Rule Ensembles for Multi-Target Regression

Timo Aho
Department of Software Systems
Tampere University of Technology
Korkeakoulunkatu 1, FI-33101 Tampere, Finland
Email: timo.aho@tut.fi

Bernard Ženko and Sašo Džeroski
Department of Knowledge Technologies
Jožef Stefan Institute
Jamova cesta 39, SI-1000 Ljubljana, Slovenia
Emails: {bernard.zenko, saso.dzeroski}@ijs.si

***Abstract*—Methods for learning decision rules are being successfully applied to many problem domains, especially where understanding and interpretation of the learned model is necessary. In many real life problems, we would like to predict multiple related (nominal or numeric) target attributes simultaneously. Methods for learning rules that predict multiple targets at once already exist, but are unfortunately based on the covering algorithm, which is not very well suited for regression problems. A better solution for regression problems may be a rule ensemble approach that transcribes an ensemble of decision trees into a large collection of rules. An optimization procedure is then used for selecting the best (and much smaller) subset of these rules, and to determine their weights.**

Using the rule ensembles approach we have developed a new system for learning rule ensembles for multi-target regression problems. The newly developed method was extensively evaluated and the results show that the accuracy of multi-target regression rule ensembles is better than the accuracy of multi-target regression trees, but somewhat worse than the accuracy of multi-target random forests. The rules are significantly more concise than random forests, and it is also possible to create very small rule sets that are still comparable in accuracy to single regression trees.

***Keywords*-Multi-Target Prediction, Rule Learning, Regression**

I. Introduction

The most commonly addressed problem in machine learning is that of how to predict the value of a single target attribute or class. There exist, however, many real life problems where we would like to predict multiple target attributes at once, instead of only a single one. A typical example from the environmental sciences would be the task of predicting species communities [1]. Here we are interested in the abundances of a set of different species living in the same environment. These species represent the target attributes, which might, but need not be related. If our only goal is to achieve high predictive accuracy, a collection of single-target models should be sufficient to solve the problem. However, if we are also interested in interpretability, the collection of single-target models is much more complex and harder to interpret than a single model that jointly predicts all target attributes [2]–[4]. An additional benefit of the multi-target models is that they are less likely to overfit the data and are frequently more accurate than the corresponding collections of single-target models [2], [3], [5], [6].

Besides decision trees, rule sets are one of the most expressive and human readable model representations, and are frequently used when an interpretable model is desired. The majority of rule learning methods, including existing methods for learning multi-target rules, are based on the sequential covering algorithm [7], originally designed for learning ordered rule lists for binary classification domains. Unfortunately, on both single- and multi-target regression problems the accuracy of rule sets that are learned with this approach is considerably worse than that of other regression methods, i.e., regression trees (cf. [8] for an empirical comparison). An alternative rule learning method that performs well also on (single-target) regression problems is the approach of *rule ensembles* as implemented in the RuleFit method [9]. The method starts with creating an ensemble of decision trees, which are considered as an initial collection of rules. An optimization procedure is then used with the purpose of finding an optimal weight for each of these rules. During the optimization, one tries to assign as many weights as possible to zero in order to learn small and interpretable models, while not compromising their accuracy. In this paper we adopt this approach for learning multi-target regression rule ensembles, and propose an algorithm that can learn unordered rule sets that are both accurate and interpretable.

The paper is organized as follows. In Section 2, we briefly present the related work on multi-target prediction, rule learning, and rule ensembles. The newly proposed algorithm for learning multi-target regression rule ensembles is presented in Section 3. In Section 4, we describe the experimental evaluation setting, and in Section 5 the experimental results. The last section concludes and gives some directions for further research.

II. Related Work

The algorithm presented in this paper is related to several existing approaches for multi-target prediction. The multi-target prediction task is defined as follows. We are given a set of learning examples E of the form $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$ is a vector of K descriptive attributes and $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$ is a vector of T target attributes.

Our task is to learn a model that, given a new unlabeled example $\boldsymbol{x}$, can predict the values of all target attributes $\boldsymbol{y}$ simultaneously. Several standard learning methods such as neural networks, decision trees, classification rules and random forests have already been extended towards multi-target prediction [3]–[6], [10].

Since our method learns regression rules, it is highly related to rule learning [11]. A method for learning multi-target rules already exists [4], [8]. It employs the standard covering approach and can learn ordered or unordered rule sets for classification or regression domains. Its accuracy on classification domains is comparable to other classification methods like (multi-target) decision trees. However, on regression domains the accuracy is significantly worse.

An alternative approach to rule learning are so-called rule ensembles [9], [12].[1] In this paper we adopt rule ensembles as implemented in the RuleFit method. This method starts by generating a set of decision trees in much the same way as ensembles are generated by methods like bagging [13] or random forests [14]. Because such large ensembles are hard or impossible to interpret, all the trees are transcribed into a collection of rules, and an optimization procedure is used to select a small subset of rules and to determine their weights. As a result, we get a relatively small set of weighted rules. The final prediction for a given example is obtained by taking a weighted vote of all the rules that apply (i.e., cover the example). The resulting model can thus be written as: $\hat{\boldsymbol{y}} = f(\boldsymbol{x}) = w_0 + \sum_{i=1}^{M} w_i r_i(\boldsymbol{x})$,where w_0 is the baseline prediction, and the sum is the correction value obtained from M rules. The rules r_i are functions, which have a value of 1 for all examples that they cover, and 0 otherwise. During the learning phase, all the weights w_i are optimized by a gradient directed optimization algorithm.

The method presented in this paper is a generalization of RuleFit towards multi-target regression problems, and we describe it in detail in the next section.

III. Learning Rule Ensembles

Our algorithm for learning rule ensembles for multi-target regression problems (which we call FIRE – Fitted rule ensembles) is based on the RuleFit method [9]; its outline is presented in Fig. 1. We start by generating a set of diverse regression trees, which we convert to rules. We then optimize the weights of rules with a gradient directed optimization algorithm. This optimization procedure depends on the gradient threshold parameter τ, and we repeat the optimization for different values of τ in order to find a set of weights with the smallest error. In the end we remove all the rules whose weights are zero.

The resulting rule ensemble is a vector function $\boldsymbol{f}$; given an unlabeled example $\boldsymbol{x}$ it predicts the values of all target attributes:

$$\hat{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{x}) = w_0\, \boldsymbol{avg} + \sum_{i=1}^{M} w_i \boldsymbol{r}_i(\boldsymbol{x}). \qquad (1)$$

The first part ($\boldsymbol{avg}$) is a constant vector with the averages over each of the targets. The sum is the contribution of M rules: each rule $\boldsymbol{r}_i$ is a vector function that gives a constant prediction, if it covers the example $\boldsymbol{x}$, or returns zero otherwise. All the weights w are being determined during the optimization phase and our goal is to have as many weights equal to zero as possible.

Example. Let the problem domain have eight descriptive attributes $\boldsymbol{x} = (x_1, \ldots, x_8)$ and three target attributes $\boldsymbol{y} = (y_1, y_2, y_3)$. A hypothetic rule ensemble, comprising a constant vector and two rules, that predicts all the target values of this domain simultaneously could be:

$$\begin{aligned}\hat{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{x}) &= 0.95\,(16.2, 6.0, 21.1)\\ &\quad + 0.3\ [\ \textsf{IF}\ (x_8 > 3)\&(x_6 > 7.2)\ \textsf{THEN}\ (2, 6, 3.7)]\\ &\quad + 0.2\ [\ \textsf{IF}\ (x_3 \leq 12.1)\ \textsf{THEN}\ (6.3, 50, -14.3)]\\ &= (15.4,\ 5.7,\ 20.0)\\ &\quad + [\ \textsf{IF}\ (x_8 > 3)\&(x_6 > 7.2)\ \textsf{THEN}\ (0.6, 1.8, 1.1)]\\ &\quad + [\ \textsf{IF}\ (x_3 \leq 12.1)\ \textsf{THEN}\ (1.3, 10, -2.9)]\end{aligned}$$

So far, we have only briefly mentioned two important aspects of our algorithm, i.e., the generation of the initial collection of trees and rules and the weight optimization procedure. We will now describe them in more detail. The basic decision tree learning method that is used within the GenerateSetOfTrees procedure of algorithm FIRE (Fig. 1) is the predictive clustering tree learning method [10] that can learn multi-target regression trees. A set of diverse trees is generated with the multi-target implementation of the random forest ensemble method [6], but in order to increase the tree variance, the maximum tree depth is limited to a random value chosen according to a probability distribution

Input: learning examples E
Output: rules R with their weights W

1: $T \leftarrow \text{GenerateSetOfTrees}(E)$
2: $R \leftarrow \text{ConvertTreesToRules}(T)$
3: $ERR_{\min} \leftarrow \infty$
4: **for** $\tau = 0.0$ **to** 1.0 **with** *step* **do**
5: $\quad (W_\tau, ERR_\tau) \leftarrow \text{OptimizeWeights}(R, E, \tau)$
6: $\quad$ **if** $ERR_\tau < ERR_{\min}$ **then**
7: $\quad\quad (W_{\text{opt}}, ERR_{\min}) \leftarrow (W_\tau, ERR_\tau)$
8: $\quad$ **end if**
9: **end for**
10: $(R, W) \leftarrow \text{RemoveZeroWeightedRules}(R, W_{\text{opt}})$
11: **return** (R, W)

Figure 1. FIRE — Algorithm for learning rule ensembles for multi-target regression.

[1] Strictly speaking, any set of (unordered) rules can be called a rule ensemble, however, in this paper, rule ensemble is understood as a set of unordered rules whose predictions are combined via weighted voting.

proposed by [9], and used in the original RuleFit method. The average depth of trees is specified as a parameter to the algorithm. It should be emphasized here that this parameter only limits the average depth of generated trees, trees with larger depth can still be generated.

All regression trees generated are transcribed into rules within the ConvertTreesToRules procedure. Each leaf of each tree is converted to a rule and its predictions r_t are normalized in the following way. The predicted value r'_t of each target attribute t is changed to $r_t = r'_t/r'_m$, where r'_m is the maximum absolute value of any predicted target attribute value by this rule: $m = \arg\max_i |r'_i|$. Thus for the normalized rules it holds that $|r_t| \leq 1$ and $r_m = 1$. Such a normalization roughly equalizes the rules before the optimization phase.

The last part of the algorithm that deserves a detailed description is the optimization within the OptimizeWeights procedure of algorithm FIRE (Fig. 1) that determines optimal weights of the rules. For this purpose, the RuleFit method uses a gradient directed optimization method [15] and a squared loss function L_t:

$$L_t\left(f_t(\boldsymbol{x}), y_t\right) = \left(f_t(\boldsymbol{x}) - y_t\right)^2/2, \tag{2}$$

which is applicable to a single target attribute only; here $f_t(\boldsymbol{x})$ is the predicted value and y_t is the true value. If we want to use the above optimization algorithm for multi-target problems, we have to define a loss function that is convex. A simple solution is to take the above squared loss function for each of the T target attributes and aggregate them by taking their average:

$$L\left(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y}\right) = \frac{1}{T}\sum_{t=1}^{T} L_t\left(f_t(\boldsymbol{x}), y_t\right). \tag{3}$$

Such an aggregated loss function is convex and enables efficient computation of gradients.

To equalize the contributions of different targets to the aggregated loss function, we normalize the target values by shifting their mean to zero and dividing them by $2\sigma_t$, where σ_t is the standard deviation of a given target attribute. Assuming a normal distribution, this should put 95% of all values within the $[-1, 1]$ interval.

In this kind of optimization, we usually add to the loss function a regularization part of the form $\sum_{i=1}^{M} |w_i|^\alpha$ to keep the weights smaller or zero and add stability to the optimization procedure. Popular values for α include $\alpha = 2$ (L2 or ridge) and $\alpha = 1$ (L1 or lasso). For gradient directed optimization, a very similar effect to this kind of regularization can also be achieved in a different and more efficient way [15]. Instead of adding the regularization term to the loss function, we can explicitly control the number of weights that are changed during every optimization iteration in the following way. Let M be the number of weights that we are optimizing with a gradient method. Instead of allowing changes to all the weights simultaneously, we only allow changes to the weights w_j whose gradients g_j have a value above a threshold

$$|g_j| \geq \tau \cdot \max_{0 \leq k \leq M} |g_k|. \tag{4}$$

With $\tau = 0$, we are changing all the weights during every iteration, resulting in a behavior similar to ridge regularization. On the other side, if $\tau = 1$, only one gradient during every iteration is modified and the behavior is similar to lasso regularization. In our case, lasso regularization seems better, because it has been shown to lead to many weights being set to zero [16], which means simpler and more interpretable models with fewer rules. However, in practice it is hard to predict which τ value will result in the most accurate model. We overcome this problem by trying a set of different τ values (Fig. 1, line 5) and estimating their accuracy on a separate validation set (the validation set is the same for all τ). In the end, the model with the smallest validation error is selected.

Input: rules R, learning examples E and gradient treshold parameter τ
Output: weights W and an error estimate ERR

1: $W_0 = \{0, 0, \ldots, 0\}$
2: $(E_t, E_v) \leftarrow \mathrm{SplitSet}(E)$ {Training and validation}
3: **for** $i = 0$ to Maximum number of iterations **do**
4: **do** every 100 iterations
5: **if** $\mathrm{Error}(E_v, R, W_i)$ increased
6: $W_i \leftarrow \mathrm{WeightsWithSmallestError}(E_v, R)$
7: ReduceStepSize()
8: **end if**
9: **end do**
10: $G \leftarrow \mathrm{ComputeGradients}(E_t, R, W_i)$
11: **if** Limit of allowed nonzero weights is reached **then**
12: $G \leftarrow \{g_k \in G | w_k \in W_i : w_k \neq 0\}$
13: **end if**
14: $G_{\max} \leftarrow \{g_j \in G |\ |g_j| \geq \tau \max_k |g_k|\}$
15: $W_{i+1} \leftarrow \mathrm{ChangeWeightsWithStep}(G_{\max}, W_i)$
16: **end for**
17: $W \leftarrow \mathrm{WeightsWithSmallestError}(E_v, R)$
18: $ERR \leftarrow \mathrm{Error}(E_v, R, W)$
19: **return** (W, ERR)

Figure 2. OptimizeWeights — Algorithm for gradient directed optimization.

The complete optimization algorithm is presented in Fig. 2. We start with all the weights set to zero, and splitting the example set E into a training set E_t and a validation set E_v. The main idea is to repeat computing gradients g_k for each of the weights (line 8) and then changing the selected weights w_j into the most promising direction for a predefined step size (lines 12–13).

In addition to this base idea, there are some important

details. First, on every 100-th iteration we check if we are overfitting (line 4), i.e., if the validation error starts to increase. In case the validation error increased, we do not stop the optimization (as the algorithm by [15] does), but return to the iteration with the lowest validation error and continue with a smaller gradient step size, e.g., the step size can be multiplied by 0.1 (lines 5–6). Second, we can define a number of nonzero weights in advance (lines 9–11), which makes a suitable parameter for setting the accuracy vs. simplicity trade-off. An extensive experimental evaluation of the algorithm's performance is presented in the next section.

IV. Experimental Setup

In the experimental evaluation we investigate three issues. First, we compare the accuracy and model size of FIRE to regression trees [10] and random forests [14] on single-target regression data sets in order to show that our implementation is also applicable to standard regression problems. Second, we compare FIRE to the same methods on multi-target regression data sets; this is the key part of the evaluation. As described in Section III, our algorithm has a parameter that can be used for limiting the total number of nonzero weights, i.e., the number of rules. All the above experiments were done with two values of this parameter: with an arbitrary limit of 20 rules, and without any limitation on the size. If we limit the maximum size of the models we of course get less accurate models. An investigation of this issue is the focus of the third part of the experimental evaluation.

The regression trees [10] and random forests [6] used in our experiments, as well as our FIRE algorithm, are implementedin the CLUS predictive clustering framework [17].[2] All the parameters for regression trees and random forests were set to their default values. Random forests used 100 trees. Where possible, the parameters of FIRE were set to the values used in [9]. When generating the initial set of trees (GenerateSetOfTrees procedure) we used 100 random trees with an average depth of 3. The optimization procedure (Fig. 1, line 5) was run with gradient threshold parameter τ values ranging from 0 to 1 in 0.1 increments. In the OptimizeWeights procedure (Fig. 2), the initial learning set E was split into $2/3$ for training (E_t) and $1/3$ for validation (E_v). The maximum number of optimization iterations was 10,000. The threshold for detecting error increase (line 5) was 1.1, the starting step size for changing the weights was 10, but it was automatically multiplied by the factor 0.1 (i.e., reduced) if overfitting occured (Fig. 2, line 6).

The data sets used in the experiments, together with their properties and references, are presented in Table I. Fifteen single-target regression data sets are taken from standard ML data repositories. Publicly available multi-target data sets, however, are scarce; in addition to one public one, we collected ten previously analyzed data sets for which we provide references. Due to time limitations we have restricted the number of examples in each data set; for data sets with more than 2,000 examples, we used random subsample of size 2,000 and ignored the rest.

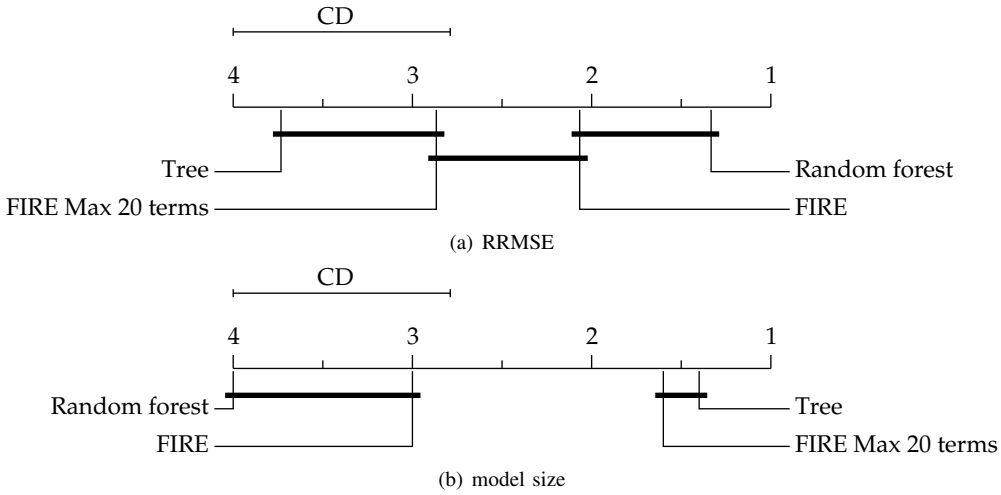
[2]Available at `http://www.cs.kuleuven.be/~dtai/clus` under GNU General Public License.

Table I
DATA SETS USED IN THE EXPERIMENTAL EVALUATION. SINGLE-TARGET DATA SETS ARE TAKEN FROM STANDARD ML REPOSITORIES BY UCI, DELVE AND LUÍS TORGO. A BULLET DENOTES THAT A RANDOM SUBSAMPLE OF A DATA SET WAS USED.

DATA SET	# EXS	# DES ATTS	# TAR ATTS	SOURCE
ABALONE •	2,000	8	1	UCI
AILERONS	1,533	40	1	TORGO
AUTO-MPG	398	7	1	UCI
CENSUS •	2,000	8	1	DELVE
CLOUD	108	8	1	UCI
CPU ACTIVITY •	2,000	12	1	DELVE
DELTA-AILERONS •	2,000	5	1	TORGO
META-DATA	528	21	1	UCI
PBC	418	18	1	UCI
QUAKE •	2,000	3	1	UCI
ROBOT ARM•	2,000	8	1	DELVE
SENSORY	576	11	1	UCI
SERVO	167	4	1	UCI
STRIKE	625	6	1	UCI
VETERAN	137	7	1	UCI
COLLEMBOLAN	393	48	3	[18]
EDM	154	16	2	[19]
FOREST KRAS •	2,000	160	11	[20]
FOREST SLIVNICA •	2,000	149	2	[21]
META LEARNING	42	56	10	[22]
MICROARTHROPODS	1,944	142	3	[1]
SIGMEA REAL	817	4	2	[23]
SIGMEA SIMUL. •	2,000	10	3	[24]
SOLAR FLARE	323	10	3	UCI
VEGETATION •	2,000	64	11	[25]
WATER QUALITY	1,060	16	14	[26]

The accuracy of the learned regression models is estimated for each target attribute by the relative root mean squared error (RRMSE). The size of regression trees and random forests is measured as the number of tree leaves in all the trees. The size of FIRE models is measured as the number of rules. All the above measures are estimated with 10-fold cross-validation, where the folds for each data set are the same for all the algorithms.

To test whether any of the observed differences between the algorithms are significant, we followed the methodology suggested by [27]: first we use the Friedman test to check if there are any statistically significant differences between the compared algorithms. If the answer is positive, we additionally use the Nemenyi post-hoc test to figure out what these differences are, and we present them on the average ranks diagrams. These diagrams show all the compared algorithms in the order of their average ranks; the best are on the right and the worst are on the left side of the diagram. The algorithms that differ by less than a critical distance for a *p-value* $= 0.05$ are connected with a horizontal bar, and



(a) RRMSE

(b) model size

Figure 3. Average ranks diagrams on *single-target* data for *RRMSE* (a) and *model size* (b). Better algorithms are on the right-hand side, the ones that do not differ significantly by the Nemenyi test (*p-value* $= 0.05$) are connected with a horizontal bar. CD is the critical distance.

are not significantly different. We perform such significance testing for RRMSE and for model size.

However, when testing the differences in RRMSE for multi-target data we have two possibilities. We can either treat each of the target attributes of all data sets as an independent measurement, or we can compute the average over all targets within each data set and consider such averages as independent measurements. The argument against the first option is that target attributes within one data set are probably not independent and as a result our test will show more significant differences than there actually are. The argument against the second option is that when computing averages across all target attributes within a data set, we are actually summing apples and oranges, and the resulting average is probably not a valid quantity. In the absence of a better solution, we present tests of RRMSE differences for both options. The results of the experimental evaluation are presented in the next section.

V. Results

As already mentioned in the previous section, we performed three groups of experiments. We evaluated our implementation first on single-target and then on multi-target regression data sets. The latter is the most important part, since it shows whether our generalization of rule ensembles towards multi-target regression is successful. Finally, we investigated the influence of rule ensemble size on accuracy.

A. Single-Target Regression

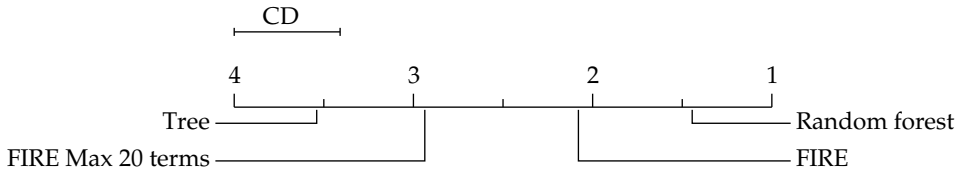
On single-target data sets we have compared regression trees, random forests and two versions of FIRE: one without any limitation on the model size, and one with the maximum number of rules set to 20. The average RRMSEs over all 15 data sets are 0.74, 0.66, 0.67, and 0.71, respectively. The corresponding average models sizes are 26.9, 36,475, 409, and 19.5. We have omitted the detailed results due to space limitations. The Friedman test shows that the RRMSEs are statistically different with a *p-value* $= 2.4 \cdot 10^{-6}$ and model sizes with a *p-value* $= 7.6 \cdot 10^{-9}$. The average ranks for all four algorithms together with the results of the Nemenyi test are given in Fig. 3, separately for RRMSE and model size. The better algorithms are the ones with higher ranks (with 1 being the highest rank) and are placed on the right-hand side of the diagram. Algorithms whose ranks differ by less than a critical distance (CD) are not significantly different with a *p-value* $= 0.05$.

From the RRMSE diagram (Fig. 3a) we can see that random forests are the most accurate method, followed by the unlimited and the limited versions of FIRE and regression trees. However, for the adjoining algorithms, the difference is not statistically significant. By limiting the number of FIRE rules we therefore still get reasonably accurate models. The diagram for model size (Fig. 3b) shows that regression trees and the size limited FIRE both generate significantly smaller models than the unlimited FIRE and random forests. While the unlimited version of FIRE generates smaller models than random forests, the difference is below the significance threshold. We argue that these results show that our implementation of rule ensembles performs well also on single-target regression problems.

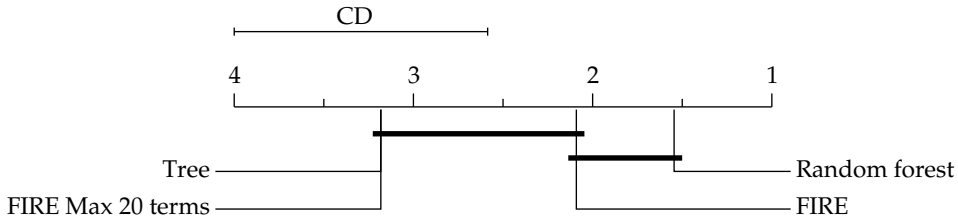
B. Multi-Target Regression

The detailed results of the algorithm comparison on multi-target regression data are presented in Table II. Random forests generate the most accurate models on 8, the unlimited version of FIRE on 4 and regression trees on 3 data sets (on three data sets multiple methods perform equally well). However, the differences in model sizes are very large.

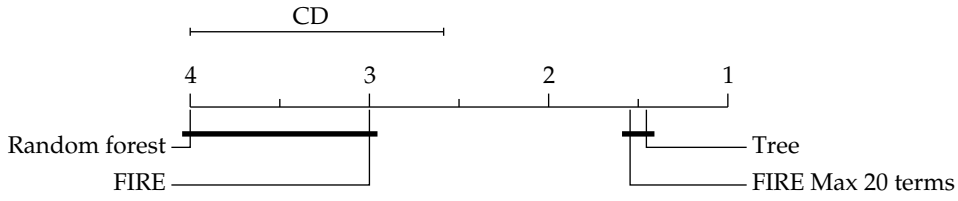
The Friedman test shows that the RRMSE values of algorithms are significantly different with a *p-value* $< 2.2 \cdot 10^{-16}$,



(a) RRMSE, per target evaluation



(b) RRMSE, per data set target average evaluation



(c) size

Figure 4. Average ranks diagrams on *multi-target* data for *RRMSE evaluated on separate targets* (a), *on target averages within data sets* (b) and *model size* (c). Better algorithms are on the right-hand side, the ones that do not differ significantly by the Nemenyi test (*p-value* $= 0.05$) are connected with a horizontal bar. CD is the critical distance.

if we treat each target separately, and with a *p-value* $= 4.1 \cdot 10^{-3}$, if we compare target averages over each data set. The model sizes are different with a *p-value* $= 1.6 \cdot 10^{-6}$. The average ranks and results of the Nemenyi test are given in Fig. 4 for RRMSE evaluated on separate targets (a), RRMSE evaluated on target averages within data sets (b), and model size (c).

Looking at diagram (a), the ranking of algorithms is similar as in the case of single-target data sets: random forests and the unlimited FIRE are more accurate than the limited FIRE and regression trees. However, due to the smaller critical distance, all differences are now significant. Evaluation over target averages within data sets (b) shows a similar picture, but because the sample size is smaller (11 data sets vs. 63 targets), the critical distance is larger and differences are less significant: only random forests are significantly better than regression trees and the limited version of FIRE. The diagram for size (c) is very similar as in the single-target case: the limited FIRE and regression trees are significantly smaller than the unlimited FIRE and especially than random forests. While the difference in size between random forests and the unlimited FIRE is not significant, the average size of a random forest is almost 65 times larger than the average size of a FIRE model (Table II). However, the difference in average accuracy is small.

C. Model Size Limitation for FIRE

Experiments presented in the previous two subsections included two versions of the FIRE algorithm, one with the maximum model size set to 20, and one without any model size restrictions. In this subsection, we present experiments with different values of the maximum model size parameter, which show how this model size limit influences the accuracy of models. We use values of 10, 20, 30, 40, 50, and ∞. Due to space limitations, we omit the detailed results and only present the average ranks diagrams in Fig. 5. Diagram (a) shows the results on single-target data. While all the differences in RRMSE are not significant, it is clear that increasing the model size improves the accuracy. Diagrams (b) and (c) show RRMSE on multi-target data for per-target evaluation and for per-data set target average evaluation, respectively. Because of a larger sample, there are more significant differences in (b) than in (c), however, what is common to both diagrams is the trend that models with less terms are also less accurate. The size limitation parameter can therefore be used as an accuracy for simplicity (and interpretability) trade-off setting.

Another interesting conclusion that we can draw from these diagrams is that while a model size of 20 seems enough to get models that are not significantly less accurate than the unlimited models for single-target domains, this is not the case for multi-target domains. Here, at least 50 rules are needed for an accuracy that is not significantly worse

Table II

DETAILED RESULTS OF ALGORITHM COMPARISON ON *multi-target* DATA. FOR EACH DATA SET WE FIRST GIVE THE *average RRMSE over all targets*, AND THEN THE *RRMSE for each target separately,* TOGETHER WITH STANDARD DEVIATION. IN EACH ROW, THE SMALLEST ERROR IS TYPESET IN BOLD. WE ALSO GIVE MODEL SIZES (#) FOR EACH OF THE ALGORITHMS. THE TWO FINAL ROWS GIVE THE AVERAGE RRMSES OVER ALL TARGETS, OVER DATA SET TARGET AVERAGES AND AVERAGE MODEL SIZE OVER ALL DATA SETS.

DATA SET TARGET ATTRIBUTES	TREE RRMSE	#	RANDOM FOREST RRMSE	#	FIRE RRMSE	#	FIRE 20 RULES RRMSE	#
COLLEMBOLA	0.97	3	**0.92**	13,193	0.94	505	0.96	20
SPECIES-NB	0.98 ±0.14		**0.94** ±0.14		0.96 ±0.15		0.98 ±0.16	
ABUD-TOTAL	0.96 ±0.47		**0.93** ±0.51		0.95 ±0.50		0.96 ±0.48	
ABUD-F.QUAD	0.96 ±0.14		**0.89** ±0.11		0.90 ±0.15		0.95 ±0.14	
EDM	0.72	11	**0.69**	2,923	**0.69**	677	0.71	20
D-FLOW	0.68 ±0.40		**0.66** ±0.30		**0.66** ±0.35		0.72 ±0.34	
D-GAP	0.75 ±0.09		0.71 ±0.07		0.71 ±0.14		**0.69** ±0.07	
FOREST KRAS	0.77	55	**0.66**	75,769	0.70	773	0.73	20
CC	0.64 ±0.03		**0.56** ±0.02		0.60 ±0.02		0.65 ±0.02	
FSH	0.66 ±0.05		**0.56** ±0.04		0.59 ±0.03		0.64 ±0.04	
DELVEG	0.64 ±0.04		**0.56** ±0.03		0.60 ±0.03		0.65 ±0.03	
VPV1-HMX	0.76 ±0.09		**0.65** ±0.09		0.68 ±0.08		0.72 ±0.09	
VPV1-H99	0.74 ±0.05		**0.62** ±0.04		0.66 ±0.03		0.71 ±0.03	
VPV1-H95	0.74 ±0.05		**0.63** ±0.04		0.66 ±0.03		0.71 ±0.04	
VPV1-H75	0.75 ±0.05		**0.64** ±0.04		0.67 ±0.03		0.71 ±0.04	
VPV1-H50	0.77 ±0.05		**0.67** ±0.05		0.70 ±0.05		0.74 ±0.06	
VPV1-H25	0.83 ±0.11		**0.73** ±0.12		0.76 ±0.11		0.78 ±0.12	
VPV1-H10	0.92 ±0.19		**0.81** ±0.23		0.83 ±0.22		0.85 ±0.22	
VPV1-H05	0.98 ±0.25		**0.86** ±0.30		0.87 ±0.29		0.89 ±0.29	
FOREST SLIVNICA	0.62	124	**0.54**	73,127	**0.54**	564	0.61	20
HEIGHT	0.64 ±0.05		**0.55** ±0.09		0.57 ±0.08		0.64 ±0.08	
COVER	0.61 ±0.03		0.52 ±0.05		**0.51** ±0.05		0.58 ±0.05	
META LEARNING	1.35	2	0.92	1,401	**0.86**	354	0.98	20
LTREE	1.47 ±0.72		0.93 ±0.47		**0.85** ±0.41		0.98 ±0.45	
C50-RULES	1.46 ±0.70		0.93 ±0.47		**0.84** ±0.40		0.96 ±0.43	
LINDISCR	1.12 ±0.52		0.86 ±0.39		**0.82** ±0.34		0.90 ±0.45	
MLCIB1	1.49 ±0.70		0.96 ±0.46		**0.86** ±0.41		0.99 ±0.43	
MLCNB	1.29 ±0.54		0.91 ±0.42		**0.90** ±0.40		1.01 ±0.42	
RIPPER	1.33 ±0.65		0.92 ±0.46		**0.80** ±0.42		0.90 ±0.41	
CLEM-RBFN	1.11 ±0.33		0.89 ±0.29		**0.84** ±0.27		1.04 ±0.40	
C50-TREE	1.47 ±0.69		0.94 ±0.44		**0.85** ±0.39		0.98 ±0.43	
CLEM-MLP	1.22 ±0.42		0.95 ±0.37		**0.93** ±0.37		1.03 ±0.40	
C50-BOOST	1.51 ±0.70		0.95 ±0.45		**0.88** ±0.40		0.99 ±0.44	
MICROARTHROPODS	0.77	52	**0.73**	12,411	0.74	443	0.86	20
ACARI	0.76 ±0.17		**0.71** ±0.18		0.73 ±0.15		0.83 ±0.19	
COLLEMBOLAN	0.74 ±0.13		**0.74** ±0.17		**0.74** ±0.14		0.82 ±0.16	
SH-BIODIV	0.81 ±0.03		**0.75** ±0.03		0.76 ±0.04		0.91 ±0.06	
SIGMEA REAL	**0.61**	12	0.65	22,506	0.66	209	0.69	20
MFO	**0.62** ±0.37		0.67 ±0.42		0.70 ±0.50		0.69 ±0.48	
MSO	**0.61** ±0.44		0.62 ±0.46		0.62 ±0.43		0.69 ±0.45	
SIGMEA SIMULATED	**0.04**	39	**0.04**	21,093	**0.04**	397	0.09	20
DISP-RATE	**0.04** ±0.02		**0.04** ±0.01		**0.04** ±0.01		0.10 ±0.02	
DISP-SEEDS	**0.03** ±0.02		**0.03** ±0.01		0.04 ±0.01		0.08 ±0.01	
SOLAR FLARE	**0.99**	2	1.05	3,974	1.00	24	1.02	20
C-CLASS	**0.99** ±0.31		1.03 ±0.29		1.00 ±0.33		1.01 ±0.32	
M-CLASS	**0.97** ±0.39		1.04 ±0.37		0.99 ±0.39		1.01 ±0.40	
X-CLASS	**1.01** ±0.72		1.07 ±0.69		1.02 ±0.72		1.04 ±0.72	

CONTINUED ON THE NEXT PAGE . . .

Table II
CONTINUED FROM THE PREVIOUS PAGE.

DATA SET TARGET ATTRIBUTES	TREE RRMSE	#	RANDOM FOREST RRMSE	#	FIRE RRMSE	#	FIRE 20 RULES RRMSE	#
VEGETATION	0.94	28	**0.81**	76,154	0.84	682	0.89	20
NUMBER-SPP	0.93 ±0.06		**0.76** ±0.04		0.79 ±0.04		0.90 ±0.05	
DEM	0.91 ±0.04		**0.67** ±0.03		0.71 ±0.04		0.80 ±0.07	
TWI	0.88 ±0.16		**0.72** ±0.18		0.76 ±0.16		0.80 ±0.16	
SOLAR	**1.00** ±0.08		**1.00** ±0.07		**1.00** ±0.07		**1.00** ±0.07	
THINVK	0.98 ±0.09		**0.94** ±0.09		0.96 ±0.09		0.97 ±0.09	
THK	0.98 ±0.09		**0.94** ±0.09		0.96 ±0.09		0.97 ±0.09	
EFF-RAIN	0.89 ±0.06		**0.67** ±0.05		0.72 ±0.05		0.82 ±0.08	
MINT-JUL	0.92 ±0.03		**0.71** ±0.03		0.74 ±0.03		0.82 ±0.04	
MAX-FEB	0.92 ±0.04		**0.68** ±0.04		0.74 ±0.06		0.83 ±0.06	
GRND-DPTH	0.96 ±0.15		**0.83** ±0.16		0.84 ±0.15		0.89 ±0.15	
SALINITY	1.02 ±0.24		**0.94** ±0.20		0.96 ±0.20		0.98 ±0.24	
WATER QUALITY	0.96	5	**0.90**	41,543	0.94	687	0.94	20
CLAD-SP	0.99 ±0.11		**0.93** ±0.11		0.96 ±0.14		0.97 ±0.12	
GONG-INC	1.00 ±0.06		**0.97** ±0.07		1.00 ±0.07		0.99 ±0.06	
OEDO-SP	1.00 ±0.12		**0.94** ±0.10		0.97 ±0.09		0.97 ±0.11	
TIGE-TEN	0.95 ±0.15		**0.88** ±0.14		0.93 ±0.14		0.91 ±0.14	
MELO-VAR	0.98 ±0.15		**0.91** ±0.14		0.95 ±0.16		0.95 ±0.14	
NITZ-PAL	0.89 ±0.05		**0.83** ±0.06		0.88 ±0.06		0.85 ±0.06	
AUDO-CHA	0.98 ±0.13		**0.95** ±0.14		0.97 ±0.13		0.97 ±0.13	
ERPO-OCT	0.97 ±0.10		**0.91** ±0.07		0.95 ±0.07		0.94 ±0.09	
GAMM-FOSS	0.91 ±0.07		**0.80** ±0.05		0.86 ±0.05		0.88 ±0.07	
BAET-RHOD	0.98 ±0.09		**0.90** ±0.09		0.94 ±0.09		0.95 ±0.10	
HYDRO-SP	0.97 ±0.10		**0.91** ±0.10		0.97 ±0.11		0.96 ±0.10	
RHYA-SP	0.95 ±0.12		**0.90** ±0.12		0.94 ±0.10		0.93 ±0.11	
SIMU-SP	1.00 ±0.10		**0.94** ±0.10		0.99 ±0.11		0.99 ±0.10	
TUBI-SP	0.89 ±0.09		**0.85** ±0.09		0.90 ±0.08		0.88 ±0.09	
AVERAGE – TARGETS	0.92		**0.79**		0.80		0.85	
AVERAGE – DATA SETS	0.80	30.3	**0.72**	31,281	**0.72**	483.2	0.77	20

than that of the full model. Of course, the exact values depend on the domain, the number of target attributes, and relations between them. But it could be expected that the task of modeling a multi-target domain is harder than the task of modeling a single-target domain, and therefore the corresponding models have to be more complex.

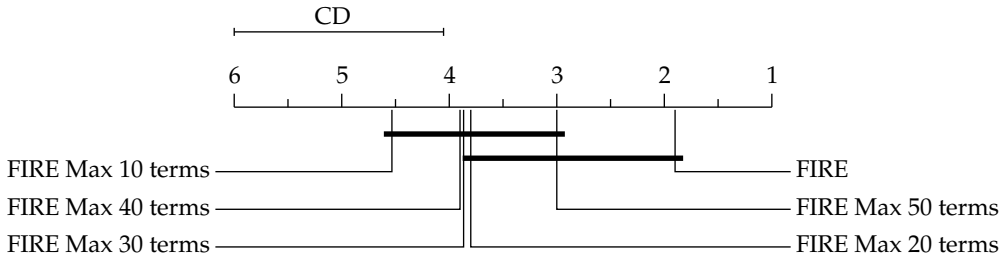
VI. CONCLUSIONS AND FURTHER WORK

In many application areas there is a need for methods that can learn interpretable multi-target models, i.e., models that predict several target attributes simultaneously. Rules are arguably one of the most interpretable model types, and we presented a method FIRE that can learn multi-target regression rule ensembles. We have adopted a rule ensembles approach and generalized it to multi-target regression domains. Our implementation has a simple parameter for limiting the number of rules in the learned model. This enables us to trade accuracy for size (interpretability) of learned models. We evaluated our algorithm with two parameter values: one that limits the number of rules to maximum 20, and one that has no size restrictions, and compared it to regression trees and random forests. We also investigated how the size limit affects the accuracy.

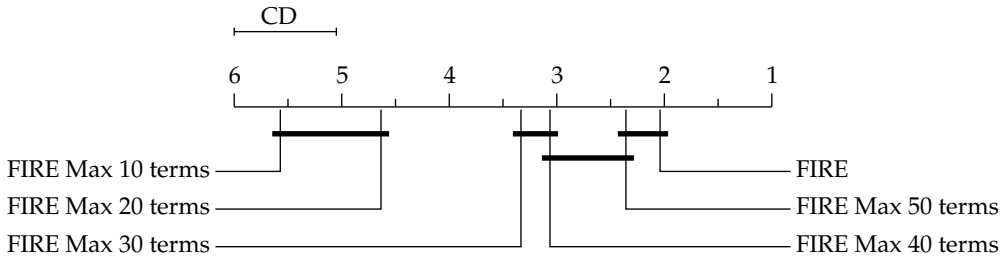
First, we evaluated FIRE on single-target domains in order to show that our implementation of rule ensembles also works on standard regression problems. The results show that, depending on the number of rules limit, the accuracy of FIRE is between the accuracies of regression trees and random forests. However, regression trees and both versions of FIRE generate smaller models than random forests. This suggests that FIRE can offer accuracy comparable to that of random forests, which are much larger.

Second and most important, we have evaluated FIRE on multi-target domains. The results are similar to the ones on single-target domains. Random forests and the unlimited FIRE are more accurate than the limited FIRE and regression trees. Again, model size of regression trees and the limited FIRE is significantly smaller than model size of the unlimited FIRE and random forests. Even though the difference in size between random forests and the unlimited FIRE is not significant, the average size of a random forest is almost 65 times larger than the average size of a FIRE model. Although the unlimited FIRE tends to generate somewhat less accurate models than random forest, these models are much smaller than random forests. Therefore, we believe the unlimited FIRE is a good choice for modeling multi-target regression problems.

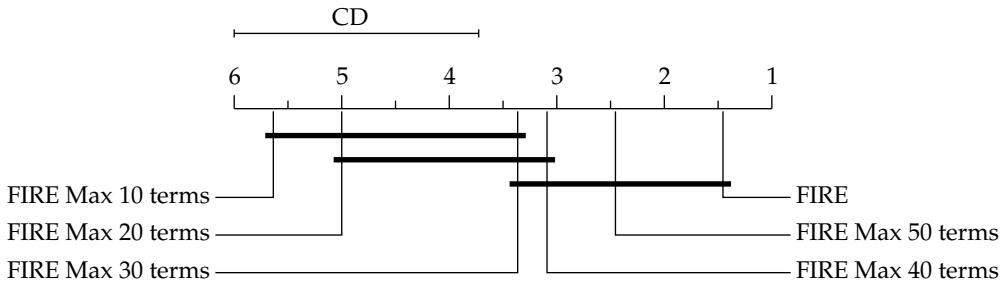
Finally, the investigation of the influence of the maximum model size on the accuracy confirms that this parameter can be successfully used as an accuracy for simplicity trade-



(a) single-target problems



(b) multi-target problems, per target evaluation



(c) multi-target problems, per data set target average evaluation

Figure 5. Average ranks diagrams of FIRE with different model size limitations for *RRMSE* on *single-target data* (a), on *multi-target data evaluated on separate targets* (b), and on *multi-target data evaluated on target averages within data sets* (c). Better algorithms are on the right side, the ones that do not differ significantly by the Nemenyi test ($p\text{-value} = 0.05$) are connected with a horizontal bar. CD is the critical distance.

off setting. The results show the general trend of larger models having better accuracy. The fact that the trend is more evident in the multi-target domains can be attributed to multi-target tasks being more complex and demanding more complex models for optimal accuracy.

Let us conclude with some ideas for further work. The original RuleFit method can also generate models that consist not only of rules, but also of simple linear functions (linear terms). We believe that the additions of linear terms could further improve the accuracy of our method. The experimental evaluation could also be extended by a comparison to existing multi-target regression rules, i.e., predictive clustering rules [8]. The optimal model size depends on the domain at hand. We believe it should be possible to automatically determine the optimal model size, which would be a compromise between accuracy and interpretability.

REFERENCES

[1] D. Demšar, S. Džeroski, T. Larsen, J. Struyf, J. Axelsen, M. B. Pedersen, and P. H. Krogh, "Using multi-objective classification to model communities of soil microarthropods," *Ecological Modelling*, vol. 191, no. 1, pp. 131–143, 2006.

[2] H. Blockeel, "Top-down induction of first order logical decision trees," Ph.D. dissertation, Katholieke Universiteit Leuven, Department of Computer Science, Leuven, Belgium, 1998.

[3] E. Suzuki, M. Gotoh, and Y. Choki, "Bloomy decision tree for multi-objective classification," in *Proceedings of the Fifth European Conference on Principles of Data Mining and Knowledge Discovery*, ser. LNCS, L. D. Raedt and A. Siebes, Eds. Berlin, Germany: Springer, 2001, pp. 436–447.

[4] B. Ženko and S. Džeroski, "Learning classification rules for multiple target attributes," in *Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, ser. LNCS, T. Washio, E. Suzuki *et al.*, Eds. Berlin, Germany: Springer, 2008, pp. 454–465.

[5] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[6] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, "Ensembles of multi-objective decision trees," in *Proceedings of the 18th*

European Conference on Machine Learning, ser. LNCS, J. N. Kok, J. Koronacki *et al.*, Eds. Berlin, Germany: Springer, 2007, pp. 624–631.

[7] R. S. Michalski, "On the quasi-minimal solution of the general covering problem," in *Proceedings of the Fifth International Symposium on Information Processing*, vol. A3, Switching Circuits, Bled, Yugoslavia, 1969, pp. 125–128.

[8] B. Ženko, "Learning predictive clustering rules," Ph.D. dissertation, University of Ljubljana, Faculty of computer and information science, Ljubljana, Slovenia, 2007.

[9] J. H. Friedman and B. E. Popescu, "Predictive learning via rule ensembles," *The Annals of Applied Statistics*, vol. 2, no. 3, pp. 916–954, 2008.

[10] H. Blockeel, L. D. Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proceedings of the 15th International Conference on Machine Learning*, J. W. Shavlik, Ed. San Francisco, CA: Morgan Kaufmann, 1998, pp. 55–63.

[11] P. Flach and N. Lavrač, "Rule induction," in *Intelligent Data Analysis*, M. R. Berthold and D. J. Hand, Eds. Berlin, Germany: Springer, 2003, pp. 229–267, second edition.

[12] K. Dembczyński, W. Kotłowski, and R. Słowiński, "Maximum likelihood rule ensembles," in *Proceedings of the Twenty-Fifth International Conference on Machine Learning*, ser. AICPS, W. W. Cohen, A. McCallum, and S. T. Roweis, Eds. Berlin, Germany: ACM, 2008, pp. 224–231.

[13] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[14] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] J. H. Friedman and B. E. Popescu, "Gradient directed regularization," Stanford University, Stanford, CA, Tech. Rep., 2004.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[17] H. Blockeel and J. Struyf, "Efficient algorithms for decision tree cross-validation," *Journal of Machine Learning Research*, vol. 3, pp. 621–650, 2002.

[18] C. Kampichler, S. Dzeroski, and R. Wieland, "Application of machine learning techniques to the analysis of soil ecological data bases: relationships between habitat features and collembolan community characteristics," *Soil Biology and Biochemistry*, vol. 32, no. 2, pp. 197–209, 2000.

[19] A. Karalič and I. Bratko, "First order regression," *Machine Learning*, vol. 26, no. 2-3, pp. 147–176, 1997.

[20] S. Džeroski, A. Kobler, V. Gjorgjioski, and P. Panov, "Using decision trees to predict forest stand height and canopy cover from LANSAT and LIDAR data," in *Proceedings of the 20th International Conference on Informatics for Environmental Protection*, K. Tochtermann and A. Scharl, Eds. Achen, Germany: Shaker, 2006, pp. 125–133.

[21] D. Stojanova, "Estimating forest properties from remotely sensed data by using machine learning," Master's thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, 2009.

[22] S. Džeroski, L. Todorovski, and H. Blockeel, "Relational ranking with predictive clustering trees," in *Proceedings of the Workshop on Active Mining (ICDM)*. IEEE Computer Society, 2002, pp. 9–15.

[23] D. Demšar, M. Debeljak, C. Lavigne, and S. Džeroski, "Modelling pollen dispersal of genetically modified oilseed rape within the field," in *Abstracts, 90th ESA Annual Meeting*. Montreal, Canada: The Ecological Society of America, 2005, p. 152.

[24] S. Džeroski, N. Colbach, and A. Messéan, "Analysing the effect of field character on gene flow between oilseed rape varieties and volunteers with regression trees," in *Proceedings of the Second International Conference on Co-existence between GM and non-GM based Agricultural Supply Chains*, A. Messéan, Ed. Montpellier, France: Agropolis Productions, 2005, pp. 207–211.

[25] D. Kocev, S. Džeroski, M. D. White, G. R. Newell, and P. Griffioen, "Using single and multi target regression trees and ensembles to model a compound index of vegetation condition," *Ecological Modelling*, vol. 220, no. 8, pp. 1159–1168, 2009.

[26] S. Džeroski, D. Demšar, and J. Grbović, "Predicting chemical parameters of river water quality from bioindicator data," *Applied Intelligence*, vol. 13, no. 1, pp. 7–17, 2000.

[27] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.