DOI: https://doi.org/10.1016/j.ins.2018.03.033

Semi-supervised trees for multi-target regression

Jurica Levatić[a,b], Dragi Kocev[a,b,c], Michelangelo Ceci[c,d], Sašo Džeroski[a,b]

[a]*Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia*
[b]*Jožef Stefan International Postgraduate School, Ljubljana, Slovenia*
[c]*Department of Computer Science, University of Bari Aldo Moro, Bari, Italy*
[d]*CINI, Consorzio Interuniversitario Nazionale per l'Informatica, Italy*

Abstract

The predictive performance of traditional supervised methods heavily depends on the amount of labeled data. However, obtaining labels is a difficult process in many real life tasks, and only a small amount of labeled data is typically available for model learning. As an answer to this problem, the concept of semi-supervised learning has emerged. Semi-supervised methods use unlabeled data in addition to labeled data to improve upon the performance of supervised methods.

It is even more difficult to get labeled data for data mining problems with structured outputs, since several labels need to be determined for each example. Multi-target regression (MTR) is one type of a structured output prediction problem, where we need to simultaneously predict multiple continuous variables. Despite the apparent need for semi-supervised methods able to deal with MTR, only few such methods are available and even those are difficult to use in practice and/or their advantages over supervised methods for MTR are not clear.

This paper presents an extension of predictive clustering trees for MTR and ensembles thereof towards semi-supervised learning. The proposed method preserves the appealing characteristic of decision trees, while enabling the use of unlabeled examples. We perform an extensive empirical evaluation in both an inductive and a transductive semi-supervised setting. The results show that the proposed method improves the performance of supervised predictive clustering trees and enhances their interpretability (due to reduced tree size), whereas in the ensemble learning scenario, it outperforms

*Corresponding author
Email address: `jurica.levatic@ijs.si` (Jurica Levatić)

its supervised counterpart in the transductive setting. The proposed methods have a mechanism for controlling the influence of unlabeled examples, which makes them highly useful in practice: This mechanism can protect them against a degradation of performance of their supervised counterparts – an inherent risk of semi-supervised learning. The proposed methods also outperform two existing semi-supervised methods for MTR.

Keywords: Semi-supervised learning, Multi-target regression, Structured outputs, Predictive Clustering Trees, Random forests

1. Introduction

In supervised learning, the goal is to learn, from a set of examples with known labels, a function that outputs a prediction for a previously unseen example. Supervised algorithms often need a large amount of labeled data to obtain satisfactory predictive performance. However, in many real life problems, only a few labeled examples are available to learn from, due to expensive and/or time-consuming annotation procedures. The concept of semi-supervised learning (SSL) emerged in the 1970's as an answer to this problem [12]. Semi-supervised algorithms use unlabeled examples (often freely available in vast amounts), in addition to labeled ones, in order to achieve better performance than algorithms using only labeled examples.

Independently from the development of SSL, the need to mine knowledge from structured data was recognized as a machine learning problem of great importance [19, 29]. The output (i.e., the target) space is *structured* in many applications of predictive modelling, meaning that the values to be predicted are structured (rather than scalar).Multi-target regression (MTR) is a type of structured output prediction (SOP) task where multiple continuous variables are simultaneously predicted: The structured value to be predicted in this case is a tuple of real numbers. MTR has applications in many real life problems, with prominent examples in ecology, such as predicting the abundance of different species living in the same habitat [17], and predicting different properties of forests from remote sensing data [26, 44].

Since the establishment of SSL as a research topic, the scientific community has devoted a lot of effort to SSL for the classical (i.e., unstructured) data mining tasks: classification and regression [53]. Unfortunately, this has not been the case for SOP tasks, although the need for SSL is even stronger there: The labeling process is even more expensive and laborious in SOP, since several simple/primitive labels (or one structured/complicated label) need to be provided for each example. Furthermore, the existing SSL methods for SOP largely deal with discrete types of outputs, such as a recent method proposed by Du [20]. There are only a few examples of SSL methods dealing with the task of MTR [35, 23, 31, 10]. In addition these have several serious limitations, including (a) applicability only to a certain domain, (b) unclear advantage as compared to supervised learning due to insufficient evaluation, (c) high computational complexity, and (d) high risk of performance degradation as compared to the underlying supervised method. Finally, none of the available semi-supervised methods for MTR produces interpretable models, which often is an important asset in the context of knowledge discovery for predictive modelling.

As an attempt to overcome these limitations, we propose to perform SSL for MTR with a popular data mining method - decision trees. More specifically, we extend the approach of predictive clustering trees (PCTs) for MTR [4, 45], as well as ensembles thereof [28], towards the SSL framework. We also thoroughly evaluate the proposed methods on a wide variety of data-sets from different domains and extensively discuss their characteristics from several viewpoints, including predictive performance, computational complexity, model size, and sensitivity to parameters. Our empirical evaluation shows that the proposed extension of PCTs towards SSL improves the predictive performance of PCTs and enhances their interpretability. Furthermore, ensembles (i.e., random forests) of semi-supervised PCTs for MTR can improve the predictive performance of ensembles of supervised PCTs.

The remainder of this paper is organized as follows. The next section clarifies our motivation and outlines the main contributions of this study. Section 3 presents the background of the work, which includes a discussion on related work and a brief description of the predictive clustering framework. Section 4 describes the proposed method, while Section 5 specifies the experimental design. The results of the empirical investigations are presented and discussed in Section 6. Finally, Section 7 concludes the paper.

2. Motivation and Contributions

As already noted in the Introduction, the selection of semi-supervised methods for the task of MTR is very limited, even though the need for semi-supervised methods is arguably even greater in the context of structured output (due to the increased complexity of labeling). Semi-supervised learning for single-target regression, on the other hand, has seen much more development [53]. In principle, it is possible to decompose a MTR problem into several (local) single-target ones, and then apply some of the available semi-supervised methods for single-target regression (i.e., local methods). However, the methods that learn to predict all of the target variables simultaneously (i.e., global methods) are typically more computationally efficient, produce simpler models, and overfit less than the local methods [26, 28, 33]. In addition, global models can yield better predictive performance than local models [26, 25, 47]. Although recent studies showe that more sophisticated local models (where outputs of local models are taken as inputs for other local models) for MTR can perform better than state-of-the-art global approaches for MTR [42].

Given the above-mentioned advantages of global methods, in this work, we aim to develop global semi-supervised methods for MTR. To this end, we consider predictive clustering trees (PCTs) for MTR [4, 45]. PCTs are generalization of standard decision trees towards predicting structured outputs, such as tuples of continuous/discrete variables, hierarchies of classes, and time series. We consider PCTs as a very natural candidate for extension towards SSL since they are situated at the intersection of predictive modelling (supervised learning) and clustering (unsupervised learning).

More specifically, in contrast to classical decision/regression trees, PCTs use a flexible definition of descriptive, target, and clustering attributes. Descriptive attributes are used to divide examples into groups (i.e., define splits), clustering attributes to evaluate the quality of candidate splits, and target attributes to calculate the predicted values 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 attribute used for the purpose of split evaluation. In other words, the similarity of examples is explicitly enforced only on the target attribute, and splits are selected which divide examples into groups of examples with similar values for the target attribute. In PCTs, the clustering attributes can overlap with the target and/or the descriptive attributes.

We capitalize on these properties of PCTs to extend them towards SSL: We propose to perform SSL with PCTs that would group examples similar in both the descriptive and the target space during PCT construction. This can be 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 would be able to produce clusters compact in both the descriptive and the target space. This could be viewed as enforcing the popular semi-supervised smoothness assumption which 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* [12]. In fact, in order to benefit from unlabeled data, semi-supervised methods have to make assumptions about the distribution of the unlabeled data with respect to (several) target variables. The above-mentioned assumption presupposes a smooth prediction function in the highly populated regions.

We illustrate the benefits of the above-described extension of PCTs towards SSL on the toy example of modelling a step function $f(x) = \{1, x \leq 1.5; 2, x > 1.5$ given in Fig. 1. We generated two clusters of 50 data points each, where the descriptive space of examples is sampled for two normal distributions ($N(1, 0.25)$ for the first cluster and $N(2, 0.25)$ for the second), while the target values of examples are generated according to the above step function with some random noise added. Data obtained in such a way complies with the semi-supervised smoothness assumption. Four points were selected at random as labeled examples, while the remaining data points were used as unlabeled examples and test examples. The supervised PCT positions the split in the middle of a gap between the labeled examples from different clusters. However, this split cuts through the dense cluster of (unlabeled) data and consequentially the supervised prediction function (i.e., average of the target values of examples in the leafs) has a big change in the highly populated region. The semi-supervised PCT, on the other hand, positions the split so that is separates well both the labeled and unlabeled data, leading to a prediction function that matches the step function more closely (i.e., a split is close to 1.5) and results in the lower error as compared to the supervised PCT. Note that the semi-supervised PCT "sees" only the descriptive space values of unlabeled examples.

We argue that, by extending PCTs towards SSL, it is possible to obtain predictive models which preserve the appealing characteristics of decision trees and are more accurate than the models learned with labeled data alone. This would be achieved by exploiting both unlabeled and labeled data on one hand, and the properties of global predictive models for SOP on the other. The contributions of this paper are summarized as follows:

- We develop a method for semi-supervised learning of interpretable and global models for MTR;

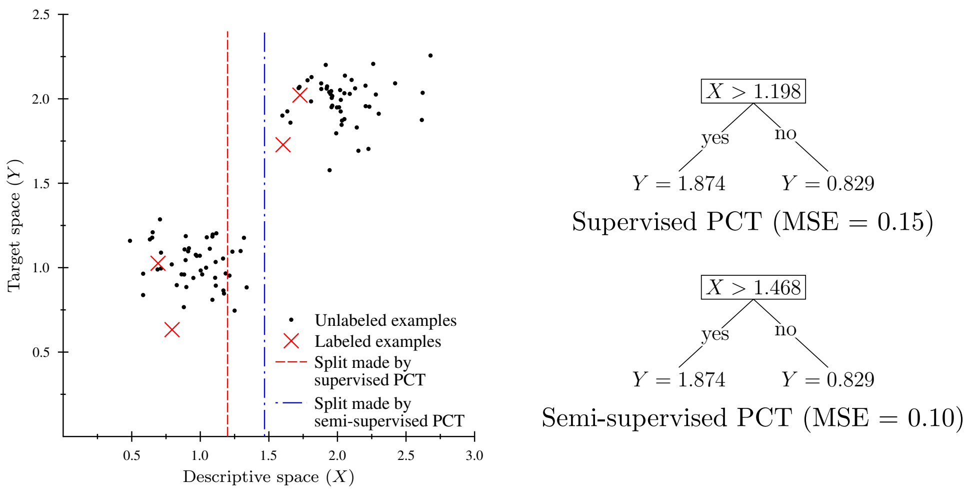


Figure 1: A semi-supervised and a supervised PCT built by using toy data. MSE denotes the means squared error.

- We explore the performance of the proposed semi-supervised trees for MTR in the ensemble setting;
- We perform an extensive empirical evaluation of the proposed method on 15 MTR datasets from various domains, investigating several properties of the methods: predictive performance, influence of the amount of labeled data, model size, and sensitivity to parameters;
- We explore the use of feature weighting to reduce the influence of irrelevant attributes;
- We extensively discuss the characteristics of the proposed approach from the viewpoint of practical utility and the possible uses of the method in fields of data mining other than SSL.

3. Background

This work is motivated by research at the intersection between the fields of semi-supervised learning and multi-target regression. In the following subection, we discuss related work from both research fields, with the focus on limitations of the currently available semi-supervised methods for MTR, and on the novelties and contributions of our work with respect to the most closely related work. We also provide a description of predictive clustering trees for MTR and ensembles thereof, which are the starting point for the semi-supervised methods we propose.

3.1. Related work

Zhang et al. [51] and Cardona et al. [11] proposed semi-supervised methods based on Gaussian processes for a task related to (but different than) MTR: *multi-task* regression. Multi-target regression and multi-task regression[1] are similar, but some notable differences exist between the two. In multi-task learning, we have several tasks of single-target prediction with different training sets (with possibly different descriptive attributes) and the emphasis is on learning transfer between the tasks. Navaratnam et al. [35] have proposed a SSL method for MTR, which is also based on Gaussian processes. This method is, however, specialized for application in a specific domain, namely computer vision.

The methods proposed by Gönen and Kaski [23] and Brouard et al. [10] are rare examples of semi-supervised methods which can handle both discrete and continuous types of structured outputs. Among other types of structured output prediction tasks, their methods can also handle MTR. Both methods are based on kernels. Gönen and Kaski [23] proposed the Kernelized Bayesian Matrix Factorization (KBMF) method, where multiple sources of information can be exploited as different kernels in multiple kernel learning. Brouard et al. [10] proposed Input Output Kernel Regression (IOKR) which can exploit the structure both in the input and the output space by defining different kernels for the input and for the output space. A similar principle, i.e., defining separate kernels for input and the output spaces, can be obtained with the KBMF method,. The competitive advantages of the KBMF and IOKR methods in the context of semi-supervised learning for MTR are, however, not entirely clear. Namely, the KBMF method has not been tested in its semi-supervised version on the task of MTR, while the IOKR method was tested on only one MTR dataset. In contrast, we extensively evaluate the proposed semi-supervised methods for MTR on a wide variety of benchmark

[1]Note that multi-task regression is sometimes referred to as a multi-output regression which is also a synonym for multi-target regression.

datasets showing their utility over a variety of domains. Furthermore, the KBMF and IOKR methods are arguably difficult to use for non-experts, since the user needs to define kernels that are appropriate for the task at hand and optimize several parameters, while the semi-supervised methods we propose here have only one parameter to tune (which is tuned automatically).

In a previous study, we have developed a SSL method based on self-training of random forests of PCTs for MTR [31], where the most reliable predictions on unlabeled data are iteratively used to re-train the model. The previous method and the one we propose in this paper are used for the same task, i.e., SSL for MTR. However, the two methods are different in the way they exploit the unlabeled examples. More specifically, they are based on different assumptions of semi-supervised learning: Self-training assumes that the most confident predictions are correct, while the method proposed in this work assumes smoothness of the prediction function in highly populated regions. The next difference comes from the perspective of utility and efficiency. First, self-training requires repeated re-training of the base model; therefore, its computational demand is much higher than the demand of the method proposed here. Next, the self-training approach is, in general, prone to error propagation, i.e., it can degrade the performance of its base model if erroneous predictions enter the training set. The method proposed in this work, on the other hand, has a built-in safety mechanism due to which it has very low risk of performance degradation (as compared to its supervised counterpart).

Finally, none of the above-mentioned semi-supervised methods for MTR produce interpretable models. Interpretable models are, however, of great importance in many applications of machine learning where knowledge discovery is of interest. 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 how it produces the predictions. The semi-supervised algorithm proposed in this work preserves the interpretability of supervised PCTs, while enabling the exploitation of unlabled data.

The semi-supervised PCTs proposed here are related to the work of Blockeel et al. [4], who proposed clustering trees which can consider both descriptive and target attributes in the evaluation of splits. Blockeel et al. [4] suggested that, when the class information is missing, considering both descriptive and target attributes in the evaluation of splits can improve the predictive performance of clustering trees. The semi-supervised decision trees that we propose in this work are similar in nature to the ones proposed by Blockeel et al. [4], but there are several key differences. First of all, in experiments with missing class information, Blockeel et al. [4] did not consider tasks of predicting structured outputs. Second, we introduce an additional parameter, by which we control the amount of supervision in the decision trees (i.e., the influence of the target relative to the descriptive attributes). 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. A similar concept was considered by Ženko [48] who proposed predictive clustering rules (PCRs). In the context of learning PCRs for multi-target classification, the use of a heuristic was proposed that guides the rule learning process and takes into account (with a trade-off parameter) both the descriptive and the target attributes. However, Ženko [48] considered learning of such PCRs only in a supervised learning context. We recently explored decision trees that use both the descriptive and the target attributes for evaluation of splits in the context of semi-supervised learning for binary and multi-class classification [32], however, it is not clear whether the findings from the simpler tasks will transfer also to the MTR task.

Independently of SSL, MTR receives increasing attention by the research community [6]. Several machine learning methods, popular for regression, have been implemented also for the task of MTR, such as decision trees [2, 45], support vector machines [50], k-nearest neighbors [38] and ensemble-based methods [1, 28, 47, 42]. Here, ensemble-based methods present the state-of-the-art regarding predictive performance. However, out of the available methods for MTR, only decision trees produce interpretable models, which is an important property in many domains, such as biology, medicine and chemoinformatics. Our study builds upon the work of Struyf and Džeroski [45] and Kocev et al. [28], extending the decision tree algorithm of Struyf and Džeroski and random forests algorithm of Kocev et al. towards semi-supervised learning.

3.2. Predictive clustering trees for MTR

The PCT framework views a decision tree as a hierarchy of clusters, where the top-node corresponds to one cluster containing all the data. This cluster is recursively partitioned into smaller clusters while moving down the tree. The PCT framework is implemented in the CLUS system [28, 45], available at `http://sourceforge.net/projects/clus`.

PCTs are induced with a standard *top-down induction of decision trees* (TDIDT) algorithm (see Table 1), which takes as input a set of examples E and outputs a tree. The heuristic that is used for selecting the tests to put in internal tree nodes is the reduction of variance caused by partitioning the examples according to the tests. By maximizing the variance reduction, the cluster homogeneity is maximized and the predictive performance is improved.

The main difference between the algorithm for learning PCTs and a standard decision tree learner is that the former considers the variance function and the prototype function (that computes a label for each leaf) as *parameters* that can be instantiated for a given learning task. So far, PCTs have been instantiated for the following tasks [28]: multi-target prediction (which includes MTR), hierarchical multi-label classification, and prediction of time-series.

In this article, we focus on the task of MTR, which we formalize as follows. Given:

- A description (or input) space X spanned by D descriptive variables, i.e., $X = (X_1, \ldots, X_D)$.
- A target (or output) space Y spanned by T continuous target variables, i.e., $Y = (Y_1, \ldots, Y_T)$.
- A set of examples E, where each example is a pair consisting of one element from the descriptive space and one element from the target space (the example's label), i.e., $E = \{(x_i, y_i) : x_i \in X,\ y_i \in Y,\ 1 \leq i \leq N\}$, and N is the number of examples.
- A quality criterion q.

Find: a function $f : X \to Y$ such that f maximizes q. In this work, the function f is represented with decision trees, i.e., PCTs.

The quality criterion q is based on the reduction of overall variances which is calculated as the average of the variances of the target variables (see Table 1, procedure BestTest, line 4). For each set of examples E, the variance is computed as follows:

$$Var_f(E) = \frac{1}{T} \cdot \sum_{i=1}^{T} Var_i(E), \tag{1}$$

where $Var_i(E)$ 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:

$$Var_i(E) = \frac{\sum_{j=1}^{N}(y_{i,j})^2 - \frac{1}{N} \cdot \left(\sum_{j=1}^{N} y_{i,j}\right)^2}{N}, \tag{2}$$

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 (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 Table 1, procedure PCT, line 7). In PCTs for MTR the prototype function calculates the mean vector of all target variables Y for the training examples that belong to the leaf.

Table 1: The top-down induction algorithm for PCTs.

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

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

3.3. Ensembles of predictive clustering trees for MTR

Kocev et al. [28] implemented ensembles of PCTs for structured outputs in the CLUS system. The ensembles of PCTs are constructed by using the bagging [7] and random forests [9] methods, which are often used in the context of decision trees. Bagging 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 [7] 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.

A random forest 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 1) is changed to a 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 MTR, a corresponding type of PCTs is used as a base model, i.e., PCTs for MTR. The prediction of an ensemble for a new instance is obtained by combining the predictions of all the base predictive models from the ensemble. Namely, for the MTR task, predictions of the base models are combined by taking their average.

4. Semi-supervised learning of PCTs for MTR

In this section, we present the proposed algorithm for semi-supervised learning of predictive clustering trees (SSL-PCTs) for MTR. Before describing the algorithm in detail, we first need to define the task of *semi-supervised multi-target regression*. We formalize it as follows.

Given:

- A description (or input) space X spanned by D descriptive variables, i.e., $X = (X_1, \dots, X_D)$.
- A target (or output) space Y spanned by T continuous target variables, i.e., $Y = (Y_1, \dots, Y_T)$.

- A set of labeled examples E_l, where each example is a pair consisting of one element from the descriptive space and one element from the target space (the example's label), i.e., $E_l = \{(x_i, y_i) : x_i \in X,\ y_i \in Y,\ 1 \leq i \leq N_l\}$, and N_l is the number of labeled examples.

- A set of unlabeled examples E_u, which consists only of elements from the descriptive space, i.e., $E_u = \{x_i : x_i \in X,\ 1 \leq i \leq N_u\}$, and N_u is the number of unlabeled examples.

- A quality criterion q, e.g., which rewards models with low predictive error.

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

Note that, even though the supervised and semi-supervised methods have the same goal (i.e., they optimize the same quality criterion q), their success is characterized differently. Namely, the success of supervised methods is measured by their predictive performance in absolute terms, while the success of semi-supervised methods is judged by their predictive performance relative to the corresponding supervised methods. A successful semi-supervised method should be able to outperform the corresponding supervised method while using the same set of labeled examples E_l and additional set of unlabeled examples E_u.

In semi-supervised learning, there are two different settings: inductive and transductive semi-supervised learning. Inductive learning is concerned with predicting the labels of examples unseen during learning, while transductive learning is concerned with predicting only the labels of unlabeled examples in the training set (i.e., labels of examples in E_u). More specifically, the goal of inductive SSL is to obtain a predictive model that can be then applied to other examples, while the goal of transductive SSL is to obtain the labels of the unlabeled examples used during learning. All of the inductive SSL methods can be evaluated in both inductive and transductive setting. The semi-supervised methods we propose here can work both in the inductive and transductive settings.

Our extension of the supervised PCTs towards SSL for MTR goes along two dimensions. The first extension as compared to the classical algorithm for the induction of PCTs concerns the input, which, in case of semi-supervised PCTs, consists of both the 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 the descriptive attributes in the identification of the best split. This is achieved by adapting the variance function used for learning of supervised PCTs, which takes into account only the target attributes (Eq. 1). The variance function used to learn semi-supervised PCTs, on the other hand, is defined as a weighted sum of the variance functions over the target space (Var_f^Y) and over the descriptive space (Var_f^X):

$$Var_f(E) = w \cdot Var_f^Y(E) + (1 - w) \cdot Var_f^X(E), \tag{3}$$

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 employed during 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 6.2.

The variance of a set of examples E over the *target* space ($Var_f^Y(E)$) is calculated similarly as in the supervised PCTs (Eq. 1), i.e., as the average of the variances of the target variable. However, we re-define the variance of individual target attributes ($Var_i(E)$) in order to handle missing values (i.e., unlabeled examples):

$$Var_i(E) = \frac{\sum_{j=1}^{K_i} (y_{i,j})^2 - \frac{1}{K_i} \cdot \left(\sum_{j=1}^{K_i} y_{i,j} \right)^2}{K_i}, \tag{4}$$

where N is the number of examples, and K_i is the number of examples with known (non-missing) values of the i^{th} attribute. Note that, the number of examples with non-missing values is usually the same across all target attributes (i.e., $K_i = |E_l|, i \in 1, \ldots, T$). However, these numbers can differ if partially labeled examples are present in the dataset, i.e., examples which have known values for some of the target variables, and unknown for the others. The above definition of variance (Eq. 4) enables exploitation of such examples, i.e., semi-supervised PCTs can learn from, labeled, unlabeled, as well as partially labeled examples.

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

$$Var_f(E, X) = \frac{1}{D} \cdot \left(\sum_{X_i \text{ is numeric}} Var_i(E) + \sum_{X_j \text{ is nominal}} Gini_j(E) \right), \tag{5}$$

where $Var_i/Gini_j$ is the variance/Gini score of individual numeric/nominal descriptive attributes, respectively. Var_i is calculated analogously to Eq. 4, whereas $Gini_j$ is calculated as follows:

$$Gini_j(E) = 1 - \sum_{k=1}^{C_j} \left(\frac{|\{e : e \in E \text{ such that } e \text{ has class } c_k\}|}{K_j} \right)^2 = 1 - \sum_{k=1}^{C_j} \hat{p_k}, \tag{6}$$

where C_j is the number class values of descriptive attribute X_j, and $\hat{p_k}$ is the empirical probability of class value c_k estimated by considering only examples with a known value for attribute X_j, with K_j being their number.

As in supervised PCTs, the variances or Gini scores of the individual attributes are normalized, so that each attribute contributes equally to the overall variance. The normalization is performed by dividing the variance/Gini score estimates of the individual attributes in Eqs. 4 and 6 on the set of examples in the current tree node with the variance/Gini score of the attribute on the entire available training set.

Note that the proposed algorithm is not limited to using 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 impurity measures. We chose to use the Gini score because it is one of the most popular measures for impurity of nominal variables. The other obvious choice would be Information gain: However, Gini score and information gain perform very similarly and it is mostly not possible to decide which of the two measures to prefer [40]. Similarly, for measuring the impurity of numeric variables, we choose variance, because it is the *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., $K_i = 0$ in Eq. 4). The first extreme opens the question: *How to calculate the prototype function for such a leaf?* In fact, the prototype function of semi-supervised 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., the 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/branches 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[2].

Having 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 a SSL algorithm to construct the members of the ensemble (as described in Section 3.3), instead of using trees learned with a supervised algorithm. The only difference is that the bootstrap samples are

[2]Note that, the second extreme could be handled differently, e.g., by estimating the variance with the variance on the entire training set, or by ignoring such attributes in the calculation of the overall variance (Eq. 3). We have explored these two solutions as well, but they perform very similarly or slightly worse than the solution that uses the variance of the parent node.

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 perform stratified bootstrap sampling, considering the proportions of labeled and unlabled examples in the stratification. This is to avoid having bootstrap samples made out of only unlabeled examples. We denote the semi-supervised random forest build in such a way as SSL-RF, while supervised supervised random forest is denoted as CLUS-RF.

4.1. Feature weighted semi-supervised PCTs for MTR

Unlike some machine learning methods, such as k-nearest neighbors, PCTs (and, more generally, decision/regression 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 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 less informative ones are associated with a lower score. Cunningham and Delany [16] 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 feature ranking for MTR based on a random forest of PCTs [27]. 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: $\sigma_i = \hat{\sigma_i} / \max(\hat{\sigma_1}, \hat{\sigma_2}, \ldots, \hat{\sigma_D})$. The variance of each descriptive attribute X_i is then multiplied by its normalized importance score σ_i in the variance function of SSL-PCTs:

$$Var_f^X(E) = \frac{1}{D} \cdot \left(\sum_{X_i \text{ is numeric}} \sigma_i \cdot Var_i(E) + \sum_{X_j \text{ is nominal}} \sigma_j \cdot Gini_j(E) \right). \quad (7)$$

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

4.2. Computational complexity analysis

To analyze the computational complexity of SSL-PCTs, we first recall the procedures that contribute to the computational complexity of *supervised* PCTs. These are as follows: sorting the values of D descriptive attributes ($\mathcal{O}(DN \log N)$), calculating the best split for T target variables ($\mathcal{O}(TDN)$), and applying the split to the N (labeled) training examples ($O(N)$). Assuming that the depth of the tree is in the order of $\mathcal{O}(\log N)$ [49], the total computational complexity of constructing a single PCT is $\mathcal{O}(DN \log^2 N) + \mathcal{O}(TDN \log N) + \mathcal{O}(N \log N)$.

We then consider what changes from supervised PCTs to SSL-PCTs. This 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 both D descriptive attributes and T target variables when the split is calculated, thus the complexity of this step is $\mathcal{O}((T + D)DN)$. The total computational complexity of learning a single SSL-PCT is thus $\mathcal{O}(DN \log^2 N)) + \mathcal{O}((T + D)DN \log N) + \mathcal{O}(N \log N)$.

The upper bound of the computational complexity of global random forests of SSL-PCTs is $k(\mathcal{O}(D'N' \log^2 N') + \mathcal{O}((T + D)D'N' \log N'))$, where N' is the size of the bootstrap samples and D' is the size of the feature subsets at each tree node and k is the number of base SSL-PCTs. The feature ranking adds the computational cost of random permutations of the values in the out-of-bag sample ($N'' = N - N'$) and sorting the examples through the tree. Both operations are performed for each descriptive attribute and their cost is $O(DN'' + D \log N)$. This computational overhead is small compared to the overall cost of learning the random forests. Furthermore, note that the number of examples here is E_l, since the feature weights are calculated using only the labeled examples.

5. Experimental design

5.1. Data description

We use 15 datatsets with multiple continuous target variables to evaluate the predictive performance of the proposed methods. The datasets come from several different domains, and vary in their size, number of attributes and number of target variables.

Table 2: Characteristics of the datasets. N: number of instances, D/C: the number of descriptive attributes (nominal/continuous), T: number of target variables.

Dataset (Reference)	Domain	N	D	T
Enb [46, 41]	Energy efficiency	768	0/8	2
Eunite [14]	Electricity load forecasting	8064	0/34	5
Forestry Kras [21]	Ecology	60607	0/160	11
Forestry LIDAR IRS [43]	Ecology	2731	0/29	2
Forestry LIDAR LandSat [43]	Ecology	6218	0/150	2
Forestry LIDAR Spot [43]	Ecology	2731	0/49	2
RF1 [41]	Ecology	9125	0/64	8
SCM1D [41]	Economy	9803	0/280	16
SCM20D [41]	Economy	9803	0/61	16
Scpf [42]	Human behavior	1137	19/4	3
Soil quality [17]	Ecology	1944	0/142	3
Solar flare 2 [3]	Astronomy	1066	10/0	3
Vegetation clustering [22]	Ecology	27522	0/66	10
Vegetation condition [26]	Ecology	16967	1/39	7
Water quality [5]	Ecology	1060	0/16	14

The majority of the data are from the area of environmental sciences. To begin with, *Forestry Kras*, *Forestry LIDAR IRS*, *Forestry LIDAR LandSat* and *Forestry LIDAR Spot* datasets concern the task of predicting forest properties from remotely sensed data. Second, the task in the *RF1* dataset is to predict river network flows 48 hours ahead. Next, the *Soil quality* and *Water* *quality* datasets concern habitat modelling of soil and water organisms, respectively. Finally, *Vegetation condition* and *Vegetation clustering* datasets are concerned with predicting several indicators of condition of indigenous and other vegetation in Victoria, Australia.

From the domain of economy, we consider the *SCM1D* and *SCM20D* datasets, 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.

The *Scpf* dataset is concerned with human behavior, namely, with the task for prediction of the number of views, clicks and comments that a specific issue (i.e., an article) concerning public space and service will receive. The data comes from the issues collected in Oakland, Richmond, New Haven and Chicago. The task for the *Eunite* dataset is forecasting of the maximal daily electrical load, while the *Enb* dataset 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.

5.2. Experimental setup and evaluation procedure

We have proposed semi-supervisd PCTs (SSL-PCT) and feature weighted semi-supervised PCTs (SSL-PCT-FR). As a baseline for comparison with these methods we use the standard PCT algorithm for MTR, denoted as Base-PCT. This is the most reasonable baseline, since the goal is to precisely measure the contribution of unlabeled data to the overall performance under the same conditions. By comparing to Base-PCTs, we answer the main question of this study: *Are SSL-PCTs able to improve over standard supervised PCTs?*. In all of the single-tree experiments, both supervised and semi-supervised trees are pruned with the procedure used in M5 regression trees [39]. 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 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 MTR (i.e., CLUS-RF) where Base-PCTs are used as base models. We construct random forests consisting of 100 trees. The trees in the random forest are not pruned and the number of random features considered at each internal node is set to $\lfloor \log_2(D) + 1 \rfloor$, where D is the total number of features [9].

As additional semi-supervised methods for comparison, we use the self-training for MTR [31] and the KBMF method [23]. As a base method for self-training, we use CLUS-RF. To select the most reliable predictions during the self-training iterations, we use the automatic threshold selection procedure *AutomaticOOBInitial* [31]. As a stopping criteria for the self-training, we use the Airbag procedure [30]. This stopping criterion monitors the out-of-bag error [8] of an ensemble to automatically stop the self-training procedure in the case of predictive performance degradation. For the KBMF method, we use the same parameter settings recommended in the original paper [23]. This method uses twin kernels, the first kernel matrix K_x is calculated by using the Gaussian kernel whose width is selected as the square root of the dimensionality of the descriptive space (i.e., $\sqrt{D}$), while the second kernel matrix K_y is calculated as the Pearson correlation coefficient between the target variables. The number of components R is selected from $1, 2, \ldots, 15$ on the basis of the training set performance. The σ_y parameter and the kernel weights $(\alpha_\eta, \beta_\eta, \alpha_\lambda, \beta_\lambda)$ are set to one, the standard deviations (σ_g, σ_h) are set to $(0.1, 0.1)$. For fair comparisons, KBMF is compared with non-ensemble PCTs, while the self-training approach, which is based on CLUS-RF, is compared with other ensemble methods.

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 the ratio of labeled (relative to unlabeled) data ranges as follows: 5%, 10%, 20% and 30% of labeled examples.

As discussed in Section 4, in semi-supervised learning two evaluation scenarios exist: Evaluation on unlabeled examples used during learning (i.e., transductive evaluation), and evaluation on examples unseen during learning (i.e., inductive evaluation). We evaluate the methods considering both scenarions. To this end, we consider a procedure similar to the 10 fold cross validation. First, we randomly divide the dataset into 10 folds, where 9 are used for training and 1 for testing. Next, from the training folds we randomly select the labeled data used for training the predictive models (both supervised and semi-supervised), while the remaining examples (of the training folds) serve as unlabeled data (we temporarily remove their labels). Finally, the models are tested on the test fold (i.e., inductive evaluation) and on the unlabeled examples with their true labels restored (i.e., transductive evaluation). The procedure is repeated so that each fold is used exactly once as the test set, giving 10 results, which are averaged to obtain a single predictive performance estimation.

Note that the results of transductive and inductive evaluation are not directly comparable, since they are obtained on different test sets. Furthermore, in the inductive evaluation, models that are learned with different amounts of labeled data are evaluated on the same test set, thus the results are directly comparable, whereas in the transductive evaluation, the test set changes as the amount of labeled data changes.

For each training/test split of the cross validation, we optimize the weight parameter w by performing internal 3-fold cross-validation on the available labeled part of the training set. During the internal 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.

We assess the predictive performance of the algorithms by using the average relative root-mean-square-error (RRMSE), defined as follows:

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

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 $\overline{y}_i$ is the mean value (on the training set) of the j^{th} target variable.

To investigate whether the observed differences in performance among the methods are statistically significant, we follow the recommendations given by Demšar [18]. More specifically, we use the corrected Friedman test and the post-hoc Nemenyi test [36]. 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 methods based on the predictive clustering trees are implemented in the Java programming language (version 1.6), while the KBMF method is implemented in the R programming language.

6. Results and discussion

In this section, we present the results of the empirical evaluation of semi-supervised and supervised PCTs for MTR. We first analyze the predictive performance of the methods by using different amounts of labeled data. We then investigate the influence of the weight parameter w, which controls the amount of supervision in the models. Furthermore, we discuss the interpretability and model size of the models. Finally, we analyze the influence of unlabeled data on the performance of SSL-PCTs.

6.1. Analysis of the predictive performances

In this section, we analyze the predictive performance of the methods on the 15 MTR datasets considered in this study, with varying amounts of labeled data. We first focus on the single tree methods (i.e., Base-PCT, SSL-PCT and SSL-PCT-FR) and the KBMF method, followed by the ensemble methods (i.e., CLUS-RF, SSL-RF, SSL-RF-FR and Self-training).

6.1.1. Single tree methods

Table 3 presents the results of single tree methods and the KBMF method. Clearly, semi-supervised PCTs (i.e., SSL-PCT and SSL-PCT-FR) improve over supervised PCTs on most of the datasets for all different amounts of labeled data considered. This is observed in both the transductive and the inductive evaluation scenarios, though, semi-supervised PCT win over supervised PCTs on more occasions in the inductive scenario, suggesting that semi-supervised PCTs have improved capability to generalize on unseen data. The above mentioned observations validate the ability of the proposed semi-supervised approach to successfully exploit unlabeled data.

As mentioned, semi-supervised PCTs win over supervised PCTs on most of the datasets considered, but not on all of them. This is not surprising since the semi-supervised methods are in general known to be domain-dependent [13]. In other words, there are no universally good semi-supervised methods,

Table 3: Results (RRMSE) of the single tree and the KBMF methods. For each dataset, the best result is marked in bold (separately for transductive and inductive evaluation). The '•'/'◦' symbols denote that the SSL-PCTs improve/degrade the performance of Base-PCT, while *W/T/L* denotes the number of wins, ties and loses against Base-PCT. DNF denotes that the method did not finish within 10 days under the available resources.

	Transductive evaluation				Inductive evaluation			
	BasePCT	SSL-PCT	SSL-PCT-FR	KBMF	BasePCT	SSL-PCT	SSL-PCT-FR	KBMF
5% labeled								
Enb	0.447	**0.406** •	**0.406** •	1.084	0.444	**0.409** •	**0.409** •	1.073
Eunite	0.885	**0.873** •	0.876 •	0.999	1.149	1.075 •	1.091 •	**0.999**
F. Kras	0.725	0.714 •	**0.712** •	DNF	0.777	**0.743** •	0.744 •	DNF
F. IRS	0.479	**0.471** •	0.472 •	1.109	0.485	**0.478** •	0.481 •	1.118
F. LandSat	0.854	**0.831** •	0.846 •	1.177	0.842	**0.811** •	0.828 •	1.19
F. Spot	**0.504**	0.509 ◦	0.507 ◦	1.102	0.505	**0.504** •	0.508 ◦	1.12
RF1	**0.402**	**0.402**	**0.402**	0.991	**0.941**	**0.941**	**0.941**	1.054
SCM1D	**0.643**	**0.643**	**0.643**	DNF	**0.71**	**0.71**	**0.71**	DNF
SCM20D	**0.846**	0.851 ◦	0.855 ◦	DNF	0.923	0.922 •	**0.905** •	DNF
Scpf	0.979	0.986 ◦	**0.957** •	1.981	1.01	0.988 •	**0.943** •	1.561
Soil Quality	**1.017**	**1.017**	**1.017**	1.213	**1.025**	**1.025**	**1.025**	1.283
Solar Flare 2	**1**	**1**	1.002 ◦	1.645	0.81	**0.8** •	0.811 ◦	1.128
V. Clustering	**0.894**	**0.894**	0.896 ◦	DNF	**0.942**	**0.942**	0.944 ◦	DNF
V. Condition	0.739	**0.728** •	0.73 •	DNF	0.753	0.741 •	**0.738** •	DNF
Water Quality	**1**	**1**	**1**	1.692	**1**	**1**	**1**	1.836
W/T/L:		6/6/3	7/4/4			10/5/0	8/4/3	
10% labeled								
Enb	**0.332**	**0.332**	**0.332**	1.071	**0.32**	**0.32**	**0.32**	1.014
Eunite	0.842	0.819 •	**0.816** •	1	1.101	1.083 •	1.083 •	**0.999**
F. Kras	0.7	0.689 •	**0.687** •	DNF	0.757	**0.733** •	0.735 •	DNF
F. IRS	**0.439**	**0.439**	**0.439**	1.031	**0.439**	**0.439**	**0.439**	1.02
F. LandSat	0.788	**0.776** •	0.79 ◦	1.161	0.78	0.77 •	**0.752** •	1.17
F. Spot	0.484	0.482 •	**0.481** •	1.027	0.488	**0.484** •	**0.484** •	1.028
RF1	**0.315**	**0.315**	**0.315**	0.996	**0.984**	**0.984**	**0.984**	1.014
SCM1D	**0.602**	**0.602**	**0.602**	DNF	**0.687**	**0.687**	**0.687**	DNF
SCM20D	**0.795**	0.804 ◦	0.801 ◦	DNF	0.893	0.886 •	**0.882** •	DNF
Scpf	0.933	**0.931** •	**0.931** •	1.691	0.905	**0.891** •	0.904 •	1.387
Soil Quality	**0.964**	1 ◦	1.001 ◦	1.075	1.072	**1** •	1.012 •	1.282
Solar Flare 2	1.028	1.021 •	**0.992** •	1.322	**0.817**	0.847 ◦	0.846 ◦	1.193
V. Clustering	**0.862**	0.864 ◦	**0.862**	DNF	0.918	0.91 •	**0.907** •	DNF
V. Condition	0.714	**0.71** •	0.711 •	DNF	**0.719**	0.739 ◦	0.741 ◦	DNF
Water Quality	1	**0.981** •	0.983 •	1.672	1	**0.994** •	1.002 ◦	1.799
W/T/L:		8/4/3	7/5/3			9/4/2	8/4/3	
20% labeled								
Enb	**0.241**	**0.241**	**0.241**	0.734	**0.242**	**0.242**	**0.242**	0.673
Eunite	0.8	**0.775** •	**0.775** •	1	1.125	1.167 ◦	1.165 ◦	**1**
F. Kras	0.673	**0.666** •	0.667 •	DNF	0.75	**0.727** •	0.741 •	DNF
F. IRS	0.398	**0.396** •	**0.396** •	0.871	0.408	**0.405** •	**0.405** •	0.824
F. LandSat	0.72	**0.714** •	0.72	1.176	0.72	**0.706** •	0.72	1.194
F. Spot	0.44	0.429 •	**0.428** •	0.84	0.443	0.432 •	**0.429** •	0.807
RF1	**0.241**	**0.241**	**0.241**	0.997	**0.976**	**0.976**	**0.976**	1.012
SCM1D	**0.557**	**0.557**	**0.557**	DNF	**0.684**	**0.684**	**0.684**	DNF
SCM20D	**0.74**	**0.74**	**0.74**	DNF	**0.925**	**0.925**	**0.925**	DNF
Scpf	1.074	1.04 •	**0.922** •	1.479	0.919	**0.834** •	0.881 •	1.17
Soil Quality	**0.892**	**0.892**	**0.892**	0.989	**0.964**	**0.964**	**0.964**	1.312
Solar Flare 2	1.006	0.996 •	**0.995** •	2.176	1.288	1.274 •	**1.194** •	1.464
V. Clustering	0.833	**0.831** •	0.833	DNF	0.898	**0.89** •	0.898	DNF
V. Condition	0.698	0.694 •	**0.693** •	DNF	0.728	**0.714** •	0.717 •	DNF
Water Quality	0.993	**0.973** •	0.977 •	1.6	0.998	**0.995** •	0.997 •	1.841
W/T/L:		10/5/0	9/6/0			10/4/1	7/7/1	
30% labeled								
Enb	**0.209**	**0.209**	**0.209**	0.405	**0.231**	**0.231**	**0.231**	0.463
Eunite	0.774	0.751 •	**0.748** •	1	1.179	1.171 •	1.17 •	**1**
F. Kras	0.657	0.654 •	**0.65** •	DNF	0.747	**0.722** •	0.727 •	DNF
F. IRS	0.393	**0.392** •	0.393	0.724	0.402	**0.4** •	0.401 •	0.673
F. LandSat	**0.691**	0.715 ◦	0.704 ◦	1.122	0.692	0.691 •	**0.679** •	1.153
F. Spot	**0.42**	**0.42**	**0.42**	0.718	**0.429**	**0.429**	**0.429**	0.697
RF1	**0.283**	**0.283**	**0.283**	0.997	**0.998**	**0.998**	**0.998**	1.012
SCM1D	**0.532**	**0.532**	**0.532**	DNF	**0.667**	**0.667**	**0.667**	DNF
SCM20D	**0.694**	**0.694**	**0.694**	DNF	**0.932**	**0.932**	**0.932**	DNF
Scpf	0.97	0.913 •	**0.912** •	1.426	0.898	0.881 •	**0.878** •	1.154
Soil Quality	**0.863**	**0.863**	**0.863**	0.947	**1.011**	**1.011**	**1.011**	1.335
Solar Flare 2	1.024	1.025 ◦	**1.016** •	3.111	1.591	1.427 •	**1.178** •	2.985
V. Clustering	0.817	**0.815** •	**0.815** •	DNF	0.875	**0.873** •	**0.873** •	DNF
V. Condition	0.687	0.686 •	**0.685** •	DNF	0.728	0.715 •	**0.709** •	DNF
Water Quality	0.983	**0.967** •	**0.967** •	1.561	0.995	**0.988** •	**0.988** •	1.814
W/T/L:		7/6/2	7/7/1			9/6/0	9/6/0	

i.e., such that would always perform better than supervised methods. This observation is confirmed also in our results. Namely, certain datasets are obviously not suitable for semi-supervised PCTs. More specifically, semi-supervised PCTs mostly do not improve over supervised PCTs on the following datasets: *Enb*, *RF1*, *SCM1D*, *SCM20D* and *Soil quality*. The other 10 datasets are, on the other hand, better suited for semi-supervised PCTs.

The KBMF method performed the worst in most of the experiments, though it achieves the best result for *Eunite* dataset (inductive evaluation). Note that, due to the high computational complexity of this method, we were not able to complete experiments for datasets with a large number of examples. In the experimental evaluation of the KBMF method, we used the same parameter settings as the authors of the method [23], i.e., the same kernels and their parameters. Possibly, better performance could be achieved with different kernels and/or their better parametrization. This aspect highlights one advantage of the methods based on decision trees: They are very easy to use, since they do not require much of an expert knowledge. That is, tree-based methods are not very sensitive to parameters, and the values of the parameters that are generally good are known, such as the number of trees for ensemble methods, or the number of randomly selected features at each node of random forests.

We next present the average ranks diagrams for single tree methods (Figure 2). Note that the KBMF method is not included into the average ranks diagrams since the results of this method for some of the datasets are missing due to the above mentioned reason.

We can observe that semi-supervised PCTs (i.e., SSL-PCT and SSL-PCT-FR) are always ranked better than the supervised PCTs (i.e., Base-PCT) – for all the amounts of labeled data considered in both the transductive and inductive evaluation. Statistical significance is obtained when using 20% of the labeled data (both SSL-PCT and SSL-PCT-FR) in transductive evaluation and for 5% (only SSL-PCT) and 30% (only SSL-PCT-FR) of the labeled data in inductive evaluation.

Next, we analyze the effect of feature weighting to the predictive performance of the proposed semi-supervised PCTs. We can observe that in both transductive and inductive evaluation, the SSL-PCT method is ranked better than its feature weighted counterpart SSL-PCT-FR for 5% to 20% of labeled data. However, as the amount of labeled data is increased to 30%, SSL-PCT-FR becomes better ranked. Recall that (Section 4.1) feature weights are calculated by using the available labeled examples. Thus, it is

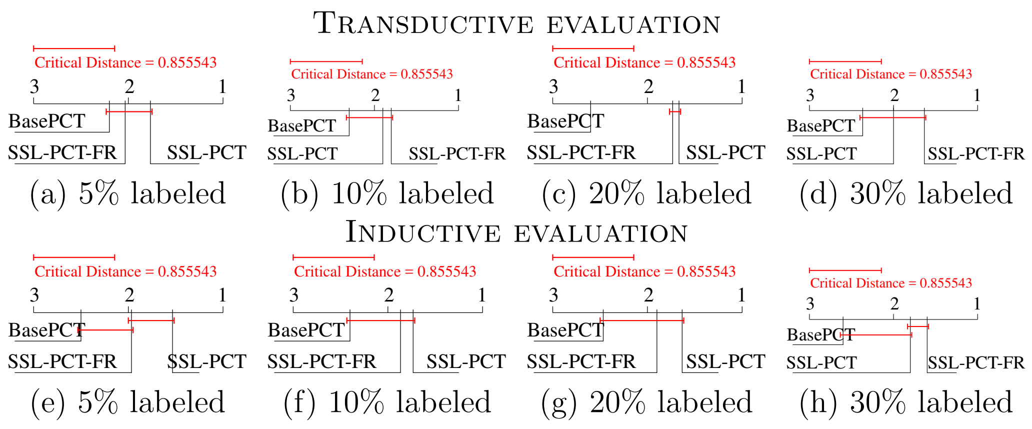


Figure 2: Average ranks diagrams for the performance of the single tree methods. Each graph presents the ranking among the algorithms (the algorithms positioned at the right-most side are the best performing) and the statistical significance of the difference between pairs of algorithms (if their distance is less than the critical distance (at *p-value* = 0.05) there is no statistically significant difference between the two).

possible that as the amount of labeled data increases, better feature weights can be calculated, allowing feature weighted SSL-PCTs to perform better than SSL-PCTs.

6.1.2. Ensemble methods

In the case of single-trees, we observed clear advantage of semi-supervised PCTs over supervised ones, however, this is not entirely preserved in the ensemble setting. Namely, even though semi-supervised random forests (i.e., SSL-RF ad SSL-RF-FR) can improve over supervised random forests (i.e., CLUS-RF), this is observed at fewer occasions as compared to single tree methods. Furthermore, the results show that the improvement of SSL-PCT(-FR) over Base-PCT does not guarantee the improvement of SSL-RF(-FR) over CLUS-RF, for example as observed on *Vegetation Condition* dataset. The opposite is also observed, i.e., SSL-RF(-FR) can improve over CLUS-RF even if SSL-PCT(-FR) does not improve over Base-PCTs, such as on *SCM1D* and *SCM20D* datasets.

The self-training method achieves the best result on some of the datasets (especially in inductive setting). However, the issue of error-propagation of this method is evident. Namely, self-training iteratively uses its own most reliable predictions as additional data in the training process. An error, once made, can reinforce itself in the subsequent iterations, leading to the degra-

Table 4: Results (RRMSE) of the ensemble methods. For each dataset, the best result is marked in bold (separately for transductive and inductive evaluation). The '●'/'○' symbols denote that the semi-supervised method improves/degrades the performance of CLUS-RF, while $W/T/L$ denotes the number of wins, ties and loses against CLUS-RF.

	Transductive evaluation				Inductive evaluation			
	CLUS-RF	SSL-RF	SSL-RF-FR	Self-training	CLUS-RF	SSL-RF	SSL-RF-FR	Self-training
5% labeled								
Enb	0.326	**0.303** ●	0.31 ●	0.328 ○	0.335	**0.309** ●	0.312 ●	0.34 ○
Eunite	0.784	0.775 ●	**0.774** ●	0.785 ○	1.063	1.077 ○	1.074 ○	**1.062** ●
F. Kras	**0.627**	**0.627**	**0.627**	**0.627**	**0.664**	**0.664**	**0.664**	0.668 ○
F. IRS	0.399	0.399	**0.396** ●	0.402 ○	0.404	0.404	**0.4** ●	0.413 ○
F. LandSat	**0.721**	0.727 ○	0.723 ○	0.734 ○	**0.724**	0.727 ○	0.727 ○	0.753 ○
F. Spot	0.437	0.437	**0.435** ●	0.44 ○	0.442	0.442	**0.44** ●	0.452 ○
RF1	**0.279**	**0.279**	**0.279**	0.28 ○	**0.622**	**0.622**	0.634 ○	0.623 ○
SCM1D	0.499	0.493 ●	**0.492** ●	0.503 ○	**0.587**	0.59 ○	**0.587**	0.618 ○
SCM20D	0.657	**0.647** ●	0.65 ●	0.659 ○	**0.768**	0.769 ○	**0.768**	0.784 ○
Scpf	0.915	0.915	0.915	**0.914** ●	**0.867**	**0.867**	**0.867**	**0.867**
Soil Quality	0.904	0.904	**0.903** ●	0.916 ○	**0.963**	**0.963**	0.975 ○	0.975 ○
Solar Flare 2	1.015	1.015	1.015	**1.011** ●	0.853	0.853	0.853	**0.842** ●
V. Clustering	**0.768**	**0.768**	**0.768**	0.777 ○	**0.816**	**0.816**	**0.816**	0.835 ○
V. Condition	**0.664**	**0.664**	**0.664**	**0.664**	**0.67**	**0.67**	**0.67**	0.673 ○
Water Quality	0.955	0.957 ○	0.958 ○	**0.952** ●	0.982	0.98 ●	0.979 ●	**0.975** ●
W/T/L:		4/9/2	7/6/2	3/2/10		2/9/4	4/7/4	3/1/11
10% labeled								
Enb	**0.237**	0.241 ○	0.241 ○	0.241 ○	**0.245**	**0.245**	0.247 ○	0.253 ○
Eunite	0.738	**0.73** ●	**0.73** ●	0.739 ○	1.046	1.062 ○	1.057 ○	**1.04** ●
F. Kras	**0.606**	**0.606**	**0.606**	0.607 ○	**0.654**	**0.654**	**0.654**	0.658 ○
F. IRS	0.368	0.367 ●	**0.365** ●	0.371 ○	0.369	0.37 ○	**0.366** ●	0.38 ○
F. LandSat	**0.65**	**0.65**	**0.65**	0.664 ○	**0.643**	**0.643**	**0.643**	0.678 ○
F. Spot	**0.398**	**0.398**	**0.398**	0.402 ○	**0.398**	**0.398**	0.402 ○	0.413 ○
RF1	**0.226**	**0.226**	**0.226**	0.227 ○	0.62	0.62	0.62	**0.613** ●
SCM1D	0.453	**0.446** ●	**0.446** ●	0.457 ○	**0.571**	0.576 ○	0.573 ○	0.598 ○
SCM20D	0.596	**0.57** ●	0.573 ●	0.598 ○	**0.76**	0.775 ○	0.77 ○	0.773 ○
Scpf	**0.897**	**0.897**	0.9 ○	**0.897**	**0.845**	**0.845**	0.847 ○	0.848 ○
Soil Quality	**0.854**	0.857 ○	0.86 ○	0.867 ○	0.953	0.96 ○	**0.951** ●	0.958 ○
Solar Flare 2	1.006	1.01 ○	1.006	**1.004** ●	0.927	0.921 ●	0.957 ○	**0.88** ●
V. Clustering	**0.739**	**0.739**	**0.739**	**0.739**	**0.794**	**0.794**	**0.794**	**0.794**
V. Condition	**0.649**	**0.649**	**0.649**	**0.649**	**0.658**	**0.658**	**0.658**	0.662 ○
Water Quality	0.943	0.946 ○	0.942 ●	**0.94** ●	0.969	0.968 ●	0.971 ○	**0.963** ●
W/T/L:		4/7/3	5/7/3	2/3/10		2/8/5	4/7/4	4/1/10
20% labeled								
Enb	**0.182**	**0.182**	**0.182**	0.191 ○	**0.205**	**0.205**	**0.205**	0.218 ○
Eunite	0.693	**0.683** ●	0.684 ●	0.694 ○	1.068	1.08 ○	1.077 ○	**1.054** ●
F. Kras	**0.586**	**0.586**	**0.586**	0.587 ○	**0.647**	**0.647**	**0.647**	0.65 ○
F. IRS	**0.336**	**0.336**	**0.336**	0.339 ○	**0.338**	**0.338**	**0.338**	0.348 ○
F. LandSat	**0.604**	**0.604**	**0.604**	0.616 ○	**0.602**	**0.602**	**0.602**	0.632 ○
F. Spot	0.366	0.366	**0.364** ●	0.369 ○	0.373	0.373	**0.371** ●	0.383 ○
RF1	**0.184**	**0.184**	**0.184**	0.185 ○	**0.623**	**0.623**	**0.623**	0.632 ○
SCM1D	0.41	**0.399** ●	**0.399** ●	0.412 ○	**0.559**	0.568 ○	0.572 ○	0.58 ○
SCM20D	0.527	0.489 ●	**0.484** ●	0.528 ○	**0.758**	0.771 ○	0.78 ○	0.764 ○
Scpf	0.917	0.921 ○	0.925 ○	**0.915** ●	0.819	0.814 ●	0.816 ●	**0.811** ●
Soil Quality	**0.808**	**0.808**	**0.808**	0.826 ○	**0.936**	**0.936**	**0.936**	0.939 ○
Solar Flare 2	1.009	**1** ●	1.001 ●	1.007 ●	1.408	**1.224** ●	1.268 ●	1.333 ●
V. Clustering	0.713	0.713	**0.711** ●	0.713	**0.772**	**0.772**	0.773 ○	**0.772**
V. Condition	**0.634**	**0.634**	**0.634**	0.635 ○	**0.649**	**0.649**	**0.649**	0.654 ○
Water Quality	0.929	0.929	**0.927** ●	0.928 ●	0.961	0.961	0.959 ●	**0.954** ●
W/T/L:		4/10/1	7/7/1	3/1/11		2/10/3	4/7/4	4/1/10
30% labeled								
Enb	**0.163**	**0.163**	**0.163**	0.172 ○	**0.193**	**0.193**	**0.193**	0.205 ○
Eunite	0.669	**0.661** ●	**0.661** ●	0.67 ○	1.071	1.08 ○	1.082 ○	**1.064** ●
F. Kras	**0.574**	**0.574**	**0.574**	0.576 ○	**0.645**	**0.645**	**0.645**	0.646 ○
F. IRS	**0.324**	**0.324**	**0.324**	0.327 ○	**0.326**	**0.326**	**0.326**	0.335 ○
F. LandSat	**0.588**	**0.588**	**0.588**	0.593 ○	**0.582**	**0.582**	**0.582**	0.596 ○
F. Spot	**0.354**	**0.354**	**0.354**	0.356 ○	**0.359**	**0.359**	**0.359**	0.368 ○
RF1	**0.166**	**0.166**	**0.166**	0.17 ○	**0.644**	**0.644**	**0.644**	0.667 ○
SCM1D	0.381	**0.369** ●	**0.369** ●	0.383 ○	**0.554**	0.563 ○	0.563 ○	0.568 ○
SCM20D	0.485	**0.439** ●	0.44 ●	0.486 ○	**0.757**	0.781 ○	0.779 ○	0.764 ○
Scpf	0.895	0.901 ○	**0.886** ●	0.896 ○	0.815	0.818 ○	0.816 ○	**0.813** ●
Soil Quality	**0.781**	**0.781**	**0.781**	0.799 ○	0.937	0.937	0.937	**0.932** ●
Solar Flare 2	1.031	1.023 ●	**1.02** ●	1.027 ●	1.617	**1.359** ●	1.408 ●	1.519 ●
V. Clustering	0.699	0.699	**0.697** ●	0.699	**0.761**	**0.761**	0.762 ○	**0.761**
V. Condition	**0.626**	**0.626**	**0.626**	0.627 ○	**0.643**	**0.643**	**0.643**	0.647 ○
Water Quality	0.923	0.923	0.923	**0.921** ●	0.954	0.954	0.954	**0.952** ●
W/T/L:		4/10/1	6/9/0	2/1/12		1/10/4	1/9/5	5/1/9

dation of the performance of the base method – this is observed for several datasets. This consideration makes the semi-supervised random forests and the self-training method somehow complementary, i.e., they usually improve CLUS-RF on different datasets.

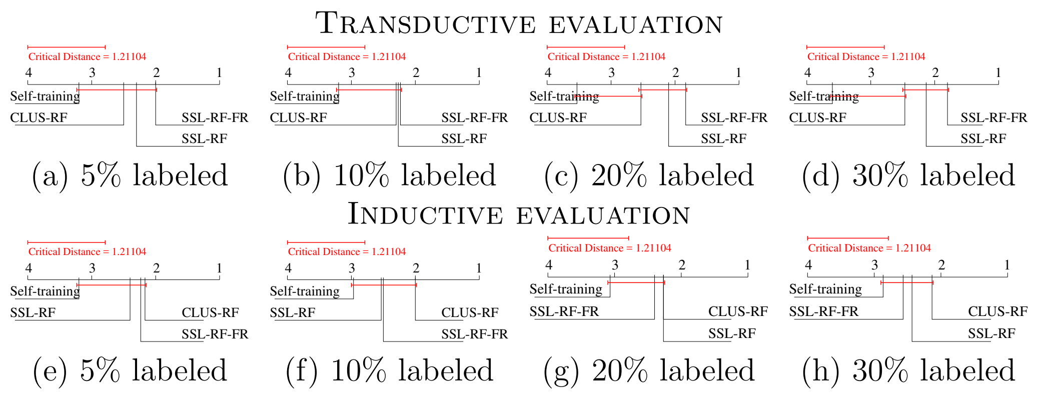
We next present the average ranks diagrams for ensemble methods (Figure 3). We can observe that semi-supervised random forests (i.e., SSL-FR and SSL-FR-FR) are always ranked better than the supervised random forests (CLUS-RF). However, this is evident only in the transductive evaluation, although statistical significance is not achieved. In inductive evaluation, CLUS-RF is ranked better than all of the other semi-supervised ensemble methods, suggesting that if there is a need to predict the labels of examples that are not available during learning, it is better to use supervised random forests. The self-training method is always ranked last, and is also statistically significantly outperformed by SSL-RF nad SSL-RF-FR for 20% and 30% of labeled examples in transductive evaluation.

In the case of ensemble methods, the SSL-RF-FR method is always ranked better than the SSL-RF method (considering transductive evaluation), suggesting that feature weighting is more beneficial for semi-supervised random forests than for semi-supervised PCTs. Possibly, this is because the feature weights are more relevant for the SSL-RF-FR method, since they are determined by using the "same" method, i.e., random forest feature importance procedure (Section 4.1).

6.2. *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 practical applications, since several researchers have found that semi-supervised learning may perform worse than supervised learning [37, 15, 52, 24].

As clarified 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 [13]. Moreover, choosing the appropriate SSL method for the problem at hand is still an open question. In other words, when performing semi-supervised learning, there is always a danger (to some extent) of unlabeled data hurting the predictive performance. Therefore, the



(a) 5% labeled (b) 10% labeled (c) 20% labeled (d) 30% labeled

(e) 5% labeled (f) 10% labeled (g) 20% labeled (h) 30% labeled

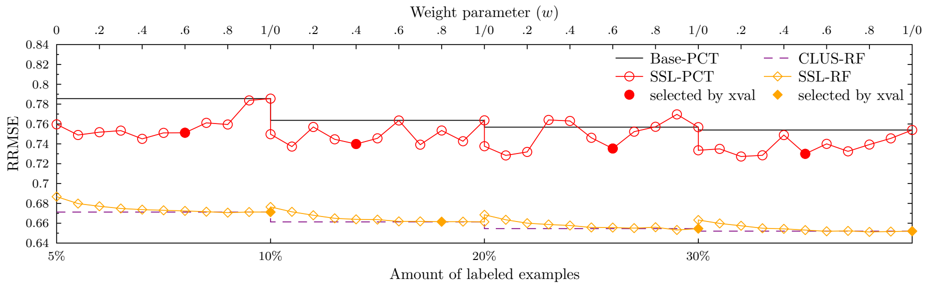
Figure 3: Average ranks diagrams for the performance of the ensemble methods. Each graph presents the ranking among the algorithms (the algorithms positioned at the right-most side are the best performing) and the statistical significance of the difference between pairs of algorithms (if their distance is less than the critical distance (at *p-value* = 0.05) there is no statistically significant difference between the two).

research community is putting effort into developing *safe* semi-supervised methods [34]. Such methods should never perform worse than their supervised counterparts.

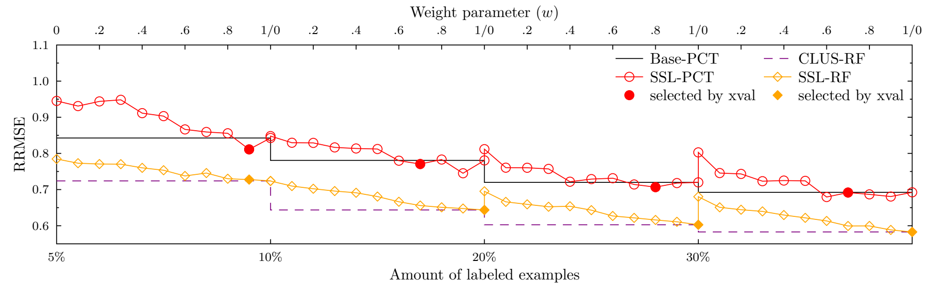
By the virtue of the w parameter, we provide a safety mechanism for SSL-PCTs. In theory, if the optimal value of the parameter w is known, SSL-PCTs and SSL-RF will never perform worse than their supervised counterparts, since Base-PCTs and CLUS-RF are special cases of SSL-PCT and SSL-RF obtained with $w = 1$. 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 Base-PCT and CLUS-RF, though this rarely happened in our empirical evaluation. Considering all the experiments in both the transductive and the inductive evaluation (Table 3), SSL-PCTs preformed better than their supervised counterpart Base-PCT on 57% of occasions, worse on 9% of occasions, and the same on 34% of occasions[3].

Figure 4 illustrates the influence of the parameter w on the predictive

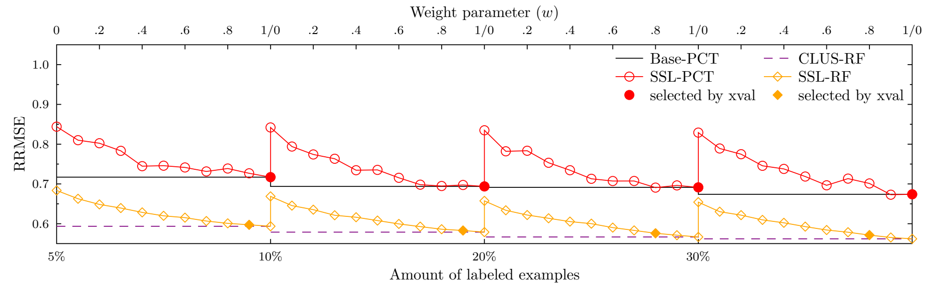
[3]The wins, ties and loses of SSL-PCTs over Base-PCTs are counted considering all the different amounts of labeled data over all the datasets that are used in the experimental evaluation.



(a) *Foresty Kras*



(b) *Forestry LIDAR LandSat*



(c) *SCM1D*

Figure 4: The effect of the w parameter on the performance of the SSL-PCT (red line) and SSL-RF (orange line) methods on 3 datasets: *Foresty Kras*, *Forestry LIDAR LandSat*, *SCM1D*. The values of the w parameter selected by cross-validation and used in the experimental evaluation are denoted with colored markers.

performance for 3 different datasets (in the transductive setting). On the *Foresty Kras* dataset (Fig. 4a), SSL-PCTs improve over Base-PCTs for almost any value of the w parameter, though, an optimal value of w changes as the amount of labeled data changes. Next, on the *Forestry LIDAR LandSat* dataset, SSL-PCT requires a high amount of supervision, in order to improve over Base-PCT. Otherwise, if w is set to a too low value, degradation of performance could occur. Finally, the *SCM1D* dataset (Fig. 4c) is an example of a dataset where using the proposed methods does not improve over its supervised counterparts: Regardless of the value of w, or the amount of labeled data, SSL-PCT is not able to improve over Base-PCT. The value of $w = 1$ is always chosen, and with that, degradation of the performance is avoided.

As exemplified in Figure 4, a general recommendation for the value of w is difficult to provide, since the optimal value of w varies from dataset to dataset, and even within the same dataset as the amount of labeled data changes. Hence, this parameter needs to be optimized by internal cross-validation for each dataset and each amount of labeled data.

6.3. Size and interpretability of supervised and semi-supervised trees

As mentioned previously, interpretability is an important property of predictive models in applications where machine learning is not only used for predictive modelling, but also for knowledge discovery. 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, there is currently no other semi-supervised method for MTR that produces interpretable models.

Table 5 presents the model sizes of supervised and semi-supervised trees. We can observe that semi-supervised trees are, in almost all of the cases, smaller than the supervised trees. Recall that, due to the definition of the variance function used to learn semi-supervised PCTs, they group examples into clusters that are compact both in the descriptive and the target space, while the supervised trees consider only the target space. Obviously, semi-supervised trees have a more strict clustering criterion, which likely is a reason for the smaller tree size. Furthermore, Table 5 demonstrates that in situations with limited availability of labeled data, semi-supervised learning of PCTs offers better interpretability (i.e., smaller PCTs), and also, as we previously discussed, more accurate trees as compared to supervised learning of PCTs. Note that, Table 5 presents the tree sizes of SSL-PCTs, while the tree sizes of the feature weighted variant of the algorithm (SSL-PCT-FR) are not presented since the conclusion are similar.

6.4. The influence of unlabeled data

The semi-supervised PCT learning differs from supervised PCT learning in two aspects: (1) it considers both the descriptive and target space to evaluate the candidate splits, and (2) it uses unlabeled examples in addition to labeled ones. We have demonstrated that semi-supervised PCTs offer

Table 5: Model sizes, in terms of number of nodes, obtained with the supervised PCTs (Base-PCT) and the semi-supervised PCTs (SSL-PCT) on the 15 MTR datasets.

Dataset	Amount of labeled examples							
	5%		10%		20%		30%	
	Base -PCT	SSL -PCT	Base -PCT	SSL -PCT	Base -PCT	SSL -PCT	Base -PCT	SSL -PCT
Enb	6.8	4.4	12.6	12.6	29.2	29.2	41	41
Eunite	22.8	15.6	35	33	79	73.6	149.4	113.2
Forestry Kras	154.2	88.8	294	155	554.4	365.6	785	552
Forestry LIDAR IRS	22.6	13	34.6	34.6	60.6	37.6	79.2	46.6
Forestry LIDAR LandSat	17	11.2	33.6	19.2	57.6	39.4	77.8	59
Forestry LIDAR Spot	23.6	13.6	35.2	25.6	65.6	43.4	84.2	84.2
RF1	93	93	224.2	224.2	478.6	478.6	647.4	647.4
SCM1D	37	37	69.8	69.8	160.2	160.2	252.2	252.2
SCM20D	23.8	17	52.2	38.8	139	139	241.2	241.2
Scpf	3.4	1.6	4.2	5.2	6.6	20.6	9.2	33.4
Soil Quality	3.8	3.8	6.6	1	16.4	16.4	26.2	26.2
Solar Flare 2	2	1	2.8	8.8	3.8	24.4	5.2	6.2
Vegetation Clustering	38.2	38.2	76	70.4	154	131.8	217	201.6
Vegetation Condition	23.4	29.4	46.2	52.2	84	96	124.2	138
Water Quality	1	1	1	5	2.2	10.8	3	17.8
Average:	31.5	24.6	61.9	50.4	126.1	111.1	182.8	164.0

predictive performance that is highly competitive to the one of supervised PCTs, but we might question whether the improvements stem from the combination of (1) and (2), or is 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 is not supplied with unlabeled examples (denoted as SL-PCT). 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 for SL-PCT in the same way as for SSL-PCTs, i.e., via internal 3-fold cross validation.

Figure 5 presents a detailed comparison of improvement/degradation of SL-PCT and SSL-PCT over Base-PCT. We can observe that, even without unlabeled data, the algorithm for semi-supervised learning of PCTs we proposed can improve over standard supervised PCTs (Figure 5, points with the positive x values). However, SSL-PCT method improves over Base-PCT more frequently and more strongly than SL-PCT (most of the points on Figure 5 are above the diagonal), thus, it successfully exploits unlabeled data. These observations suggests that the proposed algorithm for semi-supervised learning of PCTs might be worthy of attention even in its supervised form, though if supplied with unlabeled data it has a better chance to improve the standard supervised PCT algorithm.

7. Conclusions

In this paper, we propose a method for semi-supervised learning of predictive clustering trees and random forests thereof for the task of multi-target

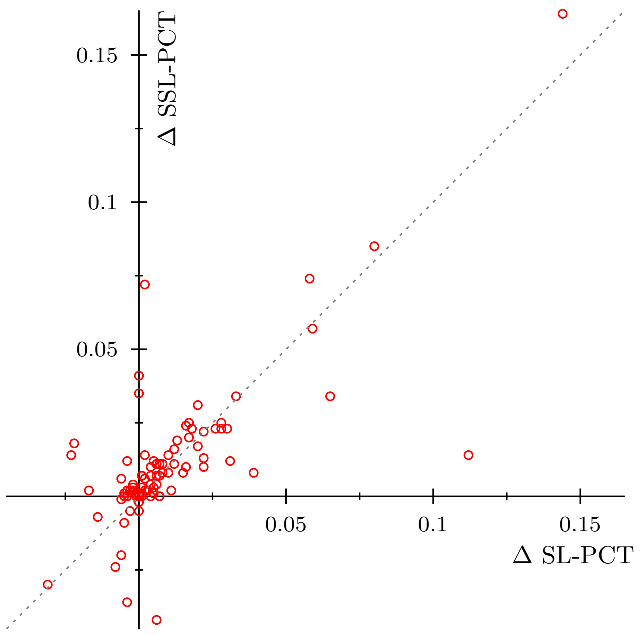


Figure 5: Comparison of the predictive performance of SSL-PCT and SL-PCT with Base-PCT across all datasets and different amounts of labeled data. ΔSSL-PCT and ΔSL-PCT denote the difference between Base-PCT and SSL-PCT or SL-PCT, respectively. Negative values mean that Base-PCT is better, while positive values mean the opposite. SSL-PCT outperforms SL-PCT above the diagonal (dashed line), while SL-PCT outperforms SSL-PCT below the diagonal line. SSL-PCT improves over Base-PCT for positive values of the y-axis, and degrades the performance of Base-PCT for negative values of the y-axis. The same relations hold for SL-PCT, but for the x-axis.

regression. We extensively evaluate the proposed methods on 15 multi-target regression datasets from different domains and analyze their predictive performance, model size and sensitivity to parameters.

We show that semi-supervised predictive clustering trees improve the performance of standard supervised predictive clustering trees and enhance their interpretability (due to the reduced model size). Therefore, in situations where unlabeled data are available and labeled data are a limited asset, semi-supervised predictive clustering trees should be preferred over supervised predictive clustering trees – if the focus is on the knowledge discovery aspect of predictive modelling. Otherwise, if the focus is on the predictive performance, semi-supervised random forests may offer better performance than supervised random forests in a transductive setting.

The variant of semi-supervised random forests that weight the descriptive features by their relevance performed favourably to the variant without feature weighting. In the case of single trees, feature weighting was beneficial for the performance of semi-supervised predictive clustering trees only if enough labeled data was available (i.e., at least 30%). Smaller amounts of labeled data were apparently insufficient to estimate feature relevance well enough.

The proposed methods represent a step towards 'safe' semi-supervised methods, i.e., methods that would always guarantee better or at least equal performance as compared to supervised methods: Due to the built-in safety mechanism, they seldom degraded the performance of their supervised counterparts.

The proposed methods have some intrinsic properties which could be useful in other fields of machine learning besides semi-supervised predictive modelling. 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 this area as well. Next, the method has the intrinsic capability of handing partially labeled data - a scenario highly relevant for tasks with structured outputs. When some parts of the output labels are missing, the corresponding examples are often discarded, or completed by missing data imputation. Our approach could be an elegant way to avoid both of these (undesired) solutions. Finally, 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 (e.g., feature ranking with random forests).

As a direction for future work, we point out the possibility to easily extend the proposed approach to other types of structured outputs (other than multi-target regression), such as multi-target classification, hierarchical multi-label classification and time-series prediction. Therefore, our immediate research efforts will follow this direction.

Acknowledgments

We acknowledge the financial support of the Slovenian Research Agency, via the grant P2-0103 and a young researcher grant to the first author, as well as the European Commission, via the grants ICT-2013-612944 MAESTRA, ICT-2013-604102 HBP and H2020-ICT-688797 TOREADOR.

References

[1] Aho, T., Ženko, B., Džeroski, S., Elomaa, T., 2012. Multi-target regression with rule ensembles. Journal of Machine Learning Research 13 (1), 2367–2407.

[2] Appice, A., Džeroski, S., 2007. Stepwise induction of multi-target model trees. In: Proc. of ECML 2007. Vol. 4701 of LNCS. pp. 502–509.

[3] Asuncion, A., Newman, D., 2007. UCI machine learning repository. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`

[4] Blockeel, H., De Raedt, L., Ramon, J., 1998. Top-down induction of clustering trees. Proc. of the 15th Int'l Conf. on Machine learning, 55–63.

[5] Blockeel, H., Džeroski, S., Grbović, J., 1999. Simultaneous prediction of multiple chemical parameters of river water quality with TILDE. In: Principles of Data Mining and Knowledge Discovery. Vol. 1704 of LNCS. pp. 32–40.

[6] 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.

[7] Breiman, L., 1996. Bagging predictors. Machine Learning 24 (2), 123–140.

[8] Breiman, L., 1996. Out-of-bag estimation. Tech. rep., University of California.

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

[10] 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.

[11] Cardona, H. D. V., Álvarez, M. A., Orozco, A. A., 2015. Convolved Multi-output Gaussian Processes for Semi-Supervised Learning. Vol. 9279 of Lecture Notes in Computer Science. Springer, Berlin, pp. 109–118.

[12] Chapelle, O., Schölkopf, B., Zien, A., 2006. Semi-supervised Learning. Vol. 2. MIT Press.

[13] 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.

[14] 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.

[15] Cozman, F., Cohen, I., Cirelo, M., 2002. Unlabeled data can degrade classification performance of generative classifiers. In: Proc. of the 15th International Florida Artificial Intelligence Research Society Conference. pp. 327–331.

[16] Cunningham, P., Delany, S. J., 2007. k-nearest neighbour classifiers. Tech. rep., UCD-CSI-2007-4.

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

[18] Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30.

[19] 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.

[20] Du, X., 2017. Semi-supervised learning of local structured output predictors. Neurocomputing 220, 151–159.

[21] Džeroski, S., Kobler, A., Gjorgjioski, V., Panov, P., 2006. Using decision trees to predict forest stand height and canopy cover from Landsat and LIDAR data. In: Proc. of the 20th Int'l Conf. on Informatics for Environmental Protection. pp. 125–133.

[22] Gjorgjioski, V., Džeroski, S., 2003. Clustering analysis of vegetation data. Tech. rep., Jožef Stefan Institute.

[23] Gönen, M., Kaski, S., 2014. Kernelized Bayesian Matrix Factorization. IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (10), 2047–2060.

[24] Guo, Y., Niu, X., Zhang, H., 2010. An extensive empirical study on semi-supervised learning. In: Proc. of 10th Int'l Conf. on Data Mining. pp. 186–195.

[25] 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.

[26] 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.

[27] Kocev, D., Slavkov, I., Džeroski, S., 2013. Feature ranking for multi-label classification using predictive clustering trees. In: International Workshop on Solving Complex Machine Learning Problems with Ensemble Methods, in Conjunction with ECML/PKDD. pp. 56–68.

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

[29] 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.

[30] Leistner, C., Saffari, A., Santner, J., Bischof, H., 2009. Semi-supervised random forests. In: Proc. of the 12th Int'l Conf. on Computer Vision. pp. 506–513.

[31] Levatić, J., Ceci, M., Kocev, D., Džeroski, S., 2017. Self-training for multi-target regression with tree ensembles. Knowledge-Based Systems 123, 41–60.

[32] Levatić, J., Ceci, M., Kocev, D., Džeroski, S., 2017. Semi-supervised classification trees. Journal of Intelligent Information Systems, In press, DOI: 10.1007/s10844–017–0457–4.

[33] Levatić, J., Kocev, D., Džeroski, S., 2014. The importance of the label hierarchy in hierarchical multi-label classification. Journal of Intelligent Information Systems 45 (2), 247–271.

[34] Li, Y.-F., Zhou, Z.-H., 2011. Towards making unlabeled data never hurt. In: Proc. of the 28th International Conference on Machine Learning. pp. 1081–1088.

[35] Navaratnam, R., Fitzgibbon, A., Cipolla, R., 2007. The joint manifold model for semi-supervised multi-valued regression. In: Proc. of the 11th IEEE Int'l Conf. on Computer Vision. pp. 1–8.

[36] Nemenyi, P. B., 1963. Distribution-free multiple comparisons. Ph.D. thesis, Princeton University, Princeton, NY, USA.

[37] 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.

[38] Pugelj, M., Džeroski, S., 2011. Predicting structured outputs k-nearest neighbours method. In: Discovery Science. Vol. 6926 of LNCS. pp. 262–276.

[39] Quinlan, J. R., 1992. Learning with continuous classes. In: Proc. of the 5th Australian Joint Conference on Artificial Intelligence. Singapore, pp. 343–348.

[40] 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.

[41] Spyromitros-Xioufis, E., Groves, W., Tsoumakas, G., Vlahavas, I., 2014. Multi-label classification methods for multi-target regression. arXiv preprint arXiv:1211.6581.

[42] 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.

[43] Stojanova, D., 2009. Estimating forest properties from remotely sensed data by using machine learning. MSc thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

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

[45] Struyf, J., Džeroski, S., 2006. Constraint based induction of multi-objective regression trees. In: Knowledge Discovery in Inductive Databases. Vol. 3933 of LNCS. pp. 222–233.

[46] 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.

[47] Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., Vlahavas, I., 2014. Multi-target regression via random linear target combinations. In: Machine Learning and Knowledge Discovery in Databases. Vol. 8726 of LNCS. pp. 225–240.

[48] Ženko, B., 2007. Learning predictive clustering rules. PhD thesis, Faculty of Computer Science, University of Ljubljana, Ljubljana, Slovenia.

[49] Witten, I. H., Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

[50] 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.

[51] Zhang, Y., Yeung, D.-Y., 2009. Semi-supervised multi-task regression. In: Machine Learning and Knowledge Discovery in Databases. Vol. 5782 of LNCS. pp. 617–631.

[52] Zhou, Z.-H., Li, M., 2007. Semi-supervised regression with co-training style algorithms. IEEE Transaction in Knowledge Data Engineering 19 (11), 1479–1493.

[53] Zhu, X., 2008. Semi-supervised learning literature survey. Tech. rep., Computer Sciences, University of Wisconsin-Madison.