Machine Learning (2020) 109:1179–1204
https://doi.org/10.1007/s10994-019-05829-8

Feature ranking for multi-target regression

Matej Petković[1,2] · Dragi Kocev[1,2] · Sašo Džeroski[1,2]

Received: 1 July 2018 / Revised: 12 June 2019 / Accepted: 6 July 2019 / Published online: 27 August 2019

Abstract
In this work, we address the task of feature ranking for multi-target regression (MTR). The task of MTR concerns problems with multiple continuous dependent/target variables, where the goal is to learn a model for predicting all of them simultaneously. This task is receiving an increasing attention from the research community, but performing feature ranking in the context of MTR has not been studied thus far. Here, we study two groups of feature ranking scores for MTR: scores (Symbolic, Genie3 and Random Forest score) based on ensembles (bagging, random forests, extra trees) of predictive clustering trees, and a score derived as an extension of the RReliefF method. We also propose a generic data-transformation approach to MTR feature ranking and thus have two versions of each score. For both groups of feature ranking scores, we analyze their theoretical computational complexity. For the extension of the RReliefF method, we additionally derive some theoretical properties of the scores. Next, we extensively evaluate the scores on 24 benchmark MTR datasets, in terms of the quality of the ranking and the computational complexity of producing it. The results identify the parameters that influence the quality of the rankings, reveal that both groups of methods produce relevant feature rankings, and show that the Symbolic and Genie3 score, coupled with random forest ensembles, yield the best rankings.

Keywords Feature ranking · Multi target regression · Tree based methods · Relief

1 Introduction

Single target regression (STR) is the predictive modeling task of learning a model able to predict the values of a single numeric target variable. STR can be generalized to multi-target

Editors: Takuya Kida, Takeaki Uno, Tetsuji Kuboyama, Akihiro Yamamoto.

✉ Matej Petković
matej.petkovic@ijs.si

Dragi Kocev
dragi.kocev@ijs.si

Sašo Džeroski
saso.dzeroski@ijs.si

1 Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

2 Jožef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia

regression (MTR), where the goal is to learn a model that predicts several (at least two) target variables simultaneously. While STR is a well established research topic, MTR is only recently attracting interest in the research community (Kocev et al. 2013; Spyromitros-Xioufis et al. 2016; Borchani et al. 2015). MTR is a structured output prediction task with applications in a wide range of real life problems.

Prominent examples for MTR come from ecology and include predicting the abundance of different species sharing the same habitat (Džeroski et al. 2000), predicting forest properties (Kocev et al. 2009), chemometrics to infer concentrations of several analytes from multivariate calibration using multivariate spectral data (Burnham et al. 1999), real-time prediction of multiple gas tank levels of the Linz Donawitz converter gas system (Han et al. 2012), simultaneous estimation of different biophysical parameters from remote sensing images (Tuia et al. 2011), channel estimation through the prediction of several received signals (Sanchez-Fernandez et al. 2004) etc. Many other applications can be found in Sect. 4.2, where the data used in our experiments are described.

A possible way to approach a MTR problem is problem transformation, which transforms one MTR problem into several STR problems and builds one predictive model for each target separately. Another way to approach the problem is by algorithm adaptation, i.e., to adapt STR methods to handle several targets simultaneously. For example, regression trees can be generalized so that the heuristic function used to select splits considers the multiple targets and the leaves make predictions for all targets. By building a single model, we benefit regarding time-complexity, and also exploit the potential relations between the multiple targets: this results in more interpretable and compact models, as demonstrated when predicting communities of different species (Kocev and Džeroski 2013). For an overview of MTR methods, we refer the reader to Borchani et al. (2015).

The algorithm adaptation approaches, e.g., predictive clustering trees (PCTs) (Blockeel 1998), typically use search heuristics for MTR which aggregate those for different STR problems. As described later in Sect. 2.1, the most commonly used aggregation is the simple average. However, if we happen to have some background knowledge about the MTR problem, PCTs allow for incorporating this knowledge into the tree induction process via different weights for different problems.

Another important task in machine learning - which is the main topic of this work - is feature ranking, which is typically seen as a data preprocessing step. By using some scoring function, the scores $importance(x_i)$ of descriptive attributes (features) x_i are estimated and an ordering (ranking) of the features is made, based on their estimated importances. There are two main reasons for doing this. First, we may want to reduce the dimensionality of the input space, so that only the features that contain the most information about the target(s) are kept in the dataset. By doing this, we decrease the amount of memory/time needed to build a predictive model, while the performance of the model is not degraded. Particularly, when the dimensionality of the input data is extremely high, the performance of the model can also increase with the dimensionality reduction, since most of the original features are expected to be noisy. Second, dimensionality reduction typically results in models that are easier to understand, which is useful when a machine learning expert works in collaboration with a domain expert. Predictive models, such as decision trees, are easier to interpret when a small number of relevant features are used to learn them.

There is a plethora of feature ranking methods for the machine learning tasks of single target regression and classification (Stańczyk and Jain 2015). However, in the case of MTR, the task of feature ranking has not been studied to a great extent. To the best of our knowledge, our earlier work (Petković et al. 2017) is among the first in that direction.

In the field of statistics, few feature ranking methods can be found. In contrast to the proposed feature ranking methods that can handle both numeric and nominal features, the main drawback of these methods is that they allow only for numeric features, since they typically assume a (generalized) linear model that describes the dependence of the targets $\boldsymbol{y}$ on the features $\boldsymbol{x}$. One such method is forward selection (Brobbey 2015). It starts with a constant mapping $\boldsymbol{y} = c \in \mathbb{R}$, and repeatedly adds the most significant feature that improves the model. The sooner a feature is included in the model, the greater its importance is.

In this work, we propose two groups of feature ranking scores. The first group is based on ensembles of PCTs, which are a generalization of decision trees able to handle various types of structured output prediction tasks, including MTR. The proposed scores exploit different properties of the ensemble learning mechanism to estimate feature importances. The second group contains the score obtained by generalizing RReliefF (Robnik-šikonja and Kononenko 2003) towards the task of MTR. The key observation on which this score is based is that the distance function on the target space used in RReliefF can be generalized to more complex spaces than $\mathbb{R}$. We refer to these scores as MTR scores.

Another approach to feature ranking for MTR is to perform data transformation in the same way as in binary relevance for multi-label classification (Tsoumakas and Katakis 2007), where a separate classifier is learned for each label. We employ the same strategy: we calculate separate importance scores for each target variable, and then aggregate the feature importance scores into a single score. We thus define an analogue for every MTR score and refer to these scores as $\overline{\text{STR}}$ scores. The $\overline{\text{STR}}$ scores, e.g., those based on RReliefF, have already been shown to have state-of-the-art performance in the case of STR problems. Hence, they can serve as an additional baseline in the evaluation.

The ensemble-based feature ranking methods belong to the class of embedded methods, where the feature importance estimation is embedded in the decision model. In contrast, the RReliefF-based feature ranking methods belong to the group of filter methods (Guyon and Elisseeff 2003). The main difference between the two groups is that filter methods do not need to construct any decision/predictive model in the process of feature importance estimation. A third group of related methods are wrapper methods, which typically solve the problem of feature selection directly, i.e., try to find the most useful set of features (for some predictive model) without constructing a feature ranking.

An initial investigation of the proposed feature ranking methods for MTR based on ensemble methods has been presented by Petković et al. (2017). We extend that work along several major dimensions:

1. We propose an extension of RReliefF for MTR and a $\overline{\text{STR}}$ version of it, giving
    (a) Theoretical and empirical computational complexity analysis;
    (b) Discussion of the theoretical properties of the scores;
    (c) Parametrization of the methods and empirical performance evaluation;
2. We present a theoretical and an empirical computational complexity analysis for the MTR rankings derived by the ensemble-based scores;
3. We analyze the performance of $\overline{\text{STR}}$ rankings based on ensemble scores, giving
    (a) Theoretical and empirical computational complexity analysis;
    (b) Empirical performance evaluation;
    (c) Comparison of the MTR and $\overline{\text{STR}}$ rankings based on ensemble scores in terms of time complexity and performance;

Table 1 Notation in the paper

Symbol	Meaning
x_i, y_j	i-th feature and j-th target
D, T	Numbers of features and targets
$\mathcal{X}_i, \mathcal{Y}_j$	The domains of the i-th feature and the j-th target
$\mathcal{X} \subseteq \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$	Descriptive domain
$\mathcal{Y} \subseteq \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_T$	Target domain
$\boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}$	Descriptive and target part of an example
$\mathscr{D} \subseteq \mathcal{X} \times \mathcal{Y}$	A dataset
$\mathscr{D}_{\text{TRAIN}}, \mathscr{D}_{\text{TEST}} \subseteq \mathscr{D}$	Training and testing part of dataset
$M = \lvert\mathscr{D}_{\text{TRAIN}}\rvert$	Number of examples in the training set

4. We give a comparison of ensemble-based and Relief-based scores ($\overline{\text{STR}}$ and MTR rankings) in terms of time complexity and performance.

The notation used throughout the paper is given in Table 1. The rest of the paper is organized as follows. In Sect. 2.1, we define the ensemble-based scoring functions. Next, in Sect. 2.2, we introduce MTR-Relief and present some novel theoretical results that apply also to the standard RReliefF. We then describe the $\overline{\text{STR}}$ approaches. Furthermore, we perform computational complexity analysis of the methods in Sect. 3. Next, in Sect. 4, the experimental design for our extensive evaluation of the proposed methods on benchmark MTR datasets is presented and the obtained results are discussed in Sect. 5. Finally, we conclude in Sect. 6, where we also provide some directions for further work.

2 Methods

In this section, we describe the proposed methods and the necessary background. We start with the ensemble-based methods (Sect. 2.1), continue with MTR-Relief (Sect. 2.2), and finish with the generic $\overline{\text{STR}}$ approach (Sect. 2.3). The PCT framework, as well as MTR-Relief is implemented in the CLUS system (available at http://source.ijs.si/ktclus/clus-public/) (Table 2).

2.1 Ensemble based methods

2.1.1 Predictive clustering trees and ensembles thereof

The PCT framework views a decision tree as a hierarchy of clusters: the root of a PCT corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The leaves represent the clusters at the lowest level of the hierarchy and each leaf is labeled with its cluster's prototype (prediction). PCTs generalize decision trees and can be used for a variety of learning tasks, including clustering and different types of structured output prediction tasks, e.g., multi-target regression, multi-label classification, hierarchical multi-label classification, time series prediction etc. (Blockeel 1998; Kocev et al. 2013). The generalization is based on appropriately adapting the heuristic for inducing PCTs and the prototype function to the given structured output prediction task.

Table 2 The proposed groups of multi-target regression feature ranking scores

Symbolic score, computed from random forests, bagging and extra trees ensemble:
Uses only the depth(s) of nodes at which a feature appears in the trees in the ensemble
Genie3 score, computed from random forests, bagging and extra trees ensemble:
Uses the values of the variance reduction function at the nodes at which a feature appears in the trees in the ensemble
Random Forest score, computed from random forests and bagging ensemble:
Permutes the values of each feature and measures the resulting reduction in performance of the ensemble on out-of-bag examples
MTR-Relief score, an extension of RReliefF, builds no predictive model:
Examines the similarities in feature and output values for randomly selected instances and their neighbours

The first group consists of the Symbolic, Genie3 and Random Forest scores, and the second group consists of MTR-Relief score. For every score, we propose two versions: MTR and $\overline{\text{STR}}$

PCTs are induced with the standard top-down induction of decision trees algorithm (Breiman et al. 1984) presented in Algorithm 1. It takes as input a set of examples E and outputs a tree. The heuristic h that is used for selecting the best test at a node is the weighted impurity of the subgroups of instances of the partitions (lines 3 and 4), induced by the tests. By minimizing it (line 5 of the Algorithm 2), the algorithm is guided towards small trees with good predictive performance. If there are no candidate tests, a leaf is created and the prototype of the instances belonging to that leaf is computed. The main difference between the algorithm for learning PCTs and other algorithms for learning decision trees is that the former considers the impurity function and the prototype function (that computes predictions in leaves) as parameters that can be instantiated for a given learning task.

Algorithm 1 PCT(E)

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

Algorithm 2 BestTest(E)

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

In this work, we focus on the task of MTR and define the impurity function using the variances of the target variables. First, we denote the variance of the target y_j over subset of examples $E \subseteq \mathscr{D}_{\text{TRAIN}}$ as $Var_j(E)$. We then compute the weights $w_j = Var_j(\mathscr{D}_{\text{TRAIN}})$ and use them as normalization factors in the definition of the impurity function:

$$impu(E) = \frac{1}{T}\sum_{j=1}^{T}\frac{1}{w_j}Var_j(E).$$

In a leaf L, the prototype function returns a vector with the average values of the target variables calculated for the examples belonging to L.

Next, we use three types of ensembles to calculate feature importance scores (i.e., feature rankings). These ensembles have PCTs as their base predictive models (Kocev et al. 2013). An ensemble is a set of base predictive models constructed with a given algorithm. The prediction for each new example is made by combining the predictions of the models from the ensemble. In regression tasks, this is typically achieved by taking the average of the predictions of the base models.

A necessary condition for an ensemble to be more accurate than any of its individual members, is that the members are accurate and diverse models (Hansen and Salamon 1990). There are several ways to introduce diversity among the PCTs in an ensemble. We describe and make use of three of them.

Random forests (RF) and bagging Instead of being learned from the original dataset $\mathscr{D}_{\text{TRAIN}}$, each tree in the ensemble is built from a different bootstrap replicate $\mathcal{B}$ of the dataset $\mathscr{D}_{\text{TRAIN}}$, called bag. The examples $\mathscr{D}_{\text{TRAIN}} \setminus \mathcal{B}$ are called out-of-bag examples (OOB). Additionally, the line 2 of the *BestTest* procedure (see Algorithm 2) is modified to change the feature set during learning by introducing randomization in the test selection. More precisely, at each node in a decision tree, a random subset of the input attributes is taken, and the best test is selected from the splits defined by these attributes. The number of attributes that are retained is given as a function of the total number of descriptive attributes D, e.g., $\lceil\sqrt{D}\rceil$, $\lceil\log_2(D)\rceil$, etc. In the special case when we keep all of the attributes, we obtain the bagging procedure.

Extra trees ensembles (ET) Here, at each node all attributes are considered (as in bagging), but we do not evaluate all tests that the attributes yield. Rather, we choose randomly only one per attribute. Among these D tests, we choose the best one. From the bias-variance point of view, the rationale behind the Extra-Trees method is that the explicit randomization of the cut-point and attribute combined with ensemble averaging should be able to reduce variance more strongly than the weaker randomization schemes used by other methods (Geurts et al. 2006). Note that ET uses the initial dataset $\mathscr{D}_{\text{TRAIN}}$ for learning the base predictive models and does not make bootstrap replicates.

2.1.2 Ensemble based scores

Feature ranking is obtained by exploiting the ensemble structure of the learning algorithm. Due to its simplicity, we first describe the *Symbolic score*. Then, we discuss the *Genie3* (Huynh-Thu et al. 2010) and the *Random Forest score*[1] (Breiman 2001).

In the following, we denote a tree as $\mathcal{T}$, whereas $\mathscr{N} \in \mathcal{T}$ denotes a node. Trees form a forest $\mathcal{F}$. Its size (the number of trees in the forest) is denoted as $|\mathcal{F}|$. The set of all internal nodes of a tree $\mathcal{T}$ in which the attribute x_i appears as part of a test is denoted as $\mathcal{T}(x_i)$.

[1] To prevent any confusion, Random Forest score will be always in singular form and capitalized, whereas the ensemble method random forests will be in plural form and not capitalized.

Symbolic score Let $d(\mathcal{N})$ denote the *depth* of $\mathcal{N} \in \mathcal{T}$: if $\mathcal{N}$ is the root of $\mathcal{T}$, then $d(\mathcal{N}) = 0$. Otherwise, $d(\mathcal{N}) = 1 + d(\text{parent}(\mathcal{N}))$. In the simplest version of the score, we count how many times a given attribute occurs in the tests in the internal nodes of the trees. Since the attributes that appear closer to the root are intuitively more important than those that appear deeper in the trees, we introduce the parameter $w \in (0, 1]$ and define the importance of the feature x_i as

$$importance_{\mathrm{SYMB}}(x_i) = \frac{1}{|\mathcal{F}|} \sum_{\mathcal{T} \in \mathcal{F}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} w^{d(\mathcal{N})}. \tag{1}$$

The symbolic score can be computed from all three types of ensembles that we use. Its simplest version corresponds to setting w to 1.

Genie3 score The main motivation for the Genie3 ranking is that splitting the current subset $E \subseteq \mathcal{D}_{\mathrm{TRAIN}}$, according to a test where an important attribute appears, should result in high impurity reduction. The Genie3 importance of the attribute x_i is thus defined as

$$importance_{\mathrm{GENIE3}}(x_i) = \frac{1}{|\mathcal{F}|} \sum_{\mathcal{T} \in \mathcal{F}} \sum_{\mathcal{N} \in \mathcal{T}(x_i)} h^*(\mathcal{N}),$$

where $h^*(\mathcal{N})$ is the value of the variance reduction function described in the *BestTest* procedure. Since h^* is proportional to $|E|$, greater emphasis is again put on the attributes higher in the tree, where $|E|$ is larger. The Genie3 score is applicable to all three ensemble methods that we use.

Random Forest (RF) score This feature ranking method tests how much the noise in a given feature decreases the predictive performance of the trees in the forest. The greater the performance degradation, the more important the feature is. This score uses the internal out-of-bag estimates of the error, therefore it cannot be used with ensembles of ET, where all trees are learned on the whole dataset.

Once a tree $\mathcal{T}$ is grown, the algorithm evaluates the performance of the tree by using the corresponding $\mathrm{OOB}_{\mathcal{T}}$ examples. This results in the predictive error $Err(\mathrm{OOB}_{\mathcal{T}}) \geq 0$, where we assume that lower error value corresponds to better predictions. To assess the importance of the feature x_i for the tree $\mathcal{T}$, we randomly permute its values in the set $\mathrm{OOB}_{\mathcal{T}}$ and obtain the set $\mathrm{OOB}^i_{\mathcal{T}}$. Then, the error $Err(\mathrm{OOB}^i_{\mathcal{T}})$ is computed and the importance of the feature x_i for the tree $\mathcal{T}$ is defined as the relative increase of error after noising. The final Random Forest score of the feature is the average of these values across all trees in the forest:

$$importance_{\mathrm{RF}}(x_i) = \frac{1}{|\mathcal{F}|} \sum_{\mathcal{T} \in \mathcal{F}} \frac{Err(\mathrm{OOB}^i_{\mathcal{T}}) - Err(\mathrm{OOB}_{\mathcal{T}})}{Err(\mathrm{OOB}_{\mathcal{T}})}.$$

2.2 MTR-Relief

In this section, we describe the proposed extension of the RReliefF method towards MTR (dubbed MTR-Relief). We provide technical preliminaries, give the description of the score and discuss some theoretical properties of RReliefF.

2.2.1 MTR-Relief definition

The first member of the RELIEF family of feature ranking algorithms is Relief, which can handle only binary targets (Kira and Rendell 1992). ReliefF and RReliefF extend the RELIEF family of algorithms for the tasks of multi-class classification (handling nominal target variables) and regression (handling numeric target variables), respectively (Robnik-šikonja and Kononenko 2003).

All methods of the RELIEF family assign to each feature x_i a weight w_i that is a measure of its importance. The expected value of w_i has a nice probabilistic interpretation in the case when both the target and x_i are nominal (Robnik-šikonja and Kononenko 2003): simplified to some extent, we have the relation $\mathbb{E}[w_i] = p_1 - p_2$ where

$$p_1 = P(\text{ different value of } x_i \mid \text{different target value}) \tag{2}$$

$$p_2 = P(\text{different value of } x_i \mid \text{same target value}) \tag{3}$$

However, in the case of regression, this relation serves only as a motivation. For a fixed x_i, we first define the events `diffFeat/sameFeat` (two instances have different/same value of x_i) and `diffTarget/sameTarget` (two instances have different/same target value), and the associated probabilities $P_{\texttt{event}} = P(\texttt{event})$ and $P_{\texttt{event1, event2}} = P(\texttt{event1} \wedge \texttt{event2})$. After applying the Bayes rule to Eqs. (2) and (3), we obtain

$$\mathbb{E}[w_i] = \frac{P_{\text{diffFeat, diffTarget}}}{P_{\text{diffTarget}}} - \frac{P_{\text{diffFeat}} - P_{\text{diffFeat, diffTarget}}}{1 - P_{\text{diffTarget}}} \tag{4}$$

where the probabilities are modeled as distances

$$P_{\text{diffFeat}} \approx d_i, \;\; P_{\text{diffTarget}} \approx d_{\mathcal{Y}} \text{ and } P_{\text{diffFeat, diffTarget}} \approx d_i d_{\mathcal{Y}}.$$

By adapting the distance $d_{\mathcal{Y}}$, we can extend the Relief algorithm to address different machine learning tasks. In our particular case of MTR, the distances are defined as follows. To be consistent with the RReliefF algorithm, we set

$$d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) = \begin{cases} \mathbf{1}[\boldsymbol{x}_i^1 \neq \boldsymbol{x}_i^2] & : \mathcal{X}_i \text{ nominal} \\ \frac{|\boldsymbol{x}_i^1 - \boldsymbol{x}_i^2|}{\max_{\boldsymbol{x}} \boldsymbol{x}_i - \min_{\boldsymbol{x}} \boldsymbol{x}_i} & : \mathcal{X}_i \subseteq \mathbb{R} \end{cases}, \tag{5}$$

where max and min go over the known examples $\boldsymbol{x}$. The analogous approach is taken for the metrics d_j on the sets $\mathcal{Y}_j$, $1 \leq j \leq T$, but here only the numeric part of Eq. (5) applies. The distances on the descriptive domain $\mathcal{X}$ and the target domain $\mathcal{Y}$ between $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, and $\boldsymbol{y}^1 = \boldsymbol{y}(\boldsymbol{x}^1)$ and $\boldsymbol{y}^2 = \boldsymbol{y}(\boldsymbol{x}^2)$, respectively, are defined as

$$d_{\mathcal{X}}(\boldsymbol{x}^1, \boldsymbol{x}^2) = \frac{1}{D}\sum_{i=1}^{D} d_i(\boldsymbol{x}^1, \boldsymbol{x}^2) \tag{6}$$

$$d_{\mathcal{Y}}(\boldsymbol{y}^1, \boldsymbol{y}^2) = \frac{1}{T}\sum_{j=1}^{T} d_j(\boldsymbol{y}^1, \boldsymbol{y}^2) \tag{7}$$

In Eq. (7), we use the ℓ_1 metric to be consistent with the RReliefF definition from Eq. (6), thus making RReliefF a special instance of MTR-Relief when $T = 1$.

The calculation of the weights $w_i = \mathit{importance}(x_i)$ using MTR-Relief is outlined in Algorithm 3, where we generalize the pseudocode of the RReliefF algorithm (Robnik-šikonja and Kononenko 2003). For each of the m iterations, we randomly select an example $\boldsymbol{r}$ from $\mathscr{D}$ (line 4) and find its k nearest neighbors (line 5), using the distance from Eq. (6). After that, we use the neighbors to update the estimates of probabilities that appear in the definition of the weights in Eq. (4) for all attributes (lines 8–10). The estimates of probabilities are updated with the weighted average of the distances between $\boldsymbol{r}$ and its neighbors. The weight $\delta(\ell)$ for the ℓ-th nearest neighbor is proportional to $\exp(-(\sigma\ell)^2)$, and it is normalized ($\sum_{\ell=1}^{k}\delta(\ell) = 1/mk$) to ensure that $w_i \in [-1, 1]$, when the algorithms finishes. The parameter σ is user defined. Finally, the weights w_i are computed (line 12).

Algorithm 3 MTR-Relief($\mathscr{D}$, m, k, σ)

```
1: $\boldsymbol{P}_{\text{diffFeat, diffTarget}}, \boldsymbol{P}_{\text{diffFeat}}$ = lists of length $D$ consisting of zeros
2: $P_{\text{diffTarget}} = 0.0$
3: for $\iota = 1, 2, \ldots, m$ do
4:     $\boldsymbol{r}$ = random example from $\mathscr{D}$
5:     $I_1, I_2, \ldots, I_k = k$ nearest neighbors of $\boldsymbol{r}$
6:     for $\ell = 1, 2, \ldots, k$ do
7:         $P_{\text{diffTarget}} \mathrel{+}= \delta(\ell)d_{\mathcal{Y}}(\boldsymbol{r}, I_\ell)$
8:         for $i = 1, 2, \ldots, D$ do
9:             $\boldsymbol{P}_{\text{diffFeat}}[i] \mathrel{+}= \delta(\ell)d_i(\boldsymbol{r}, I_\ell)$
10:            $\boldsymbol{P}_{\text{diffFeat, diffTarget}}[i] \mathrel{+}= \delta(\ell)d_i(\boldsymbol{r}, I_\ell)d_{\mathcal{Y}}(\boldsymbol{r}, I_\ell)$
11: for $i = 1, 2, \ldots, D$ do
12:     $w_i = \frac{\boldsymbol{P}_{\text{diffFeat, diffTarget}}[i]}{P_{\text{diffTarget}}} - \frac{\boldsymbol{P}_{\text{diffFeat}}[i] - \boldsymbol{P}_{\text{diffFeat, diffTarget}}[i]}{1 - P_{\text{diffTarget}}}$
```

As the number of iterations m increases, the estimates of the probabilities are expected to be more accurate. However, note that we do not need more than M iterations. The value of k should be small enough to capture the local structure in the data and capture the interactions between features (Robnik-šikonja and Kononenko 2003). When k gets bigger, we may want to weight neighbors' contributions with σ.

2.2.2 Theoretical analysis of relief algorithms

Below we analyze some properties of the algorithms from the Relief family. The motivation for the definition of Relief is, as mentioned before, an elegant probabilistic interpretation of the weights in the case when both the feature and the target are nominal. In that case, the feature x_i is rewarded in a particular iteration, i.e., the weight update is positive, if (loosely speaking) two instances are more probable to have different values of x_i if they belong to different classes as opposed to the case when they are of the same class.

In the following, we fix the feature x_i. First, we specify the condition when the weight w_i is positive. As a consequence, we will derive the result indicating the most rewarded features by the regression versions of Relief. For simplicity, we first assume that $k = 1$ and $\sigma = 0$. After the proof for this case, we will generalize the results. In the following, $\boldsymbol{r}_\iota$ is the randomly chosen instance in the ι-th iteration, and $\boldsymbol{s}_\iota$ is its nearest neighbor.

Theorem 1 *The feature x_i is rewarded (gets a positive weight update) in the ι-th iteration if, and only if, the target distance $d_{\mathcal{Y}}(\boldsymbol{r}_\iota, \boldsymbol{s}_\iota)$ is not smaller than the average target distance, computed by the algorithm.*

Proof We first take two arbitrary events A and B and simplify the expression $P(A \mid B) - P(A \mid \neg B)$ to

$$\frac{P(AB) - P(A)P(B)}{P(B)(1 - P(B))}. \tag{8}$$

We then set A and B to `diffFeat` and `diffTarget`, respectively (defined in Sect. 2.2.1). The weight w_i, computed by Algorithm 3 is a special instance of Eq. (8):

$$w_i = \frac{\frac{1}{m}\sum_\iota \alpha_\iota \tau_\iota - \left(\frac{1}{m}\sum_\iota \alpha_\iota\right)\left(\frac{1}{m}\sum_\iota \tau_\iota\right)}{\left(\frac{1}{m}\sum_\iota \tau_\iota\right)\left(1 - \frac{1}{m}\sum_\iota \tau_\iota\right)}, \tag{9}$$

where we introduced $\tau_\iota = d_{\mathcal{Y}}(\boldsymbol{r}_\iota, \boldsymbol{s}_\iota)$ and $\alpha_\iota = d_i(\boldsymbol{r}_\iota, \boldsymbol{s}_\iota)$. If we rearrange the terms in Eq. (9) and define $\tau = \sum_\iota \tau_\iota$, we obtain

$$w_i = \frac{m}{\tau(m-\tau)} \sum_\iota \left(\tau_\iota - \frac{1}{m}\tau\right)\alpha_\iota \tag{10}$$

Since $0 \le \tau_\iota \le 1$ and τ/m is the average computed distance in the target space, we have completed the proof. □

Note that this theorem also applies for the standard (single target) regression task. In the case, when $k > 1$ and $\sigma = 0$, we get additional inner sums $\sum_\ell$ over the neighbors and the terms τ_ι, α_ι and $a_\iota\tau_\iota$ are respectively replaced by $\sum_{\ell=1}^{k} \tau_{\iota,\ell}$, $\sum_{\ell=1}^{k} \alpha_{\iota,\ell}$ and $\sum_{\ell=1}^{k} \alpha_{\iota,\ell}\tau_{\iota,\ell}$. Here, the terms $\alpha_{\iota,\ell}$ and $\tau_{\iota,\ell}$ are defined analogously to α_ι and τ_ι. The additional index ℓ specifies the neighbor to which the distance is computed, and instead of Eq. (10) we now have

$$w_i = \frac{mk}{\tau(mk-\tau)} \sum_\iota \sum_\ell \left(\tau_{\iota,\ell} - \frac{1}{mk}\tau\right)\alpha_{\iota,\ell},$$

where $\tau = \sum_\iota \sum_\ell \tau_{\iota,\ell}$.

If $\sigma \neq 0$, the generalization of the upper formula is obvious.
As a corollary, we derive the result indicating the most rewarded features by the regression versions of Relief. Before proceeding to the result, we define the *centralization* of the random variable X as the random variable $X - \mathbb{E}[X]$. In the formulation of the next theorem, we treat the features as random variables.

Theorem 2 *For a given feature, it is optimal if its centralization is locally positively linearly dependent on the target.*

Proof The features are sorted with respect to the weights w_i, but the order of the features does not change if we define the weights as

$$\omega_i = \sum_\iota \left(\tau_\iota - \frac{1}{m}\tau\right)\alpha_\iota,$$

since we have rescaled the weights from Eq. (10) by multiplying them by a positive constant. If we define the vectors $\boldsymbol{\alpha} = (\alpha_\iota)_\iota$ and $\boldsymbol{\tau} = (\tau_\iota - \tau/m)_\iota$ and fix their Euclidean norms, then, by the Cauchy-Schwarz inequality, it follows that $\omega_i \le \|\boldsymbol{\alpha}\|\,\|\tau\|$. We also know that the maximum value for ω_i is achieved when $\boldsymbol{\alpha} = \beta\boldsymbol{\tau}$ for some $\beta \ge 0$. □

Note that the locality in the theorem originates from the pairs of instances for which the distances d_i and $d_{\mathcal{Y}}$ are computed. Theorem 1 gives some insight into the Relief procedure on its own, but it mostly serves as an intermediate step for proving Theorem 2 which actually characterizes the features that the algorithm may have some bias toward.

2.3 $\overline{\text{STR}}$ approach

If a structured target domain $\mathcal{Y}$ can be decomposed into primitive parts, we can compute a feature ranking (or build a predictive model) for each part separately, and than average (combine) the results to obtain the final ranking. One of the prominent examples of this technique from predictive the modeling field is the binary relevance approach for multi-label classification (Tsoumakas and Katakis 2007).

To the best of our knowledge, there is no previous use of approaches based on binary relevance for performing MTR feature ranking. We build on this idea and formulate a generic algorithm for MTR feature ranking in Algorithm 4. For the base feature ranking method F, we use RReliefF and the three ensemble-based scores from Sect. 2.1.2.

Algorithm 4 $\overline{\text{STR}}$-ranking(F, $\mathcal{D}$, ...)

```
1: for $j = 1, 2, \ldots, T$ do
2:     $\mathcal{D}_j = \{(\boldsymbol{x}, y_j(\boldsymbol{x})) \mid (\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})) \in \mathcal{D}\}$
3:     compute $importance_j(x_i)$ by $F(\mathcal{D}_j, \ldots)$, for all $i$
4: for $i = 1, 2, \ldots, D$ do
5:     $importance(x_i) = \sum_{j=1}^{T} importance_j(x_i)/T$
```

As stated in the discussion earlier in Sect. 1, simple (non-weighted) averaging is an option when one does not have any background knowledge of the data. However, some of the targets can be given a higher importance on the overall ranking if, e.g., a domain expert so desires.

3 Computational complexity

In this section, we analyze the time and space complexity of the proposed algorithms using the big $\mathcal{O}$ notation. The variables that are not explicitly defined here, are listed in Table 1.

3.1 Ensemble scores

To simplify the conclusions, the analysis is carried out under the assumption that the trees in the forest are balanced (Kocev et al. 2013), i.e., a tree consists of $\mathcal{O}(M)$ nodes, has depth $\mathcal{O}(\log_2 M)$ and $M_{\mathcal{N}} = \mathcal{O}(M/2^{d(\mathcal{N})})$ examples reach a node $\mathcal{N}$ at depth $d(\mathcal{N})$. We also assume that all features are numeric, since i) handling them is more time consuming than handling nominal features and ii) the majority of features in the datasets in our experiments (see Sect. 4.2) is numeric.

For each considered x_i, we first need to sort the examples in the given node ($\mathcal{O}(M_{\mathcal{N}} \log M_{\mathcal{N}})$) and then find the best split ($\mathcal{O}(T M_{\mathcal{N}})$). The complexities for inducing an ensemble of size S are then: $\mathcal{O}(SDM \log M(\log M + T))$ for bagging, $\mathcal{O}(S\sqrt{D}M \log M$ $(\log M + T))$ for RF and $\mathcal{O}(SDTM \log M)$ for ET. The additional cost for updating the importances of all features is $\mathcal{O}(SM)$ for the Symbolic and the Genie3 score, and $\mathcal{O}(SDM \log M)$ for the Random Forest score.

If we assume $T = 1$, we obtain the complexity of growing a single STR ensemble. Since we have to grow T of them, the complexities are now $\mathcal{O}(SDMT \log^2 M)$ for bagging, $\mathcal{O}(S\sqrt{D}TM \log^2 M)$ for RF and $\mathcal{O}(SDTM \log M)$ for ET. Hence, the complexity of growing the ET ensemble stays the same, whereas the other two increase. The additional per-tree cost for updating importances is the same, thus we have T-times more work with updating the importances in the $\overline{\text{STR}}$ case.

We draw the following conclusions. First, the cost of updating importances is negligible in the case of Symbolic and Genie3 scores. Second, growing a STR ensemble is by factor $\min\{T, \log M\}$ more time consuming that growing a MTR ensemble. However, STR trees are usually bigger than their MTR analogues and thus tend to take longer to grow in practice. Third, since it is not much more time consuming to evaluate all splits that a feature yields, as compared to evaluating a single split, random forests and bagging ensembles time complexity is (also in practice) comparable to the time complexity of extra trees ensembles.

3.2 Relief scores

First, we analyze the time complexity for MTR-Relief. Since the space-partitioning data structures, such as kD trees (Friedman et al. 1977) do not perform well when the dimension D is high, we use a brute-force method for finding the nearest neighbors. Finding k nearest neighbors takes $\mathcal{O}(MD)$ steps, assuming that an update of the heap of the current k nearest neighbors ($\mathcal{O}(\log k)$) is negligible in comparison to a distance computation ($\mathcal{O}(D)$).

Before we start updating the probabilities, we compute the distances in the target space between $\boldsymbol{r}$ and its neighbors, which takes $\mathcal{O}(kT)$ time. Updating the probabilities now takes only $\mathcal{O}(kD)$ time.

Computation of the weights takes $\mathcal{O}(D)$ time. Thus, the overall time taken by MTR-Relief is $\mathcal{O}(m[MD + kT + kD] + D) = \mathcal{O}(m[MD + kT])$. Since usually $kT < MD$, this amounts to $\mathcal{O}(mMD)$.

Hence, the time needed for computing $importance_j(x_i)$, for all i and a fixed j, in the $\overline{\text{STR}}$-Relief is $\mathcal{O}(mMD)$, hence the overall time needed here is $\mathcal{O}(TmMD)$, which is higher than the time complexity of MTR-Relief by a factor of T.

However, note that the nearest neighbors do not depend on the target space, hence we can make the naive implementation of $\overline{\text{STR}}$-Relief more efficient, if we update the probabilities in parallel, for each target. This comes with the cost of a larger memory consumption. The additional space that is needed in MTR-Relief is $\mathcal{O}(D)$: the lists $\boldsymbol{P}_{\text{diffFeat, diffTarget}}$, $\boldsymbol{P}_{\text{diffFeat}}$ and the list storing the w_is. In the case of parallel updates for $\overline{\text{STR}}$-Relief, we need $\mathcal{O}(TD)$ memory, which might be a considerable amount if the numbers for features and targets are large. The same time-space consumption relation holds when we want to compute the rankings for different values of parameters: either time or space consumption increases by a factor of the number of combinations we want to study.

4 Experimental design

In this section, we present in detail the experimental design used to evaluate the performance of the proposed methods. We begin by stating the experimental questions. We then briefly summarize the MTR datasets used in this study. Next, we present the evaluation procedure. Finally, we give the specific parameters instantiations for each of the methods.

4.1 Experimental questions

We design the experimental evaluation focusing on several research questions referring to the parametrization of the proposed methods, the relevance of the obtained feature rankings, and their (relative) performance:

1. Which ensemble method is the most appropriate for a given importance score in ensemble-based feature ranking?
2. How do the number of neighbors k, the number of iterations m and the weighting parameter σ influence the rankings obtained with MTR-Relief?
3. How do the MTR rankings compare to their single target ($\overline{\text{STR}}$) counterparts?
4. Can the knowledge of feature importance (scores) lead to better predictive performance of a regressor, i.e., are the obtained feature rankings relevant?
5. Which ranking method yields the best feature rankings overall?

4.2 Datasets

We use 24 MTR benchmark problems. Table 3 presents the basic statistics of the datasets. The number of features ranges from 6 to 576 and the features are mainly numeric. The number of targets ranges from 2 to 16, while the number of examples is in the range between 103 and 60607. The datasets come from different domains: the *ATP* datasets concern the prediction of airline tickets prices. The *collembola*, *forestry*, *soil quality*, and *vegetation condition* datasets

Table 3 Description of the benchmark problems in terms of the number of nominal and numeric descriptive attribute, the number of targets, and the number of examples

Dataset	Nominal	Numeric	Targets	Examples
ATP1d (Spyromitros-Xioufis et al. 2016)	0	411	6	337
ATP7d (Spyromitros-Xioufis et al. 2016)	0	411	6	296
collembola (Kampichler et al. 2000)	8	39	3	393
EDM (Karalič and Bratko 1997)	0	16	2	154
ENB (Tsanas and Xifara 2012)	0	8	2	768
Forestry Kras (Stojanova et al. 2010)	0	160	11	60,607
Forestry LIDAR IRS (Stojanova 2009)	0	29	2	2730
Forestry LIDAR Landsat (Stojanova 2009)	0	150	2	6218
Forestry LIDAR Spot (Stojanova 2009)	0	49	2	2730
jura (Goovaerts 1997)	0	15	3	359
OES10 (Spyromitros-Xioufis et al. 2016)	0	298	16	403
OES97 (Spyromitros-Xioufis et al. 2016)	0	263	16	334
osales (Kaggle 2012)	0	401	12	639
RF1 (Spyromitros-Xioufis et al. 2016)	0	64	8	9125
RF2 (Spyromitros-Xioufis et al. 2016)	0	576	8	9125
SCM1d (Spyromitros-Xioufis et al. 2016)	0	280	16	9803
SCM20d (Spyromitros-Xioufis et al. 2016)	0	61	16	8966
scpf (Kaggle 2013)	0	23	3	1137
sigmeareal (Demšar et al. 2005)	0	6	2	817
sigmeasim (Demšar et al. 2005)	2	9	2	10,368
slump (Yeh 2007)	0	7	3	103
soil quality (Demšar 2006)	0	156	3	1944
vegetation condition (Kocev et al. 2009)	1	39	7	16,967
water quality (Džeroski et al. 2000)	0	16	14	1060

contain environmental data, *EDM* stands for electrical discharge machining, *ENB* and *water quality* originate from studies of water quality. *Jura* contains measurements of heavy metals concentrations. *OES* stands for occupational employment survey. The datasets *osales* (online product sales) and *scpf* (see-click-predict fix) originate from two Kaggle competitions, The *RF1* and *RF2* datasets describe river flows. The *SCM1d* and *SCM20d* datasets were constructed for a competition in the domain of supply chain management. The *sigmeareal* and *sigmeasim* datasets describe GMO (herbicide resistant) crops. Finally, the *slump* dataset concerns prediction of properties of concrete slump.

4.3 Evaluation methodology

We adopted the following evaluation methodology to answer the above research questions and to properly assess the performance of the proposed methods. First, we randomly divide each dataset $\mathscr{D}$ into 2/3 for training ($\mathscr{D}_{\text{TRAIN}}$) and 1/3 for testing ($\mathscr{D}_{\text{TEST}}$). A ranking is computed from the training part only, and evaluated on the testing part. This is repeated 10 times and the performance measures are averaged at the end.

The quality of a ranking is assessed by using the kNN algorithm where instead of the standard Euclidean distance, its weighted version was used. For two input vectors $\boldsymbol{x}^1$ in $\boldsymbol{x}^2$, the distance d between them is defined as

$$d(\boldsymbol{x}^1, \boldsymbol{x}^2) = \sqrt{\sum_{i=1}^{D} w_i d_i^2(\boldsymbol{x}_i^1, \boldsymbol{x}_i^2)}, \tag{11}$$

where d_i is defined by Eq. (5). The weights are set to $w_i = \max\{importance(x_i), 0\}$, since they need to be made non-negative to ensure that d is well defined, and also to ignore the attributes that are of lower importance than a randomly generated attribute would be.

The evaluation through a kNN predictive model was chosen because of two main reasons. First, this is a distance based model, hence, it can easily make use of the information contained in the feature importances in the learning phase. Second, kNN is simple: its only parameter is the number of neighbors, which we set to 5. In the prediction stage, the neighbors' contributions to the predicted value are equally weighted, so we do not introduce additional parameters that would influence the performance.

A further argument for using kNN as an evaluation model is as follows. If a feature ranking is meaningful, then when the feature importances are used as weights in the calculation of the distances kNN should produce better predictions as compared to kNN without using these weights (Cunningham and Delany 2007; Wettschereck 1994).

We assess the predictive performance with the average relative root mean squared error $\overline{\text{RRMSE}}$. If we denote the predicted value of the target y_j by $\hat{y}_j(\boldsymbol{x})$ and the variance of the target y_j on $\mathscr{D}_{\text{TRAIN}}$ by $Var_j(\mathscr{D}_{\text{TRAIN}})$, then the RRMSE for this target is defined as

$$\text{RRMSE}(y_j) = \sqrt{\frac{1}{|\mathscr{D}_{\text{TEST}}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathscr{D}_{\text{TEST}}} \frac{(\boldsymbol{y}_j(\boldsymbol{x}) - \hat{y}_j(\boldsymbol{x}))^2}{Var_j(\mathscr{D}_{\text{TRAIN}})}},$$

where $\boldsymbol{y}_j(\boldsymbol{x})$ is the true value of the y_j for the example $\boldsymbol{x}$. From this, we compute $\overline{\text{RRMSE}} = \frac{1}{T}\sum_{j=1}^{T} \text{RRMSE}(y_j)$.

4.4 Statistical analysis of the results

For comparing two algorithms, we use the Wilcoxon's test, and for comparing more than two algorithms, we use the Friedman's test. In both cases, the null hypothesis H_0 is that all considered algorithms have the same performance. If H_0 is rejected by the Friedman's test, we additionally apply Nemenyi's post-hoc test to investigate where the statistically significant differences between *any* two algorithms occur. A detailed description of all tests is available in Demšar (2006).

When performing more than one Wilcoxon's test for a given hypothesis, we control the false discovery rate by the Benjamini–Hochberg procedure (Benjamini and Hochberg 1995): let p_i be the i-th smallest among the obtained p values, and t the number of tests. Let i_0 be the largest i, such that

$$p_i \leq \frac{i}{t}\alpha =: \hat{\alpha}_i. \tag{12}$$

Then, we can reject the hypotheses that correspond to p values p_i, for $1 \leq i \leq i_0$.

The results of the Nemenyi tests are presented through average ranks diagrams. Each diagram shows the average rank of the algorithm over the considered datasets, and the critical distance, i.e., the distance by which the average ranks of two considered algorithms must differ to be considered statistically significantly different. Additionally, the groups of algorithms among which no statistically significant differences occur are connected with a line.

Before proceeding with the statistical analysis, we round the performances to three decimal points. In the analysis, the significance level was set to $\alpha = 0.05$.

4.5 Parameter instantiation for the ensemble scores

The algorithm for inducing an ensemble of PCTs for MTR takes as input the following parameters: the number of base predictive models in the forest (all ensemble types), minimal number of examples in a leaf of a tree (all ensemble types), and the feature subset size (random forests only). In all cases, we grow 100 trees, whose leaves must contain at least two examples each. Additionally, the feature subset size in the case of random forests is set to $\lceil\sqrt{D}\rceil$.

Next, recall that the Symbolic score requires selecting a value for w. In a preliminary study of the MTR versions of the scores, we investigated the influence of several values of w, i.e., $w \in \{0.25, 0.5, 0.75, 1\}$, on the performance of the produced feature rankings. We performed Friedman's test and it turns out that the differences among the rankings are not statistically significant in the case of bagging (p value is 0.379) and ET (p value is 0.441), whereas in the case of RF, they are (p value is 0.0148). In the RF case, we can proceed to Nemenyi's test, whose results are shown in Fig. 1.

The diagram reveals that only the Symbolic score with weight $w = 1.0$ is statistically significantly worse than the rankings with weights $w = 0.5$ and $w = 0.25$. This can be

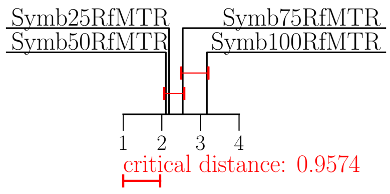


Fig. 1 The average ranks diagram for Nemenyi's test, comparing different values of the parameter w, $w \in \{0.25, 0.5, 0.75, 1.0\}$, for the Symbolic score computed from the RF ensemble

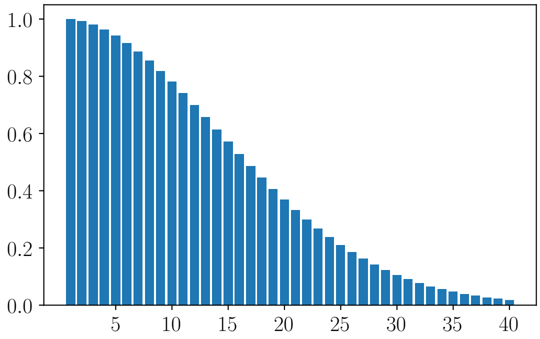


Fig. 2 The influence for the neighbors k in the range $1 \leq k \leq 40$ for the Relief score when $\sigma = 0.05$. The weights are normalized so that the weight of the nearest neighbor equals 1

explained by Eq. (1): the value of 1.0 for the weight is the only one where the depth of the node where an attribute appears is not taken into account when computing the relevance.

Since the average ranks of the ranking methods with $w = 0.25$, $w = 0.5$, $w = 0.75$ and $w = 1.0$ are respectively 2.31, 2.27, 2.56 and 2.85 for bagging, and 2.40, 2.40, 2.35 and 2.88 for ET, the ranking *Symb50* is a reasonable choice for all three ensembles, since it is always ranked at least second. The reason for this is less obvious, but we hypothesize that it could be an artifact of the algorithm for inducing ensembles. Namely, the splits in the nodes of the trees are binary. If we assume that the best test in an internal node $\mathcal{N} \in \mathcal{T}$ partitions the examples approximately in half, then the attribute in $\mathcal{N}$'s test influences one half of the instances that arrive to its parent; hence, the parent should receive twice as large a reward as each of its two children. The value $w = 0.5$ was, for this same reason, selected also in the $\overline{\text{STR}}$ versions of the scores.

4.6 Parameter instantiation for the relief score

Since the sizes of datasets range over different orders of magnitude, the number of iterations m is specified as a proportion of the size of $\mathcal{D}_{\text{TRAIN}}$. The considered values are $m \in \{1\%, 5\%, 10\%, 25\%, 50\%, 100\%\}$. On the other hand, since the number of neighbors k controls the level of locality, it is better given in absolute values. Our choice is to consider the values $k \in \{1, 5, 10, 15, 20, 25, 30, 40\}$.

The influence of weighting is investigated with two values of the weighting parameter that result in two extreme weighting schemes: $\sigma = 0$ corresponds to no weighting, i.e., all neighbors have equal influence, and $\sigma = 0.05$, that results in a weighting scheme where the possible 40th nearest neighbor has only $\sim 2\%$ of the influence of the nearest neighbor, see Fig. 2.

5 Results and discussion

In this section, we present and discuss the results of the comprehensive experimental evaluation. First, we discuss the results for the ensemble-based scores. Next, we present the parametrization of MTR-Relief. At the end, we compare the performance of the proposed ensemble-based and Relief-based feature ranking methods.

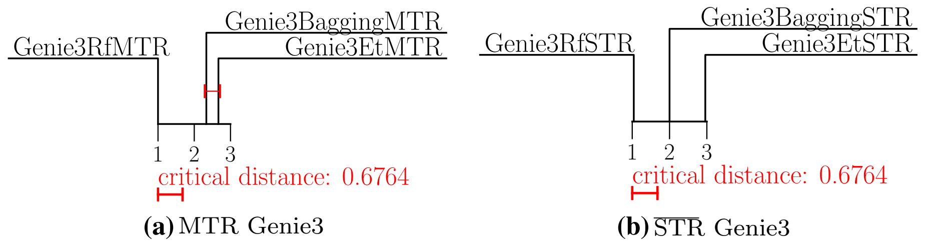


Fig. 3 The average ranks diagrams from Nemenyi's post-hoc test, applied to the times needed to compute feature rankings with different ensemble methods for the MTR (**a**) and the $\overline{\text{STR}}$ (**b**) version of the Genie3 score

5.1 Ensemble-based feature ranking

5.1.1 Which ensemble method is the most appropriate

For a given fixed score (Genie3, Symb50 or RF) and ranking type (MTR or $\overline{\text{STR}}$), we compare the performance of the different ensemble methods in terms of $\overline{\text{RRMSE}}$ to select the most suitable one for a given score.

For the MTR rankings, the outcome can be summarized as follows. The Friedman test for Genie3 and Symb50 scores did not reject the null hypotheses, with p values of 0.476 and 0.877, respectively. The Wilcoxon test for the RF score rejected the null hypotheses with $p = 0.0170$ and indicated that the bagging method is statistically significantly better than the RF ensemble method.

If methods perform equally well, then the most efficient should be preferred. Hence, we compare the different ensemble methods for the scores Genie3 and Symb50 in terms of time. The differences are significant: the p values are 1.00×10^{-15} and 3.12×10^{-14} respectively. Thus, we can proceed to Nemenyi's post-hoc tests that reveal that, in both cases, the RF ensemble method is significantly faster than the other two methods. This means that it suffices to consider $\sqrt{D}$ features in a tree node to obtain a good split (hence, RF is faster than bagging), but only D test candidates for ET do not suffice (since the trees in ET ensemble must be considerably bigger than the trees in a RF). Since the average ranks diagrams are qualitatively the same, we show only the one for the Genie3 score in Fig. 3a.

For the STR rankings, there are no statistically significant differences regarding the quality of the ranking, i.e., the Friedman tests (for Genie3 and Symb50 scores) and the Wilcoxon test (RF score) do not reject the null hypotheses. Conversely, all three null hypotheses for the time complexity analysis are rejected with p values of 0.0, 0.0 and 1.82×10^{-5} for the scores Genie3, Symb50 and RF, respectively. The differences are now it the case of Genie3 and Symb50 even more profound. In addition to the RF ensemble method being faster than the other two methods, bagging is now also significantly faster than the ET method. The average ranks diagram for the Genie3 score is shown in Fig. 3b.

5.1.2 Comparison of MTR and $\overline{\text{STR}}$ rankings

For each of the 8 combinations of ranking score and ensemble method, we compare the quality of MTR and $\overline{\text{STR}}$ rankings with the Wilcoxon test. The smallest of the eight p values is $p_1 = 0.0298$ in the case of the Genie3 score, coupled with bagging. Since this is not smaller than Benjamini–Hochberg correction $\hat{\alpha}_1 = 0.00625$ from Eq. (12), we cannot reject any hypothesis. On the other hand, every MTR ranking is significantly faster than its $\overline{\text{STR}}$

Table 4 The results of the Wilcoxon tests that compare the performance of standard 5NN to its weighted-distance version

Score-ensemble-version	p_i	$\hat{\alpha}_i$
Genie3-RF-$\overline{\text{STR}}$	$\mathbf{7.59 \times 10^{-5}}$	8.33×10^{-3}
Genie3-RF-MTR	$\mathbf{2.28 \times 10^{-4}}$	1.00×10^{-2}
Symb50-RF-MTR	$\mathbf{3.40 \times 10^{-3}}$	1.25×10^{-2}
Symb50-RF-$\overline{\text{STR}}$	$\mathbf{9.71 \times 10^{-3}}$	1.67×10^{-2}
RF-Bagging-MTR	$\mathbf{2.49 \times 10^{-3}}$	2.50×10^{-2}
RF-RF-$\overline{\text{STR}}$	8.64×10^{-2}	5.00×10^{-2}

The ith row contains the name of the *score-ensemble-version* triplet that provided the feature importances used in the tests of weighted 5NNs against standard 5NN; the p value p_i; and the corrected value $\hat{\alpha}_i$ Statistically significant results are shown in bold

counterpart: all p values now equal 1.8×10^{-5} and all null hypotheses can be rejected. Thus, the MTR rankings should be preferred.

5.1.3 Are the obtained feature rankings relevant?

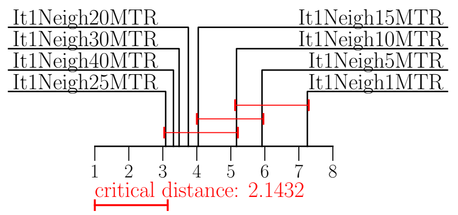
Here, we investigate whether 5NN prediction can benefit from using the additional information from the feature importances. To this end, we compare the performance of 5NN without and with feature importances. We test the relevance of both the MTR and the $\overline{\text{STR}}$ version of each score, coupled with the most suitable ensemble method, as determined in Sect. 5.1.1. Hence, six comparisons are made by using the Wilcoxon's test, where we test the standard 5NN against the weighted 5NN whose feature weights are defined via feature importances as in Eq. (11)

Table 4 gives the results of the statistical evaluation. It shows that the hypotheses are rejected in the case of the Genie3 and the Symb50 scores (in favor of the weighted 5NN). This holds for both the MTR and the $\overline{\text{STR}}$ versions of the scores. The hypothesis was also rejected in the case of the MTR version of the Random Forest score, again in favor of the weighted 5NN. Therefore, we can conclude that using feature ranking is beneficial in these five cases, i.e., the obtained feature importances are relevant and meaningful.

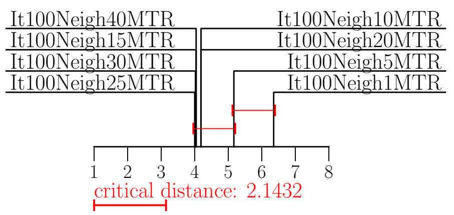
5.2 Relief feature ranking

5.2.1 Influence of the neighbor weighting σ

For both proposed methods (MTR-Relief and $\overline{\text{STR}}$-Relief), we performed 48 Wilcoxon's tests comparing the weighted variant of the algorithm ($\sigma = 0.05$) to its non-weighted variant ($\sigma = 0$). The other two parameters, i.e., m and k were fixed. Here, we will report only the lowest p values for each of the versions. In the case of MTR-Relief, the lowest one was $p_1 = 0.0653$, while in the case of $\overline{\text{STR}}$-Relief, the lowest one was $p_1 = 0.0864$. Thus, the differences are not statistically significant, even before we apply the Benjamini–Hochberg procedure. Since the two quite extreme weighting schemes result in rankings with only minor differences in quality, we conclude that the influence of the weighting parameter is limited. Therefore, we consider only the non-weighted variants of the algorithms in the further analysis.



(a) $m = 1\%$



(b) $m = 100\%$

Fig. 4 The influence of the number of neighbors (k): the average ranks diagrams from Nemenyi's post-hoc test for MTR-Relief with $m = 1\%$ (**a**) and $m = 100\%$ (**b**)

5.2.2 Influence of the number of neighbors *k*

To assess the influence of the number of neighbors, we compare the quality of the rankings obtained by varying the values of k, while the number of iterations (value of m) and ranking type (MTR or $\overline{\text{STR}}$) are fixed. This is done by applying Friedman tests to the described groups of algorithms. The obtained p values corresponding to the 10 groups with $m \leq 50\%$ are all smaller than 10^{-10}. The remaining two p values are $p = 0.00438$ (MTR ranking) and $p = 1.26 \times 10^{-4}$ ($\overline{\text{STR}}$ ranking), hence all null hypotheses can be rejected and we can say that the number of neighbors has considerable influence on the quality of the feature ranking.

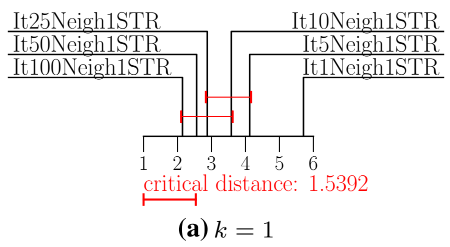
We proceed to Nemenyi's post-hoc tests. In Fig. 4, we show average ranks diagrams for the two extreme values of m: $m \in \{1\%, 100\%\}$, for the MTR rankings. The diagrams for $\overline{\text{STR}}$ rankings (and other values of m) are similar. Using more neighbors is obviously better when $m = 1\%$ (Fig. 4a). In this case, only a small portion of $\mathscr{D}_{\text{TRAIN}}$ gets to influence the ranking, thus the ranking overfits to the structure of $\mathscr{D}_{\text{TRAIN}}$ in the neighborhood of the chosen instances. We can mitigate this by examining larger neighborhoods, i.e., choosing more neighbors whose influence is then averaged. In the case of $m = 1\%$, this is the only mechanism that reduces overfitting, hence the strong influence on the performance. In the results with $m = 100\%$ (Fig. 4b), we get to examine the whole $\mathscr{D}_{\text{TRAIN}}$, so the algorithm does not overfit to the extent observed before. However, the diagrams reveal that choosing only one neighbor is still not sufficient, but we do not need more than 10 neighbors to obtain good performance.

5.2.3 Influence of the number of iterations *m*

We perform analogous experiments to the ones described in the previous section, where the number of neighbors k and ranking type are fixed, while the values of the parameter m vary.

The p values from Friedman tests are all smaller than 2.0×10^{-6}, hence in all of the cases, the number of iterations influences the ranking quality. We then perform the follow-up Nemenyi's post-hoc tests and show the resulting average ranks diagrams for the two most extreme cases: $k \in \{1, 40\}$, but now for the $\overline{\text{STR}}$ rankings (Fig. 5).

When $k = 1$ (Fig. 5a), the general more-is-better trend is obvious. This can be again explained by a slightly modified overfitting argument. Since the datasets originate from real-world domains, they must contain some (random) noise. In the case of $k = 1$, its influence can be diminished only by taking into account more and more instances. Approximately the same holds for $k = 40$ (Fig. 5b), but in this case, the average ranks of the scores for $m \geq 10$ are closer. Furthermore, they show that the only values of m that are always part



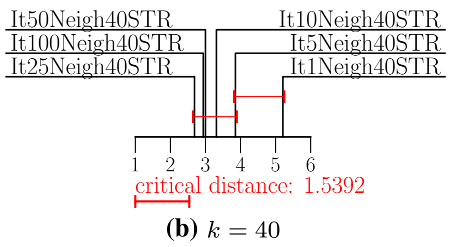


Fig. 5 The influence of the number of iterations (m): the average ranks diagrams from Nemenyi's post-hoc test for $\overline{\text{STR}}$-Relief with $k = 1$ (**a**) and $k = 40$ (**b**)

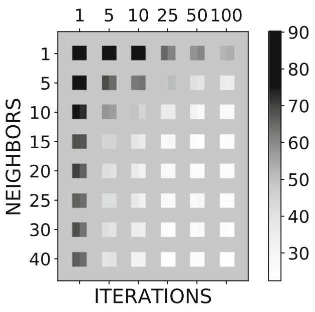


Fig. 6 Heat map of the average ranks of different options for the value pairs (m, k). The background of the map corresponds to the average rank of the non-weighted 5NN, whereas each square with coordinates (m, k) represents the average ranks of the regressor which uses the MTR-Relief (the left part of the square) or $\overline{\text{STR}}$-Relief (the right part of the square) ranking, computed with these parameter values

of the top-performing group of scores, are 25%, 50% and 100%. Similar conclusions can be made by inspecting the other diagrams.

5.2.4 Comparison of MTR and $\overline{\text{STR}}$ rankings

We first compare MTR ans $\overline{\text{STR}}$ rankings based on the Relief score graphically by showing a heat map, where the color corresponds to the average rank of the corresponding regressor. In Fig. 6, there are 48 squares, each of them corresponding to a (m, k) pair of values. Additionally, each square is divided in half. Its left and right part correspond to MTR-Relief and $\overline{\text{STR}}$-Relief, respectively. The background of the heat map is of the color that corresponds to the performance of the non-weighted 5NN regressor. An inspection of the graph reveals that no differences are to be expected between the MTR and $\overline{\text{STR}}$ versions, since most of the time a difference is not visible between the left and right part of each square. The smallest among the 48 p values from the Wilcoxon tests, where we compare MTR and $\overline{\text{STR}}$ rankings is $p = 0.0163$ and is obtained when $m = 10\%$ and $k = 25$. However, if we apply the Benjamini–Hochberg correction, there are no statistically significant differences.

5.2.5 Are the obtained feature rankings relevant?

In Fig. 6, we can additionally see that some parameter settings clearly lead to irrelevant rankings, e.g., $m = 1\%$ and $k = 1$, for which the average ranks for the MTR and $\overline{\text{STR}}$ weighted 5NN predictive performance are 89.4 and 90.1, whereas the baseline 5NN regressor

Table 5 The results of the Wilcoxon's tests that compare the performance of standard 5NN to its weighted-distance version

m-k-version	p_i	$\hat{\alpha}_i$
It100Neigh15MTR	**0.00664**	0.00833
It50Neigh15STR	0.0119	0.01
It25Neigh40STR	0.0975	0.0125
It25Neigh40MTR	0.100	0.0167
It100Neigh30STR	0.145	0.025
It100Neigh40MTR	0.153	0.05

The ith row contains the values of the parameters of the Relief ranking that provided the feature importances and was tested against standard 5NN; the p value p_i; and the corrected value $\hat{\alpha}_i$
Statistically significant result is shown in bold

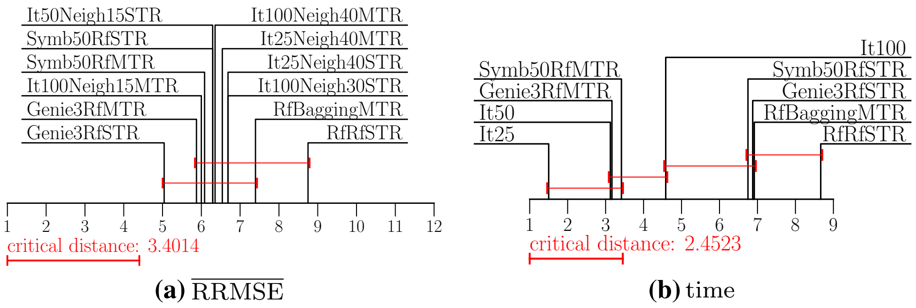


Fig. 7 Comparison of the 12 ranking scores: the average ranks diagrams from Nemenyi's post-hoc test for $\overline{\text{RRMSE}}$ (**a**) and time (**b**). Due to the implementation of Relief, the running times for Relief depend only on the number of iterations

has the average rank of 48.5. Hence, we take a similar approach to that in Sect. 5.1.3, where we tested for the relevance of ensemble scores. We determine the best three parameter settings for each of the rankings (MTR and $\overline{\text{STR}}$) and then apply the Wilcoxon tests for the weighted 5NN against the non-weighted 5NN baseline. The results are given in Table 5.

In contrast to the ensemble scores relevance analysis, now only one ranking performs statistically significantly better than the baseline. This is the MTR version of the score when $m = 100\%$ and $k = 15$.

5.3 Which ranking method yields the best feature rankings?

In this section, we compare the six candidates from the Ensemble-based rankings and the six candidates from the Relief-based rankings. The comparison is made in terms of $\overline{\text{RRMSE}}$ and time, by applying Friedman's test. The corresponding p values are $p_{\overline{\text{RRMSE}}} = 0.117$ and $p_{\text{time}} = 0.0$. The follow-up Nemenyi's test produced the average ranks diagrams shown in Fig. 7. Note that the critical distance in the $\overline{\text{RRMSE}}$-based diagram should be ignored ($p_{\overline{\text{RRMSE}}} > 0.05$) and we focus only on the average ranks. It seems that the quality of different scores is quite similar: most of them are grouped together with the exception of the $\overline{\text{STR}}$ rankings of Genie3 and RF score as two outliers.

On the other hand, when we compare how much time is needed to compute the rankings (Fig. 7b), we can identify more groups. For example, the fastest group is formed by the Relief scores with $m \in \{25\%, 50\%\}$ and the MTR rankings of the Genie3 and Symbolic scores. The

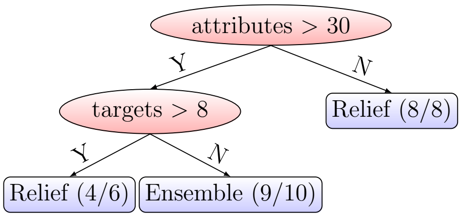


Fig. 8 The PCT constructed from the meta dataset, where Y and N denote Yes and No branches. In every leaf node, the majority target statistics are shown, e.g., in the leaf *Relief (4/6)*, there are 6 examples. On 4 of them, Relief performs better

Table 6 Cross-validated performance of the meta decision tree, where y denotes true values and $\hat{y}$ denotes predicted values

y \ $\hat{y}$	Ensemble	Relief
Ensemble	9	2
Relief	3	10

slowest ones are the STR rankings from the ensemble-based scores, together with the MTR ranking of the RF score where bagging is used.

Since our efficient implementation allows for computing all Relief scores at once, the induction time for these scores depends only on the number of iterations, hence the shortened descriptions of the MTR and $\overline{\text{STR}}$ rankings (only the number of iterations is shown in Fig. 7b).

We next perform a meta-learning analysis of the obtained results. If we want to get a recommendation on using an appropriate feature algorithm for a given new dataset, the average ranks in Fig. 7a do not provide a unique recommendation, since there are too many candidates. Also, we want to understand when the ensemble-based rankings perform better than the Relief-based rankings, considering some basic properties of the datasets. To do so, we describe each dataset as a $(\boldsymbol{x}, y)$ pair, where the descriptive part $\boldsymbol{x}$ consists of the basic dataset statistics shown in Table 3: number of examples, features and targets. The target part for a particular dataset was $y =$ Ensemble if an ensemble method performed better on this dataset, and $y =$ Relief if a Relief method was better. From this meta dataset, we build a single PCT, which is shown in Fig. 8. It seems that the number of examples does not influence the better performing score. We can conclude that the Ensemble group of rankings performs better for datasets that have a higher descriptive dimension $D > 30$ and a lower number of targets ($T \leq 8$).

The accuracy of the tree is 79.2% and was estimated via tenfold cross validation. The corresponding confusion matrix is shown in Table 6.

5.4 Absolute performance of rankings

So far, the statistical analysis shown operated only with the ranks of the algorithms on different datasets and we have not shown yet any actual performance figures in terms of $\overline{\text{RRMSE}}$. The purpose of analysis of these results is twofold: we can explore (i) whether the previously discovered statistically significant differences are also practically relevant, and (ii) which datasets represent harder and which datasets represent easier feature ranking (and predictive modelling) problems.

Inspecting Fig. 7a, we can see that the best average rank among the ensemble-based scores belongs to the $\overline{\text{STR}}$-Genie3 rankings, computed from the random forest ensemble. The best ranked score from Relief group is MTR-Relief ranking, computed with $m = 100\%$ iterations

Table 7 The performance (in terms of $\overline{\text{RRMSE}}$) of the baseline 5NN model and the weighted models that correspond to the best rankings from the ensemble and the Relief group of scores

	Baseline 5NN	Best ensemble score Genie3-RF-STR	Best relief score It100Neigh15MTR
ATP1d	0.467	0.45	0.452
ATP7d	0.664	0.653	0.678
collembola	1.006	0.992	1.006
EDM	0.784	0.776	0.759
ENB	0.306	0.155	0.148
Forestry Kras	0.588	0.59	0.587
Forestry LIDAR IRS	0.425	0.347	0.404
Forestry LIDAR Landsat	0.535	0.49	0.528
Forestry LIDAR Spot	0.414	0.367	0.402
jura	0.836	0.749	0.762
OES10	0.6	0.588	0.598
OES97	0.535	0.532	0.534
osales	0.946	0.868	0.932
RF1	0.239	0.228	0.251
RF2	0.509	0.313	0.529
SCM1d	0.306	0.302	0.301
SCM20d	0.318	0.32	0.317
scpf	1.059	0.989	0.934
sigmeareal	0.977	0.906	0.912
sigmeasim	0.141	0.116	0.052
slump	0.795	0.766	0.756
soil quality	0.732	0.731	0.729
vegetation condition	0.662	0.665	0.68
water quality	0.985	0.983	0.98

and $k = 15$ neighbors. We present performance figures for these two rankings, together with the baseline non-weighted 5NN in Table 7.

We can see that sometimes, the baseline 5NN performs considerably worse than at least one of the other two predictive models. For example, this is the case for the ENB, RF2 and sigmeasim datasets. Interestingly, ENB and sigmeasim have only 8 and 11 features respectively. For the considered methods, these datasets present not that hard feature ranking problems. On the other hand, the statistically significant differences are sometimes not practically relevant, e.g., the relative decrease of the $\overline{\text{RRMSE}}$ after computing the ranking on the SCM datasets is quite small. These datasets present harder feature ranking problems. However, as evident from the statistical tests, the baseline is quite consistently the worst performing of the three predictive models.

6 Conclusion

In this work, we focus on the task of feature ranking for multi-target regression. More specifically, we propose methods for feature ranking that exploit the potential relatedness among multiple targets to produce a single feature ranking valid for all of the targets. Namely, we propose methods for feature ranking that are based on ensemble learning and on the Relief family of feature ranking algorithms. In the former group of methods, we consider three ensemble learning methods (bagging, random forests and extremely randomized trees) coupled with three scores (Genie3, Random Forest and Symbolic). The ensemble methods use predictive clustering trees for MTR as base predictive models. The latter group considers an extension of RReliefF towards the task of MTR (MTR-Relief). We have also proposed to use the 'single' target variants of the proposed methods in the context of MTR by averaging the importance scores obtained for each of the targets.

We analyze the proposed methods along several dimensions. First, we perform a theoretical computational complexity analysis of all of the methods. Next, we complement the computational complexity analysis with runtime comparisons. Furthermore, we parametrize the proposed methods. Finally, we benchmark the performance of the proposed methods on 24 datasets by using the feature importances as weights in 5-nearest neighbors (5NN) prediction.

The comprehensive experimental evaluation reveals the following. First, five out of the six ensemble-based ranking methods yield relevant feature rankings when they are computed using the most appropriate tree ensemble, the only exception being the $\overline{\text{STR}}$ ranking with Random Forest score. The evaluation identified two best ranking methods (in terms of performance and efficiency): Genie3 and Symbolic scores coupled with random forests.

Second, with the Relief rankings, only a single setting was found yielding statistically significant differences in performance: the MTR version of Relief with $m = 100\%$ and $k = 15$. When we compared the top 12 ranking algorithms, there were no statistically significant differences among them. However, further analysis revealed that this is due to the properties of datasets, since we can rather accurately determine which of the two groups (ensemble-based or Relief-based) of algorithms will perform better on a given dataset.

Third, the analysis shows that the MTR versions of ensemble-based rankings are statistically significantly faster than their $\overline{\text{STR}}$ counterparts, but not worse in terms of quality. In the case of Relief, there are no differences. All in all, we would suggest using the Genie3 and the Symbolic scoring coupled with random forest ensemble method, since they both yield relevant rankings and are fast enough.

There are several directions for the future work. First, we plan to consider a different approach to defining the distance measure in MTR-Relief (or even in RReliefF). We could use a distance based on the probability densities $f(x_i)$ and $f(y_j)$, since this is the usual continuous analogue of the probabilities $P(x_i = x)$ in the discrete case. Next, we will extend the proposed methods to other structured output prediction tasks, such as multi-label classification, and hierarchical multi-label classification. Furthermore, we will investigate the influence of the ensemble size on the produced feature rankings (thus further reducing the computational cost). Finally, we will develop similar approaches for learning feature rankings for MTR tasks on data streams and for semi/un-supervised MTR tasks.

Acknowledgements We acknowledge the financial support of the Slovenian Research Agency via the Grants P2-0103 and a young researcher Grant to MP, as well as the European Commission, through the Grants MAESTRA (Learning from Massive, Incompletely annotated, and Structured Data) and HBP (The Human Brain Project), SGA1 and SGA2. SD also acknowledges support by Slovenian Research Agency (via Grants J4-7362, L2-7509, and N2-0056), the European Commission (Project LANDMARK) and ARVALIS (Project BIODIV). The computational experiments presented here were executed on a computing infrastructure from the Slovenian Grid (SLING) initiative.

References

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, *57*(1), 289–300.

Blockeel, H. (1998). *Top-down induction of first order logical decision trees*. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium.

Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A survey on multi-output regression. *Data Mining and Knowledge Discovery*, *5*(5), 216–233.

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

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

Brobbey, A. (2015). Variable selection in multivariate multiple regression. Master's thesis, Memorial University of Newfoundland, St John's, NL, Canada.

Burnham, A. J., MacGregor, J. F., & Viveros, R. (1999). Latent variable multivariate regression modeling. *Chemometrics and Intelligent Laboratory Systems*, *48*(2), 167–180.

Cunningham, P., & Delany, S. J. (2007). k-Nearest Neighbour Classifiers. Technical report, University College Dublin, Dublin, Ireland.

Demšar, D., Debeljak, M., Džeroski, S., & Lavigne, C. (2005). Modelling pollen dispersal of genetically modified oilseedrape within the field. In *Proceedings of annual meeting of the Ecological Society of America*.

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

Džeroski, S., Demšar, D., & Grbović, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, *13*, 7–17.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, *3*(3), 209–226.

Geurts, P., Erns, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *36*(1), 3–42.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford: Oxford University Press.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Han, Z., Liu, Y., Zhao, J., & Wang, W. (2012). Real time prediction for converter gas tank levels based on multi-output least square support vector regressor. *Control Engineering Practice*, *20*(12), 1400–1409.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*, 993–1001.

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

Kaggle. (2012). Kaggle: Online product sales. https://www.kaggle.com/c/online-sales. Accessed June 12, 2018.

Kaggle. (2013). Kaggle: See click predict fix. https://www.kaggle.com/c/see-click-predict-fix. Accessed June 12, 2018.

Kampichler, C., Džeroski, S., & Wieland, R. (2000). 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*, *32*(2), 197–209.

Karalič, A., & Bratko, I. (1997). First order regression. *Machine Learning*, *26*(2–3), 147–176.

Kira, K., & Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the tenth national conference on artificial intelligence* (pp. 129–134). AAAI Press, San Jose, California.

Kocev, D., & Džeroski, S. (2013). Habitat modeling with single- and multi-target trees and ensembles. *Ecological Informatics*, *18*, 79–92.

Kocev, D., Džeroski, S., White, M., Newell, G., & Griffioen, P. (2009). Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modelling*, *220*(8), 1159–1168.

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

Petković, M., Džeroski, S., & Kocev, D. (2017). Feature ranking for multi-target regression with tree ensemble methods. In Yamamoto, A., Kida, T., Uno, T., & Kuboyama, T. (Eds.), *Discovery science* (pp. 171–185). Berlin: Springer.

Robnik-šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning Journal*, *55*, 23–69.

Sanchez-Fernandez, M., de-Prado-Cumplido, M., Arenas-Garcia, J., & Perez-Cruz, F. (2004). Svm multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE Transactions on Signal Processing*, *52*(8), 2298–2307.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., & Vlahavas, I. (2016). Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, *104*(1), 55–98.

Stańczyk, U., & Jain, L. C. (Eds.). (2015). *Feature selection for data and pattern recognition. Studies in computational intelligence*. Berlin: Springer.

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

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

Tsanas, A., & Xifara, A. (2012). Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, *49*, 560–567.

Tsoumakas, G., & Katakis, I. (2007). Multi label classification: An overview. *International Journal of Data Warehouse and Mining*, *3*(3), 1–13.

Tuia, D., Verrelst, J., Alonso, L., Perez-Cruz, F., & Camps-Valls, G. (2011). Multioutput support vector regression for remote sensing biophysical parameter estimation. *IEEE Geoscience and Remote Sensing Letters*, *8*(4), 804–808.

Wettschereck, D. (1994). *A study of distance based algorithms*. Ph.D. thesis, Oregon State University, Corvallis, OR.

Yeh, I.-C. (2007). Modeling slump flow of concrete using second-order regressions and artificial neural networks. *Cement and Concrete Composites*, *29*, 474–480.