

SEMI-SUPERVISED LEARNING FOR STRUCTURED OUTPUT PREDICTION

Jurica Levatić

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

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

Evaluation Board:

Asst. Dr. Bernard Ženko, Chair, Jožef Stefan Institute, Ljubljana, Slovenia
Prof. Dr. Hendrik Blockeel, Member, KU Leuven, Leuven, Belgium
Prof. Dr. Michelangelo Ceci, Member, University of Bari Aldo Moro, Bari, Italy

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL



Jurica Levatić

SEMI-SUPERVISED LEARNING FOR STRUCTURED OUTPUT PREDICTION

Doctoral Dissertation

POLNADZOROVANO UČENJE ZA NAPOVEDOVANJE
STRUKTURIRANIH VREDNOSTI

Doktorska disertacija

Supervisor: Prof. Dr. Sašo Džeroski

Co-Supervisor: Dr. Dragi Kocev

Ljubljana, Slovenia, April 2017

Abstract

In this thesis, we are dealing with the task of semi-supervised learning (SSL) for structured output prediction (SOP). In contrast to traditional supervised machine learning methods, which use only labeled data, semi-supervised methods additionally use unlabeled data. Due to the laborious annotation procedure, labeled data are a limited asset in many real-life problems, which can hinder the predictive performance of supervised algorithms. Unlabeled data, on the other hand, are often much easier to obtain. SSL aims to exploit unlabeled data to achieve better performance than can be achieved by labeled data alone.

Structured output prediction is concerned with predicting structured, rather than scalar values, such as multiple classes/variables, hierarchies or sequences. Such (i.e., structured) outputs are encountered in many applications of predictive modeling. Compared to SSL for primitive outputs, SSL for SOP received much less attention in the scientific community, although the need for SSL is even stronger there: Obtaining labels of structured data is even harder. Furthermore, this field lacks interpretable methods and methods that can handle various SOP task.

To overcome these issues, we extend the predictive clustering (PC) framework towards SSL. The PC framework is implemented using predictive clustering trees (PCTs) which can efficiently handle various SOP tasks. In this thesis, we contribute to the field of SSL for SOP with two classes of methods stemming from the PC framework. The proposed methods can handle the following SOP tasks: multi-target regression, multi-label classification and hierarchical multi-label classification.

The first class of methods is based on the self-training paradigm – it uses its own most reliable predictions in the learning process. The second class of methods is based on the extension of the variance functions of PCTs in order for it to accommodate both labeled and unlabeled examples. We extensively evaluated the proposed methods on a wide variety of datasets from various domains and with different types of structured output.

The self-training approach offers a state-of-the-art predictive performance on multi-target regression problems, while producing black-box models and with the cost of increased computational complexity (due to iterative training of the base model) as compared to supervised random forests. Semi-supervised predictive clustering trees, on the other hand, produce readily interpretable models, which are often considerably more accurate than the corresponding supervised models for structured outputs. The semi-supervised predictive clustering trees (and ensembles thereof) also exhibit attractive predictive performance on machine learning tasks with primitive outputs.

Finally, we conduct two case studies where we demonstrate the practical usability of the proposed methods. In the first case study, we apply the self-training method for multi-target regression to predict the properties of potential antimalarial medications and show that the semi-supervised model is more accurate than the supervised model. In the second case study, we demonstrate on the problem of water quality prediction that semi-supervised PCTs can efficiently solve a practically relevant machine learning task: learning from partially labeled data.

Povzetek

V pričujoči disertaciji se ukvarjamo s polnadzorovanim učenjem za napovedovanje strukturiranih vrednosti. Metode nadzorovanega učenja med učenjem uporabljajo samo označene podatke, medtem ko metode polnadzorovanega učenja uporabljajo tudi neoznačene. Dostop do označenih podatkov je pogosto omejen zaradi težavnega postopka označevanja, kar lahko omeji napovedno moč metod nadzorovanega učenja. Po drugi strani je neoznačene podatke pogosto veliko lažje pridobiti. Cilj polnadzorovanega učenja jih je izkoristiti in s tem doseči napovedno moč, ki je z uporabo samo označenih podatkov nedosegljiva.

Pri napovedovanju strukturiranih vrednosti napovedujemo vrednosti, kot so terice, hierarhije ali zaporedja. Z napovedovanjem takih vrednosti se v strojnem učenju pogosto srečamo. Napovedovanju primitivnih vrednosti je znanstvena srenja na področju polnadzorovanega učenja namenila veliko manj pozornosti kot napovedovanju strukturiranih, čeprav je potreba po slednjem pri polnadzorovanem učenju še večja. Označevanje strukturiranih podatkov je namreč še težavnejše. Poleg tega nam primanjkuje interpretabilnih metod s tega področja ter metod, ki lahko napovedujejo različne vrste strukturiranih vrednosti.

Da bi presegli omenjene pomanjkljivosti, smo napovedno razvrščanje razširili tudi na področje polnadzorovanega učenja. Napovedno razvrščanje je implementirano s pomočjo dreves za napovedno razvrščanje, ki lahko učinkovito opravljajo različne naloge napovedovanja strukturiranih vrednosti. V pričujoči disertaciji prispevamo k področju polnadzorovanega učenja za napovedovanje strukturiranih vrednosti z dvema razredoma metod, ki izhajata iz napovednega razvrščanja. Metode, ki jih predlagamo, lahko obravnavajo naslednje naloge napovedovanja strukturiranih vrednosti: večciljno regresijo, večciljno klasifikacijo ter hierarhično večoznačno klasifikacijo.

Prvi razred metod temelji na 'samoučenju', ki uporablja svoje najzanesljivejše napovedi med procesom učenja. Drugi razred metod temelji na drevesih za napovedno razvrščanje, ki jih nadgradimo tako, da lahko izkoristijo tako označene kot tudi neoznačene podatke. Metode, ki jih predlagamo, smo temeljito ovrednotili na vrsti podatkovij, ki prihajajo iz različnih domen ter imajo različne tipe strukturiranih vrednosti.

Pristop samoučenja ponuja vrhunsko napovedno moč pri nalogah večciljne regresije, a gradi neinterpretabilne modele ter je (zaradi iterativne npravi treniranja osnovnega modela) računsko zahtevnejši kot metoda naključnih gozdov. Po drugi strani vodijo polnadzorovana drevesa za napovedno razvrščanje do interpretabilnih modelov, ki so pogosto precej natančnejši kot modeli nadzorovanih dreves za napovedno razvrščanje. Polnadzorovana drevesa za napovedno razvrščanje ter iz njih sestavljeni polnadzorovani naključni gozdovi so privlačni tudi zaradi svoje napovedne moči pri napovedovanju primitivnih vrednosti.

Na koncu izvedemo študijska primera, kjer pokažemo praktično uporabnost predlaganih metod. V prvem študijskem primeru uporabljamo metodo samoučenja, da bi napovedali lastnosti potencialnih zdravil za malarijo. Pokažemo, da je polnadzorovani model natančnejši od nadzorovanega. V drugem študijskem primeru napovedujemo kakovost vode in pokažemo, da lahko polnadzorovana drevesa za napovedno razvrščanje učinkovito rešijo pomembno nalogo strojnega učenja: učenje iz delno označenih podatkov.

Contents

List of Figures	xiii
List of Tables	xix
List of Algorithms	xxv
Abbreviations	xxvii
1 Introduction	1
1.1 Motivation	3
1.2 Goals, Hypotheses and Methodology	4
1.3 Contributions	6
1.4 Organization	7
2 Semi-Supervised Learning	9
2.1 Definition of Semi-Supervised Learning	9
2.1.1 The task of supervised learning	9
2.1.2 The task of semi-supervised learning	10
2.2 Existing Semi-Supervised Learning Approaches	10
3 Structured Output Prediction	13
3.1 Definition of Structured Output Prediction Tasks	13
3.1.1 The task of predicting multiple targets	13
3.1.2 The task of hierarchical multi-label classification	14
3.2 Related Work	15
3.2.1 Multi-target prediction	17
3.2.1.1 Multi-label classification	17
3.2.1.2 Multi-target regression	20
3.2.2 Hierarchical multi-label classification	22
4 Semi-Supervised Learning for Structured Output Prediction	25
4.1 The Task of Semi-Supervised Learning for Structured Output Prediction . .	25
4.2 Related Work	26
4.2.1 Semi-supervised methods for specific types of structured outputs . .	26
4.2.2 Semi-supervised methods for SOP	28
4.2.3 Issues within the current semi-supervised methods for SOP	29
5 Predictive Clustering	31
5.1 Predictive Clustering Trees for Structured Oustputs	31
5.1.1 PCTs for multiple target variables	32
5.1.1.1 PCTs for multiple continuous variables	32
5.1.1.2 PCTs for multiple discrete variables	33

5.1.2	PCTs for hierarchical multi-label classification	34
5.2	Ensembles for Predicting Structured Outputs	36
6	Self-Training for Predicting Structured Outputs with Tree Ensembles	39
6.1	Methodology	39
6.1.1	Reliability scores for structured outputs	40
6.1.1.1	Reliability scores for multi-target regression	41
6.1.1.2	Reliability scores for (hierarchical) multi-label classification	44
6.1.1.3	Benchmark reliability scores	44
6.1.2	Threshold selection	45
6.1.3	Stopping criteria	46
6.1.4	Computational complexity	47
6.2	Experimental Design	47
6.2.1	Data description	47
6.2.1.1	Multi-target regression datasets	47
6.2.1.2	Multi-label classification datasets	48
6.2.1.3	Hierarchical multi-label classification datasets	49
6.2.2	Experimental setup and evaluation procedure	50
6.2.3	Experimental questions	52
6.3	Result and Discussion	53
6.3.1	Multi-target regression	53
6.3.1.1	Comparison of self-training and supervised random forests	53
6.3.1.2	Analysis of performance of different reliability scores	55
6.3.1.3	The influence of the τ parameter	57
6.3.1.4	Automatic threshold selection	58
6.3.1.5	Per-target performance analysis	60
6.3.2	(Hierarchical) multi-label classification	61
6.4	Summary of the Results	63
7	Semi-Supervised Predictive Clustering Trees	65
7.1	Methodology	65
7.1.1	Semi-supervised learning of PCTs	67
7.1.1.1	Feature weighted semi-supervised PCTs	69
7.1.1.2	Computational complexity analysis	70
7.2	Experimental Design	70
7.2.1	Data description	70
7.2.1.1	Multi-target regression datasets	70
7.2.1.2	Multi-label classification datasets	71
7.2.1.3	Hierarchical multi-label classification datasets	72
7.2.2	Experimental setup and evaluation procedure	73
7.3	Results and Discussion	74
7.3.1	Analysis of predictive performance	75
7.3.1.1	Multi-target regression	75
7.3.1.2	Multi-label classification	75
7.3.1.3	Hierarchical multi-label classification	78
7.3.2	Statistical analysis of the predictive performances	80
7.3.3	Influence of the w parameter	80
7.3.4	Model size and interpretability of supervised and semi-supervised trees	83
7.3.5	Influence of the unlabeled data	86
7.4	Summary of the Results	88

8	Semi-Supervised PCTs for Primitive Outputs	91
8.1	Experimental Design	91
8.1.1	Data description	91
8.1.2	Experimental setup and evaluation procedure	92
8.2	Results and Discussion	94
8.2.1	Analysis of predictive performance	94
8.2.1.1	Binary classification	94
8.2.1.2	Multi-class classification	94
8.2.1.3	Regression	97
8.2.2	Statistical analysis of the predictive performance	97
8.2.3	Interpretability and size of the trees	99
8.3	Summary of the Results	102
9	Case Studies	103
9.1	Improving QSAR Models by Exploiting Unlabeled Data from Public Databases	103
9.1.1	Experimental design	104
9.1.1.1	Data description	104
9.1.1.2	Extraction of unlabeled data from the ChEMBL database .	105
9.1.1.3	Experimental setup and evaluation procedure	105
9.1.2	Results	105
9.1.3	Conclusions	107
9.2	Handling Partially Labeled Data in Predicting Water Quality of Agricultural Fields in Ireland	107
9.2.1	Experimental design	107
9.2.1.1	Description of the dataset	107
9.2.2	Experimental setup	109
9.3	Results	110
9.4	Conclusions	111
10	Conclusions	113
10.1	Contributions	113
10.2	Discussion	114
10.3	Further Work	115
Appendix A Complete Results of the Evaluation of the CLUS-ST Algorithm		117
A.1	Multi-Target Regression	117
A.2	Multi-Label Classification	125
A.3	Hierarchical Multi-Label Classification	128
Appendix B Additional Results of the Evaluation of the SSL-PCT Algorithm		131
B.1	Predictive Performance of Unpruned Trees	131
B.2	Model Sizes of Unpruned Trees	135
B.3	Influence of the w Parameter	136
B.4	Statistical analysis of the predictive performances of unpruned trees	143
References		145
Bibliography		161
Biography		163

List of Figures

Figure 1.1:	Semi-supervised decision boundary (continuous line), which considers both labeled and unlabeled examples, separates the two classes better than the supervised decision boundary (dashed line) which considers only labeled examples.	2
Figure 3.1:	An example of a dataset with multiple continuous targets used for modeling the condition of indigenous vegetation (Kocev, Džeroski, White, Newell, & Griffioen, 2009). The descriptive variables are obtained from a geographical information system, while the target variables are indices describing the condition of the vegetation. The illustration is taken from Kocev (2011).	14
Figure 3.2:	An example of a dataset with multiple discrete targets used for habitat modeling of bioindicator organisms (Sašo Džeroski, Demšar, & Grbović, 2000). The descriptive variables are chemical parameters of water samples, while the target variables are the abundances of 14 bioindicator organisms. The illustration is taken from Kocev (2011).	15
Figure 3.3:	An example of a dataset for hierarchical multi-label classification of medical X-ray images (Dimitrovski, Kocev, Loskovska, & Džeroski, 2008). The illustration is taken from Kocev (2011).	16
Figure 4.1:	An example of a dataset with multiple continuous targets used for modeling the condition of indigenous vegetation (Kocev, Džeroski, White, Newell, & Griffioen, 2009). The descriptive variables are obtained from a geographical information system, while the target variables are indices describing the condition of the vegetation. The dataset contains labeled, unlabeled and partially labeled examples.	27
Figure 5.1:	Example of a predictive clustering tree for predicting multiple continuous variables, which predicts several indicators of the condition of indigenous vegetation in Victoria, Australia (Kocev, Džeroski, White, Newell, & Griffioen, 2009). The tree was learned by using the data shown in Figure 3.1.	33
Figure 5.2:	Example of a predictive clustering tree for predicting multiple discrete variables, which predicts the presence of several bioindicator organisms in water samples on the basis of chemical parameters (Sašo Džeroski, Demšar, & Grbović, 2000). The tree was learned by using the data shown in Figure 3.2.	34
Figure 5.3:	Toy examples of hierarchies structured as a tree (a) and as a directed acyclic graph (c). Class label names contain information about the position in the hierarchy, e.g., $c_{2.1}$ is a subclass of c_2 . The set of classes $S_1 = \{c_1, c_2, c_{2.2}\}$, shown in bold (b) are represented as a vector (L_k). . .	35

- Figure 5.4: Example of a predictive clustering tree for predicting multiple hierarchically organized discrete variables. The tree predicts the presence of several organisms living in the Danish agricultural soils (Levatić, Kocev, Debeljak, & Džeroski, 2015). The numbers beside the taxa represent the probabilities of presence (i.e., the mean vectors of the examples in the leaf). Only the taxa present with probability greater than 0.6 are listed. 36
- Figure 6.1: The self-training approach to SOP. The left panel gives a schematic representation of the self-training approach, with all of its components. The right panel summarizes all of the possible choices we consider with respect to reliability scores, thresholding procedures, and stopping criteria. 51
- Figure 6.2: Average ranks diagrams for the performance of the supervised algorithms (CLUS-RF and MSVR) and semi-supervised self-training with various reliability scores. The percentage of labeled data varies from 1% to 50%. Each graph represents two different kinds of information: the ranking among the algorithms (the algorithms positioned at the rightmost side of each graph are the best performing) and the statistical significance of the difference between pairs of algorithms (if their distance is less than the critical distance (at $p\text{-value} = 0.05$) there is no statistically significant difference between the two). The critical distance for all diagrams is 7.14886. 54
- Figure 6.3: Average ranks for different percentages of labeled data (from 1% to 50%) of supervised methods (red lines), self-training with the proposed reliability scores (black and orange lines) and self-training with benchmark scores (blue and green lines). 55
- Figure 6.4: The distributions of per-target errors (discretized into bins) made on unlabeled examples after the initial iteration of self-training are very skewed for the target variables of the *RF1* dataset when 10% of the data are labeled. These are depicted for the first two target variables (a, b), similar distributions are observed also for the other six target variables of the *RF1* dataset. The reliability scores (discretized into bins) for unlabeled examples calculated by averaging normalized per-target errors are biased towards high reliability (c, e), while the scores calculated by averaging ranks of per-target errors are normally distributed (d, f). . . 56
- Figure 6.5: A comparison of the predictive performance of random forests (CLUS-RF) and self-training with different reliability scores (*VarianceNormAvg*, *Random* and *OracleRankAvg*) on the *RF1* dataset illustrates the possible effects of threshold selection: A too permissive threshold for the *VarianceNormAvg* reliability score leads to worse performance than random instance selection, while a too strict threshold for the *OracleRankAvg* reliability score does not allow self-training to benefit from the unlabeled data. 58
- Figure 6.6: Average ranks for different percentages of labeled data (from 1% to 50%) of supervised methods (red lines), and self-training *VarianceNormAvg* with the baseline automatic thresholding algorithms (orange and green lines) and self-training *VarianceNormAvg* with the proposed automatic thresholding algorithms (black and blue lines). 59

Figure 6.7:	Analysis of per-target performance for the RF1 dataset, in terms of difference in performance (ΔRMSE) between CLUS-RF and self-training with the <i>VarianceNormAvg</i> reliability score. Positive values suggest that self-training is better, while negative that CLUS-RF is better. Zero means that there is no difference in performance.	61
Figure 6.8:	Comparison of predictive performance of supervised random forests (black line) and self-training with <i>ClassNormAvg</i> (red line) and <i>Oracle</i> (green line) reliability scores on the task of multi-label classification (a) and hierarchical multi-label classification (b). The percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99.	62
Figure 7.1:	An illustration of the splits made by semi-supervised and supervised PCT for a (single-target) regression task. (a) A semi-supervised PCT detects 3 clusters of examples by the help of the distribution of unlabeled examples, instead of 2 clusters, as detected by the supervised PCT. (b) This leads to better predictive performance of the semi-supervised tree, since the labels of examples which are close to each other in dense regions are also close.	66
Figure 7.2:	Predictive performance of the supervised and semi-supervised methods on the multi-target regression datasets.	76
Figure 7.3:	Predictive performance of the supervised and semi-supervised methods on the multi-label classification datasets.	77
Figure 7.4:	Predictive performance of the supervised and semi-supervised methods on the hierarchical multi-label classification datasets.	79
Figure 7.5:	The effect of the w parameter on the performance of the SSL-PCT (red line) and SSL-RF (orange line) methods on 3 datasets with different types of structured outputs: Emotions (MLC), Soil quality (MTR) and Danish farms (HMLC). The values of the w parameter selected by cross-validation and used in the experimental evaluation are denoted with colored markers.	82
Figure 7.6:	Average tree size per value of the w parameter across (all datasets and amounts of labeled data).	85
Figure 7.7:	Comparison of the predictive performance of the pruned and unpruned supervised (SL-PCTs) and semi-supervised trees (SSL-PCTs) shows that SSL-PCTs are less affected by pruning since the performance of pruned and unpruned trees is very well correlated.	86
Figure 7.8:	Comparison of the predictive performance of SSL-PCT and SL-PCT ^{D+T} with the SL-PCT across all datasets and different amounts of labeled data. $\Delta\text{SSL-PCT}$ and $\Delta\text{SL-PCT}$ denote the difference between the SL-PCT and SSL-PCT or SL-PCT ^{D+T} , respectively. Negative values mean that SL-PCT is better, while positive values mean the opposite. SSL-PCT outperforms SL-PCT ^{D+T} above the diagonal (dashed line), while SL-PCT ^{D+T} outperforms SSL-PCT below the diagonal line. SSL-PCT improves over SL-PCT for positive values of the y -axis, and degrades the performance of SL-PCT for negative values of the y -axis. The same relations hold for SL-PCT ^{D+T} , but for x -axis.	87
Figure 8.1:	Accuracy of the supervised and semi-supervised methods on the binary classification datasets.	95

Figure 8.2:	Accuracy of the supervised and semi-supervised methods on the multi-class classification datasets.	96
Figure 8.3:	RRMSE of the supervised and semi-supervised methods on the regression datasets.	98
Figure 8.4:	Examples of trees obtained with the supervised SL-PCT algorithm (a and c), and the semi-supervised SSL-PCT algorithm (b and d) on the Diabetes and Cardiocotography3 datasets.	100
Figure 9.1:	Comparison of predictive performance for random forests (CLUS-RF) and semi-supervised self-training (CLUS-ST).	106
Figure 9.2:	Comparison of predictive performance of random forest (CLUS-RF) and semi-supervised self-training (CLUS-ST) with varying threshold for reliability of predictions exemplifies the possible effects of choosing a sub-optimal threshold.	106
Figure 9.3:	Venn diagram of observation points with know values for the three target variables. There are 351 examples with all three targets known, 176 examples with know values for P and Q , but not for N , and 176 examples with know values for Q , but not for P and N	108
Figure 9.4:	Predictive performance of the local and global models build from labeled and partially labeled data.	110
Figure 9.5:	Multi-target tree learned from labeled and partially labeled examples. .	111
Figure A.1:	Average ranks for different percentages of labeled data (from 1% to 50%) of supervised methods (red lines), self-training with the best reliability score based on variance (black), the best score based on random forest proximities (orange), the best benchmark score (green), and random score (blue).	125
Figure A.2:	Comparison of predictive performance for supervised random forests (black line) and self-training (red line) on MLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figure (a).	125
Figure A.2:	Comparison of predictive performance for supervised random forests (black line) and self-training (red line) on MLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figures (b) to (e).	126
Figure A.2:	Comparison of predictive performance for supervised random forests (black line) and self-training (red line) on MLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figures (f) to (i).	127
Figure A.3:	Comparison of predictive performance for supervised random forests (black line), self-training (red line) and self-training with Oracle score (green line) on HMLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figures (a) to (c).	128

Figure A.3:	Comparison of predictive performance for supervised random forests (black line), self-training (red line) and self-training with Oracle score (green line) on HMLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figures (d) to (g).	129
Figure B.1:	Predictive performance of the supervised and semi-supervised methods on the multi-target regression datasets. Single tree methods (SL-PCT, SSL-PCT and SSL-PCT-FR) are not pruned.	132
Figure B.2:	Predictive performance of the supervised and semi-supervised methods on the multi-label classification datasets. Single tree methods (SL-PCT, SSL-PCT and SSL-PCT-FR) are not pruned.	133
Figure B.3:	Predictive performance of the supervised and semi-supervised methods on the hierarchical multi-label classification datasets. Single tree methods (SL-PCT, SSL-PCT and SSL-PCT-FR) are not pruned.	134
Figure B.4:	The effect of the w parameter on the performance of the SSL-PCT (red line) and SSL-RF (orange line) methods on multi-target regression datasets. The values of the w parameter selected by cross-validation and used in the experimental evaluation are denoted with colored markers.	136
Figure B.4:	(continued)	137
Figure B.4:	(continued)	138
Figure B.5:	The effect of the w parameter on the performance of the SSL-PCT (red line) and SSL-RF (orange line) methods on multi-label classification datasets. The values of the w parameter selected by cross-validation and used in the experimental evaluation are denoted with colored markers.	138
Figure B.5:	(continued)	139
Figure B.5:	(continued)	140
Figure B.6:	The effect of the w parameter on the performance of the SSL-PCT (red line) and SSL-RF (orange line) methods on hierarchical multi-label classification datasets. The values of the w parameter selected by cross-validation and used in the experimental evaluation are denoted with colored markers.	141
Figure B.7:	(continued)	142
Figure B.8:	(continued)	143

List of Tables

Table 6.1:	An illustrative example of using the procedure for calculating of the reliability scores according to different aggregation and normalization schemes. The same examples e_1, e_2 and e_3 are used in both the upper and lower part of the Table.	42
Table 6.2:	Characteristics of the MTR datasets. N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous) and T is the number of target variables.	48
Table 6.3:	Characteristics of the MLC datasets: N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous), $\mathcal{L}$ is the number of labels, $\overline{\mathcal{L}_L}$ is the average number of labels per example.	48
Table 6.4:	Characteristics of the HMLC datasets: N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous), $\mathcal{H}$ is the type of the hierarchy, $ \mathcal{H} $ is the number of nodes in the hierarchy, $\mathcal{H}_d$ is the maximal depth of the hierarchy, $\overline{\mathcal{L}_L}$ is the average number of labels per example.	49
Table 6.5:	Results of the Wilcoxon signed-rank test applied to the performance of self-training with <i>VarianceNormAvg</i> reliability score and the performance of supervised random forest (CLUS-RF) on the 9 datasets considered in this study. In bold, we report significant p -values (< 0.05). . .	57
Table 6.6:	P-values of the Wilcoxon signed-rank test applied to the performances (RMSE) of a supervised random forest (CLUS-RF) and self-training with algorithms for automatic threshold selection ($T_OOBInitial + Airbag$ and $Tavg10\% + Airbag$) on the 9 datasets considered in this study. In bold, we report significant p -values (< 0.05). The '-' sign denotes that CLUS-RF is better, while '+' that self-training is better.	59
Table 6.7:	Average number of iterations (across the 9 dataset considered in this study) of the self-training with different algorithms for automatic reliability threshold selection and manual threshold selection (<i>VarianceNormAvg</i>).	60
Table 6.8:	Average learning time in seconds (across the 9 dataset considered in this study) of supervised methods (CLUS-RF and MSVR), self-training with different algorithms for automatic reliability threshold selection and manual threshold selection (<i>VarianceNormAvg</i>). For the MSVR algorithm, the learning time spent for parameter optimization is presented in brackets.	60
Table 7.1:	Characteristics of the datasets. N : number of instances, D/C : the number of descriptive attributes (discrete/continuous), T : number of target variables.	71
Table 7.2:	Characteristics of the MLC datasets: N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous), $\mathcal{L}$ is the number of labels, $\overline{\mathcal{L}_L}$ is the average number of labels per example.	72

Table 7.3:	Characteristics of the HMLC datasets: N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous), $\mathcal{H}$ is a type of the hierarchy, $ \mathcal{H} $ is the number of nodes in the hierarchy, $\mathcal{H}_d$ is the maximal depth of the hierarchy, $\overline{\mathcal{L}_L}$ is the average number of labels per example.	72
Table 7.4:	P -values of the Wilcoxon signed-rank test applied to the performances (AUPRC) of the supervised and semi-supervised single trees (SL-PCT, SSL-PCT and SSL-PCT-FR) on the datasets considered in this study: 12 for multi-target regression, 12 for multi-label classification and for 12 hierarchical multi-label classification. In bold, we report significant p -values (< 0.05). In comparison of two methods, i.e., Method1 vs. Method2, the '-' sign denotes that the sum of ranks where the first method outperformed the second is higher than the sum of ranks where the second method outperformed the first. The '+' sign denotes the opposite.	81
Table 7.5:	Model sizes, in terms of number of nodes, obtained with the supervised PCTs (SL-PCT) and the semi-supervised PCTs (SSL-PCT) on the 12 MTR datasets, 12 MLC datasets and 12 HMLC datasets.	84
Table 8.1:	Characteristics of the binary classification datasets. N : number of instances, D/C : number of descriptive attributes (discrete/continuous), P : percentage of the majority class.	92
Table 8.2:	Characteristics of the multi-class classification datasets. N : number of instances, D/C : number of descriptive attributes (discrete/continuous), K : number of classes, P : percentage of the majority class.	92
Table 8.3:	Characteristics of the regression datasets. N : number of instances, D/C : number of descriptive attributes (discrete/continuous).	93
Table 8.4:	P -values of the Wilcoxon signed-rank test applied to the performances of the supervised and semi-supervised algorithms. In bold, we report significant p -values (< 0.05). In comparison of two methods, i.e., Method1 vs. Method2, the '-' sign denotes that the sum of ranks where the first method outperformed the second is higher than the sum of ranks where the second method outperformed the first. The '+' sign denotes the opposite.	99
Table 8.5:	Model sizes, in terms of number of nodes, obtained with the supervised algorithm (SL-PCT) and the semi-supervised algorithm (SSL-PCT).	101
Table 9.1:	The sizes of single tree models in terms of number of nodes. For single-target models (i.e., FL-ST-PCT and PL-ST-PCT), the size represents the sum of sizes of trees for all of the three target variables.	111

Table A.1:	RMSE on the 9 considered datasets obtained with 1% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_{OOB} , $T_{OOBInitial}$, $T_{top10\%}$ and $T_{avg10\%}$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).	118
------------	---	-----

Table A.2:	RMSE on the 9 considered datasets obtained with 5% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_{OOB} , $T_{OOBInitial}$, $T_{top10\%}$ and $T_{avg10\%}$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).	119
------------	---	-----

Table A.3:	RMSE on the 9 considered datasets obtained with 10% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_{OOB} , $T_{OOBInitial}$, $T_{top10\%}$ and $T_{avg10\%}$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).	120
------------	--	-----

Table A.4:	RMSE on the 9 considered datasets obtained with 15% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $Ttop10\%$ and $Tavg10\%$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).	121
------------	--	-----

Table A.5:	RMSE on the 9 considered datasets obtained with 20% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $Ttop10\%$ and $Tavg10\%$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).	122
------------	--	-----

Table A.6:	RMSE on the 9 considered datasets obtained with 30% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $Ttop10\%$ and $Tavg10\%$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).	123
------------	--	-----

Table A.7:	RMSE on the 9 considered datasets obtained with 50% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_{OOB} , $T_{OOBInitial}$, $T_{top10\%}$ and $T_{avg10\%}$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).	124
Table B.1:	Model sizes of pruned trees, in terms of number of nodes, obtained with the supervised PCTs (SL-PCT) and the semi-supervised PCTs (SSL-PCT) on the 12 multi-target regression datasets, 12 multi-label classification datasets and 12 hierarchical multi-label classification datasets. The methods are not pruned.	135
Table B.2:	P -values of the Wilcoxon signed-rank test applied to the performances ($\overline{\text{AUPRC}}$) of the supervised and semi-supervised single trees (SL-PCT, SSL-PCT and SSL-PCT-FR) on the datasets considered in this study: 12 for multi-target regression, 12 for multi-label classification and for 12 hierarchical multi-label classification. The methods are not pruned. In bold, we report significant p -values (< 0.05). In comparison of two methods, i.e., Method1 vs. Method2, the '-' sign denotes that the sum of ranks where the first method outperformed the second is higher than the sum of ranks where the second method outperformed the first. The '+' sign denotes the opposite.	143

List of Algorithms

Algorithm 5.1:	The top-down induction algorithm for PCTs.	32
Algorithm 5.2:	The ensemble learning algorithms: bagging and random forests. Here, E is the set of the training examples, k is the number of trees in the forest, and $f(D)$ is the size of the feature subset considered at each node during tree construction for random forests.	37
Algorithm 6.1:	The learning algorithm for self-training with random forests of PCTs (CLUS-ST). Here, E_l is a set of the labeled examples, E_u is a set of unlabeled examples, k is the number of trees in the forest, D is the total number of descriptive attributes, $f(D)$ is a function which gives the size of the feature subset for random forests considered at each node during tree construction, and τ is the threshold for the reliability of predictions.	40
Algorithm 6.2:	The learning algorithm for self-training with automatic threshold selection based on out-of-bag errors (CLUS-SSL-AUTO). Here, E_l is a set of labeled examples, E_u is a set of unlabeled examples, k is the number of trees in the forest, D is the total number of descriptive attributes, $f(D)$ is a function which gives the size of the feature subset for random forests considered at each node during tree construction, and <i>initial</i> is an indicator variable specifying whether the threshold is automatically selected only after the first iteration or throughout all the iterations.	46

Abbreviations

BR	...	Binary relevance
DAG	...	Directed acyclic graph
HMLC	...	Hierarchical multi-label classification
HSC	...	Hierarchical single-label classification
<i>k</i> NN	...	K-nearest neighbors
MLC	...	Multi-label classification
MSVR	...	Multi-output support vector regression
MTC	...	Multi-target classification
MTCTs	...	Multi-target classification tree
MTDT	...	Multi-target predictive clustering tree
MTR	...	Multi-target regression
MTRT	...	Multi-target regression tree
OOB	...	Out-of-bag
PC	...	Predictive clustering
PCT	...	Predictive clustering tree
QSAR	...	Quantitative structure-activity relationship
SOP	...	Structured output prediction
SSL	...	Semi-supervised learning
SSL-PCT	...	Semi-supervised predictive clustering trees
SVR	...	Support vector regression

Chapter 1

Introduction

This thesis falls within the area of artificial intelligence (AI) (Russell, Norvig, Canny, Malik, & Edwards, 2003), or more precisely into the sub-field of AI called machine learning. Machine learning studies algorithms that can learn from data and improve with experience (Mitchell, 1997). The data typically consists of training examples (instances), while experience equates to the amount of training examples that are available to learn from. In the classical machine learning setting, the data are given in the form of a table, where each row of the table is an example and each column is an attribute describing the example.

The most common task of machine learning is *supervised learning* (or predictive modeling), where the goal is to predict the value of one attribute of the examples (called a target attribute or output) by using the values of other attributes (called descriptive or input attributes). The training examples used in supervised learning, i.e., examples with known value of the target attribute, are called *labeled examples*. The target attribute of the labeled examples can be considered as a "teacher" that guides the machine learning algorithm during the learning process. The goal is then to learn a model that correctly predicts the values of the target attribute of examples unseen during learning. Examples of supervised machine learning methods include decision trees, decision rules, support vector machines (SVMs), neural networks and Bayesian networks.

Another common machine learning task is *unsupervised learning* (or descriptive modeling) (Langley, 1996), where a target attribute does not exist and the goal is to infer interesting structures from data or to provide general descriptions of the examples. The training examples used in unsupervised learning, i.e., examples with only descriptive attributes available, are called *unlabeled examples*. Examples of unsupervised machine learning methods include cluster analysis and principal-component analysis (Bishop, 2007).

Supervised methods often need a large amount of labeled data to learn a predictive model with a satisfying predictive performance. However, in many real-life problems, such as phonetic annotation of human speech, protein 3D structure prediction, and spam filtering, only a few labeled examples are available to learn from because of the expensive and/or time-consuming annotation procedures. Contrary to labeled examples, unlabeled examples are often freely available in vast amounts. For example, human speech can be recorded from radio broadcasts; DNA sequences of proteins can be extracted from gene databases. The concept of semi-supervised learning (SSL) emerged in the 1970's as an answer to the problem of labeled data scarcity (Chapelle, Schölkopf, & Zien, 2006; Zhu & Goldberg, 2009). The idea of SSL is to get better performance by exploiting the freely/easily available information in the form of unlabeled examples.

SSL is positioned between the two previously mentioned major machine learning paradigms: supervised and unsupervised learning. Semi-supervised methods typically extend either supervised or unsupervised methods towards the other learning paradigm. In SSL,

we distinguish between semi-supervised predictive modeling (i.e., semi-supervised classification or regression) and semi-supervised (or constrained) clustering. The latter aims to improve clustering from unlabeled data alone by including additional 'supervised information' in the form of *must-link* and *cannot-link* constraints. The former uses unlabeled examples in addition to the labeled ones, in order to achieve better predictive performance than the one that can be achieved by using only labeled examples. In this thesis, we are concerned with semi-supervised predictive modeling. Figure 1.1 shows a simple example of how unlabeled data can help to induce a semi-supervised predictive model that is better in separating the classes than a supervised predictive model.

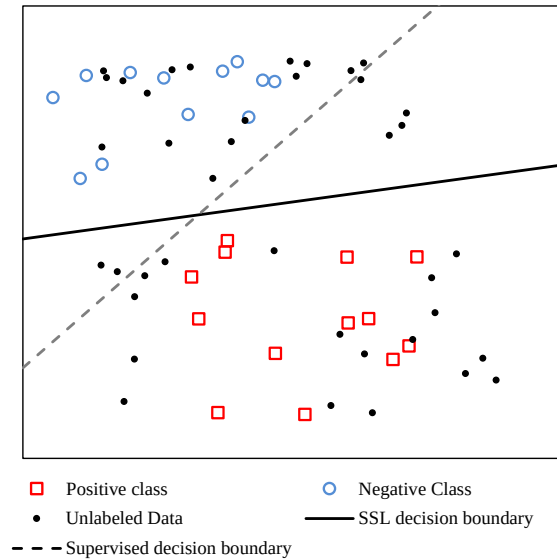


Figure 1.1: Semi-supervised decision boundary (continuous line), which considers both labeled and unlabeled examples, separates the two classes better than the supervised decision boundary (dashed line) which considers only labeled examples.

Independently from the development of SSL, the need to mine knowledge from structured data has become prominent recently (Kriegel et al., 2007, 1; Dietterich, Domingos, Getoor, Muggleton, & Tadepalli, 2008). In many applications of predictive modeling, the output (i.e., the target) values to be predicted are structured, rather than being scalar values. Such machine learning tasks are called structured output prediction (SOP) tasks. In this case, dependencies or other relations may exist between the multiple classes/target variables, which can, for example, be organized into a hierarchy or a sequence.

The main types of structured outputs encountered in the machine learning and data mining literature are: multi-target classification (where there are two or more nominal/discrete target variables), multi-target regression (where there are two or more numeric target variables), hierarchical multi-label classification (where classes are organized in a tree-shaped hierarchy or a directed acyclic graph) and time-series prediction (where multiple numeric target variables are organized in a sequence). Domains (tasks) with such types of structured output include: life sciences (gene function prediction, predicting properties of molecules), ecology (habitat and community modeling), multimedia (annotation of images and videos), and the semantic web (categorization and analysis of text and web).

The methods for predicting structured outputs can be divided into two major groups of methods: local and global methods (Bakir et al., 2007). Local methods first decompose the structured outputs into smaller, simpler components, then construct a separate model

for each of the components and combine these models to obtain a structured prediction. Global methods (also known as 'big-bang' methods) handle structured outputs 'in a single run' by constructing a single model that can predict the complete structured output.

The problem of labeled data availability is even more pronounced in SOP tasks: The labeling process is even more expensive and laborious in SOP, since several labels need to be provided for each example. For example, consider the task of quantitative structure-activity relationship (QSAR) modeling, where the goal is to predict biological effects of chemicals on the basis of their structural and/or physiochemical properties. It is very tedious and expensive to measure biological effects of chemical compounds (DiMasi, Hansen, & Grabowski, 2003). Consequentially, labeled datasets in the domain of QSAR modeling typically range from tens to hundreds and rarely surpass a thousand molecules. At the same time, information about structural and physiochemical properties of a huge amount of unlabeled compounds is freely available through public databases (e.g., ChEMBL¹ database). Another prominent example comes from the ecological modeling domain, where some attribute values are easily available (e.g. temperature, humidity) whereas some other attribute values have to be manually collected/measured by experts and thus, can be the subject of the prediction process (e.g., abundance of specific species which populate the river). Obviously, in the latter case, data collection is very expensive and time consuming, so only few observations can be obtained with limited resources available (Biber, 2013).

Considering the points given above, the development of semi-supervised methods which can handle complex output spaces has high practical utility and is, therefore, an important research topic.

1.1 Motivation

Since the establishment of SSL as a research topic, the scientific community devoted a lot of effort to developing SSL methods for the classical (primitive) data mining tasks: classification and regression (Zhu, 2008). Unfortunately, this has not been the case for SOP tasks, although the need for SSL is even stronger there.

In principle, it is possible to perform SSL for SOP by decomposing a structured problem into several (local) single-target ones, and then apply some of the available semi-supervised methods for single-target classification/regression. However, global models for SOP have been, in general, found to yield better predictive performance than local models, since they deal better with challenges inherent to structured outputs (Kocev, Džeroski, White, Newell, & Griffioen, 2009; Han, Liu, Zhao, & Wang, 2012; Tsoumakas, Spyromitros-Xioufis, Vrekou, & Vlahavas, 2014; Levatić, Kocev, & Džeroski, 2014). In particular, global models are typically more computationally efficient, produce simpler models, and overfit less (Kocev et al., 2009; Kocev, Vens, Struyf, & Džeroski, 2013; Levatić, Kocev, & Džeroski, 2014). Thus, our goal in this thesis is to develop global semi-supervised methods for SOP.

The SSL methods for SOP that are found in the literature are not interpretable and are typically focused only on a specific type of structured output (e.g., sequence learning). Furthermore, the majority of proposed methods are applied and evaluated only on one particular domain, such as text mining or image classification. Furthermore, the existing SSL methods for SOP largely deal with discrete types of outputs. There are only a few examples of SSL methods dealing with tasks involving continuous types of structured outputs (Navaratnam, Fitzgibbon, & Cipolla, 2007; Y. Zhang & Yeung, 2009; Levatić, Ceci, Kocev, & Džeroski, 2015a, 2017a), and even fewer examples of SSL methods that can deal with both continuous and discrete types of structured outputs (Gönen & Kaski, 2014; Brouard, Szafranski, & d'Alché-Buc, 2016). Finally, many existing SSL methods for

¹<https://www.ebi.ac.uk/chembl/>

SOP suffer from high computational complexity and are not applicable to large problems and problems with a large size of the output.

Taking into consideration the points discussed above, there is an apparent gap between the need for SSL methods able to efficiently deal with structured outputs (especially with continuous types thereof), and the methods currently available. In this thesis, we attempt to bridge this gap by extending the predictive clustering (PC) framework (Blockeel, 1998) towards semi-supervised learning.

The PC framework is implemented using decision trees (called predictive clustering trees or PCTs) (Blockeel, De Raedt, & Ramon, 1998; Struyf & Džeroski, 2006), decision rules (called predictive clustering rules) (Ženko, 2007), as well as ensembles thereof (Kocev et al., 2013; Aho, Ženko, Džeroski, & Elomaa, 2012). These methods use a heuristic function to split the instances into clusters, and a prototype function to determine the prediction of the model. The heuristic function used in the PC framework aims to minimize the intra-cluster variance and maximize the inter-cluster variance. By choosing appropriate variance and prototype functions, the PC framework can handle different prediction tasks, including structured output prediction. So far, the PC framework has been adapted for multi-target prediction (Struyf & Džeroski, 2006), hierarchical multi-label classification (Vens, Struyf, Schietgat, Džeroski, & Blockeel, 2008) and time-series prediction (Slavkov, Gjorgjioski, Struyf, & Džeroski, 2010).

In particular, we consider the PC framework appealing for extension to the semi-supervised setting since (1) current (supervised) methods based on predictive clustering are capable of predicting several types of structured outputs, (2) the learning of predictive models is computationally efficient, and (3) there are several options for extending this paradigm towards the semi-supervised context.

1.2 Goals, Hypotheses and Methodology

The main goals of this dissertation are as follows:

- Goal 1** Contribute to the largely unexplored field from machine learning: semi-supervised learning for structured output prediction.
- Goal 2** Extend the PC framework towards semi-supervised learning for structured output prediction.
- Goal 3** Design and implement semi-supervised methods within the PC framework which can efficiently address various types of structured outputs.
- Goal 4** Design and implement semi-supervised methods within the PC framework which will retain the interpretability of the current models in the PC framework.
- Goal 5** Improve the predictive performance of the state-of-the-art supervised methods for structured output prediction by exploiting unlabeled data.
- Goal 6** Explore the predictive performance and the characteristics of the developed semi-supervised methods on datasets from various domains.

In order to achieve the goal of semi-supervised extension of the PC framework with the above-mentioned characteristics, we consider two orthogonal approaches: (1) self-training, an iterative “wrapper” approach for existing methods in the PC framework, where the most reliable predictions on unlabeled examples are iteratively added to the training set, and (2) a distance based approach, where the variance measures used by the existing methods in the PC framework are extended to exploit both labeled and unlabeled data.

We then form the following hypotheses:

- Hypothesis 1** Existing supervised methods for structured output prediction available in the predictive clustering framework can be extended towards semi-supervised learning.
- Hypothesis 2** Semi-supervised methods based on the predictive clustering framework achieve better predictive performance than their supervised counterparts.
- Hypothesis 3** The success (i.e., the ability to outperform supervised methods) of the developed semi-supervised methods for structured output prediction depends on the domain and/or the available amount of labeled data.
- Hypothesis 4** The developed semi-supervised methods can solve new practically relevant machine learning tasks, which could not be efficiently solved by using existing methods.

In order to achieve the goals and prove the hypotheses stated above, we use the following methodology:

Iterative wrapper approach To develop the iterative “wrapper” approach, we follow the self-training methodology (Yarowsky, 1995). In self-training, the model is iteratively re-trained on the training set that is augmented at each iteration with the models’ most reliable predictions on unlabeled data, until a stopping criteria is met. When designing a semi-supervised method based on the self-training paradigm, two key questions need to be addressed: (1) how to estimate the reliability of predictions and (2) how to determine the threshold on reliability scores. The latter question corresponds to determining which unlabeled examples (together with their predicted label) should be moved at each iteration to the training set. Both of the questions are very important, since they are determinant for the success of self-training (i.e., success to outperform its base supervised method). Namely, if erroneous predictions enter the training set, mistakes can reinforce themselves throughout iterations and degrade the performance of the base method.

To estimate the reliability of structured output predictions and to determine the threshold on reliability scores, we rely on the mechanisms of ensembles of predictive clustering trees for structured outputs (Kocev et al., 2013). We use the variance of votes of individual members of the ensemble (Bosnić & Kononenko, 2008) and the out-of-bag error estimates (Breiman, 1996b) in conjunction with the random forest similarity measure between examples (Breiman, 2001). Furthermore, to develop a stopping criterion which automatically stops learning if degradation of performance is detected, we consider the airbag approach (Leistner, Saffari, Santner, & Bischof, 2009; Levatić et al., 2017a).

Distance-based approach Predictive clustering trees (Blockeel, 1998) use variance and prototype functions as an input parameter to handle several types of structured output: multi-target regression/classification (Struyf & Džeroski, 2006), hierarchical multi-label classification (Vens et al., 2008) and time-series prediction (Slavkov et al., 2010). We extend these variance functions to consider both labeled and (partially) unlabeled data. Semi-supervised predictive clustering trees implemented in such a way preserve the appealing characteristics of decision trees, such as interpretability and robustness to missing values. In order to benefit from the unlabeled data, the extension of the variance functions follow the semi-supervised smoothness assumption (Chapelle et al., 2006): *If two points x_i and x_j in a high density region of the input space are close, then also their outputs y_i and y_j should be close.*

Evaluation The performance of semi-supervised methods is domain-dependent (Chawla & Karakoulas, 2005); therefore, we evaluate the performance of the proposed semi-supervised methods on a number of datasets from various domains, such as ecology, economy and chemoinformatics. We use various amounts of labeled data for both the supervised and semi-supervised methods to explore the influence of the amount of labeled data on the predictive performance. Additionally, we perform case studies to explore the ability of the proposed semi-supervised methods to solve practically relevant tasks specific to the problems with structured outputs: Learning from examples with partially labeled output.

Implementation All of the semi-supervised methods are implemented within the open source machine learning system – CLUS (<https://sourceforge.net/projects/clus/>).

1.3 Contributions

In this thesis, we propose two classes of semi-supervised algorithms for structured output prediction that address the issues of currently available semi-supervised methods for SOP. The methodology presented in this thesis was partially published in several conference and journal publications (Levatić, Ceci, Kocev, & Džeroski, 2014; Levatić, Supek, & Džeroski, 2015; Levatić et al., 2015a, 2015b, 2017a, 2017b; Levatić, Kocev, Ceci, & Džeroski, 2017). The complete list of papers related to this thesis is given in the Bibliography section. The main contributions of the work presented in this thesis are summarized as follows:

Contribution 1 *The first semi-supervised method for multi-target regression based on self-training.*

As we previously mentioned, semi-supervised methods for continuous types of structured outputs are particularly scarce. We propose a self-training method for multi-target regression based on ensembles of PCTs, which is, to the best of our knowledge, one of the very few general purpose semi-supervised methods for this type of structured output.

The main component needed for self-training is the reliability scoring function, which estimates the reliability of prediction on unlabeled data. However, to the best of our knowledge, reliability estimates in the context of multi-target regression were not studied before. We propose several reliability scoring functions for multi-target regression predictions, based on mechanisms provided within the ensemble learning paradigm. Furthermore, we propose benchmark reliability scores, and use them to empirically determine the upper bounds on the performance of SSL for multi-target regression with self-training.

Selection of the threshold on reliability scores is an unresolved issue in self-training – even for primitive outputs. Thus, we propose an algorithm for automatic threshold selection in the context of self-training for SOP, which provides a practical solution for exploiting the potential of SSL with self-training.

Contribution 2 *A general semi-supervised predictive clustering framework for structured output prediction*

We propose an extension of PCTs towards semi-supervised learning, based on the novel variance measures which exploit both labeled and unlabeled data. The proposed extension of PCTs resolves several previously stated issues of the currently available semi-supervised methods for SOP. First of all, it is general in terms of the type of the structured output: It can support the tasks of predicting multiple continuous targets, predicting multiple discrete targets, and hierarchical multi-label classification. Next, it is computationally efficient and

can produce interpretable models. Finally, it represents a step towards the desired semi-supervised method, which would always guarantee improvement over supervised methods (or at least equal performance). Namely, the proposed method very seldom degrades the performance of supervised predictive clustering trees, due to its 'safety' mechanism.

Contribution 3 *Extensive empirical evaluation of the proposed methods*

The extensive empirical evaluation is a contribution orthogonal to the other contributions. Namely, we perform it for all of the proposed semi-supervised methods over a variety of benchmark datasets from various domains, including ecology, economy, gene function prediction, QSAR modeling and image and text processing. We compare the predictive performance of the proposed methods with state-of-the-art supervised methods for SOP. We show that self-training for multi-target regression can outperform supervised random forests and multi-output support vector machines. Furthermore, we show that the distance-based approaches can outperform by a large degree supervised PCTs and ensembles thereof on machine learning tasks with structured outputs as well as on tasks with primitive outputs (i.e., classification and regression).

We investigate the influence of the amount of labeled data on the performance of the proposed methods. Next, we extensively discuss the characteristics of the proposed approach from several viewpoints related to their practical usability, such as, computational complexity, model size and sensitivity to parameters.

Contribution 4 *Case studies in two practically relevant domains*

We apply the proposed semi-supervised methods for SOP in two practically relevant domains: QSAR modeling and ecology. Each application provides a contribution to the respective domain.

In the first case study, we propose a novel approach for building QSAR models: The use of public databases of drug-like molecules as a source of unlabeled data for QSAR modeling with semi-supervised learning algorithms. We use the proposed self-training method for multi-target regression to develop a QSAR model for predicting properties of potential antimalarial medications. We show that the developed model yields better predictive performance as compared to the model built with supervised random forests.

In the second case study, we demonstrate on the problem of water quality prediction, that the proposed distance-based approach can efficiently solve a new practically relevant machine learning task: learning from partially labeled data. We show that better predictive performance can be achieved if partially labeled data are exploited by semi-supervised predictive clustering trees, rather than discarded.

1.4 Organization

This introductory chapter presents the general context of the thesis, gives motivation for performing the research, states the main goals and hypotheses and lists the main scientific contributions. The rest of the thesis is organized as follows.

Chapter 2 defines the task of semi-supervised learning and gives an overview of existing approaches for primitive machine learning tasks, such as classification and regression.

Chapter 3 first defines the SOP tasks that we consider in this thesis, namely predicting multiple targets and hierarchical multi-label classification. The second part of the chapter gives an overview of supervised machine learning approaches for SOP.

Chapter 4 defines the task of semi-supervised learning for structured output prediction and gives an overview of the research on SSL for SOP related to this thesis.

Chapter 5 describes the relevant background to this thesis. Namely, it describes supervised PCTs, which are the basis of the semi-supervised methods proposed in this thesis. First, extensions of PCTs for the machine learning tasks considered in this thesis are described, followed by a description of ensembles of PCTs for predicting structured outputs.

Chapters 6 and 7 present the main contributions of this thesis: The two classes of semi-supervised methods for structured output prediction. Chapter 6 presents the semi-supervised method based on the self-training paradigm, while Chapter 7 presents semi-supervised methods based on extensions of variance functions of PCTs. Both chapters are organized as follows. First the proposed method is presented in detail, then the experimental design is described, followed by a presentation of the experimental evaluation and summarization of results.

Chapter 8 presents the empirical evaluation of the semi-supervised PCTs on the machine learning tasks of binary classification, multi-class classification and regression. These tasks have (i.e., a single discrete or numeric value is predicted), however, semi-supervised predictive clustering trees can be readily applied to them.

Chapter 9 presents two case studies which demonstrate the application of the developed methods on practically relevant domains. The first case study is from the domain of QSAR modeling, while the second is from the domain of ecology.

Finally, Chapter 10 concludes the dissertation. It presents a summary of the thesis, its original contributions and directions for further work.

Chapter 2

Semi-Supervised Learning

In the first part of this chapter, we establish a formal definition of semi-supervised learning for traditional (i.e., primitive) machine learning tasks, i.e., classification and regression. We proceed with giving an overview of existing semi-supervised approaches for these tasks.

2.1 Definition of Semi-Supervised Learning

In order to establish a definition of semi-supervised learning, let us first define the task of *supervised* learning.

2.1.1 The task of supervised learning

Given:

- A descriptive (or input) space $X = X_1 \times X_2 \times \dots \times X_D$ spanned by D descriptive variables that consist of values of primitive data types (boolean, discrete or continuous).
- A target (or output) attribute Y that can be either continuous or discrete.
- A set of examples $E = \{(x_i, y_i) \mid x_i \in X, y_i \in Y, 1 \leq i \leq N\}$, where each example $e_i \in E$ is a pair consisting of one element from the descriptive space and one element from the target attribute (i.e., the example's label), and N is the number of examples.
- A quality criterion q , which rewards models with high predictive accuracy (or low predictive error).

Find: A function $f : X \rightarrow Y$ such that f maximizes q .

If the output variable Y is discrete, the learning task is called *classification*, while if Y is continuous, it is called *regression*.

As previously discussed, machine learning algorithms learn and improve with experience. In traditional supervised learning, the algorithms learn from labeled examples, i.e., from the set E . However, in practice, labels may be difficult to obtain, thus E may be small. On the other hand, unlabeled examples are often abundant and are easily available. The semi-supervised algorithms combine both labeled and unlabeled data in order to build better predictive models. We define the task of semi-supervised learning as follows.

2.1.2 The task of semi-supervised learning

Given:

- A descriptive (or input) space $X = X_1 \times X_2 \times \dots \times X_D$ spanned by D descriptive variables that consist of values of primitive data types (boolean, discrete or continuous).
- A target (or output) attribute Y , that can be either continuous or discrete.
- A set of *labeled examples* $E_l = \{(x_i, y_i) \mid x_i \in X, y_i \in Y, 1 \leq i \leq N_l\}$, where each example is a pair consisting of one element from the descriptive space and one element from the target attribute (i.e., the example's label), and N_l is the number of labeled examples.
- A set of *unlabeled examples* $E_u = \{x_i \mid x_i \in X, 1 \leq i \leq N_u\}$, which consists only of elements from the descriptive space, and N_u is the number of unlabeled examples.
- A quality criterion q , which rewards models with high predictive accuracy (or low predictive error).

Find: A function $f : X \rightarrow Y$, by using both E_l and E_u , such that f maximizes q .

Note that, even though the same quality criterion q is optimized in the previously defined supervised learning tasks and the semi-supervised learning task, different training sets are used: semi-supervised learning exploits unlabeled examples E_u , in addition to the labeled ones E_l . Typically, in applications of semi-supervised learning, the number of unlabeled examples is much greater than the number of labeled examples, i.e., $N_u \gg N_l$. The semi-supervised method is considered successful if it learns a model that outperforms the model learned by its supervised counterpart from E_l only.

In semi-supervised learning, there are two different settings: inductive and transductive semi-supervised learning. Note that, in supervised learning, all the examples in the training set are labeled; therefore, we are always interested in predictive performance on a separate test set. In semi-supervised learning, on the other hand, the training set contains unlabeled examples. Thus, we can have two different goals. The first is the same as in supervised learning, i.e., to predict the labels of new unseen examples. The second is to predict only the labels of unlabeled examples in the training set. The former is called *inductive learning*, while the latter is called *transductive learning*. Inductive semi-supervised methods are more versatile than the transductive methods, since they can make predictions on both the unlabeled data used during learning, and new unseen examples. As discussed below, some of the semi-supervised methods are inherently transductive, e.g., graph based methods, which cannot make prediction on unseen data, without retraining the model. This somewhat limits the practical usability of transductive methods.

2.2 Existing Semi-Supervised Learning Approaches

In order for SSL to work, the knowledge contained in the unlabeled data has to carry some information about the class labels, i.e., the output attribute. If this prerequisite is fulfilled, we can draw on unlabeled data by making certain assumptions about the behavior of labels with respect to the structure of unlabeled data. Within SSL, many different methods have been proposed. They are usually grouped on the basis of the assumption(s) they implement as follows (Zhu, 2008): low-density separation methods, graph-based methods, generative models and self-training alike approaches. These four groups of semi-supervised

methods are described in the remainder of this section. The last group of methods, i.e., self-training, is more thoroughly described, since this approach is used in the method proposed in Chapter 6.

Low-density separation methods These methods assume that the decision boundary should lie in a low-density data region. Semi-supervised SVMs are a typical representative of this group (Bennett & Demiriz, 1999; Joachims, 1999; Ceci, 2008). In supervised SVMs only the labeled data is used to find a maximum margin decision boundary, while semi-supervised SVMs try to find the decision boundary which has the maximum margin on both the labeled and unlabeled data. However, finding the exact solution for semi-supervised SVMs is NP-hard (Zhu, 2008), therefore, current implementations, either calculate an approximate solution (Chapelle, Sindhwani, & Keerthi, 2008), or cannot handle more than a few hundred unlabeled examples. Recently, Ding, Zhu, and Zhang (2015) gave a more detailed overview of the research on semi-supervised SVMs.

Graph-based methods Graph-based methods (Zhu, 2005; C. Zhang & Wang, 2009) use unlabeled data as an additional source of information to infer the structure of a graph. These methods typically implement the so-called manifold assumption, which assumes that the (high-dimensional) data lie on a low-dimensional manifold. The graph can be considered as a low dimensional representation of the (high dimensional) data. Labels are propagated through the graph assuming label smoothness over the graph, i.e., points that are close along the manifold are assumed to have similar labels.

Graph-based methods are inherently transductive (i.e., they do not build a model for the whole space, but only infer the labels of the test set). Graph-based SSL is one of the most active research areas in the SSL community, but it is still facing open questions, such as: how to automatically select an appropriate pair-wise similarity measure and how to handle large scale datasets. Namely, the main problem of these methods is their high computational complexity, which is often cubic in the number of examples. Recently, some alternative and more efficient solutions were proposed (Liu, Wang, & Chang, 2012).

Generative models Generative methods (Zhu, 2008) assume a probabilistic model of the data and use unlabeled, together with labeled data, to estimate the most probable model parameters. Such methods have also been investigated in the multi-relational data mining setting (Malerba, Ceci, & Appice, 2009; Ceci et al., 2012). The success of generative models depends largely on choosing a probabilistic model which is appropriate for the data. Once the probabilistic model is chosen (e.g., Gaussian mixture models), a maximum likelihood estimate of the parameters can be calculated (e.g., by using the Expectation-Maximization algorithm (Moon, 1996)), followed by a calculation of class distributions using the Bayes' rule.

Self-training methods Self-training (or self-labeling) is an iterative approach that, in addition to labeled data, uses unlabeled data and predictions of their labels in the learning process. Namely, at each iteration, unlabeled examples with reliable predictions of their outputs are selected and added to the training set, followed by re-training of the base method. This process is iterated until a stopping criteria is met. For example, until no unlabeled example is added to the training set.

Semi-supervised methods based on the self-training paradigm assume that the most reliable predictions on unlabeled examples are correct. Thus, the most important ingredient of self-training is a reliability scoring function, which should be able to discern correct from wrong prediction. However, defining a good reliability scoring function is not an easy task;

therefore, the main pitfall of the self-training is reinforcement of mistakes. Namely, if a wrong prediction enters the training set, such a mistake can reinforce itself in the next iterations, leading to a degradation of the performance of the base supervised method.

The self-training paradigm is very popular in semi-supervised learning due to practical reasons: It can be "wrapped" around any (supervised) learning algorithm. That is, any supervised method can be turned into a semi-supervised method with the help of self-training, given that reliability of predictions of that method can be estimated¹.

Self-training was first proposed by Yarowsky (1995) for word sense disambiguation. Since then, it has had a number of successful applications, including detection of objects in images (Rosenberg, Hebert, & Schneiderman, 2005; Leistner et al., 2009), identification of subjective nouns (Riloff, Wiebe, & Wilson, 2003) and learning human motion over time (Bandouch, Jenkins, & Beetz, 2012). Abney (2004) and Haffari and Sarkar (2007) performed a theoretical analysis of the self-training algorithm. They showed that the self-training framework minimizes the same objective function of the base model (but in a different way). In the specific case considered by the authors, it minimizes a cross-entropy based objective function, given that the base model reduces the same objective function.

An overwhelming majority of studies relying on self-training to perform SSL are concerned with the task of classification. However, there is a handful of methods based on self-training (or co-training) for the task of (single-target) regression (Brefeld, Gärtner, Scheffer, & Wrobel, 2006; Z.-H. Zhou & Li, 2007; Appice, Ceci, & Malerba, 2009; M.-C. Yang & Wang, 2013).

Methods based on the self-training paradigm come in many different flavors, such as co-training (Blum & Mitchell, 1998) and tri-training (Z.-H. Zhou & Li, 2005). The former approach trains models on independent attribute sets (i.e., views) to avoid biasing the models to their own prediction errors. The latter approach does not require independent views, but instead it uses three different classifiers. Each classifier receives predictions for which the other two classifiers agree. Triguero, García, and Herrera (2015) recently gave a detailed overview of self-training methods (and methods alike) and performed an extensive empirical comparison of them.

Related to self-training is *active learning* (Freund, Seung, Shamir, & Tishby, 1997) which tries to minimize the labeling effort by selecting unlabeled examples that are the most valuable to the algorithm to learn from. Contrary to self-training, active learning does not search for the most reliable predictions, but for the unlabeled examples which are the most problematic for the base method. Such examples are then posed as queries to an expert that provides correct labels for them. Hybrid approaches that combine active learning and self-training were also proposed (Farouk Abdel Hady & Schwenker, 2010).

¹Self-training which operates without reliability scoring function was also proposed by Culp and Michailidis (2003)

Chapter 3

Structured Output Prediction

Traditionally, machine learning is mostly concerned with predicting a single class (i.e., classification) or a scalar number (i.e., regression). However, in many practical applications of machine learning, the output has an inherent structure. For example, in gene function prediction, the output is structured as a hierarchy (Schietgat et al., 2010). Due to its practical importance, the task of structured output prediction is gaining more and more attention within the machine learning community. In fact, "mining complex knowledge from complex data" has been recognized as one of the main challenges of machine learning (Q. Yang & Wu, 2006; Kriegel et al., 2007, 1; Dietterich et al., 2008).

In general, an arbitrary structure can be imposed on an output (Bakir et al., 2007). In the machine learning literature, several types of structured outputs prediction tasks are encountered, such as time-series prediction (Slavkov et al., 2010), label sequence learning (Brefeld & Scheffer, 2006), multi-target prediction (Struyf & Džeroski, 2006) and hierarchical multi-label classification (Vens et al., 2008). In this thesis, we are concerned with the latter two tasks, namely the task of predicting multiple targets and hierarchical multi-label classification. In this chapter, we first formally define these structured output prediction tasks, and then give an overview of the related work.

3.1 Definition of Structured Output Prediction Tasks

Here, we formally define machine learning tasks for the types of structured outputs considered in this thesis: predicting multiple targets and hierarchical multi-label classification. We follow the suggestions from Sašo Džeroski (2007) where predictive modeling is defined over arbitrary types of input and output data.

3.1.1 The task of predicting multiple targets

In multi-target prediction problems (also referred to as multi-objective or multi-output prediction) the task is to predict several discrete or continuous values. If the targets are discrete, the task is called multi-target classification (MTC), or multi-label classification (MLC) if the targets are binary (Tsoumakas, Katakis, & Vlahavas, 2009). If the targets are continuous, the task is called multi-target (or multi-variate) regression (MTR) (Borchani, Varando, Bielza, & Larrañaga, 2015). We define the task of predicting multiple targets as follows:

Given:

- A descriptive (or input) space $X = X_1 \times X_2 \times \dots \times X_D$ spanned by D descriptive variables that consist of values of primitive data types (boolean, discrete or continuous).

- A target (or output) space $Y = Y_1 \times Y_2 \times \dots \times Y_T$ spanned by T target variables that can be either continuous or discrete.
- A set of examples $E = \{(x_i, y_i) \mid x_i \in X, y_i \in Y, 1 \leq i \leq N\}$, where each example $e_i \in E$ is a pair consisting of one element from the descriptive space and one element from the target space (i.e., the example's label), and N is the number examples.
- A quality criterion q , which rewards models with high predictive accuracy (or low predictive error).

Find: A function $f : X \rightarrow Y$ such that f maximizes q .

Illustrative examples of datasets for the tasks of multi-target regression and multi-target classification are presented in Figures 3.1 and 3.2, respectively.

Site ID	Descriptive variables					Target variables						
	LandCover	TempRange	NativeTreeProb	GrassProb1Ha_RegionStdDev	...	Large tree score	Tree canopy score	Understorey score	Litter score	Logs score	Weeds score	Recruitment score
ID1	2.0	225.0	0.0	1.79	...	7	3	15	3	5	15	5
ID2	4.0	278.0	2.0	12.31	...	6	3	10	5	5	13	3
ID3	8.0	191.0	54.0	6.52	...	0	5	10	5	0	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 3.1: An example of a dataset with multiple continuous targets used for modeling the condition of indigenous vegetation (Kocev, Džeroski, White, Newell, & Griffioen, 2009). The descriptive variables are obtained from a geographical information system, while the target variables are indices describing the condition of the vegetation. The illustration is taken from Kocev (2011).

3.1.2 The task of hierarchical multi-label classification

The hierarchical multi-label classification (HMLC) denotes classification problems where each example can have more than one class (multiple labels) and classes are organized in a hierarchical structure, i.e., an example that belongs to some class automatically belongs to all its superclasses (C. Silla & Freitas, 2011; Vens et al., 2008). Closely related to HMLC is hierarchical classification, where classes are also organized in a hierarchy, however each example can be labeled with only one class (i.e., one path in the hierarchy). The most prominent areas of application of HMLC are gene/protein function prediction and text categorization, followed by music genre, image, speech and phoneme classification (C. Silla & Freitas, 2011). We formally define the task of HMLC as follows:

Given:

- A descriptive (or input) space $X = X_1 \times X_2 \times \dots \times X_D$ spanned by D descriptive variables that consist of values of primitive data types (boolean, discrete or continuous).

Sample ID	Descriptive variables						Target variables													
	Temperature	K ₂ Cr ₂ O ₇	NO ₂	Cl	CO ₂	...	<i>Cladophora</i> sp.	<i>Gongrosira incrustans</i>	<i>Oedogonium</i> sp.	<i>Stigeoclonium tenue</i>	<i>Melosira varians</i>	<i>Nitzschia palea</i>	<i>Audouinella chalybea</i>	<i>Erpobdella octoculata</i>	<i>Gammarus fossarum</i>	<i>Baetis rhodani</i>	<i>Hydropsyche</i> sp.	<i>Rhyacophila</i> sp.	<i>Simulim</i> sp.	<i>Tubifex</i> sp.
ID1	0.66	0.00	0.40	1.46	0.84	...	1	0	0	0	0	1	1	0	1	1	1	1	1	1
ID2	2.03	0.16	0.35	1.74	0.71	...	0	1	0	1	1	1	1	0	1	1	1	1	1	0
ID3	3.25	0.70	0.46	0.78	0.71	...	1	1	0	0	1	0	1	0	1	1	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 3.2: An example of a dataset with multiple discrete targets used for habitat modeling of bioindicator organisms (Sašo Džeroski, Demšar, & Grbović, 2000). The descriptive variables are chemical parameters of water samples, while the target variables are the abundances of 14 bioindicator organisms. The illustration is taken from Kocev (2011).

- A target space Y , defined with a hierarchy of classes $(C, \leq_h)$, where C is a set of classes and $\leq_h$ is a partial order among them, representing the superclass relationship, i.e., $\forall c_1, c_2 \in C, c_1 \leq_h c_2$ if and only if c_1 is a superclass of c_2 .
- A set of examples $E = \{(x_i, Y_i) \mid x_i \in X, Y_i \subseteq C, 1 \leq i \leq N\}$, where each example $e_i \in E$ is a pair of a tuple x_i from the descriptive space and a set S_i from the target space, and each set satisfies the hierarchy constraint, i.e., $c \in S_i, c' \in C, c' \leq_h c \Rightarrow c' \in Y_i$, and N is the number of examples.
- A quality criterion q , which rewards models with high predictive accuracy (or low predictive error).

Find: a function $f : X \rightarrow 2^C$ (where 2^C is the power set of C) such that f maximizes q and the predictions made by f satisfy the hierarchy constraint, i.e., $c \in f(x), c' \in 2^C, c' \leq_h c \Rightarrow c' \in f(x)$.

Figure 3.3 presents an illustrative example of the dataset for HMLC from the domain of annotation of medical X-ray images. The hierarchy there is tree-shaped and represents a taxonomy of body parts and organs that can be identified in X-ray images. The hierarchy can also be in the form of a directed acyclic graph (DAG), as it is the case, for example, in the task of gene function prediction (Schietgat et al., 2010).

3.2 Related Work

The methods for predicting structured outputs can be grouped into local and global methods (Bakir et al., 2007; C. Silla & Freitas, 2011). The local methods are also known as problem transformation methods. Such methods decompose the structured output into smaller components, then use standard classification/regression machine learning algorithms to construct a separate model for each of the components. The predictions of these models are then combined to obtain a structured prediction. The advantage of such an approach is that it enables the use of traditional learning methods that are otherwise not capable of dealing with structured outputs. The main drawback of this approach is that it

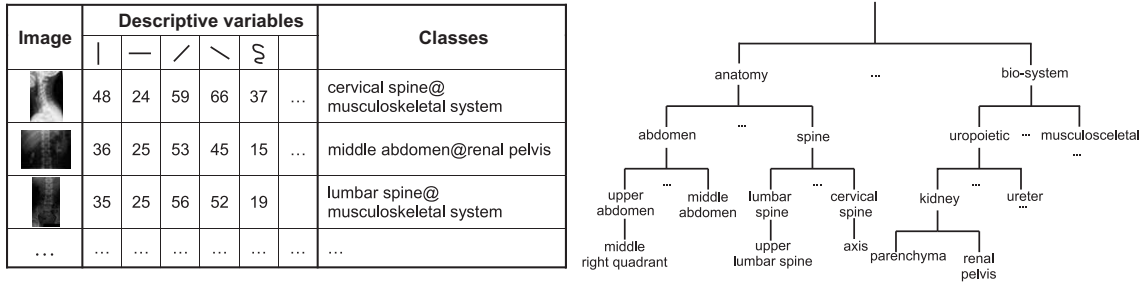


Figure 3.3: An example of a dataset for hierarchical multi-label classification of medical X-ray images (Dimitrovski, Kocev, Loskovska, & Džeroski, 2008). The illustration is taken from Kocev (2011).

ignores the structure of the output, which may affect the predictive performance (Borchani et al., 2015).

The global methods, on the other hand, handle structured outputs 'in one go' by constructing a single model that can predict the complete structured output. Such methods are also known as the 'big-bang' methods or the algorithm adaptation methods. The latter naming comes from the fact that the global methods often extend standard machine learning methods so that they can handle structured outputs. Global methods have several advantages over local methods. First, they exploit the dependencies defined (or implied) by the structure of the output (e.g., hierarchical structure of the class), which can result in better predictive performance (Kocev et al., 2009; Han et al., 2012; Tsoumakas et al., 2014; Levatić, Kocev, & Džeroski, 2014). Next, local methods require building a separate model for each component of the structured output, which can easily amount to several hundreds of models (e.g., hierarchies in functional genomics have several hundreds of nodes) and quickly becomes computationally expensive or even infeasible. Additionally, global methods are usually easier to interpret since they often produce smaller models, i.e., models whose size is smaller than the sum of the sizes of all 'local' models. Due to all of the listed advantages, in this thesis we aim to develop the global semi-supervised methods for structured output prediction.

The methods designed for predicting a specific type of structured output are predominant in the literature. However, methods that can handle various types of structured outputs can also be found. PCTs (Blockeel et al., 1998; Kocev et al., 2013) (and ensembles thereof) fall into this category. They have been implemented and evaluated on a number of SOP tasks, namely multi-target prediction (Struyf & Džeroski, 2006), multi-label classification (Madjarov, Kocev, Gjorgjevikj, & Džeroski, 2012), hierarchical multi-label classification (Vens et al., 2008) and time-series prediction (Slavkov et al., 2010). By defining appropriate variance and prototype functions, PCTs can also be extended to handle arbitrary types of structured outputs.

Next, maximum-margin methods (such as SVMs) (Ricci & Perfetti, 2008) and Conditional Random Fields (Lafferty, McCallum, & Pereira, 2001) are prominent methods used for SOP. There, versatility in terms of structured outputs is achieved by defining an output-specific loss function which measures the difference between structures. This is similar to defining an output-specific variance and prototype functions in PCTs (see Chapter 5), but these methods, contrary to PCTs, do not produce interpretable models and often have high computational complexity. Furthermore, these methods are mostly tested on discrete types of structured outputs, such as trees, sequences or alignments (Joachims, 2006). Rarely, the methods for SOP are tested on both the continuous and discrete outputs.

Geurts, Wehenkel, and D'Alché-Buc (2006) proposed to use a kernel trick to generalize

the decision tree algorithm to any type of structured output for which a kernel can be defined. The kernel is defined via a mapping that projects elements from the output space to a Hilbert space. The variance can then be calculated in this space, and a standard top-down tree induction algorithm (Breiman, Friedman, Olshen, & Stone, 1984) can be used to grow trees. This approach is similar to PCTs, where, to define a variance function, a distance between elements of structured output is needed. However, contrary to PCTs, in output kernel trees, obtaining predictions for new examples is not straightforward, since it requires the pre-image of the kernel. Calculation of the pre-image can be intractable, thus Geurts et al. (2006) propose approximation techniques.

In the remainder of this section, we give an overview of related work on predicting the structured outputs defined earlier in this chapter, namely, multi-target prediction and hierarchical multi-label classification.

3.2.1 Multi-target prediction

Closely related to multi-target prediction is *multi-task learning* (Caruana, 1997). There, multiple tasks that share a representation are learned together, with the goal to improve models by transferring knowledge between related tasks. This paradigm includes predicting a single variable from multiple input spaces, predicting multiple variables from multiple input spaces, and predicting multiple variables from a single input space. In this thesis, we are concerned with the last task, which we refer to as *multi-target prediction*. Most of the methods from the literature can handle either multiple discrete variables (multi-label classification) or multiple continuous variables (multi-target regression), thus we review methods for these two tasks separately.

3.2.1.1 Multi-label classification

Multi-label classification (MLC) is a very active field of research. For example, according to the Scopus¹ search engine, 262 articles were published on the topic of MLC in the computer science field in 2016, while, in the same year, only 13 were published on the topic of MTR. Here, we give an overview of the established MLC methods and the most recent advances in the field. For additional details on MLC, we refer the reader to several review papers published recently (Tsoumakas et al., 2009; M.-L. Zhang & Zhou, 2014; Gibaja & Ventura, 2015). Local (or problem transformation methods) for MLC are usually grouped as follows (Tsoumakas et al., 2009): binary relevance methods, label power-set methods and binary pairwise methods.

Binary relevance Binary relevance (BR) is a simple approach, where the multi-label problem is transformed into several binary classification problems in a *one-versus-all* manner. BR was criticized (T. Zhou, Tao, & Wu, 2012) since it disregards dependencies between labels, it may cause sample imbalance after transformation, and it may significantly increase the computational time in high dimensional output spaces. Read, Pfahringer, Holmes, and Frank (2011), on the other hand, highlighted the advantages of BR. First, it scales linearly with the number of labels, while some methods have much higher computational complexity (e.g., the label power-set methods). Second, since in BR all the labels are modeled independently, they can be easily added or removed without affecting the rest of the model.

¹<https://www.scopus.com/>

Classifier chains Related to BR are *classifier chains* (Read et al., 2011; Enrique Sucar et al., 2014; B. Chen, Li, Zhang, & Hu, 2016), where again the multi-label problem is transformed into several binary classification problems, however the classifiers are linked into a chain as follows. The training set of the i^{th} classifier in a chain predicting the i^{th} label is augmented with predictions of labels coming from preceding classifiers in a chain. Recently, Alali and Kubat (2015) proposed the PRUDENT method, which stacks two layers of classifiers. The first layer corresponds to the BR paradigm, while classifiers from the second layer receive as inputs the outputs of the models from the first layer. The method also prunes irrelevant label dependencies and uses confidence scores to reduce error propagation from the first to the second layer. The PRUDENT method was shown to perform favorably as compared to BR and classifier chains. Classifier chains face scalability issues, thus Read, Martino, Olmos, and Luengo (2015) presented an alternative that does not model a full chain and can scale to thousands of labels – the *classifier trellis*.

Label power-set methods The *label power-set* methods (Boutell, Luo, Shen, & Brown, 2004; Read, Pfahringer, & Holmes, 2008; Tsoumakas & Vlahavas, 2007) cast a MLC problem into multi-class classification problem by creating a new meta-class for each possible subset of labels that appears in the training set. An obvious limitations of this approach is that it can generate a very high number of meta-classes (i.e., up to $2^T - 1$). Furthermore, the label power-set methods have a tendency to over-fit, since they can only learn combinations of labels that are observed in the training data (Read et al., 2008; Tsoumakas & Vlahavas, 2007). This approach typically works well with a small number of labels, but it quickly deteriorates as the number of labels increases (Cheng & Hüllermeier, 2009). To resolve this issue, Read (2008) proposed a pruned power-set problem transformation, where only sets of labels that occur more than a predefined number of times are transformed to meta-classes. Tsoumakas, Katakis, and Vlahavas (2008) proposed an efficient label power-set method named HOMER, where a hierarchy of labels is first constructed, followed by building a classifier for the label sets in each node of the hierarchy.

Binary pairwise approaches In the *binary pairwise* approach, a binary classification model is trained for each (ordered) pair of labels. Each of the $T(T - 1)/2$ classifiers is trained by using instances labeled with the first label of the pair as positive examples, and instances labeled with the second label as negative examples. To obtain predictions for new examples, the majority voting algorithm is used. Examples are then sorted according to their sum of votes, followed by applying a label ranking algorithm to predict the relevant labels. The main drawback of this approach is quadratic complexity in terms of labels. The binary pairwise approach is naturally more appropriate for learning pairwise preferences (Read et al., 2011), but has also been adapted to MLC by Fürnkranz, Hüllermeier, Loza Mencía, and Brinker (2008) who proposed *calibrated label ranking*. Park and Fürnkranz (2007) proposed the quick weighted voting, which is a more effective voting strategy than the majority voting used in calibrated label ranking. The quick weighted voting was adapted to MLC by Loza Mencía, Park, and Fürnkranz (2010). Madjarov, Gjorgjevikj, and Džeroski (2012) proposed a two-stage architecture for learning multi-label classifiers, the first layer consists of BR models, while the second consists of pairwise models.

Almost all popular (single-target) classification algorithms were adapted for MLC.

K-nearest neighbors M.-L. Zhang and Zhou (2007) were the first to propose a multi-label k -nearest neighbors (k NN) algorithm, named ML- k NN. They used maximum a posteriori principle on the label sets of nearest neighbors to predict the labels of a new instance.

Spyromitros, Tsoumakas, and Vlahavas (2008) performed MLC with k NN via BR, computing the similarity matrix only once to reduce the time complexity. Pugelj and Džeroski (2011) generalized the k NN algorithm to several types of structured outputs (including MLC). Finally, M.-L. Zhang (2009) proposed an improved version of the ML- k NN algorithm, which, contrary to the original ML- k NN, does not assume the independence of labels and examples. This is achieved by using a coupled similarity measure which considers both the descriptive attributes and labels.

Neural networks Crammer and Singer (2003) proposed the multi-label perception algorithm where one perception per label is used and the predictions are calculated via the inner products. M.-L. Zhang and Zhou (2006) adapted the back propagation algorithm for MLC by proposing a novel error function which ranks the labels belonging to an example higher than the labels that do not belong to that example. Finally, M.-L. Zhang (2009) adapted the radial basis function neural network for MLC. Recently, deep neural networks for MLC were proposed (Yeh, Wu, Ko, & Wang, 2017).

SVMs Several researchers applied the BR paradigm to perform MLC with SVMs (Boutell et al., 2004; Gonçalves & Quaresma, 2004). Elisseeff and Weston (2001) proposed a ranking-based SVM method which improved over BR with SVMs. Recently, J. Xu (2014) proposed an improvement of the ranking-based SVM which drastically reduces computational time. Recently, W.-J. Chen, Shao, Li, and Deng (2016) implemented the multi-label version of the twin support vector machines algorithm (Jayadeva, Khemchandani, & Chandra, 2007), which does not require separating hyperplanes to be parallel. This method was shown to outperform the ranking-based SVMs for MLC.

Bayesian networks S. Wang, Wang, Wang, and Ji (2014) proposed to model the dependencies among labels with a Bayesian network. Nodes of the network represent labels, while links and conditional probabilities are used to model the probabilistic dependencies among labels. The inferred network structure of the labels can then be used to enhance the performance of any multi-label classifier.

Rule learning Ženko and Džeroski (2008) proposed predictive clustering rules for multi-target prediction. Recently, Loza Mencía and Janssen (2016) proposed two approaches (i.e., a stacking and a separate-and-conquer approaches) to learn label-dependent single-label rules. Such rules predict a single-label but can contain conditions on other labels in order to exploit possible label dependencies.

Decision trees (Blockeel et al., 1998) proposed the concept of PCTs where the variance and prototype function are considered as an input parameters. Among other types of structured outputs, PCTs can handle multi-target prediction. Clare and King (2001) modified the definition of entropy and proposed a decision tree algorithm to predict multi-label gene expression data. The MLC method of De Comit  , Gilleron, and Tommasi (2003) produces sets of rules that can be viewed as a tree. Q. Wu, Ye, Zhang, Chow, and Ho (2015) proposed a tree learning method for MLC where a SVMs are trained at each node in an one-versus-all fashion to determine the splits. This method showed a good predictive performance, however, it has a very high computational cost since many SVMs need to be trained. Furthermore, it produces a potentially very large tree, since at each split there can be as much branches as there are labels. A more efficient approach that combines decision trees and SVMs was proposed by Gjorgjevikj, Madjarov, and Džeroski (2013). There, a multi-label decision tree is built first, followed by training of local binary relevance SVMs

in leafs of the tree. Multi-label decision tree algorithms were proposed recently (Al-Otaibi, Kull, & Flach, 2014; Al-Otaibi, Kull, & Flach, 2016) which try to capture local label dependencies at each node of the tree. For a set of dependent labels, a single (global) tree is constructed, while for independent labels, one (local) tree per-label is build.

Ensemble approaches Schapire and Singer (2000) proposed two extensions of the popular boosting algorithm method (ADABOOST) to MLC (ADABOOST.MH and ADABOOST.MR). De Comit   et al. (2003) adapted the alternating decision tree algorithm (Freund & Mason, 1999) to MLC and used it as an input to ADABOOST.MH. Kocev et al. (2013) proposed bagging and random forests of PCTs that can handle various types of structured outputs, including MLC. Su and Rousu (2015) proposed an ensemble of max-margin graph labeling classifiers, where a graph is defined over a set of labels to model possible dependencies. The method performed competitively to ADABOOST.MH and bagging of SVMs. Recently, Q. Wu, Tan, Song, Chen, and Ng (2016) proposed a bagging of multi-label classification trees (ML-Forest) where label dependencies are automatically modeled by keeping track of a group of labels that are relevant for examples as they are divided during the tree construction. Q. Wu et al. (2016) performed an empirical comparison with their method and many other MLC approaches. Similarly as many MLC studies have shown before, the study of Q. Wu et al. (2016) also confirms that none of the methods perform the best according to all performance measures. According to their results, however, the following methods are consistently top ranked: BR, classifier chains (Read et al., 2011), HOMER (Tsoumakas et al., 2008), ML-Forest (Q. Wu et al., 2016), random forests of PCTs (Kocev et al., 2013) and the two-stage architecture (Madjarov, Gjorgjevikj, & D  zeroski, 2012). The ML-Forest performs competitively to other ensemble approaches for MLC (HOMER and random forests of PCTs), however, random forests of PCTs are superior in terms of computational efficiency.

Recently, Madjarov, Kocev, et al. (2012) performed an extensive empirical comparison of multi-label classification methods, where they compared several problem transformation and algorithm adaptation methods across a large number of datasets. Their results are consistent with the ones by Q. Wu et al. (2016): The best performing method typically changes depending on the performance metric, while the top ranked methods are: BR, classifier chains, HOMER and random forests of PCTs. Madjarov, Kocev, et al. (2012) conclude that, considering both the predictive performance and efficiency, the state-of-the-art methods are random forests of PCTs for MLC (Kocev et al., 2013) and the HOMER method (Tsoumakas et al., 2008).

3.2.1.2 Multi-target regression

Several local approaches to deal with MTR were proposed. The simplest is a binary relevance (i.e., a single-target) approach, where a MTR problem with T targets is transformed to T single-target regression problems. After transformation, any existing regression algorithm can be applied. Recently, several problem transformation methods were proposed that are inspired by approaches used in multi-label classification. Tsoumakas et al. (2014) proposed an ensemble method for MTR based on random linear combinations of the target variables, while Spyromitros-Xioufis, Groves, Tsoumakas, and Vlahavas (2014) proposed regressor stacking and regressor chaining. Contrary to binary relevance, these approaches are designed to model the dependencies between targets.

Regressor stacking and chaining The regressor stacking starts by building a separate model for each target (the same as in binary relevance), followed by the second step,

where an additional meta-model is learned for each target. Each meta-model is learned from the transformed training set, where the original training set is augmented with T descriptive attributes (one for each target) consisting of predictions obtained from the first-stage models. The idea is that, meta-models can correct the prediction of the first-stage models by using information about the predictions of first-stage models. The regressor chaining follows a principle similar to previously described classifier chains. Namely, a chain of single-target regression models is trained (one for each target variable), where a training set of the model is augmented with predictions of preceding models in the chain. The order of target variables in a chain is determined with a random permutation. Since regressor chaining is sensitive to ordering of targets in a chain, an ensemble of regressor chains was also proposed (Spyromitros-Xioufis et al., 2014), where the procedure is repeated several times with different random permutations on bootstrap samples of the training set and the final predictions are obtained by taking the mean. The empirical evaluation of Spyromitros-Xioufis et al. (2014) suggests that when a strong base regressor is used, such as bagging (Breiman, 1996a), it is hard to improve over the single-target approach (even for state-of-the-art global approaches). However, when a weaker regressor was used, such as ridge regression (Hoerl & Kennard, 1970), regressor stacking and especially ensembles of regressor chains improved over single-target approach. However, this comes with an additional computational cost, and may hinder solving MTR tasks with many targets.

Many machine learning methods, popular for regression, have been adapted also for the task of MTR according to the algorithm adaptation principle, i.e., to simultaneously predict all the targets with a single global model. These methods can be grouped as follows (Borchani et al., 2015): statistical methods, multi-target support vector regression, multi-target regression trees, and rule methods. We briefly overview these groups, while for more details we refer the reader to a recent overview paper by Borchani et al. (2015) and references therein.

Statistical methods In statistics, an early method for MTR was proposed by Brown and Zidek (1980) who adapted the standard ridge regression (Hoerl & Kennard, 1970) to multi-variate ridge regression. Later, Breiman and Friedman (1997) proposed the Curds&Whey method, where correlations between the target variables are modeled in a post-processing phase. Recently, Abraham et al. (2013) proposed a method based on coupling regression methods with geometric quantile mapping that preserves the relationships among the target variables (including non-linear relationship) while minimizing the residual errors. Recently, Gillberg et al. (2016) proposed a Bayesian reduced rank regression method that deals with structured noise in MTR.

SVMs Many approaches have been proposed that extend the support vector regression (SVR) to MTR (Vazquez & Walter, 2003; Sánchez-Fernández, de-Prado-Cumplido, Arenas-García, & Pérez-Cruz, 2004; Cai & Cherkassky, 2009; W. Zhang, Liu, Ding, & Shi, 2012; Brouwer, Kubicki, Sofo, & Giles, 2014; S. Xu, An, Qiao, Zhu, & Li, 2013; Chung, Kim, Lee, & Kim, 2015; Yanyan Yang, Chen, & Dong, 2015). Researchers considered different strategies in order to take into account possible dependencies between target variables and improve accuracy of multi-target support vector regression (MSVR). For example, Vazquez and Walter (2003) proposed the multi-target version of the Kriging method to exploits the correlations between targets with MSVR, and showed that their solution is more accurate than predicting targets independently. W. Zhang et al. (2012) cast a multi-target problem into a single-target one in the extended descriptive space and then apply least squares SVR. S. Xu et al. (2013), on the other hand, developed efficient multi-output least squares SVR.

Recently, Yanyan Yang et al. (2015), proposed MSVR where an additional regularization term is introduced to account for correlations between the target variables. This method showed better predictive performance and resilience to noise than the method proposed by S. Xu et al. (2013).

Multi-target regression trees One of the first multi-target regression tree methods was proposed by De’Ath (2002), who modified the impurity measure of the standard CART algorithm as the sum of squared error over the multiple target variables. Struyf and Džeroski (2006) proposed constraint-based multi-objective regression trees, where the user can specify the size and accuracy constraints in order to trade-off between interpretability and accuracy of the trees. Appice and Džeroski (2007) proposed a method for stepwise induction of multi-target model trees, where at each leaf node linear regression models are used to predict the target variables. Kocev et al. (2013) implemented ensembles of multi-target regression trees by using random forests (Breiman, 2001) and bagging techniques (Breiman, 1996a). Recently, D’Ambrosio, Aria, Iorio, and Siciliano (2017) generalized regression trees to multivalued numerical response variables, where the output space can contain intervals, multiple numeric variables, or modal variables, where a frequency, probability or weight is assigned to each value.

Rules Very few rule-based methods for MTR are available. Aho, Ženko, and Džeroski (2009) introduced rule ensembles for MTR. Their algorithm, named FIRE, decomposes a random forest for MTR into a large collection of rules and then an optimization procedure selects the best rule to obtain a concise set of rules comparable in accuracy with a single multi-target regression tree. Aho et al. (2012) improved the FIRE algorithm by combining rules with simple linear functions.

K-nearest neighbors The popular k-nearest neighbors algorithm was also extended to MTR setting by Pugelj and Džeroski (2011).

Out of the available methods for MTR, only trees produce interpretable models (and to some degree rule ensembles for MTR (Aho et al., 2009; Aho et al., 2012)), which is an important property in many domains, such as medicine and chemoinformatics. In this thesis, we build upon the work of Struyf and Džeroski (2006) and Kocev et al. (2013). We extend their work to the semi-supervised learning task, with the goal to exploit appealing characteristics of their methods: interpretability of trees (Struyf & Džeroski, 2006) and state-of-the-art predictive performance of ensembles (Kocev et al., 2013).

3.2.2 Hierarchical multi-label classification

C. Silla and Freitas (2011) presented an extensive survey of methods for HMLC. They categorize HMLC methods along several criteria, depending on *(i)* whether the method can predict single or multiple paths of the hierarchy; *(ii)* whether the method can predict only leaf nodes or any node of the hierarchy; *(iii)* which type of hierarchy the method can handle (i.e., tree, DAG, or both) and *(iv)* whether the method is local or global.

In the task of HMLC, there are four different approaches that can be used to build local models (C. Silla & Freitas, 2011): flat classification, local classifiers per level, local classifiers per parent node, and local classifiers per node (C. Silla & Freitas, 2011). In flat classification, a separate local classifier is learned for each node of the class hierarchy, where examples labeled with a label corresponding to that node are considered as positive and all the others as negative. In other local approaches (local classifiers per level, per parent node, and per node), hierarchical relationships among classes are taken into account by virtue

of training separate local classifiers only with the subset of examples which are labeled with a specific part of the class hierarchy. More specifically, the local classifiers per level approach consists of training one multi-class classifier for each level of the class hierarchy to differentiate between nodes at each level of class hierarchy. Next, the local classifiers per parent node approach (also known as hierarchical single-label classification (HSC)) builds a multi-class classifier for each parent node in the class hierarchy to distinguish between its child nodes. Finally, the local classifiers per node approach consists of training one binary classifier for each node of the class hierarchy.

C. Silla and Freitas (2011) analyzed the results of many studies and concluded that, when solving a hierarchical (multi-label) classification problem, any hierarchical classification approach (local or global) is usually better than the flat classification approach. Vens et al. (2008) investigated the performance of predictive clustering trees for HMLC over a large collection of datasets from functional genomics. They compared global trees and two local approaches (local classifiers per parent node and local classifiers per node) and found that global trees outperform both local approaches, while the local classifiers per parent node outperform local classifiers per node for DAGs; for tree-shaped hierarchies the performances of local approaches were similar. Levatić, Kocev, and Džeroski (2014) investigated the performance of global hierarchical/flat methods (i.e., HMLC and MLC) and local hierarchical/flat methods (i.e., binary relevance and HSC) over datasets from various domains. The results showed that the global hierarchical approaches perform the best on datasets with a well-populated class hierarchy (i.e., where the average number of labels per example is greater than 2), while local hierarchical approach (i.e., HSC) performed the best when the number of labels per example is closer to one. However, considering computational efficiency and interpretability (i.e., the model size), the global HMLC models are the best choice. Recently, Madjarov, Gjorgjevikj, Dimitrovski, and Džeroski (2016) showed that MLC tasks can be enhanced with an artificial (i.e., a data-derived) label hierarchy.

Given the conclusions of many studies listed above, considering hierarchical relationships among labels is undoubtedly beneficial for classification performance. We next present an overview of methods for HMLC according to the learning technique that is used to build a predictive model.

Kernel-based methods Several kernel-based methods for the task of HMLC were proposed. In these methods, SVMs are learned for each class separately and then combined so that the predictions are consistent with the hierarchical relationships (Obozinski, Lanckriet, Grant, Jordan, & Noble, 2008; Barutcuoglu, Schapire, & Troyanskaya, 2006; Guan et al., 2008; Valentini, 2011). Rousu, Saunders, Szedmak, and Shawe-Taylor (2006) presented a more direct method that does not require a second step to make sure that the hierarchy constraint is satisfied. Their method is based on a large margin method for structured output prediction which defines a joint feature map over the input and the output space. The common feature of the kernel-based methods for HMLC is that they produce uninterpretable predictive models, and are often computationally expensive.

Methods based on trees Decision tree-based methods take a notable place among approaches for HMLC. Contrary to the previously described methods, decision trees are easily interpreted by domain experts. Clare and King (2003) adapted the well-known decision tree algorithm C4.5 (Quinlan, 1993) for the task of HMLC. Their version of C4.5 (called C4.5H) uses the sum of entropies of the class variables to select the best split. C4.5H predicts classes on several levels of the hierarchy, assigning a larger cost to misclassification higher up in the hierarchy. Blockeel, Bruynooghe, Džeroski, Ramon, and Struyf (2002), Blockeel, Schietgat, Struyf, Džeroski, and Clare (2006) proposed the idea

of using PCTs (Blockeel, 1998) for HMLC tasks. The work Blockeel et al. (2006) presents the first thorough empirical comparison between an HMLC and HSC decision tree method in the context of tree-shaped class hierarchies. Vens et al. (2008) extend the algorithm towards hierarchies structured as directed acyclic graph. Cerri, Pappa, Carvalho, and Freitas (2013) presented an extensive evaluation of methods for HMLC based on decision tree over a large number of datasets and performance measures. The choice of the best method is not conclusive since different methods perform the best depending on the performance metric, however global methods are preferred since they produce simpler models. Furthermore, Cerri et al. (2013) recommend the use of precision-recall curves (Davis & Goadrich, 2006) and hierarchical precision and recall (Kiritchenko, Famili, Matwin, & Nock, 2006) performance measures. Kocev et al. (2013) extended PCTs for HMLC to an ensemble setting and showed that ensembles significantly outperform single PCTs.

Methods based on networks The protein-protein interaction networks and functional association networks are frequently employed by HMLC methods to leverage protein function prediction (Y. Chen & Xu, 2004; Mostafavi, Debajyoti, Warde-Farley, Grouios, & Morris, 2008; Tian et al., 2008). Stojanova, Ceci, Malerba, and Dzeroski (2013) extended the PCT for HMLC to a network setting and defined a notion of autocorrelation for such a setting. The results of their study show that the proposed method, which exploits gene/protein properties and network information, achieves better predictive performance than the methods that use these two sources separately.

Other approaches There are several other methods for HMLC based on different approaches. Kiritchenko et al. (2006) performed hierarchical text categorization by expanding label sets of training examples to make them consistent with a given class hierarchy. Standard multi-class learning algorithm is then applied to the modified multi-label data, followed by re-labeling of the inconsistently classified test instances. C. N. Silla and Freitas (2009) adapted the naïve Bayes approach for HMLC. Otero, Freitas, and Johnson (2010) and Cerri, Barros, and de Carvalho (2012) used search heuristics to construct HMLC rules. In their other work, Cerri, Barros, and de Carvalho (2014) represented class hierarchy as a sequence of connected artificial neural networks, where the output of the one network is used as the input of the next network in the sequence. Alaydie, Reddy, and Fotouhi (2012) proposed a boosting-based method for HMLC, where at each iteration the label hierarchy is used to select the training set for each classifier. Barros, Cerri, Freitas, and de Carvalho (2013) proposed a method for HMLC based on the probabilistic clustering with the expectation-maximization algorithm. Bi and Kwok (2015) proposed a hierarchically aware loss function, appropriate for both tree and DAG hierarchies and developed a Bayes-optimal classifier for HMLC by using this loss function. The proposed method was shown to perform superior to local classifier per node with SVMs and decision trees for HMLC (Vens et al., 2008). Recently, Sun, Zhao, Cao, and Hao (2016) proposed to transform a HMLC task into the optimal path prediction problem and used network flow solvers to train a model in polynomial time complexity. They showed that their method for HMLC performs favorably to decision trees (Vens et al., 2008), neural networks (Cerri et al., 2014), and probabilistic clustering (Barros et al., 2013).

Chapter 4

Semi-Supervised Learning for Structured Output Prediction

In previous two chapters, we gave an overview and formal definitions of semi-supervised learning (for unstructured tasks) and structured output prediction. Now we are ready to define the topic of this thesis, which combines these two machine learning fields. In the first part of this chapter, we formally define semi-supervised learning for structured output prediction, while in the second part, we give an overview of related work on this topic.

4.1 The Task of Semi-Supervised Learning for Structured Output Prediction

In Chapter 3, we have defined the (supervised) machine learning tasks of multi-target prediction and hierarchical multi-label classification. In these tasks, each example $e \in E$ is a tuple of one element from the descriptive space X and one from the target space Y . In the semi-supervised learning task, the latter can be unknown: Some examples consist only of an element from the descriptive space. We define the task of semi-supervised learning for structured output prediction as follows.

Given:

- A descriptive (or input) space $X = X_1 \times X_2 \times \dots \times X_D$ spanned by D descriptive variables that consist of values of primitive data types (boolean, discrete or continuous).
- A target (or output) space S , which is defined specifically for each type of structured output (e.g., multiple target prediction or hierarchical multi-label classification).
- A set of *labeled examples* $E_l = \{(x_i, y_i) \mid x_i \in X, y_i \in S, 1 \leq i \leq N_l\}$, where each example is a pair consisting of one element from the descriptive space and one element from the target space (i.e., the example's label), and N_l is the number of labeled examples.
- A set of *unlabeled examples* $E_u = \{x_i \mid x_i \in X, 1 \leq i \leq N_u\}$, which consists only of elements from the descriptive space, and N_u is the number of unlabeled examples.
- A quality criterion q , which rewards models with a high predictive accuracy (or low predictive error).

Find: a function $f : X \rightarrow S$, by using both E_l and E_u , such that f maximizes q .

Similarly as in SSL for unstructured outputs (as defined in Chapter 2), in SSL for SOP the same quality criterion q is optimized as in the previously defined supervised SOP tasks (Chapter 3) but the training sets are different: SSL uses unlabeled examples E_u , in addition to the labeled ones E_l . Again, the goal of semi-supervised methods for SOP is not to achieve a high predictive performance in absolute terms, but rather to achieve a higher predictive performance in comparison to the corresponding supervised methods for SOP.

In machine learning tasks with unstructured output (i.e., regression and classification), the data can be either labeled or unlabeled. In tasks with structured outputs, however, it is possible that for a given example only a part of the structured output is known. We refer to such data as *partially labeled examples*. In most of the experiments in this thesis we consider labeled and (completely) unlabeled examples, since such an experimental scenario is almost exclusively used in semi-supervised literature. However, the semi-supervised method proposed in Chapter 7 can handle also partially labeled data. In Chapter 9 we thus present a case study, where we show the advantages of this method in comparison to the methods that cannot handle partially labeled data.

Figure 4.1 presents an illustrative example of the dataset for the task of semi-supervised learning for multi-target regression. The dataset contains labeled, unlabeled and partially labeled examples.

4.2 Related Work

The majority of work in the field of semi-supervised learning is dealing with prediction tasks with primitive outputs (i.e., classification and regression) (Chapelle et al., 2006; Zhu, 2008), while semi-supervised learning for SOP received less attention. Even within the field of SSL for SOP, there is quite a discrepancy between the amount of work devoted to different types of structured outputs. Namely, semi-supervised learning for MTR and HMLC received hardly any attention, with only a few publications on these topics available. On the other hand, many semi-supervised methods for MLC were proposed.

Note that semi-supervised methods were also proposed for types of structured outputs other than the ones we are concerned with in this thesis (i.e., MTR, MLC and HMLC). Such as sequence labeling (Suzuki, Fujino, & Isozaki, 2007; Brefeld, Büscher, & Scheffer, 2005; P. S. Dhillon, Sellamanickam, & Selvaraj, 2011; Chang, Ratnikov, & Roth, 2012; P. Dhillon, Keerthi, Bellare, Chapelle, & Sellamanickam, 2012), parse tree prediction (Brefeld & Scheffer, 2006), or time-series classification (Kim, 2013).

In this section, we first give an overview of semi-supervised methods that are designed to handle only specific types of structured outputs, i.e., either MTR, MLC or HMC. We then give an overview of semi-supervised methods for structured output prediction, i.e., semi-supervised methods designed to handle several different types of structured outputs. Finally, we discuss several issues we have identified within the currently available semi-supervised methods for SOP.

4.2.1 Semi-supervised methods for specific types of structured outputs

To begin with, Ceci (2008) proposed a spectral graph transducer method for hierarchical text categorization in the transductive setting and showed advantage over the supervised version of the algorithm. Santos and Canuto (2014) developed a semi-supervised method for HMLC based on the self-training (similarly as the method we propose in Chapter 6).

Similarly to the HMLC task, the selection of semi-supervised methods for the task of

Site ID	Descriptive variables					Target variables						
	LandCover	TempRange	NativeTreeProb	GrassProb1Ha_RegionStdDev	::	Large tree score	Tree canopy score	Understorey score	Litter score	Logs score	Weeds score	Recruitment score
ID1	2.0	225.0	0.0	1.79	...	7	3	15	3	5	15	5
ID2	4.0	278.0	2.0	12.31	...	6	3	10	5	5	13	3
ID3	8.0	191.0	54.0	6.52	...	0	5	10	5	0	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...
IDN _l	1.0	233.0	0.0	0.68	...	0	4	15	5	5	13	5
IDN _{l+1}	2.0	200.0	13.0	5.29	...	?	?	?	?	?	?	?
IDN _{l+2}	4.0	201.0	0.0	0.68	...	?	?	?	?	?	?	?
IDN _{l+3}	2.0	216.0	8.0	1.84	...	?	?	?	?	?	?	?
IDN _{l+4}	3.0	222.0	0.0	0.36	...	?	?	?	?	?	?	?
IDN _{l+5}	2.0	198.0	0.0	9.68	...	?	?	?	?	?	?	?
...	...	...	...	...	...	...	...	...	...	...	...	...
IDN _u	3.0	189.0	2.0	3.85	...	?	?	?	?	?	?	?
IDN _{u+1}	9.0	207.0	24.0	7.45	...	?	6	9	20	14	0	13
IDN _{u+2}	9.0	193.0	9.0	11.86	...	?	9	?	8	?	12	?
IDN _{u+3}	6.0	156.0	35.0	1.83	...	?	13	6	3	?	?	?
IDN _{u+4}	9.0	165.0	31.0	13.44	...	1	?	?	?	10	19	?
...	...	...	...	...	...	...	...	...	...	...	...	...

Labeled examples

Unlabeled examples

Partially labeled examples

Figure 4.1: An example of a dataset with multiple continuous targets used for modeling the condition of indigenous vegetation (Kocev, Džeroski, White, Newell, & Griffioen, 2009). The descriptive variables are obtained from a geographical information system, while the target variables are indices describing the condition of the vegetation. The dataset contains labeled, unlabeled and partially labeled examples.

MTR (or related tasks) is very limited. Y. Zhang and Yeung (2009) proposed a semi-supervised method based on Gaussian processes for a task related to MTR: *multi-task* regression. Multi-target regression and multi-task regression¹ are similar machine learning tasks, however some notable differences exist. In multi-task learning, in general, the aim is to predict multiple single-target variables with different training sets (with possibly different descriptive attributes) and the emphasis is on learning transfer between the tasks. Navaratnam et al. (2007) have proposed a SSL method for MTR, which is also based on Gaussian processes, but is specialized for applications in computer vision. Recently, another semi-supervised method for multi-task regression based on Gaussian processes was proposed by Cardona, Álvarez, and Orozco (2015), where Expectation Maximization

¹Note that multi-task regression is sometimes referred to as a multi-output regression which is also a synonym for multi-target regression, thus confusions may arise.

algorithm that uses labeled and unlabeled data is used to estimate the parameters of joint Gaussian processes. Finally, we have developed a semi-supervised method for MTR based on self-training of random forests of PCTs (Levatić et al., 2015a, 2017a), where the most reliable predictions on unlabeled data are iteratively used to re-train the model.

As previously mentioned, as opposed to the tasks of MTR and HMLC, a number of semi-supervised methods for the task of MLC were proposed. Many of which are graph-based (G. Chen, Song, Wang, & Zhang, 2008; Zha, Mei, Wang, Wang, & Hua, 2009; J. Wang, Zhao, Wu, & Hua, 2011; Kong, Ng, & Zhou, 2013; Y. Wang, Pei, Lin, Zhang, & Zhang, 2014; B. Wang & Tsotsos, 2016). Consequentially, these methods are transductive (see Section 2.2), meaning that they provide predictions only for unlabeled data that is used during the learning process, and cannot generalize to unseen examples. Furthermore, graph-based semi-supervised methods are often very computationally expensive (Liu et al., 2012). Yuhong Guo and Schuurmans (2012) proposed a transductive method which learns a new data representation of the labeled and unlabeled data, in which the label correlations are more apparent, while simultaneously training a supervised large margin classifier (on a new representation). L. Wu and Zhang (2013) pointed out the importance of inductive models, i.e., the ones that can generalize to unseen data, and proposed such a semi-supervised method for MLC. Their approach is based on iterative re-learning of a SVM-based classifier, which maximizes a margin by considering both the known labels and the predicted labels of unlabeled examples. Other recently proposed semi-supervised methods for MLC include semi-supervised multi-label boosting (Zhao & Zhai, 2015), semi-supervised k NN (d Lucena & Prudencio, 2015), co-training (M. Xu, Sun, & Jiang, 2014) and binary relevance (Švec, 2014).

4.2.2 Semi-supervised methods for SOP

Several authors proposed methods based on the maximum-margin approach, i.e., SVM-like methods. Altun, McAllester, and Belkin (2006) developed a semi-supervised maximum-margin approach for SOP similar to SVMs, while Y. Li and Zemel (2014) developed a maximum-margin approach based on posterior regularization. Brefeld and Scheffer (2006) used the co-training paradigm and the principle of maximizing the consensus among multiple independent hypotheses. This principle was then developed into a semi-supervised support vector learning algorithm for joint input-output spaces and arbitrary loss functions. Next, Zien, Brefeld, and Scheffer (2007) proposed transductive SVMs for SOP.

Conditional Random Fields (Lafferty et al., 2001) are widely used for discrete structured outputs, mainly sequence labeling and similar tasks. Several studies extended this approach to a semi-supervised setting (Y. Wang, Haffari, Wang, & Mori, 2009; Subramanya, Petrov, & Pereira, 2010; P. S. Dhillon et al., 2011).

Recently, Jiang, Jia, Sheng, and LeMieux (2016) and Du (2017) proposed semi-supervised methods for discrete structured outputs (namely, MLC, HMLC and sequence learning) where they utilized the k NN method to account for the structure in the input space. Jiang et al. (2016) constructed a k NN graph which represents a manifold of the input space, which is then used to regularize the learning of the output space. Du (2017) pointed out that a single predictor may not be sufficient to model several possible local distributions in the datasets - which is more likely to occur in a presence of a large amount of unlabeled data. To resolve this issue, Du (2017) proposed to learn local models for different parts of the datasets based on nearest neighborhoods of examples.

Sellamanickam, Tiwari, and Selvaraj (2012) developed probabilistic models based on entropy and label distribution regularization and applied it to hierarchical and multi-label classification, while Suzuki et al. (2007) proposed a hybrid discriminative-generative method, where the discriminative method is enhanced with labels on unlabeled data pro-

vided by the generative method. Next, Hu and Kwok (2010) proposed a graph based manifold regularization method and applied it to hierarchical classification.

The methods proposed by Gönen and Kaski (2014) and Brouard et al. (2016) are rare examples of semi-supervised methods which can handle both discrete and continuous types of structured outputs, though, the method of Gönen and Kaski (2014) was tested (in its semi-supervised version) only on datasets with discrete outputs, namely MLC. Both methods are based on kernels. Gönen and Kaski (2014) proposed kernelized Bayesian matrix factorization, while Brouard et al. (2016) proposed Input Output Kernel Regression (IOKR) which enables to exploit the structure both in the input and the output space. In this thesis, we consider only the structure of the output space.

4.2.3 Issues within the current semi-supervised methods for SOP

There is a number of issues we identified within the available semi-supervised methods for SOP. First of all, none of the above-mentioned semi-supervised methods for SOP are interpretable. Interpretable models are, however, of great importance in many applications of machine learning. They can help experts to extract knowledge from the data, explain the data, or even form new (testable) hypotheses. Furthermore, we postulate that experts which are not familiar with machine learning are more willing to trust and use machine learning methods if they can easily understand the model and "see" how it made the predictions. In Chapter 7, we propose semi-supervised predictive clustering trees for SOP. The proposed algorithm preserves the interpretability of supervised PCTs, while enabling the exploitation of unlabeled data. To the best of our knowledge, no other semi-supervised method for SOP which produces interpretable models is available.

Next, even though some of the above-mentioned semi-supervised methods for SOP are designed to handle several types of structured outputs, they are not extensively evaluated. That is, they are usually evaluated only on some specific case of structured output, or on a very limited number of datasets with different structured outputs. Most commonly datasets from the domains of text and image classification are considered. Furthermore, most of the methods deal only with discrete types of structured outputs, while very few methods can handle both the discrete and continuous types of structured outputs. More specifically, the popular semi-supervised approaches for SOP - conditional random fields and maximum-margin methods - assume discrete outputs and are computationally expensive. Though, Subramanya et al. (2010) developed very efficient semi-supervised conditional random fields.

Contrary to common practices in the field of SSL for SOP, we evaluate the methods proposed in Chapter 6 and 7 on a large number of datasets from various domains with discrete (i.e., MLC and HMLC tasks) and continuous (i.e., MTR task) structured outputs.

The semi-supervised PCTs (Chapter 7) are inspired by the work of Blockeel et al. (1998), who proposed clustering trees and suggested that they may be useful when class information is missing. They construct the decision trees in an unsupervised manner, where distances among examples take into account all descriptive attributes. Class labels (i.e., target variables) are not used during the tree construction phase, but only to assign labels at leaf nodes once the tree has been constructed. This concept can be considered as a semi-supervised cluster-then-label approach (Dara, Kremer, & Stacey, 2002; Demiriz, Bennett, & Embrechts, 1999; Andrew B Goldberg, Zhu, Singh, Xu, & Nowak, 2009).

The semi-supervised decision trees that we propose in this work are similar in nature to the ones proposed by Blockeel et al. (1998), but there are several key differences. To construct the trees, we consider simultaneously the descriptive and the target attributes. In other words, clustering and predictive modeling are performed simultaneously, which enables us to fully exploit both the labeled and unlabeled examples. Next, we introduce an

additional parameter, by which we control the amount of supervision in the decision trees. This allows us to build fully supervised trees, semi-supervised trees, or fully unsupervised trees, depending on the specific needs of the problem at hand. Similarly to this concept, Ženko (2007) considered predictive clustering rules where the heuristic that guides the rule learning process takes into account both the descriptive and the target attributes, however Ženko (2007) constructed such rules only in a supervised learning context.

Chapter 5

Predictive Clustering

In this chapter, we present tree-based methods for predicting structured outputs, which are the basis of the semi-supervised methods proposed in this thesis. We first present the predictive clustering trees (PCTs). In particular, we present the instantiations of the PCTs for predicting multiple continuous variables, predicting multiple discrete variables, and hierarchical multi-label classification. We then present ensembles of PCTs for predicting structured outputs.

5.1 Predictive Clustering Trees for Structured Outputs

The PCT framework views a decision tree as a hierarchy of clusters, where the top-node corresponds to one cluster containing all data. This cluster is recursively partitioned into smaller clusters while moving down the tree. The PCT framework is implemented in the CLUS system (Kocev et al., 2013), available at <http://sourceforge.net/projects/clus>.

PCTs are generalization of standard decision trees (for example, see C4.5 algorithm (Quinlan, 1993)) toward predicting structured outputs. PCTs can predict several types of structured output, including tuples of discrete/real values and hierarchies of classes. For each type, two functions need to be defined: prototype function and variance function. The prototype function returns a representative structured value (i.e., a prototype) given a set of such values. It is used to compute a label for each leaf of the tree. The variance function describes how homogeneous a set of structured values is, and it is used to find the best splits in nodes of the tree.

PCTs are induced with a standard *top-down induction of decision trees* (TDIDT) algorithm (Breiman et al., 1984). The PCT algorithm, presented in Algorithm 5.1, takes as input a set of examples E and outputs a tree. The heuristic (h) that is used for selecting the tests (t) is the reduction of variance caused by partitioning ($\mathcal{P}$) the instances (see the BestTest procedure in Algorithm 5.1). By maximizing the variance reduction the cluster homogeneity is maximized. If no acceptable test can be found, that is, if none of the candidate tests reduce the variance significantly or the number of examples in a node is below the user-specified threshold, then the algorithm creates a leaf and computes the prototype of the instances belonging to that leaf.

The main difference between PCTs and standard decision trees is that the former consider the variance function and the prototype function as parameters that can be instantiated for a given learning task (i.e., a given type of the structured output). So far, the PCTs have been instantiated for the following tasks: multiple targets prediction (Struyf & Džeroski, 2006; Kocev, Vens, Struyf, & Džeroski, 2007; Levatić et al., 2015a), hierarchical multi-label classification (Vens et al., 2008; Levatić, Kocev, & Džeroski, 2014; Levatić, Kocev, Debeljak, & Džeroski, 2015) and prediction of time-series (Slavkov et al., 2010).

Algorithm 5.1: The top-down induction algorithm for PCTs.

Procedure PCT**Input:** A dataset E **Output:** A predictive clustering tree $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$ **if** $t^* \neq \text{none}$ **then** **foreach** $E_i \in \mathcal{P}^*$ **do** $\text{tree}_i = \text{PCT}(E_i)$ **end** **return** $\text{node}(t^*, \bigcup_i \{\text{tree}_i\})$ **else** **return** $\text{leaf}(\text{Prototype}(E))$ **end****Procedure** BestTest**Input:** A dataset E **Output:** the best test (t^*) , its heuristic score (h^*) and the partition $(\mathcal{P}^*)$ it induces on the dataset (E) $(t^*, h^*, \mathcal{P}^*) = (\text{none}, 0, \emptyset)$ **foreach** possible test t **do** $\mathcal{P} =$ partition induced by t on E $h = \text{Var}_f(E, Y) - \sum_{E_i \in \mathcal{P}} \frac{|E_i|}{|E|} \text{Var}_f(E_i, Y)$ **if** $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$ **then** $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$ **end****end****return** $(t^*, h^*, \mathcal{P}^*)$

In this thesis, we propose semi-supervised PCTs, and ensembles thereof, for the tasks of multi-target prediction and hierarchical multi-label classification. In the remainder of this chapter, we will present *supervised* PCTs and ensembles of PCTs for these tasks in more detail. In Chapters 6 and 7, we will present the extension of these methods to the semi-supervised setting.

5.1.1 PCTs for multiple target variables

PCTs that are able to predict multiple targets simultaneously are called multi-target predictive clustering trees (MTDTs). The MTDTs that predict multiple continuous variables are called multi-target regression trees (MTRTs), while the MTDTs that predict multiple discrete variables are called multi-target classification trees (MTCTs).

5.1.1.1 PCTs for multiple continuous variables

The variance function for MTRTs is calculated as the average of the variances of the target variables. For each set of examples E with target space Y consisting of T continuous variables $Y_1, Y_2, \dots, Y_T$, the variance function is defined as follows:

$$\text{Var}_f(E, Y) = \frac{1}{T} \cdot \sum_{i=1}^T \text{Var}(E, Y_i), \quad (5.1)$$

where $\text{Var}(E, Y_i)$ is the variance of the i^{th} target variable Y_i for a set of examples E . The variance of the i^{th} target variable is calculated as follows:

$$\text{Var}(E, Y_i) = \frac{\sum_{j=1}^N (y_{j,i})^2 - \frac{1}{N} \cdot \left(\sum_{j=1}^N y_{j,i} \right)^2}{N}, \quad (5.2)$$

where, $y_{j,i}$ is the value of the i^{th} target variable for the j^{th} example, and $N = |E|$ is the number of examples. The variances of the targets are normalized, so that each target contributes equally to the overall variance. The normalization is performed by dividing the above estimates (5.2) with the variance of the target variable on the entire available training set.

In the prediction phase, for each new example, the algorithm identifies the leaf it belongs to and returns the value predicted by a prototype function associated to that leaf (see Algorithm 5.1, procedure PCT). In MTRTs the prototype function calculates the mean vector of all target variables for the training examples that belong to the leaf.

Figure 5.1 presents an example of PCT for predicting multiple continuous variables. The internal nodes of the tree contain tests on the descriptive variables and the leaves store the predictions.

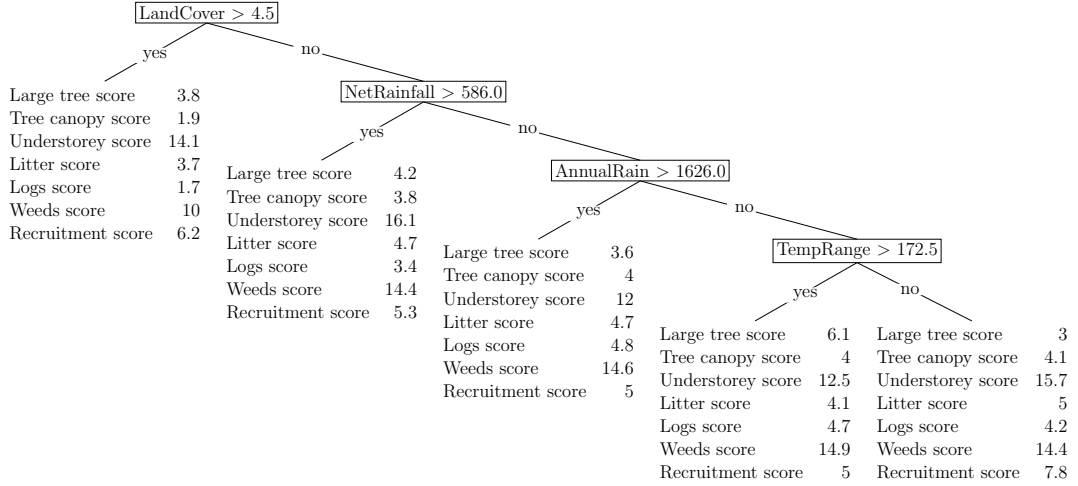


Figure 5.1: Example of a predictive clustering tree for predicting multiple continuous variables, which predicts several indicators of the condition of indigenous vegetation in Victoria, Australia (Kocev, Džeroski, White, Newell, & Griffioen, 2009). The tree was learned by using the data shown in Figure 3.1.

5.1.1.2 PCTs for multiple discrete variables

The variance function for the MTCTs is computed as the average of the gini indices of the target variables. For each set of examples E with target space Y consisting of T discrete target variables $Y_1, Y_2, \dots, Y_T$, the variance function is defined as follows:

$$Var_f(E, Y) = \frac{1}{T} \cdot \sum_{i=1}^T Gini(E, Y_i), \quad (5.3)$$

where $Gini(E, Y_i)$ is the gini score of the i^{th} target variable Y_i for a set of examples E . The gini score of the i^{th} target variable is calculated as follows:

$$Gini(E, Y_i) = 1 - \sum_{j=1}^{C_i} p_j^2, \quad (5.4)$$

where C_i is the number of classes of the i^{th} target variable (e.g., if Y_i is binary, then $C_i = 2$), and p_j is the empirical probability of a class c_j (i.e., the proportion of examples in a set E belonging to the class c_j).

Alternatively, one can also use the sum of the entropies of class variables as a variance function, i.e., $Var_f(E, Y) = \sum_{i=1}^T Entropy(E, Y_i)$ (this definition has also been used in the context of multi-label prediction (Clare, 2003)). The CLUS system also implements other variance functions, such as reduced error, gain ratio and the m -estimate.

The prototype function returns a vector of probabilities that an instance belongs to a given class for each target variable. To determine the predicted classes, the user can specify a threshold on probabilities, or the most probable (majority) class value for each target can be calculated. In this thesis, we use majority class.

Figure 5.2 presents an example of PCT for predicting multiple continuous variables.

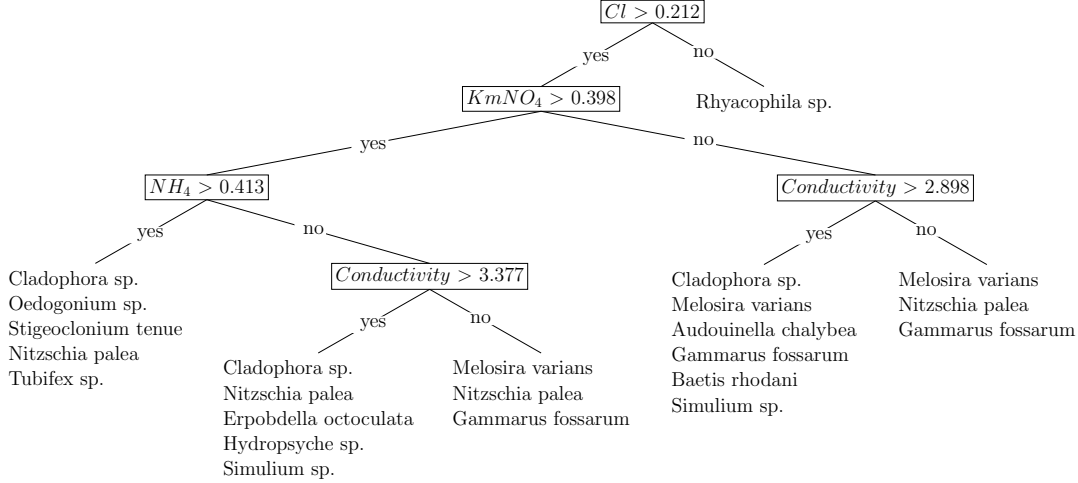


Figure 5.2: Example of a predictive clustering tree for predicting multiple discrete variables, which predicts the presence of several bioindicator organisms in water samples on the basis of chemical parameters (Sašo Džeroski, Demšar, & Grbović, 2000). The tree was learned by using the data shown in Figure 3.2.

5.1.2 PCTs for hierarchical multi-label classification

Recall that, in HMLC, the target space Y is defined as a hierarchy of classes $(C, \leq_h)$, where $C = \{c_i : c_1, c_2 \in C, c_1 \leq_h c_2 \iff c_1 \text{ is a superclass of } c_2\}$. The set of labels of example e_i is represented as a vector L_i with binary components, where the i^{th} component of the vector is 1 if the example belongs to class c_i and 0 otherwise. The arithmetic mean of a set of such vectors contains as i^{th} component the proportion of examples of the set belonging to class c_i . Now, the variance function over a set of examples E is calculated as the average squared distance between each example's class vector (L_i) and the set's mean class vector ($\bar{L}$):

$$Var_f(E, Y) = \frac{1}{|E|} \cdot \sum_{e_i \in E} d(L_i, \bar{L})^2. \quad (5.5)$$

In the HMLC context, it makes sense to consider the similarities at higher levels of the hierarchy as more important than the similarities at lower levels. This is reflected in the distance measure used in the above formula, which is a weighted Euclidean distance:

$$d(L_1, L_2) = \sqrt{\sum_{l=1}^{|C|} w(c_l) \cdot (L_{1,l} - L_{2,l})^2}, \quad (5.6)$$

where $L_{i,l}$ is the l^{th} component of the class vector L_i of an instance e_i , $|C|$ is the number of classes in the hierarchy (i.e., the size of the class vector), and the class weights $w(c)$ decrease with the depth of the class in the hierarchy. More precisely, $w(c) = w_0^{depth(c)}$, where $depth(c)$ denotes the depth of the class c in the hierarchy, and $0 < w_0 < 1$. Note

that, class weights can be calculated recursively, i.e., $w(c) = w_0 \cdot w(\text{par}(c))$, where $\text{par}(c)$ denotes the parent of class c . In this thesis, we use $w_0 = 0.75$, as recommended by Vens et al. (2008).

For example, consider the toy class hierarchy shown in Figure 5.3a, and two data examples: (X_1, Y_1) and (X_2, Y_2) that belong to the classes $Y_1 = \{c_1, c_2, c_{2.2}\}$ (boldface in Figure 5.3b) and $Y_2 = \{c_2\}$, respectively. We represent classes Y_1 and Y_2 with a vector representations L_1 and L_2 , respectively, where consecutive components represent membership in the classes $c_1, c_2, c_{2.1}, c_{2.2}$ and c_3 , in that order (preorder traversal of the tree of class labels). The distance is then calculated as follows:

$$\begin{aligned} d(L_1, L_2) &= d([1, 1, 0, 1, 0], [0, 1, 0, 0, 0]) \\ &= \sqrt{w_0^1 \cdot (1 - 0)^2 + w_0^1 \cdot (1 - 1)^2 + w_0^2 \cdot (0 - 0)^2 + w_0^2 \cdot (1 - 0)^2 + w_0^1 \cdot (0 - 0)^2} \\ &= \sqrt{w_0 + w_0^2}. \end{aligned} \tag{5.7}$$

The definition of $w(c)$ allows the classes to be structured in a form of a directed acyclic graph (DAG). Figure 5.3c depicts an example of a directed acyclic graph structured hierarchy (the class c_6 has two parents: c_1 and c_4). In a general case, a directed acyclic graph-like hierarchy can have two interpretations: if an example belongs to a given class c , it either a) also belongs to all super-classes of c , or b) belongs to at least one superclass of c . Here, we adapt the first case, i.e., the multiple inheritance interpretation.

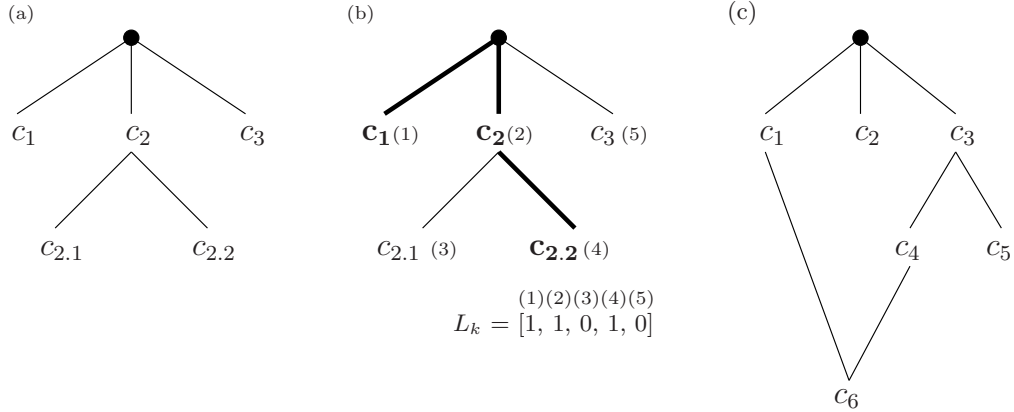


Figure 5.3: Toy examples of hierarchies structured as a tree (a) and as a directed acyclic graph (c). Class label names contain information about the position in the hierarchy, e.g., $c_{2.1}$ is a subclass of c_2 . The set of classes $S_1 = \{c_1, c_2, c_{2.2}\}$, shown in bold (b) are represented as a vector (L_k).

The variance function used for tree-shaped hierarchies uses the weighted Euclidean distance between the class vectors (as defined in Equation 5.6), where the weight of a class depends on its depth in the hierarchy. Note that when the hierarchy is shaped as a DAG, then the classes do not have a single path from the top-node (e.g., see class c_6 in Figure 5.3c), i.e., the depth of a class is not unique. To resolve this issue, we adopt the recommendation of Vens et al. (2008): The weight of a given class is calculated as an average of all the weights according to possible paths from the top-node to that class.

A classification tree stores in a leaf the majority class for that leaf, which is the tree's prediction for all examples that arrive in the leaf. In the case of HMLC, an example may have multiple classes, thus the notion of *majority class* does not apply in a straightforward

manner. Instead, the mean $\bar{L}$ of the class vectors of the examples in the leaf is stored as a prediction. Note that the value for the i^{th} component of $\bar{L}$ can be interpreted as the probability that an example arriving at the given leaf belongs to class c_i .

The prediction for an example that arrives at the leaf can be obtained by applying a user defined threshold τ to the probability; if the i^{th} component of $\bar{L}$ is above τ , then the examples belong to class c_i . When a PCT is making a prediction, it preserves the hierarchy constraint (the predictions comply with the parent-child relationships from the hierarchy) if the values for the thresholds τ are chosen as follows: $\tau_i \leq \tau_j$ whenever $c_i \leq_h c_j$ (c_i is ancestor of c_j). The threshold τ is selected depending on the context. The user may set the threshold such that the resulting classifier has high precision at the cost of lower recall or vice versa, to maximize the F-score, to maximize the interpretability or plausibility of the resulting model etc. In this work, we use a threshold-independent measure (i.e., precision-recall curves) to evaluate the performance of the models.

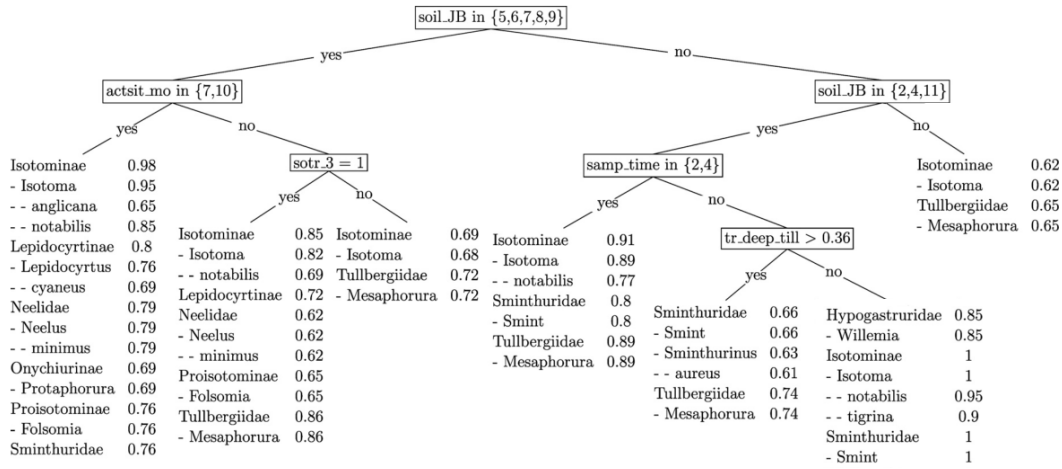


Figure 5.4: Example of a predictive clustering tree for predicting multiple hierarchically organized discrete variables. The tree predicts the presence of several organisms living in the Danish agricultural soils (Levatić, Kocev, Debeljak, & Džeroski, 2015). The numbers beside the taxa represent the probabilities of presence (i.e., the mean vectors of the examples in the leaf). Only the taxa present with probability greater than 0.6 are listed.

5.2 Ensembles for Predicting Structured Outputs

We consider ensembles of PCTs for structured output prediction, as implemented by Kocev et al. (2013) in the CLUS system. The ensembles of PCTs are constructed by using the bagging (Breiman, 1996a) and random forests (Breiman, 2001) methods, which are often used in the context of decision trees. The algorithms of these ensemble learning methods are presented in Algorithm 5.2. Bagging (Algorithm 5.2, left) is an ensemble method that constructs the different classifiers by making bootstrap replicates of the training set and using each of these replicates to construct a predictive model. Each bootstrap sample is obtained by randomly sampling training instances, with replacement, from the original training set, until an equal number of instances as in the training set is obtained. Breiman (1996a) showed that bagging can give substantial gains in predictive performance, when applied to an unstable learner (i.e., a learner for which small changes in the training set result in large changes in the predictions), such as classification and regression tree learners.

Algorithm 5.2: The ensemble learning algorithms: bagging and random forests. Here, E is the set of the training examples, k is the number of trees in the forest, and $f(D)$ is the size of the feature subset considered at each node during tree construction for random forests.

Procedure Bagging**Input:** A dataset E ,a number of trees k **Output:** Forest of PCTs $F = \emptyset$ **for** $i = 1$ **to** k **do** $E_i = \text{bootstrap}(E)$ $T_i = \text{PCT}(E_i)$ $F = F \cup \{T_i\}$ **end****return** F **Procedure** RForest**Input:** A dataset E , a number of trees k , a feature subset size $f(D)$ **Output:** Forest of PCTs $F = \emptyset$ **for** $i = 1$ **to** k **do** $E_i = \text{bootstrap}(E)$ $T_i = \text{PCT_rnd}(E_i, f(D))$ $F = F \cup \{T_i\}$ **end****return** F

A random forest (Algorithm 5.2, right) is an ensemble of trees, where diversity among the predictors is obtained by using bootstrap replicates as in bagging, and additionally by changing the set of descriptive attributes during learning. To learn a random forest, the PCT algorithm for tree construction (Algorithm 5.1) is changed to *PCT_rnd*: randomized version of the selection of attributes, which replaces the standard selection of attributes. More precisely, at each node in the decision trees, a random subset of the descriptive attributes is taken, and the best attribute is selected from this subset. The number of attributes that are retained is given by a function f of the total number of descriptive attributes D (e.g., $f(D) = 1$, $f(D) = \lfloor \sqrt{D} + 1 \rfloor$, $f(D) = \lfloor \log_2(D) + 1 \rfloor \dots$). By setting $f(D) = D$, we obtain the bagging procedure.

To construct an ensemble model for predicting a certain type of structured output, corresponding type of PCTs is used as a base model. For example, to construct an ensemble for the HMLC task, PCTs for HMLC are used as a base model. The prediction of an ensemble for a new instance is obtained by combining the predictions of all the base predictive models from the ensemble. For regression tasks, predictions of the base models are combined by taking their average, while for classification tasks, different aggregation schemes can be applied, such as the majority of probability distribution voting. We used probability distribution voting, as suggested by Bauer and Kohavi (1999). In the probability distribution voting, the base trees predict the probability that an example belongs to each possible class. Thus, each base tree gives its vote (i.e., probability estimate) for each class separately. At the end, the predicted class is the one that has the highest sum of probabilities from all base trees.

Chapter 6

Self-Training for Predicting Structured Outputs with Tree Ensembles

In this chapter, we present the semi-supervised method for predicting structured outputs based on the self-training paradigm. We extend the self-training approach to perform SSL for structured output prediction by using random forests of predictive clustering trees. In self-training, a model iteratively uses its own most reliable predictions in the learning process. Namely, reliable predictions on the unlabeled examples are added to the training set as additional examples used to learn the models. Hence, a good measure for the reliability of predictions is essential. We propose several such estimates, based on mechanisms provided within ensemble learning. In addition, we use benchmark scores (oracle and random) to empirically determine the performance limits of self-training. We also propose an approach to automatically select a threshold for the identification of the most reliable predictions to be used in the next iteration. Finally, we perform an empirical evaluation of the proposed method on a large collection of datasets.

6.1 Methodology

In this section, we present our iterative semi-supervised approach for the task of SOP. The approach is based on the predictive clustering framework and the self-training paradigm. In a nutshell, the proposed approach works as follows. To begin with, a predictive model (i.e., a random forest of PCTs) is constructed by using all of the available labeled examples (the training set). Next, the predictive model is applied to the unlabeled examples, i.e., it produces labels for the unlabeled examples. The examples with the most reliable predictions are then selected and added to the training set. Next, a new predictive model is constructed using the updated training set. The learning algorithm continues until a stopping criterion is satisfied. The pseudo-code of the self-training algorithm for MTR with ensembles of PCTs (named CLUS-ST) is outlined in Algorithm 6.1.

In the description of the CLUS-ST algorithm, three key notions require a precise definition: 1) the reliability scoring function, 2) the threshold on the reliability scores and 3) the stopping criterion. Self-training strongly relies on the assumption that its own (most reliable) predictions are correct. Hence, the most crucial part of the algorithm is to define an appropriate reliability scoring function. A good reliability scoring function should be able to discern correct predictions (thus assigning them high scores) from wrong predictions (thus assigning them low scores).

We define the reliability score of a predictive model $\mathcal{M}$ as a function:

$$\text{Reliability}^{\mathcal{M}} : E_u \rightarrow [0, 1], \quad (6.1)$$

where E_u is a set of unlabeled examples. For a given example $e \in E_u$, $\text{Reliability}^{\mathcal{M}}(e)$ denotes an estimate of the probability that the prediction $\mathcal{M}(e)$ is correct, i.e., the reliability score is a proxy for the accuracy of the prediction.

In this thesis, we consider three types of structured outputs: multi-target regression, multi-label classification and hierarchical multi-label classification. Thus, we need to define reliability estimates for each of these tasks. In the context of structured output prediction, reliability estimates have not been studied yet. Thus, in Section 6.1.1, we first propose a general framework for the calculation of reliability estimates of structured outputs. Next, in Sections 6.1.1.1 and 6.1.1.2, we present the reliability estimates for multi-target regression and (hierarchical) multi-label classification predictions, respectively. In Section 6.1.1.3 we present the benchmark scores (oracle and random).

To select the unlabeled examples that should be added to the training set, a threshold ($\tau \in [0, 1]$) on the reliability score needs to be set. If the reliability of the prediction for an unlabeled example is greater than τ , the example is moved from the unlabeled set (E_u) to the training set (E_l), together with the predicted values of its target variables. We discuss the threshold selection in Section 6.1.2 and we propose a procedure for automatic threshold selection. Self-training is iterated until the stopping criterion is met, in Section 6.1.3, we present and discuss several different stopping criteria. Finally, in Section 6.1.4 we discuss the computational complexity of the proposed self-training algorithm.

Algorithm 6.1: The learning algorithm for self-training with random forests of PCTs (CLUS-ST). Here, E_l is a set of the labeled examples, E_u is a set of unlabeled examples, k is the number of trees in the forest, D is the total number of descriptive attributes, $f(D)$ is a function which gives the size of the feature subset for random forests considered at each node during tree construction, and τ is the threshold for the reliability of predictions.

Procedure CLUS-ST($E_l, E_u, \tau, k, f(D)$)

Output Forest

repeat

$F = \text{RForest}(E_l, k, f(D))$

$E'_u = \text{predict}(F, E_u)$

foreach $e_u \in E'_u$ **do**

if $\text{Reliability}(e_u) \geq \tau$ **then**

 move e_u from E'_u to E_l

 drop e_u from E_u

end

end

until $\text{stopping_criterion}()$;

6.1.1 Reliability scores for structured outputs

Within each iteration of the self-training algorithm (see Algorithm 6.1), we solve a structured output prediction task which consists of many 'components'. For example, in MTR there are several continuous target variables. With T we will denote the number of target variables in the context of predicting multiple targets, or the number of classes in the class hierarchy in the context of HLMC. The existing reliability scoring functions are tailored

for single-target problems, i.e., classification and regression (Bosnić & Kononenko, 2008). We propose to calculate the reliability scores for a given unlabeled example with a structured output as follows. We first calculate T per-target reliability scores, i.e., we apply a reliability scoring function to each of the components of structured output separately. Next, the T reliability scores are aggregated into a single reliability score. The threshold τ is then applied to the aggregated reliability score.

We explore two aggregation schemes for the per-target reliability scores: *averaging-based* and *minimum-based* aggregation. The averaging-based scheme consists of simple averaging of the per-target reliability scores, whereas the minimum-based aggregation scheme is more conservative. There, the examples' prediction is considered as reliable as the prediction of its least reliable target. The two aggregation schemes are defined as follows:

$$Reliability_{Avg}(e) = \frac{1}{T} \sum_{i=1}^T \overline{Reliability}_i(e), \quad (6.2)$$

$$Reliability_{Min}(e) = \min_{i=1,\dots,T} (\overline{Reliability}_i(e)), \quad (6.3)$$

where $\overline{Reliability}_i(e)$ is a normalized (in $[0,1]$) reliability score for the i^{th} target variable (or i^{th} class) of an example e .

We consider two strategies of normalization of per-target scores: min-max normalization and ranking-based normalization. Formally:

$$\overline{Reliability}_i^{Norm}(e_u) = \frac{Reliability_i(e_u) - \min_{j=1\dots|E_u|} Reliability_i(e_j)}{\max_{j=1\dots|E_u|} Reliability_i(e_j) - \min_{j=1\dots|E_u|} Reliability_i(e_j)}, \quad (6.4)$$

$$\overline{Reliability}_i^{Rank}(e_u) = 1 - \frac{Rank_i(Reliability_i(e_u)) - 1}{|E_u| - 1}, \quad (6.5)$$

where $Reliability_i(e_u)$ is the reliability score for the i^{th} target variable of the example e_u , $|E_u|$ is the number of unlabeled examples, and the $Rank_i$ function gives ranks to unlabeled examples according to the reliability scores of the i^{th} target variable.

The use of the ranking-based normalization strategy is motivated by the following. It can happen that the distributions of the per-target scores can be very different, thus introducing different biases to the aggregated reliability score. Ranking-based normalization, instead of considering the absolute values of the per-target reliability scores (as in min-max normalization), considers the relative differences between the per-target ranks.

In Table 6.1, we give an illustrative example of the procedure for calculating the reliability scores according to the previously defined aggregation schemes and normalization strategies. The task considered in this example has two target variables.

6.1.1.1 Reliability scores for multi-target regression

The reliability scores for MTR ($Reliability_i(e)$) are obtained by exploiting mechanisms provided directly by the ensemble learning paradigm. More specifically, we propose two scores based on (1) the variance of the votes of the base predictive models in an ensemble and (2) random forest proximities (Breiman, 2001). These reliability scoring functions are explained in more detail in the remainder of this section.

Table 6.1: An illustrative example of using the procedure for calculating of the reliability scores according to different aggregation and normalization schemes. The same examples e_1, e_2 and e_3 are used in both the upper and lower part of the Table.

Normalized per-target reliability scores				
	Reliability scores		Aggregated scores	
	$Reliability_1^{Norm}$	$Reliability_2^{Norm}$	$Reliability_{1,2}^{Norm}_{Avg}$	$Reliability_{1,2}^{Norm}_{Min}$
e_1	0.8	0.1	$\frac{0.8+0.1}{2} = 0.45$	0.1
e_2	0.3	0.5	$\frac{0.3+0.5}{2} = 0.4$	0.3
e_3	0.9	0.4	$\frac{0.9+0.4}{2} = 0.65$	0.4

Ranked per-target reliability scores				
	Reliability scores		Aggregated scores	
	$Reliability_1^{Rank}$	$Reliability_2^{Rank}$	$Reliability_{1,2}^{Rank}_{Avg}$	$Reliability_{1,2}^{Rank}_{Min}$
e_1	$1 - \frac{2-1}{3-1} = 0.5$	$1 - \frac{3-1}{3-1} = 0$	$\frac{0.5+0}{2} = 0.25$	0
e_2	$1 - \frac{3-1}{3-1} = 0$	$1 - \frac{1-1}{3-1} = 1$	$\frac{0+1}{2} = 0.5$	0
e_3	$1 - \frac{1-1}{3-1} = 1$	$1 - \frac{2-1}{3-1} = 0.5$	$\frac{1+0.5}{2} = 0.75$	0.5

Variance score The variance score exploits the voting mechanism of the ensemble model (Carney, Cunningham, & Bhagwan, 1999). Namely, when a prediction is made for an unlabeled example by a random forest, we consider it reliable if the predictions of the individual trees in the ensemble are coherent. This variance measure has been previously used in the context of bagging, where it was found to perform the best among a variety of approaches for estimating the reliability of regression predictions (in an extensive empirical comparison (Bosnić & Kononenko, 2008)).

In each iteration of our self-training approach, a random forest $\mathcal{M}$ with k PCTs is built. The PCTs are trained on a set of labeled examples E_l and applied to a set of unlabeled examples E_u . First, for each unlabeled example $e_u \in E_u$, the per-target standard deviation r_u^i of votes of the ensemble is calculated as:

$$r_u^i = \sqrt{\frac{1}{k-1} \sum_{j=1}^k \left(tree_j^i(e_u) - \mathcal{M}^i(e_u) \right)^2}, \quad i = 1, \dots, T, \quad (6.6)$$

where $tree_j^i(e_u)$ is the vote (i.e., prediction) for e_u returned by the j^{th} tree for the i^{th} target, $\mathcal{M}^i$ is the prediction for e_u returned by the ensemble for the i^{th} target (i.e., the average of the votes across all trees for the i^{th} target), and T is the number of target variables.

In this case, in order to give high reliability scores for small standard deviation we define:

$$Reliability_i(e_u) = -r_u^i, \quad i = 1, \dots, T, \quad (6.7)$$

This reliability score can be theoretically motivated by the properties of variance. In fact, the variance represents (informally) how much the predictions differ from the mean. If we assume that the errors of the ensemble are distributed according to a normal distribution (this assumption is reasonable for most of the ensemble learning approaches and follows the central limit theorem (Bühlmann, 2012)), r_u^i can be considered as a good approximation of the variance of the errors, which we can directly associate to the concept of reliability. Actually, our approach follows some previous studies (Bosnić & Kononenko, 2008) where the variance of predictions obtained by bagging of artificial neural networks was used to estimate reliability.

Combining this reliability score with the two normalization schemes and with the two aggregation schemes leads to four configurations, named *VarianceNormAvg*, *VarianceNormMin*, *VarianceRankAvg* and *VarianceRankMin*.

Random forest proximities score This reliability scoring function relies heavily on the random forest inner mechanisms. More specifically, we exploit two such mechanisms: the measure of proximity between examples and out-of-bag error estimation. A random forest has an intrinsic proximity measure (Breiman, 2001). For two examples e and f , the proximity $p(e, f)$ is defined as the proportion of trees in the forest where e and f end in the same leaf.

When constructing ensembles from bootstrap replicates of the training data, the performance can be estimated using the out-of-bag error, i.e., the error on the examples that were not used to construct the individual base models. Breiman (1996b) presented empirical evidence that the out-of-bag error is a very accurate estimate of a nodes' errors in a regression tree. More generally, the out-of-bag error is a good estimate of the error which an ensemble model will make on unseen examples.

We propose to join these two mechanisms (proximity between examples and out-of-bag error estimation) into a reliability scoring function. Namely, we extrapolate the expected error of unlabeled examples by using the out-of-bag errors of the labeled examples in their proximity. In this way, we obtain an estimate of the ensembles' predictive performance (and conversely error) for that example. In particular, we first compute the out-of-bag error for each labeled example and then, for each unlabeled example, we calculate its proximities to all the labeled examples. If the unlabeled example e_u often falls in the same leaf with a labeled example e_l (i.e., $p(e_u, e_l)$ is high), according to the above discussion, the out-of-bag error of the labeled example e_l is a good estimate of the error of the unlabeled example e_u . Thus, if the out-of-bag errors of labeled examples in the close proximity of the unlabeled example e_u are low, then it is expected that the error of e_u will also be low (i.e., the prediction is correct).

More formally, the per-target expected error ($expErr_u^i$) for each example ($e_u \in E_u$) is calculated as follows:

$$expErr_u^i = \sum_{e_l \in E_l} (p(e_u, e_l) \cdot OOBError_i(e_l)), \quad i = 1, \dots, m, \quad (6.8)$$

where $OOBError_i(e_l)$ is the out-of-bag error for the i^{th} target of the labeled example e_l , T is the number of targets and $p(\cdot, \cdot)$ is the proximity function.

Our reliability scoring function based on the random forest proximities is similar to reliability estimation based on local neighbourhoods (Bosnić & Kononenko, 2008) – both functions consider the local neighbourhood of an example to estimate the reliability of its prediction. However, none of the scores there takes prediction errors into account. Briesemeister et al. (Briesemeister, Rahnenführer, & Kohlbacher, 2012) proposed a reliability estimate based on the errors in the local environment in the training set. This method is model-independent and requires estimation of errors on the training set, optimization of the number of neighbours and the calculation of distances among examples. In contrast, the score proposed in this paper is tailored to random forests and adds almost no additional computational overhead, except for counting the number of times a pair of examples ended in the same leaf.

As an error measure for $OOBError_i(e_l)$, we use the root-mean-square error (RMSE). Moreover, in order to give high reliability scores for small out-of-bag errors, we define:

$$Reliability_i(e_u) = -expErr_u^i, \quad i = 1, \dots, m, \quad (6.9)$$

Combining this reliability score with the two normalization schemes and the two aggregation schemes leads to four configurations, named as *RForestProxNormAvg*, *RForestProxNormMin*, *RForestProxRankAvg* and *RForestProxRankMin*.

6.1.1.2 Reliability scores for (hierarchical) multi-label classification

In MLC and HMLC, the set of labels of each example is represented as a vector with binary components, where i^{th} component of the vector is set to 1 if the example belongs to class c_i and 0 otherwise (Sections 5.1.1.2 and 5.1.2). The prediction of an ensemble of PCTs for a given unlabeled example is obtained by combining the predictions of all the base PCTs of the ensemble. Each PCT casts a vote for each of the classes $c_i, i \in 1, \dots, T$. We use these votes to estimate the reliability of predictions. The procedure for reliability score calculation is slightly different for the MLC and HMLC tasks.

In HMLC, for a given unlabeled example e_u , we estimate its per-class reliabilities as a proportion of PCTs in an ensemble that voted for a given class:

$$r_u^i = \frac{\left| \left\{ j \mid tree_j^i(e_u) = 1 \right\} \right|}{k}, i = 1, \dots, T, \quad (6.10)$$

where $tree_j^i(e_u)$ is the vote (i.e., prediction) for example e_u returned by the j^{th} tree for the class c_i , and k is the number of trees in the ensemble.

Each class c_i can have a positive (i.e., $c_i = 1$) and a negative (i.e., $c_i = 0$) label. However, in reliability score calculation for HMLC, we consider only votes for positive labels, since only positive labels (i.e., paths in hierarchy) are explicitly predicted in HMLC.

In MLC, on the other hand, both positive and negative labels are explicitly predicted. predicting 0 with high certainty is also considered as highly reliable. In other words, predicting a negative label with high certainty is also considered as highly reliable, thus negative labels are also used to estimate per-class reliabilities:

$$r_u^i = \max \left\{ \frac{\left| \left\{ j \mid tree_j^i(e_u) = 1 \right\} \right|}{k}, \frac{\left| \left\{ j \mid tree_j^i(e_u) = 0 \right\} \right|}{k} \right\}, i = 1, \dots, T. \quad (6.11)$$

Per-class reliability scores are then combined with two normalization schemes and with the two aggregation schemes leading to four configurations, named *ClassNormAvg*, *ClassNormMin*, *ClassRankAvg* and *ClassRankMin*.

6.1.1.3 Benchmark reliability scores

To assess the potential of the self-training paradigm and the reliability scoring function, we propose two benchmark scores: the random and the oracle scores. These two benchmark scores determine the potential of the self-training approach and of the proposed reliability scoring functions. The random score provides information on whether using reliability scores improves the predictive performance of self-training or not. If the use of a random reliability scoring function yields similar results as the use of a non-random reliability scoring function, this means that the reliability scoring function is not useful. The oracle score provides insight into whether the performance of self-training improves if the 'true' reliabilities of the predictions are given.

When self-training is performed with the random score, unlabeled examples are added to the training set in a random order. With the oracle score, they are added in the best possible order, that is, examples where the model makes the smallest actual error are added first. A good reliability score for self-training yields performance better than that of the

random score, and as close as possible to the one of the oracle score. The *random score* assigns a random number between 0 and 1 to each unlabeled example.

For this purpose, we generate pseudo-random numbers according to the uniform distribution in $[0, 1]$. New random scores are generated at each iteration of self-training and are assigned to the examples remaining in the unlabeled set.

The *oracle score* is defined on the basis of the actual error a predictive model makes on unlabeled examples. The SSL algorithms are typically evaluated by using the set of entirely labeled data, where unlabeled data are simulated by temporarily ignoring the labels. Knowing the exact labels of all data enables us to measure the errors made on the unlabeled data. Note that this score can not be used in practical applications of SSL. We use this score to assess how far the proposed reliability scoring functions are from the theoretically optimal score.

The oracle score is calculated as follows. In each iteration of self-training, predictions are made for the unlabeled examples. For each unlabeled example, we measure its per-target errors using the predicted and the real labels. The obtained per-target errors are then normalized to $[0, 1]$ the same way as in the other two reliability scores, so that a small (per-target) error leads to high (per-target) reliability.

Random reliability scores and oracle reliability scores with the two normalization schemes and the two aggregation schemes lead to five configurations, named *Random*, *OracleNormAvg*, *OracleNormMin*, *OracleRankAvg* and *OracleRankMin*.

6.1.2 Threshold selection

Once the predictions on unlabeled examples are made and reliability scores are calculated, it is necessary to choose the examples which will be considered as training examples in the next iteration (i.e., the ones with reliable predictions). For this purpose, a threshold on the reliability scores is needed. A properly selected threshold is of great importance for the optimal performance of the self-training approach. If an inappropriate threshold is selected, such that it allows examples with erroneous predictions to enter the training set, the performance of self-training can deteriorate.

In this work, we propose an approach to automatically identify an appropriate threshold for the reliability of predictions. A simple solution, already proposed and empirically evaluated for binary classification in (Tanha, van Someren, & Afsarmanesh, 2015), is to select 10% of the most-reliable predictions and use the mean of their reliability score as a threshold. Henceforth, we will call this solution *Tavg10%*. A variant of this solution is to include the top 10% most-reliable predictions in the next iteration. Henceforth, we will call this solution *Ttop10%*.

A more sophisticated alternative, which we propose in this paper, is to simultaneously exploit the errors and reliability scores of out-of-bag examples. In particular, at each iteration of self-training, out-of-bag reliability scores for labeled examples are calculated. Out-of-bag reliability scores of labeled examples are calculated in a similar way as the reliability scores of unlabeled examples, with the difference that, for each labeled example, only the trees for which that example was out-of-bag are considered. Then, for each candidate threshold on the reliability score, our approach evaluates if the mean of the out-of-bag error between the examples with reliability score greater than the considered threshold is *statistically significantly different* from the mean of the out-of-bag error of all the examples. Note that, in this procedure, we use only 'original' labeled examples, i.e., unlabeled examples which are added to the training set are not considered.

The basic idea is that, at each iteration, the algorithm should select the threshold such that examples with reliability scores greater than the threshold contribute to *significant* reduction in error. Candidate thresholds are all the out-of-bag reliability scores of labeled

examples, taken in descending order. When the test accepts the hypothesis that the means are different, the algorithm stops and returns the last considered threshold. The threshold found in such a way is then applied to select unlabeled examples to be added to the training set. If statistical significance cannot be reached for any of the candidate thresholds, the algorithm returns 0, meaning that all of the unlabeled examples will be added to the training set at the next iteration.

The statistical test we consider is the two-sample t-test for equal means (with a significance level of 0.05). In this algorithm for automatic threshold selection, we do not only consider trees (of the random forest) obtained at the last iteration, but all the trees obtained over all the considered iterations, thus guaranteeing stability of the threshold values. Henceforth, we will call this solution T_OOB and we will compare this solution with the variant $T_OOBInitial$ that determines the threshold only after the first iteration of the self-training framework, and uses this threshold throughout all the following iterations. The pseudo-code of the self-training method with the proposed algorithms for automatic threshold selection is presented in Algorithm 6.2.

Algorithm 6.2: The learning algorithm for self-training with automatic threshold selection based on out-of-bag errors (CLUS-SSL-AUTO). Here, E_l is a set of labeled examples, E_u is a set of unlabeled examples, k is the number of trees in the forest, D is the total number of descriptive attributes, $f(D)$ is a function which gives the size of the feature subset for random forests considered at each node during tree construction, and *initial* is an indicator variable specifying whether the threshold is automatically selected only after the first iteration or throughout all the iterations.

Procedure: CLUS-ST-Auto

Input: E_l , E_u , k , $f(D)$, *initial*

Output: Forest

$F = \text{RForest}(E_l, k, f(D))$

$E'_u = \text{predict}(F, E_u)$

if (*initial* = *true* and *isFirstIteration*())
or *initial* = *false* **then**

$\tau = \text{automaticOOB}(F, E_l^{\text{original}})$

end

foreach each $e_u \in E'_u$ **do**

if $\text{Reliability}^F(e_u) \geq \tau$ **then**

 move e_u from E'_u to E_l

 drop e_u from E_u

end

end

Until *stopping_criterion*()

return F

Procedure: automaticOOB

Input: Forest, E

Output: Threshold τ

$\text{OOBErrAll} = \text{getOOBErrors}^F(E)$

$\text{reliabilityScores} = \text{Reliability}_{OOB}^F(E)$

$\text{sortDescending}(\text{reliabilityScores})$

for each $\tau' \in \text{reliabilityScores}$ **do**

$E' = \{e \in E \mid \text{Reliability}_{OOB}^F(e) \geq \tau'\}$

$\text{OOBErrSubset} = \text{getOOBErrors}^F(E')$

 p-value =

 t-test(OOBErrAll , OOBErrSubset)

if p-value < 0.05 **then**

return τ'

end

end

return 0

6.1.3 Stopping criteria

In this work, we consider two different stopping criteria for self-training. The first one is the commonly used stopping criterion which stops self-training if no unlabeled example is moved from the set of unlabeled examples (E_u) to the training set (E_l). This stopping criterion is satisfied when one of the following conditions is met: (1) the reliability scores for all the unlabeled examples are lower than the threshold, or (2) all unlabeled examples have been moved to the training set.

The second stopping criterion we consider, named *Airbag* (Leistner et al., 2009), is designed to automatically stop the self-training procedure in the case of predictive performance degradation. This criterion has been evaluated on classification tasks, while here we consider it for structured output prediction tasks. At each iteration, the out-of-bag error of the model is recorded. If an increase in the out-of-bag error is detected from one iteration to the next one, then the self-training procedure does not continue, the current model is discarded, and the model learned in the previous iteration is considered as the final model. Similarly to the procedure for automatic threshold selection (Section 6.1.2), the calculation of the out-of-bag error is based on the 'original' labeled examples (and not on examples labeled during the self-training procedure). Note that in the case of *Airbag* stopping criteria, the first stopping criterion is also used in conjunction.

6.1.4 Computational complexity

The computational complexity of the self-training approach is the product of the number of iterations and the complexity of learning the base predictive model at each iteration. The theoretical upper bound for the number of iterations is equal to the number of unlabeled examples, assuming one unlabeled example is added to the training set per iteration. Note that this upper bound is typically never reached.

The complexity of learning the base model used in this thesis, i.e., training random forests for predicting structured outputs, depends linearly on the number of decision trees, logarithmically on the number of descriptive attributes and $N \log N$ on the number of training instances (Kocev et al., 2013). Note that (1) the number of training instances is not constant in self-training, as it increases with each iteration when unlabeled examples are added to the training set and (2) the cost of the automatic threshold selection algorithm (Section 6.1.2) is linear in the number of training instances N .

6.2 Experimental Design

In this section, we first describe the datasets used in the experimental evaluation. Next, we present the evaluation procedure, the specific parameter settings of the algorithms and the performance metrics. Finally, we state the main experimental questions of the study and describe the strategies used to answer them.

6.2.1 Data description

6.2.1.1 Multi-target regression datasets

To evaluate the predictive performance of the proposed self-training methods in the context of MTR, we use nine datasets with multiple continuous target variables. The datasets come from two different domains: environmental sciences and economy.

The majority of the data are from the area of environmental sciences. To begin with, the task in the *RF1* dataset (Spyromitros-Xioufis et al., 2014) is to predict river network flows 48 hours ahead. Next, the *SIGMEA real* and *SIGMEA simulated* datasets (D. Demšar, Debeljak, Lavigne, & Džeroski, 2005) consist of data about pollen dispersal rates on fields with genetically modified oilseed rape. Furthermore, the *Soil quality* (D. Demšar et al., 2006) and *Water quality* (Blockeel, Džeroski, & Grbović, 1999) datasets concern habitat modelling of soil and water microorganisms, respectively. Next, the *Solar flare-2* dataset (Lichman, 2013) is about the number of solar flares of a certain class that occur over a 24 hour period. Finally, the *Vegetation condition* dataset (Kocev et al., 2009) contains remote sensed vegetation data. From the domain of economy, we consider the *SCM1D* and

Table 6.2: Characteristics of the MTR datasets. N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous) and T is the number of target variables.

Dataset	Domain	N	D/C	T
RF1	Ecology	9125	0/64	8
SCM1D	Economy	9803	0/280	16
SCM20D	Economy	9803	0/61	16
SIGMEA real	Ecology	817	0/4	2
SIGMEA simulated	Ecology	10368	2/9	2
Soil quality	Ecology	1944	0/142	3
Solar flare-2	Ecology	1066	10/0	3
Vegetation condition	Ecology	16967	1/39	7
Water quality	Ecology	1060	0/16	14

SCM20D datasets (Spyromitros-Xioufis et al., 2014), where the task is to predict the price of 16 products for the next day and their mean price over the next 20 days, respectively. We can observe (see Table 6.2) that the datasets vary in their size, number of attributes and number of target variables.

6.2.1.2 Multi-label classification datasets

Table 6.3: Characteristics of the MLC datasets: N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous), $\mathcal{L}$ is the number of labels, $\overline{\mathcal{L}_L}$ is the average number of labels per example.

Dataset	Domain	N	D/C	$\mathcal{L}$	$\overline{\mathcal{L}_L}$
Birds	Audio	645	2/258	19	1.014
Corel5k	Images	5000	499/0	374	3.522
Emotions	Music	594	0/72	6	1.869
Enron	Text	1702	1001/0	53	3.378
Medical	Text	978	1449/0	45	1.245
Scene	Images	2407	0/294	6	1.074
Slovenian rivers	Ecology	1060	0/16	14	5.073
Yeast	Biology	2417	0/103	14	4.237

To evaluate the predictive performance of the proposed self-training methods in the context of MLC, we use 8 datasets, which come from seven domains: audio, image, text and music classification, ecology and biology. The main statistics of the datasets are given in Table 6.3. We can observe that the datasets vary in the size, number of attributes, number of labels and label cardinality.

The task at the Birds dataset (Briggs et al., 2013) is to predict which of the 19 pre-defined bird species are present in a ten-second audio recording. Birds are widely used as biological indicators, thus collecting data about birds is important in ecology research. Acoustic monitoring presents a cheap and efficient alternative to human surveys in collecting such data. In the Emotions dataset (Trohidis, Tsoumakas, Kalliris, & Vlahavas, 2008) examples are songs, where each song can be labeled with six emotions: sad-lonely, angry-aggressive, amazed-surprised, relaxing-calm, quiet-still, and happy-pleased.

The following two datasets are from the domain of image classification. The task at the Corel5k (Duygulu, Barnard, de Freitas, & Forsyth, 2002) dataset is to annotate the image with keywords describing the image, such as "cat", "forest", "grass" or "tiger", while the task at the Scene (Boutell et al., 2004) dataset is to identify the following six contexts on images: beach, sunset, field, fall-foliage, mountain, and urban.

Two datasets are from the domain of text categorization: Medical and Enron. Each example in the Medical (Read et al., 2011) dataset is a free-text summary of the patient's symptom history. The goal is to annotate text with the probable diseases. Enron is a labeled subset of the Enron corpus (Klimt & Yang, 2004), prepared and annotated by the UC Berkeley Enron Email Analysis Project. The corpus contains the e-mails from senior Enron officials. The goal is to categorize emails into several categories, such as genre, included/forwarded information, primary topics, and messages with an emotional tone.

The Yeast dataset (Elisseeff & Weston, 2001) is a biological dataset, where examples are genes and the goal is to predict their 14 biological functions (labels).

In the Slovenian rivers dataset the task is prediction of water organisms that live in the water of Slovenian rivers (Sašo Džeroski, Demšar, & Grbović, 2000). The data about the organisms was collected during six years (1990 to 1995) of monitoring of water quality performed by the Hydro-meteorological Institute of Slovenia (now Environmental Agency of Slovenia). Multiple labels correspond to different species of the organisms.

6.2.1.3 Hierarchical multi-label classification datasets

Table 6.4: Characteristics of the HMLC datasets: N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous), $\mathcal{H}$ is the type of the hierarchy, $|\mathcal{H}|$ is the number of nodes in the hierarchy, $\mathcal{H}_d$ is the maximal depth of the hierarchy, $\overline{\mathcal{L}_L}$ is the average number of labels per example.

Dataset	Domain	N	D/C	$ \mathcal{H} $	$\mathcal{H}_d$	$\overline{\mathcal{L}_L}$	
Danish farms	Ecology	1944	132/5	Tree	72	3	7
Slovenian rivers	Ecology	1060	0/16	Tree	724	4	25
ImCLEF07A	Images	11006	0/80	Tree	96	3	1
ImCLEF07D	Images	11006	0/80	Tree	46	3	1
ExprYeast-FunCat	Biology	3783	4/547	Tree	417	4	2.28
SeqAra-FunCat	Biology	3718	2/4448	Tree	196	4	0.94
Enron	Text	1648	0/1001	Tree	54	3	2.84

To evaluate the predictive performance of the proposed self-training methods in the context of HLMC, we use seven datasets, which come from four domains: habitat modeling, image classification, text classification and functional genomics. The main statistics of the datasets are given in Table 6.4. We can observe that the datasets vary in the size, number of attributes and characteristics of the label hierarchy.

Habitat modeling (Sašo Džeroski, 2009) focuses on spatial aspects of the distribution and abundance of plants and animals. It studies the relationships between environmental variables and the presence/abundance of plants and animals. We construct habitat models for river water organisms living in Slovenian rivers (Sašo Džeroski et al., 2000) and for soil microarthropods from Danish farms (D. Demšar et al., 2006). The Slovenian rivers is the same dataset used in the context of MLC (see Table 6.3). The data for the soil microarthropods from Danish farms describes four experimental farming systems (observed during the period 1989-1993) and a number of organic farms (observed during the period

2002-2003). The structured output space in these case studies is the taxonomic hierarchy of the species. Since different species are considered in the two domains, their respective output spaces will be different.

In image classification, the goal is to automatically annotate the image content with labels. The labels typically represent visual concepts that are present in the images. In this work, we are concerned with the annotation of medical X-ray images. We use two datasets from the 2007 CLEF cross-language image retrieval campaign (Dimitrovski, Koccev, Loskovska, & Džeroski, 2008): ImCLEF07A and ImCLEF07D. The goal in these datasets is to recognize which part of the human anatomy is present in the image or the orientation of the body part, respectively. Images are represented by using edge histograms. An edge histogram represents the frequency and the directionality of the brightness changes in the image. The structured output space consists of labels organized in hierarchy. They correspond to the anatomical (ImCLEF07A) and directional (ImCLEF07D) axis of the IRMA (Image Retrieval in Medical Applications) code (Lehmann, Schubert, Keyzers, Kohnen, & Wein, 2003).

Text classification is the problem of automatic annotation of textual documents to one or more categories. We used two datasets from this domain: Enron and Reuters. Enron (Klimt & Yang, 2004) is the same dataset used in the context of MLC (see Table 6.3), however, here the categories for e-mail classification are organized hierarchically. Reuters is a subset of the 'Topics' category of the Reuters Corpus Volume I (RCV1) (Lewis, Yang, Rose, & Li, 2004). RCV1 is a collection of English language stories published by the Reuters agency between August 20, 1996, and August 19, 1997. Stories are categorized into hierarchical groups according to the major subjects of a story, such as Economics, Industrial or Government. In both domains, the text documents are described with their respective bag-of-words representation.

In functional genomics, various data (e.g., DNA microarray measurements) are used to describe gene and protein functions. Machine learning methods are a valuable tool to predict gene functions from a predefined set of functions (Schietgat et al., 2010). Predicted functions with the highest confidence can be used to guide lab experiments and reduce the number of needed tests. We used two datasets concerned with two important model organisms: SeqAra-FunCat dataset (Clare, 2003) is concerned with gene functions of *Arabidopsis thaliana*, whereas ExprYeast-FunCat dataset (Clare, 2003) is concerned with gene functions for *S. cerevisiae* or baker's yeast. Descriptive attributes for the former dataset consist of the data calculated from amino acid sequences, such as amino acid ratios, molecular weight and sequence length. Descriptive attributes for the latter dataset consist of microarray gene expression levels measured under various experimental conditions, such as heat shock or nitrogen depletion. Gene functions for both datasets come from FunCat catalogue of gene functions (Ruepp et al., 2004), which has a tree-shaped hierarchy.

6.2.2 Experimental setup and evaluation procedure

The SSL method for MTR proposed in this study (CLUS-ST) iteratively trains random forest ensembles for MTR. Hence, we compare the predictive performance of the self-training approach to the performance of a supervised random forest (CLUS-RF) – a baseline in the experimental evaluation. Additionally, we compare the proposed self-training approach to supervised multi-output support vector regression¹ (MSVR) (Sánchez-Fernández et al., 2004; Brouwer et al., 2014). A schematic representation of the self-training pipeline is presented in Figure 6.1.

In all of the experiments, we construct random forests consisting of 100 predictive clus-

¹<https://github.com/wjb19/mimo-svr>

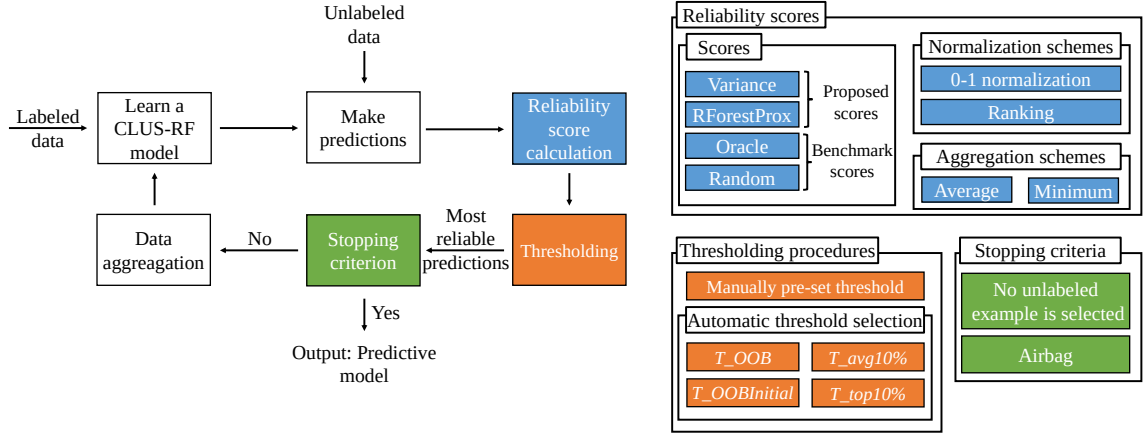


Figure 6.1: The self-training approach to SOP. The left panel gives a schematic representation of the self-training approach, with all of its components. The right panel summarizes all of the possible choices we consider with respect to reliability scores, thresholding procedures, and stopping criteria.

tering trees for MTR. The trees are not pruned and the number of random features at each internal node is set to $\lfloor \log_2(D) + 1 \rfloor$, where D is the total number of features (Breiman, 2001). We used MSVR with a radial basis kernel while optimizing the C ($2^{-5}, 2^{-3}, \dots, 2^{13}$) and σ ($2^{-2}, 2^{-1}, \dots, 2^7$) parameters in a grid search procedure by performing 10 fold cross-validation on the labeled part of the data. The epsilon parameter was fixed ($\epsilon = 0.001$) (Kuhn & Johnson, 2013).

The experimental evaluation uses different amounts of labeled data for both the supervised and SSL methods. To this end, we vary the proportion of labeled data used. The ratio of labeled (relative to unlabeled) data ranges in the following set: $\{1\%, 5\%, 10\%, 15\%, 20\%, 30\%, 50\%\}$.

For the CLUS-ST algorithm, a threshold τ on the reliability score needs to be set. In order to estimate the potential performance of the various reliability scoring functions (independently from the threshold), we consider 15 manually defined thresholds: $\tau = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99\}$, resulting in 15 executions of the CLUS-ST algorithm. From these executions, we report the best predictive performance. Obviously, we also evaluate performances obtained when the threshold is automatically set according to the four algorithms described in Section 6.1.2 (baselines: $T_{avg10\%}$, $T_{top10\%}$; our algorithms: $T_{OOBInitial}$ and T_{OOB}). For these four algorithms, we also investigate the effect of the Airbag stopping criterion.

The labeled data used for training the predictive models (both supervised and semi-supervised) are randomly selected from the available training data, while the remaining examples serve both as unlabeled data and as a test set. Namely, we temporarily remove their labels and provide the examples to the algorithm to serve as unlabeled data during training. At the end of self-training, the performance of the obtained model is evaluated on the test set composed of the unlabeled examples with the true labels restored (i.e., we consider a transductive-learning-setting). For fair comparison, supervised random forests are trained only on the labeled part of the data and evaluated on the same test set.

Note that, even though we used the transductive evaluation scenario, the proposed self-training approach is inductive method, i.e., it can make predictions on examples unseen during learning without the need to retrain the model.

The performance of self-training likely depends on the distribution of the initial labeled

training data. The performance of semi-supervised methods is, in general, known to be domain-dependent (Chawla & Karakoulas, 2005). To evaluate the robustness of our approach to the effect of possible different distributions of the labeled data, we repeated the random selection of labeled data (as described above) 10 times with different random initialization, to create 10 different labeled training sets. The predictive performance reported in the results is the average of the performance values obtained from the 10 runs.

We assess the predictive performance of the algorithms on MTR datasets by using the *root-mean-square-error* (RMSE) defined as follows:

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^m RMSE_i^2},$$

where m is the number of target variables and $RMSE_i$ is root mean square error of the i^{th} target variable.

For datasets with discrete outputs, namely for MLC and HMLC tasks, we evaluate the algorithms by using the area under the Precision-Recall curve (AUPRC), and in particular, the area under the average Precision-Recall curve ($\overline{AUPRC}$) as suggested by Vens et al. (2008). The precision and recall values are micro-averaged as follows:

$$\overline{Prec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i}, \quad \text{and} \quad \overline{Rec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i},$$

where i ranges over all classes.

To investigate whether the observed differences in performance among the methods are statistically significant, we follow recommendations of Demšar (J. Demšar, 2006). More specifically, to statistically compare the predictive performance of two methods over multiple datasets, we use the Wilcoxon signed ranks test (Wilcoxon, 1945), while for comparison of multiple methods, we use the corrected Friedman test and the post-hoc Nemenyi test (Nemenyi, 1963). We present the result from the Nemenyi post-hoc test with an average ranks diagram. The ranks are depicted on an axis, in such a manner that the best ranking algorithms are at the right-most side of the diagram. The algorithms that do not differ significantly (in performance) for a significance level of 0.05 are connected with a line.

All experiments were performed on a computer cluster which has 44 nodes and 984 central processing units (CPUs) in total: 9 nodes with 16 CPUs with an AMD Opteron processor at 800GHz on 64 GB of RAM with the Fedora 24 operating system, 10 nodes with 24 CPUs with an AMD Opteron processor at 1900GHz on 128 GB of RAM with the Fedora 24 operating system, and 25 nodes with 24 CPUs with an AMD Opteron processor at 1400GHz on 256 GB of RAM with the Fedora 24 operating system. The CLUS-ST and CLUS-RF algorithms are implemented in the Java programming language (version 1.6.), while MSVR is implemented in the C++ programming language.

6.2.3 Experimental questions

The principal goal of SSL is to improve the predictive performance of supervised learning by exploiting the information contained in unlabeled data. Therefore, the first question we want to answer in this study is: *Can the proposed self-training method improve over the performance of supervised learning?* We address this question by comparing the predictive performance of the proposed self-training method to the predictive performance of a supervised random forest. In this comparison, we consider that the optimal threshold for reliability scores is provided by an ‘oracle’, i.e., out of the 15 threshold candidates considered, we use the one which performs the best on the test set. Note that such a threshold

selection procedure is not attainable in practice, however it suffices for answering the first research question.

If the answer to the first question is favourable, we can then proceed to the following two questions: (1) *Is the performance increased due to the usage of reliability scores?*, and (2) *Which reliability score produces the largest improvement in performance?* To this end, we compare the proposed reliability scores to the benchmark scores (random score and oracle score). For question (1), we compare the performance of the proposed reliability scores with the performance obtained using random scores. If the proposed reliability scores have better predictive performance than random scores, then it is worthwhile to use a reliability scoring function and to look for an optimal function for this purpose. As for question (2), we compare the performance of the proposed reliability scores with the performance obtained using oracle scores. We expect that the oracle score performs best and we are optimistic that the performance of the proposed reliability scores is reasonably close to the performance of the oracle score.

Finally, we address the following question: *Is an automatic threshold selection algorithm able to exploit the advantage (if any) introduced by the reliability scores used in self-training?* To answer this question we evaluate the four different automatic threshold selection algorithms implemented in CLUS-ST and compare them with the supervised random forest.

6.3 Result and Discussion

In this section, we present the results obtained from the empirical evaluation of the proposed self-training algorithm (CLUS-ST) on different structured output predictions tasks: predicting multiple continuous targets, predicting multiple discrete targets and hierarchical multi-label classification. We first present the evaluation of the methods' performance obtained using various portions of the data as training. Second, we compare the performance of SSL (CLUS-SSL) with the best performing reliability score with supervised learning (CLUS-RF). Third, we illustrate a complete run of the CLUS-ST method on a single dataset. Next, we compare the performance of CLUS-ST with supervised learning (CLUS-RF) when automatic threshold selection algorithms are used. Finally, we examine the per-target performance of the proposed method. In this section we report summarized results: The complete and raw results are reported in Appendix A.

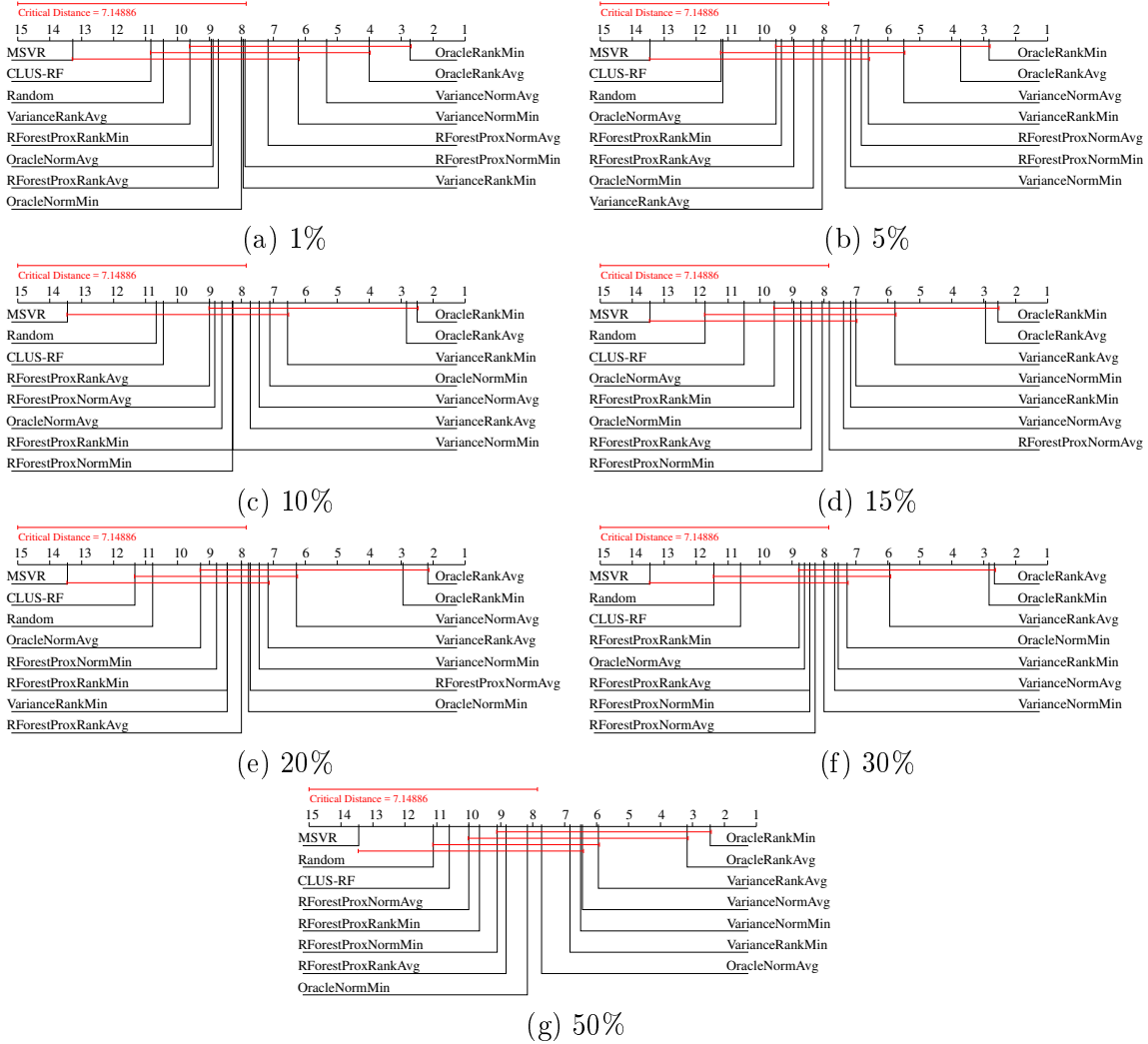
6.3.1 Multi-target regression

6.3.1.1 Comparison of self-training and supervised random forests

The first results we present are summarized in Figures 6.2 and 6.3. Figure 6.2 gives the statistical evaluation of the performance of various methods using different portions of labeled data. The Nemenyi post-hoc analysis is performed for each portion of labeled data separately. Figure 6.3 summarizes the average rank diagrams and presents a global overview of the experimental evaluation. More specifically, this figure is obtained by plotting the multiple critical diagrams from the Nemenyi post-hoc test (from Figure 6.2) jointly as follows. On the x-axis we depict the percentage of labeled data used, while on the y-axis we depict the average rank of a given method when applied to such data. These lines should not be treated as curves but as a parallel coordinate representation of the results of the statistical analysis. Figure 6.3 thus facilitates easier interpretation and understanding of the statistical analysis.

From the results, we can make several interesting observations. First of all, a major conclusion is that using unlabeled data (SSL) clearly improves the predictive performance

Figure 6.2: Average ranks diagrams for the performance of the supervised algorithms (CLUS-RF and MSVR) and semi-supervised self-training with various reliability scores. The percentage of labeled data varies from 1% to 50%. Each graph represents two different kinds of information: the ranking among the algorithms (the algorithms positioned at the rightmost side of each graph are the best performing) and the statistical significance of the difference between pairs of algorithms (if their distance is less than the critical distance (at $p\text{-value} = 0.05$) there is no statistically significant difference between the two). The critical distance for all diagrams is 7.14886.

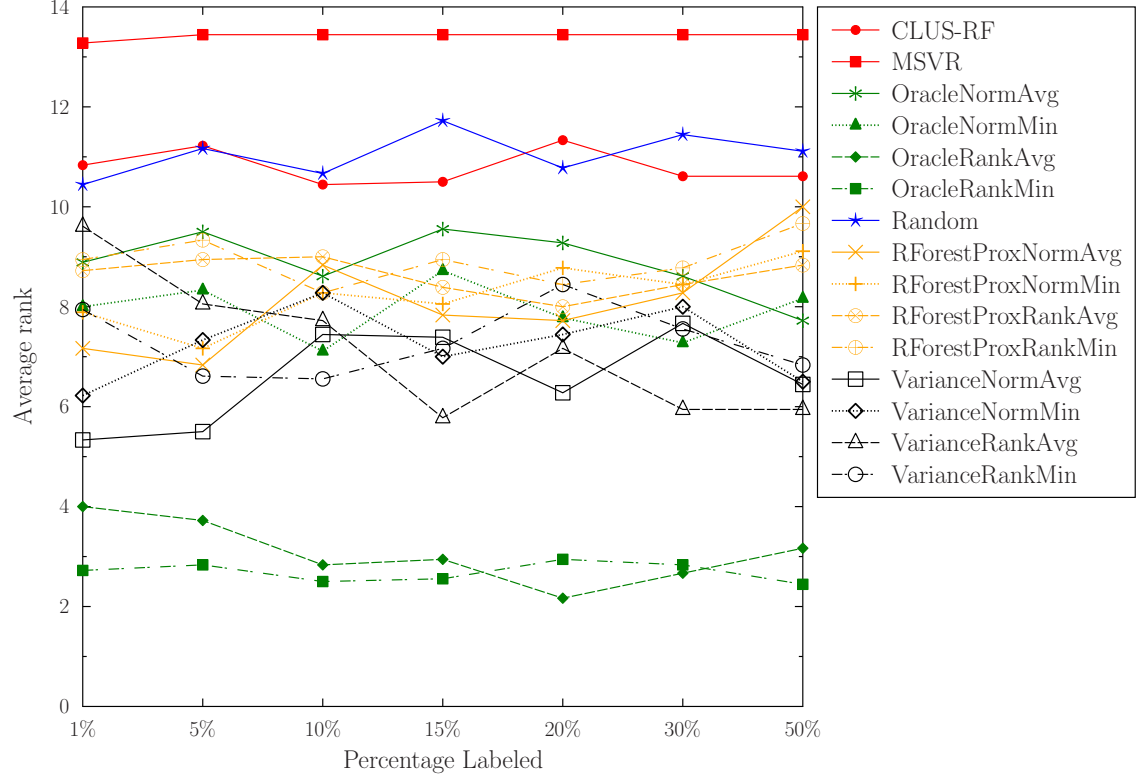


of supervised random forests (CLUS-RF). Moreover, SSL using the benchmark oracle score with ranking-based aggregation (*OracleRankAvg* and *OracleRankMin*) is statistically significantly better than the supervised method in all of the settings with different percentages of labeled data. Furthermore, as hypothesized, the results of the proposed reliability scores are bounded by the results of the benchmark scores: *Random* is always ranked the lowest, while *OracleRankAvg* and *OracleRankMin* are always ranked the highest (Figure 6.3). Finally, as expected, CLUS-RF and *Random* show similar behaviours.

A comparison of supervised and semi-supervised random forests to support vector regression (MSVR) reveals that both supervised and semi-supervised random forests outperform MSVR by a large margin. MSVR is better than CLUS-ST and CLUS-RF only on 1

out of 9 datasets (Soil quality), while on the other 8 datasets, random forests are superior to MSVR across all percentages of labeled data (Tables A.1 to A.7).

Figure 6.3: Average ranks for different percentages of labeled data (from 1% to 50%) of supervised methods (red lines), self-training with the proposed reliability scores (black and orange lines) and self-training with benchmark scores (blue and green lines).



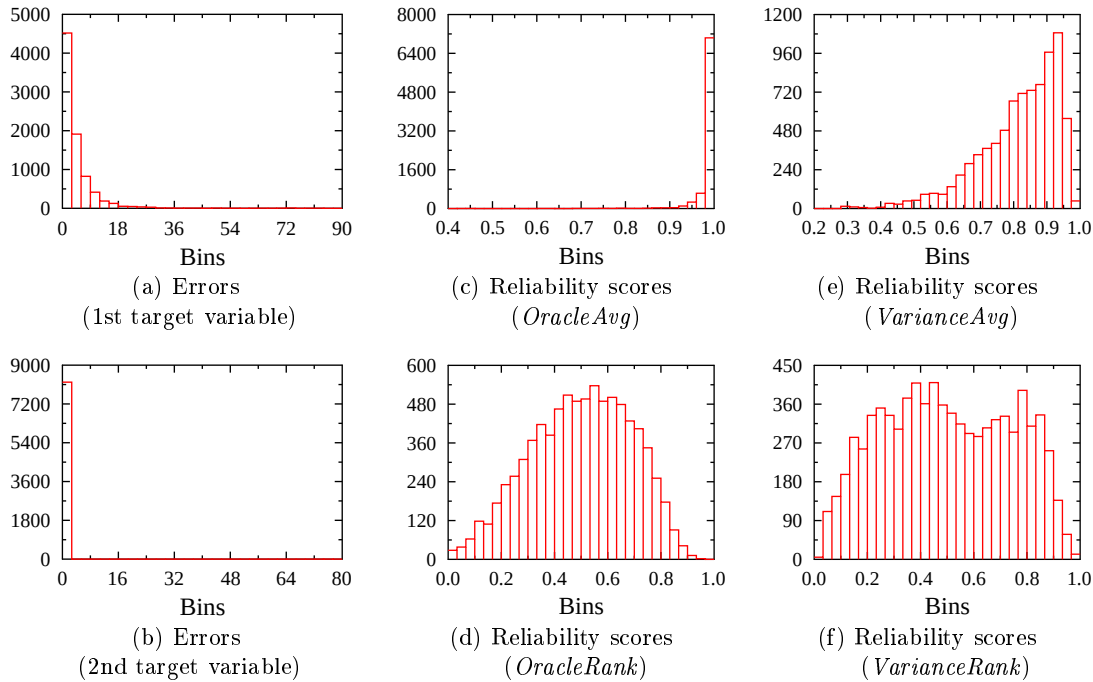
6.3.1.2 Analysis of performance of different reliability scores

Next, we examine the contributions of the specific reliability scoring functions to the predictive performance (Figure 6.2). All the proposed reliability scores (in all their variants) perform better than the *Random* score. This observation confirms that the used estimators of the reliability of predictions help to discern correct from erroneous predictions. Considering this and the fact that the oracle score with ranking-based averaging yields the best results, we conclude that instance selection in self-training for MTR strongly influences the predictive performance. In other words, if unlabeled instances are selected based on the reliability of their predictions during self-training, rather than selected randomly, better performance is achieved.

This finding is somewhat in disagreement with the empirical evaluation of self-training and co-training performed in the context of classification by Yuanyuan Guo, Niu, and Zhang (2010). They concluded that selecting the most confident unlabeled examples is not necessarily superior to random selection. In our study, random selection typically led to degradation of performance of the base model, and performed worse than the proposed reliability scores. Our results suggest that a proper reliability measure (i.e., one able to discern correct from erroneous predictions) will yield better performance of self-training than random instance selection.

The better ranking of SSL methods with the proposed reliability scores as compared to

Figure 6.4: The distributions of per-target errors (discretized into bins) made on unlabeled examples after the initial iteration of self-training are very skewed for the target variables of the *RF1* dataset when 10% of the data are labeled. These are depicted for the first two target variables (a, b), similar distributions are observed also for the other six target variables of the *RF1* dataset. The reliability scores (discretized into bins) for unlabeled examples calculated by averaging normalized per-target errors are biased towards high reliability (c, e), while the scores calculated by averaging ranks of per-target errors are normally distributed (d, f).



the supervised method is consistent across the different percentages of labeled data. Reliability scores based on the variance of votes of an ensemble, in most of the cases, perform better than reliability scores based on random forest proximities. Among the variance-based reliability scores, *VarianceNormAvg* is clearly one of the best performing. Moreover, the difference in performance between the two best scoring functions (*OracleRankAvg* and *OracleRankMin*) is not statistically significant.

The proposed reliability scores use two aggregation schemes (average and minimum), and two normalization schemes (min-max and ranking-based normalization) for the per-target scores. The results suggest that the two aggregation schemes do not show significant differences in performance. On the other hand, between the two normalization approaches, min-max normalization is favourable, since *VarianceNorm* and *RForestProxNorm* are (in most of cases) ranked better than their counterparts based on ranking (*VarianceRank* and *RForestProxRank*). However, this is not the case with the benchmark oracle score: *OracleRank* has better predictive performance than *OracleNorm*. We find this to be an artefact of the (skewed) distribution of errors on unlabeled examples. We further illustrate this phenomenon in Figure 6.4 for the *RF1* dataset (similar observations are made also for other datasets). In particular, we report the frequencies of the RMSE errors discretized into bins in Figure 6.4a and 6.4b. We see that the distributions of per-target errors of unlabeled examples are highly skewed, where few examples have much higher errors than the rest of examples.

Next, recall that the oracle scores (*OracleRank* and *OracleNorm*) are calculated by

using the actual errors on unlabeled examples. The *OracleNorm* reliability score averages (or takes the minimum of) the normalized per-target errors, therefore, such distributions of per-target errors yield reliability scores biased towards high reliability scores (i.e., the majority of unlabeled examples are given a reliability score close to one, Figure 6.4c). In other words, during the initial iterations of self-training, erroneously predicted unlabeled examples are easily deemed reliable, and are thus likely to be added to the training set. A prediction error made in early iterations of self-training can propagate itself in the next iterations, leading to a degradation of the performance.

Regardless of the relatively poor performance of self-training with the *OracleNorm* score, the reliability scores given by *OracleNorm* are still “oracle” scores, i.e., from two unlabeled examples, the one with the smaller error will always get the higher reliability score. It can be expected that fine tuning the reliability threshold in the region of high reliability (from 0.9 to 0.99) would likely result in improved performance of *OracleNorm*. The *OracleRank* reliability score does not suffer from this problem, since the *ranks* of the per-target errors are averaged (and not the errors themselves), which yields normally distributed (i.e., unbiased) scores (Figure 6.4d).

An additional observation we can make from Figures 6.4e and 6.4f is that the frequency distribution of the (Norm and Rank) variance reliability scores conforms to the frequency distribution of the corresponding (Norm and Rank) oracle reliability scores. This is an indication that our reliability scores, in this particular dataset, approximate the real distributions of the errors and, thus, identify the best predictions to be considered as reliable.

Based on this analysis, we select the best performing aggregation scheme for each of the different types of reliability scores (Variance, Random forest proximities and Oracle) as follows: *VarianceNormAvg*, *RForestProxNormMin* and *OracleRankMin*. To better illustrate the difference in the performance, we compare the performance of these against the performance of supervised random forests (CLUS-RF) and random scores (*Random*). The results of the statistical analysis (Friedman test and Nemenyi post-hoc analysis) are given in Figure A.1. The findings from this simplified analysis concur with the findings above.

As noted before, among the proposed reliability scores, one of the best performing ones is *VarianceNormAvg*. To inspect if the differences in performance at the various percentages of labeled data between this reliability score and supervised learning (CLUS-RF) are statistically significant, we use the Wilcoxon signed-rank test. The results of the Wilcoxon test are given in Table 6.5. They show that self-training with *VarianceNormAvg* is statistically significantly better than CLUS-RF (p -value < 0.05) for all percentages of labeled data.

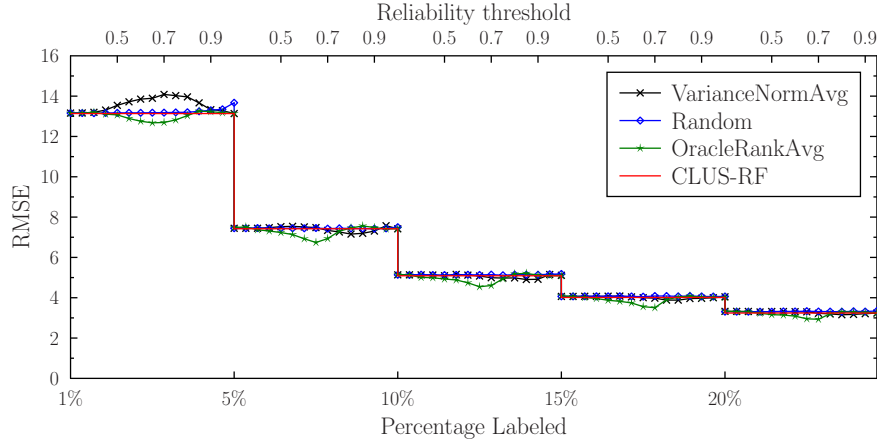
Table 6.5: Results of the Wilcoxon signed-rank test applied to the performance of self-training with *VarianceNormAvg* reliability score and the performance of supervised random forest (CLUS-RF) on the 9 datasets considered in this study. In bold, we report significant p -values (< 0.05).

% Labeled	1%	5%	10%	15%	20%	30%	50%
p -value	1.5×10^{-5}	1.7×10^{-8}	2.2×10^{-6}	4×10^{-8}	1.7×10^{-6}	4.2×10^{-5}	4.8×10^{-6}

6.3.1.3 The influence of the τ parameter

In order to demonstrate the possible effects of the value of the reliability threshold on the performance of self-training, we present an example of results obtained on the RF1 dataset by applying the different thresholds (Figure 6.5). These results show that selecting a too permissive threshold (i.e., smaller value for the threshold) can allow wrongly predicted

Figure 6.5: A comparison of the predictive performance of random forests (CLUS-RF) and self-training with different reliability scores (*VarianceNormAvg*, *Random* and *OracleRankAvg*) on the RF1 dataset illustrates the possible effects of threshold selection: A too permissive threshold for the *VarianceNormAvg* reliability score leads to worse performance than random instance selection, while a too strict threshold for the *OracleRankAvg* reliability score does not allow self-training to benefit from the unlabeled data.



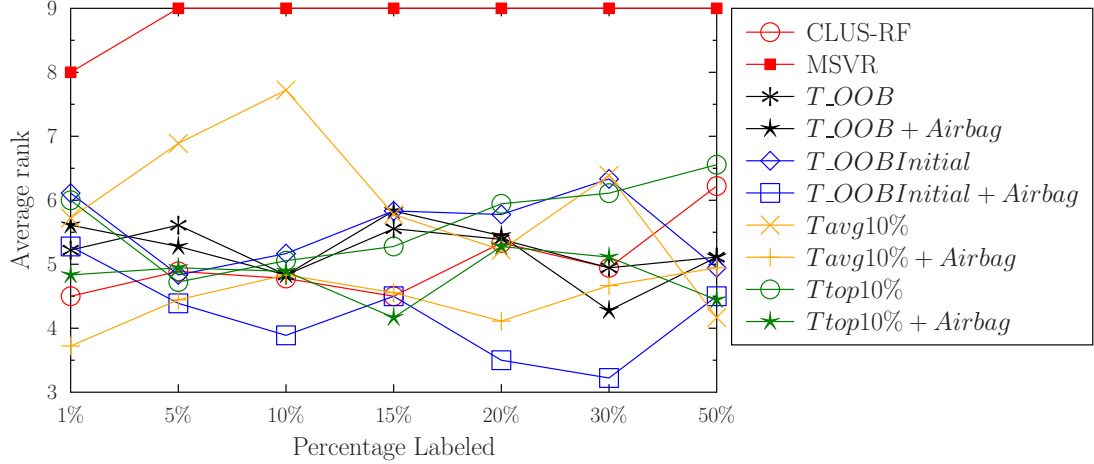
examples to enter into the training set, leading to a degradation of the performance. For example, if a threshold smaller than 0.85 on the *VarianceNormAvg* reliability score at 1% of labeled data is chosen, a performance worse than with the random instance selection can be observed (Figure 6.5). On the other hand, a too stringent threshold can prevent self-training to learn from unlabeled data, i.e., self-training will not improve the performance of supervised learning. For example, if a threshold greater than 0.8 is set for the *OracleRankAvg* reliability score at 1% or 5% of labeled data on the RF1 dataset, none, or very few, of the unlabeled examples enter the training set, meaning that we miss the opportunity to improve performance by using unlabeled data (Figure 6.5).

6.3.1.4 Automatic threshold selection

After the former discussion, it is clear that the reliability threshold selection is critical to the success of the self-training approach to SSL. We next investigate the effectiveness of the algorithms for automatic threshold selection that we have proposed. Figure 6.6, allows us to compare results obtained by CLUS-RF and MSVR with results obtained when the four automatic threshold selection algorithms (*Tavg10%*, *Ttop10%*, *T_OOBInitial* and *T_OOB*) are used in CLUS-SSL (with the *VarianceNormAvg* reliability score). The results confirm that automatic threshold selection is not an easy task in self-training and that choosing the best algorithm for this task is crucial to profit from the semi-supervised learning framework. In the specific case we consider, we can see that the algorithm proposed by Tanha et al. (2015) (i.e., *Tavg10%*) in most of cases, puts semi-supervised learning at a disadvantage. This is not the case for the algorithms we propose in this paper (*T_OOB* and *T_OOBInitial*). These algorithms materialize the SSL advantage by exploiting the out-of-bag error of the random forest (even if, according to the Wilcoxon signed rank test, there is no clear statistical evidence that *T_OOB* and *T_OOBInitial* outperform CLUS-RF). Moreover, adapting the threshold at each iteration is generally better than choosing it only once after the initial iteration (*T_OOB* vs *T_OOBInitial*).

When we consider automatic threshold selection in combination with the Airbag stopping criterion (Figure 6.6), we can see that we are able to considerably improve performance

Figure 6.6: Average ranks for different percentages of labeled data (from 1% to 50%) of supervised methods (red lines), and self-training *VarianceNormAvg* with the baseline automatic thresholding algorithms (orange and green lines) and self-training *VarianceNormAvg* with the proposed automatic thresholding algorithms (black and blue lines).



and outperform, also with a statistically significant difference, CLUS-RF. This is especially true for *T_OOBInitial* when the percentage of labeled examples is greater than 10% (see Table 6.6). When we have a small percentage of labeled examples, the simple algorithm *Tavg10%* outperforms *T_OOBInitial*. This difference between the two algorithms for automatic threshold selection was expected. In fact, the out-of-bag error is computed only on labeled examples and this can lead to wrong decisions about thresholds when we have very few labeled examples. Moreover, *Tavg10%* outperforms CLUS-RF when the percentage of labeled examples is very small ($= 1\%$, see Table 6.6 and Figure 6.6). We notice (Table 6.7) that the Airbag stopping criteria strongly reduces the average number of iterations, confirming that a large number of iterations, without an optimal threshold, is not beneficial.

Since the self-training method iteratively re-trains the base method, obviously, potential predictive performance of self-training comes with a cost of increased learning time. Table 6.8 reports learning times of CLUS-RF, MSVR and self-training. We can observe that the learning time of self-training can be up to several hundred times larger than the one of CLUS-RF (e.g, for *Tavg10%* or *T_OOBInitial*). However, as we previously stated, the Airbag stopping criterion is not only beneficial for predictive performance, but it also greatly reduces the number of iterations of self-training. Since the learning time of self-training is strongly correlated with the number of iterations (Tables 6.7 and 6.8),

Table 6.6: P-values of the Wilcoxon signed-rank test applied to the performances (RMSE) of a supervised random forest (CLUS-RF) and self-training with algorithms for automatic threshold selection (*T_OOBInitial* + *Airbag* and *Tavg10%* + *Airbag*) on the 9 datasets considered in this study. In bold, we report significant p -values (< 0.05). The '-' sign denotes that CLUS-RF is better, while '+' that self-training is better.

% Labeled	1%	5%	10%	15%	20%	30%	50%
<i>T_OOBInit.</i> + <i>Airbag</i>	0.41 (+)	0.162 (-)	0.251 (-)	0.034 (+)	0.01 (+)	0.008 (+)	0.012 (+)
<i>Tavg10%</i> + <i>Airbag</i>	0.13 (+)	0.075 (-)	0.286 (-)	0.33 (+)	0.32 (+)	0.186 (-)	0.015 (+)

Table 6.7: Average number of iterations (across the 9 dataset considered in this study) of the self-training with different algorithms for automatic reliability threshold selection and manual threshold selection (*VarianceNormAvg*).

Method	Percentage labeled						
	1%	5%	10%	15%	20%	30%	50%
VarianceNormAvg	15.3	37.4	21.0	64.5	31.2	34.9	45.2
T_OOB	22.3	42.3	36.8	35.2	34.7	34.9	31.1
$T_OOB + Airbag$	6.6	7.2	6.7	7.6	8.3	8.5	7.9
$T_OOBInitial$	21.2	67.5	49.6	51.3	52.1	58.3	40.0
$T_OOBInitial + Airbag$	3.4	3.5	3.5	3.2	3.2	3.1	2.8
$Ttop10\%$	11.0	11.0	11.0	11.0	11.0	11.0	11.0
$Ttop10\% + Airbag$	3.8	3.3	3.2	3.1	3.4	2.9	2.7
$Tavg10\%$	77.5	90.0	82.0	73.8	69.3	63.4	40.0
$Tavg10\% + Airbag$	4.1	2.8	2.7	2.5	2.7	2.4	2.4

Table 6.8: Average learning time in seconds (across the 9 dataset considered in this study) of supervised methods (CLUS-RF and MSVR), self-training with different algorithms for automatic reliability threshold selection and manual threshold selection (*VarianceNormAvg*). For the MSVR algorithm, the learning time spent for parameter optimization is presented in brackets.

Method	Percentage labeled						
	1%	5%	10%	15%	20%	30%	50%
CLUS-RF	2.4	3.5	4.5	5.8	6.6	9.5	13.9
MSVR	0.01(256.5)	0.1(1437.4)	0.2(1864.2)	0.5(3113.1)	1.7(6972.1)	2.4(13354.4)	8.5(31843.7)
VarianceNormAvg	116.9	872.8	464.9	667.9	625.9	1133.6	1989.6
T_OOB	542.7	1205	689.2	671.5	649.5	781.7	867.5
$T_OOB + Airbag$	149.9	163.6	72.5	88.1	98.2	128	167.5
$T_OOBInitial$	588.4	2118.1	1281.6	1518	1548	1704.3	1001.8
$T_OOBInitial + Airbag$	69.5	32.7	24.2	20.3	22.3	24.9	40.4
$Ttop10\%$	189.8	195.8	205.8	201.1	207.5	225.3	238.3
$Ttop10\% + Airbag$	10.3	12.2	16.7	18.5	25.2	26.3	37.7
$Tavg10\%$	1871.3	2365.9	2104.2	1847.8	1745.4	1577.3	1013.3
$Tavg10\% + Airbag$	10.6	8.6	10.7	13.4	16.8	21.1	37.2

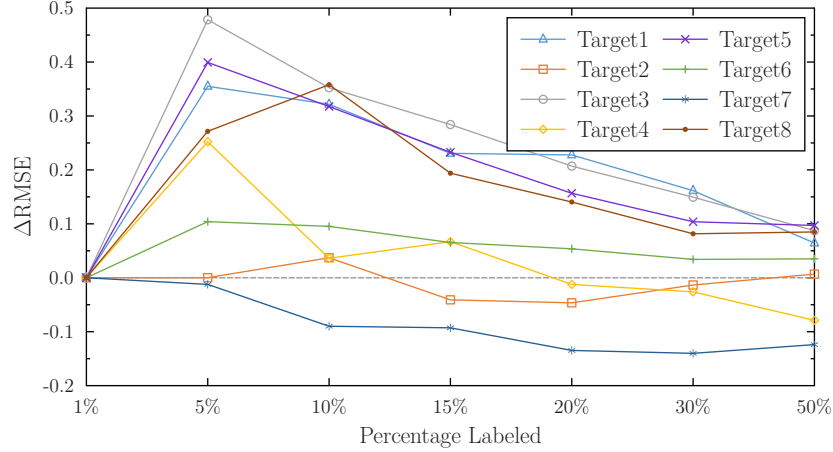
the learning time of self-training with the Airbag stopping criterion is greatly reduced if compared to self-training without this stopping criterion. The learning time of the best performing methods (i.e., $T_OOBInitial + Airbag$ and $Tavg10\% + Airbag$) is typically only 3 to 5 times larger than the learning time of CLUS-RF. The MSVR algorithm typically runs faster than CLUS-RF or self-training. This is in line with the time complexity analysis reported in Section 6.1.4, where we point out that the time complexity of our approach is $N \log N$ in the number of training instances, logarithmic in the number of attributes and linear in the number of iterations, whereas the time complexity of MSVR is linear in the number of training instances, linear in the number of descriptive attributes and linear in the number of iterations² (Sánchez-Fernández et al., 2004). However, much more time is used in MSVR for parameter optimization, especially as the amount of labeled data increases. Note that the predictive performance of MSVR can vary extremely depending on parameter values; therefore, it is advisable to perform parameter optimization.

6.3.1.5 Per-target performance analysis

Finally, Figure 6.7 presents a different perspective on the results, highly relevant to MTR problems, by illustrating the per-target RMSE performance of self-training. More specifi-

²MSVR performs a quasi-Newton approach in which each iteration better approximates the model.

Figure 6.7: Analysis of per-target performance for the RF1 dataset, in terms of difference in performance (ΔRMSE) between CLUS-RF and self-training with the *VarianceNormAvg* reliability score. Positive values suggest that self-training is better, while negative that CLUS-RF is better. Zero means that there is no difference in performance.



cally, it presents the per-target RMSE improvements of self-training with the *VarianceNormAvg* reliability score over supervised random forest (CLUS-RF). These results show that the improvement provided by the SSL setting is not necessarily uniform across all of the different target variables. Self-training improves performance over CLUS-RF for all of the targets when the percentage of labeled examples is relatively small. Otherwise it improves the performance for most of the target variables, but can degrade the performance for some target variables (e.g., target no. 7 for $>5\%$ of labeled data).

6.3.2 (Hierarchical) multi-label classification

In this section, we report the results of the self-training method on MLC and HMLC datasets. Since the results on all the datasets considered are similar, here we report the results on one MLC and one HMLC dataset, i.e., Yeast and ImCLEF7D, respectively. The complete results are presented in Appendix A.2 and A.3). In all of the experiments, *ClassNormAvg* reliability score was used (see Section 6.1.1.2).

We can observe that (Figure 6.8, Appendix A.2 and A.3), contrary to the MTR task, self-training for MLC and HMLC was not able to improve notably over supervised random forests on any of the datasets considered. More specifically, considering all of the 15 threshold candidates for reliability score, predictive performance of CLUS-ST was the same, worse or only slightly better than the performance of CLUS-RF on all of the MLC and HMLC datasets considered. This finding gives a negative answer to the first experimental question we pose (see Section 6.2.3), thus, for the MLC and HMLC tasks, we do not proceed with further investigation. Note that, even though only results with the *ClassNormAvg* reliability score are presented, we have also performed experiments with other reliability scores, (i.e., *ClassNormMin*, *ClassRankAvg* and *ClassRankMin*), however the findings are similar, thus we do not report results in this thesis.

Next, we analyze the possible reasons for the failure of self-training at the MLC and HMLC tasks. For this purpose, we performed the experimental investigation with the *OracleNormAvg* reliability score (Figure 6.8, Appendix A.2 and A.3). We can observe that, self-training with oracle score was able to improve over supervised random forests. More specifically, improvements are noted on the majority of MLC datasets (Figure 6.8a

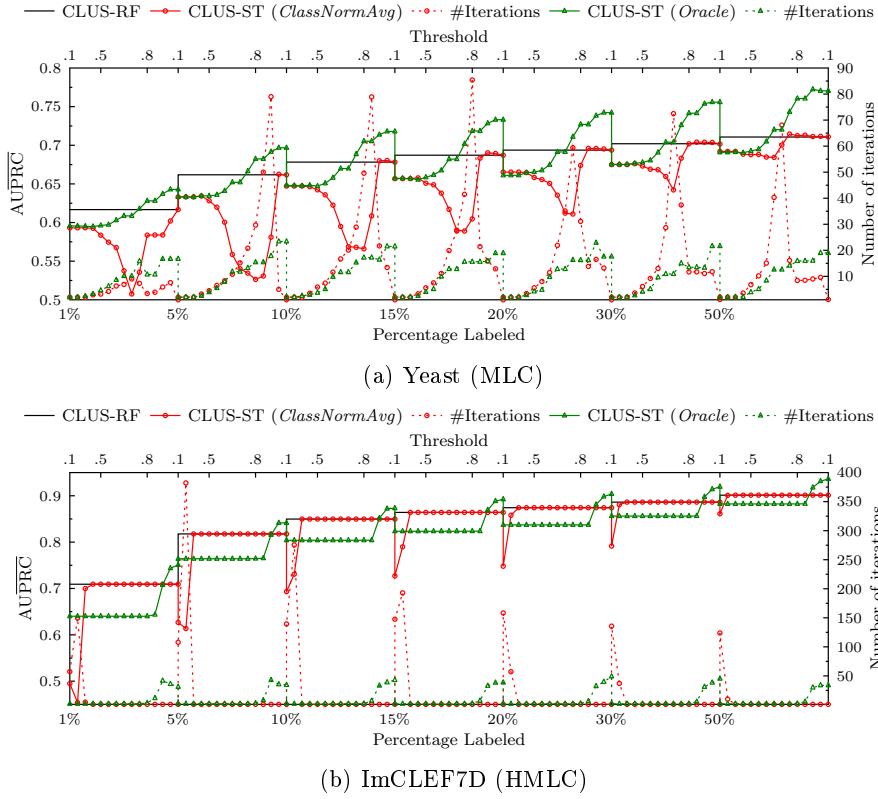


Figure 6.8: Comparison of predictive performance of supervised random forests (black line) and self-training with *ClassNormAvg* (red line) and *Oracle* (green line) reliability scores on the task of multi-label classification (a) and hierarchical multi-label classification (b). The percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99.

and Appendix A.2), and on some of the HMLC datasets (Figure 6.8b and Appendix A.3). This demonstrates that the self-training method has the potential to outperform supervised methods for MLC and HMLC tasks (given that a good reliability score is available), however, it seems that the reliability scores we use in this thesis for the MLC and HMLC tasks (Section 6.1.1.2) are not successful in discerning correct from erroneous predictions. Indeed, if we analyze the number of iterations performed by the self-training with *ClassNormAvg* reliability score, (Figures 6.8, Appendix A.2 and A.3), we can observe that they are closely correlated with the performance deterioration: The more iterations are performed, the worse performance becomes due to error propagation. On the other hand, the increase in number of iterations of self-training with the oracle score is accompanied by an increase in the predictive performance. Recall that reliability scores for MLC and HMLC tasks used in this thesis (Section 6.1.1.2) are based on empirical probabilities of classes. A similar definition of reliability score was successfully used in the context of self-training for (single-label) classification (Tanha et al., 2015), however it seems that in the case of structured outputs, a more sophisticated definition is needed.

It seems that, in the context of self-training, the task of SSL for MLC and HMLC is more difficult than SSL for MTR. We hypothesize that this is due to the following. At the MLC and HMLC tasks, the predicted classes can either be correct or wrong. On the other hand, at the MTR task, predictions can be 'slightly wrong' since the target variables are continuous. Thus, at the MTR task, even if the model makes small mistakes for some target

variables, the predictions can still be close to the 'real' values and the overall performance can improve. Contrary, at the MLC and HMLC tasks, every mistake the model makes will increase the probability that the same mistake is made again at the next iteration, which can lead to error propagation and loss of predictive performance.

Somewhat different findings to ours were reported in the study of Santos and Canuto (2014), who proposed a self-training based method for HMLC, and claimed its successful application. Unlabeled instances there are selected at random, i.e., no reliability scoring function was considered. The experimental setup of the study of Santos and Canuto (2014) is designed to show whether the semi-supervised version of the algorithm can achieve similar predictive performance as the base supervised algorithm, while using less labeled examples. More precisely, the semi-supervised method was provided with 75% of the labeled examples used by the supervised method, while the rest of 25% of examples were unlabeled. Santos and Canuto (2014) conclude that the self-training for HMLC *"did not deteriorate significantly the performance of the classifiers"*. From such experimental setup, however, it is not clear whether the semi-supervised method can outperform the supervised method with the *same* amount of labeled data. The main proposition of the semi-supervised learning is to improve the performance of supervised methods with the help of unlabeled data. We maintain that, in order to verify this proposition, the supervised and semi-supervised methods should be compared under the same conditions, i.e., with the same amount of labeled data. Otherwise, such implications cannot be made. The question is, why should one prefer semi-supervised methods, which are inherently more complex than supervised methods, if they yield the same performance? In such cases, we consider a simpler method (i.e., a supervised method) should be preferred.

6.4 Summary of the Results

In this section, we summarize the findings from the empirical evaluation of the proposed self-training method for structured output prediction. In the proposed approach, a model uses its own most reliable predictions in an iterative fashion. Therefore, a proper measure for the reliability of predictions is of crucial importance. We propose two reliability scoring functions for multi-target regression predictions, two aggregation schemes for merging per-target scores into a global score and two normalization techniques. The proposed reliability scoring functions are based on the mechanisms provided by ensemble learning: the variance of the votes of an ensemble and the estimate of errors of unlabeled examples by using out-of-bag labeled examples in their random forest proximities.

In our experiments we investigated whether the proposed self-training approach can improve the performance of the supervised random forest. Next, we analyzed the performance of different reliability scores and the influence of the τ parameter - a threshold on reliability scores. Finally, we proposed the algorithm for automatic threshold selection, and compared its performance with solutions proposed in the literature previously.

We performed the empirical evaluation over a wide range of datasets. In particular, we used 9 datasets for the task of multi-target regression, 8 datasets for the task of multi-label classification and 7 datasets for the task of hierarchical multi-label classification. We summarize the main findings as follows:

- The proposed self-training algorithm for structured output prediction showed different performance on different structured output prediction tasks: On datasets for multi-target regression, the method was able to outperform supervised random forests, while on datasets for (hierarchical) multi-label classification, the method was not successful.

- On datasets for multi-target regression, self-training with any of the eight proposed reliability measures was able to consistently improve over supervised random forests and outperform supervised multi-output support vector regression. In particular, among the two strategies for merging per-target scores, the averaging of the scores showed to be more favorable than the averaging of their ranks. The averaged variance-based reliability score (*VarianceNormAvg*), which is based on the variance of votes of an ensemble, was the best performing reliability score among the proposed reliability scores. Random selection during self-training is clearly inferior to the use of reliability scores: Better performance is achieved if unlabeled examples are selected on the basis of reliability scores, than if they are selected randomly.
- Analysis of the influence of the τ parameter on the predictive performance showed that the best threshold depends on the specific dataset and can vary depending on the amount of labeled data, this choosing the best threshold for the reliability of predictions is an important issue. To this end, we showed that automatic identification of the threshold is possible but, depending on the amount of labeled examples that is available, different algorithms should be used. The algorithm automatic threshold selection based on the exploitation of out-of-bag errors (*T_OOBInitial*) is recommended when more than 5% of labeled (as compared to unlabeled) data is available. For very small amounts of labeled data (<5%) estimates of out-of-bag errors are not reliable; therefore, a simple solution is more suitable: using the average score of the top 10% of the most reliable predictions. The use of the Airbag stopping criteria during the automatic threshold selection proved to be beneficial in all cases.
- Analysis of possible reasons of failure of the self-training method at the MLC and HMLC tasks showed that self-training has the potential to improve over supervised random forests for MLC and HMLC - given that a good reliability score is available. However, rather simplistic reliability scores for MLC and HMLC tasks used in this thesis are not able to discern correct from erroneous predictions, thus a more sophisticated definition is needed.

Chapter 7

Semi-Supervised Predictive Clustering Trees

In this chapter, we propose an algorithm for semi-supervised learning of PCTs (SSL-PCT). In contrast to the self-training approach (Chapter 6), the algorithm presented in this chapter exploits unlabeled examples directly (i.e., without predicting their labels) in the tree construction process in order to improve the quality of the splits. To find the best splits, (supervised) PCTs use a variance function that measure the cluster homogeneity. We extend supervised PCTs to a semi-supervised setting by defining new variance functions that consider both labeled examples and unlabeled examples. The proposed extension of PCTs preserves the appealing characteristics of decision trees, including model interpretability.

7.1 Methodology

In order to benefit from unlabeled data, semi-supervised methods have to make assumptions about the distribution of the unlabeled data with respect to the target variable(s). The idea of the popular semi-supervised smoothness assumption is to smooth the prediction function in highly populated regions. It states that *if two points X_i and X_j in a high density region are close, then also their outputs Y_i and Y_j should be close* (Chapelle et al., 2006). In this thesis, we propose semi-supervised PCTs motivated by this assumption. The fact that PCTs are situated at the intersection of supervised and unsupervised learning, makes them a very natural candidate for such an extension towards SSL. PCTs combine elements of predictive modelling (i.e., supervised learning) and elements of clustering (i.e., unsupervised learning). During the tree construction, clusters of examples similar to each other are identified. In addition, each cluster is associated with a cluster description and a predictive model. Cluster descriptions are used to assign new instances to clusters, while the associated predictive models predict the target value(s).

PCTs have a more flexible definition of descriptive, target, and clustering attributes than classical decision trees. Descriptive attributes are used to divide examples into groups, clustering attributes to evaluate the quality of the candidate splits, and target attributes to calculate the prediction in the leafs of the tree. In standard decision trees, there is no distinction between clustering and target attributes, i.e., the target attribute is the only clustering attribute used for the purpose of split selection. In other words, the similarity of examples is explicitly enforced only on the target attribute, and the desired splits are the ones which divide examples into groups of examples that have similar values for the target attribute. In PCTs, the clustering attributes can overlap with the target and/or the descriptive attributes. If the semi-supervised smoothness assumption holds, outputs of unlabeled examples should be coherent with the outputs associated to labeled examples

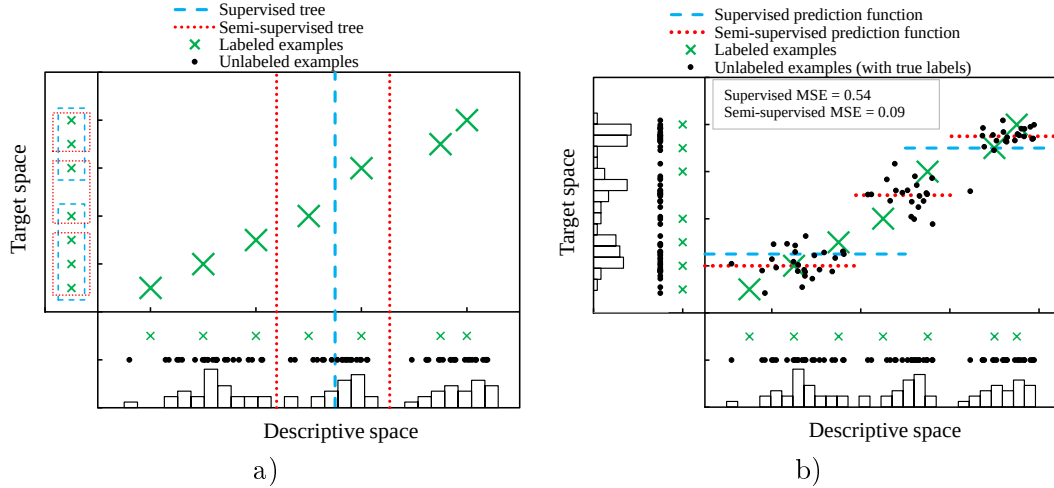


Figure 7.1: An illustration of the splits made by semi-supervised and supervised PCT for a (single-target) regression task. (a) A semi-supervised PCT detects 3 clusters of examples by the help of the distribution of unlabeled examples, instead of 2 clusters, as detected by the supervised PCT. (b) This leads to better predictive performance of the semi-supervised tree, since the labels of examples which are close to each other in dense regions are also close.

falling in the same leafs of the tree.

We exploit these properties of PCTs and propose to perform SSL by enforcing the similarity of examples in both the descriptive and the target space during PCT construction. This is achieved by using descriptive attributes in addition to target attributes as clustering attributes. Consequently, this allows us to exploit both labeled and unlabeled examples (for which only descriptive attributes are known) in tree construction. Such trees are able to produce clusters compact in both the descriptive and the target space.

Consider the illustrative example given in Figure 7.1. A supervised tree considers only the target space for the evaluation of splits, hence it identifies two clusters of examples (Figure 7.1a). However, the distribution of unlabeled examples in the descriptive space suggests that there are, in fact, three clusters. By considering both the descriptive and the target space in the evaluation of splits, the semi-supervised tree is able to identify them (Figure 7.1a). Consequently, the semi-supervised tree achieves better predictive performance than the supervised tree, due to the fact that the smoothness assumption holds, i.e., the labels of examples from high density regions are close to each other (Figure 7.1b).

Previously, Blockeel et al. (1998) proposed clustering trees and suggested that they may be useful when class information is missing. They construct the decision trees in an unsupervised manner, where distances among examples take into account all descriptive attributes. Class labels (i.e., target variables) are not used during the tree construction phase, but only to assign labels at leaf nodes once the tree has been constructed. This concept can be considered as a semi-supervised *cluster-then-label* approach (Dara et al., 2002; Demiriz et al., 1999; Andrew B Goldberg et al., 2009). The semi-supervised decision trees that we propose in this work are similar in nature to the ones proposed by Blockeel et al. (1998), but there are several key differences. To construct the trees, we consider simultaneously the descriptive and the target attributes. In other words, clustering and predictive modeling are performed simultaneously, which enables us to fully exploit both the labeled and unlabeled examples. Next, we introduce an additional parameter, by which

we control the amount of supervision in the decision trees. This allows us to build fully supervised trees, semi-supervised trees, or fully unsupervised trees, depending on the specific needs of the data mining problem at hand. Similar concept was considered by Ženko (2007) who constructed predictive clustering rules where the heuristic that guides the rule induction takes into account (with a trade-off parameter) both the descriptive and the target attributes. However, Ženko (2007) considered such rules only in a supervised learning context.

In the remainder of this section, we will present in more detail the above-described extension of PCTs for the semi-supervised task.

7.1.1 Semi-supervised learning of PCTs

The semi-supervised approach proposed in this study is based on the standard *top-down induction of decision trees* (TDIDT) algorithm (see Table 5.1), which takes as input a set of examples E and outputs a tree. The heuristic (h) that is used for selecting the tests (t) to put in internal tree nodes is reduction of variance caused by partitioning ($\mathcal{P}$, Table 5.1) the examples according to the tests.

The first extension to the classical algorithm for the induction of PCTs concerns the input: the dataset we consider (for example, see Table 5.1) takes as input the whole set of examples, that is, labeled and unlabeled examples. This means that $E = E_l \cup E_u$, where E_l is the part of the dataset with known labels and E_u is the part with unknown labels.

The second extension concerns, as mentioned before, the variance function that takes into account both the target and descriptive attributes in the identification of the best split. This is achieved by adapting the variance function of supervised PCTs. Recall that the variance function of supervised PCTs takes into account only the target attributes (Eq. 5.1, 5.3 and 5.5). The variance function of semi-supervised PCTs, on the other hand, is calculated as a weighted sum of variance over the target space Y and variance over the descriptive space X :

$$Var_f(E, Y, X) = w \cdot Var_f(E, Y) + (1 - w) \cdot Var_f(E, X), \quad (7.1)$$

where $w \in [0, 1]$ is the weight parameter that controls how much the target space and the descriptive space contribute to the variance function. Consequently, this controls the amount of supervision in the learning of semi-supervised PCTs. Values of the parameter w close to 1 emphasize more the target space, and consequently labeled examples affect the construction of the tree more than unlabeled examples (i.e., there is more supervision). On the other hand, values of w closer to 0 put more emphasis on the descriptive space, thus unlabeled examples affect the tree construction more than labeled examples and the tree learning algorithm receives less supervision. The w parameter enables the learning of semi-supervised PCTs to range from fully supervised trees (i.e., $w = 1$) to completely unsupervised trees (i.e., $w = 0$). The ability to control the influence of unlabeled examples with the w parameter is very important, since different datasets may require different amounts of supervision. This aspect is discussed in more detail in Section 7.3.3.

The variance of a set of examples E with *target* space Y is calculated differently depending on a type of structured output at hand:

$$Var_f(E, Y) = \begin{cases} \frac{1}{T} \cdot \sum_{i=1}^T Var(E, Y_i), & \text{if } Y \text{ consists of } T \text{ continuous variables} \\ \frac{1}{T} \cdot \sum_{i=1}^T Gini(E, Y_i), & \text{if } Y \text{ consists of } T \text{ discrete variables} \\ \frac{1}{\sum_{l=1}^{|C|} w(c_l)} \left(\frac{1}{|E|} \cdot \sum_{e_i \in E_l} d(L_i, \bar{L})^2 \right), & \text{if } Y \text{ is a hierarchy of classes} \end{cases} \quad (7.2)$$

Note that the three ways for calculation of variance defined in Eq. 7.2, correspond to variance functions defined for PCTs for MTR (Eq. 5.1), MLC (Eq. 5.3) and HMLC (Eq. 5.5), respectively. However, contrary to supervised PCTs (Ch. 5), here E may contain both labeled E_l and unlabeled E_u examples, thus we need to re-define the equations for the computation of the variance of individual attributes in order to take into account the missing values. Let N be the number of examples (including both labeled and unlabeled), and let K_i be the number of examples with known (non-missing) values of the i^{th} attribute Y_i ¹. We now re-define the calculation of variance for the continuous target variables and gini score for the discrete target variables:

$$Var(E, Y_i) = \frac{\frac{N-1}{K_i-1} \cdot \sum_{j=1}^{K_i} (y_{i,j})^2 - N \cdot \left(\frac{1}{K_i} \cdot \sum_{j=1}^{K_i} y_{i,j} \right)^2}{N}, \quad (7.3)$$

$$Gini(E, Y_i) = 1 - \sum_{j=1}^{C_i} \left(\frac{|\{e : e \in E \text{ such that } e \text{ has class } c_j\}|}{K_i} \right)^2 = 1 - \sum_{j=1}^{C_i} \hat{p}_j, \quad (7.4)$$

where C_i is the number of classes of the Y_i , and $\hat{p}_j$ is the empirical probability of a class c_j considering only examples with a known value for variable Y_i . Note that for the HMLC task the variance for the output space is calculated only on the labelled data (see Eq. 7.2).

The descriptive variables can either be numeric or nominal, thus the variance of a set of examples E with *descriptive* space X consisting of D descriptive attributes is calculated as follows:

$$Var_f(E, X) = \frac{1}{D} \cdot \left(\sum_{X_i \text{ is numeric}} Var(E, X_i) + \sum_{X_j \text{ is nominal}} Gini(E, X_j) \right), \quad (7.5)$$

where the variance or the gini score of individual descriptive attributes is calculated analogous to the one of target attributes, as given in Eq. 7.3 and 7.4.

Similarly as in the supervised PCTs, the variances or gini scores of individual attributes the targets are normalized, so that each attribute contributes equally to the overall variance. The normalization is performed by dividing the estimates of individual attributes in Eq. 7.3 and 7.4 with the variance or gini score of the attribute on the entire available training set.

Note that the proposed algorithm is not limited to the Gini index or variance as impurity measures. In principle, other impurity measures could be used. For example, the PCT framework also implements the sum of the entropies of class variables, reduced error, gain ratio and m-estimate. We chose to use the Gini index because it is one of the most popular measures for impurity of nominal variables. The other obvious choice would be Information gain, however, Gini index and Information gain perform very similarly and mostly it is not possible to decide which of the two measures to prefer (Raileanu & Stoffel, 2004). Similarly, for measuring impurity of numeric variables, we choose variance, because variance is *de facto* standard used for regression trees.

When building semi-supervised trees, two extreme cases may occur: (1) A leaf of the tree may contain only unlabeled examples, and (2) The calculation of variance may be necessary for attributes where all the examples have missing values or only one example has a known (non-missing) value (e.g., $K_i \leq 1$ in Eq. 7.3). The first extreme opens the question: *How to calculate the prototype function for such a leaf?* In fact, the prototype

¹The number of examples with non-missing values is usually the same across all target attributes, however these numbers can differ if partially labeled examples are present in the dataset, such as in the case study presented in Section 9.2.

function of SSL-PCTs is calculated in the same way as in supervised PCTs, but by using only the *labeled* training examples that belong to the given node. We handle the extreme case of no labeled examples in a leaf by returning the prototype of the first parent of such a leaf that contains labeled examples. Nodes of the tree that contain only unlabeled examples are not divided anymore, while in leaf nodes that contain labeled examples we allow no less than 2 of those. The same criterion is used in supervised PCTs, i.e., a minimum number of (labeled) examples in a leaf node is set to 2.

The second extreme occurs when a candidate split needs to be evaluated, such that all the examples in one of the resulting groups have missing values for one of the attributes. For example, this occurs when the split is performed such that only unlabeled examples travel to one of the branches of the decision tree. We handle this extreme by estimating the variance with the variance of the parent node.

Once we have presented the induction of SSL-PCTs, we can now extend the semi-supervised learning solution to learn random forests of SSL-PCTs. By using SSL-PCTs, it is possible to build semi-supervised random forests by simply using trees learned with such an algorithm to construct the members of the ensemble (as described in Section 5.2), instead of using trees learned with a supervised algorithm. The only difference is that the bootstrap samples are obtained from the whole set of examples E , which includes both labeled and unlabeled examples. In semi-supervised ensembles, we modify the bootstrap sampling procedure to the stratified bootstrap sampling considering the labeled and unlabeled examples. This is to avoid having bootstrap samples made out of only unlabeled examples.

7.1.1.1 Feature weighted semi-supervised PCTs

Unlike some machine learning methods, such as k NN, PCTs (and, more generally, decision trees) are robust to irrelevant features. Only the most informative features are used as tests when building (supervised) trees; therefore, irrelevant features are likely to be ignored. However, in SSL-PCTs the robustness to irrelevant features may be compromised, since both target and descriptive features contribute to the evaluation of tests. Moreover, as anticipated before, different features can have a different impact on the predictive capabilities of the learned models. To deal with these issues, we propose to use feature weighting to enhance the robustness to irrelevant/less relevant features in SSL-PCTs.

In the literature, methods for feature ranking are used to select the most important features, and to discard the irrelevant ones. An importance score, which corresponds to the informativeness of a feature, is assigned to each descriptive feature. More informative features are associated with a higher score, while the less informative ones are associated with a lower score. Cunningham and Delany (2007) showed that weighting features with importance scores helps the k -nearest neighbors method in dealing with irrelevant features. In a similar manner, we use importance scores as feature weights when building SSL-PCTs.

More specifically, for each descriptive attribute X_i , we determine its importance score $\hat{\sigma}_i$ by using a feature ranking based on a random forest of PCTs (Kocev, Slavkov, & Džeroski, 2013). The feature ranking of the descriptive variables is obtained by exploiting the intrinsic mechanism of random forests: it uses the internal out-of-bag (OOB) estimates of the error and noising of the descriptive variables to estimate their importance. To create each tree from the forest, the algorithm first creates a bootstrap replicate. The samples that are not selected for the bootstrap are called OOB samples, and are used to evaluate the performance of each tree from the forest. The rationale behind this method is that if a descriptive variable is important for the output variables, then adding noise to its OOB values will yield an increase of the error of the base predictive model.

This method is executed on the available labeled E_l portion of the data prior to building

SSL-PCTs. The obtained importance scores are then normalized according to the following formula: $\sigma_i = \hat{\sigma}_i / \max(\hat{\sigma}_1, \hat{\sigma}_2, \dots, \hat{\sigma}_D)$. The variance of each descriptive attribute X_i is then multiplied by its normalized importance score σ_i in the variance function of descriptive space of SSL-PCTs:

$$Var_f(E, X) = \frac{1}{D} \cdot \left(\sum_{X_i \text{ is numeric}} \sigma_i \cdot Var(E, X_i) + \sum_{X_j \text{ is nominal}} \sigma_j \cdot Gini(E, X_j) \right), \quad (7.6)$$

In this way, irrelevant features (i.e., features with low importance score) contribute less to the calculation of variance. We denote semi-supervised PCTs where feature weighting is used with SSL-PCT-FR.

7.1.1.2 Computational complexity analysis

The procedures that contribute to the computational complexity of *supervised* PCTs are as follows: sorting the values of D descriptive attributes ($O(DN \log N)$), calculating the best split for T target variables ($O(TDN)$), and applying the split to the N (labeled) training examples ($O(N)$). Assuming that the depth of the tree is in the order of $O(\log N)$ (Witten & Frank, 2005), the total computational complexity of constructing a single PCT is $O(DN \log^2 N) + O(TDN \log N) + O(N \log N)$. What changes from supervised PCTs to SSL-PCTs is, first the value of N : In the case of semi-supervised PCTs, the number of training examples is equal to the number of labeled and unlabeled examples combined, i.e., $N = N_l + N_u$, instead of $N = N_l$. Second, SSL-PCTs consider also descriptive attributes when the split for T target variables is calculated, thus the complexity of this step is $O((T + D)DN)$. The total computational complexity of calculating a single SSL-PCT is $O(DN \log^2 N) + O((T + D)DN \log N) + O(N \log N)$.

7.2 Experimental Design

In this section, we first describe the datasets used in the experimental evaluation. Next, we state the main experimental questions of the study, we present the evaluation procedure, the specific parameter settings of the algorithms and the performance metrics.

7.2.1 Data description

7.2.1.1 Multi-target regression datasets

We use 12 datasets with multiple continuous target variables to evaluate the predictive performance of the proposed SSL-PCTs. The datasets come from several domains. The characteristics of the datasets are given in Table 7.1. We can observe that the datasets vary in their size, number of attributes and number of target variables.

The majority of the data are from the area of environmental sciences. To begin with, the task in the *RFI* dataset (Spyromitros-Xioufis et al., 2014) is to predict river network flows 48 hours ahead. Next, the Soil quality (D. Demšar et al., 2006) and Water quality (Blockeel et al., 1999) datasets concern habitat modeling of soil and water organisms, respectively. Finally, Vegetation condition (Kocev et al., 2009) and Vegetation clustering (Gjorgjioski & Džeroski, 2003) datasets are concerned with predicting several indicators of condition of indigenous vegetation in Victoria, Australia.

From the domain of economy, we consider the SCM1D and SCM20D datasets (Spyromitros-Xioufis et al., 2014), where the task is to predict the price of 16 products for the next day and their mean price over the next 20 days, respectively.

Table 7.1: Characteristics of the datasets. N : number of instances, D/C : the number of descriptive attributes (discrete/continuous), T : number of target variables.

Dataset	Domain	N	D/C	T
Bicycle	Human behavior	17379	0/12	3
Enb	Energy efficiency	768	0/8	2
Eunite	Electricity load forecasting	8064	0/34	5
RF1	Ecology	9125	0/64	8
SCM1D	Economy	9803	0/280	16
SCM20D	Economy	9803	0/61	16
Scpf	Human behavior	1137	19/4	3
Soil quality	Ecology	1944	0/142	3
Solar flare 2	Astronomy	1066	10/0	3
Vegetation clustering	Ecology	27522	0/66	10
Vegetation condition	Ecology	16967	1/39	7
Water quality	Ecology	1060	0/16	14

The Bicycle (Fanaee-T & Gama, 2013) and Scpf (Spyromitros-Xioufis, Tsoumakas, Groves, & Vlahavas, 2016) datasets are concerned with various aspects of human behavior. The task at the former dataset is to predict the number of daily uses of rental bicycle sharing system coming from casual users or registered users, and the total number of daily rents. The task at the Scpf dataset is to predict the number of views, clicks and comments that a specific issue will receive. The data comes from the issues collected in Oakland, Richmond, New Haven, Chicago.

The task at the Eunite dataset (B.-J. Chen, Chang, & Lin, 2004) is forecasting of the maximal daily electrical load, while the Enb dataset (Tsanas & Xifara, 2012; Spyromitros-Xioufis et al., 2014) is concerned with predicting the energy efficiency of buildings, i.e., heating and cooling load requirements. Finally, the Solar flare 2 dataset is concerned with prediction of the occurrence of 3 types of solar flares, on the basis of observed characteristics of the Sun.

7.2.1.2 Multi-label classification datasets

To evaluate the predictive performance of the proposed self-training methods in the context of MLC, we use 12 datasets, which come from seven domains: audio, image, text and music classification, and biology, ecology and media. The main statistics of the datasets are given in Table 7.2. We can observe that the datasets vary in their size, number of attributes, number of labels and label cardinality.

Most of the datasets are described in Section 6.2.1.2, except Genbase, Mediana, and SIGMEA real datasets. Here, we describe these three datasets in more detail.

The Mediana dataset (Skrjanc, Grobelnik, & Zupanec, 2001) is concerned with predicting readers of major Slovenian newspapers. The data was collected by the Mediana research institute through questionnaires where examinees had to answer various questions related to the media, personal activities, interests, life style, income, demography etc.

The task at the Genbase dataset (Diplaris, Tsoumakas, Mitkas, & Vlahavas, 2005) is motif-based protein classification: On the basis of motifs, the proteins are classified to several important protein families. We removed several attributes from this dataset, which contained only constant values.

The SIGMEA real dataset (D. Demšar et al., 2005) is a discretized version of the dataset about pollen dispersal on fields with genetically modified oilseed rape, which was

Table 7.2: Characteristics of the MLC datasets: N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous), $\mathcal{L}$ is the number of labels, $\overline{\mathcal{L}_L}$ is the average number of labels per example.

Dataset	Domain	N	D/C	$\mathcal{L}$	$\overline{\mathcal{L}_L}$
Bibtex	Text	7395	1836/0	159	2.402
Birds	Audio	645	2/258	19	1.014
Emotions	Music	594	0/72	6	1.869
Corel5k	Images	5000	499/0	374	3.522
Enron	Text	1702	1001/0	53	3.378
Genbase	Text	662	1186/0	27	1.252
Mediana	Media	7953	21/58	5	1.205
Medical	Text	978	1449/0	45	1.245
Scene	Images	2407	0/294	6	1.074
SIGMEA real	Ecology	817	0/4	2	0.726
Slovenian rivers	Ecology	1060	0/16	14	5.073
Yeast	Biology	2417	0/103	14	4.237

Table 7.3: Characteristics of the HMLC datasets: N is the number of instances, D/C is the number of descriptive attributes (discrete/continuous), $\mathcal{H}$ is a type of the hierarchy, $|\mathcal{H}|$ is the number of nodes in the hierarchy, $\mathcal{H}_d$ is the maximal depth of the hierarchy, $\overline{\mathcal{L}_L}$ is the average number of labels per example.

Dataset	Domain	N	D/C	$\mathcal{H}$	$ \mathcal{H} $	$\mathcal{H}_d$	$\overline{\mathcal{L}_L}$
Danish farms	Ecology	1944	132/5	Tree	72	3	7
Diatoms	Images	3119	0/371	Tree	377	3	0.94
Slovenian rivers	Ecology	1060	0/16	Tree	724	4	25
ImCLEF07A	Images	11006	0/80	Tree	96	3	1
ImCLEF07D	Images	11006	0/80	Tree	46	3	1
Cellcycle-GO	Functional genomics	3766	0/77	DAG	4126	12	35.91
Church-GO	Functional genomics	3764	1/26	DAG	4126	12	35.89
Derisi-GO	Functional genomics	3733	0/63	DAG	4120	12	35.99
Eisen-GO	Functional genomics	2425	0/79	DAG	3574	11.12	39.04
Expr-GO	Functional genomics	3788	4/547	DAG	4132	12	35.87
Pheno-GO	Functional genomics	1592	69/0	DAG	3128	12	36.43

introduced in 6.2.1.1.

7.2.1.3 Hierarchical multi-label classification datasets

To evaluate the predictive performance of the proposed self-training methods in the context of HMLC, we use 12 datasets, which come from four domains: ecology, image classification, biology, text classification and functional genomics. The main statistics of the datasets are given in Table 7.3. We can observe that the datasets vary in their size, number of attributes, number of labels and label cardinality.

The description of all datasets but Diatoms and datasets from the functional genomics domain can be found in Section 6.2.1.3. The Diatoms dataset (Dimitrovski, Koccev,

Loskovska, & Džeroski, 2012) consists of microscopic images of diatom algae. Each image is annotated with the diatom taxa that is present on the image. The hierarchical relationship among labels are taxonomic ranks of diatoms.

The datasets from the function genomics domain (Vens et al., 2008) are concerned with prediction of gene functions of the *Saccharomyces cerevisiae* (i.e., baker’s or brewer’s yeast), which is a classic model organism in biology. The different datasets include different types of bioinformatic data, such as phenotype, homology, and gene expression. Different sources of data highlight different aspects of gene function. The label hierarchy, i.e., gene functions correspond to the Gene Ontology.

7.2.2 Experimental setup and evaluation procedure

We have proposed semi-supervised PCTs (SSL-PCT) and feature weighted semi-supervised PCTs (SSL-PCT-FR). As a baseline for comparison with these methods across different structured output prediction tasks, we use the standard PCT algorithms for MTR, MLC and HMLC (see Chapter 5), denoted as SL-PCT. This is the most reasonable baseline, since the goal is to precisely measure the contribution of the unlabeled data to the overall performance under the same conditions. By comparing to SL-PCTs, we answer the main question of this study: *Are SSL-PCTs able to improve over standard supervised PCTs?*

We also explore the predictive performance of semi-supervised random forests, denoted as SSL-RF and SSL-RF-FR, where SSL-PCTs and SSL-PCT-FRs are used as base models, respectively. We compare the SSL-RF method with baseline supervised random forests for structured output prediction (i.e., CLUS-RF) where the SL-PCT algorithm is used as a base model. We construct random forests consisting of 100 trees. The trees are not pruned and the number of random features at each internal node is set to $\lfloor \log_2(D) + 1 \rfloor$, where D is the total number of features (Breiman, 2001).

In all of the single tree experiments, the trees are pruned. We use the pruning procedure as used in the M5 regression trees (Quinlan, 1993). In particular, the pruning procedure compares the error estimates obtained by pruning a node or not. The error estimates are based on the training cases and corrected in order to take into account the complexity of the model in the node.

We use various amounts of labeled data for both the supervised and semi-supervised methods to explore the influence of the amount of labeled data on the predictive performance of the methods. We perform experiments where we vary the absolute number of labeled examples in the following set: $\{50, 100, 200, 350, 500\}$.

While the experimental setup that varies the relative amount of labeled data is more frequently used in the studies of SSL, we consider the experimental setup which varies the absolute numbers of labeled examples to be highly relevant as well. By considering this setup, it is easier to identify the applicability range of a given SSL method with respect to the amount of labeled data. Moreover, this approach is closer to a real-world setup: typically, the number of labeled examples is known beforehand, while the number of unlabeled examples can vary. The main premise of SSL is that a limited amount of labeled data is available, thus it is expected that the success of semi-supervised methods will depend on the amount of labeled data. By varying the amount of labeled data, it is possible to identify for which amounts SSL methods outperform the supervised ones. When the relative amounts of labeled data are considered, the applicability range becomes a ‘moving target’, since, for example, 10% of labeled data can correspond to a few tens of examples in one dataset, and a few thousands examples in another dataset.

The labeled data used for training the predictive models (both supervised and semi-supervised) are randomly selected from the available training data, while the remaining examples serve both as unlabeled data and as a test set. Namely, we temporarily remove

their labels and provide the examples to the algorithm to serve as unlabeled data during training. When the models are built, the performance is evaluated on the test set composed of the unlabeled examples with their true labels restored. In other words, the evaluation of the semi-supervised methods is performed in the context of transductive learning. For a fair comparison, the supervised methods are trained only on the labeled part of the data and evaluated on the same test set. To get more stable results, the procedure is repeated 10 times with different random initializations. The predictive performance reported in the results is the average of the performance values obtained from the 10 runs.

Note that, even though we used the transductive evaluation scenario, the semi-supervised PCTs are inductive, i.e., they can make predictions on examples unseen during learning without the need to retrain the model.

For each run, we optimize the weight parameter w by performing 3-fold cross-validation on the available labeled part (i.e., the labeled part of the training set). During the cross-validation procedure, semi-supervised methods are supplied with available unlabeled examples (without their respective true labels). We consider values of the parameter w from 0 to 1 with a step of 0.1.

On datasets with continuous target space, namely MTR task, we assess the predictive performance of the algorithms by using the average relative root-mean-square-error (RRMSE), defined as follows:

$$RRMSE = \frac{1}{T} \sum_{i=1}^T \sqrt{\frac{\sum_{j=1}^{N_{test}} (y_{j,i} - \hat{y}_{j,i})^2}{\sum_{j=1}^{N_{test}} (y_{j,i} - \bar{y}_i)^2}},$$

where T is the number of target variables, $y_{j,i}$ is the value of the i^{th} target variable for the j^{th} example of the test set, $\hat{y}_{j,i}$ is prediction of the j^{th} example, N_{test} is the number of examples in the test set, and $\bar{y}_i$ is the mean value (on the training set) of the i^{th} target variable.

For datasets with discrete outputs, namely for MLC and HMLC tasks, we evaluate the algorithms by using the area under the Precision-Recall curve (AUPRC), and in particular, the area under the average Precision-Recall curve (AUPRC) as suggested by Vens et al. (2008). The precision and recall values are micro-averaged as follows:

$$\overline{Prec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i}, \quad \text{and} \quad \overline{Rec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i},$$

where i ranges over all classes.

To investigate whether the observed differences in performance among the methods are statistically significant, we follow the recommendations outlined by J. Demšar (2006). Considering our main experimental question, we use the non-parametric Wilcoxon paired signed rank test (Wilcoxon, 1945). More specifically, we compare the predictive performance of two methods over multiple datasets. In all the experiments reported in this study, the significance level is set to 0.05.

7.3 Results and Discussion

In this section, we present the results obtained from the empirical evaluation of the proposed semi-supervised predictive clustering trees (and ensembles thereof) on different structured output predictions tasks: multi-target regression, multi-label classification and hierarchical multi-label classification. We first present the results of the empirical evaluation of semi-supervised and supervised PCTs (and ensembles thereof) for each structured output

prediction task. Additionally, we then perform statistical analysis to investigate whether the observed differences in predictive performance between the methods are statistically significant. Next, we investigate the influence of the weight parameter w , which controls the amount of supervision in the models. Furthermore, we discuss the size and interpretability of the models. Finally, we analyze the influence of the unlabeled data on the performance of SSL-PCTs.

7.3.1 Analysis of predictive performance

In this section, we present the predictive performance of the semi-supervised and supervised PCTs (and ensembles thereof) on the MTR, MLC and HMLC tasks, with different numbers of labeled examples ranging from 50 to 500.

7.3.1.1 Multi-target regression

Figure 7.2 presents the predictive performance (RRMSE) of semi-supervised (SSL-PCT, SSL-PCT-FR, SSL-RF and SSL-PCT-FR) and supervised methods (SL-PCT and CLUS-RF) on the 12 multi-target regression datasets, with an increasing amount of labeled data.

First of all, we can observe improvement of the semi-supervised predictive clustering trees (SSL-PCT) over supervised ones on several datasets, namely Bicycle, RF1, Soil quality, Solar flare 2 and Scpf. On these datasets, SSL-PCT is superior in performance to SL-PCTs for all amounts of labeled data (except for 50 labeled examples at Soil quality and Bicycle datasets). Improvement of SSL-PCT over SL-PCT is also observed on SCM20D and Water quality datasets, but only for some amounts of labeled data. On the remaining datasets, the performance of SSL-PCT and SL-PCT is the same or very similar. The different performance of SSL methods on different datasets is somewhat expected, since success of the semi-supervised methods is known to be domain-dependent (Chawla & Karakoulas, 2005).

The feature weighted semi-supervised method (SSL-PCT-FR) and non feature weighted one (SSL-PCT) mainly perform similar, with some benefits of features weighing observed on RF1 and Soil quality datasets. However, features weighing can also cause loss of predictive performance, e.g., on SCM1D dataset for 50 and 100 of labeled examples. Therefore, the observed empirical evidence does not support the use of feature weighting by default when building SSL-PCTs for MTR.

Contrary to semi-supervised single trees, semi-supervised random forests mainly do not improve their supervised counterpart (SSL-RF). Some improvement is observed on Eunite and Solar flare 2 datasets. On one dataset, namely Water quality, degradation of performance is observed for 50 to 200 of labeled examples. The improvement of SSL-PCTs over SL-PCTs does not guarantee the improvement of SSL-RF over CLUS-RF (e.g., RF1, Soil quality, and Scpf), and vice versa, SSL-RF can improve over CLUS-RF even if SSL-PCTs does not improve over SL-PCTs (e.g., Eunite dataset). All in all, the results suggest that random forests for MTR mainly do not benefit from addition of the unlabeled data. In other words, outperforming a supervised random forest is a harder task than outperforming a supervised tree.

On most of the datasets, the learning curves are in accordance with the hypothesized shape, i.e., the predictive performance improves with the increase of the number of labeled data. However, there are some exceptions. Namely, for Soil quality, Solar flare 2 and Scpf datasets, the predictive error sometimes increases as more labeled data are added. While this deserves further investigation, at present we hypothesize that this could happen as the additional labeled examples allow bigger trees to grow, but are insufficient to cover all the created clusters, i.e., to yield good estimates for the target variable values in the leaves.

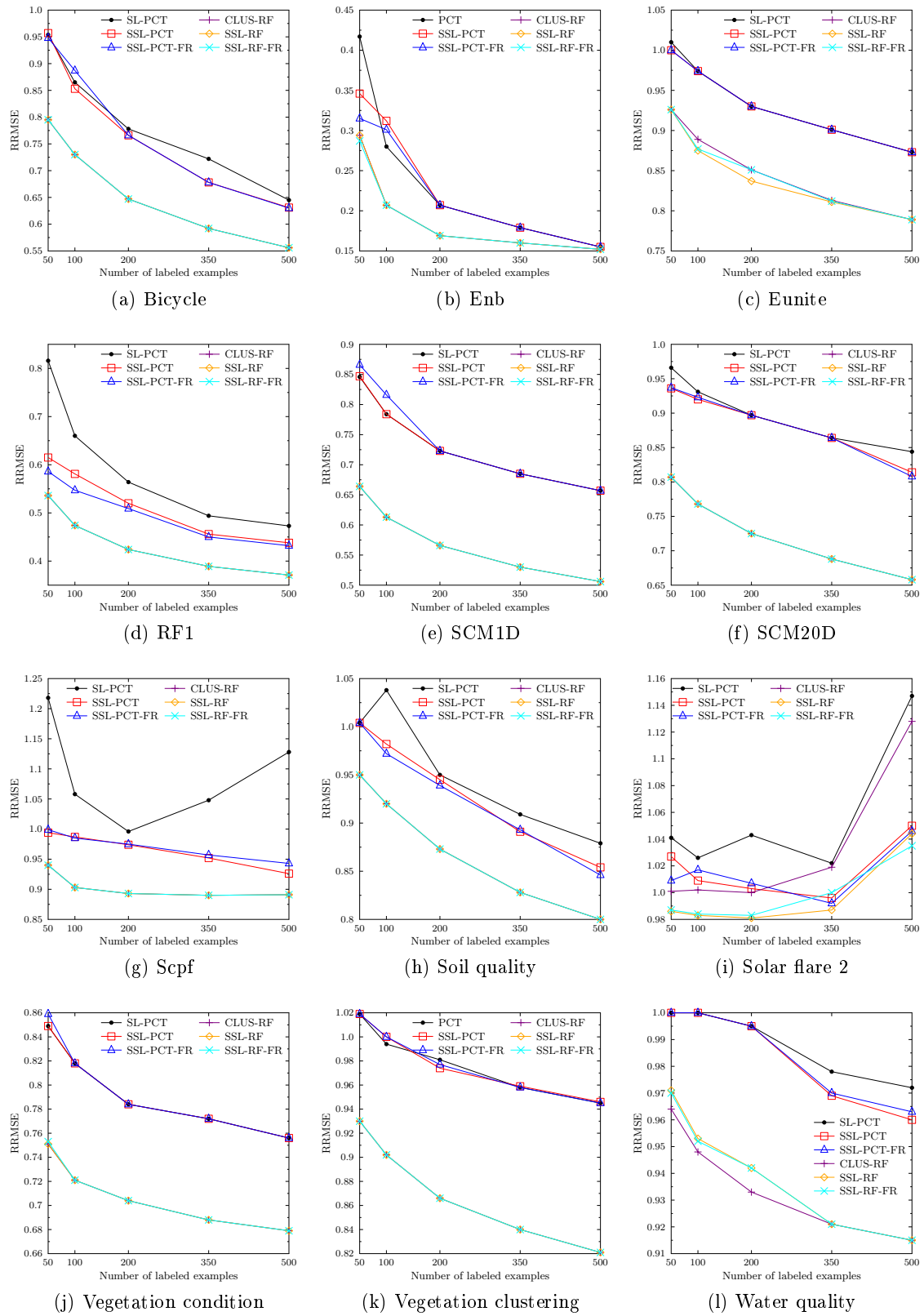


Figure 7.2: Predictive performance of the supervised and semi-supervised methods on the multi-target regression datasets.

7.3.1.2 Multi-label classification

Figure 7.3 presents the predictive performance ($AUPRC$) of semi-supervised (SSL-PCT, SSL-PTC-FR, SSL-RF and SSL-RF-FR) and supervised methods (SL-PCT and CLUS-RF) on the 12 MLC datasets, with an increasing amount of labeled data.

We can clearly observe that semi-supervised PCTs are superior to SL-PCTs on most of the datasets. Namely, on 8 out of 12 datasets, either SSL-PCTs or SSL-PTC-FRs (or both) dominate the performance of SL-PCTs by a good margin. On the other four datasets, namely Corel5k, Emotions, Mediana and SIGMEA real, the performance of supervised and semi-supervised PCTs is mostly the same, or similar to the performance of SL-PCTs.

Intuitively, the improvement of semi-supervised over supervised methods should decrease as the number of labeled examples increases, and eventually semi-supervised and supervised methods are expected to converge to the same/similar performance. On 8 datasets where SSL-PCTs improve over SL-PCTs, the "convergence point" at 500 of labeled examples is reached only on Genbase database. With about 650 of examples, this dataset is rather small. On the other 7 datasets, semi-supervised PCTs improve over SL-PCTs for nearly amounts of labeled data considered (i.e., from 50 to 500), and it seems that the trend may continue even for more than 500 of labeled examples. However, this needs to be further investigated.

The feature weighted semi-supervised method (SSL-PCT-FR) and non feature weighted one (SSL-PCT) have similar trends in predictive performance, however, on some datasets there are notable differences. Namely, on Birds and Scene datasets feature weighting is beneficial for predictive performance of SSL-PCTs, and even necessary for improvement over SL-PCTs on Birds dataset with ≥ 350 of labeled examples. On the other hand, feature weighting clearly hurts the predictive performance of the SSL-PCT method on the Bibtex dataset. Thus, similarly as observed for the MTR task (Section 7.3.1.1), feature weighting may help in certain cases, however the empirical results cannot support the use of it by default when building SSL-PCTs for MLC.

We next compare the performance of semi-supervised random forests (SSL-RF) and supervised random forests (CLUS-RF). Contrary to the MTR task (Section 7.3.1.1), here CLUS-RF improve over CLUS-RF on several datasets: Bibtex, Corel5k, Genbase, Medical, SIGMEA real, and somewhat on Emotions and Enron datasets. However, as compared to single trees, the improvements of SSL-RF over CLUS-RF are observed on fewer datasets, and are smaller. The observations made at MTR task (Section 7.3.1.1) is also made for the MLC task: the improvement of SSL-PCTs over SL-PCTs does not guarantee the improvement of SSL-RF over CLUS-RF (e.g., Mediana, and Yeast datasets), and vice versa, SSL-RF can improve over CLUS-RF even if SSL-PCTs does not improve over SL-PCTs (e.g., SSL-RF-FR on Emotions dataset for 200 and 350 labeled examples). Similarly as observed for the single trees, there is no apparent advantage of feature weighting when semi-supervised random forests are built, even though it is somewhat helpful on Emotions and Enron datasets.

7.3.1.3 Hierarchical multi-label classification

Figure 7.4 presents the predictive performance ($AUPRC$) of semi-supervised (SSL-PCT, SSL-PTC-FR, SSL-RF and SSL-RF-FR) and supervised methods (SL-PCT and CLUS-RF) on the 12 hierarchical multi-label classification datasets, with an increasing amount of labeled data.

We can observe different behavior on 6 functional genomics datasets and 6 datasets from other domains. Namely, on all 6 datasets of the latter group, semi-supervised PCTs

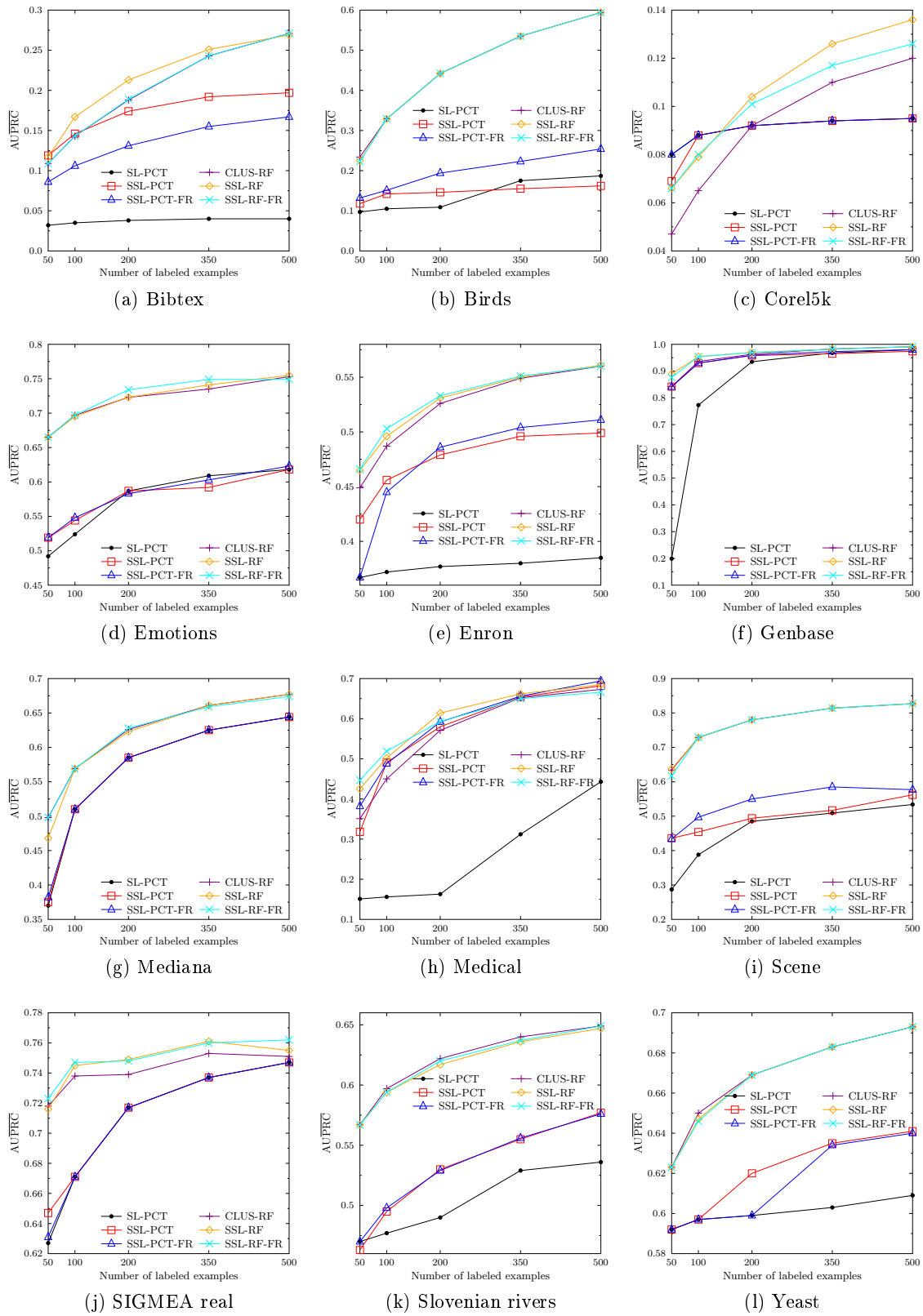


Figure 7.3: Predictive performance of the supervised and semi-supervised methods on the multi-label classification datasets.

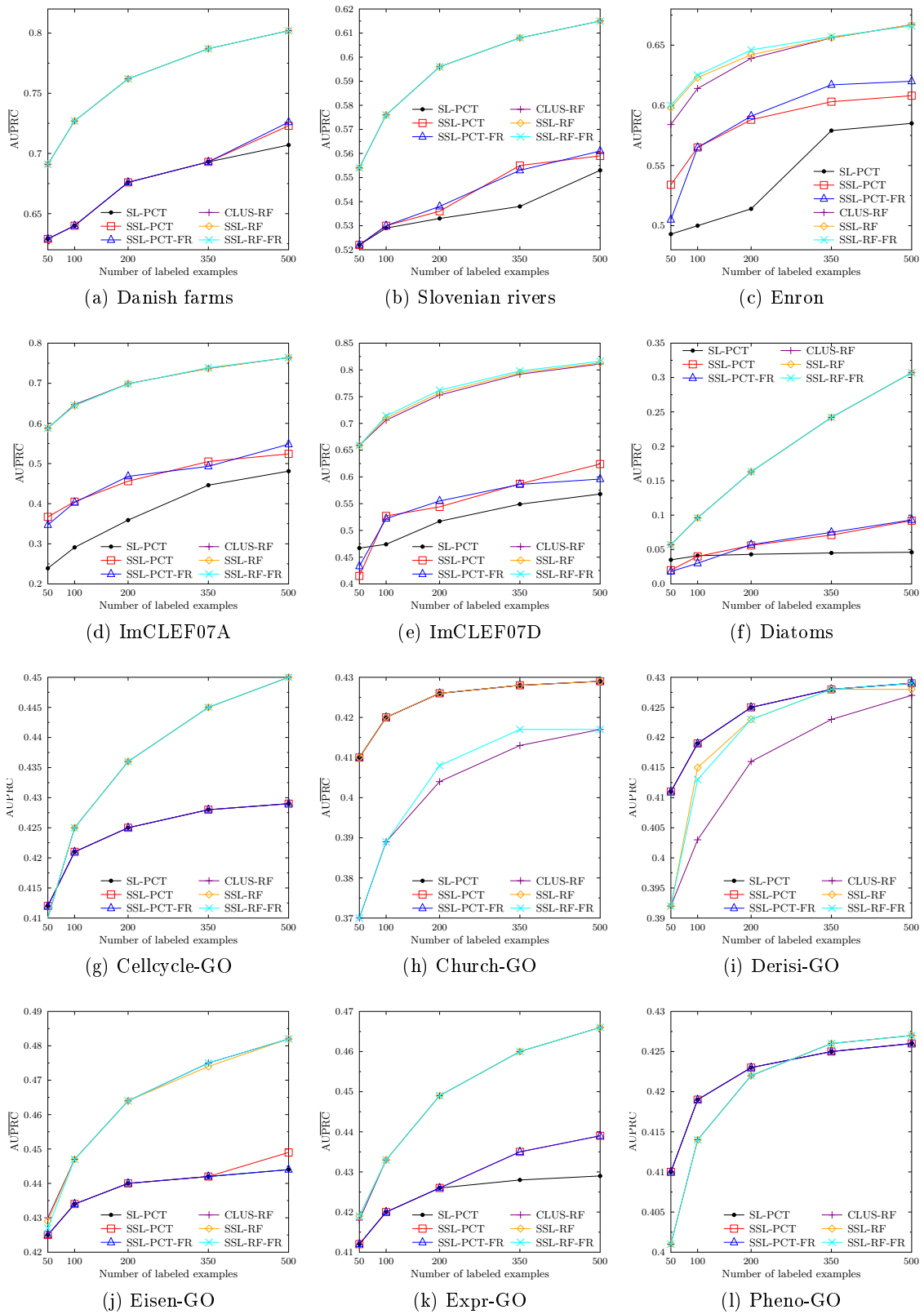


Figure 7.4: Predictive performance of the supervised and semi-supervised methods on the hierarchical multi-label classification datasets.

are improving over SL-PCTs – albeit not necessarily always for all amounts of labeled data. This is the case on Enron and ImCLEF07A datasets, where both SSL-PCTs and SSL-PCT-FR dominate the performance of SL-PCTs, while it seems that on other dataset at least 100 (Slovenian rivers and ImCLEF07D) or 500 (Danish farms) of labeled examples are needed to improve over SL-PCTs. Similarly as observed as the MLC tasks, it seems that the trend of improvement of semi-supervised PCTs over SL-PCTs may continue even when the number of labeled examples exceeds 500.

On the other hand, semi-supervised PCTs are not so successful on functional genomics datasets. Analysis of the tree sizes (see Section 7.3.4) reveals a explanation for such results. Namely, on all the 6 functional genomics datasets, and almost all different amounts of labeled data, both supervised and semi-supervised trees are composed of only one node. Note that, these datasets have very large label hierarchies which are very sparsely populated. It seems that for such datasets the amount of labeled data we considered (i.e., up to 500 of labeled examples) is not sufficient to build trees – either supervised or semi-supervised. In fact, semi-supervised trees have more than one node on Expr-GO (≥ 350 of labeled examples) and Eisen (for 500 labeled examples) datasets, and those are exactly the occasions where they improve over supervised trees. We thus hypothesize that for larger amounts of labeled data, SSL-PCTs could outperform supervised PCT also on functional genomics datasets.

On the HMLC task, the feature weighted semi-supervised method (SSL-PCT-FR) and non feature weighted one (SSL-PCT) mostly have very similar performance, i.e., there is no apparent benefit of feature weighting.

The semi-supervised random forests (SSL-RF and SSL-RF-FR) improve over supervised random forests (CLUS-RF) only on few of the datasets. Namely, on the initial part of the learning curve of the Enron dataset, and on Church-GO and Derisi-GO datasets. On other datasets it seems that unlabeled data is not beneficial for the performance of random forests of PCTs for HMLC.

7.3.2 Statistical analysis of the predictive performances

The results of the statistical analysis (Table 7.4) show that SSL-PCTs and SSL-PCT-FR are statistically significantly better than the SL-PCTs for most of the different amounts of labeled examples considered on all three SOP tasks. More specifically, at the MTR and HMLC tasks, usually at least 200 of labeled examples are needed to achieve statistical significance. On the MLC task, on the other hand, SSL-PCT achieves statistically significantly better result than SL-PCT up to 200 of labeled examples. On this task, the feature weighted variant of the algorithm is more successful: It is statistically significantly better than SL-PCT for all amounts of labeled examples considered.

Considering the feature weighted and non feature weighted semi-supervised methods (both single trees and ensembles), in most cases there is no statistically significant difference between them, except at the HMLC task for 200 of labeled examples, where SSL-PCT-FR statistically significantly outperforms SSL-PCT.

As discussed previously, semi-supervised random forests improve over supervised ones at fewer cases as compared to single trees. The statistically significant improvement over CLUS-RF is observed only for the MLC task at 200 of labeled examples and the HMLC task at 350 of labeled examples.

In none of the cases, the proposed semi-supervised methods perform statistically significantly worse than their supervised counterparts.

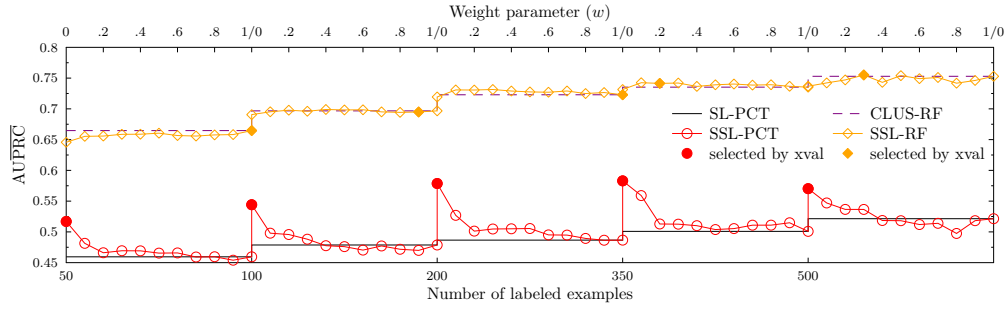
Table 7.4: P -values of the Wilcoxon signed-rank test applied to the performances (AUPRC) of the supervised and semi-supervised single trees (SL-PCT, SSL-PCT and SSL-PCT-FR) on the datasets considered in this study: 12 for multi-target regression, 12 for multi-label classification and for 12 hierarchical multi-label classification. In bold, we report significant p -values (< 0.05). In comparison of two methods, i.e., Method1 vs. Method2, the '−' sign denotes that the sum of ranks where the first method outperformed the second is higher than the sum of ranks where the second method outperformed the first. The '+' sign denotes the opposite.

Methods			Number of labeled examples				
			50	100	200	350	500
MULTI-TARGET REGRESSION							
SL-PCT	vs.	SSL-PCT	0.071 (+)	0.117 (+)	0.025 (+)	0.034 (+)	0.023 (+)
SL-PCT	vs.	SSL-PCT-FR	0.055 (+)	0.48 (+)	0.025 (+)	0.025 (+)	0.01 (+)
SSL-PCT	vs.	SSL-PCT-FR	0.695 (+)	0.937 (−)	0.754 (−)	0.937 (+)	0.239 (+)
CLUS-RF	vs.	SSL-RF	0.969 (+)	0.48 (+)	0.48 (+)	0.367 (+)	0.48 (+)
CLUS-RF	vs.	SSL-RF-FR	0.875 (+)	0.48 (+)	0.969 (+)	0.367 (+)	0.48 (+)
SSL-RF	vs.	SSL-RF-FR	0.969 (+)	0.505 (−)	0.367 (−)	0.367 (−)	0.48 (+)
MULTI-LABEL CLASSIFICATION							
SL-PCT	vs.	SSL-PCT	0.012 (+)	0.008 (+)	0.008 (+)	0.117 (+)	0.071 (+)
SL-PCT	vs.	SSL-PCT-FR	0.008 (+)	0.008 (+)	0.023 (+)	0.012 (+)	0.008 (+)
SSL-PCT	vs.	SSL-PCT-FR	0.969 (+)	0.666 (+)	0.556 (+)	0.078 (+)	0.182 (+)
CLUS-RF	vs.	SSL-RF	0.209 (+)	0.126 (+)	0.078 (+)	0.092 (+)	0.182 (+)
CLUS-RF	vs.	SSL-RF-FR	0.17 (+)	0.117 (+)	0.013 (+)	0.638 (+)	0.695 (−)
SSL-RF	vs.	SSL-RF-FR	0.937 (+)	0.209 (+)	0.754 (−)	0.327 (−)	0.388 (−)
HIERARCHICAL MULTI-LABEL CLASSIFICATION							
SL-PCT	vs.	SSL-PCT	0.937 (+)	0.147 (+)	0.034 (+)	0.025 (+)	0.008 (+)
SL-PCT	vs.	SSL-PCT-FR	1 (−)	0.158 (+)	0.034 (+)	0.025 (+)	0.01 (+)
SSL-PCT	vs.	SSL-PCT-FR	0.388 (−)	0.136 (−)	0.034 (+)	0.695 (−)	0.347 (+)
CLUS-RF	vs.	SSL-RF	0.367 (+)	0.136 (+)	0.099 (+)	0.347 (+)	0.136 (+)
CLUS-RF	vs.	SSL-RF-FR	1 (−)	0.347 (+)	0.136 (+)	0.034 (+)	0.48 (+)
SSL-RF	vs.	SSL-RF-FR	0.367 (−)	0.666 (+)	1 (−)	0.239 (+)	0.969 (−)

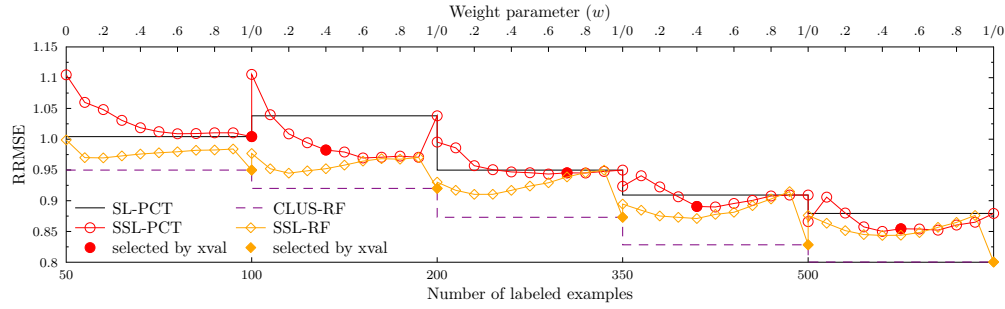
7.3.3 Influence of the w parameter

The w parameter controls the amount of supervision in the models. A value of $w = 0$ results in completely unsupervised PCTs, then, as w increases, PCTs rely more on labeled and less on unlabeled data, and end in being completely supervised for $w = 1$. The ability to fine tune the SSL-PCTs for a given dataset by controlling the amount of influence of unlabeled data is very important in practice, since it can protect semi-supervised PCTs and ensembles thereof from degrading the performance (as compared to their supervised counterparts). When performing semi-supervised learning, there is always a danger (to some extent) of unlabeled data negatively affecting the predictive performances. Several researchers have found that semi-supervised learning may perform worse than supervised learning (Nigam, McCallum, Thrun, & Mitchell, 2000; Cozman, Cohen, & Cirelo, 2002; Z.-H. Zhou & Li, 2007; Yuanyuan Guo et al., 2010). Furthermore, as mentioned before, the success of semi-supervised methods was found to be domain-dependent, i.e., in SSL there are no universally good methods as in supervised learning (Chawla & Karakoulas, 2005). Moreover, choosing the appropriate SSL method for the problem at hand is still an open question. Thus, even if the main focus in SSL is to outperform the supervised methods, care needs to be taken to make the semi-supervised methods safe, i.e., make sure they do not perform worse than their corresponding supervised methods.

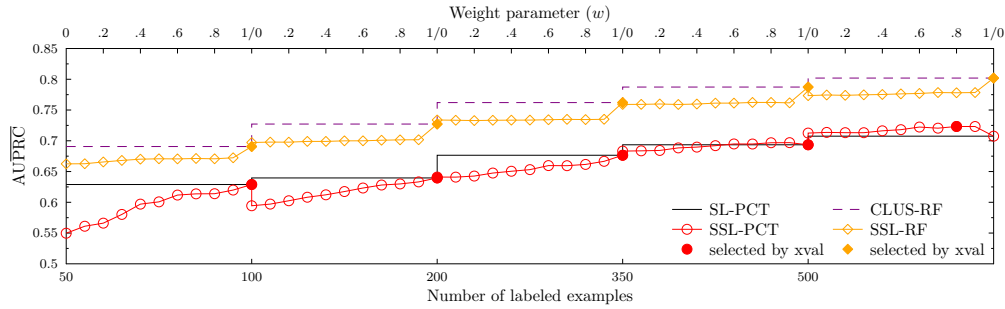
By the virtue of the w parameter, we provide a safety mechanism for SSL-PCTs. In theory, if the optimal value of the parameter w is known, SSL-PCTs and SSL-RF will



(a) Emotions (MLC)



(b) Soil quality (MTR)



(c) Danish farms (HMLC)

Figure 7.5: The effect of the w parameter on the performance of the SSL-PCT (red line) and SSL-RF (orange line) methods on 3 datasets with different types of structured outputs: Emotions (MLC), Soil quality (MTR) and Danish farms (HMLC). The values of the w parameter selected by cross-validation and used in the experimental evaluation are denoted with colored markers.

never perform worse than their supervised counterparts, since SL-PCTs and CLUS-RF are special cases of SSL-PCT and SSL-RF when $w = 1$. However, since the w parameter is optimized via cross-validation on the available labeled data, it is possible that the chosen value of w will not be the right one to achieve the optimal test set performance. Thus, in practice, SSL-PCT and SSL-RF can also perform worse than SL-PCT and CLUS-RF, though this rarely happened in our empirical evaluation (Figures 7.2, 7.3 and 7.4). Considering all the experiments (Figures 7.2, 7.3 and 7.4), SSL-PCTs performed better than their supervised counterpart (SL-PCT) on 52% of occasions, worse on 9% of occasions, and the same on 39% of occasions. Moreover, the degradation of performance, if it happened, was always for a very small amount relative to the improvements of SSL-PCT over SL-PCT. For example, the average relative improvement of SSL-PCTs over SL-PCTs (considering all the experiments) is 40%, while the average degradation is 7%.

Figure 7.5 exemplifies the influence of the parameter w on the predictive performance for 3 different datasets with different types of structured output. Corresponding figures for all the other datasets are given in Appendix B.3.

The Emotions dataset (Fig. 7.5a) requires no supervision, since decreasing the value of w leads to better predictive performance of the SSL-PCT method. For the Soil quality dataset (Fig. 7.5b), the optimal value of w changes with the amount of labeled data. For 50 labeled examples, SSL-PCT is not able to improve over SL-PCT, therefore, the value of $w = 1$ is chosen to avoid degradation of the performance. As the number of labeled examples increases beyond 50, moderate amount of supervision is desired for SSL-PCT method to achieve the optimal performance and outperform SL-PCT. Finally, the Danish farms dataset (Fig. 7.5c) is an example of a dataset where more supervision (i.e., higher w) leads to better predictive performance of SSL-PCT. However up to 500 of labeled examples, the SSL-PCT method is not able to improve over its supervised counterpart, therefore the value of $w = 1$ is chosen to prevent the degradation of the performance.

A general recommendation for the value of w is difficult to provide, since, as exemplified on Figure 7.5, the optimal value of w can vary from dataset to dataset, and even within the same dataset as the amount of labeled data changes. As previously discussed, semi-supervised methods are known to be domain-dependent, i.e., the success of semi-supervised methods can vary from one dataset to another. In other words, different datasets may require different amount of supervision. Since, the w parameter controls the amount of supervision, it is not surprising that different values of w are appropriate for different datasets. Hence, this parameter, as done in our empirical evaluation, needs to be optimized by cross-validation for each specific use case.

7.3.4 Model size and interpretability of supervised and semi-supervised trees

As has been previously stressed out, interpretability is an important property of the models in many applications of machine learning. The models produced by semi-supervised PCTs are readily interpretable, since they are in the form of decision trees. To the best of our knowledge, currently there is no other semi-supervised method for structured output prediction that produces interpretable models.

The interpretability of the tree shaped models is governed by their size. That is, a very large tree may be hard to analyze, and vice versa, a tree with fewer nodes may be easier to understand. Though, the size of the tree is usually a trade-off between accuracy and interpretability. Namely, very small trees are easy to understand, but they may fail to capture structural information in the data and therefore fail to deliver satisfactory predictive performance. On the other hand, larger trees may avoid these issues but with a cost of reduced understandability. Certainly, increased size of the tree does not necessarily imply improved predictive capability, due to the danger of overfitting. In general, it is hard to tell (*a priori*) when the tree is "big enough", i.e., whether it overfits or underfits.

Here we discuss the differences in model sizes between the supervised and semi-supervised PCTs, which are given in Table 7.5. We can observe that, on average, the semi-supervised trees are somewhat larger than the supervised trees. This is intuitive since semi-supervised algorithms uses much more data to grow the trees, i.e., labeled and unlabeled examples. If we focus on individual datasets, we can observe that the size of both the supervised and semi-supervised trees is mainly in the range of few tens of nodes. Which, we maintain, is a reasonable size for manual inspection. However, there are a few exceptions. Semi-supervised trees are sometimes, with few hundreds of nodes, much larger than the corresponding supervised tree. In particular, this can be observed on the following datasets: Bicycle (≥ 200 labeled), RF1, SCM20D (500 labeled), Mediana (≥ 350 labeled), Danish

Table 7.5: Model sizes, in terms of number of nodes, obtained with the supervised PCTs (SL-PCT) and the semi-supervised PCTs (SSL-PCT) on the 12 MTR datasets, 12 MLC datasets and 12 HMLC datasets.

Dataset	Number of labeled examples									
	50		100		200		350		500	
	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT
MULTI-TARGET REGRESSION										
Bicycle	3.8	37.6	9.6	52.4	14.4	98.8	31.8	138.2	55.4	203.6
Enb	9.4	42.8	21	67.6	42.8	42.8	57.4	57.4	71.4	71.4
Eunite	2.4	1	4	4	6.8	6.8	14.2	14.2	21.6	21.6
RF1	6.8	70.6	17	134.2	33.8	235	68.4	358.2	101.4	463.8
SCM1D	3.8	6	7.4	7.4	14.2	14.2	24.4	24.4	33.6	33.6
SCM20D	3.2	4.4	4.8	3.4	8.8	8.8	18.6	18.6	27.6	473
Scpf	3.8	4	5.2	4	8.6	7.2	9	8.6	11.4	12.4
Soil quality	1.8	1.8	3.4	13.6	7.6	15.8	10	45.6	15.6	52.4
Solar flare 2	2.8	20.4	3.4	30.8	3.8	40.2	5	28.6	6.2	25.4
Vegetation clustering	1.6	1.6	2.4	1	5	21.8	11	39.6	14.4	26.6
Vegetation condition	4.2	4.2	4.6	4.6	7.4	7.4	11.8	11.8	14.6	14.6
Water quality	1	1	1	1	1.8	1.8	4.4	21.4	6	24.8
Average:	3.7	16.3	7.0	27.0	12.9	41.7	22.2	63.9	31.6	118.6
MULTI-LABEL CLASSIFICATION										
Bibtex	1	21	1	21.4	1	19.4	1	19.4	1	19
Birds	1	15.8	1	20.8	1.2	15.2	2.6	13	4	12.8
Corel5k	1	33.6	1	1	1	1	1	1	1	1
Emotions	3	18.8	5	19	7.4	7.4	11.8	19.2	14.8	14.8
Enron	1	17.6	1	19.2	1	21.4	1	21.2	1.2	20.2
Genbase	2.6	22.8	21.4	23	31.2	26.6	37.2	36.8	43	41.4
Mediana	1.4	32.8	4	4	7.8	7.8	10.2	10.2	12.4	12.4
Medical	1	15.4	1	41.8	1	43.4	3.2	63.6	13.6	63.2
Scene	7.8	25.8	12.4	28.4	19.8	28.4	31.8	29	36.6	37.2
SIGMEA real	3	35.2	3.2	3.2	4.6	4.6	6.6	6.6	8.4	8.4
Slovenian rivers	1	39.2	1	64	1.4	67.6	3	70.8	3.8	55.8
Yeast	1	1	1	1	1	25	1.2	25	2.2	25
Average:	2.1	23.3	4.4	20.6	6.5	22.3	9.2	26.3	11.8	25.9
HIERARCHICAL MULTI-LABEL CLASSIFICATION										
Danish farms	1	1	1.4	1.4	3.2	3.2	6	6	8.8	257.6
Slovenian rivers	1	19.8	1	18.2	1	47.4	1.4	52.4	3	50.8
Enron	1	11.6	1	13.8	1.6	15.8	5.6	16.6	6.6	17.2
ImCLEF07A	1	42.8	2	86.6	6.6	133.4	12.8	150.4	19.4	177.4
ImCLEF07D	1	47.4	1.8	215	4.4	295.2	7	159.2	15	172.2
Diatoms	1	49.6	1	56.2	1	57.4	1	72.4	1	89.8
Cellcycle-GO	1	1	1	1	1	1	1	1	1	1
Church-GO	1	1	1	1	1	1	1	1	1	1
Derisi-GO	1	1	1	1	1	1	1	1	1	1
Eisen-GO	1	1	1	1	1	1	1	1	1	27.4
Expr-GO	1	1	1	1	1	1	1	9.2	1	9.2
Pheno-GO	1	1	1	1	1	1	1	1	1	1
Average:	1.0	14.9	1.2	33.1	2.0	46.5	3.3	39.3	5.0	67.1

farms (500 labeled), ImCLEF07A and ImCLEF07D (≥ 200 labeled). These cases may be unfeasible for analysis.

We search for the reason of such big semi-supervised tree in these cases. Figure 7.6 presents the analysis of the influence of the w parameter on the tree size. it reveals that unsupervised trees ($w = 0$) are much bigger in size than semi-supervised ($0 < w < 1$) or supervised ($w = 1$) tree. Unsupervised trees do not rely on the output space at all, therefore, it is understandable that in the presence of a very large number of unlabeled

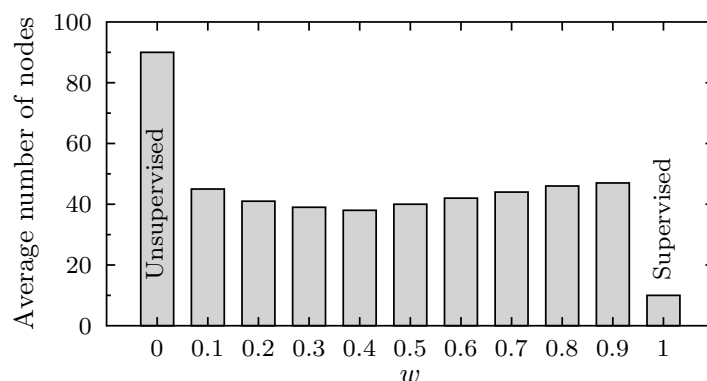


Figure 7.6: Average tree size per value of the w parameter across (all datasets and amounts of labeled data).

data, big trees are grown. Some of the cases mentioned previously, where SSL-PCTs yield very big trees, are indeed unsupervised (please refer to Appendix B.3). Recall that, the w parameter was optimized for predictive performance, but by increasing the value of w (i.e., increasing the degree of supervision) the trade-off between tree size and model performance can be achieved. For example, SSL-PCT algorithm on RF1 dataset with 100 of labeled examples and $w = 0.1$ achieves almost the same RRMSE as with the optimal weight $w = 0$, but the resulting tree has only 16 nodes, i.e., it is 8 times smaller.

Another aspect that (differently) affects the tree size of supervised and semi-supervised trees is pruning. Supervised trees, and in particular tree for MLC and HMLC tasks, are often pruned down to a single node², while this is very rarely observed at semi-supervised trees (Table 7.5). Note that for the 6 HMLC datasets coming from the functional genomics domain, both supervised and semi-supervised trees mostly have only one node. Recall that, these datasets have very large label hierarchies which are very sparsely populated. The amount of labeled data we considered (i.e., up to 500 of labeled examples) seems not sufficient to build meaningful trees – either supervised or semi-supervised.

Due to the effect of excessive pruning, the average tree size (as depicted on Figure 7.6) of supervised trees is much smaller than the average size of semi-supervised trees. It seems such extreme pruning is prevalent on datasets with large number of labels and low label cardinality, such as Bibtex, Enron or Diatoms (Section 7.2.1). The target space on these datasets resembles a sparse 0/1 matrix. We hypothesize that the variance estimates on such target space are uninformative. Therefore, the pruning procedure (which is based on the variance estimates) severely prunes the trees. Semi-supervised PCTs, which also use the descriptive space to estimate the variance, do not suffer from such a excessive pruning. The following observation supports our hypothesis: The value of the w parameter which gives the optimal predictive performance on MLC and HMLC datasets is often 0 (Appendix B.3), i.e., SSL-PCTs do not rely on the target space at all.

Arguably, semi-supervised trees sometimes may have unfair advantage, since they compete with a default model (i.e., completely pruned tree). Therefore, we also report the performance of unpruned trees in Appendix B.1. We can observe that all of the conclusions (with respect to the predictive performance) made previously for pruned trees (Section 7.3.1), hold also for unpruned trees. In fact, unpruned semi-supervised trees perform even more favorably: For all amounts of labeled data considered, they are statistically

²Note that, we have also experimented with pruning procedures different than M5 pruning, namely C4.5 pruning (Quinlan, 1993), and F-test pruning (Vens et al., 2008), however the findings are similar.

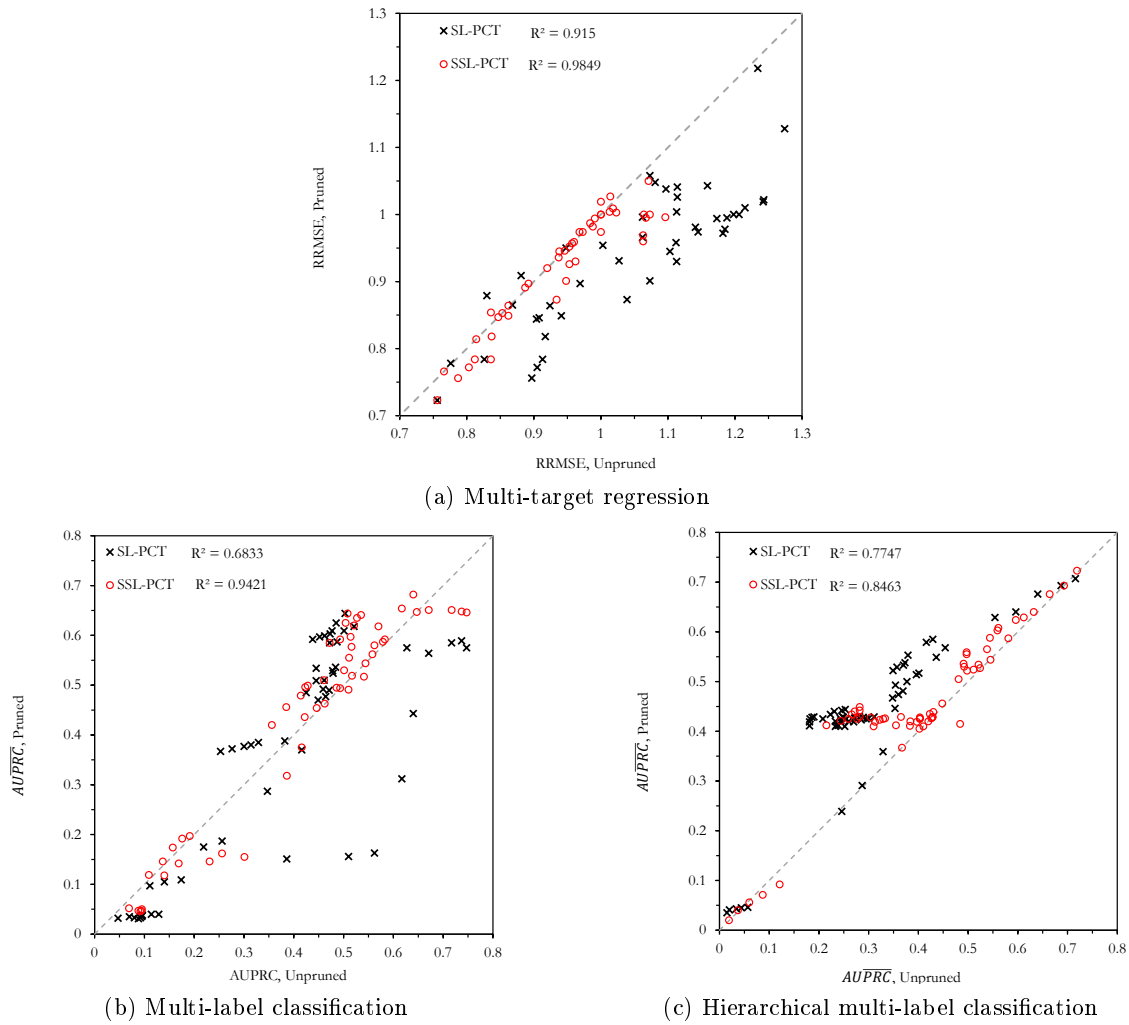


Figure 7.7: Comparison of the predictive performance of the pruned and unpruned supervised (SL-PCTs) and semi-supervised trees (SSL-PCTs) shows that SSL-PCTs are less affected by pruning since the performance of pruned and unpruned trees is very well correlated.

significantly better than the unpruned supervised trees (Table B.2).

We study the effect of pruning on the performance of the trees in more detail. Figure 7.7 presents the comparison of the predictive performance of supervised (SL-PCTs) and semi-supervised trees (SSL-PCTs) before and after pruning. First of all, we can observe that the pruning is mostly beneficial for the predictive performance, both for SL-PCTs and SSL-PCTs, however, there are some examples where pruning severely degrades the performance of supervised trees (Figure 7.7b, lower-right quadrant). Next, we can observe that SSL-PCTs are less affected by pruning (even though their size is drastically reduced, see Tables 7.5 and B.1), since predictive performance of pruned and unpruned SSL-PCTs is mostly similar. As this is generally not the case with SL-PCTs, we conjecture that semi-supervised PCTs are less prone to overfitting than supervised PCTs.

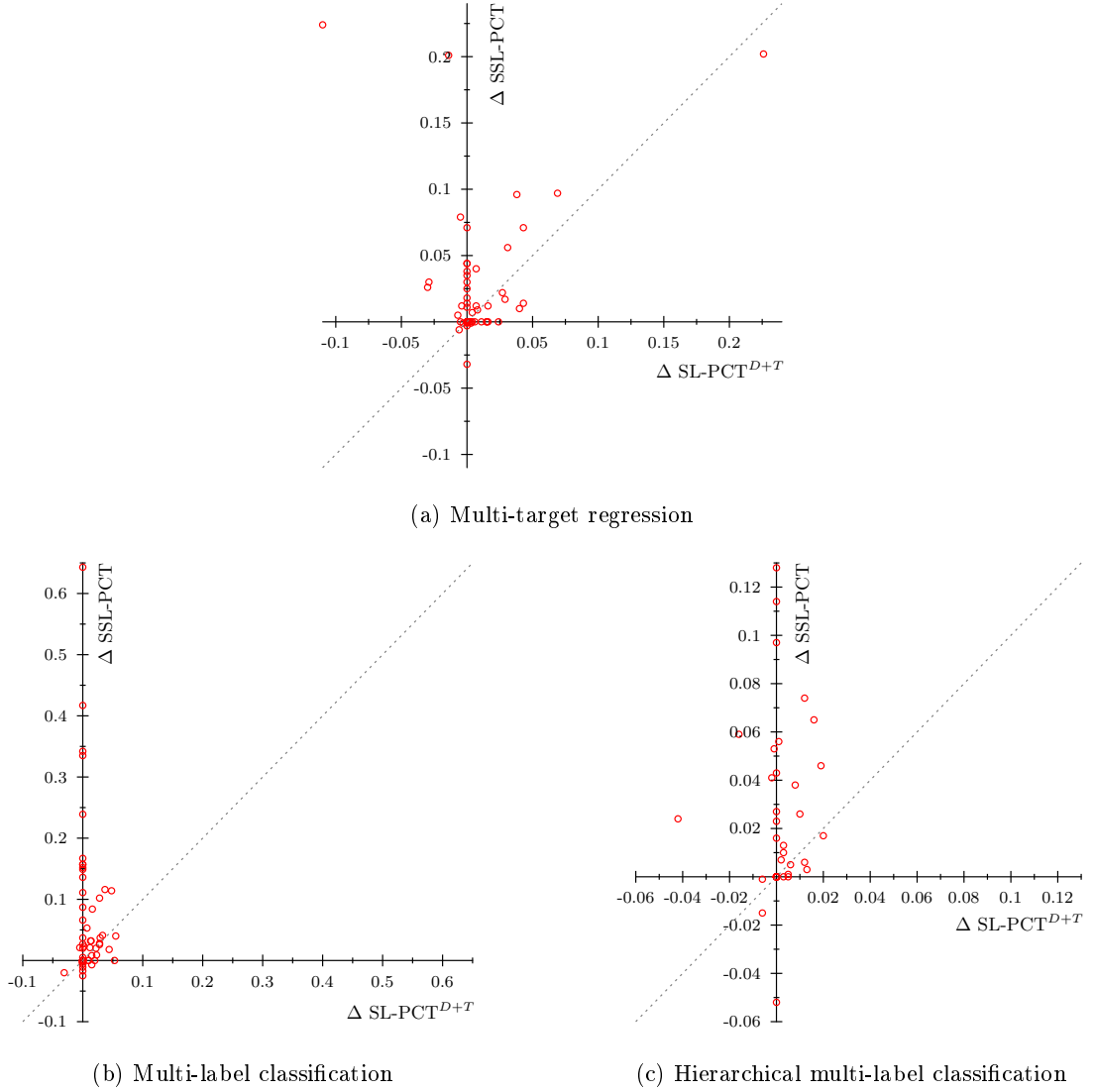


Figure 7.8: Comparison of the predictive performance of SSL-PCT and SL-PCT^{D+T} with the SL-PCT across all datasets and different amounts of labeled data. $\Delta\text{SSL-PCT}$ and $\Delta\text{SL-PCT}$ denote the difference between the SL-PCT and SSL-PCT or SL-PCT^{D+T} , respectively. Negative values mean that SL-PCT is better, while positive values mean the opposite. SSL-PCT outperforms SL-PCT^{D+T} above the diagonal (dashed line), while SL-PCT^{D+T} outperforms SSL-PCT below the diagonal line. SSL-PCT improves over SL-PCT for positive values of the y -axis, and degrades the performance of SL-PCT for negative values of the y -axis. The same relations hold for SL-PCT^{D+T} , but for x -axis.

7.3.5 Influence of the unlabeled data

Note that, semi-supervised PCTs differ from supervised PCTs in two aspects: (1) they consider both the descriptive and target space to evaluate the candidate splits, and (2) they use unlabeled examples in addition to labeled ones. We have demonstrated that semi-supervised PCTs offer predictive performance that is highly competitive to the one supervised PCTs, but we might question whether the improvements stem from combination of (1) and (2), or is it maybe only (1) sufficient to improve over supervised PCTs? To this

end, we compare the predictive performance of SSL-PCTs to the supervised variant of the algorithm which uses both the descriptive and target space to evaluate the candidate splits, but it is not supplied with the unlabeled examples (denoted as SL-PCT^{D+T}). In such a way, we can precisely assess the influence of the unlabeled examples, since in both cases *de facto* the same algorithm is used and the only difference comes from the availability of unlabeled data. In the experimental analysis we optimized the value of the w parameter in the same way as for the SSL-PCTs, via internal 3-fold cross validation.

Considering all the datasets and amounts of labeled data considered, the SL-PCT^{D+T} algorithm performs better than the SL-PCT in 36% cases, the same in 54% of cases and worse in 11% of cases. Recall that, the corresponding figures for the SSL-PCTs algorithm are 52%, 39% and 9%. Thus, even without unlabeled data, the semi-supervised PCTs proposed in this thesis can improve over SL-PCTs, but they have a much better chance to do so if they are supplied with unlabeled data. The following result, however, evidences that the unlabeled data are indeed the principal component for the success of the SSL-PCT method: The average relative improvement of SL-PCT^{D+T} over SL-PCT is a mere 4%, while for SSL-PCT this figure is 40%. This observation, i.e., the importance of unlabeled data, is in line with the findings of Ženko (2007), who considered predictive clustering rules where heuristic that guides the rule learning process considers both the descriptive and target spaces (similarly as SSL-PCT). The results reported by Ženko (2007) show that the inclusion of the descriptive space into the heuristic was not beneficial for the predictive performance of predictive clustering rules. However, the study of Ženko (2007) was performed in the supervised learning context, i.e., unlabeled examples were not used.

Figure 7.8 presents a detailed comparison of improvement/degradation of SL-PCT^{D+T} and SSL-PCT over SL-PCT. As stated previously, we can observe that the SSL-PCT method improve over SL-PCT more frequently than SL-PCT^{D+T} (most of the points on the figure are above the diagonal). Furthermore, SSL-PCT yields much larger improvements over SL-PCT than SL-PCT^{D+T} (for most of the points on the figure, the improvement along the y -axis is much larger than the improvement along the x -axis). However, there is some complementarity between the two methods. Namely, SL-PCT^{D+T} sometimes improves over SL-PCT even when this is not the case with SSL-PCT (Figure 7.8, values on the positive side of the x -axis, below the dashed line). This observation suggests that the proposed algorithm for semi-supervised learning of PCTs, might be worthy of attention even in its supervised form. This, however, should be inspected in more detail in the future.

7.4 Summary of the Results

In this section, we summarize the findings from the empirical evaluation of the proposed algorithm for the semi-supervised learning of predictive clustering trees. In the proposed algorithm, both the target and the descriptive space are considered for evaluation of candidate splits. Semi-supervised trees built in such a way can exploit unlabeled examples, while preserving the appealing characteristics of the supervised trees: producing readily interpretable models and being fast to learn. To the best of our knowledge, this is currently the only available interpretable semi-supervised method for structured output prediction.

In our experiments we investigate whether the proposed semi-supervised predictive clustering trees, and ensembles thereof, can improve the performance of supervised predictive clustering trees and random forests. Next, we scrutinize various aspects of the proposed method: influence of the w parameter which controls the amount of supervision, model sizes and interpretability, and influence of the unlabeled data to the performance of the algorithm.

We performed the empirical evaluation over a wide range of datasets with different types of the structured output. In particular, we used 12 datasets for the task of multi-target regression, 12 datasets for the task of multi-label classification and 12 datasets for the task of hierarchical multi-label classification. We summarize the main findings as follows:

- The proposed semi-supervised predictive clustering trees showed attractive predictive performance on all three structured output tasks considered: On most of the datasets, it showed predictive performance superior to the one of supervised predictive clustering trees.
- Due to the parametrization which controls the amount of supervision, the proposed semi-supervised predictive clustering trees are safe to use: They either improve the accuracy of their supervised counterparts, or perform very similar to them. Very seldom did they degrade the performance of supervised predictive clustering trees, and if they did, the degradation was very small as compared to improvements.
- The superiority of semi-supervised over supervised predictive clustering trees is not preserved in the ensemble setting, though semi-supervised random forests do sometimes outperform supervised random forests.
- The results showed that weighting descriptive features by their relevance may help the predictive performance of semi-supervised predictive clustering in some cases, but the results of empirical investigation are not strong enough to advocate the use of feature weighting by default. Thus, by the principle of Occam's razor, simpler solution should be preferred, that is, non feature weighted one.
- The semi-supervised trees are generally somewhat larger than the supervised ones, meaning that they could be more challenging to interpret, though the sizes of the trees are mostly reasonable for manual inspection. In other cases, some trade-off between the tree size and predictive performance can be achieved by tuning the w parameter.
- The results suggest that semi-supervised trees overfit less and are not prone to excessive pruning as are supervised predictive clustering trees. When unpruned trees are compared, semi-supervised predictive clustering trees are even more superior to supervised ones.
- The proposed semi-supervised algorithm in its supervised version (i.e., which does not use unlabeled data, but it considers both the descriptive and target space for the evaluation of splits) can improve over the (standard) supervised predictive clustering trees, however, when the algorithm is supplied with unlabeled data, the outperformance is more probable and the improvements are much larger.

Chapter 8

Semi-Supervised PCTs for Primitive Outputs

This thesis deals with semi-supervised learning for structured output prediction. However, the methods we propose are not limited to structured outputs. They can be readily applied to machine learning problems with primitive output spaces as well, such as binary classification, multi-class classification and (single-target) regression. Note that, these machine learning tasks are special cases of structured outputs defined in Section 3.1: predicting multiple targets. Namely, (single-target) classification and regression can be considered as multi-target problems where the length of the output vector is one.

In Chapter 7 we have proposed semi-supervised predictive clustering trees and demonstrated their attractive properties on datasets with structured outputs. While machine learning tasks with structured outputs are gaining increasing attention recently (Q. Yang & Wu, 2006; Kriegel et al., 2007, 1; Dietterich et al., 2008), research on tasks with primitive outputs (i.e., classification and regression) is still prevalent in the field of machine learning. As we previously mentioned, semi-supervised predictive trees can be readily applied to such tasks.

In this Chapter, we aim to investigate whether the findings discovered on datasets with structured outputs (Chapter 7) translate also to machine learning tasks with primitive outputs. To this end, we evaluate semi-supervised predictive clustering trees on a large collection of datasets for binary classification, multi-class classification and regression. In binary classification each example can have one of the possible two labels (i.e., a positive or negative), while in multi-class classification each example can have one of the several possible labels. In regression, each example is assigned a single numerical value.

8.1 Experimental Design

In this section, we first describe the datasets used in the experimental evaluation. Next, we present the evaluation procedure, the specific parameter settings of the algorithms and the performance metrics.

8.1.1 Data description

To evaluate the predictive performance of SSL-PCTs, we use 12 binary classification datasets, 12 multi-class classification datasets and 12 regression datasets. Classification datasets were mainly obtained from the UCI repository (Lichman, 2013), OpenML repository (Vanschoren, Van Rijn, Bischl, & Torgo, 2014). Regression datasets were obtained from the Keel repository (Alcalá et al., 2010) and the repository of Luís Torgo (Torgo,

Table 8.1: Characteristics of the binary classification datasets. N : number of instances, D/C : number of descriptive attributes (discrete/continuous), P : percentage of the majority class.

Dataset (Reference)	Domain	N	D/C	P
Abalone (Lichman, 2013)	Biology	4177	1/7	50%
Adult (Lichman, 2013)	Socio-economic	48842	8/6	76%
Bank (Lichman, 2013; Moro et al., 2011)	Economy	4521	9/7	88%
Banknote (Lichman, 2013)	Images	1372	0/4	55%
Biodegradation (Lichman, 2013; Mansouri et al., 2013)	QSAR	1055	0/41	66%
Diabetes (Lichman, 2013)	Medicine	768	0/8	65%
Eyestate (Lichman, 2013)	Medicine	14980	0/14	55%
Madelon (Lichman, 2013; Guyon et al., 2004)	Synthetic	2000	0/500	50%
Mushroom (Lichman, 2013)	Biology	8124	22/0	51%
Pgp (Levatić et al., 2013)	QSAR	932	0/34	52%
Phishing (Lichman, 2013)	Security	11055	30/0	55%
Thyroid (Lichman, 2013)	Medicine	3772	21/6	93%

Table 8.2: Characteristics of the multi-class classification datasets. N : number of instances, D/C : number of descriptive attributes (discrete/continuous), K : number of classes, P : percentage of the majority class.

Dataset (Reference)	Domain	N	D/C	K	P
Baseball (Vanschoren et al., 2014; Simonoff, 2013)	Sport	1340	1/15	3	90%
Cardiotocography3 (Lichman, 2013)	Medicine	2126	0/35	3	77%
Cardiotocography10 (Lichman, 2013)	Medicine	2126	0/35	10	27%
Cmc (Vanschoren et al., 2014)	Socio-economic	1473	7/2	3	42%
Gasdrift (Lichman, 2013; Vergara et al., 2012)	Chemistry	13910	0/128	6	21%
Gasdriftdc (Lichman, 2013; Vergara et al., 2012)	Chemistry	13910	0/129	6	21%
GesturePhase (Lichman, 2013)	Video	9873	0/32	5	29%
Imagesegment (Lichman, 2013)	Images	2310	0/18	7	14%
MiceProtein (Higuera et al., 2015)	Biology	1080	3/77	8	14%
Optdigits (Lichman, 2013)	Images	5620	0/62	10	10%
Pageblocks (Lichman, 2013)	Text	5473	0/10	5	89%
Pendigits (Lichman, 2013)	Images	10992	0/16	10	10%

2016). From these repositories, we selected datasets which differ in the domains they represent, number of attributes and examples, and distribution of classes. We focus mainly on datasets which have more than a thousand of examples, so that we can simulate a scenario relevant for SSL, i.e., when large amounts of unlabeled data are available.

The characteristics of binary classification, multi-class classification and regression datasets are given in Tables 8.1, 8.2 and 8.3, respectively.

8.1.2 Experimental setup and evaluation procedure

In Chapter 7 we have proposed semi-supervised predictive clustering trees (SSL-PCT) and semi-supervised random forest tree ensemble (SSL-RF). We compare these methods to their supervised counterparts, i.e., supervised classification trees (SL-PCTs), and supervised random forests (CLUS-RF). Those are the most reasonable baselines, since the goal is to measure the contribution of the unlabeled data to the overall performance under the same conditions. In empirical evaluation of semi-supervised PCTs on structured outputs (Chapter 7), it was not clear whether feature weighting is beneficial for the predictive performance, thus, here we do not use feature weighted PCTs.

In the experiments, both supervised and semi-supervised trees are pruned, classifica-

Table 8.3: Characteristics of the regression datasets. N : number of instances, D/C : number of descriptive attributes (discrete/continuous).

Dataset (Reference)	Domain	N	D/C
2dplanes (Torgo, 2016)	Artificial	40768	0/10
Abalone (Torgo, 2016)	Biology	4177	1/7
Elevators (Torgo, 2016)	Optimal control	16559	0/18
Kinematics (Torgo, 2016)	Robotics	8192	0/8
Laser (Alcalá et al., 2010)	Optics	993	0/4
Plastic (Alcalá et al., 2010)	Plastic strength	1650	0/2
Pole (Torgo, 2016)	Telecommunication	5000	0/48
Stock (Torgo, 2016)	Economy	950	0/10
ChEMBL249 (Vanschoren et al., 2014)	QSAR	2759	1024/0
ChEMBL2850 (Vanschoren et al., 2014)	QSAR	1211	1024/0
ChEMBL2973 (Vanschoren et al., 2014)	QSAR	1521	1024/0
ChEMBL243 (Vanschoren et al., 2014)	QSAR	4442	1024/0

tion trees with the C4.5 pruning procedure, and regression trees with the M5P pruning procedure (Quinlan, 1993). For each variant which uses an ensemble learning approach (i.e. CLUS-RF and SSL-RF), we construct random forests consisting of 100 trees. The trees in random forest are not pruned and the number of random features at each internal node is set to $\lfloor \log_2(D) + 1 \rfloor$, where D is the total number of features (Breiman, 2001).

In order to explore the influence of the amount of labeled data on the predictive performance of the semi-supervised methods and compare them with their supervised counterparts, we execute experiments where we varied the absolute number of labeled examples in the set $\{25, 50, 100, \text{ and } 200, 350, 500\}$. The labeled data used for training the predictive models (both supervised and semi-supervised) are randomly selected from the available training data, while the remaining examples serve both as unlabeled data and as a test set. Namely, we temporarily remove their labels and provide the examples to the algorithm to serve as unlabeled data during training. When the models are built, the performance is evaluated on the test set composed of the unlabeled examples with their true labels restored. In other words, the evaluation of the semi-supervised methods is performed in the context of transductive learning. For a fair comparison, the supervised methods are trained only on the labeled part of the data and evaluated on the same test set. To get more stable results, the procedure is repeated 10 times with different random initializations. The predictive performance reported in the results is the average of the performance values obtained from the 10 runs.

For each run, we optimize the weight parameter w by performing 3-fold cross-validation on the available labeled part of the training set. During the cross-validation procedure, the semi-supervised methods are supplied with the available unlabeled examples. We consider values of the parameter w ranging from 0 to 1 with a step of 0.1.

For datasets with discrete outputs (i.e., binary and multi-class classification) we use accuracy to assess the predictive performance of the algorithms, while on datasets with continuous outputs we use relative root-mean-square-error (RRMSE).

To investigate whether the observed differences in performance among the methods are statistically significant, we use the non-parametric Wilcoxon paired signed rank test (Wilcoxon, 1945). More specifically, we compare the predictive performance of two methods over multiple datasets. In all the experiments reported in this study, the significance level is set to 0.05.

8.2 Results and Discussion

In this section, we present the results of the empirical evaluation. We first present results on classification datasets (i.e., binary classification and multi-class classification), followed by a presentation of results on regression datasets. Next, we perform statistical analysis of the results and we then analyze the influence of the number of labeled examples on predictive performance of semi-supervised algorithms. Finally, we present examples of supervised and semi-supervised trees and we discuss their interpretability.

8.2.1 Analysis of predictive performance

In this section, we present the predictive performance of the semi-supervised and supervised PCTs (and ensembles thereof) on machine learning tasks with primitive output space, namely binary and multi-class classification, and regression. We investigate the influence of the different amounts of labeled examples (i.e., from 25 to 500).

8.2.1.1 Binary classification

Figure 8.1 presents classification accuracy of semi-supervised (SSL-PCT and SSL-RF) and supervised methods (SL-PCT and CLUS-RF) on the 12 binary classification datasets, with an increasing amount of labeled data.

We can observe consistent improvement of SSL-PCTs over supervised SL-PCTs on a number of datasets: Abalone, Adult, Bank, Biodegradation, Diabetes, Madelon and Pgp. The degree of improvement is different from one dataset to another, from relatively small improvement (i.e., around 1% in accuracy) on Biodegradation dataset, to substantial improvement (i.e., up to 16% in accuracy) on Madelon dataset. On other datasets, SSL-PCTs and SL-PCTs seem to show similar behavior, with small improvements or degradations in performance of SSL-PCT as compared to SL-PCT. The different performance of SSL methods on different datasets is somewhat expected, since success of the semi-supervised methods is known to be domain-dependent (Chawla & Karakoulas, 2005).

We next compare the performance of semi-supervised random forests (SSL-RF) and supervised random forests (CLUS-RF). Consistent improvement of SSL-RF over CLUS-RF is observed on Abalone, Banknote, Madelon, Mushroom (for more than 100 of labeled examples) and Thyroid datasets. Similarly as it was observed for structured output spaces (Section 7.3.1) It seems that the success of SSL-RF over CLUS-RF is not directly connected with the success of its base model, i.e., SSL-PCTs. More specifically, improvement of SSL-PCTs over SL-PCTs does not guarantee the improvement of SSL-RF over CLUS-RF (e.g., Adult and Bank datasets), and vice versa, SSL-RF can improve over CLUS-RF even if SSL-PCTs does not improve over PCTs (e.g., Banknote and Thyroid datasets). This means that the advantage introduced by the adoption of the SSL approach is somehow orthogonal with respect to the advantage introduced by the ensemble learning approach.

8.2.1.2 Multi-class classification

Figure 8.2 presents classification accuracy of semi-supervised (SSL-PCT and SSL-RF) and supervised methods (SL-PCT and CLUS-RF) on 12 multi-class classification datasets, with increasing amount of labeled data.

Similarly to the case of binary classification, SSL-PCTs consistently improve over the accuracy of SL-PCTs on a number of datasets: Baseball, Cardiotocography³, Cardiotocography¹⁰, GesturePhase, Optdigits and Pendigits. On other datasets, the two methods perform very similarly, with the exception of the Cmc dataset, where SSL-PCTs are more accurate than SL-PCTs only when 100 or 200 labeled examples are available.

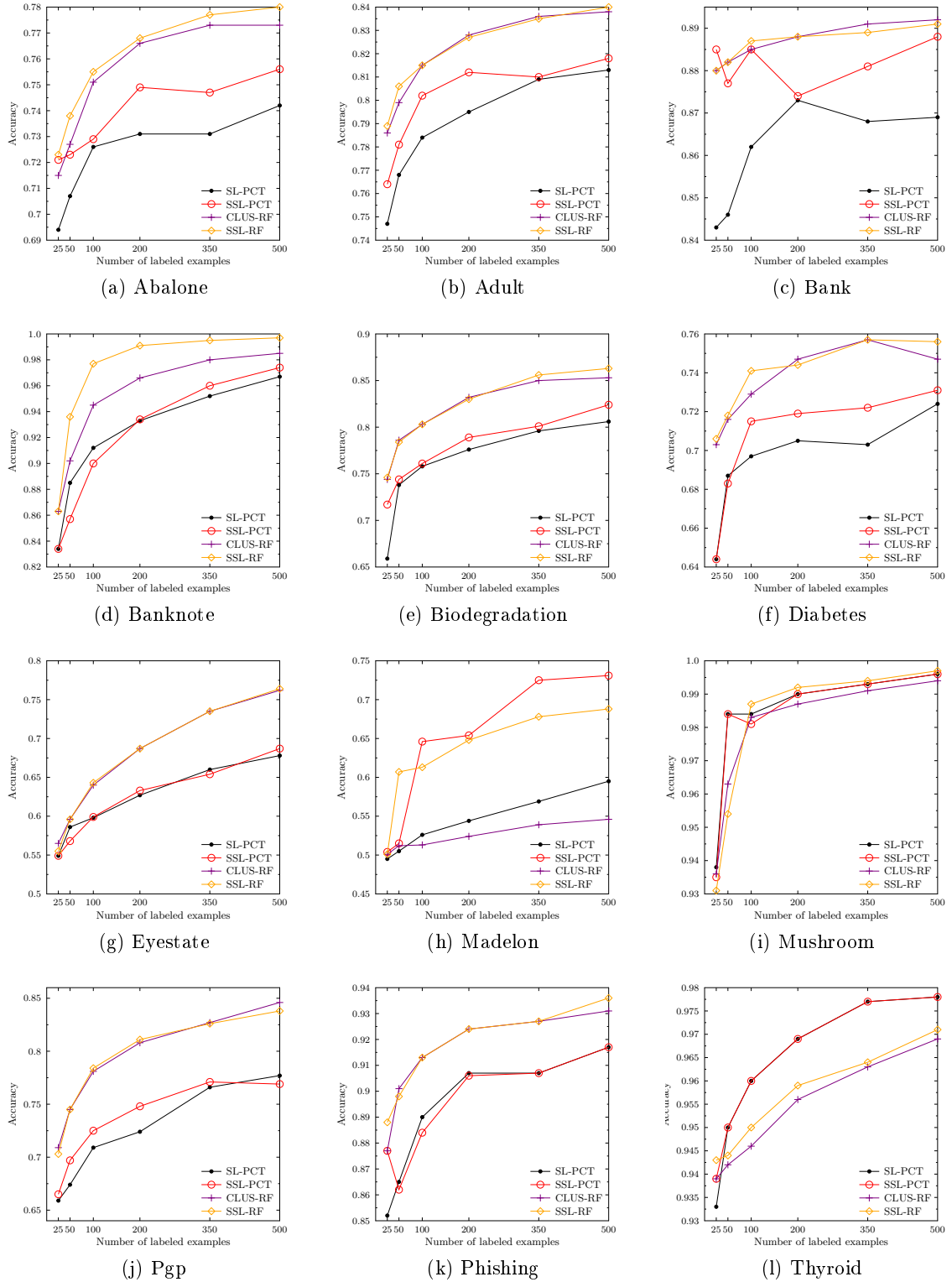


Figure 8.1: Accuracy of the supervised and semi-supervised methods on the binary classification datasets.

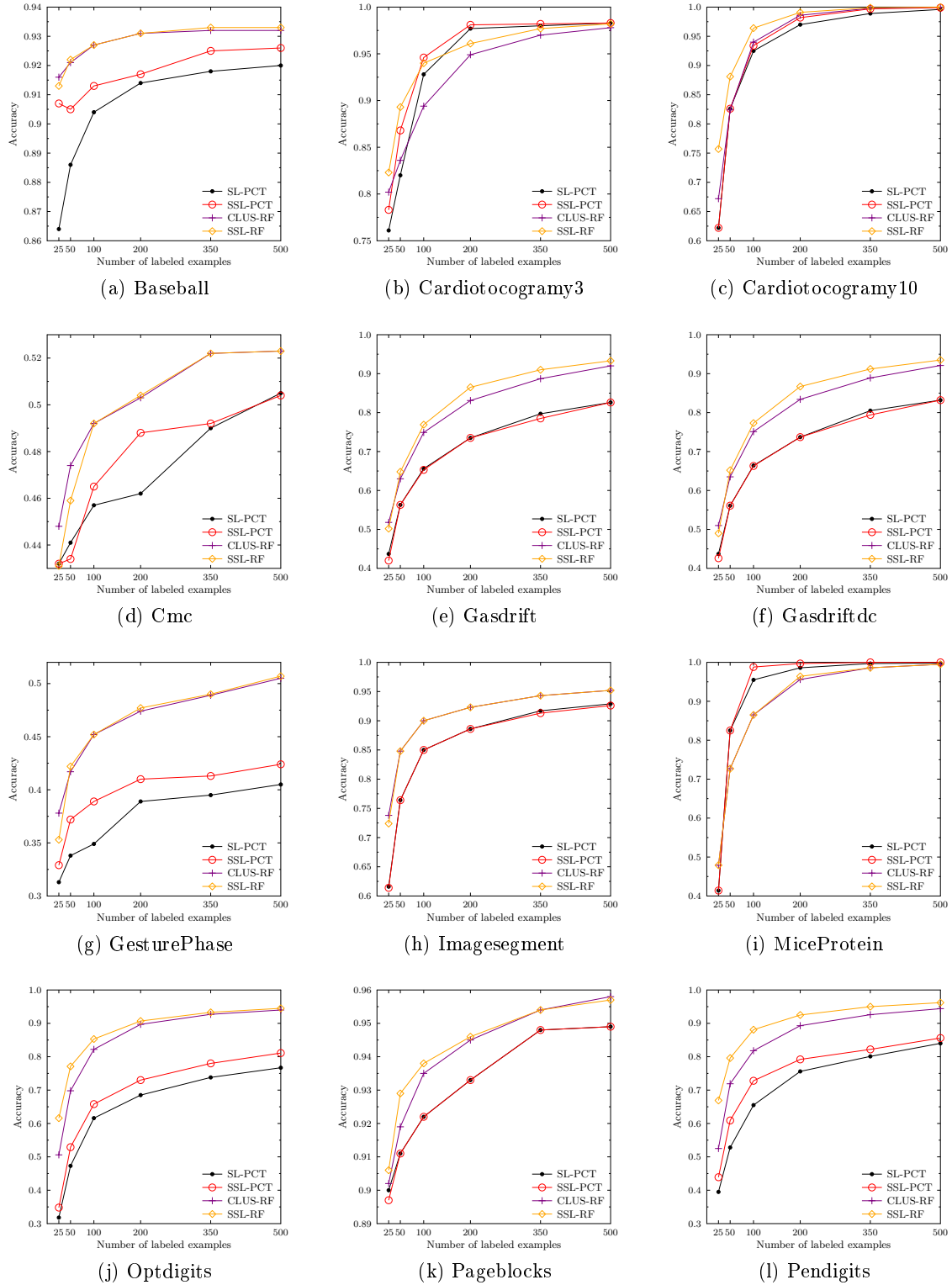


Figure 8.2: Accuracy of the supervised and semi-supervised methods on the multi-class classification datasets.

SSL-RF consistently improves over CLUS-RF on Cardiotocogramy3, Cardiotocogramy10, Gasdrift, Gasdriftsc, Optdigits, Pageblocks, and Pendigits datasets, while on the other five datasets the performance of the two methods is very similar. Again, the results suggest that SSL-RF can improve over CLUS-RF regardless of whether SSL-PCTs were able to improve over SL-PCTs or not. For example, SSL-RF was successful on Gasdrift and Gasdriftsc datasets, while SSL-PCT was not. On the other hand, the success of SSL-PCTs does not necessarily transfer to ensemble of SSL-PCTs, as it is the case on Baseball and GesturePhase datasets.

8.2.1.3 Regression

Figure 8.2 presents RRMSE of semi-supervised (SSL-PCT and SSL-RF) and supervised methods (SL-PCT and CLUS-RF) on 12 regression datasets, with increasing amount of labeled data.

We can observe that SSL-PCTs achieve lower predictive error than SL-PCTs on a number of datasets: 2dplanes, Elevators, Kinematics, Laser and Pole. Furthermore, it seems that SSL-PCTs are especially effective on QSAR datasets (ChEMBL249, ChEMBL2850, ChEMBL2973 and ChEMBL243) for small amounts of labeled data (i.e., from 25 to 100 of labeled examples).

Semi-supervised random forests achieve consistent improvement over CLUS-RF on majority of datasets. Similarly as observed before, the success of SSL-RF is not governed by the success of SSL-PCT and vice versa (e.g., Plastic and Stock datasets). In particular, the domain of QSAR modeling seems to be suitable for SSL-RF, since SSL-RF improve over CLUS-RF for almost all different amounts of labeled data on all four QSAR datasets considered.

The results suggest that the proposed semi-supervised algorithms may be especially suitable for QSAR modeling, since on all four datasets from this domain, either SSL-PCT or SSL-RF (or even both methods) improve over their supervised counterparts. This observation is appealing, since the domain of QSAR modeling is particularly suitable for semi-supervised learning: Determining the biological activity of chemical compounds is a very expensive and tedious process (DiMasi et al., 2003), while descriptions of the structure of hundreds of thousands of unlabeled compounds are freely available in public databases, such as ChEMBL (Bento et al., 2014).

8.2.2 Statistical analysis of the predictive performance

The results of the statistical analysis are presented in Table 8.4.

On binary classification datasets, statistical analysis shows that SSL-PCTs are better than the SL-PCTs for all amounts of labeled data, while statistical significance is achieved for the smallest amount of labeled data considered (i.e., 25 labeled examples) and when 200 or more of labeled examples are available. Similarly, SSL-RF are better than CLUS-RF for all amounts of labeled data, and statistically significantly better for 100 and 500 of labeled examples.

In the case of multi-class classification, SSL-PCTs are better than the SL-PCTs for all amounts of labeled data, while statistical significance is achieved for 100 and 200 labeled examples. SSL-RFs are statistically significantly better than CLUS-RF for all amounts of labeled data, except the smallest amount, i.e., 25 labeled examples.

Semi-supervised PCTs are the most effective for small amounts of labeled data: They achieve statistically significantly better performance than supervised PCTs up to 100 of labeled examples. Similar observation is made for semi-supervised random forests: They achieve statistically significantly better performance than supervised random forests for

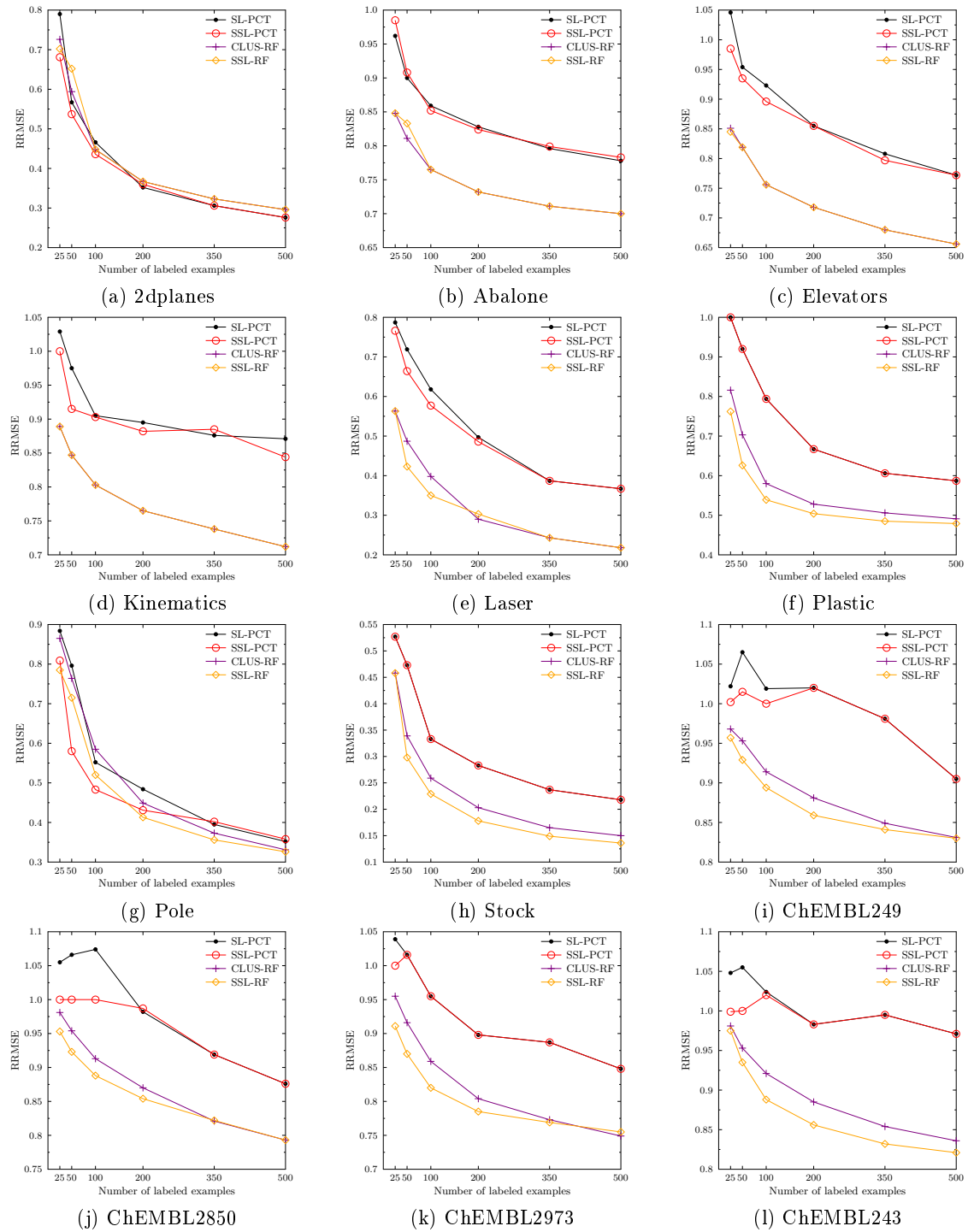


Figure 8.3: RRMSE of the supervised and semi-supervised methods on the regression datasets.

amounts of labeled data ranging from 25 to 350 examples, while statistical significance is not achieved for 500 of labeled examples.

Table 8.4: P -values of the Wilcoxon signed-rank test applied to the performances of the supervised and semi-supervised algorithms. In bold, we report significant p -values (< 0.05). In comparison of two methods, i.e., Method1 vs. Method2, the ‘-’ sign denotes that the sum of ranks where the first method outperformed the second is higher than the sum of ranks where the second method outperformed the first. The ‘+’ sign denotes the opposite.

Methods		Number of labeled examples					
		25	50	100	200	350	500
BINARY CLASSIFICATION							
SL-PCT	vs. SSL-PCT	0.009 (+)	0.388 (+)	0.066 (+)	0.005 (+)	0.019 (+)	0.019 (+)
CLUS-RF	vs. SSL-RF	0.529 (+)	0.192 (+)	0.002 (+)	0.099 (+)	0.093 (+)	0.012 (+)
MULTI-CLASS CLASSIFICATION							
SL-PCT	vs. SSL-PCT	0.248 (+)	0.084 (+)	0.014 (+)	0.007 (+)	0.192 (+)	0.081 (+)
CLUS-RF	vs. SSL-RF	0.563 (+)	0.011 (+)	0.011 (+)	0.003 (+)	0.004 (+)	0.02 (+)
REGRESSION							
SL-PCT	vs. SSL-PCT	0.011 (+)	0.010 (+)	0.004 (+)	0.367 (+)	0.480 (+)	0.583 (+)
CLUS-RF	vs. SSL-RF	0.008 (+)	0.065 (+)	0.008 (+)	0.023 (+)	0.034 (+)	0.126 (+)

8.2.3 Interpretability and size of the trees

The models produced by the semi-supervised algorithm are readily interpretable, since they are decision trees. Here, we present examples of decision trees obtained on one binary classification dataset, and one multi-class classification dataset. We discuss the differences between the trees generated by the supervised and the semi-supervised learning algorithms.

From binary classification, we present the trees obtained on the Diabetes dataset (Lichman, 2013) with 100 labeled examples (Figures 8.4a and 8.4b). The task at this dataset is to classify subjects to tested positive for diabetes (class value 1) and tested negative for diabetes (class value 0), on the basis of 8 descriptive attributes: number of times the subject was pregnant, plasma glucose concentration in an oral glucose tolerance test, diastolic blood pressure, triceps skin fold thickness, 2-hour serum insulin, body mass index (BMI), diabetes pedigree function (takes into account ancestor’s diabetes history), and age. We can observe that the semi-supervised tree (Fig. 8.4b) is slightly smaller than the supervised tree (Fig. 8.4a), making it easier to interpret. In spite of its smaller size, the semi-supervised tree is more accurate than the supervised one, meaning that the unlabeled examples contributed to make better splits in the tree. Both trees chose the same split at the root node of the tree: Patients with $BMI \leq 29.85$ are classified as tested negative for diabetes, whereas patients with $BMI > 29.85$ are further divided on the basis of other descriptive attributes into positive and negative class. Obesity is known to contribute to increased risk of diabetes, where subjects with $BMI > 30$ are considered obese (Ford, 1999). Interestingly, this coincides with the split in the top node of both decision trees. In the lower levels of decision trees, semi-supervised and supervised trees mainly choose different attributes, or different splitting points on the same attribute.

From multi-class classification, we present the trees obtained on the Cardiotocography3 dataset (Lichman, 2013) with 50 labeled examples (Figures 8.4c and 8.4d). This dataset is about classifying the state of the fetus on the basis of cardiotocograms (CTGs) into 3 classes: Normal, Suspect and Pathologic. CTGs are records of measurements of fetal heart-beat and the uterine contractions during pregnancy. From CTGs, 35 features were generated: Various features describing the histogram of foetal heart rate values (Min, Mode,

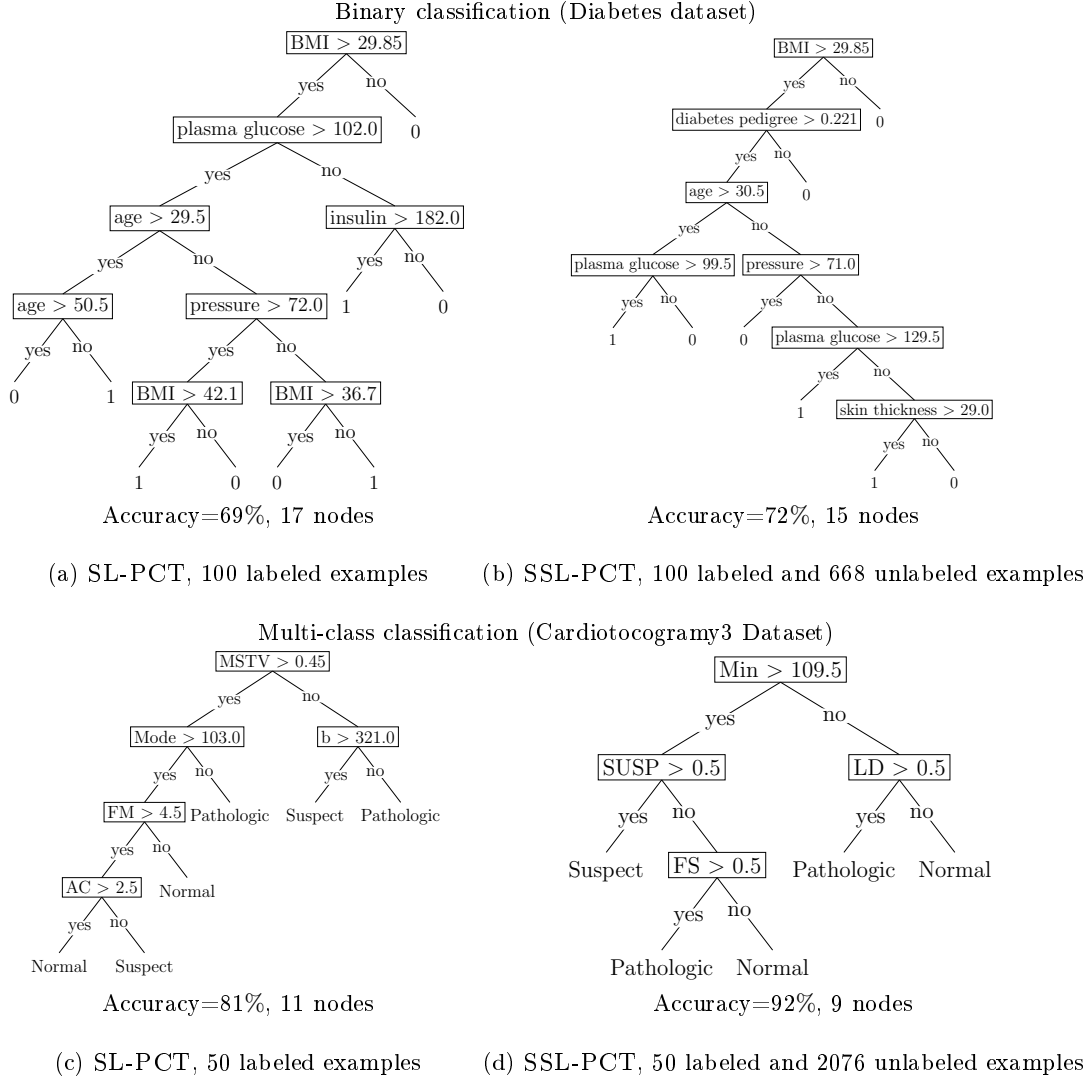


Figure 8.4: Examples of trees obtained with the supervised SL-PCT algorithm (a and c), and the semi-supervised SSL-PCT algorithm (b and d) on the Diabetes and Cardiotocogramy3 datasets.

Median etc.), mean value of short term (beat-to-beat) variability (MSTV), heartbeat accelerations (AC), foetal movement (FM), largely decelerative pattern (LD), flat-sinusoidal pattern (FS), etc. We can observe that the tree learned with the semi-supervised algorithm (Fig. 8.4d) is slightly smaller than the one learned with the supervised algorithm, (Fig. 8.4c), and that the two trees have very different structure (i.e., completely different attributes were chosen). The tree learned with the semi-supervised algorithm is much more accurate, meaning that it was able to find better splits.

Table 8.5 presents the model sizes of supervised and semi-supervised trees. We can observe that semi-supervised classification trees are, almost without exception, smaller than the supervised classification trees. Recall that, because of the way the impurity measure is defined, semi-supervised trees group examples into clusters that are compact both in the descriptive and the label space, while the supervised trees consider only the label space. Obviously, semi-supervised trees have a more strict clustering criterion, which likely is a reason for the smaller tree size. On average, semi-supervised are 25% smaller on

Table 8.5: Model sizes, in terms of number of nodes, obtained with the supervised algorithm (SL-PCT) and the semi-supervised algorithm (SSL-PCT).

Dataset	Number of labeled examples											
	25		50		100		200		350		500	
	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT
BINARY CLASSIFICATION												
Abalone	6.4	4.6	11	4	17.2	9.4	33.4	20	59	39	75.6	52.4
Adult	5.2	2.8	7.6	3.6	16.4	4.8	25.8	6.4	36.8	6.2	54	37.4
Bank	3.2	1	6	1.4	8.4	1	15.2	10.6	25.4	17.4	39.6	3.6
Banknote	5	5	7.6	4.6	11.2	9.2	15.8	13.2	22	18.2	22.6	21.6
Biodegradation	6.4	4.6	10.2	6.6	18.4	14.4	29.6	26.4	48.4	45.2	66.4	64.6
Diabetes	5.4	5.4	10.2	8.4	20.6	14	34.4	31.2	63.6	60	82	92
Eyestate	8.6	8.6	12	11.8	24.6	20.6	49.2	36.8	83.2	63.6	111.2	94.6
Madelon	5.6	6.2	9.2	10	18.2	17	34.2	20.4	60	34.6	82.2	40.8
Mushroom	3.2	3	3.2	3.2	3.4	3.4	4.4	4.4	6.6	6.6	8.4	8.4
Pgp	6.6	6.2	11.8	4.4	19.4	15.4	35.8	29.8	57.8	53.6	74.4	71.2
Phishing	4	3.6	6.8	3.8	8.8	8	13.2	11.8	20.2	20.2	28.8	28.8
Thyroid	2.2	1	2.6	3	4.4	4.4	5.8	5.8	8.6	8.6	9.8	9.8
Average:	5.2	4.3	8.2	5.4	14.3	10.1	24.7	18.1	41.0	31.1	54.6	43.8
MULTI-CLASS CLASSIFICATION												
Baseball	3.2	1	4.4	1.8	8	6.4	12	10.8	19	17.2	24	19.8
Cardiotocography3	3.6	2.2	6.4	6.8	9.6	11	10.2	10.6	12.2	9.8	14.8	14.8
Cardiotocography10	11.6	11.6	15.2	15.2	17.4	18.4	19	20	19	19.4	19	19
Cmc	10.2	10.2	17.8	11.4	31.4	17.8	67	37.4	101.2	68.4	129.8	84.4
Gasdrift	10.6	10.4	15	15	25	22.2	40	40	55.8	56.4	70.8	70.8
Gasdriftdc	10.6	9.6	15	15	24.4	21.6	39.4	39.4	54.4	51.6	70.2	70.2
GesturePhase	10.8	9.8	21	16.2	38	6.8	71.8	66.8	123.2	120.4	172.6	169.8
Imagesegment	12.4	11.2	14.6	14.6	18.4	18.4	23.6	23.6	32.4	31.2	39.2	38
MiceProtein	12.8	12.8	14.8	14.8	15	15	15	15	15	15	15	15
Optdigits	14.8	9.2	20.2	19.2	30.8	31.4	51	46.6	76.4	69.2	92.8	87.6
Pageblocks	3	1.4	4.8	4.8	8.2	8.2	12	9.6	18	18	21.4	18.4
Pendigits	14.8	12.4	20.8	20	28.4	32	43.4	42.4	63.2	55.6	74	72.2
Average:	9.9	8.5	14.2	12.9	21.2	17.4	33.7	30.2	49.2	44.4	62.0	56.7
REGRESSION												
2dplanes	4.4	10	9.6	18.8	17.8	32.2	32.8	47.2	50	50	61.4	61.4
Abalone	4	2	5.2	2.8	7.8	5.4	14	8	20.8	11.4	27.6	15.2
Elevators	4	1.6	4.6	2.8	13.6	4.8	19	19	29.4	10.2	36.8	15
Kinematics	3.2	1	5.6	3.2	5.8	5.4	10.4	7.8	18.8	22.2	29.8	17
Laser	5.4	6.4	11	11.4	21.2	16.6	39.4	23	87.4	87.4	106.2	106.2
Plastic	2.4	1	6.2	6.2	13	13	22.2	22.2	35.2	35.2	43	43
Pole	5.6	10	8.2	19.8	14.6	31.4	28.2	61.2	45.4	75.4	60.4	102
Stock	7.4	7.4	12.4	12.4	26	26	47.4	47.4	74.8	74.8	104	104
ChEMBL249	1.8	1.4	4.4	2	7	1	12.2	12.2	26.8	26.8	31.4	31.4
ChEMBL2850	2.8	1	4	1	12.4	1	19.8	2.2	35.6	35.6	45.4	45.4
ChEMBL2973	2.6	1	3.8	3.8	6.6	6.6	16.4	16.4	22.4	22.4	37.4	37.4
ChEMBL243	2.4	1.2	3.2	1	5	4.4	11.6	11.6	21.6	21.6	28.8	28.8
Average:	3.8	3.7	6.5	7.1	12.6	12.3	22.8	23.2	39.0	39.4	51.0	50.6

binary classification datasets, and 12% smaller on multi-class classification datasets. Semi-supervised regression trees are, on average, of a very similar size to the one of supervised regression trees.

Furthermore, Table 8.5 demonstrate that in situations with limited availability of labeled data, semi-supervised learning of PCTs may offer better interpretability (i.e., smaller trees) and frequently also more accurate trees as compared to supervised learning of PCTs (Figs. 8.1 and 8.2). For example, on the Diabetes dataset (Fig. 8.4a and b), the algorithm for learning supervised PCTs needs approximately 500 of labeled examples to achieve pre-

dictive performance comparable to the semi-supervised approach with only 100 of labeled examples (Fig. 8.1). Consequently, this can lead to very large, and hence difficult to analyze, supervised trees.

8.3 Summary of the Results

In this section, we summarize the findings from the empirical evaluation of the proposed algorithm for the semi-supervised learning of predictive clustering trees on machine learning tasks with primitive output space. In our experiments we investigate whether the proposed semi-supervised predictive clustering trees, and ensembles thereof, can improve the performance of supervised predictive clustering trees and random forests. To this end, we performed the empirical evaluation over a wide range of datasets. In particular, we used 12 datasets for the task of binary classification, 12 datasets for the task of multi-class classification and 12 datasets for the task of regression.

The main findings are similar to the ones made on machine learning tasks with structured outputs (Chapter 7): Semi-supervised predictive clustering trees improved the performance of supervised trees on most of the datasets for the three machine learning tasks considered (i.e., binary classification, multi-class classification and regression), while very seldom performing worse than supervised trees.

However, there are some notable differences between findings made for structured outputs and primitive outputs. Semi-supervised random forests were not very successful in improving supervised random forests when applied to datasets with structured outputs. On the other hand, on datasets with primitive outputs, semi-supervised random forests are even more successful than semi-supervised trees (in improving their supervised counterparts). Next, on datasets with structured output, semi-supervised trees were in general somewhat larger than supervised trees, while on datasets with primitive outputs different observations are made: Semi-supervised trees are almost without exception smaller than the supervised trees on datasets with discrete outputs (i.e., binary and multi-class classification), meaning that semi-supervised offer better interpretability. On regression datasets, semi-supervised trees and supervised trees are generally of a very similar size. Investigating the reasons behind the observed differences deserve more attention in the future.

All in all, the empirical analysis performed in this Chapter suggests that, even though semi-supervised predictive clustering trees and ensembles thereof were designed for structured output spaces, they can offer better performance than supervised trees and random forests also on machine learning tasks prevalent in the literature: binary classification, multi-class classification and regression.

Chapter 9

Case Studies

In this chapter, we present two case studies that use semi-supervised methods for structured output prediction proposed in this thesis. The studies come from two domains: quantitative structure-activity relationship (QSAR) modeling and ecology.

The first case study deals predicting cytostatic activities of potential antimalarial medications on several human cell lines (Levatić, Supek, & Džeroski, 2015). For this purpose we use the method proposed in Chapter 6: self-training method for multi-target regression based on the tree ensembles. The findings of this study are two fold. First, we show that it is possible to improve QSAR models with the proposed method, as compared to state-of-the-art method - random forests for multi-target regression. Second, we demonstrate, for the first time, that it is possible to exploit unlabeled data from public databases of bioactive drug-like molecules to improve QSAR models.

The second case study comes from the domain of ecology. The goal is to predict the quality of water samples, which is described with three continuous variables: biological water quality, phosphorus concentration and nitrate concentration. Therefore, the task there is multi-target regression. The dataset at hand contains partially labeled examples, for approximately 50% of the dataset, the value of some of the target variables is unknown. Since most of the MTR methods cannot handle partially labeled data, the researchers are often forced to discard such (partially labeled) examples. In our experiments, we explore whether this undesired solution can be avoided. To this end, we use semi-supervised trees for MTR proposed in Chapter 7, which can exploit partially labeled data in the learning process. Our results suggest that better performance can be achieved if partially labeled data are exploited by semi-supervised predictive clustering trees, rather than discarded.

9.1 Improving QSAR Models by Exploiting Unlabeled Data from Public Databases

Pharmaceutical industry spends huge amounts of money on research and development of new drugs. The estimated cost of new drug development is over 800 millions of US\$ (DiMasi et al., 2003). This cost is to a large extent related to failures: Only a small fraction of promising drugs will ever progress to clinical trials due to the undesired biological effects.

Establishing a connection between biological effects and structural and/or physico-chemical properties of chemicals is the task of quantitative structure-activity relationship or QSAR modelling. Formal studies of such relationships are the basis for the development of predictive models that can provide insight into the biological activity of a molecule without the need to synthesize it. This leads to a number of benefits, including savings in the cost and duration of product development (e.g., in the pharmaceutical or pesticide

industries), reduction of the need for animal testing, and prediction of unwelcome or toxic environmental impact (Cronin, 2010).

Most commonly, to build predictive QSAR models, supervised machine learning methods, which use only labeled data, are used. However, QSAR modeling is particularly attractive for semi-supervised learning since the premise: “labeled data are scarce, while unlabeled data abound” is generally satisfied in this domain. Public databases with (hundreds of) thousands of chemical compounds are available (such as ChEMBL or ZINC databases), while labeled datasets sizes typically range from tens to hundreds and rarely surpass a thousand molecules.

The main goal of this study is to investigate whether we can improve predictive performance of the QSAR models by exploiting such public sources of unlabeled data. We predict the *in vitro* cytostatic activity against 12 human tumor cell lines of a small dataset of primaquine-like molecules. Primaquine is a medication used in the treatment and prophylaxis of malaria. To build QSAR models, we use the semi-supervised method based on ensembles of predictive clustering trees for multi-target regression (MTR) (Levatić et al., 2015a, 2017a), presented in Chapter 6.

MTR is a structured output prediction task where the goal is to simultaneously predict multiple continuous target variables (e.g., cytostatic activity against different cell lines, as considered in this study). Semi-supervised methods able to solve MTR problems are scarce. Most commonly, SSL methods for structured output prediction are dealing with discrete output. On the other hand, SSL for single-target regression has received more attention in the past (Zhu, 2008). While it is possible to decompose a MTR problem into several single-target ones and use such methods, there are several advantages of learning a global multi-target model over learning a separate local model for each target variable: global models are typically easier to interpret, may perform better and overfit less than a collection of single-target models (Kocev et al., 2013; Levatić, Kocev, & Džeroski, 2014).

It was suggested before that SSL can improve QSAR models (Guo-Zheng, Jack, Wen-Cong, Dan, & Mary, 2008; Levatić, Džeroski, Supek, & Šmuc, 2013), however to the best of our knowledge, public databases of compounds were not considered as sources of unlabeled data for SSL. In this study, we extracted unlabeled data for SSL from ChEMBL database, which contains more than a million of small bioactive drug-like molecules. Our results show that the semi-supervised QSAR model outperformed the supervised one, suggesting that public databases of compounds can be used to improve (at no additional cost) the traditional supervised QSAR models.

9.1.1 Experimental design

9.1.1.1 Data description

The labeled dataset we consider in this study consists of 40 primaquine-like molecules (Džimbeg et al., 2008; Šimunović et al., 2009; Rajić et al., 2010; Perković et al., 2012; Pavić et al., 2014) which are currently under consideration as novel antimalarial medications. For these chemicals, 12 different properties were measured: (i) cytostatic activity against 8 different human tumor cell lines, (ii) free radical scavenging ability (interaction with DPPH), (iii) inhibition of the soybean lipoxygenase enzyme and (iv) inhibition of lipid peroxidation. The goal of QSAR models we develop in this study is to predict, as accurately as possible, these 12 properties of primaquine-like molecules.

The molecules were represented by 127 2D molecular descriptors calculated with Chemistry Development Kit (Steinbeck et al., 2003), which are used as descriptive attributes during the training of machine learning methods.

9.1.1.2 Extraction of unlabeled data from the ChEMBL database

As a source of unlabeled data for semi-supervised learning, we consider the ChEMBL database (Bento et al., 2014). This database contains almost 1.5 million of bioactive drug-like small molecules, and is one of the biggest and most complete such databases.

To obtain unlabeled molecules, we search the ChEMBL database for molecules similar to the ones in our dataset (i.e., we search for primaquine-like molecules within ChEMBL database). For each of the 40 molecules in the labeled dataset, we extract from ChEMBL all of the molecules with at least 80% similarity according to the Tanimoto similarity measure (Nikolova & Jaworska, 2003) over molecular fingerprints, ensuring that the extracted molecules are within the applicability domain of the model. From the obtained list of molecules, we remove duplicates and the ones which have identical counterpart in the labeled dataset. This procedure yielded a dataset of 61 molecules, which were fed as unlabeled data to the semi-supervised algorithm.

9.1.1.3 Experimental setup and evaluation procedure

The SSL method for MTR used this case study (CLUS-ST) iteratively trains random forest ensembles for MTR (Chapter 6). Hence, we compare the predictive performance of the self-training approach to the performance of a supervised random forest (CLUS-RF) – a baseline in the experimental evaluation. In all of the experiments, we construct random forests consisting of 100 predictive clustering trees for MTR. The trees are not pruned and the number of random features at each internal node is set to $\lfloor \log_2(D) + 1 \rfloor$, where D is the total number of features (Breiman, 2001). To assess the reliability of predictions we use variance of the votes of an ensemble (see Chapter 6, Section 6.1.1.1). More specifically, out of the four variants of the variance scores, we use the *VarianceNormAvg* score, which uses min-max normalization to normalize the per-target scores, and averaging to aggregate the per-target scores.

To evaluate the predictive performance of the models, we use a 10-fold cross validation procedure. The CLUS-RF is trained by using labeled data only, while CLUS-ST in addition uses the unlabeled data extracted from the ChEMBL database. We assess the predictive performance of the algorithms by using the relative root-mean-square-error (RRMSE), defined as follows:

$$RRMSE = \frac{1}{T} \sum_{i=1}^T \sqrt{\frac{\sum_{j=1}^{N_{test}} (y_{j,i} - \hat{y}_{j,i})^2}{\sum_{j=1}^{N_{test}} (y_{j,i} - \bar{y}_i)^2}},$$

where T is the number of target variables, $y_{j,i}$ is the value of the i^{th} target variable for the j^{th} example of the test set, $\hat{y}_{j,i}$ is prediction of the j^{th} example, N_{test} is the number of examples in the test set, and $\bar{y}_i$ is the mean value (on the training set) of the j^{th} target variable.

For the CLUS-ST algorithm, a threshold τ on the reliability score needs to be set. The threshold is then used throughout the iterations of the self-training. To determine the optimal threshold, we cross-validate the CLUS-ST algorithm on labeled part of the data with 15 different thresholds: $\tau = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99\}$. The threshold $\tau = 0.55$ was selected as an optimal one (i.e., it yielded the best predictive performance).

9.1.2 Results

The results of experimental evaluation are presented on Figure 9.1. We can observe that CLUS-ST outperforms CLUS-RF on 9 out of 12 target variables. Furthermore, overall

RRMSE error (i.e., across all 12 target variables) of CLUS-ST is lower than the error of CLUS-RF.

On the three target variables, *MCF7*, *DPPH20min* and *LoxInhib*, semi-supervised learning did not improve the performance of CLUS-RF. On the latter two target variables (*DPPH20min* and *LoxInhib*), the predictive performance of CLUS-ST and CLUS-RF is very similar, however on the former (*MCF7*), the usage of unlabeled data by CLUS-ST led to a degradation in performance of CLUS-RF.

Self-training can also degrade the performance of the underlying method, especially if a sub-optimal threshold is chosen. To exemplify this, we present the results with varying values of the reliability threshold on Figure 9.2. In particular, if a too permissive threshold is selected, it can allow wrongly predicted examples to enter in the training set. A prediction error made in the earliest iterations can reinforce itself in the next iterations, leading to degradation of performance. On the other hand, if a too stringent threshold is set, it is possible that none, or very few, of the unlabeled examples enter the training set, meaning that we miss the opportunity to improve performance by using unlabeled data.

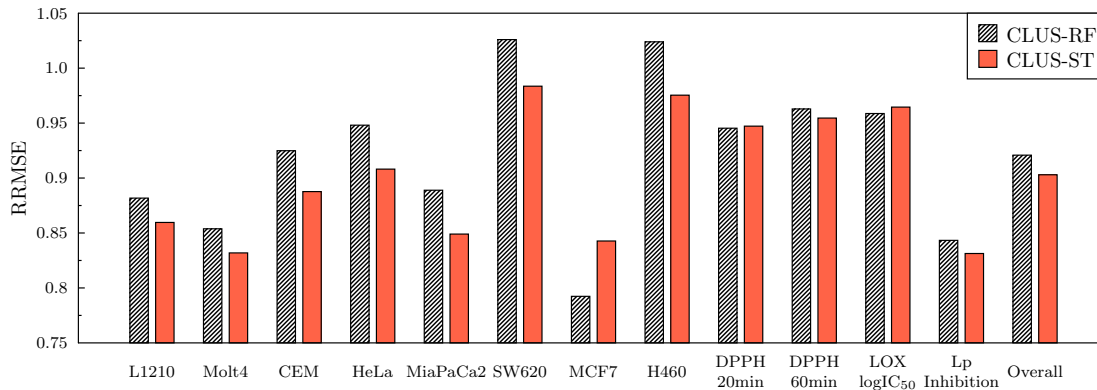


Figure 9.1: Comparison of predictive performance for random forests (CLUS-RF) and semi-supervised self-training (CLUS-ST).

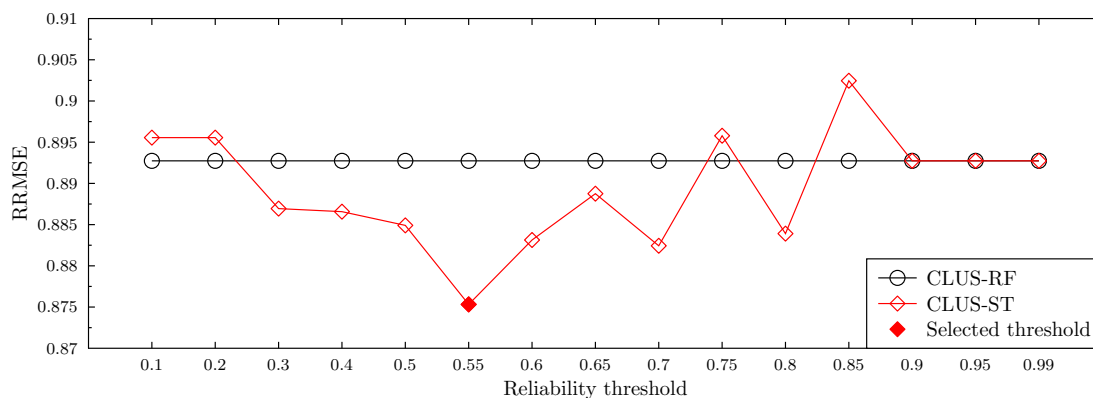


Figure 9.2: Comparison of predictive performance of random forest (CLUS-RF) and semi-supervised self-training (CLUS-ST) with varying threshold for reliability of predictions exemplifies the possible effects of choosing a sub-optimal threshold.

9.1.3 Conclusions

In this case study, we investigated whether predictive performance of QSAR models can be improved for 'free'. Namely, we propose the usage of public databases of drug-like molecules as a source of unlabeled data for QSAR modeling with semi-supervised learning algorithms. We develop a QSAR model for predicting 12 important properties of primaquine-like molecules – chemicals under consideration as potential antimalarial medications. For this purpose, we use semi-supervised method for multi-target regression based on ensembles of predictive clustering trees. We extract a set of unlabeled molecules from ChEMBL database, which are used in addition to labeled data by the semi-supervised method. Experimental evaluation showed that the semi-supervised QSAR model achieved better predictive performance than the model which relies only on labeled data. This result suggest that public databases of chemical compounds are useful source of unlabeled data for QSAR modeling with semi-supervised learning.

9.2 Handling Partially Labeled Data in Predicting Water Quality of Agricultural Fields in Ireland

It was recognized that agriculture has an impact on the surface and ground water quality in Ireland by the means of transfer of nutrients to water (Schulte, Richards, Kurz, McDonald, & Holden, 2006). The (rate of) transfer depends on various agro-meteorological factors, such as environmental pressure (soil mineralization, drought and grass growing season) and pathway attributes (soil drainage, net rainfall and rainfall intensity). In this case study, we propose to predict the quality of water in agriculture fields in Ireland on the basis of these factors. To this end, we use predictive clustering trees (Blockeel, 1998). The water quality is described by the three numeric variables: biological quality of water, phosphorus and nitrate concentrations. Thus, we are dealing with the task of predicting multiple continuous variable, i.e., multi-target regression (Section 3.1.1).

The dataset at hand contains partially labeled examples (Fig. 4.1). Namely, approximately 50% of the examples are partially labeled, i.e., values of some target variables for these examples are not known. The majority of methods for MTR cannot handle partially labeled data. Therefore, partially labeled examples have to be discarded when applying such methods. In this case study, this would mean that almost half of the dataset cannot be used. Thus, a lot of potentially useful information would be lost.

Our aim is to explore whether this undesired solution (i.e., discarding partially labeled examples) can be avoided. To this end, we use semi-supervised predictive clustering trees for MTR which can handle partially labeled data directly (Chapter 7). We compare their predictive performance to the performance of standard supervised PCTs, which cannot handle partially labeled data. In our experiments, we use single PCTs for MTR (Struyf & Džeroski, 2006) and ensembles of PCTs for MTR (Kocev et al., 2013). We build both multi-target (i.e., global) models, where all of the target variables are predicted by a single model, and single target (i.e., local) models, where a separate model is built for each target variable.

9.2.1 Experimental design

9.2.1.1 Description of the dataset

The dataset was provided by the Spatial Analysis Group at the TEAGASC Institute in Ireland, collected within a study performed by Schulte et al. (2006). The country has been divided into grid cells of 10x10km (excluding Northern Ireland). The dataset consists of

708 observations points described with 27 numeric descriptive attributes and 3 numeric target variables which we are trying to predict. Namely, *biological water quality* (Q), *phosphorus-concentration* (P) and nitrate concentration (N). All target and descriptive variables are expressed as average values of measurements recorded within each grid cell.

All of the three target variables were measured at 351 observation points, while values of some of the target variables are missing for the rest of 357 observation points. Thus there are 351 labeled examples and 357 partially labeled examples in the dataset. As shown on Figure 9.3, Q was measured at 706 observation points, P at 529, and the N at 352 points. Note that, at least one target variable was measured at every observation point, so the dataset does not contain (fully) unlabeled examples.

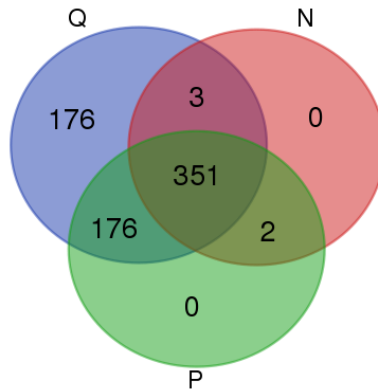


Figure 9.3: Venn diagram of observation points with know values for the three target variables. There are 351 examples with all three targets known, 176 examples with know values for P and Q , but not for N , and 176 examples with know values for Q , but not for P and N .

The 27 descriptive attributes are divided into two groups, pathway variables and environmental pressures. We briefly describe attributes from each group.

- **Pathway variables**

1. *Drainage factor*: Average infiltration capacity of the soil, ranging from 0 (entire grid cell is poorly-drained) to 1 (entire grid cell is well-drained).
2. *Net rainfall*: Rainfall minus evapotranspiration.
3. *High drain* $\{q1, q2, q3, q4, sum\}$: The number of intense drainage events ($>15\text{mm}$ per day) in the 1st, 2nd, 3rd, 4th quarter of the year, or the total number of intense drainage events per year (sum).
4. *Trafficable days*: The number of days on which the soil conditions allow traffic by animals and/or machinery.

- **Environmental pressures**

1. *SMDmax*: The maximum soil moisture deficit during summer.
2. *Growth season*: The duration of the growth season in days.
3. *N cℓs / ha farmed*: The input of nitrate (in kilos) from the excreta of cattle and sheep, expressed as an average per hectare farmed within the grid cell.
4. *N pℓp / ha farmed*: The same as 3., but for pigs and poultry.

5. *N fert / ha farmed*: The same as 3., but the input of nitrate comes from the fertilizer applied.
6. *{P c&s, P p&p, P fert} / ha farmed*: The same as 3., 4. and 5., respectively, but the input is phosphorus.
7. *Sum N-input / ha farmed*: Total input of nitrate per hectare farmed (N c&s + N p&p + N fert / ha farmed)
8. *Sum P-input / ha farmed*: Total input of phosphorus per hectare farmed (P c&s + P p&p + P fert / ha farmed)

9.2.2 Experimental setup

In this study, we compare the predictive performance of models learned from the whole dataset of 708 examples (i.e., 351 labeled and 357 partially labeled examples), to the performance of models learned only from the 351 labeled examples. We built four types of single tree models:

- *single-target* PCTs (PL-ST-PCT) learned from all the examples which have known value for the given target variable, namely 706 examples for the Q variable, from 529 for the P , and from 352 for the N .
- *multi-target* PCTs (PL-MT-PCT) learned from the whole dataset.
- *single-target* PCTs (FL-ST-PCT) learned only from the 351 labeled examples.
- *multi-target* PCTs (FL-MT-PCT) learned only from the 351 labeled examples.

Similarly, we built four types of random forest models, denoted as follows: PL-ST-RF, PL-MT-RF, FL-ST-RF and FL-MT-RF.

We evaluate the models by 10 fold cross validation. More specifically, the whole dataset (including partially labeled examples) is first randomly split into 10 folds. Next, 9 folds are used for training, and the remaining one for testing. The procedure is repeated 10 times so that each fold is used exactly once as test set. The reported results represent an average of all 10 runs. When building the models learned only from fully labeled data (i.e., FT-ST-PCT, FT-MT-PCT, FT-ST-RF and FT-MT-RF), examples which have missing value for any of the 3 target variables are removed from the training folds. Next, when learning a single-target model from partially labeled data (i.e., PL-ST-PCT and PL-ST-RF) for a given target variable, examples which have missing value for that target are removed. Finally, multi-target models learned from partially labeled data use all of the examples in the training folds.

Single tree models are pruned with the procedure used in M5P regression trees (Quinlan, 1993). We construct random forests consisting of 100 predictive clustering trees for MTR. The trees in the random forest are not pruned, and the number of randomly selected features at each internal node is set to $\lfloor \log_2(D) + 1 \rfloor$, where D is the total number of attributes (Breiman, 2001). We assess the predictive performance of the algorithms by using the relative root-mean-square-error (RRMSE), defined as follows:

$$RRMSE = \frac{1}{T} \sum_{i=1}^T \sqrt{\frac{\sum_{j=1}^{N_{test}} (y_{j,i} - \hat{y}_{j,i})^2}{\sum_{j=1}^{N_{test}} (y_{j,i} - \bar{y}_i)^2}},$$

where T is the number of target variables, $y_{j,i}$ is the value of the i^{th} target variable for the j^{th} example of the test set, $\hat{y}_{j,i}$ is prediction of the j^{th} example, N_{test} is the number

of examples in the test set, and $\bar{y}_i$ is the mean value (on the training set) of the j^{th} target variable. Note that, to ensure that the figures are comparable across different experimental settings $\bar{y}_i$ is always taken on the fully labeled part of the dataset.

9.3 Results

The results of the experimental evaluation are presented in Figure 9.4a (single trees) and Figure 9.4b (random forests). From the results, we can observe that the models which exploit partially labeled examples (i.e., PL-ST and PL-MT) achieve better predictive performance than the models use only labeled examples (i.e., FL-ST and FL-MT). This observation is made both for local and global single trees and ensembles, though it is more pronounced for ensembles.

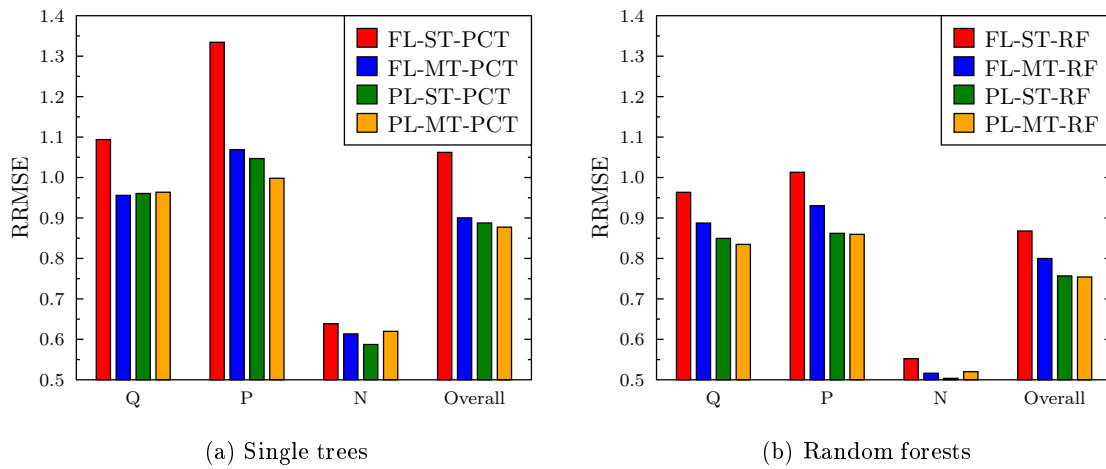


Figure 9.4: Predictive performance of the local and global models build from labeled and partially labeled data.

Partially labeled multi-target (i.e., global) models (PL-MT) achieve similar predictive performance to the partially labeled single target (i.e., local) models (PL-ST). However, global models are computationally more efficient, since only one model needs to be built, instead of one model for each target variable (as is the case with local models). Next, global models are typically easier to interpret and overfit less than local models (Kocev et al., 2009; Kocev et al., 2013; Levatić, Kocev, & Džeroski, 2014).

When the models learned from fully labeled data are compared, we can observe that fully labeled multi-target (i.e., global) models (FL-MT) achieves better predictive performance than the fully labeled single target (i.e., local) (FL-ST) models, both for single trees and random forests.

We can observe different trends in the overall predictive performance, and the per-target performances. Namely, the performance for the Q and P target variables follows a similar trend to that of the overall performance, while the performance for the target N has different trend, where all partially labeled single target model (i.e., PL-ST) achieves the best performance.

Next, we analyze the size of the (single tree) models (Table 9.1) and their interpretability. We can observe that the local models (FL-ST-PCT and PL-ST-PCT) have much larger size than the global models, making them more difficult to interpret. Furthermore, models learned from partially labeled data are somewhat larger than the models learned only from

	FL-ST-PCT	FL-MT-PCT	PL-ST-PCT	PL-MT-PCT
Model size	65	15	139	27

Table 9.1: The sizes of single tree models in terms of number of nodes. For single-target models (i.e., FL-ST-PCT and PL-ST-PCT), the size represents the sum of sizes of trees for all of the three target variables.

labeled data, which is expected since the former models are learned by using more data. The multi-target single-tree model PL-MT-PCT has 27 nodes, which is a reasonably large model which can be easily inspected by the experts.

Figure 9.5 presents the PL-MT-PCT model. The attribute appearing in the top node is the number of days on which the soil conditions allow traffic by animals and/or machinery. It has been shown that trafficking of soils when soil moisture deficits $< 10\text{mm}$ greatly increase the risk of nutrient transfer to water (Schulte et al., 2006). Next, the maximum soil moisture deficit (SMD_{max}) has been shown to be related to loss of N (Schulte et al., 2006). Indeed, this relation was captured by the left branch of the tree: Branches of the tree with high SMD_{max} have the by far the highest N concentration in their leaves. The attributes that are also frequently appearing in the tree are contributors of P and N from the excreta of animals and fertilizers.

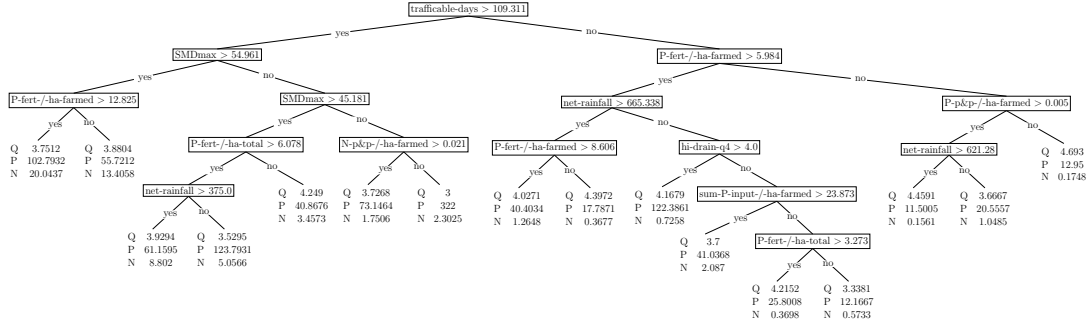


Figure 9.5: Multi-target tree learned from labeled and partially labeled examples.

9.4 Conclusions

Multi-target regression is a structured output prediction task where multiple continuous target variables are simultaneously predicted. In many practical applications it can happen that, some parts of the (structured) output are missing for some examples. That is, the examples can be only partially labeled. However, most machine learning methods for multi-target regression cannot handle such data, thus the practitioners of machine learning have to discard partially labeled data, or resort to single-target methods forcing them to lose the benefits of global models.

In this case study, we demonstrate the ability of the semi-supervised predictive clustering trees proposed in this thesis (Chapter 7) to efficiently deal with partially labeled data. We apply SSL-PCTs to predict three numerical variables related to the water quality in agricultural fields of Ireland. Approximately 50% of this dataset is partially labeled.

We compared the performance of predictive clustering trees, which are able to exploit partially labeled data, to the performance of standard supervised predictive clustering trees which can use only fully labeled data. We considered single trees and random forests, as

well as local and global models. The results of the experimental evaluation showed that better predictive performance can be achieved if partially labeled data are exploited by semi-supervised predictive clustering trees, rather than discarded.

Chapter 10

Conclusions

This thesis deals with semi-supervised learning for structured output prediction. It presents an extension of the predictive clustering framework to the semi-supervised learning task. In particular, we propose two classes of semi-supervised methods for structured output prediction stemming from this framework. Empirical evaluation showed that the proposed methods outperform their supervised counterparts on a number of datasets from different domains and with different types of structured outputs.

In the remainder of this chapter, we first summarize the scientific contributions of this thesis. We then discuss the findings of the thesis with respect to the goals and hypotheses set out in Chapter 1. Finally, we present several possible directions for future work.

10.1 Contributions

The work presented in this thesis contributes to the field of semi-supervised learning with two classes of global semi-supervised methods for structured output prediction: self-training for multi-target regression (Levatić et al., 2015a, 2017a) and semi-supervised predictive clustering trees (Levatić et al., 2017b; Levatić, Kocev, Ceci, & Džeroski, 2017).

The first class of methods is based on the **self-training paradigm** - it uses its own most reliable predictions in the learning process. We propose a self-training method for multi-target regression based on ensembles of predictive clustering trees. To the best of our knowledge, this is currently one of the very few general purpose semi-supervised methods for this type of structured output. Since, reliability of predictions in the context of multi-target regression was not studied before, we propose two different reliability scores for predictions based on intrinsic mechanisms of ensemble methods. Furthermore, we propose an algorithm for automatic selection of the appropriate threshold on reliability scores.

The second class of methods is based on the **extension of the variance functions of predictive clustering trees** in order to accommodate both labeled and unlabeled examples. This enables to build semi-supervised predictive clustering trees that can exploit unlabeled examples while preserving the appealing characteristics of supervised trees, such as interpretability and computation efficiency. Semi-supervised predictive clustering trees are general in terms of the type of the structured output: They can predict different types of structured outputs: multiple target variables and hierarchically structured classes. We propose parametrization of semi-supervised predictive clustering trees by which it is possible to control the amount of supervision, i.e., the learned models can range from fully unsupervised to fully supervised.

We perform an **extensive empirical evaluation** of the proposed methods on a wide range of datasets from different domains and with different types of structured output. We analyze the influence of the amount of labeled data to the performance of the proposed

methods, as we all various aspects of their practical usability, such as, interpretability, computational complexity, and sensitivity to parameters.

Finally, we perform **two case studies** where we demonstrate the practical usability of the proposed semi-supervised methods for structured output prediction in two practically relevant domains: quantitative structure activity relationship modeling and ecology. In the first case study, we apply the self-training method for multi-target regression to predict the properties of potential antimalarial medications and show that semi-supervised models yield better predictive performance as compared to the supervised. In the second case study, we demonstrate on the problem of water quality prediction that semi-supervised predictive clustering trees can efficiently solve practically relevant machine learning task: learning from partially labeled data. In particular, we show that better predictive performance can be achieved if partially labeled data are exploited, rather than discarded.

10.2 Discussion

The central goals of this thesis are to contribute to the field of machine learning of semi-supervised learning for structured output prediction, and to extend the predictive clustering framework towards semi-supervised learning (i.e., to improve the predictive performance of supervised methods from this framework by the virtue of unlabeled data). As we previously mentioned, we contribute to the field of semi-supervised learning with two classes of semi-supervised methods for structured output prediction based on the predictive clustering framework. Thus, the main goal of this thesis is achieved.

The goal of improving the performance of supervised methods was achieved - but only to a certain degree. Namely, the self-training approach for structured output prediction was successful only on one of the tree types of structured output we considered, i.e., multi-target regression, while on multi-label classification and hierarchical multi-label classification, it was not able to improve over supervised random forests. Though, analysis with the oracle reliability score showed that there is a potential to achieve improvement even for these tasks, however, a more sophisticated reliability scoring functions than the ones we used in this thesis are needed.

With respect to different types of structured output, the semi-supervised predictive clustering trees proved to be more versatile than the self-training approach. They outperformed (often by a large degree) supervised predictive clustering trees on all of the three different types of structured outputs considered, as well as on the machine learning tasks with primitive outputs (i.e., classification and regression). However, the advantage of semi-supervised over supervised trees did not entirely translate to the ensemble setting. when structured outputs are considered, semi-supervised random forests, built by using semi-supervised predictive clustering tree as base models, rarely did benefit from the unlabeled data. On machine learning tasks with primitive outputs, however, semi-supervised random forests were highly competitive to the supervised random forests.

According to our results, it seems that improving (with unlabeled data) a supervised random forest is a harder task than improving a supervised tree. This could be due to the fact that ensembles are very powerful predictive models, which are able to exploit all the information in a given (labeled) dataset and approach learnability of a given domain. Thus, arguably, random forests do not benefit so much from additional information that unlabeled data bring, as compared to single trees. Similar observation, but in different context was observed by Levatić, Kocev, and Džeroski (2014): While the label hierarchy improve the performance of predictive clustering trees for hierarchical multi-label classification, this is not preserved in ensemble setting.

All in all, the two classes of the proposed semi-supervised methods are somewhat

complementary. The self-training approach yielded more success in improving the state-of-the-art supervised method for structured output prediction (i.e., random forests of predictive clustering trees), which is one of the goals of this thesis. On the other hand, semi-supervised predictive clustering trees realized the two other goals of this thesis: (i) implementing a semi-supervised method within the predictive clustering framework than can efficiently address various types of structured outputs and (ii) implementing interpretable semi-supervised method within the predictive clustering framework. We find the realization of these two goals especially important, since, to the best of our knowledge, there is no other interpretable semi-supervised method for structured output prediction.

The results confirmed our hypothesis that the success of the proposed semi-supervised methods to outperform their supervised counterparts depends on the domain and/or the available amount of labeled data. Similar finding was observed before by several researches (Nigam et al., 2000; Cozman et al., 2002; Z.-H. Zhou & Li, 2007; Yuanyuan Guo et al., 2010). In other words, it was found that unlabeled data is not necessarily always beneficial for the performance of the semi-supervised methods. Sometimes it may even hurt their predictive performance. Thus, a recent trend in semi-supervised learning is to design ‘safe’ semi-supervised methods (Kawakita & Takeuchi, 2014; Y.-F. Li, Zha, & Zhou, 2017), i.e., methods that do not perform worse than their corresponding supervised methods. From this aspect, we paid a special attention to practical usability of the proposed semi-supervised methods. Namely, we implemented the Airbag stopping criteria (Leistner et al., 2009) which automatically stops the self-training if performance degradation (according to out-of-bag errors) is detected. Furthermore, we proposed a parametrization of the semi-supervised predictive clustering tree which controls the amount of supervision and acts as a safety mechanism.

To summarize, the self-training approach offers a state-of-the-art predictive performance on multi-target regression problems, while producing black-box models and with the cost of increased computational complexity (due to iterative training of the base model) as compared to supervised random forests. The semi-supervised predictive clustering trees, on the other hand, produce readily interpretable models, which are often considerably more accurate than the corresponding supervised models for structured outputs. The semi-supervised predictive clustering trees and ensembles thereof also exhibit attractive predictive performance on machine learning tasks with primitive outputs (i.e., classification and regression).

10.3 Further Work

There are several possible directions to continue the work presented in this thesis, either to resolve the limitations or drawbacks of the proposed methods we have identified, or to explore the new opportunities opened by these methods.

First of all, an obvious continuation of this work is to extend the semi-supervised methods proposed in this thesis to the types of structured outputs different than multi-target prediction and hierarchical multi-label classification. Such as, time-series classification or sequence learning. For semi-supervised predictive clustering trees this is quite straightforward: One only needs to define a variance function for a given type of structured output. To extend the self-training approach to a new type of structured output, one needs to define a reliability scoring function (for that type of output). As findings in this thesis showed, defining a good reliability scoring function is a challenging task. Such function must be able to discern correct from erroneous prediction while being computationally very efficient (since it is invoked many times). Current approaches often fail the second criteria (Bosnić & Kononenko, 2008). A rather simplistic definitions of reliability scores based on

empirical probabilities we used for multi-label classification and hierarchical multi-label classification tasks did not yield satisfactory results. Thus, for these types of outputs, a more sophisticated reliability scores are needed.

One possible direction of research on how to improve self-training for the multi-label classification and hierarchical multi-label classification tasks would be to optimize thresholds specifically for each target. The reliability scores we use in this thesis calculate an overall score for each unlabeled example. If an example is proclaimed reliable by the reliability scoring function and it enters the training set, all of its target variables will be predicted and used as labels. However, even if predictions for most of its target variables are considered reliable, the 'overall score' can mask the fact that some target variables could be less reliable (and potentially get wrong predictions). Thus, only highly reliable target variables could be predicted, leaving the others as missing values. This would require of the learning method to deal with partially labeled examples. Furthermore, the self-training approach could be extended to other similar paradigms, which are less prone to error propagation, such as co-training (Blum & Mitchell, 1998).

The semi-supervised predictive clustering trees offer an elegant solution for handling partially labeled data. Dealing with such type of data has not received much attention in the semi-supervised learning community, but it is particularly relevant for the task of structured output prediction. Thus, this direction of work merits a more detailed investigation. That is, a detailed evaluation on a larger collection of datasets (including artificial ones) with different characteristic regarding the missing output values. Additionally, the "natural" way of handling partially labeled data with semi-supervised predictive clustering trees could be compared with techniques for missing data imputation.

The semi-supervised clustering trees have intrinsic properties which could be useful in fields of data mining other than semi-supervised learning. Namely, as we demonstrated, with this approach it is possible to perform unsupervised learning, i.e., (hierarchical) clustering while simultaneously providing symbolic descriptions of the clusters. The method could be thus easily compared to methods in that area as well. Next, it could be used to perform feature ranking for semi-supervised and unsupervised learning by including it in algorithms that are based on tree models, such as random forest feature ranking.

The semi-supervised random forests for structured outputs were not so successful in outperforming the supervised random forests, as were semi-supervised predictive clustering trees in outperforming the supervised predictive clustering trees. We would like to perform an in-depth analysis in order to understand this discrepancy, and possibly improve semi-supervised random forests. Another unresolved issue we identified within semi-supervised predictive clustering trees is feature weighting - the benefits of using it are inconclusive. We would like to investigate this issue in more details. For example, by injecting randomly generated features into the dataset and analyzing the performance of (feature weighted) semi-supervised predictive clustering trees when we know that the dataset contains irrelevant features. Additionally, we would like to try feature ranking techniques other than random forest feature importance.

Finally, our results suggest that the pruning techniques currently implemented in the predictive clustering framework are sometimes too aggressive, i.e., they prune trees down to a single node. This observation is more related to the predictive clustering framework in general, than it is to the semi-supervised methods proposed in this thesis, but, nevertheless, designing pruning techniques tailored specifically for different types of structured outputs may improve the quality of predictive clustering trees.

Appendix A

Complete Results of the Evaluation of the CLUS-ST Algorithm

The results of the proposed self-training method (CLUS-ST) for predicting structured outputs were summarized and discussed in Chapter 6. Here, we present the complete results on benchmark datasets, organized by the three considered structured output prediction tasks: multi-target regression, multi-label classification, and hierarchical multi-label classification. For each task, we present the predictive performance of the self-training method with different reliability scores and supervised random forests (CLUS-RF). The amount of labeled data, relative to unlabeled data, varies from 1% to 50%.

A.1 Multi-Target Regression

Table A.1: RMSE on the 9 considered datasets obtained with 1% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $Ttop10\%$ and $Tavg10\%$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).

Method	Dataset														
	Sig.	Real	Sig.	Sim.	Soil	Q.	S.	Flare-2	Water	Q.	Veg.	Cond.	SCM20D	SCMID	RFI
CLUS-RF	6.422	0.062	0.547	45909.739	1.372	2.871	206.99	175.969	13.136	10.833 ± 4.430					
	6.555	0.185	0.217	47440.970	1.354	3.105	230.765	265.669	22.911	13.278 ± 4.631					
	6.4	0.036	0.536	45886.432	1.352	2.866	206.832	175.924	13.136	5.333 ± 4.077					
	6.408	0.036	0.538	45913.693	1.354	2.864	206.99	175.969	12.998	6.222 ± 3.465					
	6.406	0.037	0.541	46016.187	1.348	2.868	207.276	176.801	13.048	9.611 ± 3.018					
	6.388	0.039	0.538	45962.634	1.346	2.869	208.656	176.555	12.968	7.944 ± 3.321					
	6.408	0.047	0.54	45870.267	1.341	2.866	206.99	175.969	13.127	7.167 ± 2.681					
	6.405	0.048	0.54	45909.739	1.343	2.866	206.99	175.969	13.156	7.889 ± 3.286					
	6.405	0.047	0.536	45934.234	1.34	2.872	207.323	176.772	13.115	8.722 ± 3.759					
	6.416	0.046	0.539	45850.644	1.346	2.871	210.392	176.134	13.064	8.944 ± 3.283					
CLUS-RF	6.38	0.055	0.54	46262.372	1.346	2.866	207.383	176.883	13.155	10.444 ± 3.787					
	6.421	0.04	0.543	46224.958	1.341	2.864	207.229	176.619	13.061	8.889 ± 3.765					
	6.421	0.038	0.541	46000.04	1.336	2.86	207.242	176.626	13.069	8 ± 4.493					
	6.375	0.03	0.54	45765.407	1.338	2.867	206.99	175.969	12.676	4 ± 3.260					
	6.375	0.032	0.537	45694.472	1.337	2.864	206.99	175.969	12.878	2.722 ± 1.325					
	6.422	0.062	0.547	45909.739	1.372	2.871	206.99	175.969	13.136	4.5 ± 4.031					
	6.555	0.185	0.217	47440.97	1.354	3.105	230.765	265.669	22.911	8 ± 3.708					
	6.422	0.036	0.545	46251.345	1.368	2.93	207.341	178.391	13.427	5.222 ± 1.970					
	6.422	0.046	0.545	46251.345	1.368	2.905	207.341	178.391	13.461	5.611 ± 1.387					
	6.422	0.04	0.545	46251.345	1.368	3.037	207.332	178.662	13.927	6.111 ± 2.205					
T_OOBInitial + Airbag	6.422	0.046	0.545	46251.345	1.368	2.912	207.331	178.526	13.332	5.278 ± 1.716					
	6.432	0.044	0.542	46337.764	1.351	3.027	216.671	189.056	13.672	6 ± 2.872					
	6.439	0.059	0.544	46114.348	1.366	2.867	209.574	177.558	13.204	4.833 ± 2.598					
	6.408	0.037	0.543	46433.652	1.355	3.109	221.755	207.888	13.403	5.722 ± 3.528					
Tavg 10% + Airbag	6.422	0.059	0.543	45926.363	1.363	2.875	209.183	177.029	13.134	3.722 ± 2.063					

Table A.2: RMSE on the 9 considered datasets obtained with 5% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $Ttop10\%$ and $Tavg10\%$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).

Method	Dataset															
	Sig.	Real	Sig.	Sim.	Soil	Q.	S. Flare-2	Water	Q.	Veg.	Cond.	SCM20D	SCM1D	RF1	Average rank	
CLUS-RF	5.431	0.019	0.524	41384.376	1.262	2.678	177.182	143.124	7.422	11.222 ± 3.743						
	6.337	0.816	0.168	45561.610	1.304	2.988	230.311	229.473	23.985	13.444 ± 4.667						
	5.241	0.018	0.513	41384.376	1.253	2.673	177.153	143.124	7.157	5.5 ± 3.873						
	5.305	0.018	0.511	41384.376	1.255	2.676	177.182	143.079	7.359	7.333 ± 2.739						
	5.239	0.017	0.513	41334.819	1.254	2.679	178.049	144.487	7.177	8.056 ± 4.482						
	5.281	0.017	0.51	41239.703	1.256	2.676	177.207	143.582	7.269	6.611 ± 2.619						
	5.305	0.017	0.511	41191.135	1.255	2.677	177.182	143.124	7.394	6.833 ± 2.784						
	5.317	0.018	0.51	41058.845	1.255	2.677	177.182	143.116	7.416	7.167 ± 3.800						
	5.279	0.018	0.51	41347.346	1.255	2.678	178.004	144.617	7.308	8.944 ± 2.721						
	5.307	0.017	0.509	41366.16	1.257	2.678	177.978	145.404	7.366	9.333 ± 3.976						
	5.273	0.018	0.513	41865.799	1.255	2.679	178.017	144.487	7.425	11.167 ± 2.969						
	5.329	0.019	0.516	41513.983	1.251	2.677	177.716	144.101	7.173	9.5 ± 3.961						
OracleNormAvg	5.286	0.018	0.515	41600.439	1.251	2.675	177.758	144.14	7.217	8.333 ± 3.865						
OracleNormMin	5.235	0.017	0.512	41237.003	1.248	2.677	177.182	143.032	6.74	3.722 ± 2.917						
OracleRankAvg	5.248	0.017	0.509	41078.147	1.25	2.676	177.182	142.967	7.013	2.833 ± 1.199						
OracleRankMin																
CLUS-RF	5.431	0.019	0.524	41384.376	1.262	2.678	177.182	143.124	7.422	4.889 ± 3.209						
	6.337	0.816	0.168	45561.61	1.304	2.988	230.311	229.473	23.985	9 ± 3.000						
	5.414	0.019	0.518	41536.77	1.257	2.678	184.59	152.449	7.609	5.611 ± 1.799						
	5.423	0.018	0.52	41567.099	1.257	2.676	184.062	148.509	7.571	5.278 ± 1.679						
	5.418	0.017	0.517	41509.666	1.257	2.781	184.642	155.167	7.322	4.833 ± 2.634						
	5.418	0.017	0.52	41585.324	1.257	2.676	182.042	145.905	7.468	4.389 ± 1.635						
	5.35	0.017	0.514	42109.113	1.256	2.753	185.862	153.104	7.429	4.722 ± 3.270						
	5.451	0.017	0.524	41078.715	1.263	2.676	178.31	143.608	7.502	4.944 ± 3.330						
	5.427	0.019	0.517	41721.025	1.257	2.742	189.001	161.129	7.51	6.889 ± 1.949						
	5.376	0.019	0.521	41386.025	1.263	2.678	177.726	143.052	7.424	4.444 ± 2.888						

Table A.3: RMSE on the 9 considered datasets obtained with 10% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $Ttop10\%$ and $Tavg10\%$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).

		Dataset													
Method	Sig.	Real	Sig.	Sim.	Soil	Q.	S. Flare-2	Water	Q.	Veg.	Cond.	SCM20D	SCM1D	RF1	Average rank
CLUS-RF		4.922	0.015	0.527	38615.185	1.231	2.601	127.894	5.089	10.444 ± 3.948					
		5.815	0.094	0.169	43551.48	1.276	2.998	220.416	219.555	22.315	13.444 ± 4.667				
		4.918	0.014	0.519	38576.336	1.225	2.602	161.021	127.894	4.905	7.444 ± 3.820				
		4.913	0.014	0.519	38576.336	1.225	2.602	161.061	127.894	5.038	8.278 ± 3.203				
		4.901	0.014	0.517	38653.538	1.224	2.603	160.726	129.139	4.904	7.722 ± 3.684				
		4.873	0.014	0.516	38551.846	1.225	2.601	161.312	128.189	4.955	6.556 ± 2.877				
		4.904	0.014	0.518	38621.215	1.226	2.601	161.061	127.894	5.129	8.833 ± 2.773				
		4.904	0.014	0.519	38607.197	1.226	2.601	161.061	127.879	5.107	8.278 ± 3.280				
		4.882	0.014	0.516	38655.623	1.225	2.601	162.15	129.465	4.972	9 ± 3.354				
		4.894	0.014	0.517	38602.015	1.227	2.601	161.273	128.512	5.036	8.278 ± 2.224				
		4.861	0.014	0.519	39485.778	1.225	2.604	161.948	129.231	5.115	10.667 ± 400				
		4.867	0.014	0.52	39189.14	1.223	2.601	161.687	128.902	4.929	8.611 ± 3.855				
		4.866	0.014	0.519	39004.731	1.223	2.6	161.276	128.937	4.901	7.111 ± 4.053				
	4.877	0.013	0.516	38615.185	1.219	2.6	160.148	127.846	4.545	2.833 ± 2.462					
	4.848	0.013	0.515	38579.84	1.221	2.597	161.023	127.894	4.684	2.5 ± 1.458					
CLUS-RF		4.922	0.015	0.527	38615.185	1.231	2.601	161.061	127.894	5.089	4.778 ± 2.938				
		5.815	0.094	0.169	43551.48	1.276	2.998	220.416	219.555	22.315	9 ± 3.000				
		4.901	0.014	0.525	39178.759	1.225	2.602	166.109	130.989	5.264	4.833 ± 2.449				
		4.914	0.014	0.525	39148.626	1.225	2.603	164.526	129.572	5.191	4.833 ± 1.581				
		4.915	0.014	0.525	39381.705	1.224	2.605	167.385	136.241	5.073	5.167 ± 2.716				
		4.916	0.014	0.523	38966.383	1.225	2.601	163.248	128.378	5.112	3.889 ± 1.341				
		4.905	0.014	0.52	39856.55	1.228	2.646	166.137	134.911	5.02	5.056 ± 3.087				
		4.953	0.014	0.526	38539.778	1.233	2.603	161.236	128.107	5.098	4.889 ± 2.559				
		4.992	0.015	0.527	38890.317	1.229	2.609	168.485	136.906	5.143	7.722 ± 1.787				
		4.954	0.015	0.526	38581.027	1.232	2.603	161.014	127.805	5.085	4.833 ± 3.122				

Table A.4: RMSE on the 9 considered datasets obtained with 15% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $Ttop10\%$ and $Tavg10\%$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).

Method	Dataset														
	Sig.	Real	Sig.	Sim.	Soil	Q.	S. Flare-2	Water	Q.	Veg.	Cond.	SCM20D	SCM1D	RF1	Average rank
CLUS-RF	4.194		0.012		0.513		37562.992	1.214		2.568		149.976	120.878	4.014	10.5 ± 3.182
	5.194		0.219		0.165		97039.2	1.276		3.027		225.622	237.821	20.391	13.444 ± 4.667
	3.978		0.012		0.506		37568.513	1.208		2.568		149.825	120.892	3.882	7.389 ± 2.837
	4.023		0.012		0.505		37562.992	1.21		2.567		149.83	120.713	3.941	7 ± 2.905
	4.006		0.012		0.504		37509.874	1.209		2.567		149.763	121.37	3.805	5.778 ± 2.451
	4.028		0.012		0.504		37329.419	1.212		2.568		149.764	120.863	3.869	7.167 ± 3.152
	4.006		0.012		0.506		37326.996	1.209		2.568		150.034	120.872	4.049	7.833 ± 3.072
	4.013		0.012		0.506		37202.539	1.21		2.568		149.976	120.878	4.038	8.056 ± 3.225
	4.011		0.012		0.504		37316.548	1.209		2.568		151.037	122.386	3.973	8.389 ± 3.781
	4.013		0.012		0.505		37445.731	1.212		2.568		150.276	121.2	3.973	8.944 ± 1.976
	4.032		0.012		0.505		38273.226	1.21		2.572		150.947	122.107	4.054	11.722 ± 2.587
	4.035		0.012		0.507		37963.36	1.207		2.568		150.691	121.76	3.849	9.556 ± 4.065
	4.027		0.012		0.507		37829.565	1.208		2.567		150.169	121.801	3.867	8.722 ± 3.589
3.931		0.012		0.503		37475.782	1.204		2.558		147.849	120.768	3.505	2.944 ± 2.579	
3.974		0.012		0.502		37309.152	1.207		2.556		149.657	120.742	3.666	2.556 ± 1.895	
CLUS-RF	4.194		0.012		0.513		37562.992	1.214		2.568		149.976	120.878	4.014	4.5 ± 2.550
	5.194		0.219		0.165		97039.2	1.276		3.027		225.622	237.821	20.391	9 ± 3.000
	4.144		0.012		0.513		37789.77	1.209		2.568		152.247	122.217	4.126	5.556 ± 1.895
	4.148		0.012		0.513		37808.887	1.209		2.569		151.269	121.72	4.115	5.833 ± 1.369
	4.124		0.012		0.513		38108.565	1.209		2.568		154.843	126.583	4.045	5.833 ± 2.332
	4.114		0.012		0.513		37653.957	1.209		2.569		150.23	120.983	4.022	4.5 ± 1.458
	4.076		0.012		0.507		38515.324	1.209		2.595		153.988	125.946	4.016	5.278 ± 2.906
	4.166		0.012		0.513		37584.892	1.214		2.568		149.808	120.805	4.003	4.167 ± 2.622
	4.124		0.012		0.511		37734.932	1.209		2.57		154.939	126.063	4.053	5.778 ± 2.265
	4.202		0.012		0.513		37480.141	1.214		2.569		150.143	120.799	3.998	4.556 ± 3.167

Table A.5: RMSE on the 9 considered datasets obtained with 20% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $Ttop10\%$ and $Tavg10\%$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).

Method	Dataset														
	Sig.	Real	Sig.	Sim.	Soil	Q.	S.	Flare-2	Water	Q.	Veg.	Cond.	SCM20D	SCM1D	RF1
CLUS-RF	4.14	0.011	0.512	36704.708	1.203	2.544	142.221	115.072	3.241	11.333 ± 2.958					
MSVR	5.076	0.086	0.157	41336.29	1.275	3.054	231.853	213.93	23.705	13.444 ± 4.667					
VarianceNormAvg	3.953	0.011	0.507	36704.708	1.199	2.542	141.417	115.042	3.157	6.278 ± 3.270					
VarianceNormMin	3.989	0.011	0.507	36516.581	1.2	2.543	141.502	115.042	3.187	7.444 ± 2.920					
VarianceRankAvg	3.983	0.011	0.506	36442.51	1.2	2.543	141.661	115.307	3.08	7.167 ± 2.610					
VarianceRankMin	3.966	0.011	0.506	36656.879	1.202	2.544	141.424	115.125	3.158	8.444 ± 2.242					
RForestProxNormAvg	3.963	0.011	0.506	36477.326	1.2	2.544	141.592	115.046	3.283	7.722 ± 2.852					
RForestProxNormMin	3.98	0.011	0.506	36479.676	1.201	2.544	141.811	115.072	3.279	8.778 ± 2.224					
RForestProxRankAvg	3.944	0.011	0.504	36546.452	1.2	2.544	141.901	116.016	3.234	8 ± 3.363					
RForestProxRankMin	3.949	0.011	0.505	36589.821	1.202	2.544	142.145	115.119	3.222	8.444 ± 2.811					
Random	3.958	0.011	0.505	37550.074	1.2	2.547	143.082	116.181	3.305	10.778 ± 3.890					
OracleNormAvg	4	0.011	0.507	37340.94	1.198	2.542	142.098	115.869	3.223	9.278 ± 3.589					
OracleNormMin	4.006	0.011	0.507	37032.297	1.198	2.541	141.37	115.876	3.151	7.778 ± 4.583					
OracleRankAvg	3.94	0.011	0.504	36376.513	1.194	2.528	138.857	114.778	2.932	2.167 ± 2.121					
OracleRankMin	3.935	0.011	0.504	36440.297	1.198	2.526	140.932	115.059	3.001	2.944 ± 2.098					
CLUS-RF	4.14	0.011	0.512	36704.708	1.203	2.544	142.221	115.072	3.241	5.333 ± 2.411					
MSVR	5.076	0.086	0.157	41336.29	1.275	3.054	231.853	213.93	23.705	9 ± 3.000					
T_OOB	4.079	0.011	0.51	36946.38	1.2	2.545	142.477	115.683	3.29	5.389 ± 1.516					
T_OOB + Airbag	4.096	0.011	0.511	36815.638	1.2	2.545	142.024	115.454	3.293	5.444 ± 1.488					
T_OOBInitial	4.075	0.011	0.509	36991.573	1.2	2.544	145.063	118.584	3.319	5.778 ± 2.740					
T_OOBInitial + Airbag	4.107	0.011	0.51	36811.305	1.2	2.543	141.772	115.116	3.231	3.5 ± 1.601					
T_top10%	4.004	0.011	0.507	37708.519	1.2	2.564	144.544	118.808	3.301	5.944 ± 3.167					
T_top10% + Airbag	4.16	0.011	0.512	36596.684	1.204	2.544	141.842	115.115	3.265	5.278 ± 2.980					
T_avg10%	4.114	0.011	0.511	36582.057	1.2	2.544	144.564	118.045	3.269	5.222 ± 2.123					
T_avg10% + Airbag	4.123	0.011	0.511	36677.801	1.204	2.543	141.918	115.01	3.229	4.111 ± 2.859					

Table A.6: RMSE on the 9 considered datasets obtained with 30% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $Ttop10\%$ and $Tavg10\%$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).

Method	Dataset														
	Sig.	Real	Sig.	Sim.	Soil	Q.	S. Flare-2	Water	Q.	Veg.	Cond.	SCM20D	SCMID	RFI	Average rank
CLUS-RF	CLUS-RF	3.635	0.01	0.519			34586.338	1.192		2.507		130.013	106.865	2.45	10.611 $\pm$ 3.190
	MSVR	4.796	0.081	0.151			38370.49	1.276		3.341		222.24	218.396	18.38	13.444 $\pm$ 4.667
	VarianceNormAvg	3.613	0.009	0.513			34598.023	1.189		2.508		129.386	106.801	2.405	7.667 $\pm$ 3.640
	VarianceNormMin	3.628	0.01	0.512			34586.338	1.19		2.507		129.221	106.83	2.439	8 $\pm$ 1.871
	VarianceRankAvg	3.59	0.01	0.512			34581.316	1.189		2.509		128.981	106.769	2.326	5.944 $\pm$ 3.321
	VarianceRankMin	3.627	0.01	0.512			34554.094	1.191		2.506		129.209	106.882	2.402	7.556 $\pm$ 3.056
	RForestProxNormAvg	3.547	0.01	0.513			34591.64	1.189		2.507		129.84	106.783	2.505	8.278 $\pm$ 3.866
	RForestProxNormMin	3.607	0.01	0.513			34549.428	1.19		2.507		129.732	106.766	2.497	8.444 $\pm$ 3.137
	RForestProxRankAvg	3.588	0.01	0.511			34650.811	1.19		2.507		129.781	107.159	2.484	8.444 $\pm$ 3.292
	RForestProxRankMin	3.635	0.01	0.512			34495.345	1.191		2.507		129.715	106.844	2.466	8.778 $\pm$ 3.289
	Random	3.602	0.01	0.512			35459.71	1.19		2.511		130.77	107.674	2.526	11.444 $\pm$ 3.077
	OracleNormAvg	3.594	0.01	0.514			35135.863	1.187		2.503		129.588	107.462	2.461	8.611 $\pm$ 3.903
OracleNormMin	3.592	0.01	0.512			34894.938	1.188		2.502		129.064	107.497	2.454	7.278 $\pm$ 3.492	
OracleRankAvg	3.591	0.01	0.511			34504.833	1.185		2.482		125.819	105.148	2.246	2.667 $\pm$ 2.449	
OracleRankMin	3.548	0.01	0.511			34479.713	1.187		2.483		127.522	106.642	2.328	2.833 $\pm$ 2.031	
CLUS-RF	CLUS-RF	3.635	0.01	0.519			34586.338	1.192		2.507		130.013	106.865	2.45	4.944 $\pm$ 2.324
	MSVR	4.796	0.081	0.151			38370.49	1.276		3.341		222.24	218.396	18.38	9 $\pm$ 3.000
	T_OOB	3.662	0.01	0.515			34789.872	1.191		2.508		129.29	107.021	2.526	4.944 $\pm$ 1.810
	T_OOB + Airbag	3.627	0.01	0.519			34775.811	1.191		2.507		129.227	107.058	2.469	4.278 $\pm$ 2.306
	T_OOBInitial	3.684	0.01	0.518			34994.06	1.19		2.508		130.03	108.118	2.506	6.333 $\pm$ 2.236
	T_OOBInitial + Airbag	3.594	0.01	0.518			34698.755	1.19		2.508		129.701	106.747	2.448	3.222 $\pm$ 1.716
	T_top10%	3.634	0.01	0.515			35716.041	1.19		2.519		130.866	108.944	2.498	6.111 $\pm$ 2.987
	T_top10% + Airbag	3.679	0.01	0.519			34552.385	1.192		2.508		129.755	106.847	2.452	5.111 $\pm$ 2.571
	T_avg10%	3.634	0.01	0.52			34814.006	1.191		2.509		129.646	107.423	2.541	6.389 $\pm$ 2.421
	T_avg10% + Airbag	3.669	0.01	0.518			34556.359	1.192		2.508		129.902	106.85	2.437	4.667 $\pm$ 2.291

Table A.7: RMSE on the 9 considered datasets obtained with 50% of labeled data by supervised methods (CLUS-RF and MSVR), self-training with the proposed reliability scores (VarianceNormAvg, VarianceNormMin, VarianceRankAvg, VarianceRankMin, RForestProxNormAvg, RForestProxNormMin, RForestProxRankAvg and RForestProxRankMin), and self-training with benchmark scores (Random, OracleNormAvg, OracleNormMin, OracleRankAvg and OracleRankMin). The lower sub-table presents the performance of self-training with the VarianceNormAvg reliability score and several procedures for automatic selection of a threshold for the reliability score (T_OOB , $T_OOBInitial$, $T_{top10\%}$ and $T_{avg10\%}$), with or without the Airbag stopping criteria. The best result for each column of each sub-table is marked in bold. The last column presents the average rank of the methods according to their performance on the 9 datasets (calculated separately for each sub-table).

Method	Dataset									
	Sig. Real	Sig. Sim.	Soil Q.	S. Flare-2	Water Q.	Veg. Cond.	SCM20D	SCM1D	RF1	Average rank
CLUS-RF	3.428	0.008	0.522	33734.994	1.175	2.465	114.366	96.226	1.792	10.611 $\pm$ 3.110
MSVR	4.623	0.161	0.187	50224.72	1.258	3.242	219.344	292.572	16.619	13.444 $\pm$ 4.667
VarianceNormAvg	3.441	0.008	0.517	33734.994	1.172	2.464	112.99	95.937	1.783	6.444 $\pm$ 3.147
VarianceNormMin	3.399	0.008	0.517	33734.994	1.174	2.464	113.697	96.141	1.771	6.5 $\pm$ 2.550
VarianceRankAvg	3.414	0.008	0.517	33734.994	1.173	2.465	112.907	96.026	1.746	5.944 $\pm$ 2.555
VarianceRankMin	3.426	0.008	0.517	33734.994	1.174	2.464	113.365	96.037	1.752	6.833 $\pm$ 2.475
RForestProxNormAvg	3.429	0.008	0.518	33784.707	1.173	2.465	114.164	96.189	1.833	10 $\pm$ 1.854
RForestProxNormMin	3.415	0.008	0.518	33732.11	1.174	2.464	114.366	96.214	1.834	9.111 $\pm$ 3.638
RForestProxRankAvg	3.414	0.008	0.518	33787.903	1.173	2.465	114.153	96.16	1.807	8.833 $\pm$ 2.107
RForestProxRankMin	3.417	0.008	0.518	33844.657	1.174	2.465	114.199	96.181	1.79	9.667 $\pm$ 1.984
Random	3.41	0.008	0.518	34551.197	1.173	2.467	114.931	96.607	1.856	11.111 $\pm$ 4.099
OracleNormAvg	3.418	0.008	0.516	34187.455	1.171	2.455	113.596	96.453	1.825	7.722 $\pm$ 4.032
OracleNormMin	3.432	0.008	0.517	34075.344	1.171	2.453	113.631	96.277	1.796	8.167 $\pm$ 3.649
OracleRankAvg	3.411	0.008	0.517	33734.994	1.168	2.429	109.839	93.7	1.728	3.167 $\pm$ 2.727
OracleRankMin	3.391	0.008	0.516	33723.302	1.17	2.431	111.569	95.147	1.731	2.444 $\pm$ 1.960
CLUS-RF	3.428	0.008	0.522	33734.994	1.175	2.465	114.366	96.226	1.792	6.222 $\pm$ 2.108
MSVR	4.623	0.161	0.187	50224.72	1.258	3.242	219.344	292.572	16.619	9 $\pm$ 3.000
T_OOB	3.404	0.008	0.522	34021.286	1.174	2.464	113.655	96.196	1.801	5.111 $\pm$ 2.329
$T_OOB + Airbag$	3.464	0.008	0.522	33820.038	1.174	2.464	114.086	96.103	1.785	5.111 $\pm$ 1.710
$T_OOBInitial$	3.422	0.008	0.521	34237.089	1.172	2.465	113.262	96.113	1.811	4.944 $\pm$ 2.480
$T_OOBInitial + Airbag$	3.439	0.008	0.522	33807.261	1.174	2.463	114.299	96.008	1.783	4.5 $\pm$ 2.291
$T_{top10\%}$	3.504	0.008	0.519	34671.671	1.173	2.469	114.209	96.69	1.815	6.556 $\pm$ 3.087
$T_{top10\%} + Airbag$	3.406	0.008	0.521	33734.61	1.175	2.464	114.441	96.113	1.778	4.444 $\pm$ 2.732
$T_{avg10\%}$	3.421	0.008	0.521	33992.132	1.174	2.465	113.145	95.938	1.799	4.167 $\pm$ 2.151
$T_{avg10\%} + Airbag$	3.456	0.008	0.522	33743.407	1.175	2.464	114.226	96.079	1.773	4.944 $\pm$ 2.480

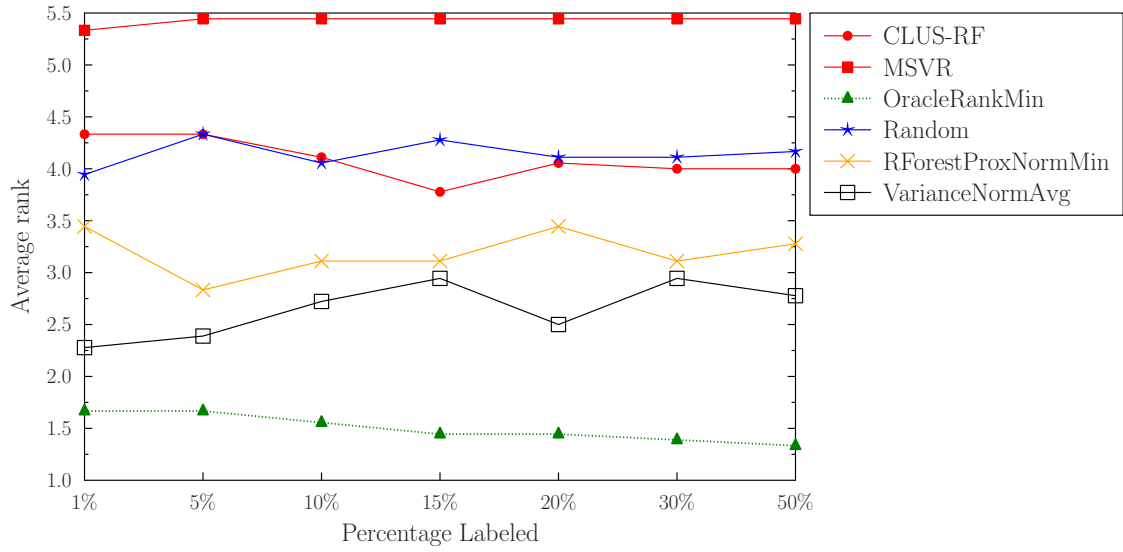


Figure A.1: Average ranks for different percentages of labeled data (from 1% to 50%) of supervised methods (red lines), self-training with the best reliability score based on variance (black), the best score based on random forest proximities (orange), the best benchmark score (green), and random score (blue).

A.2 Multi-Label Classification

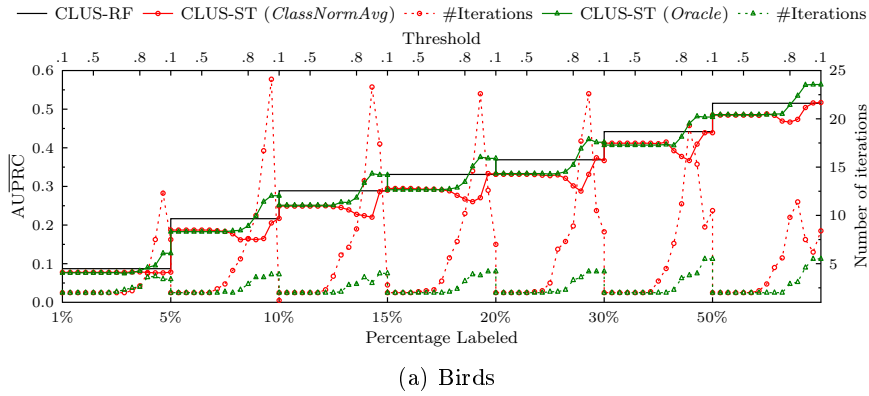


Figure A.2: Comparison of predictive performance for supervised random forests (black line) and self-training (red line) on MLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figure (a).

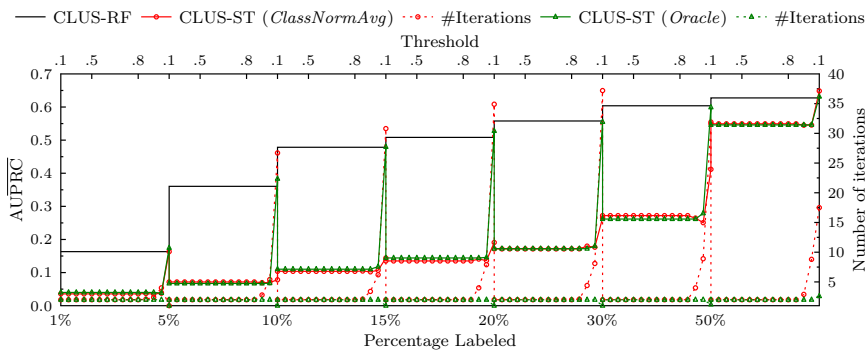
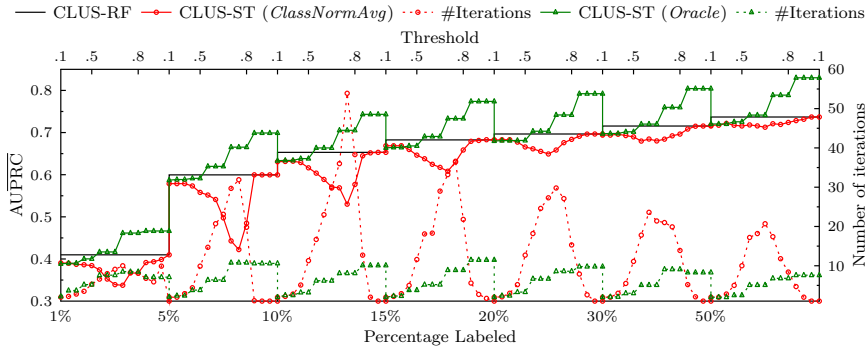
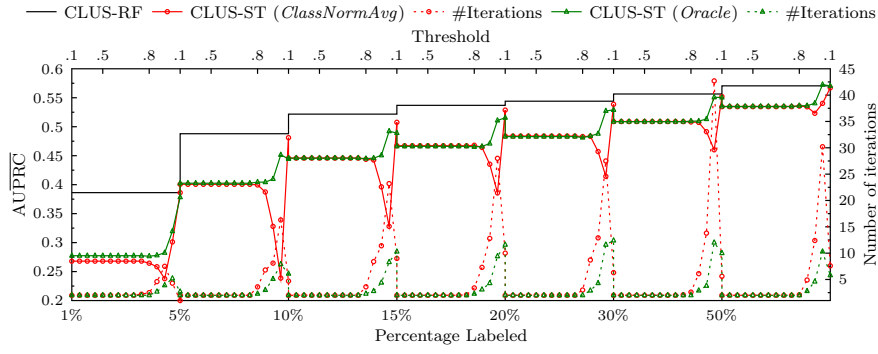
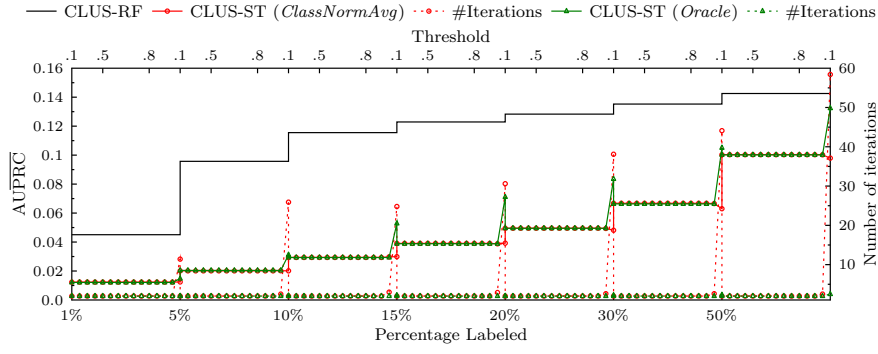


Figure A.2: Comparison of predictive performance for supervised random forests (black line) and self-training (red line) on MLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figures (b) to (e).

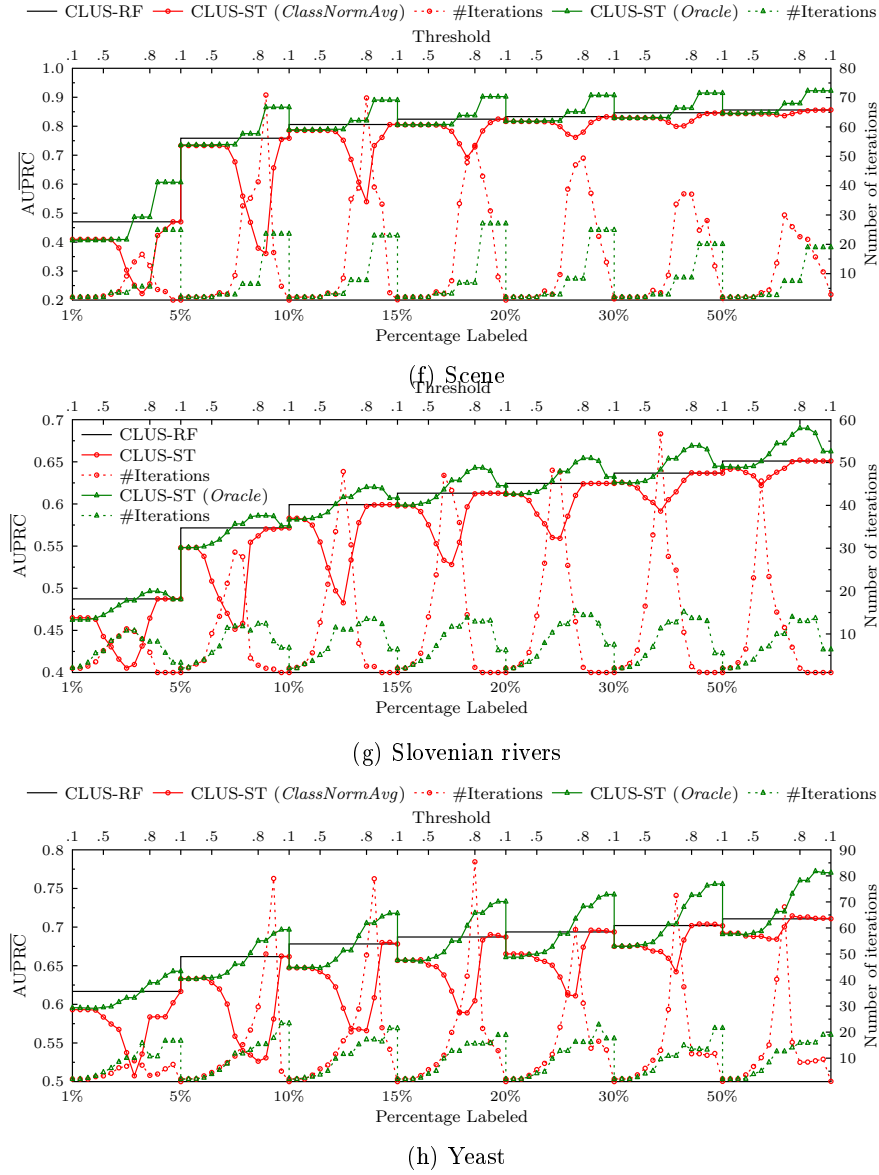
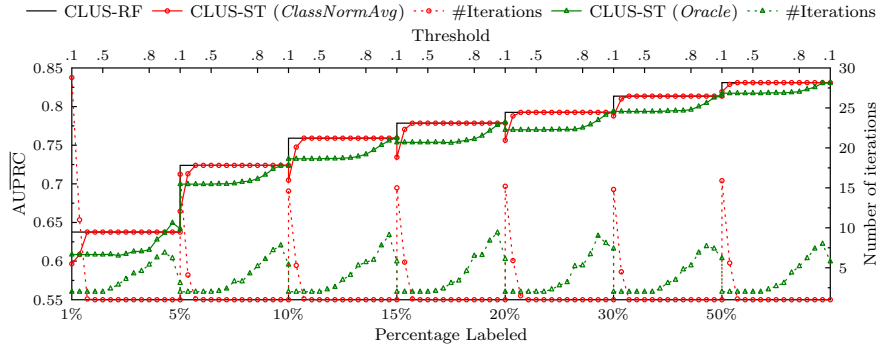
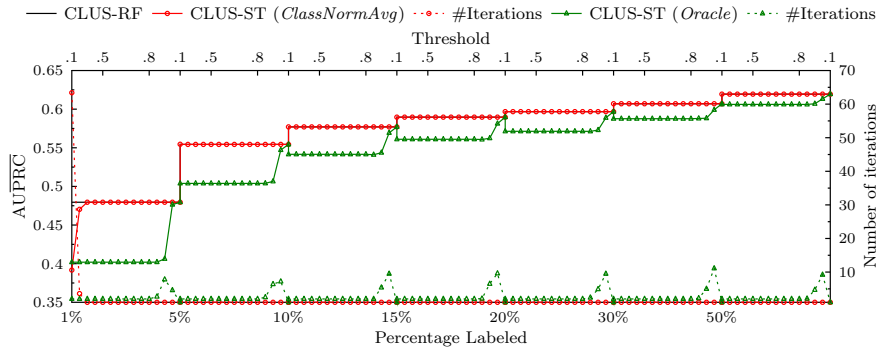


Figure A.2: Comparison of predictive performance for supervised random forests (black line) and self-training (red line) on MLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figures (f) to (i).

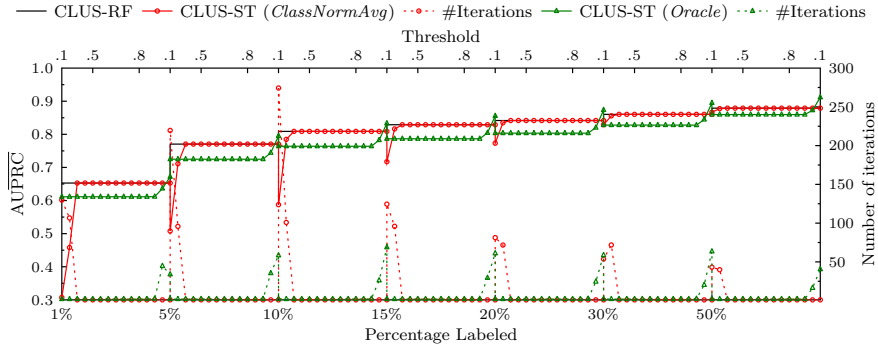
A.3 Hierarchical Multi-Label Classification



(a) Danish farms

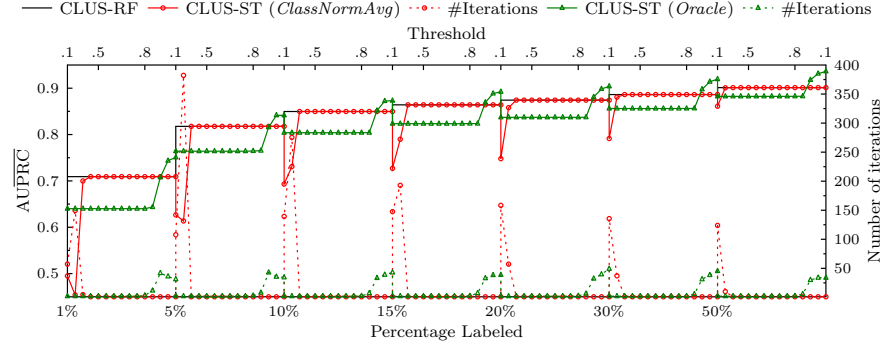


(b) Slovenian rivers

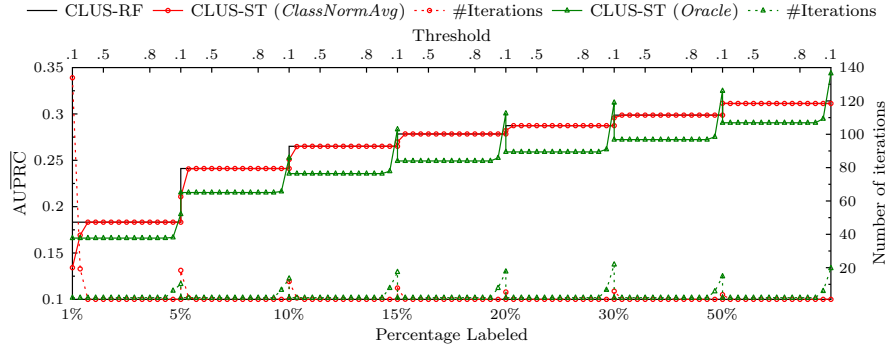


(c) ImCLEF07A

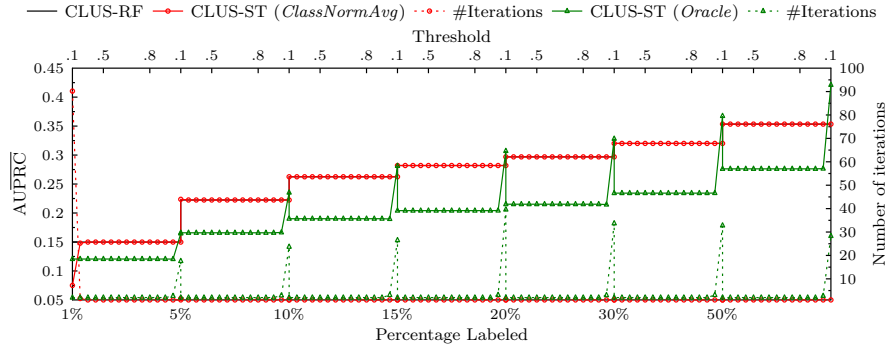
Figure A.3: Comparison of predictive performance for supervised random forests (black line), self-training (red line) and self-training with Oracle score (green line) on HMLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figures (a) to (c).



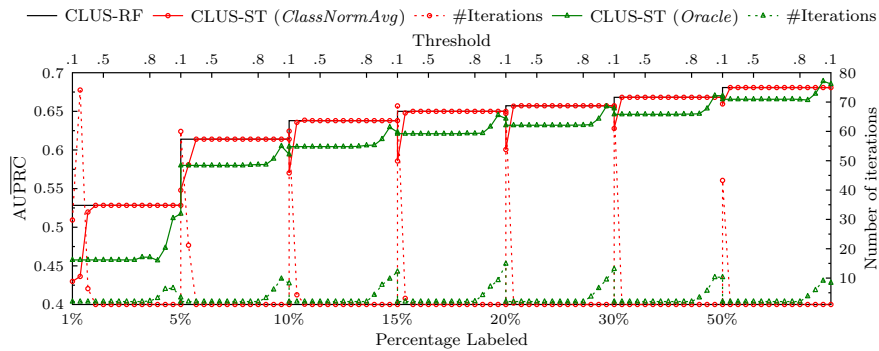
(d) ImCLEF07D



(e) ExprYeast-FunCat



(f) SeqAra-FunCat



(g) Enron

Figure A.3: Comparison of predictive performance for supervised random forests (black line), self-training (red line) and self-training with Oracle score (green line) on HMLC datasets. Percentage of labeled data varies from 1% to 50%, while within each percentage of labeled data, the threshold for reliability score varies from 0.1 to 0.99. Sub-figures (d) to (g).

Appendix B

Additional Results of the Evaluation of the SSL-PCT Algorithm

The results of the proposed semi-supervised predictive clustering trees self-training method (SSL-PCT) for predicting structured outputs were presented and discussed in Chapter 7. Here, we additionally present the results of *pruned* trees for multi-label classification, and hierarchical multi-label classification, and *unpruned* trees for multi-target regression.

B.1 Predictive Performance of Unpruned Trees

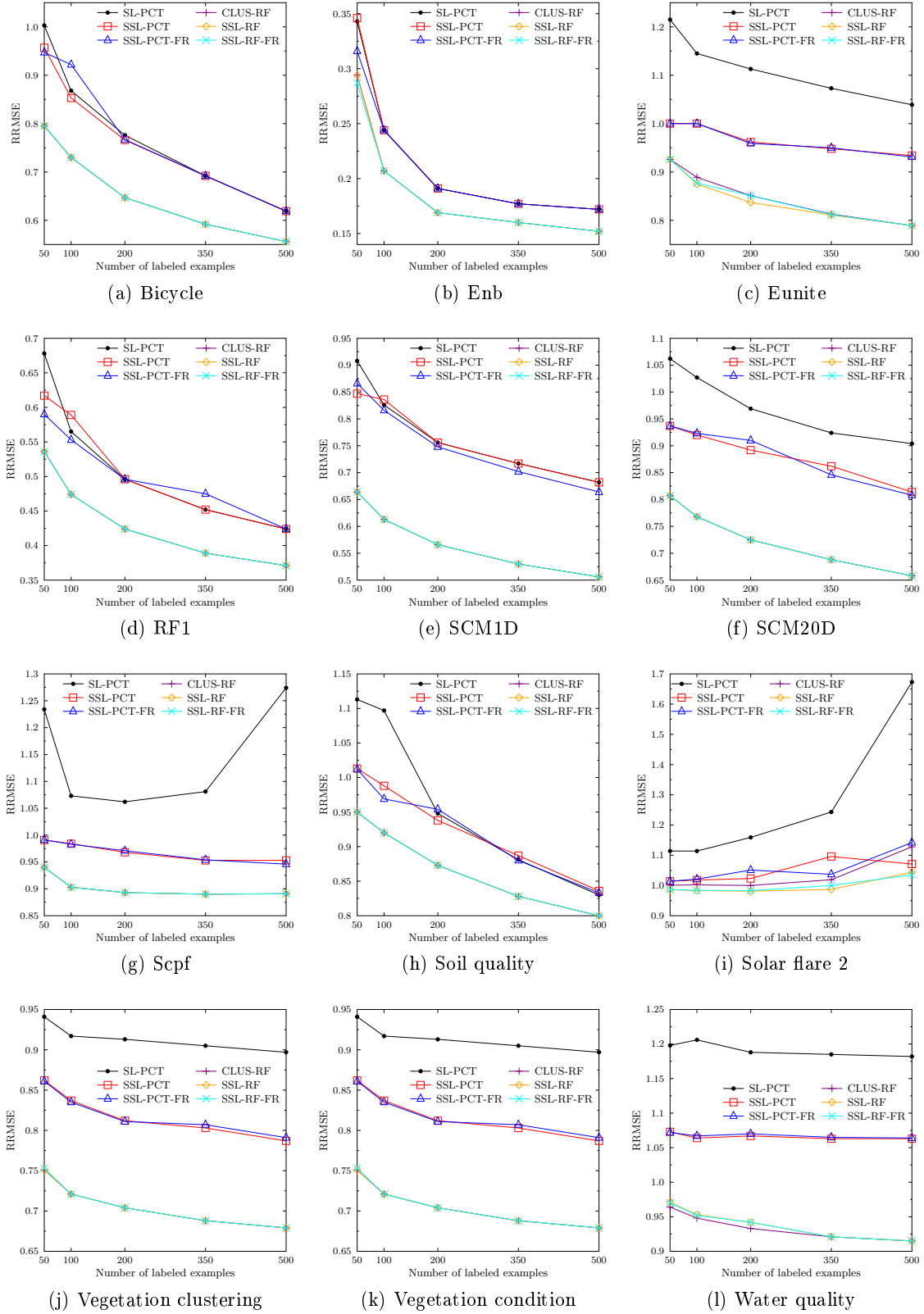


Figure B.1: Predictive performance of the supervised and semi-supervised methods on the multi-target regression datasets. Single tree methods (SL-PCT, SSL-PCT and SSL-PCT-FR) are not pruned.

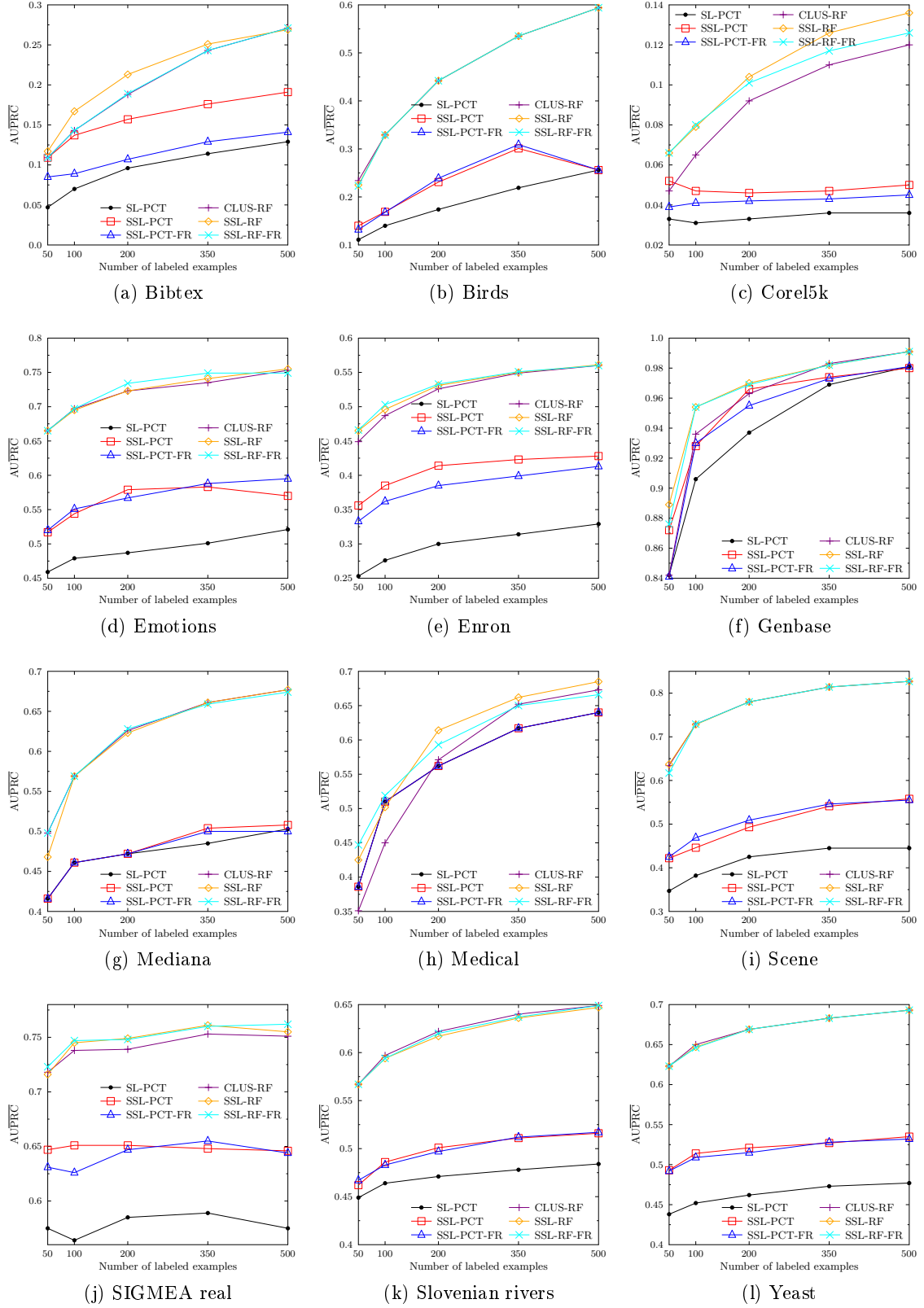


Figure B.2: Predictive performance of the supervised and semi-supervised methods on the multi-label classification datasets. Single tree methods (SL-PCT, SSL-PCT and SSL-PCT-FR) are not pruned.

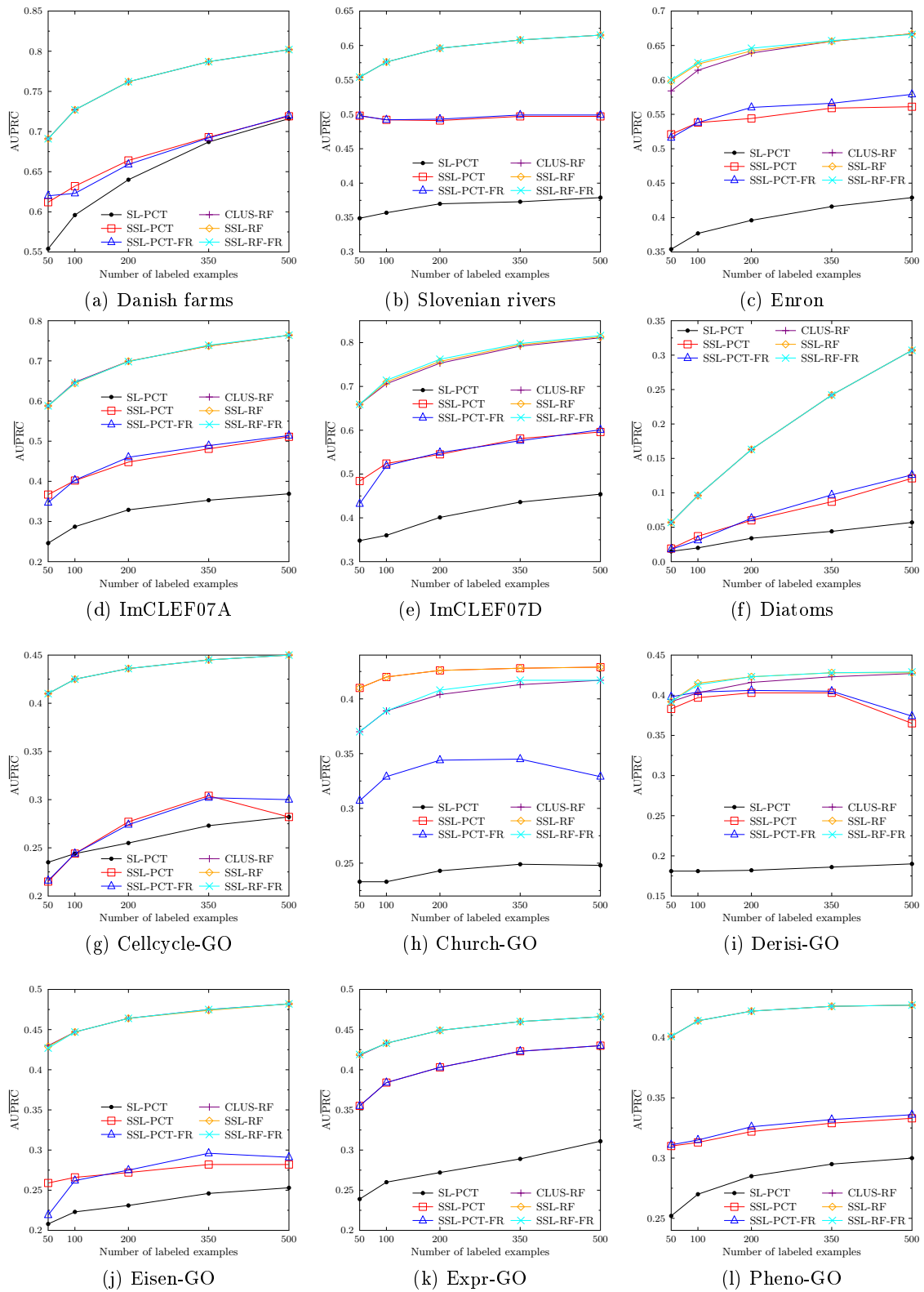


Figure B.3: Predictive performance of the supervised and semi-supervised methods on the hierarchical multi-label classification datasets. Single tree methods (SL-PCT, SSL-PCT and SSL-PCT-FR) are not pruned.

B.2 Model Sizes of Unpruned Trees

Table B.1: Model sizes of pruned trees, in terms of number of nodes, obtained with the supervised PCTs (SL-PCT) and the semi-supervised PCTs (SSL-PCT) on the 12 multi-target regression datasets, 12 multi-label classification datasets and 12 hierarchical multi-label classification datasets. The methods are not pruned.

Dataset	Number of labeled examples									
	50		100		200		350		500	
	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT	SL -PCT	SSL -PCT
MULTI-TARGET REGRESSION										
Bicycle	42.6	38.2	85	53.4	171.4	102.2	300.2	300.2	433	433
Enb	41	49.8	83.6	83.6	167.8	167.8	295.6	295.6	429.4	429.4
Eunite	42.2	1	86	1	174.6	277.6	302.6	460.8	435	628.2
RF1	41.4	80.4	85.4	152.4	171.4	171.4	304.4	304.4	429.6	429.6
SCM1D	42.6	6.2	85.6	12.2	173	173	303.4	303.4	436.6	436.6
SCM20D	42.4	5	88.2	3.6	175	20	303.2	34.8	432	529.6
Scpf	33.4	39.2	67.8	75.6	130.2	153.4	228	269.6	317.2	382.2
Soil quality	42.2	8	83.2	19.6	157.8	22.6	252.4	49.2	330	116.4
Solar flare 2	19.2	20.8	40	37.8	73.4	61.8	117.8	129.2	157.4	155.6
Vegetation clustering	43.2	1	86.4	1.8	172.2	23.6	305.4	42	432.2	27.8
Vegetation condition	34.8	17.4	71.8	35.8	145.4	67.8	256.4	104	361.6	150.6
Water quality	42	27.8	86	58.6	172.8	131.2	304.8	239.6	436.6	378
Average:	38.9	24.6	79.1	44.6	157.1	114.4	272.9	211.1	385.9	341.4
MULTI-LABEL CLASSIFICATION										
Bibtex	44.6	310.4	86.2	641.8	167.4	1132.6	290.4	1675.6	412.8	2075.2
Birds	26	67	52.6	135	104.8	200.6	179	288.6	251	251
Corel5k	43.6	355.2	87.4	413.4	169.4	589.8	294.6	970	423	1282.2
Emotions	31.6	55.2	60.4	105.2	113.6	183.6	198.4	297	277.8	396.4
Enron	41	198.4	81.6	354.6	162.8	541.2	283.2	706.2	396	814.2
Genbase	20.4	27.8	27.8	39.4	34.8	44.6	42.8	50.2	52.8	61.4
Mediana	31.8	31.8	58.2	58.2	113.6	113.6	193.4	216.2	270.6	311.2
Medical	34.6	34.6	63.2	63.2	106	106	166.6	166.6	227.2	227.2
Scene	22.4	53	36.8	116.4	66.8	215	113	358.2	162.2	489.2
SIGMEA real	24.4	35.2	52.4	67.8	98.4	134.6	177.8	217.2	255.4	300.4
Slovenian rivers	41.8	46.8	85.2	119	171.6	224.8	299.6	360.4	429.6	484.8
Yeast	37.4	51.4	74.2	99	149.2	204.4	257.6	347.6	361.2	470.8
Average:	33.3	105.6	63.8	184.4	121.5	307.6	208.0	471.2	293.3	597.0
HIERARCHICAL MULTI-LABEL CLASSIFICATION										
Danish farms	41.2	154.8	81.8	231.2	156	291.4	248.8	353	323.4	382.2
Slovenian rivers	42.6	673.6	85.4	702.2	173.6	606	306	735.6	432	816.2
Enron	42.2	579.2	83.6	611.8	164.6	758.2	288	939	412	1021.2
ImCLEF07A	29.6	44.2	55.4	103.2	103.4	357.4	171	293.8	238.2	863.2
ImCLEF07D	30.4	305.4	57	495	106.8	827.8	175	1199	242.4	1561.2
Diatoms	42.6	139	86.4	715.4	173	1125.8	294.4	1478.8	409	1195.6
Celcycle-GO	37.4	412.8	78.6	78.6	154.4	1020	271.8	1228.8	389	389
Church-GO	35.6	1.4	69.2	1	122.6	1	207.6	1	292.6	1
Derisi-GO	43.4	895.6	86.6	1076.6	172.4	1352.8	304.2	1658.6	432.8	1695.6
Eisen-GO	39	142	80	233.6	160.8	399	275	595.8	390.4	1076.8
Expr-GO	38.2	43.4	76	45.2	153	48.2	268.2	51.2	383.8	54.4
Pheno-GO	30	131.2	53.2	166.2	91.4	201.6	142	257.4	195	295.4
Average:	37.7	293.6	74.4	371.7	144.3	582.4	246.0	732.7	345.1	779.3

B.3 Influence of the w Parameter

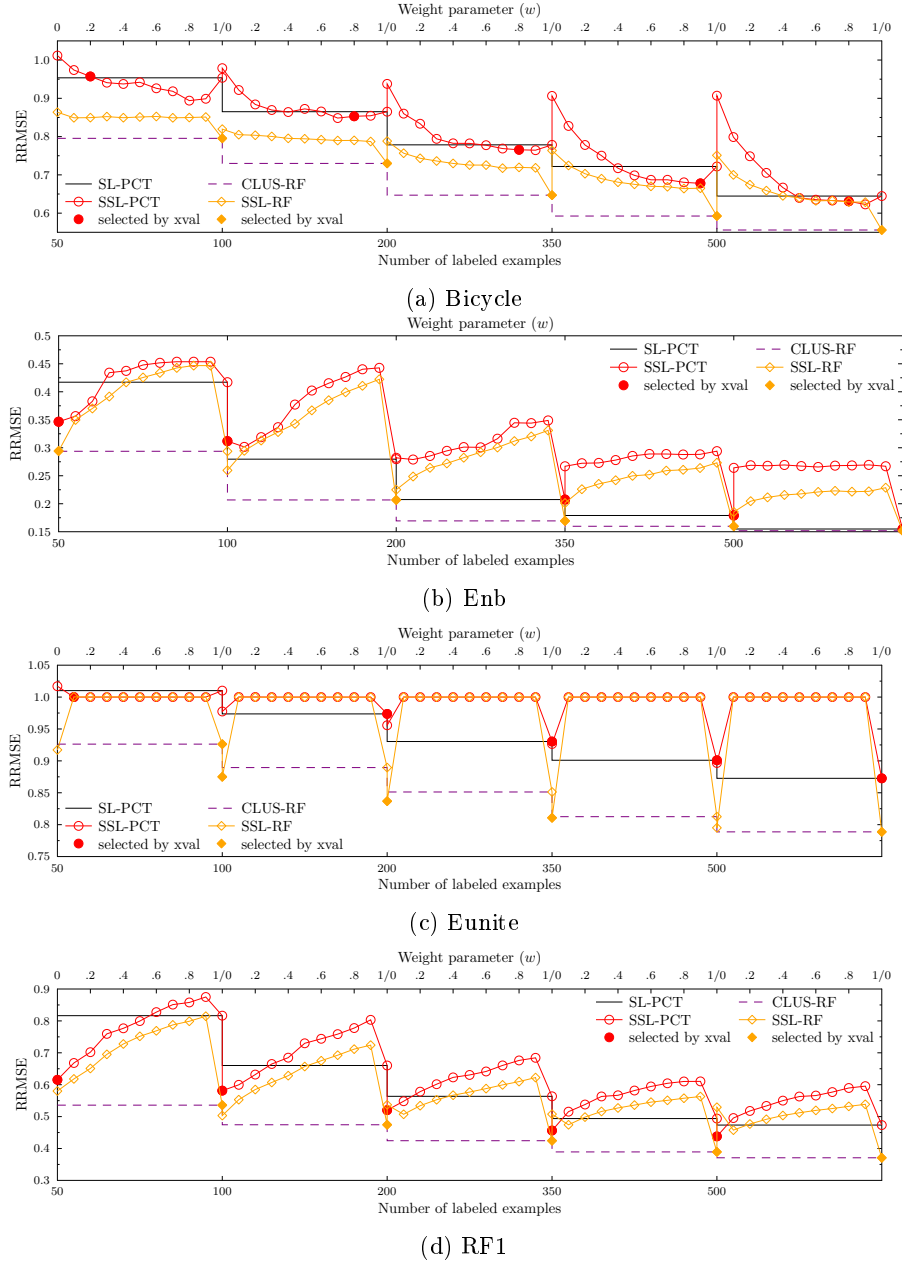
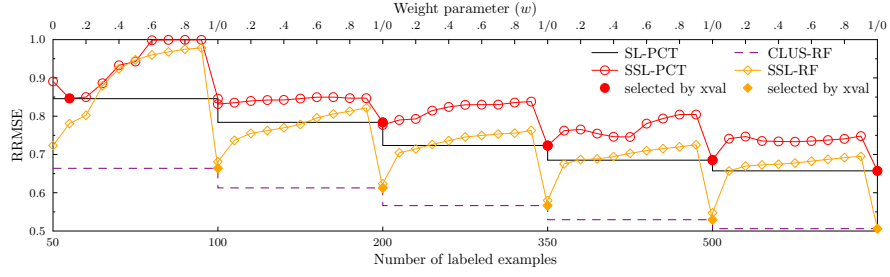
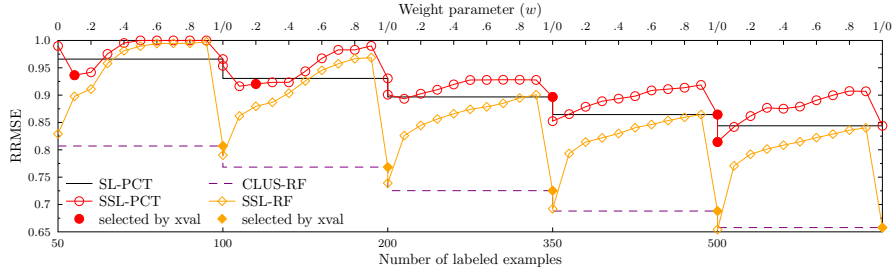


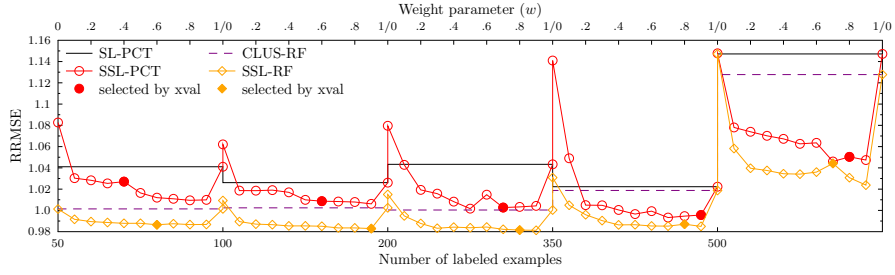
Figure B.4: The effect of the w parameter on the performance of the SSL-PCT (red line) and SSL-RF (orange line) methods on multi-target regression datasets. The values of the w parameter selected by cross-validation and used in the experimental evaluation are denoted with colored markers.



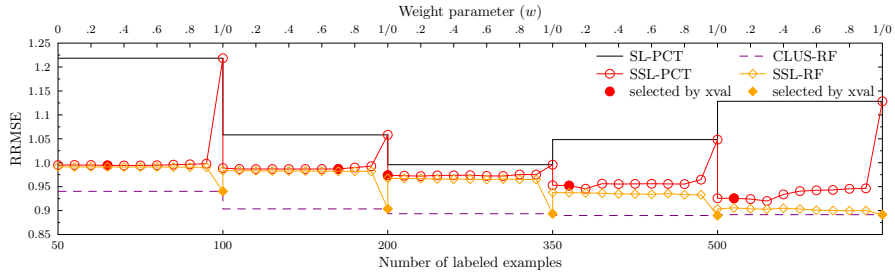
(e) SCM1D



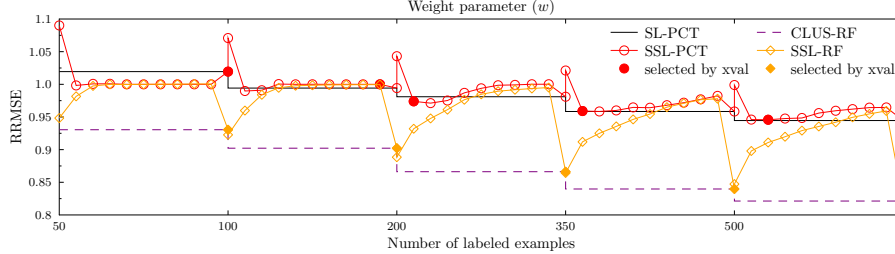
(f) SCM20D



(g) Solar flare 2

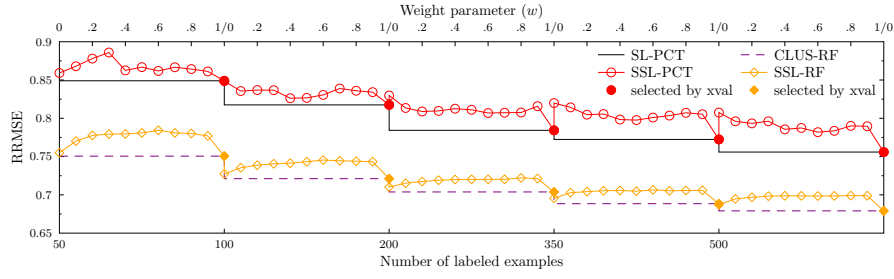


(h) Scpf

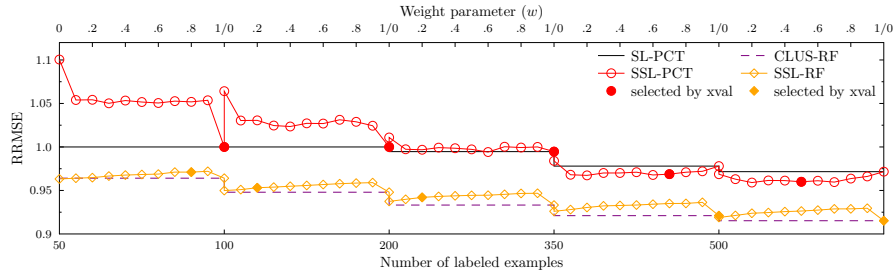


(i) Vegetation clustering

Figure B.4: (continued)

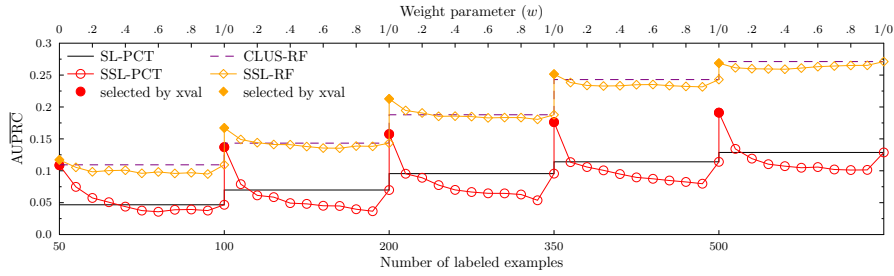


(j) Vegetation condition

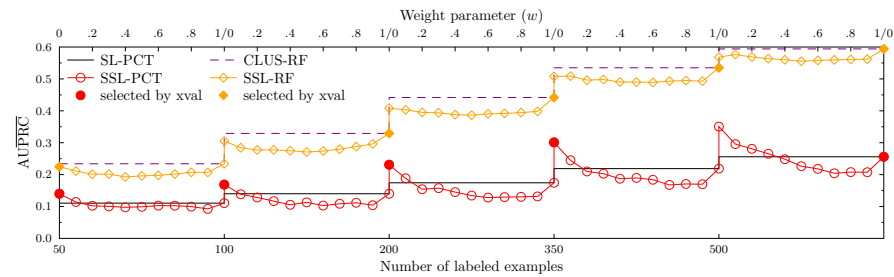


(k) Water quality

Figure B.4: (continued)

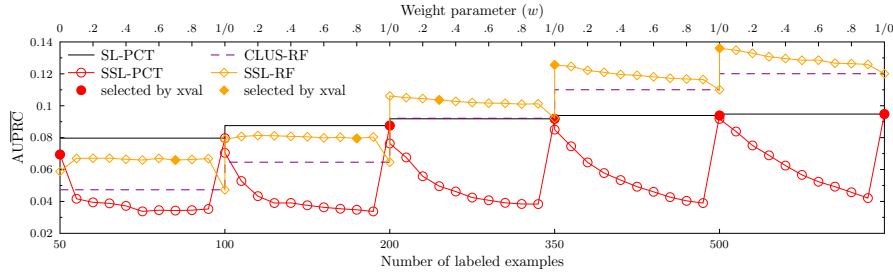


(a) Bibtex

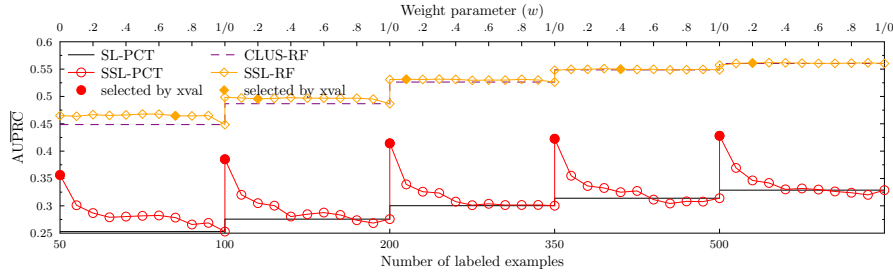


(b) Birds

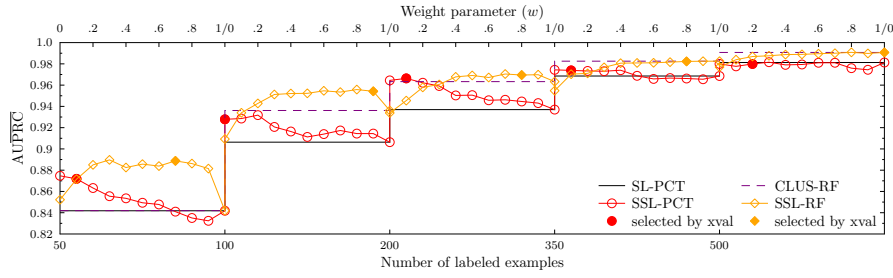
Figure B.5: The effect of the w parameter on the performance of the SSL-PCT (red line) and SSL-RF (orange line) methods on multi-label classification datasets. The values of the w parameter selected by cross-validation and used in the experimental evaluation are denoted with colored markers.



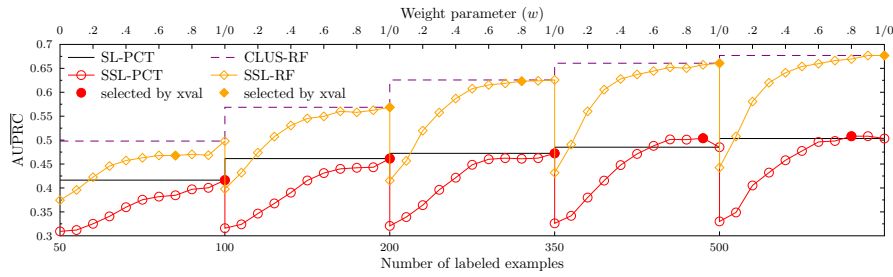
(c) Corel5k



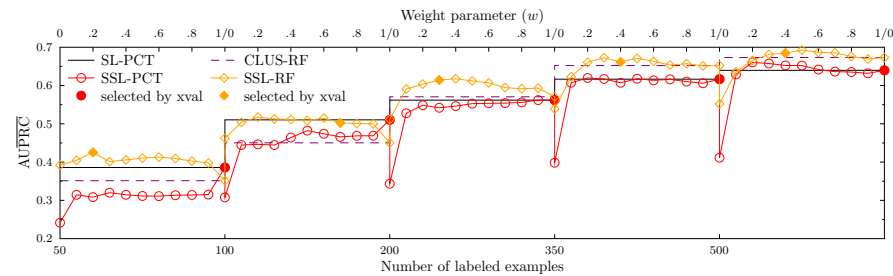
(d) Enron



(e) Genbase

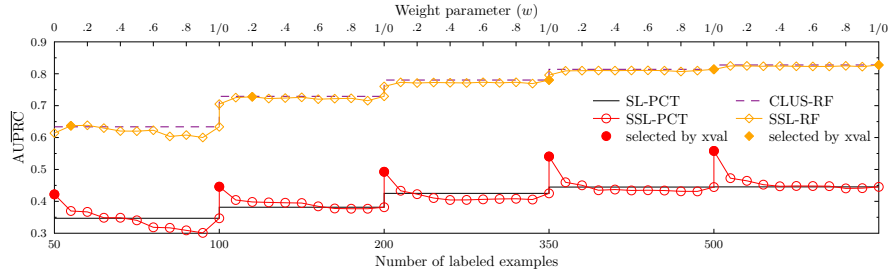


(f) Mediana

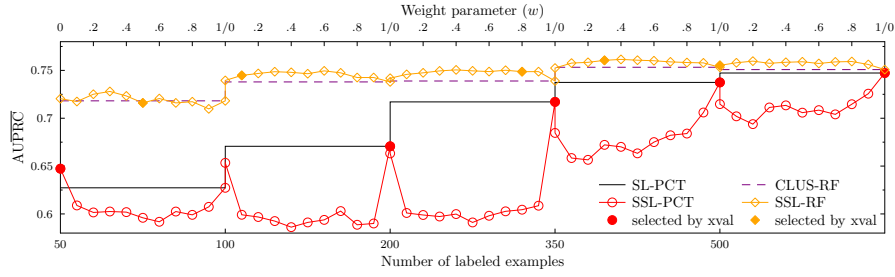


(g) Medical

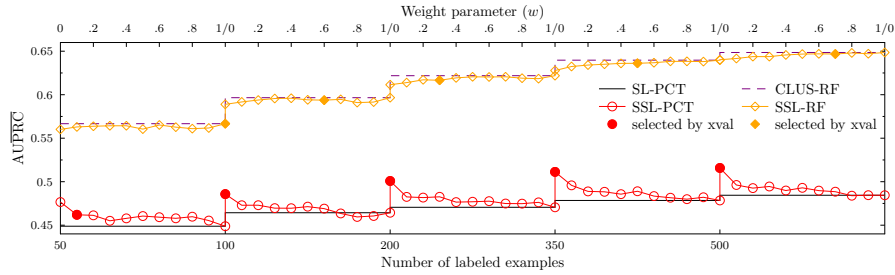
Figure B.5: (continued)



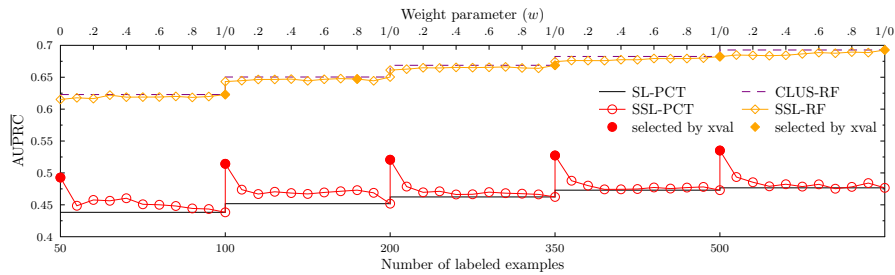
(h) Scene



(i) SIGMEA real

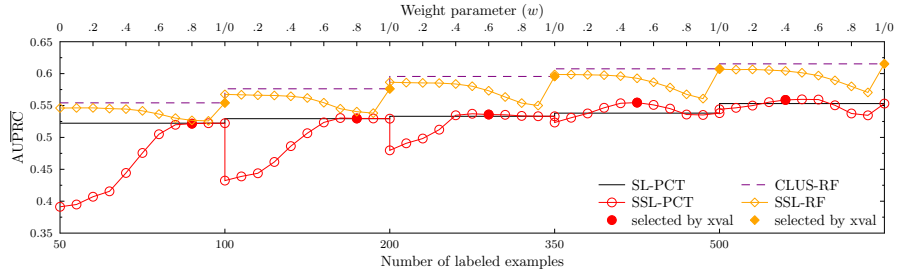


(j) Slovenian rivers

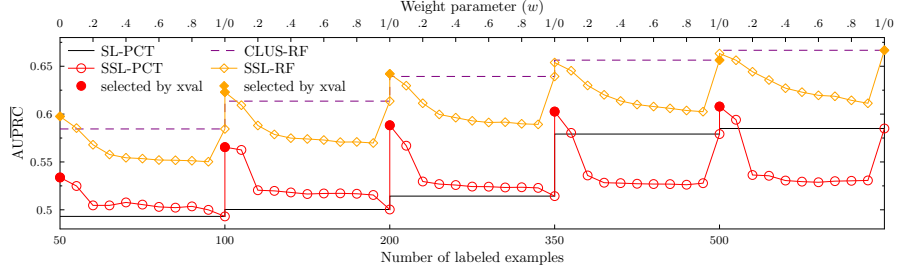


(k) Yeast

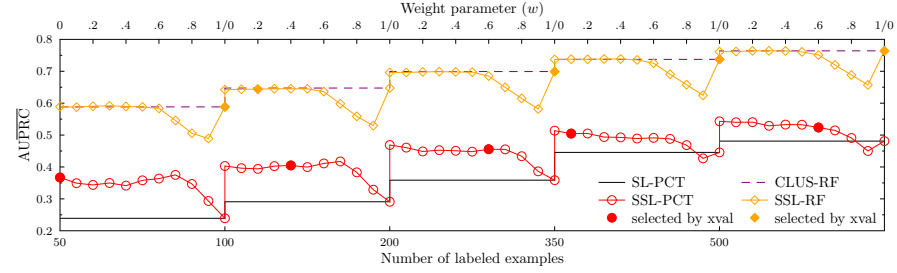
Figure B.5: (continued)



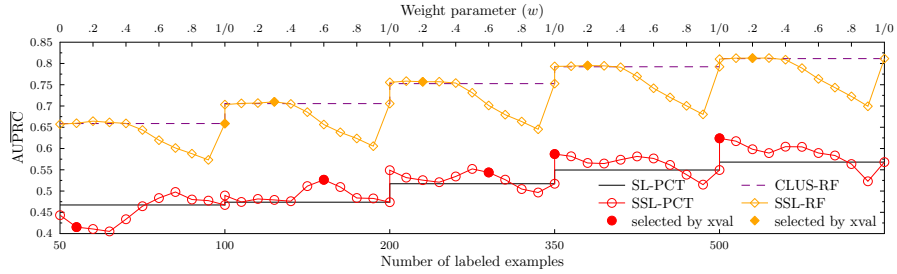
(a) Slovenian rivers



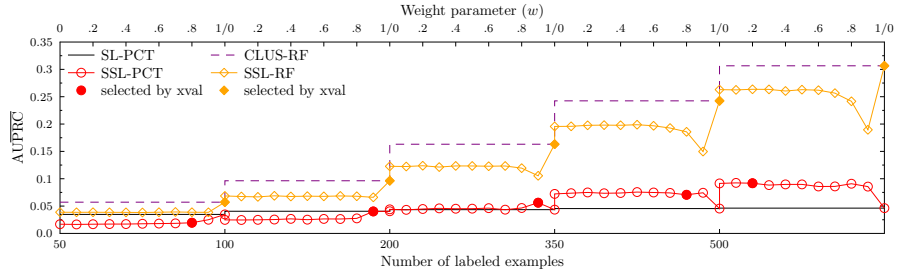
(b) Enron



(c) ImCLEF07A

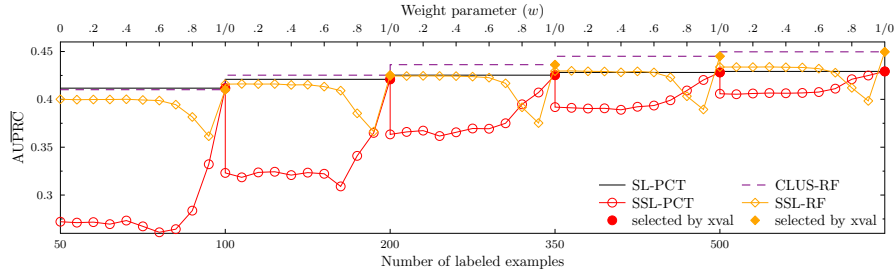


(d) ImCLEF07D

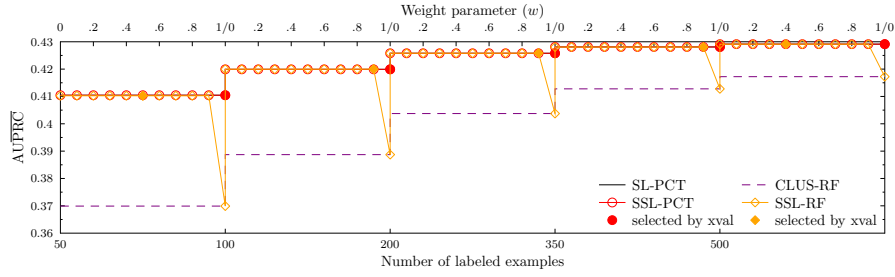


(e) Diatoms

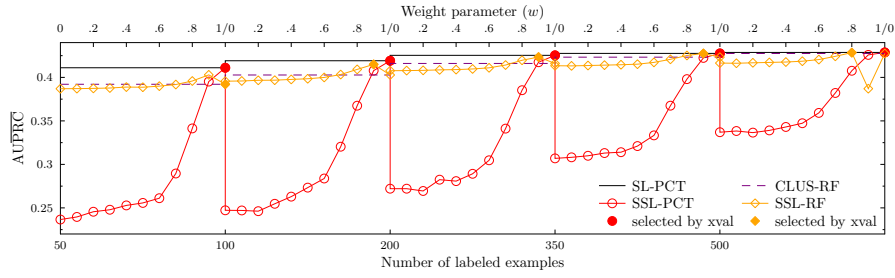
Figure B.6: The effect of the w parameter on the performance of the SSL-PCT (red line) and SSL-RF (orange line) methods on hierarchical multi-label classification datasets. The values of the w parameter selected by cross-validation and used in the experimental evaluation are denoted with colored markers.



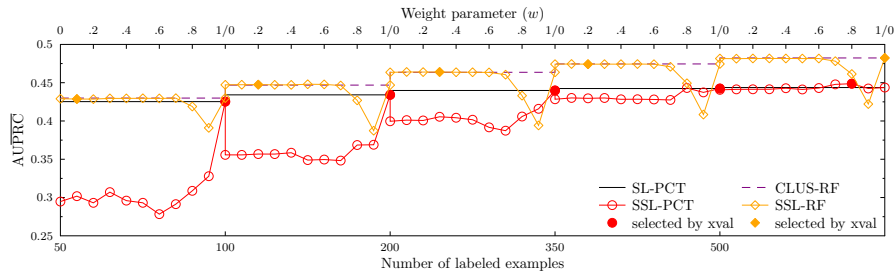
(a) Cellcycle-GO



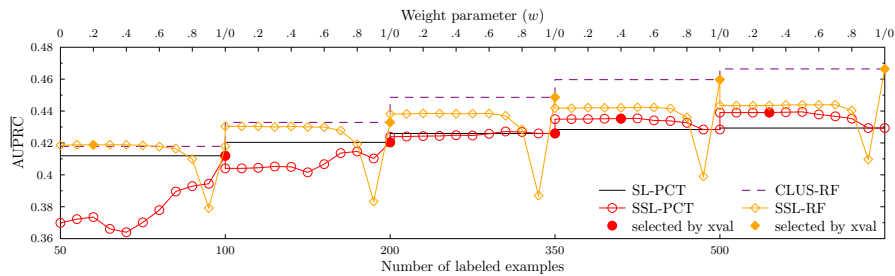
(b) Church-GO



(c) Derisi-GO



(d) Eisen-GO



(e) Expr-GO

Figure B.7: (continued)

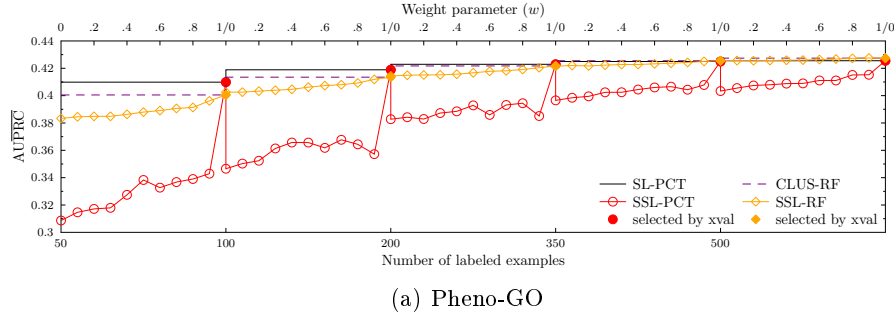


Figure B.8: (continued)

B.4 Statistical analysis of the predictive performances of unpruned trees

Table B.2: P -values of the Wilcoxon signed-rank test applied to the performances (AUPRC) of the supervised and semi-supervised single trees (SL-PCT, SSL-PCT and SSL-PCT-FR) on the datasets considered in this study: 12 for multi-target regression, 12 for multi-label classification and for 12 hierarchical multi-label classification. The methods are not pruned. In bold, we report significant p -values (< 0.05). In comparison of two methods, i.e., Method1 vs. Method2, the '-' sign denotes that the sum of ranks where the first method outperformed the second is higher than the sum of ranks where the second method outperformed the first. The '+' sign denotes the opposite.

Methods		Number of labeled examples				
		50	100	200	350	500
MULTI-TARGET REGRESSION						
SL-PCT	vs. SSL-PCT	0.003 (+)	0.01 (+)	0.004 (+)	0.023 (+)	0.023 (+)
SL-PCT	vs. SSL-PCT-FR	0.002 (+)	0.006 (+)	0.007 (+)	0.011 (+)	0.01 (+)
SSL-PCT	vs. SSL-PCT-FR	0.158 (+)	0.754 (+)	0.126 (-)	0.906 (+)	0.638 (+)
MULTI-LABEL CLASSIFICATION						
SL-PCT	vs. SSL-PCT	0.003 (+)	0.003 (+)	0.003 (+)	0.002 (+)	0.007 (+)
SL-PCT	vs. SSL-PCT-FR	0.007 (+)	0.003 (+)	0.003 (+)	0.002 (+)	0.01 (+)
SSL-PCT	vs. SSL-PCT-FR	0.078 (-)	0.196 (-)	0.108 (-)	0.875 (+)	0.126 (-)
HIERARCHICAL MULTI-LABEL CLASSIFICATION						
SL-PCT	vs. SSL-PCT	0.004 (+)	0.002 (+)	0.002 (+)	0.002 (+)	0.002 (+)
SL-PCT	vs. SSL-PCT-FR	0.005 (+)	0.002 (+)	0.002 (+)	0.002 (+)	0.002 (+)
SSL-PCT	vs. SSL-PCT-FR	0.255 (-)	0.308 (-)	0.29 (+)	0.272 (+)	0.034 (+)

References

- Abney, S. (2004). Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3), 365–395.
- Abraham, Z., Tan, P.-N., Perdinan, Winkler, J., Zhong, S., & Liszewska, M. (2013). Position preserving multi-output prediction. In H. Blockeel, K. Kersting, S. Nijssen, & F. Železný (Eds.), *Proceedings of the European Conference Machine Learning and Knowledge Discovery in Databases* (Vol. 8189, pp. 320–335). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Aho, T., Ženko, B., & Džeroski, S. (2009). Rule ensembles for multi-target regression. In *Proceedings of the 9th IEEE International Conference on Data Mining* (pp. 21–30). IEEE.
- Aho, T., Ženko, B., Džeroski, S., & Elomaa, T. (2012). Multi-target regression with rule ensembles. *Journal of Machine Learning Research*, 13(1), 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.
- Alaydie, N., Reddy, C. K., & Fotouhi, F. (2012). Exploiting label dependency for hierarchical multi-label classification. In *Proceedings of the 16th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining* (pp. 294–305). Springer Berlin Heidelberg.
- Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2010). Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3), 255–287.
- Altun, Y., McAllester, D., & Belkin, M. (2006). Maximum margin semi-supervised learning for structured variables. In *Proceedings of the Conference on the Advances In Neural Information Processing Systems* (Vol. 18, pp. 33–40).
- Appice, A., Ceci, M., & Malerba, D. (2009). An iterative learning algorithm for within-network regression in the transductive setting. In J. Gama, V. S. Costa, A. M. Jorge, & P. B. Brazdil (Eds.), *Proceedings of the 12th International Conference on Discovery Science* (Vol. 5808, pp. 36–50). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Appice, A. & Džeroski, S. [Sašo]. (2007). Stepwise induction of multi-target model trees. In J. N. Kok, J. Koronacki, R. L. d. Mantaras, S. Matwin, D. Mladenič, & A. Skowron (Eds.), *Proceedings of the 18th European Conference on Machine Learning* (Vol. 4701, pp. 502–509). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Bakir, G., Hofmann, T., Schölkopf, B., Smola, A., Taskar, B., & Vishwanathan, S. (2007). *Predicting structured data*. MIT Press.
- Bandouch, J., Jenkins, O. C., & Beetz, M. (2012). A self-training approach for visual tracking and recognition of complex human activity patterns. *International Journal of Computer Vision*, 99(2), 166–189.

- 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 H. Blockeel, K. Kersting, S. Nijssen, & F. Železný (Eds.), *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases* (Vol. 8189, pp. 385–400). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Barutcuoglu, Z., Schapire, R. E., & Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7), 830–836.
- Bauer, E. & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Machine Learning*, 36(1), 105–139.
- Bennett, K. & Demiriz, A. (1999). Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, 368–374.
- Bento, A. P., Gaulton, A., Hersey, A., Bellis, L. J., Chambers, J., Davies, M., . . . Overington, J. P. (2014). The ChEMBL bioactivity database: an update. *Nucleic Acids Research*, 42, D1083–D1090.
- Bi, W. & Kwok, J. T. (2015). Bayes-optimal hierarchical multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 27(11), 2907–2918.
- Biber, E. (2013). The challenge of collecting and using environmental monitoring data. *Ecology and Society*, 18(4).
- Bishop, C. (2007). *Pattern recognition and machine learning (information science and statistics)*. Springer.
- Blockeel, H. (1998). *Top-down induction of first order logical decision trees* (PhD Thesis, Katholieke Universiteit Leuven, Leuven, Belgium).
- Blockeel, H., Bruynooghe, M., Džeroski, S., Ramon, J., & Struyf, J. (2002). Hierarchical multi-classification. In *Proceedings of the ACM SIGKDD Workshop on Multi-relational Data Mining* (pp. 21–35).
- Blockeel, H., De Raedt, L., & Ramon, J. (1998). Top-down induction of clustering trees. In *Proceeding of the 15th International Conference on Machine learning* (pp. 55–63). Morgan Kaufmann, San Francisco, CA.
- Blockeel, H., Džeroski, S., & Grbović, J. (1999). Simultaneous prediction of multiple chemical parameters of river water quality with TILDE. In *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery* (Vol. 1704, pp. 32–40). Lecture Notes in Computer Science.
- Blockeel, H., Schietgat, L., Struyf, J., Džeroski, S., & Clare, A. (2006). Decision trees for hierarchical multilabel classification: a case study in functional genomics. In J. Fürnkranz, T. Scheffer, & M. Spiliopoulou (Eds.), *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases* (Vol. 4213, pp. 18–29). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Blum, A. & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory* (pp. 92–100). ACM New York, NY.
- Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5), 216–233.
- Bosnić, Z. & Kononenko, I. (2008). Comparison of approaches for estimating reliability of individual regression predictions. *Data and Knowledge Engineering*, 67(3), 504–516.
- Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, 37(9), 1757–1771.
- Brefeld, U., Büscher, C., & Scheffer, T. (2005). Multi-view discriminative sequential learning. In J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, & L. Torgo (Eds.), *Proceed-*

- ings of the 16th European Conference on Machine Learning* (Vol. 3720, pp. 60–71). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Brefeld, U., Gärtner, T., Scheffer, T., & Wrobel, S. (2006). Efficient co-regularised least squares regression. In *Proceedings of the 23rd International Conference on Machine learning* (pp. 137–144). ACM New York, NY.
- Brefeld, U. & Scheffer, T. (2006). Semi-supervised learning for structured output variables. In *Proceedings of the 23rd International Conference on Machine learning* (pp. 145–152).
- Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 24(2), 123–140.
- Breiman, L. (1996b). *Out-of-bag estimation*. University of California.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- 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.
- Breiman, L., Friedman, J., Olshen, R., & Stone, C. J. (1984). *Classification and regression trees*. Wadsworth & Brooks, Monterey, CA.
- Briesemeister, S., Rahnenführer, J., & Kohlbacher, O. (2012). No longer confidential: estimating the confidence of individual regression predictions. *PLoS ONE*, 7(11), e48723.
- Briggs, F., Huang, Y., Raich, R., Eftaxias, K., Lei, Z., Cukierski, W., ... Milakov, M. (2013). The 9th annual MLSP competition: new methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing* (pp. 1–8). IEEE.
- 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 preprint arXiv:1405.3564*.
- Brown, P. J. & Zidek, J. V. (1980). Adaptive multivariate ridge regression. *The Annals of Statistics*, 8, 64–74.
- Bühlmann, P. (2012). Bagging, boosting and ensemble methods. In *Handbook of computational statistics* (pp. 985–1022). Springer Berlin Heidelberg.
- Cai, F. & Cherkassky, V. (2009). SVM+ regression and multi-task learning. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 418–424). IEEE.
- Cardona, H. D. V., Álvarez, M. A., & Orozco, Á. A. (2015). Convolved multi-output gaussian processes for semi-supervised learning. In V. Murino & E. Puppo (Eds.), *Proceedings of the International Conference Image Analysis and Processing* (Vol. 9279, pp. 109–118). Lecture Notes in Computer Science. Springer International Publishing.
- Carney, J. G., Cunningham, P., & Bhagwan, U. (1999). Confidence and prediction intervals for neural network ensembles. In *Proceeding of the International Joint Conference on Neural Networks* (Vol. 2, pp. 1215–1218). IEEE.
- Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1), 41–75.
- Ceci, M. (2008). Hierarchical text categorization in a transductive setting. In *Proceedings of the 8th IEEE International Conference on Data Mining Workshops* (pp. 184–191). IEEE.
- Ceci, M., Appice, A., Viktor, H. L., Malerba, D., Paquet, E., & Guo, H. (2012). Transductive relational classification in the co-training paradigm. In P. Perner (Ed.), *Proceedings of the 8th International Conference on Machine Learning and Data Mining*

- in *Pattern Recognition* (Vol. 7376, pp. 11–25). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Cerri, R., Barros, R. C., & de Carvalho, A. C. P. L. F. (2012). A genetic algorithm for hierarchical multi-label classification. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (pp. 250–255). ACM New York, NY.
- Cerri, R., Barros, R. C., & de Carvalho, A. C. P. L. F. (2014). Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, 80(1), 39–56.
- Cerri, R., Pappa, G. L., Carvalho, A. C. P., & Freitas, A. A. (2013). An extensive evaluation of decision tree based hierarchical multilabel classification methods and performance measures. *Computational Intelligence*, 1–46.
- Chang, M.-W., Ratnov, L., & Roth, D. (2012). Structured learning with constrained conditional models. *Machine Learning*, 88(3), 399–431.
- Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Cambridge, Massachusetts: MIT Press.
- Chapelle, O., Sindhwani, V., & Keerthi, S. S. (2008). Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9, 203–233.
- Chawla, N. & Karakoulas, G. (2005). Learning from labeled and unlabeled data: an empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, 23(1), 331–366.
- 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 International Joint Conference on Neural Networks* (pp. 1458–1463). IEEE.
- Chen, G., Song, Y., Wang, F., & Zhang, C. (2008). Semi-supervised multi-label learning by solving a Sylvester equation. In *Proceedings of the SIAM international conference on data mining* (pp. 410–419). Society for Industrial and Applied Mathematics.
- Chen, B.-J., Chang, M.-W., & Lin, C.-J. (2004). Load forecasting using support vector machines: a study on EUNITE competition 2001. *IEEE Transactions on Power Systems*, 19(4), 1821–1830.
- 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.
- 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.
- Cheng, W. & Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2), 211–225.
- 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.
- Clare, A. (2003). *Machine learning and data mining for yeast functional genomics* (Doctoral dissertation, University of Wales Aberystwyth, Aberystwyth, United Kingdom).
- Clare, A. & King, R. D. [Ross D.]. (2001). Knowledge discovery in multi-label phenotype data. In L. De Raedt & A. Siebes (Eds.), *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery* (Vol. 2168, pp. 42–53). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Clare, A. & King, R. D. [Ross D.]. (2003). Predicting gene function in *Saccharomyces cerevisiae*. *Bioinformatics*, 19(S2), ii42–ii49.

- Cozman, F., Cohen, I., & Cirelo, M. (2002). Unlabeled data can degrade classification performance of generative classifiers. In *Proceedings of the 15th International Florida Artificial Intelligence Research Society Conference* (pp. 327–331).
- Crammer, K. & Singer, Y. (2003). A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3(6), 1025–1058.
- Cronin, M. T. D. (2010). Quantitative structure–activity relationships (QSARs) – applications and methodology. In T. Puzyn, J. Leszczynski, & M. T. Cronin (Eds.), *Recent advances in QSAR studies* (8, pp. 3–11). Springer Netherlands.
- Culp, M. & Michailidis, G. (2003). An iterative algorithm for extending learners to a semi-supervised setting. *Journal of Computational and Graphical Statistics*, 17(3), 545–571.
- Cunningham, P. & Delany, S. J. (2007). K-nearest neighbour classifiers. *Multiple Classifier Systems*, 34, 1–17.
- d Lucena, D. C. G. & Prudencio, R. B. C. (2015). Semi-supervised multi-label k-nearest neighbors classification algorithms. In *Proceedings of the Brazilian Conference on Intelligent Systems* (pp. 49–54). IEEE.
- D’Ambrosio, A., Aria, M., Iorio, C., & Siciliano, R. (2017). Regression trees for multivalued numerical response variables. *Expert Systems with Applications*, 69, 21–28.
- Dara, R., Kremer, S. C., Stacey, D., et al. (2002). Clustering unlabeled data with SOMs improves classification of labeled real-world data. In *Proceedings of the 15th International Joint Conference on Neural Networks* (Vol. 3, pp. 2237–2242). IEEE.
- Davis, J. & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine learning* (pp. 233–240). ACM New York, NY.
- De Comité, F., Gilleron, R., & Tommasi, M. (2003). Learning multi-label alternating decision trees from texts and data. In *Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition* (pp. 35–49). Springer-Verlag Berlin, Heidelberg.
- De’Ath, G. (2002). Multivariate regression trees: a new technique for modeling species–environment relationships. *Ecology*, 83(4), 1105–1117.
- Demiriz, A., Bennett, K. P., & Embrechts, M. J. (1999). Semi-supervised clustering using genetic algorithms. In *Proceedings of the 5th Conference on Artificial Neural Networks in Engineering* (pp. 809–814).
- Demšar, D., Debeljak, M., Lavigne, C., & Džeroski, S. (2005). Modelling pollen dispersal of genetically modified oilseed rape within the field. In *Proceedings of the Annual Meeting of the Ecological Society of America* (p. 152).
- Demšar, D., Džeroski, S., Larsen, T., Struyf, J., Axelsen, J., Pedersen, M., & Krogh, P. (2006). Using multi-objective classification to model communities of soil microarthropods. *Ecological Modelling*, 191(1), 131–143.
- Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7, 1–30.
- 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* (pp. 957–966). ACM New York, NY.
- Dhillon, P., Keerthi, S., Bellare, K., Chapelle, O., & Sellamanickam, S. (2012). Deterministic annealing for semi-supervised structured output learning. In N. D. Lawrence & M. Girolami (Eds.), *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics* (Vol. 22, pp. 299–307). Proceedings of Machine Learning Research.

- Dietterich, T. G., Domingos, P., Getoor, L., Muggleton, S., & Tadepalli, P. (2008). Structured machine learning: the next ten years. *Machine Learning*, 73(1), 3–23.
- DiMasi, J. A., Hansen, R. W., & Grabowski, H. G. (2003). The price of innovation: new estimates of drug development costs. *Journal of Health Economics*, 22(2), 151–185.
- Dimitrovski, I., Kocev, D., Loskovska, S., & Džeroski, S. (2008). Hierarchical annotation of medical images. In *Proceedings of the 11th International Multiconference Information Society* (pp. 174–181). Jožef Stefan Institute, Ljubljana.
- Dimitrovski, I., Kocev, D., Loskovska, S., & Džeroski, S. (2012). Hierarchical classification of diatom images using ensembles of predictive clustering trees. *Ecological Informatics*, 7(1), 19–29.
- Ding, S., Zhu, Z., & Zhang, X. (2015). An overview on semi-supervised support vector machine. *Neural Computing and Applications*, 1–10.
- Diplaris, S., Tsoumakas, G., Mitkas, P. A., & Vlahavas, I. (2005). Protein classification with multiple algorithms. In P. Bozanis & E. N. Houstis (Eds.), *Proceedings of the 10th Panhellenic Conference on Informatics* (Vol. 3746, pp. 448–456). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Du, X. (2017). Semi-supervised learning of local structured output predictors. *Neurocomputing*, 220, 151–159.
- Duygulu, P., Barnard, K., de Freitas, J. F. G., & Forsyth, D. A. (2002). Object recognition as machine translation: learning a lexicon for a fixed image vocabulary. In A. Heyden, G. Sparr, M. Nielsen, & P. Johansen (Eds.), *Proceedings of the 7th European Conference on Computer Vision* (pp. 97–112). Springer Berlin Heidelberg.
- Džeroski, S. [Sašo]. (2007). Towards a general framework for data mining. In *Proceedings of the 5th International Workshop on Knowledge Discovery in Inductive Databases* (pp. 259–300). Springer Berlin Heidelberg.
- Džeroski, S. [Sašo]. (2009). Machine learning applications in habitat suitability modeling. In S. E. Haupt, A. Pasini, & C. Marzban (Eds.), *Artificial Intelligence Methods in the Environmental Sciences* (pp. 397–412). Springer Netherlands, Dordrecht.
- Džeroski, S. [Sašo], Demšar, D., & Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, 13(1), 7–17.
- Džimbeg, G., Zorc, B., Kralj, M., Ester, K., Pavelić, K., Andrei, G., . . . Mintas, M. (2008). The novel primaquine derivatives of n-alkyl, cycloalkyl or aryl urea: synthesis, cytostatic and antiviral activity evaluations. *European Journal of Medicinal Chemistry*, 43(6), 1180–1187.
- Elisseeff, A. & Weston, J. (2001). A kernel method for multi-labelled classification. In *In Proceedings of the Conference on Advances in Neural Information Processing Systems* (pp. 681–687). MIT Press.
- Enrique Sucar, L., Bielza, C., Morales, E. F., Hernandez-Leal, P., Zaragoza, J. H., & Larrañaga, P. (2014). Multi-label classification with Bayesian network-based chain classifiers. *Pattern Recognition Letters*, 41, 14–22.
- Fanaee-T, H. & Gama, J. (2013). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 1–15.
- Farouk Abdel Hady, M. & Schwenker, F. (2010). Combining committee-based semi-supervised learning and active learning. *Journal of Computer Science and Technology*, 25(4), 681–698.
- Ford, E. S. (1999). Body mass index, diabetes, and c-reactive protein among us adults. *Diabetes care*, 22(12), 1971–1977.
- Freund, Y. & Mason, L. (1999). The alternating decision tree learning algorithm. In *Proceedings of the International Conference on Machine Learning* (Vol. 99, pp. 124–133). Morgan Kaufmann Publishers Inc., San Francisco, CA.

- Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3), 133–168.
- Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2), 133–153.
- Geurts, P., Wehenkel, L., & D’Alché-Buc, F. (2006). Kernelizing the output of tree-based methods. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 345–352). ACM New York, NY.
- Gibaja, E. & Ventura, S. (2015). A Tutorial on Multilabel Learning. *ACM Computing Surveys*, 47(3), 1–38.
- Gillberg, J., Marttinen, P., Pirinen, M., Kangas, A. J., Soininen, P., Ali, M., ... Kaski, S. (2016). Multiple output regression with latent noise. *Journal of Machine Learning Research*, 17(122), 1–35.
- Gjorgjevikj, D., Madjarov, G., & Džeroski, S. (2013). Hybrid decision tree architecture utilizing local SVMs for efficient multi-label learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(7).
- Gjorgjioski, V. & Džeroski, S. [S.]. (2003). *Clustering analysis of vegetation data*. Jožef Stefan Institute.
- Goldberg, A. B. [Andrew B], Zhu, X., Singh, A., Xu, Z., & Nowak, R. (2009). Multi-manifold semi-supervised learning. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics* (pp. 169–176).
- Gonçalves, T. & Quaresma, P. (2004). Using IR techniques to improve automated text classification. In F. Mezziane & E. Métais (Eds.), *Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems* (Vol. 3136, pp. 374–379). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Gönen, M. & Kaski, S. (2014). Kernelized Bayesian Matrix Factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(10), 2047–2060.
- Guan, Y., Myers, C. L., Hess, D. C., Barutcuoglu, Z., Caudy, A., & Troyanskaya, O. (2008). Predicting gene function in a hierarchical context with an ensemble of classifiers. *Genome biology*, 9(1), S3.
- Guo, Y. [Yuanyuan], Niu, X., & Zhang, H. (2010). An extensive empirical study on semi-supervised learning. In *Proceedings of the 10th IEEE International Conference on Data Mining* (pp. 186–195). IEEE.
- Guo, Y. [Yuhong] & Schuurmans, D. (2012). Semi-supervised multi-label classification: a simultaneous large-margin, subspace learning approach. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 355–370). Springer-Verlag Berlin, Heidelberg.
- Guo-Zheng, L., Jack, Y. Y., Wen-Cong, L., Dan, L., & Mary, Q. Y. (2008). Improving prediction accuracy of drug activities by utilizing unlabeled instances with feature selection. *International Journal of Computational Biology and Drug Design*, 1(1), 1–13.
- Guyon, I., Gunn, S., Ben-Hur, A., & Dror, G. (2004). Result analysis of the NIPS 2003 feature selection challenge. In *Proceedings of the 17th International Conference on Neural Information Processing Systems* (pp. 545–552). MIT Press Cambridge, MA.
- Haffari, G. & Sarkar, A. (2007). Analysis of semi-supervised learning with the Yarowsky algorithm. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence* (pp. 159–176). AUAI Press, Corvallis, Oregon.
- Han, Z., Liu, Y., Zhao, J., & Wang, W. (2012). Real time prediction for converter gas tank levels based on multi-output least square support vector regressor. *Control Engineering Practice*, 20(12), 1400–1409.

- Higuera, C., Gardiner, K. J., & Cios, K. J. (2015). Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLoS ONE*, 10(6), e0129126.
- Hoerl, A. E. & Kennard, R. W. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 55–67.
- Hu, C. & Kwok, J. T. (2010). Manifold regularization for structured outputs via the joint kernel. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1–8). IEEE.
- 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.
- 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.
- Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning* (pp. 200–209). Morgan Kaufmann Publishers Inc., San Francisco, CA.
- Joachims, T. (2006). Structured output prediction with support vector machines. In D.-Y. Yeung, J. T. Kwok, A. Fred, F. Roli, & D. de Ridder (Eds.), *Structural, Syntactic, and Statistical Pattern Recognition* (4109, pp. 1–7). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Kawakita, M. & Takeuchi, J. (2014). Safe semi-supervised learning based on weighted likelihood. *Neural Networks*, 53, 146–164.
- Kim, M. (2013). Semi-supervised learning of hidden conditional random fields for time-series classification. *Neurocomputing*, 119, 339–349.
- Kiritchenko, S., Famili, F., Matwin, S., & Nock, R. (2006). Learning and evaluation in the presence of class hierarchies: application to text categorization. In L. Lamontagne & M. Marchand (Eds.), *Proceedings of the 19th International Conference on Advances in Artificial Intelligence* (Vol. 4013, pp. 395–406). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Klimt, B. & Yang, Y. [Yiming]. (2004). The Enron corpus: a new dataset for email classification research. In J.-F. Boulicaut, F. Esposito, F. Giannotti, & D. Pedreschi (Eds.), *Proceedings of the 18th European Conference on Machine Learning* (Vol. 3201, pp. 217–226). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Kocev, D. (2011). *Ensembles for predicting structured outputs* (Doctoral dissertation, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia).
- Kocev, D., Džeroski, S., White, M. D., Newell, G. R., & Griffioen, P. (2009). Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modelling*, 220(8), 1159–1168.
- Kocev, D., Slavkov, I., & Džeroski, S. (2013). Feature ranking for multi-label classification using predictive clustering trees. In *Proceedings of the International Workshop on Solving Complex Machine Learning Problems with Ensemble Methods* (pp. 56–68).
- Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2007). Ensembles of multi-objective decision trees. In *Proceedings of the 18th European Conference on Machine Learning* (pp. 624–631). Springer Berlin Heidelberg.
- Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3), 817–833.
- 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.

- Kriegel, H.-P., Borgwardt, K., Kröger, P., Pryakhin, A., Schubert, M., & Zimek, A. (2007). Future trends in data mining. *Data Mining and Knowledge Discovery*, 15, 87–97.
- Kuhn, M. & Johnson, K. (2013). *Applied predictive modeling*. New York: Springer.
- Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning* (Vol. 1, pp. 282–289). Morgan Kaufmann Publishers Inc., San Francisco, CA.
- Langley, P. (1996). *Elements of machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Lehmann, T., Schubert, H., Keysers, D., Kohnen, M., & Wein, B. (2003). The IRMA code for unique classification of medical images. In *Proceedings of the SPIE Conference on Medical Imaging: PACS and Integrated Medical Information Systems: Design and Evaluation* (Vol. 5033, pp. 440–451).
- Leistner, C., Saffari, A., Santner, J., & Bischof, H. (2009). Semi-supervised random forests. In *Proceedings of the 12th International Conference on Computer Vision* (pp. 506–513). IEEE.
- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2014). Semi-supervised learning for multi-target regression. In *Proceedings of the 3rd International ECML-PKDD Workshop on New Frontiers in Mining Complex Patterns* (pp. 110–123).
- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2015a). Semi-supervised learning for multi-target regression. In A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, & Z. W. Ras (Eds.), *Third International ECML-PKDD Workshop on New Frontiers in Mining Complex Patterns, Revised selected papers* (Vol. 8983, pp. 3–18). Lecture Notes in Computer Science. Springer International Publishing.
- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2015b). Semi-supervised learning for multi-target regression. In *Proceedings of the 23rd Italian Symposium on Advanced Database Systems* (pp. 249–255).
- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2017a). Self-training for multi-target regression with tree ensembles. *Knowledge-Based Systems*, 123, 41–60.
- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2017b). Semi-supervised classification trees. *Journal of Intelligent Information Systems*. In press.
- Levatić, J., Čurak, J., Kralj, M., Šmuc, T., Osmak, M., & Supek, F. (2013). Accurate models for p-gp drug recognition induced from a cancer cell line cytotoxicity screen. *Journal of Medicinal Chemistry*, 56(14), 5691–5708.
- Levatić, J., Džeroski, S. [Sašo], Supek, F., & Šmuc, T. (2013). Semi-supervised learning for quantitative structure-activity modeling. *Informatika (Slovenia)*, 37(2), 173–179.
- Levatić, J., Kocev, D., Ceci, M., & Džeroski, S. (2017). *Semi-supervised trees for multi-target regression*. Under review.
- Levatić, J., Kocev, D., Debeljak, M., & Džeroski, S. (2015). Community structure models are improved by exploiting taxonomic rank with predictive clustering trees. *Ecological Modelling*, 306, 294–304.
- Levatić, J., Kocev, D., & Džeroski, S. [Sašo]. (2014). The importance of the label hierarchy in hierarchical multi-label classification. *Journal of Intelligent Information Systems*, 45(2), 247–271.
- Levatić, J., Supek, F., & Džeroski, S. (2015). Improving QSAR models by exploiting unlabeled data from public databases of bioactive drug-like molecules. In *Proceedings of the 7th Jožef Stefan International Postgraduate School Students' Conference* (pp. 114–123). Best ICT paper award.
- Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5, 361–397.

- Li, Y.-F., Zha, H.-W., & Zhou, Z.-H. (2017). Learning safe prediction for semi-supervised regression. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.
- Li, Y. & Zemel, R. (2014). High order regularization for semi-supervised learning of structured output problems. In *Proceedings of the 31st International Conference on Machine Learning* (pp. 1368–1376). PMLR.
- Lichman, M. (2013). UCI machine learning repository. Retrieved from <http://archive.ics.uci.edu/ml>
- Liu, W., Wang, J., & Chang, S.-F. (2012). Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100(9), 2624–2638.
- Loza Mencía, E. & Janssen, F. (2016). Learning rules for multi-label classification: a stacking and a separate-and-conquer approach. *Machine Learning*, 105(1), 77–126.
- Loza Mencía, E., Park, S.-H., & Fürnkranz, J. (2010). Efficient voting prediction for pairwise multilabel classification. *Neurocomputing*, 73(7–9), 1164–1176.
- 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.
- Madjarov, G., Gjorgjevikj, D., & Džeroski, S. [Sašo]. (2012). Two stage architecture for multi-label learning. *Pattern Recognition*, 45(3), 1019–1034.
- 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.
- Malerba, D., Ceci, M., & Appice, A. (2009). A relational approach to probabilistic classification in a transductive setting. *Engineering Applications of Artificial Intelligence*, 22(1), 109–116.
- Mansouri, K., Ringsted, T., Ballabio, D., Todeschini, R., & Consonni, V. (2013). Quantitative structure–activity relationship models for ready biodegradability of chemicals. *Journal of Chemical Information and Modeling*, 53(4), 867–878.
- Mitchell, T. (1997). *Machine learning*. Boston, Massachusetts: McGraw Hill.
- Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6), 47–60.
- Moro, S., Laureano, R., & Cortez, P. (2011). Using data mining for bank direct marketing: an application of the crisp-dm methodology. In *Proceedings of the 25th European Simulation and Modelling Conference* (pp. 117–121).
- Mostafavi, S., Debajyoti, R., Warde-Farley, D., Grouios, C., & Morris, Q. (2008). GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome biology*, 9(S1), S4+.
- Navaratnam, R., Fitzgibbon, A., & Cipolla, R. (2007). The joint manifold model for semi-supervised multi-valued regression. In *Proceedings of the 11th IEEE International Conference on Computer Vision* (pp. 1–8). IEEE.
- Nemenyi, P. B. (1963). *Distribution-free multiple comparisons* (Doctoral dissertation, Princeton University, Princeton, NY, USA).
- Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3), 103–134.
- Nikolova, N. & Jaworska, J. (2003). Approaches to measure chemical similarity - a review. *QSAR & Combinatorial Science*, 22(9), 1006–1026.
- Obozinski, G., Lanckriet, G., Grant, C., Jordan, M. I., & Noble, W. S. (2008). Consistent probabilistic outputs for protein function prediction. *Genome Biology*, 9(1), S6.
- 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* (pp. 74–79). IEEE.

- Al-Otaibi, R., Kull, M., & Flach, P. (2016). Declaratively capturing local label correlations with multi-label trees. In *Proceedings of the 22nd European Conference on Artificial Intelligence* (pp. 1467–1475).
- 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.
- Park, S.-H. & Fürnkranz, J. (2007). Efficient pairwise classification. In J. N. Kok, J. Koronacki, R. L. d. Mantaras, S. Matwin, D. Mladenič, & A. Skowron (Eds.), *Proceedings of the european Conference on Machine Learning* (Vol. 4701, pp. 658–665). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Pavić, K., Perković, I., Cindrić, M., Pranjić, M., Martin-Kleiner, I., Kralj, M., ... Zorc, B. (2014). Novel semicarbazides and ureas of primaquine with bulky aryl or hydroxyalkyl substituents: synthesis, cytostatic and antioxidative activity. *European Journal of Medicinal Chemistry*, 86, 502–514.
- Perković, I., Tršinar, S., Žanetić, J., Kralj, M., Martin-Kleiner, I., Balzarini, J., ... Zorc, B. (2012). Novel 1-acyl-4-substituted semicarbazide derivatives of primaquine - synthesis, cytostatic, antiviral and antioxidative studies. *Journal of Enzyme Inhibition and Medicinal Chemistry*, 28(3), 601–610.
- Pugelj, M. & Džeroski, S. [Sašo]. (2011). Predicting structured outputs k-nearest neighbours method. In T. Elomaa, J. Hollmén, & H. Mannila (Eds.), *Proceedings of the 14th International Conference on Discovery Science* (Vol. 6926, pp. 262–276). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Quinlan, R. J. (1993). *C4.5: programs for machine learning* (1st ed.). Morgan Kaufmann.
- Raileanu, L. E. & Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1), 77–93.
- Rajić, Z., Končić, M., Miloloža, K., Perković, I., Butula, I., Bucar, F., & Zorc, B. (2010). Primaquine-NSAID twin drugs: synthesis, radical scavenging, antioxidant and fe2+ chelating activity. *Acta Pharmaceutica*, 60(3), 325–337.
- Read, J. (2008). A pruned problem transformation method for multi-label classification. In *Proceedings of the New Zealand Computer Science Research Student Conference* (pp. 143–150).
- Read, J., Martino, L., Olmos, P. M., & Luengo, D. (2015). Scalable multi-output label prediction: From classifier chains to classifier trellises. *Pattern Recognition*, 48(6), 2096–2109.
- Read, J., Pfahringer, B., & Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE International Conference on Data Mining* (pp. 995–1000). IEEE.
- Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, 85(3), 333.
- Ricci, E. & Perfetti, R. (2008). Large margin methods for structured output prediction. In L. C. Jain, M. Sato-Ilic, M. Virvou, G. A. Tsihrintzis, V. E. Balas, & C. Abeynayake (Eds.), *Computational intelligence paradigms: innovative applications* (pp. 109–132). Springer Berlin Heidelberg.
- Riloff, E., Wiebe, J., & Wilson, T. (2003). Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the 7th Conference on Natural Language Learning* (pp. 25–32). Association for Computational Linguistics Stroudsburg, PA.
- Rosenberg, C., Hebert, M., & Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *Proceedings of the 7th IEEE Workshop on Applications of Computer Vision* (pp. 29–36). IEEE.

- Rousu, J., Saunders, C., Szedmak, S., & Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *The Journal of Machine Learning Research*, 7, 1601–1626.
- Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., . . . Münsterkötter, M., et al. (2004). The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18), 5539–5545.
- Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., & Edwards, D. D. (2003). *Artificial intelligence: a modern approach*. Englewood Cliffs, New Jersey: Prentice-Hall Upper Saddle River.
- 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.
- Santos, A. & Canuto, A. (2014). Applying semi-supervised learning in hierarchical multilabel classification. *Expert Systems with Applications*, 41(14), 6075–6085.
- Schapire, R. E. & Singer, Y. (2000). BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3), 135–168.
- Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., & Džeroski, S. (2010). Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11(2), 1–14.
- Schulte, R., Richards, K., Kurz, D., McDonald, E., & Holden, N. (2006). Agriculture, meteorology and water quality in ireland: a regional evaluation of pressures and pathways of nutrient loss to water. *Biology and Environment: Proceedings of the Royal Irish Academy*, 106b, 117–133.
- Sellamanickam, S., Tiwari, C., & Selvaraj, S. K. (2012). Regularized structured output learning with partial labels. In *Proceedings of the SIAM International Conference on Data Mining* (pp. 1059–1070). SIAM.
- Silla, C. N. & Freitas, A. A. (2009). A global-model naive bayes approach to the hierarchical prediction of protein functions. In *Proceeding of the 9th IEEE International Conference on Data Mining* (pp. 992–997). IEEE.
- Silla, C. & Freitas, A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2), 31–72.
- Simonoff, J. S. (2013). *Analyzing categorical data*. Springer Science & Business Media.
- Šimunović, M., Perković, I., Zorc, B., Ester, K., Kralj, M., Hadjipavlou-Litina, D., & Pontiki, E. (2009). Urea and carbamate derivatives of primaquine: synthesis, cytostatic and antioxidant activities. *Bioorganic & Medicinal Chemistry*, 17(15), 5605–5613.
- Skrjanc, M., Grobelnik, M., & Zupanec, D. (2001). Insights offered by data-mining when analyzing media space data. *Informatica (Slovenia)*, 25(3), 357–363.
- Slavkov, I., Gjorgjioski, V., Struyf, J., & Džeroski, S. (2010). Finding explained groups of time-course gene expression profiles with predictive clustering trees. *Molecular BioSystems*, 6(4), 729–740.
- Spyromitros, E., Tsoumakas, G., & Vlahavas, I. (2008). An empirical study of lazy multilabel classification algorithms. In *Proceedings of the 5th Hellenic Conference on Artificial Intelligence* (pp. 401–406). Springer Berlin Heidelberg.
- Spyromitros-Xioufis, E., Groves, W., Tsoumakas, G., & Vlahavas, I. (2014). Multi-label classification methods for multi-target regression. *arXiv preprint 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.

- Steinbeck, C., Han, Y., Kuhn, S., Horlacher, O., Luttmann, E., & Willighagen, E. (2003). The chemistry development kit (CDK): an open-source java library for chemo- and bioinformatics. *Journal of Chemical Information and Computer Science*, 43(2), 493–500.
- Stojanova, D., Ceci, M., Malerba, D., & Dzeroski, S. (2013). Using PPI network autocorrelation in hierarchical multi-label classification trees for gene function prediction. *BMC Bioinformatics*, 14, 285.
- Struyf, J. & Dzeroski, S. [Sašo]. (2006). Constraint based induction of multi-objective regression trees. In F. Bonchi & J.-F. Boulicaut (Eds.), *Proceedings of the 4th International Conference on Knowledge Discovery in Inductive Databases* (Vol. 3933, pp. 222–233). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Su, H. & Rousu, J. (2015). Multilabel classification through random graph ensembles. *Machine Learning*, 99(2), 231–256.
- Subramanya, A., Petrov, S., & Pereira, F. (2010). Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 167–176). Association for Computational Linguistics Stroudsburg, PA.
- Sun, Z., Zhao, Y., Cao, D., & Hao, H. (2016). Hierarchical multilabel classification with optimal path prediction. *Neural Processing Letters*, 1–15.
- Suzuki, J., Fujino, A., & Isozaki, H. (2007). Semi-supervised structured output learning based on a hybrid generative and discriminative approach. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 791–800). Association for Computational Linguistics Stroudsburg, PA.
- Švec, J. (2014). Semi-supervised learning algorithm for binary relevance multi-label classification. In B. Benatallah, A. Bestavros, B. Catania, A. Haller, Y. Manolopoulos, A. Vakali, & Y. Zhang (Eds.), *Proceedings of the Workshop on Web Information Systems Engineering* (Vol. 9051, pp. 1–13). Lecture Notes in Computer Science. Springer International Publishing.
- Tanha, J., van Someren, M., & Afsarmanesh, H. (2015). Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, 1–16.
- 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(S1), S7+.
- Torgo, L. (2016). Luís torgo - regression data sets. Retrieved from <http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html>
- Triguero, I., García, S., & Herrera, F. (2015). Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2), 245–284.
- Trohidis, K., Tsoumakas, G., Kalliris, G., & Vlahavas, I. P. (2008). Multi-label classification of music into emotions. In *Proceedings of the International Conference on Music Information Retrieval* (Vol. 8, pp. 325–330).
- Tsanas, A. & Xifara, A. (2012). Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49, 560–567.
- Tsoumakas, G., Katakis, I., & Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of the ECML/PKDD Workshop on Mining Multidimensional Data* (pp. 30–44).

- Tsoumakas, G., Katakis, I., & Vlahavas, I. (2009). Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook* (pp. 667–685). Springer US.
- Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., & Vlahavas, I. (2014). Multi-target regression via random linear target combinations. In T. Calders, F. Esposito, E. Hüllermeier, & R. Meo (Eds.), *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (Vol. 8726, pp. 225–240). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- Tsoumakas, G. & Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multi-label classification. In *Proceedings of the European Conference on Machine Learning* (Vol. 4701, pp. 406–417). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- 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.
- Vanschoren, J., Van Rijn, J. N., Bischl, B., & Torgo, L. (2014). Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2), 49–60.
- Vazquez, E. & Walter, E. (2003). Multi-output support vector regression. In *Proceedings of the 13th IFAC Symposium on System Identification* (pp. 1820–1825).
- Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2), 185–214.
- Vergara, A., Vembu, S., Ayhan, T., Ryan, M. A., Homer, M. L., & Huerta, R. (2012). Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166, 320–329.
- Wang, B. & Tsotsos, J. (2016). Dynamic label propagation for semi-supervised multi-class multi-label classification. *Pattern Recognition*, 52, 75–84.
- Wang, J., Zhao, Y., Wu, X., & Hua, X.-S. (2011). A transductive multi-label learning approach for video concept detection. *Pattern Recognition*, 44(10–11), 2274–2286.
- Wang, S., Wang, J., Wang, Z., & Ji, Q. (2014). Enhancing multi-label classification by modeling dependencies among labels. *Pattern Recognition*, 47(10), 3405–3413.
- Wang, Y., Haffari, G., Wang, S., & Mori, G. (2009). A rate distortion approach for semi-supervised conditional random fields. *Advances in Neural Information Processing Systems*, 22, 2008–2016.
- 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 V. S. Tseng, T. B. Ho, Z.-H. Zhou, A. L. P. Chen, & H.-Y. Kao (Eds.), *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining* (Vol. 8444, pp. 162–173). Lecture Notes in Computer Science. Springer International Publishing.
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1, 80–83.
- Witten, I. H. & Frank, E. (2005). *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*, (Vol. 29, pp. 197–212). 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.
- Wu, Q., Ye, Y., Zhang, H., Chow, T. W. S., & Ho, S. S. (2015). ML-TREE: a tree-structure-based approach to multilabel learning. *IEEE Transactions on Neural Networks and Learning Systems*, 26(3), 430–443.

- Xu, J. (2014). Multi-label core vector machine with a zero label. *Pattern Recognition*, 47(7), 2542–2557.
- Xu, M., Sun, F., & Jiang, X. (2014). Multi-label learning with co-training based on semi-supervised regression. In *Proceedings of the IEEE International Conference on Security, Pattern Analysis, and Cybernetics* (pp. 175–180). IEEE.
- 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.
- Yang, M.-C. & Wang, Y.-C. F. (2013). A self-learning approach to single image super-resolution. *IEEE Transactions on Multimedia*, 15(3), 498–508.
- Yang, Q. & Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(4), 597–604.
- Yang, Y. [Yanyan], Chen, D., & Dong, Z. (2015). Novel multi-output support vector regression model via double regularization. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics* (pp. 2697–2701). IEEE.
- Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics* (pp. 189–196). Association for Computational Linguistics Stroudsburg, PA.
- Yeh, C.-K., Wu, W.-C., Ko, W.-J., & Wang, Y.-C. F. (2017). Learning deep latent spaces for multi-label classification. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. AAAI.
- Ženko, B. (2007). *Learning predictive clustering rules* (Doctoral dissertation, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia).
- Ženko, B. & Džeroski, S. [Sašo]. (2008). Learning classification rules for multiple target attributes. In T. Washio, E. Suzuki, K. M. Ting, & A. Inokuchi (Eds.), *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining* (Vol. 5012, pp. 454–465). Lecture Notes in Computer Science. Springer Berlin Heidelberg.
- 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.
- Zhang, C. & Wang, F. (2009). Graph-based semi-supervised learning. *Artificial Life and Robotics*, 14(4), 445–448.
- Zhang, M.-L. (2009). ML-rbf: RBF neural networks for multi-label learning. *Neural Processing Letters*, 29(2), 61–74.
- 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.
- Zhang, M.-L. & Zhou, Z.-H. (2007). ML-kNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7), 2038–2048.
- 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.
- Zhang, W., Liu, X., Ding, Y., & Shi, D. (2012). Multi-output LS-SVR machine in extended feature space. In *Proceedings of the IEEE International Conference on Computational Intelligence for Measurement Systems and Applications* (pp. 130–134). IEEE.
- Zhang, Y. & Yeung, D.-Y. (2009). Semi-supervised multi-task regression. In W. Buntine, M. Grobelnik, D. Mladenić, & J. Shawe-Taylor (Eds.), *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (Vol. 5782, pp. 617–631). Lecture Notes in Computer Science. Springer Berlin Heidelberg.

- Zhao, C. & Zhai, S. (2015). Minimum variance semi-supervised boosting for multi-label classification. In *Proceedings of the IEEE Global Conference on Signal and Information Processing* (pp. 1342–1346). IEEE.
- Zhou, T., Tao, D., & Wu, X. (2012). Compressed labeling on distilled labelsets for multi-label learning. *Machine Learning*, 88(1-2), 69–126.
- Zhou, Z.-H. & Li, M. (2005). Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11), 1529–1541.
- Zhou, Z.-H. & Li, M. (2007). Semi-supervised regression with co-training style algorithms. *IEEE Transaction on Knowledge and Data Engineering*, 19(11), 1479–1493.
- Zhu, X. (2005). *Semi-supervised learning with graphs* (Doctoral dissertation, Carnegie Mellon University, Pittsburgh, Pennsylvania).
- Zhu, X. (2008). *Semi-supervised learning literature survey*. Computer Sciences, University of Wisconsin-Madison.
- Zhu, X. & Goldberg, A. B. [Andrew B.]. (2009). *Introduction to semi-supervised learning*. San Rafael, CA: Morgan and Claypool.
- Zien, A., Brefeld, U., & Scheffer, T. (2007). Transductive support vector machines for structured variables. In *Proceedings of the 24th International Conference on Machine Learning* (pp. 1183–1190).

Bibliography

Publications Related to the Thesis

Journal Articles

- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2017a). Self-training for multi-target regression with tree ensembles. *Knowledge-Based Systems*, 123, 41–60.
- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2017b). Semi-supervised classification trees. *Journal of Intelligent Information Systems*. In press.
- Levatić, J., Kocev, D., Ceci, M., & Džeroski, S. (2017). *Semi-supervised trees for multi-target regression*. Under review.

Book chapters

- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2015a). Semi-supervised learning for multi-target regression. In A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, & Z. W. Ras (Eds.), *Third International ECML-PKDD Workshop on New Frontiers in Mining Complex Patterns, Revised selected papers* (Vol. 8983, pp. 3–18). Lecture Notes in Computer Science. Springer International Publishing.

Conference and workshop papers

- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2014). Semi-supervised learning for multi-target regression. In *Proceedings of the 3rd International ECML-PKDD Workshop on New Frontiers in Mining Complex Patterns* (pp. 110–123).
- Levatić, J., Ceci, M., Kocev, D., & Džeroski, S. (2015b). Semi-supervised learning for multi-target regression. In *Proceedings of the 23rd Italian Symposium on Advanced Database Systems* (pp. 249–255).
- Levatić, J., Supek, F., & Džeroski, S. (2015). Improving QSAR models by exploiting unlabeled data from public databases of bioactive drug-like molecules. In *Proceedings of the 7th Jožef Stefan International Postgraduate School Students' Conference* (pp. 114–123). Best ICT paper award.

Other Publications

Journal Articles

- Levatić, J., Ćurak, J., Kralj, M., Šmuc, T., Osmak, M., & Supek, F. (2013). Accurate models for p-gp drug recognition induced from a cancer cell line cytotoxicity screen. *Journal of Medicinal Chemistry*, 56(14), 5691–5708.
- Levatić, J., Džeroski, S., Supek, F., & Šmuc, T. (2013). Semi-supervised learning for quantitative structure-activity modeling. *Informatica (Slovenia)*, 37(2), 173–179.

- Levatić, J., Kocev, D., Debeljak, M., & Džeroski, S. (2015). Community structure models are improved by exploiting taxonomic rank with predictive clustering trees. *Ecological Modelling*, 306, 294–304.
- Levatić, J., Kocev, D., & Džeroski, S. [Sašo]. (2014a). The importance of the label hierarchy in hierarchical multi-label classification. *Journal of Intelligent Information Systems*, 45(2), 247–271.

Book chapters

- Levatić, J., Kocev, D., & Džeroski, S. [Sašo]. (2014b). The use of the label hierarchy in hierarchical multi-label classification improves performance. In A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, & Z. W. Ras (Eds.), *Second International ECML-PKDD Workshop on New Frontiers in Mining Complex Patterns, Revised selected papers* (Vol. 8399, pp. 162–177). Lecture Notes in Computer Science. Springer International Publishing.

Conference and workshop papers

- Levatić, J., Džeroski, S., Supek, F., & Šmuc, T. (2012). Semi-supervised learning in diverse quantitative structure-activity modeling problems. In *Proceedings of the 15th International Multiconference Information Society* (pp. 60–63).
- Levatić, J., Kocev, D., & Džeroski, S. [Sašo]. (2013). The use of the label hierarchy in HMC improves performance: A case study in predicting community structure in ecology. In *Proceedings of the 3rd ECML/PKDD Workshop on New Frontiers in Mining Complex Patterns* (pp. 189–201).
- Levatić, J., Ramšak, Ž., Tjaša, S., Kocev, D., Gruden, K., & Džeroski, S. (2013). Gene function prediction for *Solanum tuberosum* from time-series gene expression data. In *Proceedings of the 5th Jožef Stefan International Postgraduate School Students' Conference* (pp. 158–167).

Biography

The author of this thesis was born on 17th of July, 1986 in Varaždin, Croatia, where he finished his primary and secondary education. In 2005, he started his studies at the Department of Mathematics, Faculty of Science, University of Zagreb in Zagreb, Croatia. He obtained a Bachelor's degree in Mathematics in 2008. In the same year, he enrolled in a graduate programme in Computer Science and Mathematics. In 2011, he successfully completed his master studies, with the thesis entitled: "Techniques for semi-supervised learning" under the supervision of Prof. Dr. Tomislav Šmuc.

In the period from August 2011 to November 2012, he worked as a research associate at BioZyne (Zagreb, Croatia). He worked on discovering potential novel anti-cancer agents with the help of machine learning. Together with colleagues from BioZyne, he published a paper about the results of this research in a prestigious scientific journal.

In the fall of 2012, he enrolled in the "Information and Communication Technologies" PhD program at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, under the supervision of Prof. dr. Sašo Džeroski and co-supervision of Dr. Dragi Kocev. Levatić holds a young researcher scholarship for doctoral studies awarded by the Slovenian Research Agency. He works at the Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia, where he is strongly involved in the EU funded FP7 project MAESTRA (Learning from Massive, Incompletely annotated, and Structured Data).

His research interests are in the area of machine learning, more specifically in the area of semi-supervised learning and structured output predication. In particular, he works on the development of a methodology which combines these two fields of machine learning. He works on the development of machine learning methods, as well as their application in various scientific disciplines, such as cheminformatics and ecology. He has published several journal papers and conference papers and has presented his work at several international conferences and workshops, both in the area of machine learning and the areas of application.

