Model Selection for Multi-Directional Ensemble of Regression and Classification Trees [⋆]

Evgeniya Korneva and Hendrik Blockeel

KU Leuven, Belgium
{evgeniya.korneva,hendrik.blockeel}@cs.kuleuven.be

Abstract. Multi-directional ensembles of Classification and Regression treeS (MERCS) extend random forests towards multi-directional prediction. The current work discusses different strategies of induction of such a model, which comes down to selecting sets of input and output attributes for each tree in the ensemble. It has been previously shown that employing multi-targets trees as MERCS component models helps reduce both model induction and inference time. In the current work, we present a novel output selection strategy for MERCS component model that takes relatedness between the attributes into account and compare it to the random output selection. We observe that accounting for relatedness between targets has a limited effect on performance and discuss the reasons why it is inherently difficult to improve the overall performance of a multi-directional model by altering target selection strategy for its component models.

Keywords: versatile models · ensemble learning · multi-directional models · multi-task learning · decision trees

1 Introduction

In practice, data analysis often happens in two steps. First, a task-specific machine learning model is built. Then, this model is used for inference. However, in many cases the user may want to perform different prediction tasks on the same data. A typical example of such a setting is missing data imputation. Moreover, sometimes not all of the prediction tasks are known beforehand. In this case, it appears beneficial to learn a single multi-purpose model of the dataset that can be re-used to solve various tasks rather then a number of different special-purpose models. An example of such a versatile model of data is MERCS.

MERCS stands for **M**ulti-directional **E**nsemble of **C**lassification and **R**egression tree**S**. The method is first introduced in [13]. The term *multi-directional* in its name refers to the fact that the resulting model is capable of answering any possible query of the data of the form "predict $\boldsymbol{Y}$ from $\boldsymbol{X}$", as opposed to conventional

[⋆] The research is supported by the Research foundation - Flanders (project G079416N, MERCS) *uni-directional* models, where the sets of the input and target variables are fixed.

MERCS essentially extends well-known random forests to multi-directional prediction by constructing ensembles of (possibly multi-target) decision trees that have different sets of input and output variables. More specifically, let $\mathcal{RF}(\mathbb{I}, \mathbb{O})$ denote a random forest that predicts a set of output variables $\mathbb{O}$ from a given set of input attributes $\mathbb{I}$. Note that $\mathbb{O}$ can contain more than one attribute. Then, for a given dataset D with attributes $\mathbb{A} = \{A_1, A_2, \ldots, A_m\}$, its MERCS model $\mathcal{M}(D)$ is formally defined as:

$$\mathcal{M}(D) = \{\mathcal{RF}_i,\ i = 1...n | \mathcal{RF}_i = \mathcal{RF}(\mathbb{I}_i, \mathbb{O}_i),\ \mathbb{I}_i \subset \mathbb{A},\ \mathbb{O}_i \subset \mathbb{A},\ \mathbb{I}_i \cap \mathbb{O}_i = \emptyset\}$$

One of the key research questions is how to decide *which* trees exactly should be included in the model, i.e., how to determine the sets of input and output attributes, $\mathbb{I}_i$ and $\mathbb{O}_i$ respectively, for every component random forest $\mathcal{RF}_i$ in the ensemble, so that the resulting model allows for prediction in any direction?

As pointed out in [13], this is already possible by learning a set of conventional single-target trees (namely, by learning one random forest per attribute in a dataset). Experimental results show, however, that employing multi-target trees as MERCS component models leads to much faster model induction and inference comparing to the straightforward baseline approach. Performance-wise, multi-target tree-based models are often inferior to their baseline counterpart. However, the decrease in predictive accuracy is considered marginal taking into account the gain in speed.

In the experiments conducted in [13], attributes are grouped into target sets randomly, while intuitively it seems that performance of the resulting model can be improved if targets predicted jointly by the same component model are somehow similar, or *related*, because in this case the multi-task model is able to to exploit potential dependencies between them [7]. But how to define the notion of *relatedness* and how to determine which attributes in the dataset are related and should be therefore predicted together?

The goal of this paper is twofold. First, we apply MERCS to regression task and compare the effect of using multi-target component models to that previously observed in the classification case. Second, we investigate the possibility to improve the predictive performance of the MERCS model consisting of multi-target component models. To that end, we propose a novel MERCS induction strategy that takes relatedness between the attributes into account. We namely discuss two different ways of measuring the relatedness.

The rest of the paper is organized as follows. Section 2 formalizes different model selection strategies for MERCS. The results of empirical evaluation of the strategies under consideration are presented in Section 3 and discussed in Section 4. Final section positions MERCS in the context of other state-of-the art versatile models and discusses the connection between models selection for MERCS and the problem of identifying related tasks in the field of multi-task learning.

2 Related Work

This section discusses the relation between MERCS and multi-task learning.

Multi-task learning (also known as multi-output, multi-target or multi-objective prediction) refers to joint prediction of multiple variables by the same model. A special case of multi-task learning is multi-label classification, when a set of binary labels is assigned to every example.

In the multi-task setting, one often faces the challenge of identifying which tasks can benefit from being learned together, and which should be treated separately. This question appears to be closely related to determining which attributes should be predicted together by the same component model in MERCS.

Breskvar et al. propose random output selection for solving multi-target classification and regression problems, which is extended towards random target grouping strategy for MERCS [2] [3]. However, generalizing more sophisticated task grouping approaches proposed in the multi-task learning field towards the multi-directional setting is more challenging.

The biggest difference between MERCS and conventional multi-target models is that the latter are still uni-directional. Indeed, such models can only predict a predefined set of targets and, as opposed to MERCS, are not capable of answering random query of data. In addition, in multi-task learning, one often aims at improving prediction performance of a multi-task model with respect to some predefined principal task, while MERCS is not a task-specific model and is expected to perform reasonably well over all possible tasks.

For instance, Piccart et al. explore inductive transfer in the context of decision tree learning. Pairwise transfer between the targets is measured as the gain in predictive performance that two-target model yields over a single-target one. The authors then propose the algorithm that, given one main target, aims at identifying its best *support set*, that is, the best subset of auxiliary targets that can be added to the model so that the resulting multi-output model shows the best predictive performance with respect to the main target.
Applying this technique to MERCS induction will not help to reduce the size of the model compared to the baseline approach: there will still be km trees in the resulting ensemble. In addition, when the number of attribute in dataset is high, computing inductive transfer as defined in [9] for all possible combinations of targets becomes computationally expensive.

An alternative clustering-based approach to discover the relation between binary classification tasks is proposed in [11]. First, empirical measure of mutual relatedness between tasks is estimated. Then, the tasks are clustered into classes of mutually related tasks.
The idea of applying clustering techniques to split dataset attributes into disjoint target sets based on their mutual relatedness seems applicable in the MERCS context. However, such a clustering, unless constrained, can result in highly unbalanced clusters with many attributes in some of them and very few in the others. Component models, however, will likely perform poorly if too many targets are needed to be predicted from too few inputs.

3 Selection Strategies for MERCS

Baseline approach

The most straightforward way to build a versatile tree-based model is to learn a random forest of k single-target trees for every single attribute i in the dataset:

$$\mathcal{M}(D) = \mathcal{RF}_i(\mathbb{A} \setminus \{A_i\}, \{A_i\}), \quad i = 1...m$$

This approach results therefore in the MERCS ensemble of km trees in total. For large m and typical k (e.g., $k = 30$), such a simple baseline strategy leads to quite a large model.

Predicting several targets simultaneously by the same component models is a reasonable solution to that problem.

Random grouping of targets

When employing multi-target trees as MERCS component models, one can neglect possible relations between the attributes and group them into sets of targets randomly.

Let us partition the m attributes of a data set into m/p disjoint subsets $\mathbb{A}^j, j = 1, ..., m/p$, of p targets each. If an ensemble of k multi-target trees is learned for each subset, the resulting MERCS model contains $n = \frac{km}{p}$ trees, rather than km:

$$\mathcal{M}(D) = \mathcal{RF}_j(\mathbb{A}^j, \mathbb{A} \setminus \mathbb{A}^j), \quad j = 1...m/p$$

Therefore, even though such an approach does not take into account relatedness between the target attributes, one can certainly expect to observe benefits in terms of speed, since the resulting MERCS model is smaller in size. Similarly to the baseline approach, each attribute is predicted by one random forest, but each random forest simultaneously predicts p targets rather than just one.

Heuristic-based grouping of targets

Suppose we have a way to measure the relatedness between any two attributes A_i and A_j in the data set. Let us denote it as r_{ij}.

We will assign attributes as targets to MERCS component models one by one in a greedy manner. At each step, attribute A_i is assigned to the most related model, i.e., the model that already predicts targets that are highly related to the current attribute. Just as in the random grouping case, the number p of targets predicted by each model (and, therefore, the total number of models n) are specified in advance.

More specifically, we propose assessing the relatedness between the attribute A_i and MERCS component model $\mathcal{RF}_k$ by computing relatedness score defined as follows:

$$\mathcal{R}(A_i, \mathcal{RF}_k) = \begin{cases} 1, & \text{if } |\mathbb{O}_k| = 0, \\ \dfrac{\frac{1}{|\mathbb{O}_k|} \sum\limits_{j: A_j \in \mathbb{O}_k} r_{ij}}{\frac{1}{m} \sum\limits_{j=1}^{m} r_{ij}}, & \text{if } 0 < |\mathbb{O}_k| < p \\ 0, & \text{if } |\mathbb{O}_k| = p. \end{cases}$$

This definition basically means the following:

- If the model does not have any targets assigned yet, the corresponding relatedness score is set to 1.
- If the model already has some targets assigned (e.g. 0 ¡ $|\mathbb{O}_k| < p$), the relatedness score is computed as the ratio of the average relatedness of the current attribute to the model's targets and its average relatedness to all of the attributes in the data set.
  The score will therefore be greater than 1 if model's targets appear more related than attributes on average (and it therefore makes sense to group them), and less than 1 if the opposite is true (and it is therefore better to assign the current attribute to a new or a more related model).
- If the model cannot take any more targets ($|\mathbb{O}_k| = p$), the relatedness score is set to zero.

The final question is how to asses relatedness between any pair of attributes itself (i.e., how to obtain r_{ij}). There are two possible ways in which one can interpret the notion of similarity between the learning tasks.

The first one is to compare the values of the target attributes themselves: if those are somewhat interconnected, the learning tasks are related. Arguably the most straightforward way to assess statistical dependency of two variables is by computing **correlation coefficient**. While expressing the degree of linear dependency between the variables, it can be a reasonable proxy for more complex measures of relation (e.g., mutual information score) since in practice even more complex relationships between two variables often have fairly linear component.

Another way to assess the relatedness between two learning tasks is to see it as a fact that they share the same input features, or *find the same input features important.* [4]
The internal estimates made by random forest can be easily used for measuring **feature importance** for predicting a particular target. [1]One can construct a matrix $C = \{c_{ij}\}$, where c_{ij} indicates the importance of the attribute A_j for predicting A_i. The importance scores c_{ij} are obtained by learning a single-target Random Forest for attribute A_i. To speed up the process, we propose learning these random forest on a small random subsample (e.g., 30%) of the data. The correlation coefficient between the rows of the matrix C reflects how similar the corresponding attributes are in terms of what inputs they find important, or, in other words, how much the corresponding attributes are related to each other.

The fundamental difference between the two aforementioned ways of assessing relatedness is that the latter directly takes the needs of the learner (in this case, decision tree) into account, while the former only relies on the tasks themselves.

To sum up, there are four different ways to induce a MERCS model that are the following:

1. Learning single-target component models according to the baseline strategy.

2. Learning multi-target component models grouping attributes determining the target sets randomly.

3. Learning multi-target component models grouping correlated targets together.

4. Learning multi-target component models grouping targets with similar feature importance vectors together.

These strategies are empirically compared in the following section.

4 Experiments

The Data

[1] One of the ways of doing this is by calculating *mean decrease impurity* for each attribute. Every node in the trees in the forest corresponds to a binary test on a single attribute, and the locally optimal test is chosen based on the impurity measure. While learning a tree, one can estimate how much each input feature decreases the weighted impurity in a tree. The impurity decrease from each attribute test can be averaged over all trees. [1]

Dataset	**# instances**, n	**# attributes**, m	**Source**
andro	48	36	[10]
edm	767	10	[10], [12]
jura	358	18	[6], [10]
oes10	402	314	[10]
oes97	333	279	[10]
rf1	9004	72	[10]
scm20d	8965	77	[10]
slump	102	10	[10], [14]
wq	1059	30	[5]

Table 1. Data sets used in the experiments

Table 1 provides a short summary of the data sets used for experiments. The data is a collection of benchmarks data sets for multi-target regression problems. Only numerical attributes (e.g., those with more than 10 distinct values) were considered. The attributes have been scaled to fall in the range between 0 and 1.

Experimental Setting

We compare four different MERCS induction strategies formulated in the previous section in terms of induction and inference time, as well predictive performance [2] of the resulting MERCS models. To that end, 10-fold cross validation is performed.
Besides, when multi-target decision trees are employed, different number of target attributes per component model (parameter p) are considered. More specifically, component models predicting 10%, 20%, 40%, 60% and 80% of the attributes in the dataset are considered (p equals to $0.1, 0.2, 0.4, 0.6$ and 0.8 respectively).

Results

Speed Figures 1 and 2 show average model induction and inference time depending on the number of targets predicted simultaneously by each of the MERCS component models, as well as on the grouping strategy employed.
As expected, the more targets are predicted jointly, the faster the MERCS model can be learned. That is because the total number of trees to be learned decreases. The gain is especially noticeable for the datasets with a large number

[2] More specifically, the predictive accuracy is evaluated based on the performance on prediction tasks of the following form:

$$A_i \leftarrow \mathbb{A} \setminus \{A_i\}, \quad i = 1...m,$$

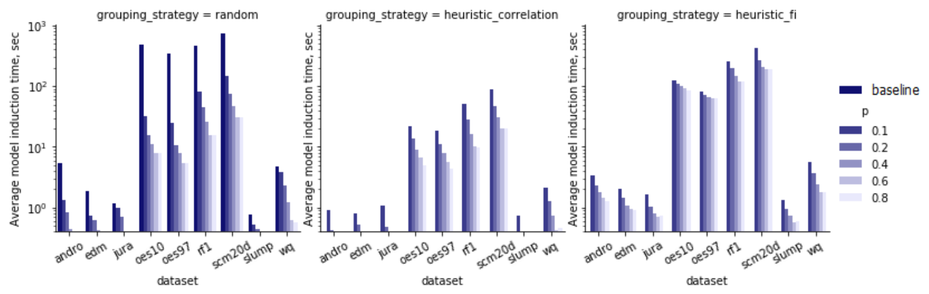

Fig. 1. The more targets are predicted by MERCS component models, the less trees need to be learned. Therefore, model induction becomes faster. However, computing heuristic function to take attribute relatedness into account slows it down.

of attributes. This goes in line with the findings presented in [13].

Grouping targets based on correlation between takes roughly the same time as when grouping them randomly. Indeed, correlation matrix is fast to compute. Obtaining feature importance scores for computing relatedness scores is, by contrast, computationally expensive. One can notice that inducing a MERCS model based on this grouping strategy can actually sometimes take longer than learning single-target component models for each of the attributes. However, if the number of attributes is very high (e.g., datasets `oes10, oes97, rf1, scm20d`), a speed up in induction time is still observed.

Interesting phenomena can be noticed on Figure 2. While inference generally becomes faster when multiple targets are predicted at once by the component models, obtaining predictions is slower when targets are grouped based on their relatedness. That means that the component tree models themselves become larger and more complex.

Performance To evaluate MERCS predictive performance, we asses the quality of individual attribute prediction by computing the corresponding root mean squared error (RMSE). We then order the four induction strategies from best to worst for each of the attribute of the dataset. The average rank associated with each of the strategy is presented on Figure 3. Furthermore, Figure 4 illustrates the distribution of the RMSEs depending on the induction strategy and parameter p for every dataset.

To begin with, unlike in the classification case considered in [13], models based on multi-target trees rarely outperform the baseline MERCS model that consists of single-target random forests for each of the attribute. The only example of a dataset for which employing multi-target trees in MERCS is actually beneficial is `rf1`. There, the MERCS model with targets grouped randomly outperforms

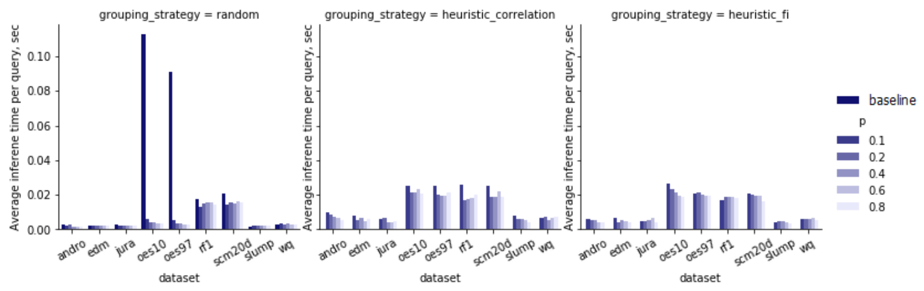

Fig. 2. Inference is generally faster when multiple attributes are predicted by a single component models. Grouping related attributes together results in more complex models, and obtaining predictions takes more time.

the baseline one for all the values of the parameter p under consideration.

The effect of taking relatedness of the attributes into account when inducing a MERCS model is limited. On the one hand, for most of the datasets, there exist at least one combination of the value of p and heuristic function that on average leads to a more accurate performance with respect to all of the attributes. On the other hand, however, in most cases the difference in the performance appears to be insignificant. For two datasets, namely `edm` and `oes97`, random grouping always works at least as well as more sophisticated heuristic-based approaches.

When comparing the two heuristic-based grouping strategies with each other, one can notice that grouping based on similarity in feature importance scores results in more accurate models for most of the datasets. The only exceptions are datasets `wq` and `edm`: for their attributes, correlation-based grouping is superior.

The next section discusses why employing a more sophisticated grouping strategy does not result in a significant improvement in predictive performance.

5 Discussion

There are a number of possible reasons why grouping attributes into target sets in a 'smart' way does not seem to improve the performance of the resulting MERCS model. They are namely the following.

- *Predicting several targets simultaneously can be beneficial for some of them and detrimental for others.*
  Any combination of the number of outputs p and grouping strategy can result in a MERCS model that is better in solving some prediction tasks and worse in solving others. This observation leads to a number of important remarks.

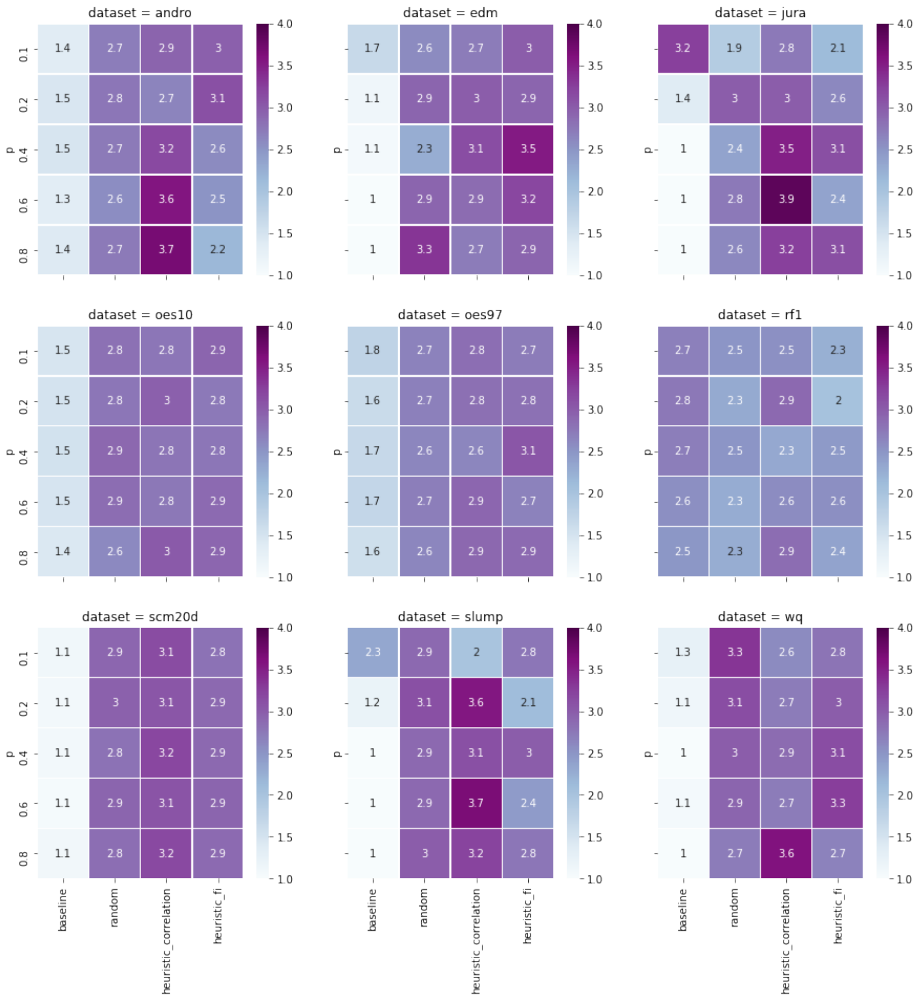

Fig. 3. `rf1` is the only dataset where MERCS based on multi-target component models outperforms the baseline model for all the number of targets considered. Grouping attributes based on similarity of their feature importance vectors results in more accurate models than when using correlation coefficient as relatedness measure.

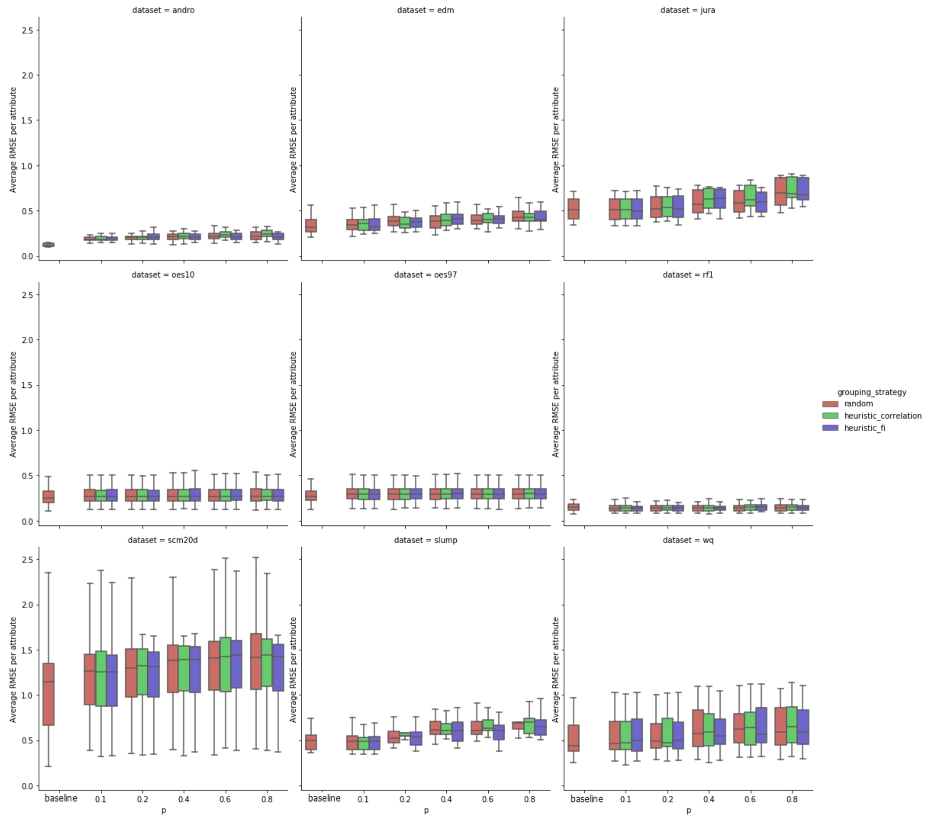

Fig. 4. Employing heuristic to group related attributes together has a limited effect on predictive performance of the resulting MERCS model. For most of the datasets, there is a combination of heuristic function and value of the parameter p that helps improve the performance of the resulting model compared to the random grouping case. However, the improvement appears to be marginal.

First, average performance will stay roughly the same (this is exactly what has been observed in the results of the experiments conducted in the current paper).

Second, none of the grouping strategies may result in a clearly superior MERCS model because it may be impossible to achieve a general improvement in predictive accuracy for every single prediction task: every new grouping is better for predicting in some directions and worse for the others.

Third, correlation coefficient between two variables is symmetric and therefore does not reflect the asymmetric nature of inductive transfer. Thus, using it as the measure of relatedness will result in suboptimal models. [4] [9]

– *Models multiple predicting related targets overfit.*

  Predicting several targets simultaneously even if they are unrelated can still be beneficial because additional targets act as regularization terms, forcing the resulting model to perform well on all the tasks and therefore making it less prone to overfitting [4], [8]. This effect disappears if targets are very similar. This observation can be confirmed by the fact that, according to our observations, inference becomes slower when the related attributes are grouped together, meaning that the models become more complex.

– *There are possibly no groups of highly related attributes in the dataset.*

  When looking for a way of grouping similar or related attributes together, one explicitly assumes that such groups of highly related attributes exist in the dataset, which may not always hold in practice. When the magnitude of relatedness is somewhat similar between all the attributes, such a grouping is not better than a random one.

– *Predictors in the component models happen to be unrelated to targets.*

  One can see correlated attributes as related ones and therefore better predicted together. However, they can also be good predictors for each other. If the value of p is too high, it can be the case that all the highly correlated attributes in the dataset are grouped together to be predicted by a single component model with he rest of the attributes as predictors. However, since by construction of the model those are less related to the targets, the performance of such a model will be poor.

– *Feature importance scores obtained from a single-target random forest may no longer be reliable when a multi-target one is constructed.*

  Indeed, we group the attributes in a target set for a multi-output component model in such a way that all of them find the same subset of input features important, i.e., splits on these same inputs occur and significantly decrease node impurity when constructing corresponding single-target trees. However, these splits may happen on completely different values. When we then predict the set of targets together, however, the split in a multi-target tree must happen on the same one, and it appears difficult to find a split that would be informative with respect to all of the targets, which results in larger, more complex trees and worse prediction performance.

On balance, employing multi-target random forests helps reduce the size of the final MERCS model in terms of the number of trees that need to be learned. That results in a significant speed-up in model induction, which is the main motivation to use multi-target component models in MERCS. However, there is a trade-off between speed and quality of the predictions: the more targets are predicted jointly, the faster the learning happens, and, typically, the worse the predictions are.

Employing 'smart' grouping strategies may help improve the predictive performance of the multi-target component models. Based on experimental results and aforementioned remarks, we can conclude that grouping attributes based on similarity of the features they find important is more appropriate than relying on the correlation coefficient between the attributes themselves.
However, the optimal value of the parameter p is not known in advance, which is the major shortcoming of the proposed approach. Besides, the resulting MERCS models almost never outperforms the baseline counterpart, which is a MERCS model based on single-target trees. Nevertheless, obtaining feature importance scores is computationally expensive, which can mitigate the gain in learning time if the number of attributes in a dataset is not very high.

To sum up, when applying MERCS in practice, one should rather stick to the baseline strategy, especially if the number of attributes in the dataset is low. If speed is a priority, opting for random grouping of targets with a relatively low number of targets per component model (e.g., 10% of the total number of attributes) is reasonable, since it can still allow for much faster induction and comparable predictive performance to those in the baseline case. Employing heuristic-based induction strategy with feature importance scores as the measure of relatedness between the attributes can be beneficial for the performance, but extra time is needed to determine the best value of the parameter p.

References

1. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
2. Breskvar, M., Kocev, D., Džeroski, S.: Multi-label classification using random label subset selections. In: International Conference on Discovery Science. pp. 108–115. Springer (2017)
3. Breskvar, M., Kocev, D., Džeroski, S.: Ensembles for multi-target regression with random output selections. Machine Learning **107**(11), 1673–1709 (2018)
4. Caruana, R.: Multitask learning. In: Learning to learn, pp. 95–133. Springer (1998)
5. Džeroski, S., Demšar, D., Grbović, J.: Predicting chemical parameters of river water quality from bioindicator data. Applied Intelligence **13**(1), 7–17 (2000)
6. Goovaerts, P.: Geostatistics for natural resources evaluation. Oxford University Press on Demand (1997)

7. Kocev, D., Vens, C., Struyf, J., Deroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition **46**(3), 817–833 (Mar 2013). https://doi.org/10.1016/j.patcog.2012.09.023, `http://linkinghub.elsevier.com/retrieve/pii/S003132031200430X`
8. Paredes, B.R., Argyriou, A., Berthouze, N., Pontil, M.: Exploiting unrelated tasks in multi-task learning. In: Artificial Intelligence and Statistics. pp. 951–959 (2012)
9. Piccart, B., Struyf, J., Blockeel, H.: Empirical asymmetric selective transfer in multi-objective decision trees. In: Discovery Science. pp. 64–75. Springer (2008)
10. Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., Vlahavas, I.: Multi-target regression via input space expansion: treating targets as inputs. Machine Learning **104**(1), 55–98 (2016). https://doi.org/10.1007/s10994-016-5546-z, `http://dx.doi.org/10.1007/s10994-016-5546-z`
11. Thrun, S., O'Sullivan, J.: Discovering structure in multiple learning tasks: The tc algorithm. In: ICML. vol. 96, pp. 489–497 (1996)
12. Tsanas, A., Xifara, A.: Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. Energy and Buildings **49**, 560–567 (2012)
13. Van Wolputte, E., Korneva, E., Blockeel, H.: MERCS: Multi-directional ensembles of regression and classification trees. In: AAAI Conference on Artificial Intelligence, North America (2018), `https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16875`
14. Yeh, I.C.: Modeling slump flow of concrete using second-order regressions and artificial neural networks. Cement and Concrete Composites **29**(6), 474–480 (2007)