proceedings of 5th Asian Control Conference (ASCC 2004)* (Vol. 2, pp. 815–818). IEEE.

Navaratnam, R., Fitzgibbon, A. W., & Cipolla, R. (2007). The joint manifold model for semi-supervised multi-valued regression. In *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV 2007)* (pp. 1–8). IEEE. doi:10.1109/iccv.2007.4408976

Nemenyi, P. (1963). *Distribution-free multiple comparisons* (Doctoral dissertation, Princeton University, New Jersey, USA).

Obozinski, G., Lanckriet, G., Grant, C., Jordan, M. I., & Noble, W. S. (2008). Consistent probabilistic outputs for protein function prediction. *Genome Biology*, *9*(Supplement 1), S6. doi:10.1186/gb-2008-9-s1-s6

Osojnik, A., Džeroski, S., & Kocev, D. (2016). Option predictive clustering trees for multi-target regression. In *Discovery Science (DS 2016)* (Vol. 9956, pp. 118–133). LNCS. Springer. doi:10.1007/978-3-319-46307-0_8

Osojnik, A., Panov, P., & Džeroski, S. (2015a). Multi-label classification via multi-target regression on data streams. In *Discovery Science (DS 2015)* (Vol. 9356, pp. 170–185). LNCS. Springer. doi:10.1007/978-3-319-24282-8_15

Osojnik, A., Panov, P., & Džeroski, S. (2015b). Tree-based approaches for multi-target regression on data streams. In *Proceedings of the 4th Workshop on New Frontiers in Mining Complex Patterns (NFMCP 2015)* (pp. 2–13).

Osojnik, A., Panov, P., & Džeroski, S. (2016). Comparison of tree-based methods for multi-target regression on data streams. In *New Frontiers in Mining Complex Patterns (NFMCP 2015)* (Vol. 9607, pp. 17–31). LNCS. Springer. doi:10.1007/978-3-319-39315-5_2

Osojnik, A., Panov, P., & Džeroski, S. (2017a). Multi-label classification via multi-target regression on data streams. *Machine Learning*, *106*(6), 745–770. doi:10.1007/s10994-016-5613-5

Osojnik, A., Panov, P., & Džeroski, S. (2017b). Tree-based methods for online multi-target regression. *Journal of Intelligent Information Systems*. doi:10.1007/s10844-017-0462-7

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). IEEE. 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. IOS Press. doi:10.3233/978-1-61499-672-9-1467

Otero, F. E., Freitas, A. A., & Johnson, C. G. (2010). A hierarchical multi-label classification ant colony algorithm for protein function prediction. *Memetic Computing*, *2*(3), 165–181. doi:10.1007/s12293-010-0045-4

Oza, N. C. (2005). Online bagging and boosting. In *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics (SMC 2005)* (Vol. 3, pp. 2340–2345). IEEE. doi:10.1109/icsmc.2005.1571498

Oza, N. C. & Russel, S. J. (2001). Experimental comparisons of online and batch versions of bagging and boosting. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001)* (pp. 359–364). ACM. doi:10.1145/502512.502565

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Dubourg, V., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Perkins, S., Lacker, K., & Theiler, J. (2003). Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, *3*(Mar), 1333–1356.

Petković, M., Džeroski, S., & Kocev, D. (2017). Feature ranking for multi-target regression with tree ensemble methods. In *Discovery Science (DS 2017)* (Vol. 10558, pp. 171–185). LNCS. Springer. doi:10.1007/978-3-319-67786-6_13

Pfahringer, B., Holmes, G., & Kirkby, R. (2007). New options for Hoeffding trees. In M. A. Orgun & J. Thornton (Eds.), *AI 2007: Advances in Artificial Intelligence* (Vol. 4830, pp. 90–99). LNCS. Springer. doi:10.1007/978-3-540-76928-6_11

Poon, H. & Domingos, P. (2011). Sum-product networks: A new deep architecture. In *Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (pp. 689–690). IEEE. doi:10.1109/iccvw.2011.6130310

Potts, D. & Sammut, C. (2005). Incremental learning of linear model trees. *Machine Learning*, *61*(1-3), 5–48. doi:10.1007/s10994-005-1121-8

Pugelj, M. & Džeroski, S. (2011). Predicting structured outputs *k*-nearest neighbours method. In *Discovery Science (DS 2011)* (Vol. 6926, pp. 262–276). LNCS. Springer. doi:10.1007/978-3-642-24477-3_22

Qu, W., Zhang, Y., Zhu, J., & Qiu, Q. (2009). Mining multi-label concept-drifting data streams using dynamic classifier ensemble. In *Advances in Machine Learning (ACML 2009)* (Vol. 5828, pp. 308–321). LNCS. Springer. doi:10.1007/978-3-642-05224-8_24

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. R. (1992). Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence (AI 1992)* (pp. 343–348). World Scientific. doi:10.1142/9789814536271

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* Morgan Kaufmann.

Rashkovska, A., Novljan, J., Smolnikar, M., Mohorčič, M., & Fortuna, C. (2015). Online short-term forecasting of photovoltaic energy production. In *Proceedings of the 2015 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT 2005)* (pp. 1–5). IEEE. doi:10.1109/isgt.2015.7131880

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian processes for machine learning.* MIT Press.

Read, J. (2008). A pruned problem transformation method for multi-label classification. In *Proceedings of 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008)* (pp. 143–150).

Read, J. (2010). *Scalable multi-label classification* (Doctoral dissertation, The University of Waikato, Waikato, New Zealand).

Read, J., Bifet, A., Holmes, G., & Pfahringer, B. (2012). Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, *88*(1-2), 243–272. doi:10.1007/s10994-012-5279-6

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 Eighth IEEE International Conference on Data Mining (ICDM 2008)* (pp. 995–1000). IEEE. doi:10.1109/icdm.2008.74

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2009). Classifier chains for multi-label classification. *Machine Learning*, *85*, 333–359. doi:10.1007/s10994-011-5256-5

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333–359. doi:10.1007/s10994-011-5256-5

Riedmiller, M. & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN 1993)* (pp. 586–591). IEEE. doi:10.1109/icnn.1993.298623

Rodrigues, P. P., Bosnić, Z., Gama, J., & Kononenko, I. (2012). Estimating reliability for assessing and correcting individual streaming predictions. In *Reliable Knowledge Discovery* (Chap. 2, pp. 29–49). Springer. doi:10.1007/978-1-4614-1903-7_2

Rodrigues, P. P., Gama, J., & Bosnić, Z. (2008). Online reliability estimates for individual predictions in data streams. In *Proceeding of the 2008 IEEE International Conference on Data Mining Workshops (ICDMW 2008)* (pp. 36–45). doi:10.1109/icdmw.2008.123

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. doi:10.1037/h0042519

Ross, G. J., Adams, N. M., Tasoulis, D. K., & Hand, D. J. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, *33*(2), 191–198. doi:10.1016/j.patrec.2011.08.019

Rousu, J., Saunders, C., Szedmak, S., & Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, *7*(Jul), 1601–1626.

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.

Rutkowski, L., Pietruczuk, L., Duda, P., & Jaworski, M. (2013). Decision trees for mining data streams based on the McDiarmid's bound. *IEEE Transactions in Knowledge and Data Engineering*, *25*(6), 1272–1279. doi:10.1109/tkde.2012.66

Salganicoff, M. (1993). *Explicit forgetting algorithms for memory based learning* (Tech. Rep. No. MS-CIS-93-80). Department of Computer and Information Science, University of Pennsylvania.

Salzberg, S. (1991). A nearest hyperrectangle learning method. *Machine Learning*, *6*(3), 251–276. doi:10.1007/bf00114779

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

Santos, A. & Canuto, A. (2014). Applying semi-supervised learning in hierarchical multilabel classification. *Expert Systems with Applications*, *41*(14), 6075–6085. doi:10.1016/j.eswa.2014.03.052

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

Schlimmer, J. C. & Fisher, D. (1986). A case study of incremental concept induction. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496–501). AAAI.

Sedgewick, R. & Wayne, K. (2011). *Algorithms*. Pearson Education.

Sellamanickam, S., Tiwari, C., & Selvaraj, S. K. (2012). Regularized structured output learning with partial labels. In *Proceedings of the 2012 SIAM International Conference on Data Mining* (pp. 1059–1070). SIAM. doi:10.1137/1.9781611972825.91

Shaker, A. & Hüllermeier, E. (2012). IBLStreams: A system for instance-based classification and regression on data streams. *Evolving Systems*, *3*(4), 235–249. doi:10.1007/s12530-012-9059-0

Sharma, N., Sharma, P., Irwin, D., & Shenoy, P. (2011). Predicting solar generation from weather forecasts using machine learning. In *Proceedings of the 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm 2011)* (pp. 528–533). IEEE. doi:10.1109/smartgridcomm.2011.6102379

Shen, F., Yu, H., Sakurai, K., & Hasegawa, O. (2011). An incremental online semi-supervised active learning algorithm based on self-organizing incremental neural network. *Neural Computing and Applications*, *20*(7), 1061–1074. doi:10.1007/s00521-010-0428-y

Shi, Z., Wen, Y., Feng, C., & Zhao, H. (2014). Drift detection for multi-label data streams based on label grouping and entropy. In *Proceedings of the 2014 IEEE International Conference on Data Mining Workshop (ICDMW 2014)* (pp. 724–731). IEEE. doi:10.1109/icdmw.2014.92

Shi, Z., Xue, Y., Wen, Y., & Cai, G. (2014). Efficient class incremental learning for multi-label classification of evolving data streams. In *Proceedings of the 2014 International Joint Conference on Neural Networks (IJCNN 2014)* (pp. 2093–2099). IEEE. doi:10.1109/ijcnn.2014.6889926

Silla Jr, C. N. & Freitas, A. A. (2009). A global-model naive Bayes approach to the hierarchical prediction of protein functions. In *Proceedings of the Ninth IEEE International Conference on Data Mining (ICDM 2009)* (pp. 992–997). IEEE. doi:10.1109/icdm.2009.85

Silla, C. N. & Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovvery*, *22*(1-2), 31–72. doi:10.1007/s10618-010-0175-9

Slavkov, I. (2012). *An evaluation method for feature rankings* (Doctoral dissertation, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia).

Slavkov, I. & Džeroski, S. (2010). Analyzing gene expression data with predictive clustering trees. In *Inductive Databases and Constraint-Based Data Mining* (Chap. 16, pp. 389–406). Springer. doi:10.1007/978-1-4419-7738-0_16

Sobhani, P. & Beigy, H. (2011). New drift detection method for data streams. In *Adaptive and Intelligent Systems* (Vol. 6943, pp. 88–97). LNCS. Springer. doi:10.1007/978-3-642-23857-4_12

Sousa, R. & Gama, J. (2017). Co-training semi-supervised learning for single-target regression in data streams using AMRules. In *Foundations of Intelligent Systems (ISMIS 2017)* (pp. 499–508). Springer. doi:10.1007/978-3-319-60438-1_49

Sousa, R. & Gama, J. (2018). Multi-label classification from high-speed data streams with adaptive model rules and random rules. *Progress in Artificial Intelligence*. doi:10.1007/s13748-018-0142-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. Springer. doi:10.1007/978-3-540-87881-0_40

Spyromitros-Xioufis, E. (2011). *Dealing with concept drift and class imbalance in multi-label stream classification* (Doctoral dissertation, Aristotle University of Thessaloniki).

Spyromitros-Xioufis, E., Groves, W., Tsoumakas, G., & Vlahavas, I. (2012). Multi-label classification methods for multi-target regression. arXiv: 1211.6581

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

Srivastava, A. N. & Zane-Ulman, B. (2005). Discovering recurring anomalies in text reports regarding complex space systems. In *Proceedings of the 2005 IEEE Aerospace Conference* (pp. 3853–3862). IEEE. doi:10.1109/aero.2005.1559692

Stanley, K. O. (2003). *Learning concept drift with a committee of decision trees* (Tech. Rep. No. AI03-302). Department of Computer Sciences, University of Texas at Austin.

Stenger, B., Thayananthan, A., Torr, P. H., & Cipolla, R. (2007). Estimating 3D hand pose using hierarchical multi-label classification. *Image and Vision Computing*, *25*(12), 1885–1894. doi:10.1016/j.imavis.2005.12.018

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

Stojanova, D., Ceci, M., Malerba, D., & Džeroski, S. (2013). Using PPI network autocorrelation in hierarchical multi-label classification trees for gene function prediction. *BMC Bioinformatics*, *14*, 285. doi:10.1186/1471-2105-14-285

Stojanova, D., Panov, P., Gjorgjioski, V., Kobler, A., & Džeroski, S. (2010). Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics*, *5*(4), 256–266. doi:10.1016/j.ecoinf.2010.03.004

Struyf, J. & Džeroski, S. (2006). Constraint based induction of multi-objective regression trees. In *Knowledge Discovery in Inductive Databases (KDID 2005)* (Vol. 3933, pp. 222–233). LNCS. Springer. doi:10.1007/11733492_13

Su, H. & Rousu, J. (2015). Multilabel classification through random graph ensembles. *Machine Learning*, *99*(2), 231–256. doi:10.1007/s10994-014-5465-9

Subramanya, A., Petrov, S., & Pereira, F. (2010). Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)* (pp. 167–176). Association for Computational Linguistics.

Sucar, L. E., 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. doi:10.1016/j.patrec.2013.11.007

Sun, Y. (2007). Iterative RELIEF for feature weighting: Algorithms, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6). doi:10.1109/tpami.2007.1093

Sun, Z., Zhao, Y., Cao, D., & Hao, H. (2017). Hierarchical multilabel classification with optimal path prediction. *Neural Processing Letters*, *45*(1), 263–277. doi:10.1007/s11063-016-9526-x

Suzuki, J., Fujino, A., & Isozaki, H. (2007). Semi-supervised structured output learning based on a hybrid generative and discriminative approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)* (pp. 791–800). The Association for Computational Linguistics.

Štrumbelj, E. & Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, *11*(Jan), 1–18.

Štrumbelj, E., Kononenko, I., & Šikonja, M. R. (2009). Explaining instance classifications with interactions of subsets of feature values. *Data & Knowledge Engineering*, *68*(10), 886–904. doi:10.1016/j.datak.2009.01.004

Švec, J. (2014). Semi-supervised learning algorithm for binary relevance multi-label classification. In *Web Information Systems Engineering – WISE 2014 Workshops* (Vol. 9051, pp. 1–13). LNCS. Springer. doi:10.1007/978-3-319-20370-6_1

Teo, C. H., Globerson, A., Roweis, S. T., & Smola, A. J. (2008). Convex learning with invariances. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)* (pp. 1489–1496). NIPS Foundation.

Tian, W., Zhang, L. V., Taşan, M., Gibbons, F. D., King, O. D., Park, J., ... Roth, F. P. (2008). Combining guilt-by-association and guilt-by-profiling to predict Saccharomyces cerevisiae gene function. *Genome Biology*, *9*(1), S7. doi:10.1186/gb-2008-9-s1-s7

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288. doi:10.1111/j.1467-9868.2011.00771.x

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD 2008)* (pp. 30–44).

Tsoumakas, G. & Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In *Machine learning: ECML 2007* (Vol. 4701, pp. 406–417). LNCS. Springer. doi:10.1007/978-3-540-74958-5_38

Tsymbal, A. (2004). *The problem of concept drift: Definitions and related work* (Tech. Rep. No. TCD-CS-2004-15). Department of Computer Science, Trinity College Dublin.

Utgoff, P. E. (1994). An improved algorithm for incremental induction of decision trees. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML 1994)* (pp. 318–325). Morgan Kaufmann.

Valente, A., Ginsburg, G., & Engelhardt, B. E. (2015). Nonparametric reduced-rank regression for multi-SNP, multi-trait association mapping. arXiv: 1512.02306

Valentini, G. (2011). True path rule hierarchical ensembles for genome-wide gene function prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *8*(3), 832–847. doi:10.1109/tcbb.2010.38

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142. doi:10.1145/1968.1972

Vapnik, V. N. & Kotz, S. (1982). *Estimation of dependences based on empirical data.* Springer.

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

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, *73*(2), 185–214. doi:10.1007/s10994-008-5077-3

Verbeeck, D. & Blockeel, H. (2015). Slower can be faster: The iRetis incremental model tree learner. In *Advances in Intelligent Data Analysis XIV* (Vol. 9385, pp. 322–333). LNCS. Springer. doi:10.1007/978-3-319-24465-5_28

Wang, B. & Tsotsos, J. (2016). Dynamic label propagation for semi-supervised multi-class multi-label classification. *Pattern Recognition*, *52*, 75–84. doi:10.1016/j.patcog.2015.10.006

Wang, J., Zhao, Y., Wu, X., & Hua, X.-S. (2011). A transductive multi-label learning approach for video concept detection. *Pattern Recognition*, *44*(10), 2274–2286. doi:10.1016/j.patcog.2010.07.015

Wang, S. [Shangfei], 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

Wang, S. [Shuo], Minku, L. L., Ghezzi, D., Caltabiano, D., Tino, P., & Yao, X. (2013). Concept drift detection for online class imbalance learning. In *Proceedings of the 2013*

International Joint Conference on Neural Networks (IJCNN 2013) (pp. 1–10). IEEE. doi:10.1109/ijcnn.2013.6706768

Wang, Y., Haffari, G., Wang, S., & Mori, G. (2009). A rate distortion approach for semi-supervised conditional random fields. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)* (pp. 2008–2016). NIPS Foundation.

Wang, Y., Pei, J., Lin, X., Zhang, Q., & Zhang, W. (2014). An iterative fusion approach to graph-based semi-supervised learning from multiple views. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2014)* (Vol. 8444, pp. 162–173). LNCS. Springer. doi:10.1007/978-3-319-06605-9_14

Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, *30*(4), 964–994. doi:10.1007/s10618-015-0448-4

Widmer, G. & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, *23*(1), 69–101. doi:10.1007/bf00116900

Widrow, B. & Hoff, M. E. (1960). *Adaptive switching circuits* (Tech. Rep. No. 1553-1). Solid-State Electronics Laboratory, Stanford University.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wu, L. & Zhang, M.-L. (2013). Multi-label classification with unlabeled data: An inductive approach. In *Proceedings of the 5th Asian Conference on Machine Learning (ACML 2013)* (Vol. 29, pp. 197–212). PMLR. PMLR.

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., & 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, M., Sun, F., & Jiang, X. (2014). Multi-label learning with co-training based on semi-supervised regression. In *Proceedings of the 2014 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC 2014)* (pp. 175–180). IEEE. doi:10.1109/spac.2014.6982681

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, Y., 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). IEEE. 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 Thirty-frst AAAI Conference on Artificial Intelligence and the Twenty-ninth Innovative Applications of Artificial Intelligence Conference (AAAI 2017)* (Vol. 4, pp. 2838–2844). AAAI.

Yoon, H., Yang, K., & Shahabi, C. (2005). Feature subset selection and feature ranking for multivariate time series. *IEEE Transactions on Knowledge and Data Engineering*, *17*(9), 1186–1198. doi:10.1109/tkde.2005.144

Zeisl, B., Leistner, C., Saffari, A., & Bischof, H. (2010). On-line semi-supervised multiple-instance boosting. In *Proceedings of the 2010 IEEE Conference on Computer Vision* *and Pattern Recognition (CVPR 2010)* (pp. 1879–1879). IEEE. doi:10.1109/cvpr.2010.5539860

Zha, Z.-J., Mei, T., Wang, J., Wang, Z., & Hua, X.-S. (2009). Graph-based semi-supervised learning with multiple labels. *Journal of Visual Communication and Image Representation*, *20*(2), 97–103. doi:10.1016/j.jvcir.2008.11.009

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). IEEE. 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. (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). IEEE. doi:10.1109/cimsa.2012.6269600

Zhang, Y. & Yeung, D.-Y. (2009). Semi-supervised multi-task regression. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2009)* (Vol. 5782, pp. 617–631). LNCS. Springer. doi:10.1007/978-3-642-04174-7_40

Zhao, C. & Zhai, S. (2015). Minimum variance semi-supervised boosting for multi-label classification. In *Proceedings of the 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP 2015)* (pp. 1342–1346). IEEE. doi:10.1109/globalsip.2015.7418417

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

Zien, A., Brefeld, U., & Scheffer, T. (2007). Transductive support vector machines for structured variables. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)* (pp. 1183–1190). ACM. doi:10.1145/1273496.1273645

Ženko, B. & Džeroski, S. (2008). Learning classification rules for multiple target attributes. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2008)* (Vol. 5012, pp. 454–465). LNCS. Springer. doi:10.1007/978-3-540-68125-0_40

Bibliography

Publications Related to the Thesis

Journal Articles

Osojnik, A., Panov, P., & Džeroski, S. (2017a). Multi-label classification via multi-target regression on data streams. *Machine Learning*, *106*(6), 745–770. doi:10.1007/s10994-016-5613-5

Osojnik, A., Panov, P., & Džeroski, S. (2017b). Tree-based methods for online multi-target regression. *Journal of Intelligent Information Systems*. doi:10.1007/s10844-017-0462-7

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 2017)* (pp. 88–93). doi:10.1109/smc-it.2017.22

Osojnik, A., Džeroski, S., & Kocev, D. (2016). Option predictive clustering trees for multi-target regression. In *Discovery Science (DS 2016)* (Vol. 9956, pp. 118–133). LNCS. Springer. doi:10.1007/978-3-319-46307-0_8

Osojnik, A., Panov, P., & Džeroski, S. (2015a). Multi-label classification via multi-target regression on data streams. In *Discovery Science (DS 2015)* (Vol. 9356, pp. 170–185). LNCS. Springer. doi:10.1007/978-3-319-24282-8_15

Osojnik, A., Panov, P., & Džeroski, S. (2015b). Tree-based approaches for multi-target regression on data streams. In *Proceedings of the 4th Workshop on New Frontiers in Mining Complex Patterns (NFMCP 2015)* (pp. 2–13).

Osojnik, A., Panov, P., & Džeroski, S. (2016a). Comparison of tree-based methods for multi-target regression on data streams. In *New Frontiers in Mining Complex Patterns (NFMCP 2015)* (Vol. 9607, pp. 17–31). LNCS. Springer. doi:10.1007/978-3-319-39315-5_2

Other Publications

Journal Articles

Osojnik, A., Panov, P., & Džeroski, S. (2016b). Modeling dynamical systems with data stream mining. *Computer Science and Information Systems*, *13*(2), 453–473. doi:10.2298/csis150518009o

Conference Papers

Osojnik, A. & Džeroski, S. (2014). Modeling dynamical systems with data stream mining. In *Discovery Science: Book of Abstracts* (p. 12).

Stepišnik Perdih, T., Osojnik, A., Džeroski, S., & Kocev, D. (2017). Option predictive clustering trees for hierarchical multi-label classification. In *Discovery Science (DS 2017)* (Vol. 10558, pp. 116–123). LNCS. Springer. doi:10.1007/978-3-319-67786-6_9

Biography

The author of this thesis was born on the 5^{th} of August, 1988 in Ljubljana, Slovenia. There, he also attended primary school, as well as secondary school and the International Baccalaureate program. In 2007, he enrolled in the Mathematics program at the Faculty of Mathematics and Physics of University of Ljubljana, Slovenia. He received his Bachelor's degree in Mathematics in 2011. He continued his studies in the same year by enrolling in the second cycle Mathematics program. In 2013, he successfully obtained his Master's degree by defending his Master's thesis entitled "Modeling dynamical systems with data stream mining" under the supervision of Prof. Dr. Sašo Džeroski and co-supervision of Prof. Dr. Andrej Bauer.

In 2013, he enrolled in the Information and Communication Technologies PhD program at the Jožef Stefan International Postgraduate School in Ljubljana, Slovenia, under the supervision of Prof. Dr. Sašo Džeroski and co-supervision of Asst. Prof. Dr. Panče Panov. For his studies, Osojnik was awarded a Young Researcher grant from the Slovenian Research Agency. His research work at the Department of Knowledge Technologies at the Jožef Stefan Institute, Ljubljana, Slovenia was also carried out in the scopes of the European Union Commission-funded projects MAESTRA (Learning from Massive, Incompletely annotated, and Structured Data) and The Human Brain Project.

His research interests are in the field of machine learning, primarily, in the fields of methods for online learning and methods for structured output prediction. He works on combining the two types of methods to enable structured output prediction on data streams. He has published several conference and journal papers in this field. In 2015, he also received the best student paper award at the Discovery Science conference for his contribution titled "Multi-label classification via multi-target regression on data streams."