Machine Learning (2020) 109:2213–2241
https://doi.org/10.1007/s10994-020-05894-4

Ensembles of extremely randomized predictive clustering trees for predicting structured outputs

Dragi Kocev[1,2,3] · **Michelangelo Ceci**[1,2,4] · **Tomaž Stepišnik**[2,3]

Received: 11 July 2019 / Revised: 20 March 2020 / Accepted: 6 July 2020 / Published online: 17 August 2020

Abstract

We address the task of learning ensembles of predictive models for structured output prediction (SOP). We focus on three SOP tasks: multi-target regression (MTR), multi-label classification (MLC) and hierarchical multi-label classification (HMC). In contrast to standard classification and regression, where the output is a single (discrete or continuous) variable, in SOP the output is a data structure—a tuple of continuous variables MTR, a tuple of binary variables MLC or a tuple of binary variables with hierarchical dependencies (HMC). SOP is gaining increasing interest in the research community due to its applicability in a variety of practically relevant domains. In this context, we consider the Extra-Tree ensemble learning method—the overall top performer in the DREAM4 and DREAM5 challenges for gene network reconstruction. We extend this method for SOP tasks and call the extension Extra-PCTs ensembles. As base predictive models we propose using predictive clustering trees (PCTs)–a generalization of decision trees for predicting structured outputs. We conduct a comprehensive experimental evaluation of the proposed method on a collection of 41 benchmark datasets: 21 for MTR, 10 for MLC and 10 for HMC. We first investigate the influence of the size of the ensemble and the size of the feature subset considered at each node. We then compare the performance of Extra-PCTs to other ensemble methods (random forests and bagging), as well as to single PCTs. The experimental evaluation reveals that the Extra-PCTs achieve optimal performance in terms of predictive power and computational cost, with 50 base predictive models across the three tasks. The recommended values for feature subset sizes vary across the tasks, and also depend on whether the dataset contains only binary and/or sparse attributes. The Extra-PCTs give better predictive performance than a single tree (the differences are typically statistically significant). Moreover, the Extra-PCTs are the best performing ensemble method (except for the MLC task, where performances are similar to those of random forests), and Extra-PCTs can be used to learn good feature rankings for all of the tasks considered here.

Keywords Multi-target regression · Multi-label classification · Hierarchical multi-label classification · Structured output prediction · Feature ranking · Ensembles · Extremely randomized trees · Predictive clustering trees

---

Editors: Larisa Soldatova, Joaquin Vanschoren.

Dragi Kocev
Dragi.Kocev@ijs.si

Extended author information available on the last page of the article

1 Introduction

Supervised learning is one of the most widely researched and investigated areas of machine learning. The goal in supervised learning is to learn, from a set of examples with a known value for a target variable, a function that outputs a prediction for the value of a previously unseen example. However, in many real life predictive modeling problems the output (i.e., the target) is structured (ISO 2007; Panov et al. 2016), meaning that there can be dependencies among the values of the target variable(s). Some examples of structured output datatypes are: a hierarchy of classes (hierarchical multi-label classification—HMC) (Silla and Freitas 2011; Stojanova et al. 2013), multiple continuous variables (multi-target regression—MTR) (Borchani et al. 2015; Spyromitros-Xioufis et al. 2016) or multiple binary variables (multi-label classification—MLC) (Madjarov et al. 2012; Tsoumakas et al. 2010).

These types of structured output prediction (SOP) problems occur very often in various domains, such as life sciences (e.g., predicting gene function, finding disease signatures, predicting toxicity of molecules), ecology (e.g., analysis of remotely sensed data and habitat modeling), multimedia (annotation and retrieval of images and videos) and the semantic web (categorization and analysis of text and web pages). Bearing in mind the needs of these application domains and the increasing quantities of structured data, Dietterich et al. (2008) and Kriegel et al. (2007) listed the task of “mining complex knowledge from complex data” as one of the most challenging problems in machine learning.

Several methods for addressing the task of SOP have been proposed (Borchani et al. 2015; Kocev et al. 2013; Tsoumakas et al. 2010). These methods can be categorized into two groups (Bakır et al. 2007): (1) local methods construct models for predicting parts of the output and then combine the individual models to obtain the overall model (i.e., they construct an architecture of several simpler models) and (2) global methods that construct models for predicting the complete structure as a whole (also known as ’big-bang’ approaches). More specifically, the local methods construct a simple model for each of the target variables separately for MTR and MLC and simple models for parts of the hierarchy for HMC (e.g., one model per hierarchy level, node or branch). For the MTR task, the number of local models to be constructed typically corresponds to the number of targets considered: for a domain with T target variables one needs to construct T predictive models—each predicting a single target. For the MLC task, the number of models is typically equal to the number of possible labels L in the binary relevance approach (Tsoumakas et al. 2010), however, this number can be much bigger in the pairwise approach ($\frac{L(L-1)}{2}$ models), where every simple model discriminates between two class labels. For the HMC task, the number of local models varies depending on the selected approach: as small as the number of levels in the hierarchy or as big as the number of classes in the hierarchy (Ceci and Malerba 2007). The multiple local models (in each of the tasks) are then combined to obtain the overall model. Conversely, the global methods construct a single predictive model that is valid for the complete structure. The prediction of an unseen example here is then obtained by passing the example through the model and retrieving its prediction.

The global methods have several advantages over the local methods (Blockeel et al. 1998; Kocev et al. 2013). First, they exploit the dependencies that exist among the components of the structured output in the model learning phase, which can result in better predictive performance. Next, they are typically more efficient: it can easily happen that the number of components in the output is very large (e.g., hierarchies in functional genomics can have several thousands of components), in which case executing a basic method for each component is not feasible. Furthermore, these dependencies produce models that are typically smaller than the sum of the sizes of the models built for each of the components.

In this paper, we propose an extension of the Extra-Trees algorithm based on the predictive clustering trees (PCTs) framework (Kocev 2011; Kocev et al. 2013). We call this extension Extra-PCTs algorithm. PCTs belong to the group of global methods and can be considered a generalization of standard decision trees towards predicting structured outputs. They offer a unifying approach for dealing with different types of structured outputs and construct the predictive models very efficiently. They are able to make predictions for several types of structured outputs: tuples of continuous/discrete variables, hierarchies of classes and time series. Furthermore, the performance of ensembles of PCTs was extensively evaluated across a variety of tasks and it was shown that they yield state-of-the-art predictive performance: For MLC, Madjarov et al. (2012) and Bogatinovski (2019) performed extensive empirical studies showing random forests of PCTs for MLC among the top-performing methods; For MTR, Mileski (2017), Levatić et al. (2018) and Breskvar et al. (2018) performed extensive comparisons to a variety of competing methods, which listed ensembles of PCTs for MTR among the top-performing methods; and for HMC, ensembles of PCTs yield state-of-the-art predictive performance (Cerri et al. 2016; Ho et al. 2018; Radivojac and colleagues 2013) and have been extensively used for gene function prediction (Radivojac and colleagues 2013; Schietgat et al. 2010; Škunca et al. 2013).

In (Kocev et al. 2013), we evaluated the construction of local and global models for SOP in the context of ensemble learning. More specifically, we focused on the two most widely used ensemble learning techniques: bagging (Breiman 1996) and random forests (Breiman 2001). We showed that both global and local tree ensembles perform better than their single model counterparts in terms of predictive power. Global and local tree ensembles perform equally well, but global ensembles are more efficient and produce smaller models, and need fewer trees in the ensemble to achieve the maximum performance.

In this paper, we investigate a new strategy for learning global models for SOP through ensemble learning. In particular, we extend the Extra-Trees algorithm to the context of SOP. The Extra-Trees algorithm, proposed by Geurts et al. (2006a), is an algorithm for tree ensemble construction, based on extreme randomization of the tree construction algorithm. The algorithm at each node of the tree randomly selects k attributes and, on each of them, randomly selects a split. The k candidate splits are then evaluated and the best split is put in the node. Geurts et al. (2006a) evaluated their approach in the context of single-target regression and classification problems, containing only numerical attributes. The bias/variance analysis of the error revealed that Extra-Trees decrease the variance, while at the same time they increase the bias. If the level of randomization is well adjusted, then the variance almost disappears at the cost of a slight increase in the bias with respect to that of standard trees. In this study, we perform an empirical evaluation of the Extra-Trees algorithm extension in SOP domains, where the descriptive attributes can be continuous, categorical or mixed (both continuous and categorical in the same dataset).

Furthermore, traditional decision tree learning mainly requires design decisions on the definition of the search space and of the heuristics used to explore the search space. In Extra-PCTs, the search space is that of all possible families of trees, where every single tree has a randomization mechanism for the determination of the split and has a prototype function associated to the leaves. The heuristics and the prototype function in the PCTs framework are based on variance reduction and are designed to work for various structured output prediction tasks (i.e., MTR, MLC and HMC). Moreover, the search space and the heuristics are not independent since the evaluation function should be coherent with the prototype function at the leaves. This means that we have proposed a method that differs from that proposed in Geurts et al. (2006a) both in the heuristics and the prototype function. In a nutshell, with the Extra-PCTs algorithm, we investigate the effect of two competing aspects on tree learning: modified search through the additional randomization of the split search procedure and the structured output space through the joint variance function.

The Extra-Trees algorithm has been successfully applied to several practically relevant domains including computer vision (Maree et al. 2005) and gene network inference (Huynh-Thu et al. 2010; Ruyssinck et al. 2014). The applications in the latter domain are especially noticeable: a variant of the method that exploits its feature ranking mechanism (GENIE3 algorithm) has been the overall top performer in the DREAM4 and DREAM5 challenges[1] for gene network inference. We fully exploit this aspect and use the Extra-Trees algorithm not only for prediction purposes, but also for feature ranking in the case of SOP: the first feature ranking method that is general enough to perform ranking for the different types of outputs with a uniform approach.

The major contribution of the work can be summarized as follows:

- An investigation of the effects of the modified search through the space of potential splits and the structured output space on tree induction.
- An extension of the Extra-Trees algorithm towards the task of structured output prediction, including multi-target regression, multi-label classification and hierarchical multi-label classification.
- A comprehensive experimental evaluation of the proposed Extra-PCTs algorithm, including its parametrization.
- A general feature ranking algorithm for an arbitrary SOP task.

The work presented in this paper builds upon our previous preliminary work presented in (Kocev and Ceci 2015), which only considers the MTR task. We extend this work along several dimensions. First of all, we extend the proposed algorithm towards a more general SOP setting, so as to also include MLC and HMC tasks. Consequently, we evaluate the proposed extension on benchmark datasets from the corresponding tasks. Moreover, we consider 11 additional datasets for the MTR task. Next, we include a comparison with the bagging of PCTs (in addition to the random forests of PCTs). Furthermore, we investigate several design choices for parametrization of the proposed method, in terms of selecting the optimal feature subset sizes. We also provide a detailed overview of the related work. Finally, we propose a feature ranking algorithm that treats all of the SOP tasks in a uniform way. We also illustrate the usefulness of the proposed feature ranking algorithm across all of the tasks. All in all, this study is qualitatively and quantitatively improved compared with the previous study.

The remainder of this paper is organized as follows. Section 2 outlines the task definitions and the related work. Section 3 presents the proposed Extra-PCTs algorithm for SOP and feature ranking. Next, Section 4 provides details on the design of the experimental evaluation, whose results are presented and discussed in Section 5. Finally, Section 6 concludes the paper and provides directions for further work.

---

[1] For more information, visit http://dreamchallenges.org/.

2 Background and related work

2.1 Definition of the tasks

A formal task definition and description for the three SOP tasks addressed in this work are given separately in Kocev et al. (2013), Madjarov et al. (2012) and Vens et al. (2008). Namely, Kocev et al. (2013) provide a definition for the of MTR, Madjarov et al. (2012) provide a definition for the of MLC, and (Vens et al. 2008) provide a definition for the of HMC. Considering all of the above definitions, we formally define the task of SOP as follows.

Given:

- A description space X spanned over D independent variables of primitive data types (discrete or continuous), i.e. for each tuple $x_i \in X, x_i = (x_{i_1}, x_{i_2}, ..., x_{i_D})$;
- A target space Y which consists of variables and a definition of some constraints $\mathcal{S}_Y$ on the variables in Y according to which variables are related/ordered;
- A set of examples E, where each example is a pair of tuples from the description and the target space, respectively, 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 in E ($N = |E|$);
- A quality criterion q, which rewards models with high predictive accuracy and low complexity.

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

Here, the function f is represented with ensembles of extremely randomized predictive clustering trees. Depending on the definition of the constraints among the variables ($\mathcal{S}_Y$), we can instantiate the three tasks as follows. If the variables in Y are continuous and $\mathcal{S}_Y$ specifies that these variables should be considered as a tuple, then the task at hand is MTR. Next, if the variables from Y are binary and $\mathcal{S}_Y$ specifies that these variables should be considered as a tuple (or as a set, cf. (Gjorgjioski et al. 2011)), then the task at hand is MLC. Finally, if the variables from Y are binary and $\mathcal{S}_Y$ specifies that there are parent-child relationships among the variables which express a partial order among them, then it is possible to express a hierarchy constraint and specify that if an item belongs to a parent class, it also belongs to a child class. In this case, the task at hand is HMC.

2.2 Related work

The *multi-target regression (MTR)* task has received increased attention by the research community over the past decade. It is also known under the name of multi-output, multi-response or multivariate regression. Borchani et al. (2015) consider two groups of methods for MTR: problem transformation and algorithm adaptation. This division corresponds to the more general grouping of methods for SOP, outlined in the introduction: local and global methods, respectively.

The simplest approach to MTR is to consider it as multiple single-target regression tasks and then apply a standard regression algorithm on each of the single-target tasks (i.e., construct local models). Within this approach it is possible to use any regression method to obtain the local predictive models and then combine their outputs to obtain the predictions for the multiple target variables (Kocev et al. 2013).

As for global methods, in statistics, Brown and Zidek (1980) extend the standard ridge regression to multivariate ridge regression, while Breiman and Friedman (1997) propose the Curds&Whey method, where the relations among the tasks are modeled in a post-processing phase. More recently, some authors investigated kernel/SVM-based methods for MTR. For example, Evgeniou et al. (2005) extend the kernel methods to the case of multi-task learning, using a particular type of kernel (multi-task kernel). They show experimentally that the support vector machines (SVMs) with multi-task kernels have significantly better performance than the ones with single-task kernels. Liu et al. (2009) propose an approach to define the loss functions on the output manifold by considering it a Riemannian submanifold, in order to include its geometric structure in the learning (regression) process. The proposed approach can be used in the context of any regression algorithm.

Another line of research adapts methods for MLC towards the task of MTR. More specifically, Tsoumakas et al. (2014) present an ensemble method for MTR that constructs new target variables via random linear combinations of existing targets. The augmented output space is then exploited by adapting the MLC algorithm RA*k*EL for MTR. Next, Spyromitros-Xioufis et al. (2016) propose two methods: the stacked single-target regression method and the ensemble of regressor chains. The former method corresponds to the binary relevance approach with the addition that it constructs meta-models that exploit the estimated values of the other target variables. The latter method corresponds to the classifier chains method for MLC (Read et al. 2011): it selects a random chain (permutation) of the target variables and builds a predictive model for each target, by considering the predictions of the targets earlier in the chain. The ensemble is constructed by multiple random selections of the chains.

Finally, there are several methods that adapt regression trees and rules for the task of MTR. First of all, Struyf and Džeroski (2006) propose multi-target regression trees and adapt pruning techniques to improve their predictive power. Next, Appice and Džeroski (2007) adapt model trees for MTR. Furthermore, Kocev et al. (2013) consider the multi-target regression trees in an ensemble setting. Finally, Aho et al. (2012) use the ensembles of multi-target regression trees to obtain multi-target regression rule sets.

Multi-label classification (MLC) is already an established predictive modeling task. The methods addressing this task are classified in two groups: problem transformation or algorithm adaptation methods (Tsoumakas et al. 2010). The problem transformation methods transform the multi-label learning problem into one or more single-label classification problems. For smaller single-label problems, there exists a plethora of machine learning algorithms.

Problem transformation methods can be grouped into three categories: binary relevance, label power-set and pair-wise methods. Binary relevance methods use the one-against-all strategy to convert the multi-label problem into several binary classification problems. A closely related method is the classifier chain method and its ensemble extension (Read et al. 2011). This method constructs L binary classifiers linked along a chain. Label power-set (LP) methods combine entire label sets into atomic (single) labels to form a single-label problem (i.e., single-class classification problem). For the single-label problem, the set of possible single labels represents all distinct label subsets from the original multi-label representation. In this way, LP based methods directly take into account the label correlations. Two representative methods are HOMER (Tsoumakas et al. 2008) and RAkEL (Tsoumakas and Vlahavas 2007). HOMER first constructs a hierarchy of the multiple labels and then constructs a classifier for the label sets in each node of the hierarchy. RAkEL constructs each base classifier by considering a small random subset of labels and learning a single-label classifier for the prediction of each element in the power-set of this subset.

Pair-wise methods perform pair-wise or round robin classification with binary classifiers, using $Q \cdot (Q-1)/2$ classifiers covering all pairs of labels (Fürnkranz 2002). To combine these classifiers, the pairwise classification method uses majority voting.

The algorithm adaptation methods extend and customize existing machine learning algorithms for the task of MLC. There are extensions of the following machine learning algorithms: boosting, k-nearest neighbors, decision trees and neural networks. The extended methods are able to directly handle multi-label data. AdaBoost.MH and AdaBoost.MR (Schapire and Singer 2000) are two extensions of AdaBoost for multi-label data. While AdaBoost.MH is designed to minimize Hamming loss, AdaBoost.MR is designed to find a hypothesis which ranks the correct labels at the top. Next, several variants for multi-label learning (ML-*k*NN) of the popular *k*-Nearest Neighbors (*k*NN) lazy learning algorithm have been proposed (Zhang and Zhou 2007). The retrieval of the *k* nearest neighbors is the same as in the traditional *k*NN algorithm. The main difference is the determination of the label set of a test example. Decision tree extension is proposed within the predictive clustering framework (Blockeel et al. 1998). A single predictive clustering tree (PCTs) is constructed by using a splitting criterion that considers all of the labels. The PCTs for MLC are also used in an ensemble setting (Kocev et al. 2013). Neural networks have also been adapted for MLC by introducing a new error function that takes multiple labels into account (Crammer and Singer 2003).

Hierarchical multi-label classification (HMC) is a variant of classification, where a single example may belong to multiple classes at the same time and the classes are organized in a form of hierarchy. Silla and Freitas (2011) survey and categorize the HMC methods, based on some of their characteristics and their application domains. Here, we present and group some existing methods based on the learning technique they use. We group the methods as follows: network based methods, kernel based methods and decision tree based methods.

The network based approaches exploit the information in interaction networks among the examples to obtain a better predictive model (Stojanova et al. 2013). Since the network based approaches are typically based on label propagation, a number of approaches have been proposed to combine predictions of functional networks with those of a predictive model. Tian et al. (2008) use logistic regression to combine predictions from a functional association network with predictions from a random forest.

Barutcuoglu et al. (2006) propose a kernel-based method that uses unthresholded SVMs learned for each class separately and then combine the SVMs by using a Bayesian network, so that the predictions are consistent with the hierarchical relationships. Valentini and Re (2009) also propose a hierarchical ensemble method that uses probabilistic SVMs as base learners. The method combines the predictions by propagating the weighted true path rule both top-down and bottom-up through the hierarchy, which ensures consistency with the hierarchy constraint. Rousu et al. (2006) present a method that defines a joint feature map over the input and output space. Next, it applies SVM based techniques to learn the weights of a discriminant function (defined as the dot product of the weights and the joint feature map). Furthermore, Gärtner and Vembu (2009) propose using counting of super-structures from the output to efficiently calculate (in polynomial time) the *argmax* of the discriminant function.

Clare (2003) adapts a decision tree algorithm to cope with the issues introduced by the HMC task and, specifically, the main contribution is to use the sum of the entropies of the class variables to select the best split. The algorithm, called C4.5H, predicts classes on several levels of the hierarchy and assigns a larger cost to misclassifications in the higher levels of the hierarchy. The resulting tree is then transformed into a set of rules, and the best rules are selected, based on a significance test on a validation set. Geurts et al. (2006b) present a decision tree based approach related to predictive clustering trees (which we use in this paper). This approach starts from a different definition of variance and then kernelizes this variance function. The result is a decision tree induction system that can be applied to structured output prediction, using a method similar to the large margin methods mentioned above. Therefore, this system could also be used for HMC after defining a suitable kernel. To this end, an approach similar to that of Rousu et al. (2006) could be used.

Blockeel et al. (2002) proposed the idea of using predictive clustering trees (Blockeel et al. 1998) for HMC tasks (PCTs for HMC). Their work presents the first thorough empirical comparison of a HMC decision tree method in the context of tree-shaped class hierarchies. Vens et al. (2008) extend the algorithm towards hierarchies structured as directed acyclic graphs (DAGs) and show that learning one decision tree for predicting all classes simultaneously outperforms learning one tree per class (even if those trees are built by taking into account the hierarchy, via so-called hierarchical single-label classification–HSC). Stojanova et al. (2013) adapt the PCTs to consider also the network context of the examples by defining a new distance function that includes also the interaction networks. Kocev et al. (2013) extend the PCT framework in the context of ensemble learning. Finally, Cerri et al. (2015) analyse decision tree methods and evaluation measures for the task of HMC.

Although all the aforementioned methods consider the learning tasks addressed in this paper, none of them proposes an elegant way to tackle all of the considered variants of SOP learning tasks with the same algorithm. Moreover, only a few of them exploit the ensemble learning paradigm, that typically provides significant improvements in the prediction capabilities when compared to their non-ensemble counterparts. In any case, none of the approaches mentioned before exploit the idea of the Extra-PCTs algorithm and extend it to deal with SOP tasks.

3 Extra-PCTs for structured output prediction

3.1 Learning a single Extra-PCT

The predictive clustering tree framework views a decision tree as a hierarchy of clusters (Blockeel et al. 1998; Kocev 2011; Kocev et al. 2013). The top-node corresponds to one cluster containing all the data, which is recursively partitioned into smaller clusters, while moving down in the tree. The PCT framework is implemented in the CLUS system.[2]

PCTs are induced with a standard *top-down induction of decision trees* (TDIDT) algorithm (Breiman et al. 1984). Table 1 outlines the general algorithm for PCT induction. It takes as input a set of examples (E) and outputs a tree. It also defines the prototype functions used in each tree leaf for predicting the label of new examples (e.g., for MTR it calculates the average values for each target from the examples belonging to a given leaf). The heuristic (h) used for selecting the tests (t) in a regular PCT, is the reduction in variance caused by the partitioning ($\mathcal{P}$) of the instances corresponding to the tests (t) (see line 7 of the FindTest procedure in Table 2). Intuitively, by maximizing the variance reduction, the cluster homogeneity is maximized and the predictive performance is improved.

---

[2] CLUS is available for download at http://clus.sourceforge.net.

Table 1 The top-down induction algorithm for PCTs. E is the dataset, k is the size of the attribute subset, t is the split test, h is the heuristic score, $\mathcal{P}$ is the partitioning of the instances, and Prototype(E) is the prototype function that calculates the predictions

```
procedure $PCT$
Input: A dataset $E$, size of attribute subset $k$
Output: A predictive clustering tree
$(t^*, h^*, \mathcal{P}^*) = \mathrm{FindTest}(E, k)$
if $t^* \neq none$ then
    for each $E_i \in \mathcal{P}^*$ do
        $tree_i = \mathrm{PCT}(E_i,\ k)$
    return $\mathrm{node}(t^*,\ \bigcup_i \{tree_i\})$
else
    return $\mathrm{leaf}(\mathrm{Prototype}(E))$
```

Table 2 Extremely randomized test selection for PCTs

```
procedure FindTest
Input: A dataset $E$, size of attribute subset $k$
Output: the selected test ($t^*$), its heuristic score ($h^*$) and the partition ($\mathcal{P}^*$) it induces on the dataset ($E$)
$(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
$A = getAttributeList(E)$
$A_s = selectAttributes(E, k)$
for each attribute $a \in A_s$ do
    $t = selectRandomTest(a)$
    $\mathcal{P}$ = partition induced by $t$ on $E$
    $h = Var(E) - \sum_{E_i \in \mathcal{P}} \frac{|E_i|}{|E|} Var(E_i)$
    if $(h > h^*)$ then
        $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
return $(t^*, h^*, \mathcal{P}^*)$
```

```
procedure selectRandomTest
Input: Attribute $a$ and partition $\mathcal{P}$
Output: A test $t$
$t = none$
$A_v = getAttributeValues(a, \mathcal{P})$
if $a$ is numerical then
    $a_M = getMaxValue(A_v)$
    $a_m = getMinValue(A_v)$
    $a_c = rndCutPoint(a_m, a_M)$
    $t = a < a_c$
if $a$ is categorical then
    $A_s = rndNonEmptySet(A_v, \mathcal{P})$
    $t = a \in A_s$
return $t$
```

The extremely randomized variant of PCTs introduces randomization in the test selection (Table 2) (Geurts et al. 2006a; Kocev and Ceci 2015). More specifically, it requires an input parameter (k) that controls the number of attributes considered at each node of the tree. The test selection procedure randomly selects k attributes and from each attribute randomly selects a split. For each of the k selected attributes, the algorithm selects the split in two different ways, depending on the attribute type. If the attribute is numeric the splitting point is selected randomly from the set of possible splitting points. Possible splitting points are found in the set of values of the attribute in the training set associated to the specific node. If the attribute is categorical (i.e., nominal), then a non-empty subset of values of the attribute in the training set associated to the specific node is randomly selected.

It is noteworthy that our approach shows the best benefits with respect to classical regression/decision tree learning in the case of categorical attributes. In fact, differently from continuous attributes, where the number of possible splitting points evaluated in classical regression/decision tree learning is proportional to the number of examples, in the case of categorical attributes the possible number of splits is proportional to 2^p, where p is the number of distinct values for the specific attribute. Although a theorem by Breiman et al. (1984) (Theorem 4.5, Proposition 8.16) proves that the best binary split can be identified among $p-1$ partitions of attribute values, this theorem requires an ordering of the attribute values on the basis of the simple mean of Y, which means that it only applies when simple regression functions are associated to the leaves (which is not the case of SOP). This discussion further motivates our approach that does not (cannot) resort to this theorem, but is potentially able to evaluate any possible partitioning of categorical attributes without posing any constraint on the partitions.

The k-candidate tests are evaluated using the variance reduction heuristic and the best test is selected. Obviously, the larger the variance reduction (h in the procedure *FindTest*-see Table 2), the better the split. If we set the value of k to 1, this algorithm works in the same way as the Random Tree algorithm proposed by Witten and Frank (2005). The advantage with respect to the Random Tree algorithm is that in the approach we adopt there is still a non-random selection based on some evaluation measure (i.e., variance reduction).

In order to take into account the structure of the output space, the variance $Var(E)$ needs to be instantiated differently for the various output structures (Kocev 2011; Kocev et al. 2013).

PCTs for MTR. For the MTR task, the variance is calculated as $Var(E) = \sum_{j=1}^{T} Var(E, Y_j)$, where $Var(E, Y_j)$ is the normalized variance (using the standard deviation of the variables) of the variable Y_j in the set E, and T is the number of target variables. The variances of the target variables are normalized so that each target variable contributes equally to the overall variance. This is due to the fact that the target variables can have completely different ranges. The prototype function returns, for each example, a vector whose elements represent the values of the target variables.

PCTs for MLC. These are PCTs able to predict multiple binary (and thus discrete) targets simultaneously. Therefore, the variance function for the PCTs for MLC is computed as the sum of the Gini indices of the target variables, i.e., $Var(E) = \sum_{j=1}^{T} Gini(E, Y_j)$. The prototype function returns a vector of probabilities that an instance belongs to a given class for each target variable. Once these probabilities are computed, a threshold can be used to determine the actual classes of instances.

PCTs for HMC **.** The variance and prototype for PCTs for the HMC are defined as follows. First, the set of labels of each example is represented as a vector with binary components; the j-th component of the vector is 1 if the example belongs to class c_j and 0 otherwise. The variance of a set of examples E is defined as the average squared distance between each example's class vector (L_i) and the set's mean class vector ($\bar{L}$): $Var(E) = \frac{1}{|E|} \cdot \sum_{E_i \in E} d(L_i, \bar{L})^2$. The similarity at higher levels of the hierarchy is more important than the similarity at lower levels. Hence, the distance measure used is a weighted Euclidean distance: $d(L_1, L_2) = \sqrt{\sum_{j=1}^{T} w(c_l) \cdot (L_{1,j} - L_{2,j})^2}$, where $L_{i,j}$ is the jth component of the class vector L_i of an instance E_i, T is the size of the class vector, and the class weights $w(c)$ decrease with the depth of the class in the hierarchy. More precisely, $w(c) = w_0 \cdot w(p(c))$, where $p(c)$ denotes the parent of class c and $0 < w_0 < 1$). The mean $\bar{L}$ of the class vectors of the examples in the leaf is stored as a prediction (prototype function). Note that the value for the ith component of $\bar{L}$ can be interpreted as the probability that an example arriving at the given leaf belongs to class c_i. The prediction for an example that arrives at the leaf can be obtained by applying a user-defined threshold τ to the probability. Moreover, when a PCT makes a prediction, it preserves the hierarchy constraint (each instance that belongs to a class c also belongs to the ancestor classes of c). More details about the variance instantiation of PCTs for the different tasks can be found in (Kocev et al. 2013; Vens et al. 2008).

3.2 Creating an ensemble of Extra-PCTs algorithm

The extremely randomized PCTs are very unstable predictive models because of the intense randomization at each node. Consequently, such PCTs are only meaningful when used in combination with an ensemble learning framework. In this work, we construct ensembles of extremely randomized PCTs (Extra-PCTs) by learning each of the base predictive models on the complete training set and each of them uses different, randomly selected attributes in the nodes. The randomization introduced at each node of the Extra-PCTs ensures that the base predictive models will be diverse among themselves. It is clear that the level of randomization is strongly dependent on the selection of the parameter K—the number of attributes that are retained at each node. Similarly as in random forests ensemble method, it is given by a function of the total number of descriptive attributes D (e.g., $k = 1$, $k = \lfloor \sqrt{D} + 1 \rfloor$, $f(D) = \lfloor log_2(D) + 1 \rfloor$, $k = D$) (Breiman 2001).

Depending on the application, one can choose to use different values for k. In this study, we investigate the effect of the function used to initialize k on the performance of the ensemble for the three SOP tasks: MTR, MLC and HMC.

In the Extra-PCTs algorithm ensemble, the prediction for a new instance is obtained by combining the predictions of all the base predictive models. For the different tasks this is performed as follows:

- For the MTR task, the prediction for each target variable is computed as the average of the predictions obtained from each tree. Note that this solution exploits possible dependencies in the output space, since clusters used for prediction (and their hierarchical organization, i.e., the tree) have been built by taking into account the whole output space.
- For the MLC task, the prediction for each target variable is computed as the average of the probabilities per class (i.e., *probability distribution vote* (Bauer and Kohavi 1999) obtained from each tree.
- For the HMC task, the prediction of the whole output hierarchy is obtained by averaging the predictions of the base predictive models (i.e., the probabilities for each of the labels) and then applying the same thresholding procedure as the prototype calculation in each tree leaf.

3.3 Feature ranking for structured outputs with Extra-PCTs

Tree-based ensembles can be used to obtain feature a ranking of the descriptive variables. Breiman (2001) proposed exploiting the random forest mechanism to obtain a feature relevance, by using random permutations of the values of the descriptive variables (in the so-called *out-of-bag*) to assess their relevance to the target variable. However, the permutation of the values of the descriptive variables (for each base predictive model) entails additional computational cost in domains with a large number of descriptive variables and a large number of examples. One way to address this issue is to consider a different variable importance score.

In order to avoid this problem, Huynh-Thu et al. (2010) propose using the reduction of the variance in the output space at each test node in the tree (the resulting algorithm is named GENIE3). Namely, the variables that reduce the variance of the output more are, consequently, more important than the ones that reduce the variance less. Hence, for each descriptive variable we measure the reduction of variance it produces when selected as a splitting variable. If a variable is never selected as a splitting variable (in any of the trees in the ensemble) then its importance will be 0.

The GENIE3 algorithm has been heavily evaluated for single-target regression tasks (e.g., for gene regulatory network reconstruction) (Huynh-Thu et al. 2010). Here, we extend it towards the three SOP tasks discussed and name the resulting algorithm ExtraPCTs-GENIE3. The basic idea adopted for feature ranking follows the spirit of GENIE3, but we use the Extra-PCTs algorithm for building the ensemble. This gives us the opportunity to directly exploit the variance functions defined in Section 3.1 when measuring the reduction of variance that a variable produces.

Recall that there are two competing aspects in the Extra-PCTs learning at play: modified search through the additional randomization of the split search procedure and the structured output space through the joint variance function. The effects of these were not previously investigated in the context of learning predictive models, even less so, this was studied for the task of feature ranking for structured output prediction. There are only a handful of methods available to perform feature ranking for a specific structured output prediction task (e.g., feature ranking for MLC or HMC) and there are no other methods outside the predictive clustering framework that can perform feature ranking for all of the structured output prediction tasks. Now, the two competing aspects of Extra-PCTs learning also influence the subsequent feature rankings—here, we illustrate that the proposed methodology indeed can yield a relevant feature ranking across the spectrum of structured output prediction tasks considered.

3.4 Computational complexity of the Extra-PCTs algorithm

One of the best advantages of the Extra-PCTs ensembles is their computational efficiency. In (Kocev et al. 2013), we discuss the computational cost of an ordinary PCT and ensembles of PCTs extensively. The computational cost of constructing an ordinary PCT for SOP can be summarized as

$$\mathcal{O}(D \cdot N \log^2 N) + \mathcal{O}(D \cdot T \cdot N \log N) + \mathcal{O}(N \log N), \tag{1}$$

where D is the number of descriptive attributes, N is the number of examples and T is the size of the output measures as the number of target variables (for MTR and MLC) or the number of classes in the hierarchy (for HMC). The three terms of the computational cost can be related to different phases of the tree learning procedure: The first term of the cost relates to the sorting of the (numeric) attributes at each node, the second term relates the cost of calculating the best split and the third term relates to sorting the examples to the subtrees, i.e., applying the split on the training instances. These terms consider that the tree is balanced and bushy, as in (Witten and Frank 2005), hence its depth is in the order of $\log N$.

The cost of constructing Extra-PCTs can be derived as follows. Two procedures are executed during the construction of each node of the tree: calculating the best split out of the k randomly selected candidate splits at a cost of $\mathcal{O}(k \cdot T \cdot N)$, and applying the split to the training instances with a cost of $\mathcal{O}(N)$. Furthermore, we assume that the tree is balanced and bushy, hence its depth is in the order of $\log N$, i.e., $\mathcal{O}(\log N)$. Bearing this in mind, the total computational cost of constructing a single tree is

Table 3 Properties of the datasets with multiple continuous targets (MTR): number of examples (N), number of descriptive attributes (discrete/continuous, D/C), and number of target attributes (T)

Name of dataset	N	D/C	T
atp1d	337	0/411	6
atp7d	296	0/411	6
collembolaV2	393	8/39	3
edm1	154	0/16	2
enb	768	0/8	2
Forestry-LIDAR-Landsat	6218	0/150	2
Forestry-LIDAR-IRS	2731	0/29	2
Forestry-LIDAR-Spot	2731	0/49	2
jura	359	0/15	3
oes10	403	0/298	16
oes97	334	0/263	16
osales	639	0/401	12
PPMI	713	0/138	35
rf1	9125	0/64	8
rf2	9125	0/576	8
scm1d	9803	0/280	16
scm20d	8966	0/61	16
scpf	1137	3/4	3
soil-quality	1944	0/142	3
VegetationCondition	16967	1/39	7
Water-quality	1060	0/16	14

$$\mathcal{O}(k \cdot T \cdot N \log N) + \mathcal{O}(N \log N). \tag{2}$$

Comparing formulas (1) and (2), we can note that Extra-PCTs have lower computational complexity as compared to regular PCTs. The ensembles usually amplify the computational cost of the base predictive models linearly with the number of base models. Consequently, the cost of an Extra-PCTs ensemble is lower than the cost of a regular ensemble.

4 Experimental design

In order to empirically evaluate Extra-PCTs, we perform experiments for all the SOP learning tasks considered: MTR, MLC and HMC. In this section, we first describe the datasets used, then we introduce the research questions guiding the experimental design and, finally, we describe the experimental setting.

4.1 Data description

The datasets with multiple continuous targets for the MTR task (21 in total, see Table 3) are mainly from the domain of ecological modeling. The datasets with multiple binary targets for the MLC task (10 in total, see Table 4) are from three domains: biology, multimedia and text categorization. The datasets that have classes organized in a hierarchy for the

Table 4 Properties of the datasets with multiple binary targets (MLC): number of examples in the training/testing dataset (N_{tr}/N_{te}), number of descriptive attributes (discrete/continuous, D/C), the total number of labels (Q) and label cardinality (l_c)

	N_{tr}/N_{te}	D/C	Q	l_c
birds	322/323	2/258	19	1.01
emotions	391/202	0/72	6	1.87
scene	1211/1159	0/294	6	1.07
yeast	1500/917	0/103	14	4.24
medical	645/333	1449/0	45	1.25
enron	1123/579	1001/0	53	3.38
corel5k	4500/500	499/0	374	3.52
tmc2007	21519/7077	500/0	22	2.16
mediamill	30993/12914	0/120	101	4.38
bibtex	4880/2515	1836/0	159	2.40

The problems are ordered by their overall complexity roughly calculated as $\#tr.e. \times D \times Q$

Table 5 Properties of the datasets with hierarchical targets (HMC): number of examples in the training/testing dataset (N_{tr}/N_{te}), number of descriptive attributes (discrete/continuous, D/C), number of classes in the hierarchy ($|\mathcal{H}|$), maximum depth of the classes in the hierarchy ($\mathcal{H}_d$), average number of labels per example ($\overline{\mathcal{L}}$), and average number of leaf labels per example ($\overline{\mathcal{L}_L}$)

Domain	N_{tr}/N_{te}	D/C	$\|\mathcal{H}\|$	$\mathcal{H}_d$	$\overline{\mathcal{L}}$	$\overline{\mathcal{L}_L}$
ImCLEF07D	10000/1006	0/80	46	3.0	3.0	1.0
ImCLEF07A	10000/1006	0/80	96	3.0	3.0	1.0
Diatoms	2065/1054	0/371	377	3.0	1.95	0.94
Enron	988/660	0/1001	54	3.0	5.30	2.84
Reuters	3000/3000	0/47236	100	4.0	3.20	1.20
WIPO	1352/358	0/74435	183	4.0	4.0	1.0
Expression–FunCat	2494/1291	4/547	475	4.0	8.87	2.29
SCOP-GO	6507/3336	0/2003	523	5.5	6.26	0.95
Sequence-FunCat	2455/1264	2/4448	244	4.0	3.35	0.94
Yeast-GO	2310/1155	5588/342	133	6.3	5.74	0.66

Note that the values for $\mathcal{H}_d$ are not always a natural number because the hierarchy has a form of a DAG and the maximum depth of a node is calculated as the average of the depths of its parents

HMC task (10 in total, see Table 5) come from various domains: biology, text classification and image annotation/classification. Note that two datasets from the biological domain have the hierarchy of labels organized as a DAG (they have GO - Gene ontology - in the dataset name), while the remaining datasets have the hierarchy of labels organized as a tree. The datasets for the MLC and HMC task come pre-divided into training and testing parts, thus, in the experiments, we use them in their original format as typically done in the literature (Kocev et al. 2013; Madjarov et al. 2012). The training part usually comprises around 2/3 of the complete dataset, while the testing part the remaining 1/3 of the dataset. For more information on the datasets, we refer the reader to the repositories available at: http://mulan.sourceforge.net/datasets-mlc.html, https://dtai.cs.kuleuven.be/clus/hmc-ens/ and http://kt.ijs.si/DragiKocev/PhD/resources/, as well as (Kocev et al. 2013; Madjarov et al. 2012) and the references therein.

In order to facilitate replication of all the experiments, we implemented the method proposed in this paper in the latest version of CLUS, already available in the public CLUS repository at http://clus.sourceforge.net.

4.2 Experimental setup

We design the experimental evaluation of the proposed method by bearing in mind the following research questions:

1. What is the number of base predictive models in the ensemble to obtain good predictive performance? Is this number stable across the tasks?
2. What is the optimal number of splits to be considered at each node in the tree construction for each of the SOP tasks?
3. Do Extra-PCTs ensembles yield better predictive performance than a single PCT?
4. How does the predictive performance of Extra-PCTs compare to the predictive performance of standard tree-ensemble methods, such as bagging and random forest of PCTs?
5. Can Extra-PCTs be used to obtain a feature ranking for domains with structured outputs?

In order to answer these five questions, we design the following experimental setup. The predictive performance of the methods on the MLC and HMC datasets is assessed using the train-test splits from the original datasets, while for the MTR datasets we perform 10-fold cross-validation.

There are multiple performance measures in use for SOP tasks. Specifically, for MTR we considered the correlation coefficient, the root mean squared error and the relative root mean squared error (RRMSE). For MLC, Madjarov et al. (2012) present several performance measures: example-based (*Hamming loss*, F_1 *score*, *accuracy*, etc.), ranking-based (*ranking loss*, *one-error*, etc.) and label-based (*micro precision*, *macro precision*, etc.). For HMC, there are Area Under the Precision-Recall Curve ($AUPRC$) and Area Under the Average Precision-Recall Curve ($\overline{AUPRC}$), presented by Vens et al. (2008).

In the results section, we will focus on RRMSE for MTR, threshold independent *ranking loss* for MLC and $\overline{AUPRC}$ for HMC. While focusing on other performance measures slightly affects the parameter instantiations, the overall conclusions from the experiments remain similar.

Next, we define the parameter values used in the algorithms for constructing the single trees and the ensembles of PCTs. For the single trees, we use F-test as a pruning mechanism (Vens et al. 2008). Specifically, we check whether a given split/test in an internal node of the tree results in a reduction in variance that is statistically significant at a given significance level. If there is no split/test that can satisfy this, then the node is converted to a leaf. An optimal significance level is selected by using internal 3-fold cross validation, from the following values: 0.125, 0.1, 0.05, 0.01, 0.005 and 0.001.

The proposed Extra-PCTs require two input parameters: feature subset size at each node (k) and number of base predictive models. In this study, we investigate the influence of k on the predictive power of the Extra-PCTs by setting its value to various fractions of D: 0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 0.75 and 1. Next, we construct ensembles with 10, 25, 50, 75, 100, 150, 200 and 250 base predictive models.

For a better comparison, we also used random forests of PCTs with the same values of k and number of trees. The influence of the feature subset size on the performance of random forests of PCTs for MTR, MLC and HMC has not been investigated before. Kocev et al. (2013) use the recommended values as from the simpler classification and regression, but discuss that using *log* from the features might be undersampling and may lead to suboptimal results.

Notably, random forests of PCTs, where we look at all the D attributes at each node, are equivalent to bagging ensembles. Following the findings from the study conducted by Bauer and Kohavi (1999), the trees in all of the ensembles were not pruned.

The feature ranking using ExtraPCTs-GENIE3 is performed as follows. For each of the tasks, we select a single dataset to showcase the potential of the algorithm. We then enlarge the feature spaces of each of the datasets by adding 100 random variables. We investigate whether ExtraPCTs-GENIE3 will place the random variables at the tail of the feature ranking. The Extra-PCTs ensembles consisted of 100 trees and k was set to D.

In order to assess the statistical significance of the differences in performance of the studied algorithms, we adopt the recommendations by Demšar (Demšar 2006) for the statistical evaluation of the results. In particular, we use the Friedman test (Friedman 1940) for statistical significance—it is a non-parametric test for multiple hypotheses testing. It ranks the algorithms according to their performance for each dataset separately, thus the best performing algorithm gets the rank of 1, second best the rank of 2 etc; and in case of ties it assigns average ranks. Afterwards, to check where the statistically significant differences appear (among which algorithms), we use the Nemenyi post-hoc test (Nemenyi 1963) when we compare all of the methods with each other. In this post-hoc test, the performance of two classifiers is significantly different if their average ranks differ more than some critical distance. The critical distance depends on the number of algorithms, number of datasets and critical value (for a given significance level) that is based on the Studentized range statistic and can be found in statistical textbooks. We present the results from the statistical analysis with *average rank diagrams* (Demšar 2006). The ranks are depicted on the axis, in such a manner that the best ranking algorithms are at the left-most side of the diagram. The lines connect the algorithms whose average ranks are smaller than the critical distance (in our study, the significance level is set to 0.05). The difference in the performance of the algorithms connected with a line is not statistically significant at the given significance level.

5 Results and discussion

In this section, we present and discuss the results from the comprehensive experimental evaluation. We first investigate the influence of the feature subset size considered at each node and the size of the ensemble (i.e., the number of base predictive models used in an ensemble). We next compare the performance of Extra-PCTs with the performance of a single PCT as well as with the performance of random forests of PCTs and bagging of PCTs. Finally, we show that Extra-PCTs can be used for performing feature ranking. All of the discussion is carried out for each of the tasks considered here: multi-target regression (MTR), multi-label classification (MLC) and hierarchical multi-label classification (HMC).

5.1 Influence of the feature subset size and the number of base predictive models

We discuss the results for each task separately. We first focus on the MTR task. Figures 1 and 2 depict the MTR performance obtained for different values of k and number of base predictive models for the Extra-PCTs and random forests of PCTs, respectively. We start by noticing that for both Extra-PCTs and random forests of PCTs increasing the number of trees in the ensemble generally improves the performance. However, the improvement of

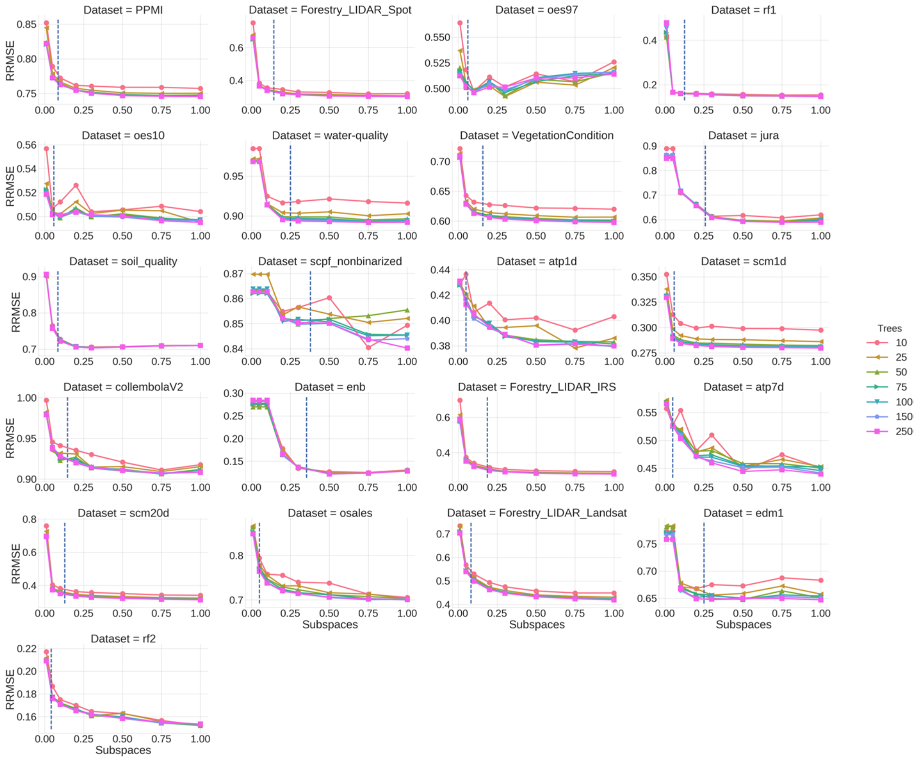


Fig. 1 Results for Extra-PCTs on MTR datasets for different numbers of attributes considered in each node (subspaces as a fraction of D) and different numbers of base predictive models (i.e., no. of trees: 10, 25, 50, 75, 100, 150, 250). The vertical dashed line is at the value for the square root from the descriptive features

the performance starts to saturate after adding 50 trees in the ensembles, i.e., after 50 trees the performance improvements are rarely noticeable. Furthermore, on the datasets with very few attributes (e.g., the dataset *enb*), the first few points show the same performance because k is the same for those ratios (0.01, 0.05 and sometimes also 0.1).

Conversely, the effect of k on performance is not as straightforward and shows more variance among the datasets. Let us first focus on Extra-PCTs. Increasing the value of k often improves the performance. The improvement is large at first, but then diminishes as k grows. This is not in line with the recommendations for single-target regression which suggest using all features (Geurts et al. 2006a). On some datasets the performance starts decreasing when k approaches D (e.g., *enb*, *soil_quality* and most notably *oes*97, where the performance diminishes much sooner). We can make similar observations for random forests of PCTs, with the addition that the performance degradation at higher values for k is more noticeable. In the extreme case of the *oes*97 dataset, random forests seem to offer the best performance at the lowest value of k.

Figures 3 and 4 show the results for the MLC task obtained with Extra-PCTs and random forests of PCTs, respectively. Increasing the number of trees again improves the performance for both methods, also with smaller improvements for higher values as for MTR. For Extra-PCTs there are three datasets, where the performance decreases with the increase in the value of k (*corel*5*k*, *bibtex*, *enron*), five if we also include the *medical*

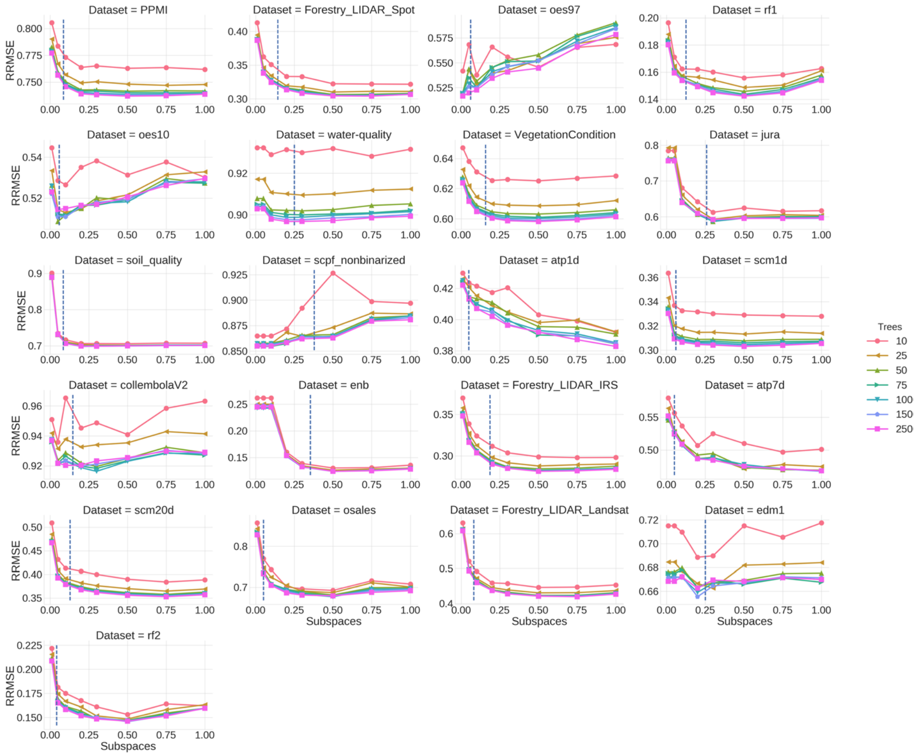


Fig. 2 Results for random forests of PCTs on MTR datasets for different numbers of attributes considered in each node (subspaces as a fraction of *D*) and different numbers of base predictive models (i.e., no. of trees: 10, 25, 50, 75, 100, 150, 250). The vertical dashed line is at the value for the square root from the descriptive features

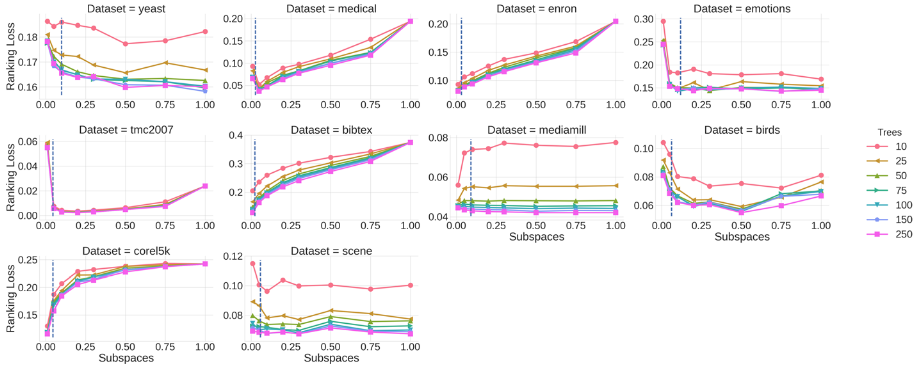


Fig. 3 Results for Extra-PCTs on MLC datasets for different numbers of attributes considered in each node (subspaces as a fraction of *D*) and different numbers of base predictive models (i.e., no. of trees: 10, 25, 50, 75, 100, 150, 250). The vertical dashed line is at the value for the square root from the descriptive features

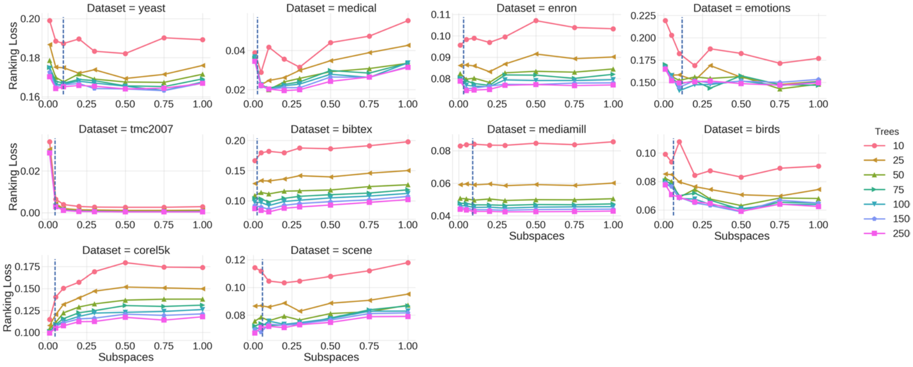


Fig. 4 Results for random forests of PCTs on MLC datasets for different numbers of attributes considered in each node (subspaces as a fraction of D) and different numbers of base predictive models (i.e., no. of trees: 10, 25, 50, 75, 100, 150, 250). The vertical dashed line is at the value for the square root from the descriptive features

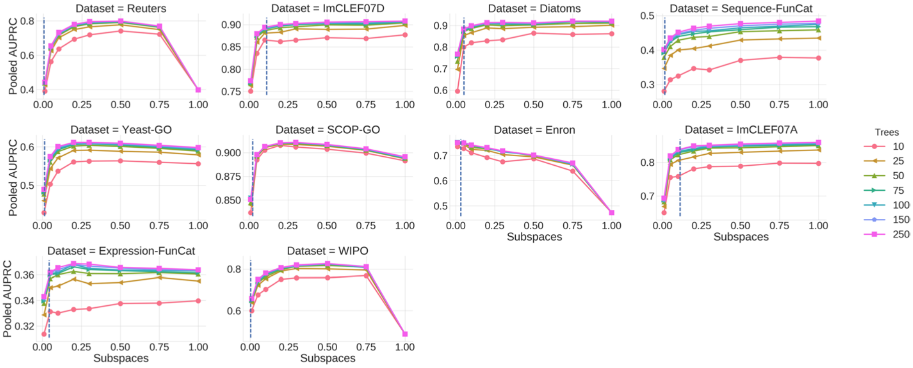


Fig. 5 Results for Extra-PCTs on HMC datasets for different numbers of attributes considered in each node (subspaces as a fraction of D) and different numbers of base predictive models (i.e., no. of trees: 10, 25, 50, 75, 100, 150, 250). The vertical dashed line is at the value for the square root from the descriptive features

and *tmc*2007 datasets, where the smallest k is an exception. These five datasets only have binary attributes. This means that for every attribute there is only one possible split. Because there is no bootstrapping performed for Extra-PCTs ensembles, all members are built on the same training set. On these datasets, k is the only thing that causes variance among the ensemble members, the larger the k the smaller the variance. When $k = D$, all ensemble members are the same, and the ensemble is equivalent to a single tree. This is the reason why on these datasets performance decreases significantly with increasing k. Additionally, on the *mediamill* and *scene* datasets there is very little difference in performance for different values of k (if we do not consider the lowest numbers of trees). However, there are still datasets where increasing the value of k to around 0.5 significantly improves the performance (e.g., *yeast* and *birds* datasets). Because random forests perform bootstrapping, which provides variance among ensemble members, this is much less of an issue. Selecting k between $0.1D$ and $0.3D$ would fit well with these datasets.

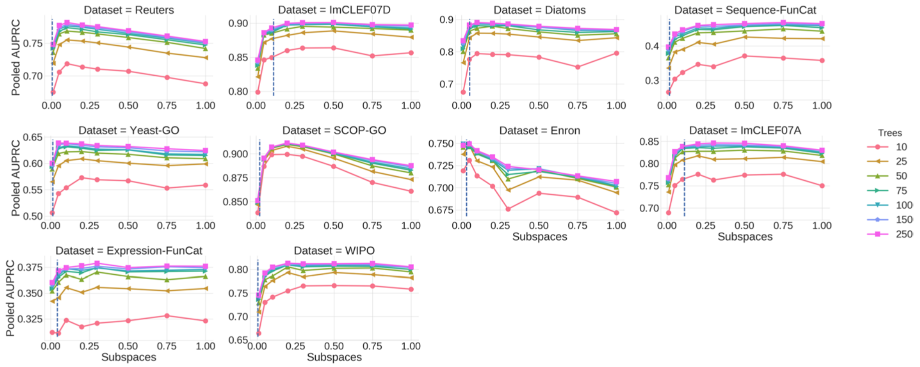


Fig. 6 Results for random forests of PCTs on HMC datasets for different numbers of attributes considered in each node (subspaces as a fraction of D) and different numbers of base predictive models (i.e., no. of trees: 10, 25, 50, 75, 100, 150, 250). The vertical dashed line is at the value for the square root from the descriptive features

The results for HMC are presented in Fig. 5 (Extra-PCTs) and in Fig. 6 (random forests). Note that here, in contrast to the previous two tasks higher value means better performance. As for the other two tasks, increasing the number of trees improves the performance for both Extra-PCTs and random forests, and the improvement of using more than 50 trees is rarely noticeable. For Extra-PCTs increasing the value of k initially quickly improves the performance, but it can later degrade it slightly. There are three exceptions to this. The first is the *enron* dataset, where the performance is constantly decreasing. Here the attributes are again all binary. The other exceptions are *Reuters* and *WIPO* datasets, where the performance significantly drops for $k = D$. The attributes in these datasets are very sparse. Because the vast majority of examples have value zero for any given attribute, different (randomly selected) split thresholds only cause a very small change in the partitions resulting from the splits. So, while the thresholds in different ensemble members are different, the tests selected and data partitions produced are mainly equal. Random forests also produce similar results, although they tend to reach their peak performance on lower values of k. Because of bootstrapping they are again resistant to extreme performance drops at $k = D$ on datasets with binary and/or sparse attributes.

Based on these results, the recommended values for these two parameters can be suggested. As noted above, having more trees in the ensemble improves the performance (at the cost of time complexity), but the gains diminish as the size of the ensembles increases. Based on the graphs from Figs. 1, 2, 3, 4, 5 and 6, we select 50 trees for both Extra-PCTs and random forests of PCTs as an optimal setting, for all the tasks considered. This is consistent with the findings of Kocev (2011), where no statistically significant improvement was found for adding more than 50 trees to an ensemble.

Selecting the value for k is less straightforward because peak performance is reached at very different values, depending on the method and dataset. For every task we select the value of k that produces the best average rank among the datasets for that task, separately for Extra-PCTs and random forests. To do so, we only compare different values of k with 50 trees in the ensemble. For Extra-PCTs we also treat datasets with only binary and/or sparse datasets separately, i.e. we select k for these datasets and other datasets separately. We also add ensembles with $k = \lceil \sqrt{D} \rceil$ (because they are often used in the literature) as well as bagging ensembles of PCTs (random forests with $k = D$),

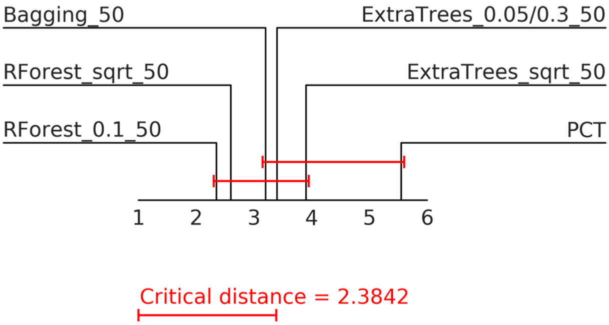


Fig. 7 Average ranking diagram comparing Extra-PCTs to single PCTs and other ensemble methods for the MTR task. Extra-PCTs with recommended k achieve the best performance on 17 of the 21 benchmark datasets

to see how well these selections of k perform. In the end, we compared 6 algorithms in three different tasks (using 21, 10 and 10 datasets for the tasks of MTR, MLC and HMC, respectively), using the Friedman test and the Nemenyi post-hoc test.

In short, for the MTR task, the average ranks of methods with different values of k are shown in Fig. 11 in the appendix. The selected values were thus $0.5D$ for random forests and $0.75D$ for Extra-PCTs (no binary/sparse datasets). Next, Figs. 12 and 14 in the appendix show the average ranks for different values of k for the MLC task. We select $0.1D$ for random forests, $0.05D$ for Extra-PCTs on sparse/binary datasets and 0.3 for Extra-PCTs otherwise. Finally, Figs. 13 and 14 in the appendix show the average ranks for different values of k for the HMC task. The best performing are $0.2D$ for random forests. For Extra-PCTs, $0.5D$ works best for datasets with sparse/binary attributes, and $0.75D$ otherwise. As we can see, the best results value of k depend on the task and algorithm, but, generally, $k = 0.5D$ provide good results for most of the combinations. A noticeable exception is MTR, where Extra-PCTs benefit from the usage of all the features. This is probably due to a combination of the effect of the finer (w.r.t. to other tasks) granularity of the evaluation measure used (i.e. RRMSE) and the additional (w.r.t. random forest) dimensions of randomization.

5.2 Extra-PCTs ensembles vs single PCTs and other ensemble methods

In the previous section, we analyzed the influence of the number of trees and the number of attributes considered at each node on the predictive performance of Extra-PCTs and random forests of PCTs. In this section, we perform a statistical analysis of their performance and compare the performance of the ensemble methods (single PCTs are used as the baseline). We consider three ensemble methods: Extra-PCTs, random forest and bagging (remember that random forests with $k = D$ is equivalent to bagging). The parameters for the methods were determined above.

The results of the statistical analysis for the MTR task are presented in Fig. 7. Extra-PCTs with recommended k have the best average rank followed by random forests with recommended k, but the difference is not statistically significant. However, random forests seem to perform quite similarly to bagging, which is not the case of Extra-PCTs. In addition, popular choices $k = \sqrt{D}$ and bagging have significantly worse performance than the best Extra-PCTs. As expected, a single PCT is statistically significantly worse than ensemble methods.

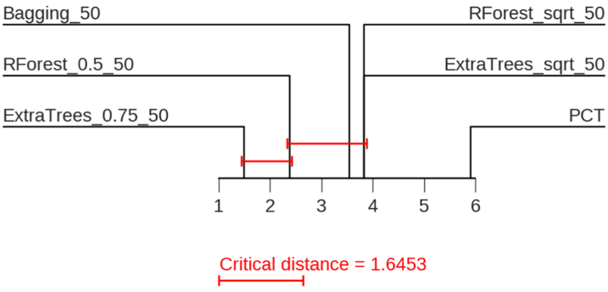


Fig. 8 Average ranking diagram comparing Extra-PCTs to single PCTs and other ensemble methods for the MLC task. Extra-PCTs with recommended k achieve the best performance on 4 of the 10 benchmark datasets

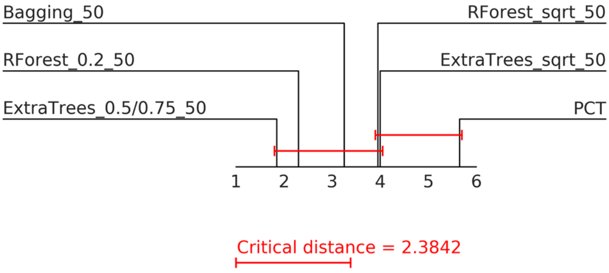


Fig. 9 Average ranking diagram comparing Extra-PCTs to single PCTs and other ensemble methods for the HMC task. Extra-PCTs with recommended k achieve the best performance on 6 of the 10 benchmark datasets

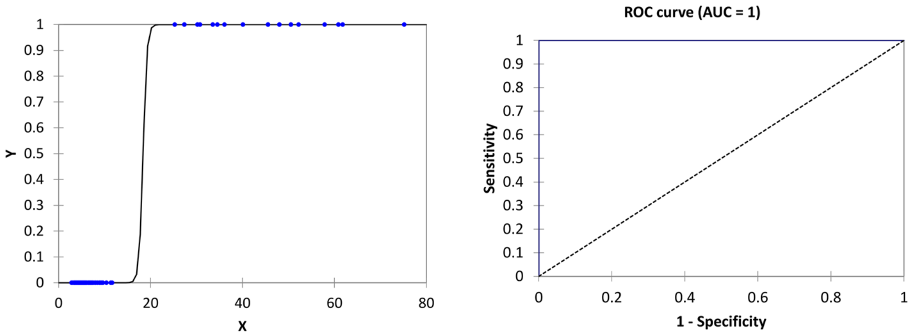
Figure 8 presents the results of the statistical analysis for the MLC task. Random forests appear to have a slight edge over other methods, however the difference to other ensemble methods was not found to be statistically significant.

The results of the statistical comparisons for the HMC task are shown in Fig. 9. Similar to the MLC task, there are no statistically significant differences among the ensemble methods, but Extra-PCTs and random forests, with the suggested values for k, have noticeably better average ranks than the others. Along with bagging, they are also statistically significantly better than single PCTs. Selecting $k = \sqrt{D}$ again does not seem optimal.

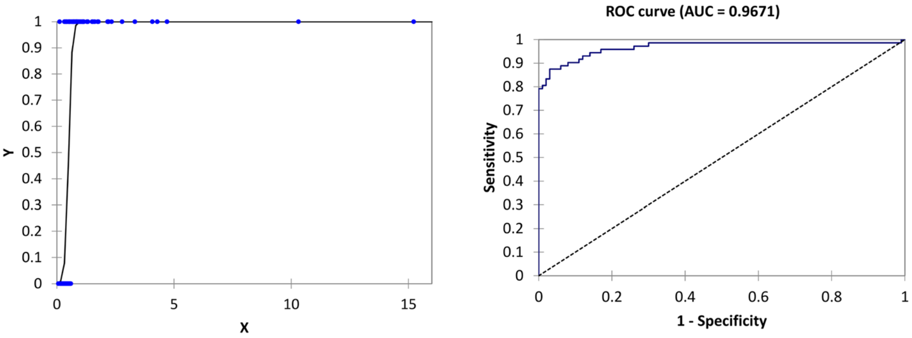
An additional finding is that the standard adopted in the literature for using the *sqrt* function for the feature subset size for random forests yields suboptimal performance when used in the context of structured output prediction. Similarly, the recommended values for Extra-PCTs for the single-target regression ($k = D$) and classification ($k = sqrt$) do not apply for the structured output prediction task.

5.3 Feature ranking for SOP

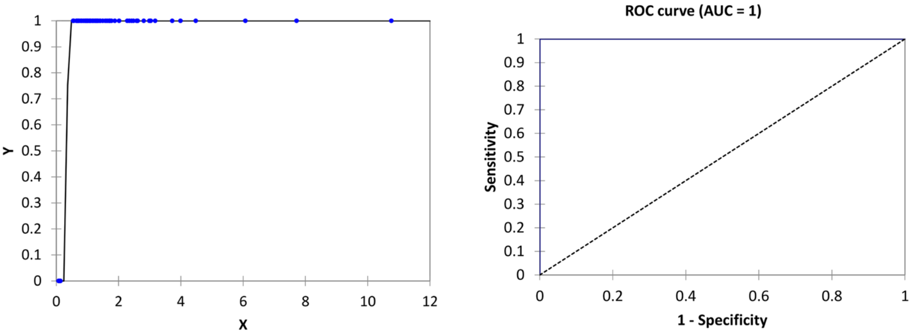
We illustrate the potential of Extra-PCTs to be used for performing feature ranking for the three tasks considered here. For each task, we select a dataset and check whether Extra-PCTs combined with GENIE3 scoring can properly delineate the real features from the 100 random features included in the dataset. We perform an analysis of the rankings using logistic regression approximation function $y = f(x)$, where x represents the score returned by the proposed solution for a given feature, and y indicates if the feature is a real feature or a random feature ($y = 0$ if the feature is a real feature and $y = 1$ if the feature is a random feature). The performance is then evaluated using ROC curves. Obtaining a value of 1 for the area under the ROC curve (AUROC) indicates that the algorithm is able to perfectly



(a) Water quality - log. regression **(b)** Water quality - ROC curve



(c) Emotions - log. regression **(d)** Emotions - ROC curve



(e) ImageCLEF07A - log. regression **(f)** ImageCLEF07A - ROC curve

Fig. 10 Feature ranking illustrative results. *Images on the left-hand side:* Points represent features. On the X axis we report the score returned by the proposed solution for the specific feature. On the Y axis we report 0 if the feature is a random feature and 1 if the feature is a real feature. The sigmoid curve is determined according to a logistic regression of Y with respect to X. The function shows a clear separation between random and real features. *Images on the right-hand side:* The ROC curve of the logistic regression approximation function $y = f(x)$, where x represents the score returned by the proposed solution for a given feature, and y indicates if the feature is a real feature or a random feature ($y = 0$ if the feature is a real feature and $y = 1$ if the feature is a random feature)

separate real features from random features. The value of 0.5 for AUROC indicates that the algorithm randomly separates real features from random features.

The obtained results of the feature ranking evaluation are shown in Fig. 10. The feature rankings correctly place the real features at the top of the ranking, ahead of the random features. For the datasets *Waterquality* (MTR) and *ImageCLEF*07*A* (HMC) the separation is perfect (AUC=1), while for the *Emotions* dataset the AUROC value is 0.9671. These results indicate that the proposed Extra-PCTs can be used for performing feature ranking for structured output prediction.

6 Conclusions

In this work, we address the task of learning Extra-PCTs ensembles of predictive models for structured output prediction (SOP). We investigate three SOP tasks: multi-target regression (the output is a tuple/vector of continuous variables), multi-label classification (the output is a tuple/vector of binary variables) and hierarchical multi-label classification (the output is a tuple/vector of binary variables organized into a hierarchy).

Ensembles have proved to be highly effective methods for improving the predictive performance of their constituent models, especially for classification and regression tree models. In particular, we consider the Extra-PCTs ensembles as predictive models. Extra-PCTs ensembles are a well established method for predictive modelling, that has been successfully applied to computer vision and gene network inference. As base predictive models, we propose using predictive clustering trees (PCTs). These can be considered as a generalization of decision trees for predicting structured outputs, including multiple continuous variables (MTR), multiple binary variables (MLC) and a hierarchy of multiple binary variables (HMC).

We perform a comprehensive experimental evaluation on 41 benchmark datasets: 21 for MTR, 10 for MLC and 10 for HMC. The selection of the datasets covers a wide range of application domains including ecology, business and life sciences. We compare the performance of three ensemble learning methods: Extra-PCTs, random forests and bagging of PCTs. Moreover, we compare the ensemble performance with a single base predictive model (i.e., a single PCT). The performance is measured with a variety of evaluation measures used for the specific tasks.

We summarize the results of the evaluation by answering the research questions that guided the experimental design:

1. *What is the number of base predictive models in the ensemble to obtain good predictive performance? Is this number stable across the tasks?*

   Including more base predictive models in the ensemble improves the predictive performance, but the improvement diminishes as ensemble sizes increase. Considering ensembles with 50 base predictive models is a good compromise between predictive power and computational efficiency. This is valid for the three tasks considered here and is in line with existing literature for non-SOP tasks.
2. *What is the optimal number of splits to be considered at each node in the tree construction for each of the SOP tasks?*

   Extra-PCTs are sensitive to the number of splits considered (k), and it proved important to treat datasets with only binary and/or sparse attributes separately. For the MTR task (no binary/sparse datasets), the recommended values are $0.75D$ for Extra-PCTs and $0.5D$ for random forests. For the MLC task, the recommended value for random forests is $0.1D$, whereas for EXTRA-PCTs it is $0.05D$ for binary/sparse datasets and $0.3D$ otherwise. Finally, for the HMC task, the recommended value for random forests is $0.2D$, and for EXTRA-PCTs it is $0.5D$ for binary/sparse datasets and $0.75D$ otherwise. Additionally, the results show that the literature recommended values for the feature subset size for random forests and EXTRA-PCTs yield suboptimal performance when used in the context of structured output prediction.

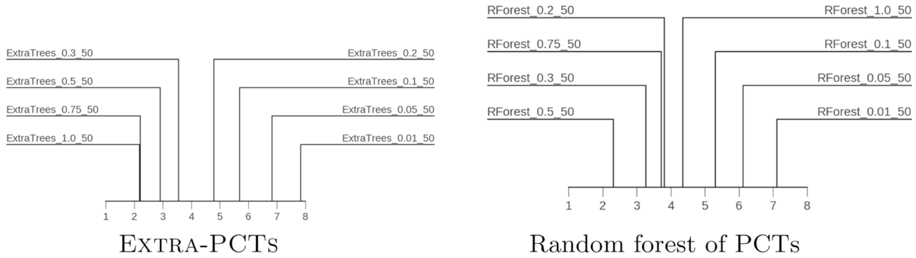
3. *Do* EXTRA-PCTs *ensembles yield better predictive performance than a single PCT?*
   The ensemble methods statistically significantly outperform single PCTs. For the MTR task, all of the ensembles are significantly better than a single PCT. For the MLC task, all of the ensembles are better than a single tree, while the difference is statistically significant only for random forests. For the HMC task, all of the ensembles are better than a single tree, while the difference is statistically significant for bagging and EXTRA-PCTs and random forests with recommended k values.
4. *How does the predictive performance of* EXTRA-PCTs *compare to the predictive performance of standard tree-ensemble methods, such as bagging and random forest of PCTs?*
   For the MTR and HMC tasks, EXTRA-PCTs are the best performing method. The differences are sometimes statistically significant for the MTR task, where EXTRA-PCTs with recommended k are significantly better than bagging, random forests with $k = \sqrt{D}$ and EXTRA-PCTs with $k = \sqrt{D}$. The best performing method for the MLC task is random forest with $0.1D$, but the differences among ensemble methods were not statistically significant.
5. *Can* EXTRA-PCTs *be used to obtain a feature ranking for domains with structured outputs?*
   The proof-of-concept experiments illustrate that EXTRA-PCTs can be used to obtain feature rankings across the SOP tasks considered here. The results show that the real features are placed at the top of the ranking, while the added random feature is at the bottom of the ranking.

We plan to extend the work along the following major dimensions. First, EXTRA-PCTs can be extended to other types of structured outputs (such as time series or tuples of mixed primitive data types, both continuous and discrete). Next, other (more complex) distance measures on structured types can be used, thus extending the applicability of the method to new domains. Furthermore, we will extend EXTRA-PCTs towards semi-supervised learning. Finally, we plan to investigate in depth the potential of EXTRA-PCTs for performing feature ranking in the context of the SOP tasks considered here.

Acknowledgements We acknowledge the financial support of the European Commission through the grants ICT-2013-612944 MAESTRA and H2020-688797 TOREADOR, as well as the Slovenian Research Agency (via the grant J2-9230 and a young researcher Grant to TS). The computational experiments were executed on a computing infrastructure from the Slovenian Grid (SLING) initiative. We finally thank Lynn Rudd for her help in reading the manuscript.

A Appendix

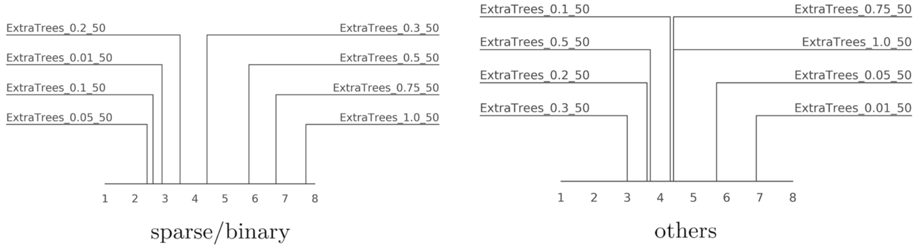
See Figs. 11, 12, 13 and 14.



Extra-PCTs

Random forest of PCTs

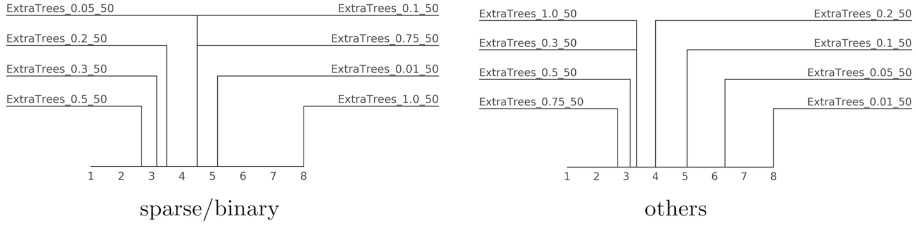
Fig. 11 Average ranking diagrams of Extra-PCTs (left) and random forests of PCTs (right) with a different number of attributes considered in each node, for the task of MTR using RRMSE as the performance measure



sparse/binary

others

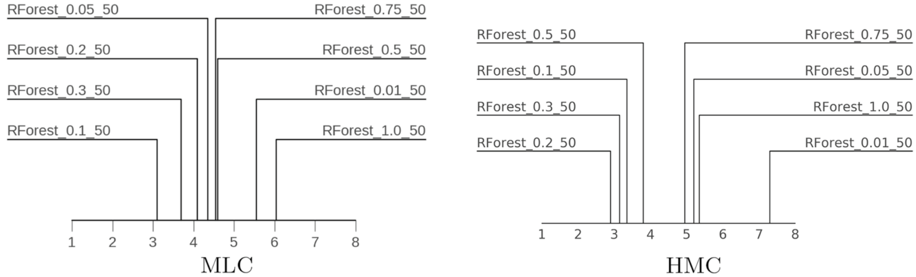
Fig. 12 Average ranking diagrams of Extra-PCTs on datasets with only binary/sparse attributes (left) and others (right), with a different number of attributes considered in each node, for the task of MLC using Ranking Loss as the performance measure



sparse/binary

others

Fig. 13 Average ranking diagrams of Extra-PCTs on datasets with only binary/sparse attributes (left) and others (right), with a different number of attributes considered in each node, for the task of HMC using *AUPRC* as the performance measure



MLC

HMC

Fig. 14 Average ranking diagrams of random forests of PCTs with a different number of attributes considered in each node, for the tasks of MLC (left) and HMC (right)

References

(2007) ISO/IEC 11404:2007–Information technology–General-Purpose Datatypes (GPD). http://www.iso.org/iso/catalogue_detail.htm?csnumber=39479

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

Appice, A., & Džeroski, S. (2007). Stepwise induction of multi-target model trees. In *Machine learning: ECML 2007, LNCS* (Vol. 4701, pp. 502–509).

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

Barutcuoglu, Z., Schapire, R. E., & Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, *22*(7), 830–836.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, *36*(1), 105–139.

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

Blockeel, H., Bruynooghe, M., Džeroski, S., Ramon, J., & Struyf, J. (2002). Hierarchical multi–classification. In *KDD-2002 Workshop Notes: MRDM 2002, Workshop on Multi-Relational Data Mining* (pp. 21–35).

Bogatinovski, J. (2019). A comprehensive study of multi-label classification methods. M.S. thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

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

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

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

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

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

Breskvar, M., Kocev, D., & Džeroski, S. (2018). Ensembles for multi-target regression with random output selections. *Machine Learning*, *107*(11), 1673–1709.

Brown, P. J., & Zidek, J. V. (1980). Adaptive multivariate ridge regression. *The Annals of Statistics*, *8*(1), 64–74.

Ceci, M., & Malerba, D. (2007). Classifying web documents in a hierarchy of categories: A comprehensive study. *Journal of Intelligent Information Systems*, *28*(1), 37–78.

Cerri, R., Pappa, G. L., Carvalho, A. C. P., & Freitas, A. A. (2015). An extensive evaluation of decision tree-based hierarchical multilabel classification methods and performance measures. *Computational Intelligence*, *31*(1), 1–46.

Cerri, R., Barros, R. C., de Carvalho, P. L. F., & Jin, A. C. Y. (2016). Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinformatics*, *17*(1), 373–374.

Clare, A. (2003). Machine learning and data mining for yeast functional genomics. Ph.D. thesis, University of Wales Aberystwyth, Aberystwyth, Wales, UK.

Crammer, K., & Singer, Y. (2003). A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, *3*, 1025–1058.

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

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.

Evgeniou, T., Micchelli, C. A., & Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, *6*, 615–637.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, *11*, 86–92.

Fürnkranz, J. (2002). Round robin classification. *Journal of Machine Learning Research*, *2*, 721–747.

Gärtner, T., & Vembu, S. (2009). On structured output training: Hard cases and an efficient alternative. *Machine Learning*, *76*, 227–242.

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

Geurts, P., Wehenkel, L., & D'Alché-Buc, F. (2006b). Kernelizing the output of tree–based methods. In *ICML '06: Proceedings of the 23rd international conference on machine learning* (pp. 345–352), ACM.

Gjorgjioski, V., Kocev, D., & Džeroski, S. (2011). Comparison of distances for multi-label classification with pcts. In *Proceedings of the 14th international multiconference-information society IS 2011* (pp. 121–124), IJS, Ljubljana.

Ho, C., Ye, Y., Jiang, C. R., Lee, W. T., & Huang, H. (2018). Hierlpr: Decision making in hierarchical multi-label classification with local precision rates.

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

Kocev, D. (2011). Ensembles for predicting structured outputs. Ph.D. thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

Kocev, D., & Ceci, M. (2015). Ensembles of extremely randomized trees for multi-target regression. In *Discovery science: 18th international conference (DS 2015), LNCS* (Vol. 9356, pp. 86–100).

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

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.

Levatić, J., Kocev, D., Ceci, M., & Džeroski, S. (2018). Semi-supervised trees for multi-target regression. *Information Sciences*, *450*, 109–127.

Liu, G., Lin, Z., & Yu, Y. (2009). Multi-output regression on the output manifold. *Pattern Recognition*, *42*(11), 2737–2743.

Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, *45*(9), 3084–3104.

Maree, R., Geurts, P., Piater, J., & Wehenkel, L. (2005). Random subwindows for robust image classification. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, *1*, 34–40.

Mileski, V. (2017). Tree methods for hierarchical multi-target regression. M.S. thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia.

Nemenyi, P.B. (1963). Distribution-free multiple comparisons. Ph.D. thesis, Princeton University, Princeton, NY, USA.

Panov, P., Soldatova, L. N., & Džeroski, S. (2016). Generic ontology of datatypes. *Information Sciences*, *329*, 900–920.

Radivojac, P., et al. (2013). A large-scale evaluation of computational protein function prediction. *Nature Methods*, *10*, 221–227.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333–359.

Rousu, J., Saunders, C., Szedmak, S., & Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, *7*, 1601–1626.

Ruyssinck, J., Huynh-Thu, V. A., Geurts, P., Dhaene, T., Demeester, P., & Saeys, Y. (2014). NIMEFI: Gene regulatory network inference using multiple ensemble feature importance algorithms. *PLoS ONE*, *9*(3), 1–13.

Schapire, R. E., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, *39*, 135–168.

Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., & Džeroski, S. (2010). Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, *11*(2), 1–14.

Silla, C., & Freitas, A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, *22*(1–2), 31–72.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., & Vlahavas, I. (2016). Multi-target regression via input space expansion: Treating targets as inputs. *Machine Learning*, *5*, 1–44.

Stojanova, D., Ceci, M., Malerba, D., & Džeroski, S. (2013). Using PPI network autocorrelation in hierarchical multi-label classification trees for gene function prediction. *BMC Bioinformatics*, *14*, 285.

Struyf, J., & Džeroski, S. (2006). Constraint based induction of multi-objective regression trees. In *Proceedings of the 4th international workshop on knowledge discovery in inductive databases KDID (LNCS 3933)* (pp. 222–233), Springer.

Tian, W., Zhang, L. V., Taşan, M., Gibbons, F. D., King, O. D., Park, J., Wunderlich, Z., Cherry, J. M., & Roth, F. P. (2008). Combining guilt–by–association and guilt–by–profiling to predict Saccharomyces cerevisiae gene function. *Genome Biology* 9(S1):S7.

Tsoumakas, G., & Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the 18th European conference on machine learning* (pp. 406–417).

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2008). Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *Proceedings of the ECML/PKDD workshop on mining multidimensional data* (pp. 30–44).

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2010). Mining multi-label data. *Data mining and knowledge discovery handbook* (pp. 667–685). Berlin: Springer.

Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., & Vlahavas, I. (2014). Multi-target regression via random linear target combinations. In *Machine learning and knowledge discovery in databases: ECML-PKDD 2014* (Vol. 8726, pp. 225–240), LNCS.

Valentini, G., & Re, M. (2009). Weighted true path rule: a multilabel hierarchical algorithm for gene function prediction. In *Proceedings of the 1st international workshop on learning from multi-label data* (pp. 133–146).

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, *73*(2), 185–214.

Škunca, N., Bošnjak, M., Kriško, A., Panov, P., Džeroski, S., Šmuc, T., et al. (2013). Phyletic profiling with cliques of orthologs is enhanced by signatures of paralogy relationships. *PLOS Computational Biology*, *9*(1), 1–14.

Witten, I. H., & Frank, E. (2005). Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

Zhang, M. L., & Zhou, Z. H. (2007). Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, *40*(7), 2038–2048.

Affiliations

Dragi Kocev[1,2,3] · **Michelangelo Ceci**[1,2,4] · **Tomaž Stepišnik**[2,3]

Michelangelo Ceci
michelangelo.ceci@uniba.it

Tomaž Stepišnik
Tomaz.Stepisnik@ijs.si

[1] Department of Computer Science, University of Bari Aldo Moro, Bari, Italy

[2] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia

[3] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

[4] CINI - Consorzio Interuniversitario Nazionale per l'Informatica, Rome, Italy