IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 27, 2020, accepted December 13, 2020, date of publication January 12, 2021, date of current version January 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051185

Multi-Target Regression Rules With Random Output Selections

MARTIN BRESKVAR AND SAŠO DŽEROSKI
Department of Knowledge Technologies, Jožef Stefan Institute, 1000 Ljubljana, Slovenia
Jožef Stefan International Postgraduate School, 1000 Ljubljana, Slovenia

Corresponding author: Martin Breskvar (martin.breskvar@ijs.si)

This work was supported in part by the Slovenian Research Agency (SRA) via the research program under Grant P2-0103, in part by young researcher grants to MB, in part by the research projects under Grant N2-0128, Grant J2-2505, and Grant J7-9400, in part by TAILOR (a project funded by the EU Horizon 2020 research and innovation programme) under Grant 952215, in part by AI4EU under Grant 825619, and in part by HBP SGA2 under Grant 785907.

ABSTRACT In this article, we address the task of multi-target regression (MTR), where the goal is to predict multiple continuous variables. We approach MTR by learning global models that simultaneously predict all of the target variables, as opposed to learning a separate model for predicting each of the target variables. Specifically, we learn rule ensembles by generating many candidate rules and assigning them weights that are then optimized in order to select the best performing subset of rules. Candidate rules are generated by transforming ensembles of generalized decision trees, called predictive clustering trees (PCTs), into rules. We propose to extend an existing multi-target regression rule learning method named FIRE by learning tree ensembles that use random output selections (ROS). Such ensembles force individual PCTs to focus only on randomly selected subsets of target variables. The rules obtained from the tree ensemble also focus on various subsets of the target variables (FIRE-ROS). We use three different ensemble methods to generate candidate rules: bagging and random forests of PCTs, and ensembles of extremely randomized PCTs. An experimental evaluation on a range of benchmark datasets has been conducted, where FIRE-ROS is compared to three interpretable methods, namely predictive clustering rules, MTR trees and the original FIRE method, as well as state-of-the-art MTR methods, in particular ensembles of extremely randomized PCTs with ROS, random linear combinations and extremely randomized MTR trees with random projections of the target space. The results show that FIRE-ROS can improve the predictive performance of the FIRE method and that it performs on par with state-of-the-art (non-interpretable) MTR methods.

INDEX TERMS Multi-target regression, rule learning, ensemble methods, structured outputs.

I. INTRODUCTION

Machine learning consists of several fields. One of them is the highly researched supervised learning, where the goal is to produce a model, that is able to take a previously unseen data example and predict the variable of interest, i.e., the target variable. The type of the target variable determines the machine learning task at hand. If the target variable is of numeric data type, the task is called regression. If the target variable is of discrete type, the task at hand is classification.

Problems, where we are only interested in predicting one target value, are very common. However, many problems exist, where we are interested in predicting more than one variable, i.e., the outputs are complex, compared to simple single value prediction scenarios. In recent years, many machine learning methods have been developed which can generate predictive models for simultaneously predicting several target values, i.e., for structured output prediction (SOP). Examples of SOP tasks include: multi-target regression (MTR), multi-label classification (MLC), time series prediction, etc. Machine learning tasks where the predicted value is a single value (numeric or discrete) are special cases of SOP tasks.

This work considers the task of MTR – predicting multiple continuous variables. Many real-life problems can be formulated as a MTR task, e.g., predicting the production of secondary metabolites in fungi [20] (life sciences), learning habitat models for a variety of species, predicting forest

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

properties and conditions, water quality prediction [10], [25] (ecology), modeling power consumption within a spacecraft [8], [32] (space operations), etc.

The main difference between SOP and single-target tasks is how the output space is exploited during model learning. The simplest way of handling many target attributes is to learn a model for each target attribute separately. Methods solving the MTR task in such a way are called *local* as they only see one target. This is a valid approach but has two main disadvantages. First, when the output space is of a high dimension (i.e., contains many target attributes), learning one model for each target attribute consumes a lot of resources. The second disadvantage is that the target attributes can be related in some way. Learning them separately prevents the potential exploitation of relations between them. In contrast, global methods build models that simultaneously predict all targets, taking into account all the target variables and their relations while building the models.

Decision rules and decision trees are among the most interpretable model types. With the rise of black-box models (e.g., deep neural networks), data scientists are facing a hard to overcome obstacle when they are tasked with explaining the predictions made by such models. There have been attempts to explain such models by providing descriptions of otherwise non-interpretable internal structures [39]. The ability to adequately explain the predictions helps with user acceptance (e.g., in medicine) and also conforms to legislation regarding fairness, equal opportunities, privacy, etc. Learning rule-based predictive models can be considered as a natural choice, when the end result should be a transparent model with interpretable predictions.

We propose a rule-learning method for the task of MTR, called Fitted Rule Ensembles with Random Output Selections (FIRE-ROS). The method extends the existing rule-learning method FIRE and learns an ensemble of predictive clustering trees (PCTs) with random output selections (ROS). The individual trees in the ensemble are transformed into a set of candidate rules which is optimized to find the best-performing subset of rules by changing the weights of individual rules.

In contrast to a preliminary presentation of our approach [7], where proof of principle is given on a single dataset, we perform an empirical evaluation over a variety of benchmark datasets. We systematically determine whether the integration of ROS tree ensembles with FIRE can improve the predictive performance of the learned models. We summarize the main contributions of this work as follows:

- An extension of a rule-learning approach for addressing the MTR task that selects random subsets of target attributes to generate candidate rules focused on those specific target attributes.
- An empirical evaluation of the proposed method on 16 benchmark datasets, where three different approaches are used to generate candidate rules. The evaluation provides a performance assessment of the original and proposed rule-based method as well as individual multi-target regression trees and predictive clustering rules (competing methods). The analysis also includes parameter setting recommendations for the proposed method.
- Rule ensembles that can contain rules that only give partial predictions and a method to learn them.
- A comparison of the proposed method with state-of-the-art and other interpretable MTR methods.

The remainder of this article is organized as follows. Section II outlines the task definition and related work. Section III presents the proposed method FIRE-ROS that uses tree ensembles and the ROS extension. Section IV provides details about the experimental design and evaluation, the results of which are presented and discussed in Section V. Finally, we conclude the article and provide directions for further work.

II. BACKGROUND AND RELATED WORK

We first formally describe the machine learning task of MTR [27]. **Given:**

- An input space X, with tuples of dimension d, containing values of primitive data types, i.e., $\forall x_i \in X, x_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_d})$,
- An output (target) space Y, with tuples of dimension t, containing real values, i.e., $\forall y_i \in Y, y_i = (y_{i_1}, y_{i_2}, \ldots, y_{i_t})$, where $y_{i_k} \in \mathbb{R}$ and $1 \leq k \leq t$
- A set of examples S, where each example is a pair of tuples from the input and the output space, i.e., $S = \{(x_i, y_i) | x_i \in X, y_i \in Y, 1 \leq i \leq N\}$ and N is the number of examples in S ($N = |S|$),
- A quality criterion c, which rewards models with high predictive accuracy and low complexity.

Find: A function $f : X \rightarrow Y$ such that f maximizes c. In this article, the function f is represented by an ensemble of multi-target regression rules.

The work presented here is related to several areas: rule learning for MTR, ensemble learning and output space decomposition. A general overview of MTR methods groups them into *problem transformation methods* and *algorithm adaptation methods* [5]. Several publications exist, where authors propose methods that cannot be categorized as strictly problem transformation or algorithm adaptation methods [22], [29], [35]–[37]. The method proposed in this article is one of those *in-between* methods. Here, we will focus only on methods that are most closely related to our work. For a more comprehensive list of related work, please refer to Borchani *et al.* [5].

We begin with **rule learning methods**. To the best of our knowledge, there are three known rule learning methods for MTR: two for batch learning and one for learning from data streams. For batch learning, predictive clustering rules (PCRs) [38] are learned by using the well known sequential covering approach, extended to multiple outputs. Fitted rule ensembles (FIRE) [1] learn rules by first learning an ensemble of generalized decision trees, transcribing them to weighted decision rules and then optimizing their

weights in order to select the best subset. The streaming method [12] adapts the currently learned rules to the changes in the data stream. If the algorithm determines, that it is no longer beneficial to learn all target attributes simultaneously, it splits the current rule into two rules where each of them learns a separate subset of the originally learned target variables.

Next, we will take a look at **ensemble methods** and discuss **output space decompositions** along the way, because they are used mainly with ensemble methods. The proposed method internally uses ensembles of generalized multi-output decision trees called Predictive Clustering Trees (PCTs), bagging and random forests of PCTs [27] as well as extremely randomized PCTs [24].

Several output space transformation methods exist, but they mainly focus on the task of MLC. Joly *et al.* [22] randomly generate a projection matrix which is then used to project the original values of output variables to a new vector space. The user can specify the desired output space dimension (usually substantially smaller than the original output space dimension). They refer to the Johnson-Lindenstrauss lemma to justify the fact that the projection matrix can have a smaller horizontal dimension than the original output space. If the output space projection matrix satisfies the lemma, the variance computations in the projected space will be ϵ-approximations of the variance in the original output space. Variance calculations are made in the projected space while the predictions are made directly in the original output space, removing the need for a decoding step. They use multi-output regression trees to calculate the variances in the projected space and then apply a threshold to obtain predictions for labels (i.e., MLC setting). No MTR results are reported. The same authors propose a gradient boosting method [21] for MTR that uses the said random projections of the output space to automatically adapt to the output correlation structure. In contrast, Breskvar *et al.* [6] propose random output selections (ROS) for ensembles of PCTs for MTR. RAkEL (Random k-labelsets) [37] is one of the most well-known ensemble methods for MLC. The main idea of RAkEL is to create an ensemble-like wrapper method that can use existing (single-target) classification algorithms as base learners. RAkEL transforms a multi-label classification problem into many single-target multi-class classification problems. An ensemble is constructed by providing a small random subset of k labels (organized as a label powerset) to each base learner. Finally, RLC (Random Linear Combinations) [36] solves the MTR task with a transformation of the output space by generating many new target variables as random linear combinations of the original target variables (the number of original target variables to be used in the linear combinations is defined by the user). Then, an arbitrarily selected multi-target algorithm is trained to predict the new target variables. In the last stage of this algorithm, the predicted target values are translated back to the original target space by inverting the random linear transformation.

III. FITTED RULE ENSEMBLES FOR MTR WITH RANDOM OUTPUT SELECTIONS (FIRE-ROS)

In this section, we propose Fitted rule ensembles with random output selections for multi-target regression – FIRE-ROS. The proposed method extends the existing MTR rule learning system FIRE [1] by integrating the ROS [6], [7] tree ensemble extension. Ultimately, we generate rule-based MTR models that contain decision rules obtained with the ROS extension. The resulting rules may make predictions only for a subset of target attributes. ROS was first implemented with ensembles of PCTs and was shown to improve their predictive performance. The proposed approach aims to improve the predictive performance of the FIRE method for MTR.

The rest of this section is divided into several parts. Because FIRE uses ensembles of MTR trees, we will first describe how such ensembles are learned and how they are extended with ROS. Next, we will explain the FIRE method in greater detail. Finally, we will describe the integration of FIRE and ROS tree ensembles, which constitutes the proposed method.

A. TREE ENSEMBLES FOR MULTI-TARGET REGRESSION WITH ROS

In our work, we use the global approach to learning an ensemble of MTR decision trees. In particular, we use ensembles of predictive clustering trees (PCTs). PCTs were first introduced as an implementation of the predictive clustering paradigm, combining supervised and unsupervised learning [3]. This approach has been used as a starting point for numerous applications and methods [1], [4], [6], [7], [27], [28], [34], [38]. All mentioned approaches, including the one proposed in this article, are implemented in the CLUS software package available at http://source.ijs.si/ktclus/clus-public.

1) PREDICTIVE CLUSTERING TREES (PCTs)

PCTs are a generalization of decision trees towards structured output prediction. Thus, learning a single PCT follows the standard top-down induction (TDI) algorithm. In contrast to standard decision trees, where the task is to model only one target attribute, PCTs can model multiple target attributes.

$$\mathrm{IR} = \mathrm{Impurity}(S) - \sum\nolimits_{S_i \in \mathcal{P}} \frac{|S_i|}{|S|} \mathrm{Impurity}(S_i) \quad (1)$$

$$\mathrm{Impurity}(S) = \frac{1}{|A_t|} \sum_{a \in A_t} \mathrm{Var}_a(S) \quad (2)$$

$$\mathrm{Var}_a(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} (a_i - \overline{a})^2 \quad (3)$$

The candidate splits are evaluated greedily, by calculating the impurity reduction heuristic (1), where S denotes a set of data examples before the split condition is applied and $\mathcal{P}$ denotes a collection of disjunct sets of data examples after the candidate split is applied (i.e., a partition). We calculate impurity (2) as the arithmetic mean of the variances (3) of individual target attributes (A_t), where $\mathrm{Var}_a(S)$ represents the

variance of the attribute a in dataset S, a_i represents the value of the attribute a in the i^{th} data example of dataset S, and $\overline{a}$ represents the mean value of attribute a in the dataset S. The candidate condition that reduces the impurity of its child nodes the most, is selected. When the TDI algorithm cannot find any more suitable splits, it creates leaf nodes (terminal nodes of the decision tree). Note, that the impurity reduction heuristic values are calculated by taking into account all target attributes. Finally, for each leaf node, a *prototype function* is generated based on the data instances that have reached that node.

At prediction time, the above-mentioned prototype function is used to calculate the predictions. In this article, we use a very simple prototype function that predicts a constant value for each target attribute. The values are calculated (during learning) as the arithmetic mean of each target attribute, based on the training data instances. All other (root and intermediate) nodes contain tests that are used to navigate data instances through the tree structure towards the leaf nodes, where predictions are calculated.

Predictive clustering trees [3] generalize the notion of decision trees, which was originally used to denote classification trees. Besides (single-target) prediction, PCTs allow also for multi-target prediction (classification and regression), as well as clustering. PCTs distinguish between descriptive attributes (that appear in tests in the tree), target attributes (for which predictions appear in tree leaves) and clustering attributes (for which the variance and variance reduction are used in tree construction). PCTs can be used in an unsupervised manner, where descriptive and clustering attributes coincide and there are no target attributes; in a semi-supervised manner, where clustering attributes include both the target and the descriptive attributes; as well as in fully supervised multi-target prediction (where clustering and target attributes coincide). It is the latter paradigm (of multi-target prediction) that has been adopted for recent decision tree implementations, such as the one in scikit-learn,[1] which however are still more specific than PCTs, while clearly being more general than the original meaning of the term decision tree.

The TDI algorithm is slightly modified when using ROS. The ROS extension forces the TDI algorithm to calculate the impurity reduction heuristic values on a subset of the target attributes. The actual subset of target attributes is selected by the ROS-extended ensemble-building algorithm. A more detailed explanation is in the following section.

2) ENSEMBLES OF PREDICTIVE CLUSTERING TREES

An ensemble is a set of base (predictive) models that we construct to lift the predictive performance over that of individual base models. We use three ensemble construction methods (bagging, random forests and extremely randomized trees) that have been extended to structured output prediction [24], [27] and later with random output selections (ROS) [6]. The base models used by the mentioned ensemble methods are PCTs.

Standard ensembles of PCTs contain trees that are learned from (and can later be used to predict) all target attributes. ROS ensembles change this by learning individual PCTs in the ensemble using different subsets of target attributes. Target subsets are selected uniformly at random before the induction of each PCT. The manipulation of the input space remains the same as with the standard ensemble algorithms.

In general, ensemble models make predictions by aggregating the predictions of individual base models. One of the most common aggregation mechanisms in regression tasks is to average the predictions $\hat{y}_i = \frac{1}{N}\sum_{\text{tree}=1}^{N} y_i^{\text{tree}}$, where i represents the i-th target attribute and N represents the number of PCTs in the ensemble. This is how standard PCT ensembles make predictions. With ROS ensembles, one can make (i) **full predictions**, where each PCT predicts all target attributes or (ii) **partial predictions**, where each PCT predicts only those target attributes, that were used during its learning. Note, that the first option is not the same as using the original ensembles, because ROS always learns individual PCTs using a subset of target attributes in the search heuristic, regardless of how predictions are made. PCTs can be forced to make predictions for all target attributes even if the tree was learned only from a subset of them. This is possible because the TDI procedure can see the values of all target attributes when generating the prototype function, as opposed to the impurity reduction heuristic calculation, where only a subset of target attributes is used.

B. FITTED RULE ENSEMBLES FOR MTR (FIRE)

The final rule ensemble model takes the form given by (4). The model consists of three parts:

1) The vector ***avg*** that provides predictions for all target attributes (each component represents a target). The value of an individual component is defined as the arithmetic mean of the values of the corresponding target attribute from the original training dataset.
2) K weighted decision rules $\boldsymbol{r}_i(\boldsymbol{x})$ that trigger only when their conditions are fulfilled. Weights are denoted with the symbol w. The weight w_i belongs to the rule r_i.
3) Optional simple linear functions (linear terms) – one term for each pair of numeric predictive and target attributes. For example, for the Soil resilience dataset [9], FIRE and FIRE-ROS would have generated 56 linear terms (7 continuous predictive attributes multiplied by 8 target attributes).

$$f(\boldsymbol{x}) = w_0\boldsymbol{avg} + \sum_{i=1}^{K} w_i \boldsymbol{r}_i(\boldsymbol{x}) + \underbrace{\sum_{t=1}^{T}\sum_{d=1}^{D} w_{(t,d)}\boldsymbol{x}_{(t,d)}}_{\text{optional}} \quad (4)$$

Each linear term $x_{(t,d)}$ influences the overall prediction of the ensemble by introducing a linear model that applies only to target t and uses the value of predictive attribute d. For example, if we are predicting 5 targets, the linear term $x_{(3,1)}$

[1] https://scikit-learn.org/

would be of the form $(0, 0, x_1, 0, 0)$, where x_1 is the value of the first predictive attribute of the instance.

The method Fitted rule ensembles for MTR (FIRE) learns MTR rules in two steps. First, it generates a large pool of candidate rules by learning a random forest ensemble of PCTs, which are then converted to MTR rules. Then, each rule is assigned an initial weight of zero and the algorithm optimizes the weights to get a model with a good predictive performance [16]. This approach assumes that the set of generated candidate decision rules contains enough good rules that will end up in the final model. If the set of candidate rules does not contain good rules, weight optimization will most likely produce a model with poor predictive performance. The success of this method is therefore dependent on the quality of the generated candidate rules.

The candidate rules are obtained by decomposing every PCT into as many rules as there are leaf nodes in it. Rules take the following form:

$$\text{IF } \textit{condition} \text{ THEN } \textit{prediction}.$$

The *condition* part of a decision rule is constructed by taking the conditions in the test nodes along the path from the root node of the tree to the last non-leaf node and using them to form a logical conjunction, e.g., $\text{condition}_1 \wedge \text{condition}_2 \wedge \ldots \wedge \text{condition}_n$. The leaf node holds the prototype function, which is copied verbatim into the *prediction* part of the decision rule, i.e., $\textit{prediction} = [p_{t_1}, p_{t_2}, \ldots, p_{t_n}]$, where p_{t_i} is the prediction for the i-th target attribute.

A gradient directed optimization approach is used to minimize the single-target (ST) squared loss, defined as $\mathrm{L_{ST}}(f(x), y) = \frac{1}{2}(f(x) - y)^2$, where $f(x)$ and y correspond to predicted and true values respectively. Aho *et al.* [1] adapt the loss function to a multi-target (MT) setting. The appropriate convex multi-target loss function (5) corresponds to the average loss over all target attributes (T).

$$\mathrm{L_{MT}}(f(x), y) = \frac{1}{T}\sum_{t=1}^{T} \mathrm{L_{ST}}(f_t(x), y_t) \tag{5}$$

FIRE and FIRE-ROS do not learn rules directly, but rather use rules that were obtained from the tree ensemble. When building standard tree ensembles, the trees are allowed to grow until only one data instance is left in a leaf node. This generates deep decision trees and consequently results in long rules, i.e., rules with many conditions. The obvious downside of this is that all rules are very specific, i.e., they apply to a very small number of data instances and are complicated to understand. Moreover, no short rules are generated to model higher-level concepts.

It is because of this, that Friedman and Popescu [16] propose to vary the maximal depth of decision trees within the ensemble. Rules obtained from such an ensemble can be short, long or somewhere in between. The depth of individual trees is determined in a randomized fashion. The number of leaf nodes t is determined for each PCT in the ensemble with a random variable: $t = 2 + \lfloor \gamma \rfloor$, where γ is drawn from an exponential distribution with probability

$$\Pr(\gamma) = \frac{e^{-\gamma/(\overline{L}-2)}}{\overline{L}-2},$$

where $\overline{L}$ is the average number of leaf nodes in all PCTs. The depth of a PCT can now be computed as $d = \lceil \log_2(\mathrm{t}) \rceil$, assuming the root node is at depth 0. $\overline{L}$ is a parameter of the algorithm.

C. INTEGRATING FIRE AND ROS

In this section, we describe the two required steps to integrate ROS tree ensembles with the FIRE rule learning system.

The first step pertains to how the candidate rules are generated. We swap the original random forests of PCTs with the ROS-generated ensembles and convert them to candidate rules. At this point, FIRE-ROS is capable of producing two types of rules: **fully-predicting** and **partially-predicting**. The first are those that give predictions for all target attributes and can be found in models generated with FIRE or FIRE-ROS. The latter are rules that only predict a subset of target attributes and can only be generated with FIRE-ROS.

The second step is more involved as it requires low-level modifications of how the algorithm calculates gradients during the gradient directed optimization. In particular, we have to modify the covariance calculations to take into account the fact that not all rules give predictions for all target attributes. FIRE calculates three types of covariances: (i) between predictions of two rules, (ii) between a prediction of a rule and a prediction of a linear term, and (iii) between predictions of two linear terms. Please refer to [1] for more details on how this is done.

To implement FIRE-ROS, modifications are required only within the first two calculations and should only be applied when partially-predicting rules are used. When fully-predicting rules are used, the optimization is technically equal for both FIRE and FIRE-ROS.

As in the original FIRE method, we also calculate the average covariance between two rules by averaging covariances of individual targets. However, in contrast to the original calculation, where all targets are predicted by all rules, we iterate over the target attributes and check whether both rules give a prediction for a specific target attribute. If they do not, we omit such predictions from the covariance calculation. When calculating the average covariance, we average only over the target attributes that were predicted by both rules, i.e., we use the number of joint target attributes $|R_{r_1} \cap R_{r_2}|$, where R_x represents the set of target attributes considered by rule x. If the two rules have a completely disjoint set of target attributes, the returned covariance is zero – the predictions of the two rules do not interfere with each other because they do not address the same target attributes. A similar approach is taken when calculating the covariance between a decision rule and a linear term. Each linear term affects only one target attribute. If the rule does not give a prediction for that target, the covariance is zero. Otherwise, the covariance calculation

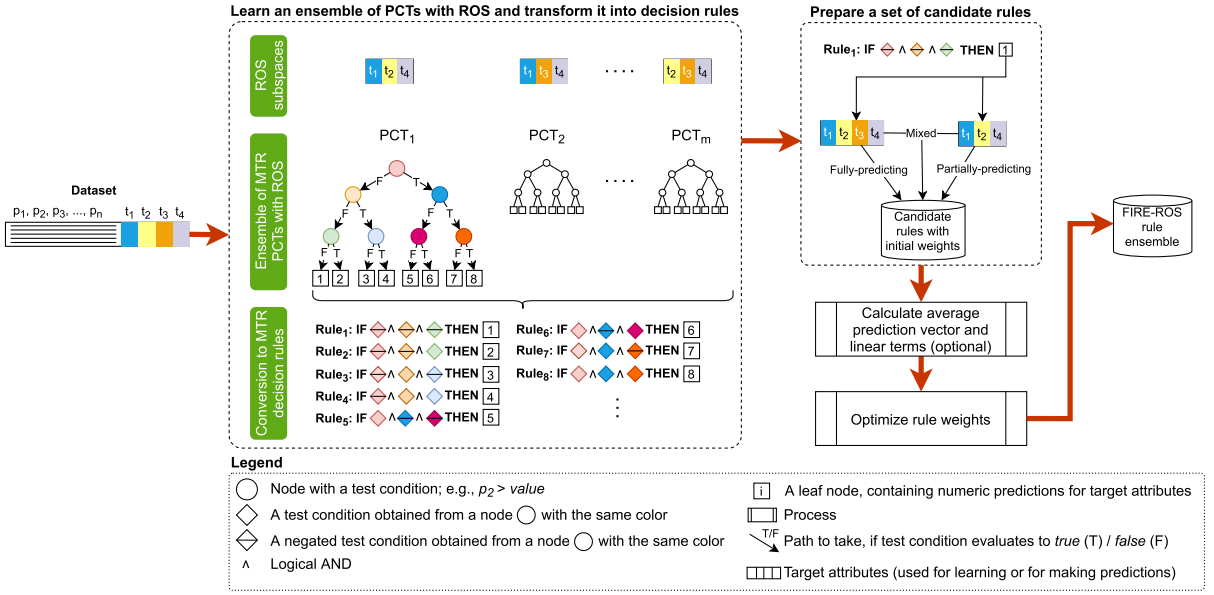


FIGURE 1. The figure depicts the process of training a FIRE-ROS model of multi-target regression (MTR) rules with random output selections (ROS) on a dataset with 4 numeric target attributes. A dataset with predictive attributes ($p_1, p_2, \ldots, p_n$) and target attributes ($t_1, \ldots, t_4$) is used to generate a set of candidate rules. The candidate set is generated by learning an ensemble of m predictive clustering trees (PCTs) with ROS. For each PCT in the ensemble, a subset of target attributes is provided to the TDI algorithm. Then, each trained PCT is decomposed into MTR rules. Depending on the user-selected prediction strategy, candidate rules can either be fully-predicting (predict values for all 4 target attributes), partially-predicting (predict values only for those target attributes that are in the ROS subspace) or mixed (both; two candidate rules are created). Before the weight optimization, average prediction vector and (optional) linear terms are generated and given initial weights. The weights of candidate rules (and linear terms) are then optimized to select the best performing set of decision rules.

is averaged over all predicted target attributes. This approach is the same for fully and partially-predicting rules.

D. GENERATING CANDIDATE RULES WITHIN FIRE-ROS

The integration of ROS tree ensembles with FIRE makes it possible to learn partially-predicting rules. Such rules are more target-specific and can potentially be more easily explained or understood by a domain expert. However, FIRE-ROS does not impose whether fully or partially-predicting rules should be used as candidates. Instead, this choice is left to the user.

With FIRE-ROS, the user has the option to generate candidate rules in three ways: (i) only fully-predicting rules or(ii) only partially-predicting rules or (iii) fully and partially-predicting rules. Our initial experiments show, that using only partially-predicting candidate rules results in models with poor predictive performance. However, giving the optimization procedure the ability to select partially-predicting rules without forcing it into doing so, resulted in an increase in predictive power. This would, ideally, guide the optimization towards a rule set that contains fully-predicting decision rules which globally cover the target space and partially-predicting rules that make local corrections for specific target attributes. Figure 1 depicts the overall process of training FIRE-ROS models.

The FIRE and FIRE-ROS approaches allow for a different kind of candidate rule generating methods. When no other sources of candidate rules are available, tree ensembles can be used to create an abundant and diverse set of candidate rules. FIRE and FIRE-ROS both use this approach by default. However, tree ensembles do not need to be the (only) source of candidate rules. It is possible to consider rules obtained from other rule learning approaches. It would also be possible to include explicit domain knowledge in the form of decision rules. While FIRE is capable of using only fully-predicting rules, with the introduction of FIRE-ROS, this limitation is removed and the candidate decision rules do not need to predict all target variables. This makes FIRE-ROS less strict than FIRE, regarding what kind of candidate rules can be used to generate an optimized rule ensemble.

IV. EXPERIMENTAL DESIGN

To evaluate the performance of FIRE-ROS ensembles, we perform experiments on various benchmark datasets. This section presents: (i) the experimental questions addressed, (ii) the evaluation measures used, (iii) the benchmark datasets and (vi) the experimental setup (including the parameter instantiations for the methods used in the experiments).

A. EXPERIMENTAL QUESTIONS

In our experiments, we construct rule-based models by using ensembles of PCTs to populate the initial pool of candidate rules. The goal of this work is to establish whether output space selections (ROS) affect the resulting predictive models and how. We investigate the generated models in terms of their **predictive performance**. We focus on (i) the predictive

performance of the proposed models as compared to that of the models produced by the original approach (FIRE) and (ii) comparison to other MTR methods. In addition to predictive performance, we also take interest in the **interpretability** of the resulting models. Rule-based models are human-readable, which is why the models and their predictions can be interpreted and understood. The experiments and their evaluation have been designed with the following research questions in mind:

1) What is the best value for the ROS output space size to be used with FIRE-ROS ensembles? Are there any differences between the three ensemble methods used with FIRE-ROS? Are there any differences between FIRE-ROS and PCT ensembles with ROS?
2) Does the use of partially-predicting rules improve the predictive performance of FIRE-ROS?
3) How do FIRE-ROS ensembles compare to the original FIRE ensembles in terms of predictive performance?
4) How do FIRE-ROS models compare to state-of-the-art MTR methods in terms of predictive performance?
5) How does the interpretability influence the predictive power of FIRE-ROS models?

B. EVALUATION MEASURES

In order to understand the effects of using ROS within FIRE, we first need to evaluate the models induced with the FIRE approach. Empirical evaluation is commonly used in machine learning. We assess the performance of a given model in terms of evaluation measures. Below, we describe the measures used for assessing predictive power and interpretability used in this article.

The **predictive performance** of a MTR model is assessed by using the average relative root mean squared error (aRRMSE), which averages the relative root mean squared errors (RRMSE) of individual target variables. RRMSE is a relative performance measure, comparing the performance of a model at hand against that of a baseline model which always predicts the arithmetic mean of all values of a given target in the learning set. Specifically, the value $\bar{y}_i$ in (6) is the prediction of the baseline model for the i-th target variable, while the value $\hat{y}_i^{(e)}$ represents the predicted value of the model under evaluation for the i-th target variable of example e. The values $y_i^{(e)}$ denote the true target values. Lower aRRMSE values are better.

$$\begin{aligned} \text{aRRMSE} &= \frac{1}{t}\sum_{i=1}^{t}\text{RRMSE}_i \\ &= \frac{1}{t}\sum_{i=1}^{t}\sqrt{\frac{\sum_{e=1}^{N_{\text{test}}}(y_i^{(e)}-\hat{y}_i^{(e)})^2}{\sum_{e=1}^{N_{\text{test}}}(y_i^{(e)}-\bar{y}_i)^2}} \end{aligned} \tag{6}$$

We also monitor how much our models **overfit** the training data by calculating their relative decrease of performance on the testing data w.r.t. that on the training data. Smaller values for the overfitting score are better (less overfitting). We calculate the overfitting score as

$$\text{OS} = \frac{\text{aRRMSE}_{\text{test}} - \text{aRRMSE}_{\text{train}}}{\text{aRRMSE}_{\text{train}}}, \tag{7}$$

where the train and test subscripts denote aRRMSE calculated on train and test sets respectively.

The **interpretability** of a model is measured in terms of its model size. Specifically, we measure the number of rules and the number of linear terms for rule-based models. For tree-based models, size is measured by the total number of leaf nodes in the tree model. In both cases, smaller values are better. We do not consider the other model types, used in this article, interpretable.

C. DATA DESCRIPTION

To evaluate the proposed method, we use 16 benchmark datasets that contain multiple continuous target attributes and are mainly from the domain of ecological modeling. Table 1 shows the main characteristics of the considered datasets. In order to have as a general evaluation as possible, we use datasets of different sizes in terms of the number of instances, number of predictive and number of target attributes. Detailed descriptions of the datasets are available in the respective references.

TABLE 1. Properties of the Considered MTR Datasets With Multiple Continuous Targets: Number of Examples (N), Number of Predictive Attributes (discrete/continuous, d/c), Number of Target Attributes (t)

#	Dataset	N	d/c	t
1	Forestry Kras [15]	60607	0/160	11
2	Vegetation Clustering [18]	29679	0/65	11
3	Vegetation Condition [25]	16967	1/39	7
4	Water quality [2, 14]	106	0/16	14
5	ATP 1D [33]	337	0/441	6
6	ATP 7D [33]	296	0/441	6
7	RF1 [33]	9125	0/64	8
8	RF2 [33]	9125	0/576	8
9	Sales [23, 33]	639	0/401	12
10	SCM 1D [33]	9893	0/280	16
11	SCM 20D [33]	8966	0/61	16
12	OES 10 [33]	403	0/298	16
13	OES 97 [33]	334	0/263	16
14	Soil resilience [9]	26	1/7	8
15	Prespa diatoms lake top 10 [26]	218	0/16	10
16	PPMI [30]	713	0/148	35

D. EXPERIMENTAL SETUP

We designed the experimental setup to answer the experimental questions posed in Section IV-A. Below, we outline the procedures for statistical analysis of the results. Then, we describe all parameter settings of the FIRE-ROS method. Finally, we describe parameter settings for the competing methods, against which we compare FIRE-ROS.

We **estimate the predictive performance** of the considered methods by using 10-fold cross-validation. To produce comparable results, all methods are given same folds to learn models. For **statistical evaluation** of the obtained results, we follow the recommendations of Demšar [11].

The Friedman test [17], with the correction by Iman and Davenport [19], is used to determine statistical significance. To detect statistically significant differences, we calculate the critical distances (CD) by applying two post-hoc statistical tests [13], [31]. Both post-hoc tests compute a critical distance between the ranks of considered algorithms. The difference is that the Nemenyi post-hoc test compares the relative performance of all considered methods (all vs. all), whereas the Bonferroni-Dunn post-hoc test compares the performance of a single method to the other methods (one vs. all). The results of these tests are presented with average rank diagrams [11], where methods connected with a line have results that are not significantly different. All statistical tests were conducted at the significance level $\alpha = 0.05$. The lower the average rank, the better the performance. A CD is drawn with a dotted blue line when the Bonferroni-Dunn post-hoc test is used. Otherwise, when the Nemenyi post-hoc test is used, a CD is drawn with a solid red line.

Here, we describe the FIRE-ROS parameters. We use three types of ensembles for FIRE-ROS: bagging and random forests of PCTs and ensembles of Extra-PCTs. All ensembles consist of 100 PCTs or Extra-PCTs. Random forests and Extra-PCT ensembles use a sampling of the predictive attribute space. The former use $\sqrt{|D|}$ and the latter $|D|$ of the predictive attributes (D) when searching for the best split in a given node. These values are recommended by the authors of the two ensemble methods. Random forests and bagging ensembles use bootstrap replication, whereas ensembles of Extra-PCTs do not (recommended by the authors). The original FIRE algorithm uses only the standard random forest ensembles of PCTs (without the ROS extension).

FIRE-ROS models are generated with four different values for the output space size: $v \in \{1/4, 1/2, 3/4, \text{Random}\}$. When subspace size is selected at random, ROS ensembles use a different subspace size for every PCT in the ensemble. All tree ensembles, used to generate candidate rules, are allowed to grow to the average depth of 3 ($\bar{L}$). For FIRE-ROS, we generate fully-predicting (F) or mixed (M) candidate rules. The former set consists only of rules that always predict all target attributes. The latter also contains partially-predicting rules, which give predictions for only a subset of target attributes. FIRE and FIRE-ROS both use linear terms. We compare FIRE-ROS with several competing MTR methods, as described below.

Predictive clustering rules (PCRs) were generated with the default parameters. We trained unordered rules using the multiplicative variant of the search heuristic. Covering weight was set to 0.1 with the covering weight threshold of 0.1. The maximal number of rules was set to 1000. Target attributes weight was set to 1.0.

When learning single **multi-target regression trees (MTRTs)**, F-test pruning was applied. The threshold for the test was determined by using 3-fold internal cross-validation.

Random projections Random forests (RP-RF) (available at https://github.com/arjoly/random-output-trees) use $m = \log |T|$ components in the projected output space where T is the set of target attributes. Also, Rademacher random projections were used for output space transformations. We used $k = |D|$ randomly chosen input features to calculate splits. The minimal number of allowed instances in a leaf node was set to 1.

The **Random Linear Combinations (RLC)** algorithm (available at http://mulan.sourceforge.net) was parametrized to use gradient boosting with 4-terminal node regression tree as the base regressor with a learning rate of 0.1 and 100 boosting iterations. The number of targets that participate in the random linear combinations was set to $k = 2$.

Ensembles of Extra-PCTs with ROS (ET-ROS) use 100 trees with $k = |D|$ randomly chosen input features to select splits. ROS was initialized with a subspace size of 3/4 and subspace averaging. These are the best-performing PCT ensembles with ROS [6].

V. RESULTS AND DISCUSSION

In this section, we present the results of experimental evaluation of the performance of FIRE-ROS and the competing methods. Results are presented in terms of predictive performance (aRRMSE, overfitting score) and interpretability (model sizes). We begin our presentation of results with an example FIRE-ROS model learned on the PPMI dataset [30]. Then, we present experimental results regarding FIRE-ROS variants with different output space sizes and ensemble methods for generating candidate rules. Next, we compare models induced with FIRE-ROS to the models induced by the state-of-the-art MTR methods. We also examine how FIRE-ROS models overfit and address the interpretability of FIRE-ROS models. The section concludes with a summary of the results.

A. AN EXAMPLE FIRE-ROS MODEL

The example FIRE-ROS model, a part of which is shown in Table 2, was induced on data collected within the Parkinson's Progression Markers Initiative (PPMI). The dataset contains the volumes of brain regions of interest (ROIs) defined from the patients' fMRIs, DaT (dopamine transporter) scans, and motor assessment scores (MDS-UPDRS). The time interval between the date of scoring and the date of fMRI scans is smaller than 6 months. The task is to predict the 35 scores of the motor impairment assessments from the extracted ROI and DaT scan features [30].

Table 2 shows the first 10 of 30 rules contained in the FIRE-ROS model, ordered by their weights. The **average** rule always predicts all target attributes. This rule is needed to make sure that the model will always give a prediction, e.g., when none of the other rules are triggered. The rules, numbered from 1 to 10, can be fully-predicting (rule 5) or partially-predicting (all remaining rules). The NP symbol is an abbreviation for "not predicted". Rules with higher weights contribute more to the overall prediction. When a prediction is to be made for a data instance, one or several rules can trigger. In those cases, the model is additive. The weighted predictions, given by the individual rules, will be added to the prediction of the **average** rule. Regardless of

TABLE 2. An Excerpt From a FIRE-ROS Model Induced on the PPMI Dataset. We Show the Average Prediction and the First 10 Rules (the Model Consists of 30 Rules in Total). Rule no. 5 is Fully-Predicting While the Other Rules are Partially-Predicting. The NP Symbol is an Abbreviation for "Not Predicted". Rules With the NP Symbol at a Given Position in the Target Value Tuple Do Not Predict the Corresponding Target Attribute. The PPMI Dataset has 35 Target Attributes. We Only Show the Values of Eight Target Attributes for Each Rule as well as the Average Prediction

#	Weight	Rule conditions	Prediction								
			NP3SPCH	NP3FACXP	NP3RIGN	NP3RIGRU	NP3RIGLU	PN3RIGRL	NP3RIGLL	...	NHY
average	1	True (always triggered)	(0.42, 0.9, 0.58, 0.96, 0.79, 0.6, 0.51, ..., 1.4)								
1	0.41	Caudate_L > 1.8 AND Putamen_R ≤ 0.95 AND Right_LOrG_lateral_orbital_gyrus ≤ 2.591	(NP, NP, NP, NP, 1.56, −0.35, NP, ..., NP)								
2	0.14	Right_Inf_Lat_Vent > 0.32973 AND Caudate_L ≤ 2.2 AND Right_PP_planum_polare > 2.028	(0.89, 1.23, 0.68, 1.55, NP, NP, NP, ..., NP)								
3	-0.1	4th_Ventricle > 1.0451 AND Putamen_R > 0.98 AND Left_MFC_medial_frontal_cortex > 1.2646	(0.6, NP, NP, NP, 1.74, 0.42, 1.21, ..., 1.12)								
4	0.09	Caudate_L ≤ 2.34 AND Putamen_L ≤ 0.77 AND Left_STG_superior_temporal_gyrus > 5.7389	(NP, 0.86, NP, 1.69, NP, 1.06, −0.07, ..., 0.91)								
5	0.09	Putamen_L ≤ 1.17 AND Right_Pallidum ≤ 1.804 AND Left_TTG_transverse_temporal_gyrus ≤ 1.814	(0.52, 0.85, 1.34, 1.34, 1.57, 0.83, 1.32, ..., 1.43)								
6	-0.08	Left_POrG_posterior_orbital_gyrus > 1.9824 AND Caudate_L > 2.21 AND Right_PHG_parahippocampal_gyrus > 2.7849	(0.85, 1.41, 0.74, 1.68, NP, NP, 0.58, ..., 1.35)								
7	-0.07	4th_Ventricle > 1.0451 AND Putamen_R ≤ 0.98 AND Left_MFC_medial_frontal_cortex > 1.2646	(0.51, NP, NP, NP, 1.69, 0.38, 0.89, ..., 1.38)								
8	0.05	Putamen_L ≤ 1.36 AND 3rd_Ventricle > 0.80176 AND Left_PoG_postcentral_gyrus ≤ 14.656	(0.78, NP, NP, NP, 0.93, 0.62, 0.65, ..., 1.43)								
9	-0.04	Putamen_R > 0.91 AND Left_Pallidum > 1.4025 AND Right_STG_superior_temporal_gyrus > 6.72	(0.98, 1.34, 0.92, 0.71, NP, 0.48, NP, ..., NP)								
10	-0.04	Putamen_R > 0.96 AND Right_LiG_lingual_gyrus > 6.794 AND Right_Cerebral_White_Matter > 196.586	(NP, NP, 0.83, 0.41, 1.74, 0.7, 1.25, ..., NP)								

which rules (if any) are triggered, FIRE-ROS models always give predictions for all target attributes.

There is a clear advantage of using rule-based models in contrast to black-box models. One can analyze how the models make their predictions by examining rule conditions, as well as understand the importance of any given rule by observing its weight. If the discovered rules are too general/specific one can increase/decrease the average depth of PCTs in the ensemble used to generate candidate rules and thus influence the number of conditions in the resulting rules.

B. FIRE-ROS PARAMETER SELECTION

In this section, we analyze the predictive performances of different FIRE-ROS variants. The average rank diagrams in Fig. 2 depict the average predictive performance of models learned by using different variants of FIRE-ROS. We use three different ensemble methods to generate the candidate rules and we analyze each of them separately. We have also experimented with two types of candidate rule sets: with fully-predicting (F) rules and with mixed-predictivity (M) rules (containing both fully-predicting and partially-predicting rules). Finally, we use different subspace sizes with ROS.

We can immediately see that, on average, models induced with mixed candidate rules perform worse than those induced with only fully-predicting rules (Fig. 2). With mixed candidate rules, varying the ROS subspace size (i.e., Random) consistently performs the worst (Fig. 2a, 2b, 2c). Generally, across all three ensemble methods, mixed rules models (M) perform better when larger ROS subspaces are used. In contrast, fully-predicting variants (F) perform best when the subspace size is smaller. In particular, subspace size $1/4$ seems to give the best results. Based on the average rank diagrams in Fig. 2, we recommend using FIRE-ROS with fully-predicting candidate rules generated from ROS tree ensembles with an output space size of $1/4$.

Table 3 shows how many times a particular FIRE-ROS variant outperformed all others. Fully-predicting rules with output space size of $1/4$ have won 27 times. The same output space size with mixed candidate rules never performed best. For larger subspace sizes ($1/2$, $3/4$), the win frequency of fully-predicting rules is on par with mixed rules. Clearly, there are datasets, where models induced with mixed candidate rules perform better. In those cases, the output space size is, for the most part, equal for all three ensemble methods.

C. SIZE RESTRICTIONS ON FIRE-ROS MODELS

In order to make FIRE-ROS models even more interpretable, we have carried out additional experiments, where we restrict the number of rules in a FIRE-ROS model. We have induced

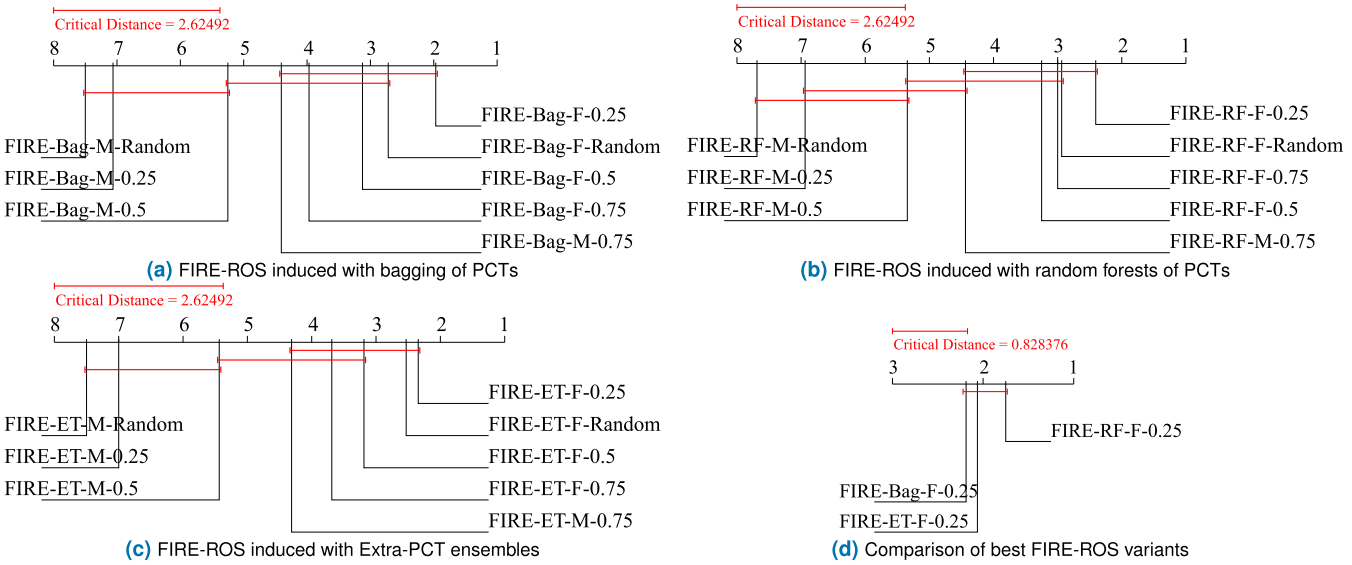


FIGURE 2. Average rank diagrams comparing the predictive performance of the different FIRE-ROS models in terms of aRRMSE. Lower ranks are better. *Bag*, *RF* and *ET* denote the use of bagging, random forests and extra tree ensembles to generate candidate rules, respectively. *M* and *F* stand for mixed and fully-predicting candidate rules respectively. The labels *0.25*, *0.5*, *0.75* and *Random* denote various output space sizes. In our experiments, the best performance of the proposed method is, on average, achieved with fully-predicting rules originating from ROS ensembles that use 1/4 of the target attributes (*0.25* denotes 25 % of all target attributes), regardless of the ensemble method used.

TABLE 3. Summary of Wins for Ensemble Methods (Columns) and ROS Subspace Sizes (Rows). Each Cell Shows Counts of How Many Times a Specific Variant of FIRE-ROS Outperformed Other Variants. The Numbers to the Left and Right of the '/' Separator Denote Fully-Predicting (F) and Mixed (M) Candidate Rule Sets Respectively. In Total, We Have 48 Data Points (16 for Each Considered Ensemble Method–Dataset Pair). For Example, for FIRE-ROS With Bagging, Best Performance is Achieved With Subspace Size 1/4 and Fully-Predicting Rules in 10 of 16 Datasets

Subspace size	Bagging	Random forest	Extra trees	Total
1/4	10/0	8/0	9/0	27/0
1/2	0/2	2/2	1/1	3/5
3/4	1/1	2/1	2/2	5/4
Random	2/0	1/0	1/0	4/0
Total	16 (13/3)	16 (13/3)	16 (13/3)	48

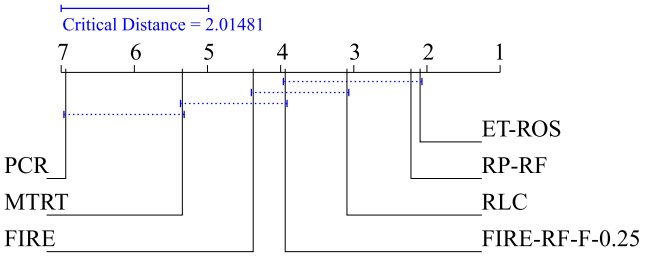


FIGURE 3. Average rank diagram comparing aRRMSE performance of FIRE-ROS with that of competing methods. Lower ranks are better. The proposed approach performs best in the group of interpretable methods. The non-interpretable state-of-the-art MTR methods outperform all interpretable models.

rule sets with a maximal number of 30, 50 and 100 rules. The results are expected and show that models with more rules have better predictive power, which is also in line with the findings of Aho *et al.* [1]. Moreover, models containing only fully-predicting rules always outperform models with included partially-predicting rules.

We also investigated whether the recommended ROS subspace size remains the same as with the full rule ensembles. The best performing subspace size for random forests and Extra-PCTs is 1/4 for all three reduced rule set sizes, which is the same as with the full size rule ensembles. A small change was observed with bagging, where *Random* became the best choice, when restricting the rule set size to 30 or 50 rules.

D. COMPARISON TO EXISTING METHODS

In order to put FIRE-ROS in the broader context of MTR methods, we compare its best variant (we use the recommended parameter settings) to other MTR methods. Fig. 3 shows the obtained results by means of an average rank diagram. The parameters settings for all the used methods are explained in detail in Section IV-D.

FIRE-ROS outperforms PCRs with a statistically significant difference in performance and MTRTs and FIRE (without a statistically significant difference). All state-of-the-art MTR methods (ET-ROS, RP-RF, RLC) outperform all interpretable models. The differences between state-of-the-art MTR methods and FIRE-ROS are not statistically significant. Please refer to Table 4 for details.

E. OVERFITTING

Fig. 4 compares the overfitting scores of interpretable models of different sizes. Smaller models overfit less, which is an expected result. The size-restricted FIRE and FIRE-ROS models are clearly the best-performing

TABLE 4. Predictive Performance (aRRMSE) for FIRE-ROS (first column) and Competing Methods. Lower Values are Better. Bolded Values Denote the Best Performamce in the Group of Interpretable Methods. The Underlined Numbers Denote the Best Performance Method Overall

	Interpretable				Not interpretable		
Dataset	FIRE-RF-F-0.25	PCR	FIRE	MTRT	RLC	ET-ROS	RP-RF
ATP 1D (6T-337)	**0.413**	0.545	0.418	0.515	0.415	0.375	0.436
ATP 7D (6T-296)	**0.523**	0.753	0.536	0.585	0.359	0.437	0.413
Forestry Kras (11T-60.607)	0.633	0.850	0.635	**0.610**	0.610	0.558	0.558
OES 10 (16T-403)	0.433	0.624	**0.425**	0.616	0.447	0.497	0.469
OES 97 (16T-334)	**0.465**	0.812	0.476	0.704	0.513	0.518	0.536
Sales (12T-639)	0.703	0.903	**0.702**	0.867	0.708	0.698	0.646
PPMI (35T-713)	**0.839**	1.002	0.847	0.868	0.767	0.746	0.731
Prespa top 10 (10T-218)	0.991	1.056	**0.981**	0.996	0.811	0.929	0.831
RF 1 (8T-9125)	0.292	0.905	0.325	**0.190**	0.238	0.147	0.141
RF 2 (8T-9125)	0.291	0.968	0.295	**0.200**	0.241	0.153	0.146
SCM 1D (16T-9.893)	**0.384**	0.684	**0.384**	0.433	0.376	0.281	0.311
SCM 20D (16T-8.966)	0.598	0.803	0.603	**0.527**	0.595	0.317	0.366
Soil resilience (8T-26)	0.914	0.916	**0.897**	0.927	0.761	0.874	0.848
Vegetation condition (7T-16.967)	**0.649**	0.736	0.650	0.659	0.647	0.599	0.605
Vegetation clustering (11T-29.679)	**0.773**	0.928	0.775	0.810	0.790	0.696	0.703
Water quality (14T-106)	**0.924**	0.979	**0.924**	0.952	0.910	0.895	0.902

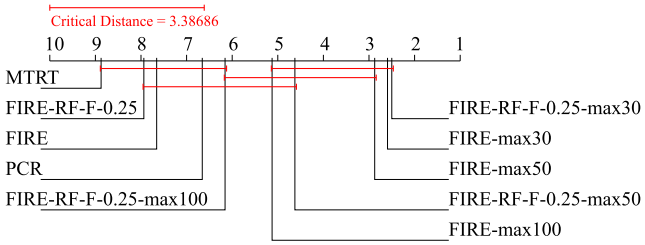


FIGURE 4. Average rank diagram comparing the overfitting score of FIRE-ROS models with those of FIRE, PCRs and MTRTs. Lower ranks are better.

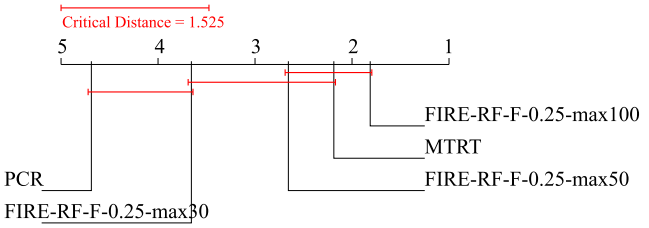


FIGURE 5. Average rank diagram comparing the predictive performance of smaller FIRE-ROS models to PCRs and MTRTs in terms of aRRMSE. Lower ranks are better.

(least overfitting), with difference among them being insignificant.

MTRT models overfit significantly more than the size-restricted rules. If we ignore the size-restricted models, PCR models are least overfitted. The proposed FIRE-ROS models overfit comparably to the models generated with FIRE.

F. INTERPRETABILITY

We compare the model sizes of the interpretable models generated with the following methods: PCRs, MTRTs, FIRE and FIRE-ROS (with the recommended settings from Section V-B). The results show, that the smallest models are produced by PCRs and MTRTs which, however, do not have high predictive power. FIRE and FIRE-ROS generate statistically significantly larger models. We also compare the predictive power of smaller FIRE-ROS models (30, 50 and 100 rules) to that of MTRTs and PCRs.

Figure 5 shows that FIRE-ROS models achieve the best predictive performance while maintaining a small number of rules in the ensemble. FIRE-ROS models with 50 and 100 rules perform significantly better than PCRs. FIRE-ROS models with 100 rules (on average) outperform MTRTs, but without statistically significant differences.

G. SUMMARY OF THE RESULTS

We summarize the main findings of the experimental work presented in the article by answering the experimental questions posed in Section IV-A. leftmargin=1.5em

1) **What is the best value for the ROS output space size to be used with FIRE-ROS ensembles? Are there any differences between the three ensemble methods used with FIRE-ROS? Are there any differences between FIRE-ROS and PCT ensembles with ROS?** The recommended subspace size for the proposed method is 1/4 with fully-predicting rules. Out of the three considered ensemble methods, random forest ensembles perform best within FIRE-ROS, but the differences in performance to bagging and Extra-PCTs are not statistically significant. Some datasets benefit from using mixed candidate rules with larger subspace sizes, i.e., 1/2, 3/4. These results differ from what has been observed with tree ensembles with ROS in [6], where ensembles of Extra-PCTs with ROS subspace size of 3/4 perform best.
2) **Does the use of partially-predicting rules improve the predictive performance of FIRE-ROS?** On average, the best performing rule ensembles use only fully-predicting rules. However, our experiments show,

that ensembles with partially-predicting rules are beneficial for some problems, as they lead to better predictive performance.

3) **How do FIRE-ROS ensembles compare to the original FIRE ensembles in terms of predictive performance?** In terms of predictive performance, the proposed FIRE-ROS ensembles perform better than the original FIRE ensembles, but the differences are not significant.
4) **How do FIRE-ROS models compare to state-of-the-art MTR methods in terms of predictive performance?** All state-of-the-art MTR methods outperform all interpretable models. FIRE-ROS is outperformed without statistically significant differences. FIRE-ROS performs significantly better than PCRs, MTRTs and the original FIRE method.
5) **How does the interpretability influence the predictive power of FIRE-ROS models?** In our experiments, the most interpretable models are MTRTs and PCRs. Both FIRE and FIRE-ROS generate statistically significantly larger models. When we limit the size of the proposed ensembles, FIRE-ROS models with 100 rules, on average, exhibit the best predictive performance (without statistically significant differences in performance). PCRs are outperformed by MTRTs and FIRE-ROS models, mostly with statistically significant differences.

VI. CONCLUSION

We have addressed the task of multi-target regression (MTR), where the goal is to learn predictive models that give predictions for several continuous values simultaneously. We have proposed FIRE-ROS – an extension of the rule learning method FIRE for MTR. We learn rules by inducing an ensemble of predictive clustering trees (PCTs) with random output selections (ROS) and decomposing it into a set of candidate rules. Each rule is assigned a weight. Later, a gradient directed optimization procedure is used to find rule weights that result in a model with the best predictive performance.

Random output selections (ROS) is an ensemble extension, where individual base models of the ensemble are learned to predict the complete target space by considering subsets of it during learning. FIRE-ROS integrates ROS ensembles of PCTs as a replacement for the previously used ensembles of standard PCTs within FIRE. The induced rule sets can now also contain rules which give predictions for a subset of the target attributes (partially-predicting rules).

We performed an experimental evaluation, where we used three ROS-enhanced ensemble methods (bagging and random forests of PCTs, ensembles of Extra-PCTs) to generate sets of candidate rules for FIRE-ROS. Model performance was evaluated on several benchmark datasets of varying sizes in terms of the number of examples, the number of predictive attributes and the number of target attributes.

Our results suggest, that FIRE-ROS models can outperform FIRE models, as well as other competing methods that generate interpretable models, namely predictive clustering rules (PCRs) and multi-target regression trees (MTRTs). The best-performing FIRE-ROS variant uses fully-predicting rules generated with a ROS-enhanced random forest ensemble of PCTs with an output space size of 1/4. We have also provided a comparison with state-of-the-art MTR methods, where we show that they, to no surprise, outperform the proposed approach, but without statistically significant differences. This is a common occurrence, stemming from the trade-off between interpretability and predictive power.

A major concern when learning predictive models is overfitting. In our analysis, we monitored overfitting by calculating the overfitting score for the interpretable models. The results show that the proposed FIRE-ROS models overfit comparably to their original FIRE counterparts and less than multi-target regression trees. We also note that the size-restricted models are the least overfitted.

Future work is planned along several directions. A concern of the current approach is the computational complexity of the optimization procedure. A faster optimization method needs to be used. Along this line, a heuristic can be used to discard similar rules according to their prediction vectors [38]. We believe that this reduction of the number of candidate rules will shrink the search space without degrading the final rule model performance. Moreover, we plan to define the loss function in such a way, that the usage of mixed candidate rules will not deteriorate overall predictive performance but rather eliminate interfering candidate rules. The first step towards this would be to introduce the concept of target importances. There are use cases, where not all targets are equally important. We plan on introducing weights into our multi-target loss function to guide the optimization towards more important targets. Next, we have made overall recommendations for the ROS parameters, based on the average rank diagrams. However, the best performing parameters seem to be domain-specific. A meta-learning approach that would select the parameters prior to learning would likely yield better results. We would also like to transfer the proposed method to the semi-supervised setting by using ensembles of semi-supervised PCTs [28] to generate candidate rules. Finally, we would like to extend this approach to other structured output prediction tasks, such as (hierarchical) MLC.

ACKNOWLEDGMENT

Some of the data used in the preparation of this article were obtained from the Parkinson's Progression Markers Initiative (PPMI) database (http://www.ppmi-info.org/data). For up-to-date information on the study, visit http://www.ppmi-info.org.

PPMI – a public-private partnership – is funded by the Michael J. Fox Foundation for Parkinson's Research and funding partners, including Abbvie, Allergan, Amathus Therapeutics, Avid Radiopharmaceuticals, Biogen, BioLegend, Bristol-Myers Squibb, Celgene, Denali Therapeutics, GE Healthcare, Genentech, GlaxoSmithKline,

Janssen Neuroscience, Lilly, Lundbeck, Merck, Meso Scale Discovery, Pfizer, Piramal, Prevail Therapeutics, Roche, Sanofi Genzyme, Servier, Takeda, Teva, UCB, Verily and Voyager therapeutics.

The computational experiments presented here were executed on the computing infrastructure of the Slovenian Grid (SLING) initiative.

We would like to thank Dragi Kocev for reading the initial draft of the article and providing comments.

REFERENCES

[1] T. Aho, B. Ženko, S. Džeroski, and T. Elomaa, "Multi-target regression with rule ensembles," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 2367–2407, 2012.

[2] H. Blockeel, S. Džeroski, and J. Grbović, "Simultaneous prediction of multiple chemical parameters of river water quality with TILDE," in *Proc. 3rd Eur. Conf. Princ. Pract. Knowl. Discovery Databases* (Lecture Notes in Artificial Intelligence), vol. 1704. Berlin, Germany: Springer, 1999, pp. 32–40.

[3] H. Blockeel, L. D. Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proc. 15th Int. Conf. Mach. Learn.* San Mateo, CA, USA: Morgan Kaufmann, 1998, pp. 55–63.

[4] H. Blockeel and J. Struyf, "Efficient algorithms for decision tree cross-validation," *J. Mach. Learn. Res.*, vol. 3, pp. 621–650, Dec. 2002.

[5] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 5, no. 5, pp. 216–233, Sep. 2015.

[6] M. Breskvar, D. Kocev, and S. Džeroski, "Ensembles for multi-target regression with random output selections," *Mach. Learn.*, vol. 107, no. 11, pp. 1673–1709, Nov. 2018.

[7] M. Breskvar, D. Kocev, and S. Džeroski, "Fitted rule ensembles for multi-target regression with random output selections," in *Proc. ETAI, Proc. Abstr. 15th Int. Conf.*, 2018, p. 25.

[8] M. Breskvar, D. Kocev, J. Levatic, A. Osojnik, M. Petkovic, N. Simidjievski, B. Zenko, R. Boumghar, and L. Lucas, "Predicting thermal power consumption of the Mars express satellite with machine learning," in *Proc. 6th Int. Conf. Space Mission Challenges Inf. Technol. (SMC-IT)*, Sep. 2017, pp. 88–93.

[9] M. Debeljak, D. Kocev, W. Towers, M. Jones, B. S. Griffiths, and P. D. Hallett, "Potential of multi-objective models for risk-based mapping of the resilience characteristics of soils: Demonstration at a national level," *Soil Use Manage.*, vol. 25, no. 1, pp. 66–77, Mar. 2009.

[10] D. Demnar, 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," *Ecol. Model.*, vol. 191, no. 1, pp. 131–143, Jan. 2006.

[11] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[12] J. Duarte and J. Gama, "Multi-target regression from high-speed data streams with adaptive model rules," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2015, pp. 1–10.

[13] O. J. Dunn, "Multiple comparisons among means," *J. Amer. Stat. Assoc.*, vol. 56, no. 293, pp. 52–64, Mar. 1961.

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

[15] S. Džeroski, A. Kobler, V. Gjorgjioski, and P. Panov "Using decision trees to predict forest stand height and canopy cover from LANDSAT and LIDAR data," in *Proc. 20th Int. Conf. Inform. Environ. Protection*. Herzogenrath, Germany: Shaker Verlag, 2006, pp. 125–133.

[16] J. H. Friedman and B. E. Popescu, "Predictive learning via rule ensembles," *Ann. Appl. Statist.*, vol. 2, no. 3, pp. 916–954, Sep. 2008.

[17] M. Friedman, "A comparison of alternative tests of significance for the problem of *m* rankings," *Ann. Math. Statist.*, vol. 11, no. 1, pp. 86–92, Mar. 1940.

[18] V. Gjorgjioski, S. Džeroski, and M. White, "Clustering analysis of vegetation data," Jožef Stefan Inst., Ljubljana, Slovenia, Tech. Rep. 10065, 2008.

[19] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the Fbietkan statistic," *Commun. Statist.-Theory Methods*, vol. 9, no. 6, pp. 571–595, Jan. 1980.

[20] S. Jancic, J. C. Frisvad, D. Kocev, C. Gostincar, S. Džeroski, and N. Gunde-Cimerman, "Production of secondary metabolites in extreme environments: Food- and airborne wallemia spp. Produce toxic metabolites at hypersaline conditions," *PLoS ONE*, vol. 11, no. 12, Dec. 2016, Art. no. e0169116.

[21] A. Joly, "Exploiting random projections and sparsity with random forests and gradient boosting methods—Application to multi-label and multi-output learning, random forest model compression and leveraging input sparsity," 2017, *arXiv:1704.08067*. [Online]. Available: http://arxiv.org/abs/1704.08067

[22] A. Joly, P. Geurts, and L. Wehenkel, "Random forests with random projections of the output space for high dimensional multi-label classification," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2014, pp. 607–622.

[23] Kaggle. (2008). *Kaggle Competition: Online Product Sales*. Accessed: Jul. 19, 2017. [Online]. Available: https://www.kaggle.com/c/online-sales/data

[24] D. Kocev and M. Ceci, "Ensembles of extremely randomized trees for multi-target regression," in *Proc. 18th Int. Conf. Discovery Sci.*, in Lecture Notes in Computer Science, vol. 9356. Cham, Switzerland: Springer, 2015, pp. 86–100.

[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," *Ecol. Model.*, vol. 220, no. 8, pp. 1159–1168, Apr. 2009.

[26] D. Kocev, A. Naumoski, K. Mitreski, S. Krstić, and S. Džeroski, "Learning habitat models for the diatom community in lake prespa," *Ecol. Model.*, vol. 221, no. 2, pp. 330–337, Jan. 2010.

[27] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, "Tree ensembles for predicting structured outputs," *Pattern Recognit.*, vol. 46, no. 3, pp. 817–833, Mar. 2013.

[28] J. Levatić, M. Ceci, D. Kocev, and S. Džeroski, "Semi-supervised classification trees," *J. Intell. Inf. Syst.*, vol. 49, no. 3, pp. 461–486, Dec. 2017.

[29] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognit.*, vol. 45, no. 9, pp. 3084–3104, Sep. 2012.

[30] K. Marek, D. Jennings, S. Lasch, A. Siderowf, C. Tanner, T. Simuni, C. Coffey, K. Kieburtz, E. Flagg, and S. Chowdhury, "The Parkinson progression marker initiative (PPMI)," *Prog. Neurobiol.*, vol. 95, no. 4, pp. 629–635, 2011.

[31] P. B. Nemenyi, "Distribution-free multiple comparisons," Ph.D. dissertation, Dept. Math., Princeton Univ., Princeton, NY, USA, 1963.

[32] M. Petkovic, N. Simidjievski, R. Boumghar, M. Breskvar, S. Dzeroski, D. Kocev, J. Levatic, L. Lucas, A. Osojnik, and B. Zenko, "Machine learning for predicting thermal power consumption of the Mars express spacecraft," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 34, no. 7, pp. 46–60, Jul. 2019.

[33] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, and I. Vlahavas, "Multi-target regression via input space expansion: Treating targets as inputs," *Mach. Learn.*, vol. 104, no. 1, pp. 55–98, Jul. 2016.

[34] J. Struyf and S. Džeroski, "Constraint based induction of multi-objective regression trees," in *Proc. 4th Int. Workshop Knowl. Discovery Inductive Databases KDID*, in Lecture Notes in Computer Science, vol. 3933. Berlin, Germany: Springer, 2006, pp. 222–233.

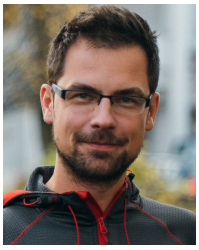
[35] P. Szymanski, T. Kajdanowicz, and K. Kersting, "How is a data-driven approach better than random choice in label space division for multi-label classification?" *Entropy*, vol. 18, no. 8, p. 282, Jul. 2016.

[36] G. Tsoumakas, E. Spyromitros-Xioufis, A. Vrekou, and I. Vlahavas, "Multi-target regression via random linear target combinations," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, in Lecture Notes in Computer Science, vol. 8726. Berlin, Germany: Springer, 2014, pp. 225–240.

[37] G. Tsoumakas and I. Vlahavas, "Random K-labelsets: An ensemble method for multilabel classification," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2007, pp. 406–417.

[38] B. Ženko, "Learning predictive clustering rules," Ph.D. dissertation, Fac. Comput. Sci., Univ. Ljubljana, Ljubljana, Slovenia, 2007.

[39] J. R. Zilke, E. L. Mencía, and F. Janssen, "Deepred–rule extraction from deep neural networks," in *Proc. 19th Int. Conf. Discovery Sci.* Cham, Switzerland: Springer, 2016, pp. 457–473.



MARTIN BRESKVAR received the Ph.D. degree from the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia. He is currently a Postdoctoral Researcher with the Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana. He has been involved in the EU-funded FP7 project MAESTRA (Learning from Massive, Incompletely Annotated, and Structured Data) and FP7 and H2020 project HBP (The Human Brain Project). His research interests include development and application of machine learning methods for predicting structured outputs, recommender systems, and knowledge graphs.

SAŠO DŽEROSKI received the Ph.D. degree in computer science from the University of Ljubljana, Slovenia, in 1995. He is currently a Senior Researcher with the Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, and also a Professor with the Jožef Stefan International Postgraduate School, Ljubljana. His research interests include artificial intelligence (AI), focusing on the development of data mining and machine learning methods for a variety of tasks - including the prediction of structured outputs and the automated modeling of dynamic systems - and their applications to practical problems from science and engineering, e.g., environmental sciences (ecology) and life sciences (biomedicine). In 2008, he was an Elected Fellow of the European AI Society for his ''Pioneering Work in the field of AI and Outstanding Service for the European AI community''. In 2015, he became a Foreign Member of the Macedonian Academy of Sciences and Arts. In 2016, he was elected a member of Academia Europaea.