

Tree-based methods for online multi-target regression

Aljaž Osojnik^{1,2}  · Panče Panov¹ · Sašo Džeroski^{1,2,3}

Received: 3 July 2016 / Revised: 4 January 2017 / Accepted: 18 April 2017
© Springer Science+Business Media New York 2017

Abstract Methods that address the task of multi-target regression on data streams are relatively weakly represented in the current literature. We present several different approaches to learning trees and ensembles of trees for multi-target regression based on the Hoeffding bound. First, we introduce a local method, which learns multiple single-target trees to produce multiple predictions, which are then aggregated into a multi-target prediction. We follow with a tree-based method (iSOUP-Tree) which learns trees that predict all of the targets at once. We then introduce iSOUP-OptionTree, which extends iSOUP-Tree through the use of option nodes. We continue with ensemble methods, and describe the use of iSOUP-Tree as a base learner in the online bagging and online random forest ensemble approaches. We describe an evaluation scenario, and present and discuss the results of the described methods, most notably in terms of predictive performance and the use of computational resources. Finally, we present two case studies where we evaluate the introduced methods in terms of their efficiency and viability of application to real world domains.

Keywords Tree-based methods · Multi-target regression · Data streams

✉ Aljaž Osojnik
aljaz.osojnik@ijs.si

Panče Panov
pance.panov@ijs.si

Sašo Džeroski
saso.dzeroski@ijs.si

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

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

³ Centre of Excellence for Integrated Approaches in Chemistry and Biology of Proteins, Jamova cesta 39, Ljubljana, Slovenia

1 Introduction

A common approach of solving complex data mining tasks is to transform them into simpler tasks, and then use known methods that solve the simple tasks. In this context, problem transformation approaches have been used to address a large variety of predictive modelling tasks, such as multi-label classification (MLC) and multi-target regression (MTR). A multi-label classification task can thus be transformed into a collection of binary classification tasks (Tsoumakas and Katakis 2007), while a multi-target regression task, which deals with predicting multiple numeric variables simultaneously, can be decomposed into several single-target regression tasks (Struyf and Dzeroski 2005; Kocev et al. 2013).

However, there are methods that forego the reduction to simpler tasks and address the complexity head-on. For example, in the case of the task of multi-target regression, methods that consider and predict all of the continuous targets at once have received considerable coverage in the literature (Struyf and Dzeroski 2005; Kocev et al. 2013). Almost exclusively, though, these methods have been introduced in the batch setting.

Recently, the streaming setting is becoming more and more prominent, due to the ever increasing presence of the Big Data paradigm. The streaming setting emphasizes several of the Big Data characteristics, i.e., the “V”s of Big Data. More specifically, streaming methods need to address Velocity – data arriving with high speed; Volume – potentially unbounded number of data instances; and Variability – potential changes in the data itself. Hence, it is of extreme importance to take these aspects into consideration when developing a method to be used in the streaming setting.

Methods that address the task of multi-target regression in a streaming setting are few and far between (e.g., Ikonmovska et al. (2011a)), especially those that predict all of the targets at the same time. In our early work, we introduced a new tree-based method, capable of addressing the task of multi-target regression by predicting all of the targets at the same time (Osojnik et al. 2016a). We performed a preliminary comparison to the simpler problem transformation approach of using a single-target tree-based method in a streaming setting and to an ensemble method, by using the proposed tree method as a base model in an online bagging ensemble setting.

In this paper, we build upon our early work. More specifically, we first discuss our adaptation of the local FIMT-DD method, which uses single-target trees for each target separately. Next, we present in depth our tree-based method for online MTR, named iSOUP-Tree, focusing on the heuristic function used in the tree induction process and the selection of the best split. The proposed method uses a single tree to predict all of the targets. Moreover, we introduce an option tree extension of our method, iSOUP-OptionTree, as well as the ensemble methods of online bagging and online random forests, which both use iSOUP-Trees as base models. Furthermore, we experimentally compare all of the methods with a focus on the predictive performance and the trade-off between the predictive performance and the consumption of resources. Finally, we present two case studies which we evaluate the introduced methods in terms of their efficiency and viability of application to real world domains.

The structure of the paper is as follows. First, we present the background and related work (Section 2). Next, we present several tree-based approaches for multi-target regression on data streams (Section 3). Furthermore, we present the research questions we address and the experimental design employed to answer them (Section 4). Finally, we conclude with a discussion of the results (Section 5), followed by two case studies from real world domains (Section 6), conclusions, and directions for further work (Section 7).

2 Background and related work

In essence, we can look at the task of multi-target regression as an extension of the single-target regression task. In the latter, only one continuous variable needs to be predicted. The task of multi-target regression (MTR) deals with predicting multiple numeric variables simultaneously, or, formally, with making a prediction \hat{y} from \mathbb{R}^n , where n is the number of targets, for a given instance \vec{x} from an input space X .

Generally, the quickest way of solving a complex task, such as multi-target regression, is to transform it into a set of simpler tasks that we know how to solve. In the case of multi-target regression, specifically, this is achieved by training a regressor for each of the targets separately, essentially resulting in a collection of regressors. The other option for addressing the task of multi-target regression is to produce a regressor which gives predictions for all of the targets simultaneously.

To distinguish between these two approaches we refer to them as local and global, respectively (Silla and Freitas 2011). Specifically, a method that uses one regressor per target is using the *local approach*, while a method that uses one regressor to predict all of the targets simultaneously is using the *global approach*.

Recent studies show, that in the batch case, the global approaches outperform the local ones (Kocev et al. 2013). Global methods tend to (implicitly) exploit the dependencies between the targets. The global approach has been considered in the batch setting by Struyf and Dzeroski (2005). In addition, Appice and Dzeroski (2007) proposed a method for step-wise induction of multi-target model trees.

Unlike the batch context, where a fixed and complete dataset is given as input to a learning method, the streaming context presents several constraints that a stream learning method must consider. The most relevant are (Bifet and Gavaldà 2009):

1. the examples arrive sequentially in a specific order;
2. the number of examples can be arbitrarily large;
3. the distribution of examples need not be stationary; and
4. after an example is processed, it is discarded or archived.

The fact that the distribution of examples is not assumed to be stationary means that methods learning in a streaming context should be able to detect and adapt to changes in the distribution (*concept drift*).

In the streaming context, some recent work has already addressed the task of single- and multi-target regression. Ikonmovska et al. (2011b) introduced an instance-incremental streaming tree-based single-target regressor (FIMT-DD) that utilizes the Hoeffding bound. This work was later extended by Ikonmovska et al. (2011a) for the task of multi-target regression (FIMT-MT). However, both of these methods had the drawback of ignoring nominal input attributes. Recently, there has been some theoretical debate whether the use of the Hoeffding bound is appropriate (Rutkowski et al. 2013), however, a recent study by Ikonmovska and Gama (2015) has shown that in practice the use of the Hoeffding bound produces good results.

Shaker and Hüllermeier (2012) introduced an instance-based system for classification and regression (IBLStreams), which can be, in principle, used for the task of multi-target regression. Recently, Duarte and Gama (2014) addressed the task of multi-target regression from high-speed data streams by using adaptive model rules (AMRules) (Duarte et al. 2016).

3 Tree-based methods for multi-target regression on data streams

In this section, we present several tree-based methods for multi-target regression, which utilize the local approach, as well as the global approach. Tree-based methods are often used, as they generally provide good results in terms of predictive performance, while also yielding interpretable models.

First, we introduce an adaptation of the FIMT-DD method that uses adaptive models in the leaves and an implementation that facilitates the use of the method for the task of multi-target regression. Next, we present four tree-based methods for multi-target regression, which use the global approach. More specifically, we describe the single-tree method (iSOUP-Tree), the option tree method (iSOUP-OptionTree), and two ensemble methods – online bagging (iSOUP-Bagging) and online random forests (iSOUP-RandomForest).

3.1 Adaptation of the FIMT-DD method for multi-target regression

One of the best known single-target tree-based regressors in the stream setting is the FIMT-DD method (Ikonovska et al. 2011b). It is based on the Hoeffding bound, which allows the generalization of observations from small samples to the underlying distribution. Similarly to Hoeffding trees for classification (Domingos and Hulten 2000), FIMT-DD uses the Hoeffding bound to determine the best splits at each node of the regression tree.

We have re-implemented the FIMT-DD method in the Java-based MOA stream-mining framework (Bifet et al. 2010) and extended it to use adaptive models in the leaves, similarly as Duarte and Gama (2014). Specifically, each leaf of the tree contains a perceptron. The perceptron is a linear function of the values of the input attributes \vec{x} that produces the prediction, i.e., $\hat{y} = \vec{w} \cdot \vec{x} + b$, where \vec{w} and b are a learned weight vector and a constant, respectively.

In the original implementation of FIMT-DD (Ikonovska et al. 2011b), the perceptron was always used to make the prediction. However, the adaptive model records the errors of the perceptrons and compares them to the errors of the mean target predictors. They predict the value of the target by computing the average value of the target over the examples observed in a given leaf. In essence, each leaf has two learners, the perceptron and the target mean predictor. The prediction of the learner that (at a given point in time) has a lower error is then used as a final prediction.

To monitor the errors, we use the faded absolute error which is calculated as

$$fMAE_{\text{learner}}(m) = \frac{\sum_{j=1}^m 0.95^{m-j} |\hat{y}(j) - y(j)|}{\sum_{j=1}^m 0.95^{m-j}},$$

where m is the number of observed examples in a leaf, $\hat{y}(j)$ and $y(j)$ are the predicted and real value for the j -th example, respectively, and $\text{learner} \in \{\text{perceptron}, \text{targetMean}\}$. In essence, the faded error is weighted towards more recent examples. Intuitively, the numerator of the fraction is the faded sum of absolute errors, while the denominator is the faded count of examples. For example, the most recent (m -th) example contributes with a weight of 1, the previous example with weight 0.95, and the first example with weight 0.95^{m-1} . This places an emphasis on the more recent examples and generally benefits the perceptron, as we expect its errors to decrease as it learns the weight vector.

In addition to the adaptation of the FIMT-DD method, we have also implemented a meta-learning method in the MOA framework that creates a collection of single-target regressors, one regressor for each target, and combines their single-target predictions into

a multi-target prediction in real-time to facilitate the use of FIMT-DD for the task of multi-target regression. This combination is referred to as the **local FIMT-DD** method.

3.2 The iSOUP-Tree method

As noted earlier, the global approach has been shown to yield good predictive performance in the case of tree-based methods in the batch setting. This has motivated the introduction of global tree-based methods for data streams, i.e., the FIMT-MT method introduced by Ikonomovska et al. (2011a). The FIMT-MT method extends FIMT-DD by replacing the use of the variance reduction heuristic with the intra-cluster variance reduction heuristic, which captures some of the dependencies of the targets. However, one of the major downsides of the FIMT-MT method, is the fact that it completely ignores nominal input attributes.

We have extended the FIMT-MT method by adding the support for nominal input attributes. In addition, we have also proposed the use of this extension to address other structured output prediction tasks, such as the task of multi-label classification (Osojnik et al. 2016b). The newly proposed method is named **incremental Structured Output Prediction Tree (iSOUP-Tree)**.

3.2.1 Tree induction

The regular top-down induction of decision trees (TDIDT) by Breiman et al. (1984) is not directly applicable in the streaming scenario. As the data examples are not all available at once, the tree induction procedure is modified to grow the tree incrementally. Specifically, as examples arrive one-by-one they are sorted into a leaf according to the (current) tree. Learning from an a single example is composed of recording of both the values of the input attributes as well as target variables. The procedure is the same as in Ikonomovska et al. (2011a), except that it occurs for each of the targets separately.

3.2.2 Heuristics

Once enough examples, or specifically, records of attribute values, accumulate in a leaf, we check whether we have sufficient statistical support to split the leaf. From the records of the values of attributes, we construct all possible (binary) splits and evaluate them according to a heuristic. In our case, we use the intra-cluster variance reduction heuristic (ICVarR), which is defined as follows

$$\text{ICVarR} = \frac{1}{n} \sum_{i=1}^n \frac{1}{\text{Var}_i(D)} \left(\text{Var}_i(D) - \frac{|D_T|}{|D|} \text{Var}_i(D_T) - \frac{|D_F|}{|D|} \text{Var}_i(D_F) \right),$$

where i indexes the target variables, D is the set of the accumulated instances in the given leaf, D_T and D_F are the subsets of D , which correspond to the sets of instances on which the considered split test is evaluated either as true or false, respectively. Var_i is the variance of the i -th target. Note that D , D_T and D_F are not actually maintained, as we do not store the actual instances. We record the statistics of the instances that are sufficient to calculate the ICVarR heuristic, i.e., the variances of each target on D , D_T , and D_F .

3.2.3 Best split selection

The splits on numeric input attributes take the form $A < c$, for some numeric value c of the attribute A . The splits on nominal attributes take the form $A = n$ for some discrete value n of the attribute A .

For each input attribute, the best split is selected and observed, according to the above heuristic. Let us denote the heuristic value of the overall best split with h_1 , i.e., this is the best split on some attribute, which is also the best among the splits on other attributes. Similarly, we denote the heuristic value of the second best split as h_2 . Let us observe the ratio $\frac{h_2}{h_1}$ as more examples become available – this ratio obviously falls in the $[0, 1]$ interval. We can observe the following sequence

$$\dots, \frac{h_2(k)}{h_1(k)}, \frac{h_2(k+1)}{h_1(k+1)}, \frac{h_2(k+2)}{h_1(k+2)}, \dots,$$

where k denotes the number of instances considered in the calculation of $h_1(k)$ and $h_2(k)$.

Let us consider the ratio $\frac{h_2(k)}{h_1(k)}$ as a random variable X_k . What we record with each incoming instance is a sample x_k from the distribution of X_k . When we have recorded enough samples, we can compute the observed average of $X_{obs} = \frac{1}{|D|} (x_1 + x_2 + \dots + x_{|D|})$. The corresponding probability distribution from which X_{obs} is sampled is then $X_{real} = \frac{1}{|D|} (X_1 + X_2 + \dots + X_{|D|})$. We then apply the Hoeffding bound (Hoeffding 1963), which allows us to make (ε, δ) -estimation in the following form

$$P(|X_{obs} - E(X_{real})| > \varepsilon) \leq 2e^{-2|D|\varepsilon^2} =: \delta. \quad (1)$$

Note that this is already a simplified form of the Hoeffding bound. We can express ε in terms of δ as

$$\varepsilon = \sqrt{\frac{1}{2|D|} \ln \frac{2}{\delta}}.$$

ε is then inferred from δ which is given as a parameter to the learner and $|D|$.

Following from (1), $E[X_{real}] \in [X_{obs} - \varepsilon, X_{obs} + \varepsilon]$ with probability $1 - \delta$. It follows that, if $X_{obs} + \varepsilon < 1$, then $E(X_{real}) < 1$. Finally, this implies that $\frac{h_2}{h_1} < 1$ (with probability $1 - \delta$), i.e., we have support to choose the best split according to which we calculated h_1 .

In the implementation of iSOUP-Trees, we use $\frac{h_2^{|D|}}{h_1^{|D|}}$ as an approximation of X_{obs} .

While all of the theoretical prerequisites for the use of the Hoeffding bound are not (necessarily) met as shown by Rutkowski et al. (2013), the use of the Hoeffding bound still produces good empirical results, as shown by Ikononovska and Gama (2015).

As we have stated before, the splitting criteria are only checked when enough examples have accumulated. Specifically, we check a given leaf each time an $N_{min} = 200$ additional examples have accumulated, i.e., at 200, 400, 600, ... examples. From the above formula, it is clear that ε decreases as the number of accumulated examples increases. As more examples accumulate, we have additional support for the split.

3.2.4 Making predictions

In each leaf, the iSOUP-Tree method uses an adaptive multi-target model, which consists of a multi-target perceptron and a multi-target target mean predictor. As in the single-target case, the multi-target perceptron produces the prediction vector as $\hat{\vec{y}} = W\vec{x} + \vec{b}$, where W is the weight matrix and \vec{b} is the additive vector of constants. On the other hand, the multi-target target mean predictor computes the prediction as the mean value of each of the targets observed at a given leaf. Individually, these learners can be seen as local, however, in conjunction with a tree building method, they constitute a global method.

For each target y_i , the errors $fMAE_{perceptron}^i$ and $fMAE_{targetMean}^i$ are recorded and the decision which of the predictions to use is made for each variable separately. Formally, for each $i \in \{1, \dots, n\}$ the prediction $\hat{y}_{perceptron}^i$ is used when $fMAE_{perceptron}^i < fMAE_{targetMean}^i$,

otherwise we use $\hat{y}_{targetMean}^i$. Consequently, a final prediction $\hat{y} = (\hat{y}^1, \dots, \hat{y}^n)$ can contain predictions made by the perceptron (for some targets), the target mean predictor (for other targets), or both.

3.3 The iSOUP-OptionTree method

While the TDIDT approach is not directly applicable in the streaming setting, the proposed iSOUP-Tree method emulate the process over time as more examples accumulate. Consequently, similarly to TDIDT approach in the batch setting, the proposed method suffer from myopia. One way of potentially addressing this problem is the use of option trees in this setting.

3.3.1 Option trees

An option tree is an extension of a regular decision (or regression tree) in that it introduces an additional type of node, i.e., the option node. This was first introduced (in the batch scenario) by Buntine (1992) and later expanded by Kohavi and Kunz (1997). In the streaming setting, option trees have been used in for the task of single target regression by Ikononovska and Gama (2015).

The myopia of the TDIDT approach results from selecting only the best split at the time, even though, this choice might not be optimal. In the batch case, this may be due to sampling artifacts or noise. In the streaming setting, this can be caused by insufficient statistical evidence for the selection of a given split.

3.3.2 Option nodes

Option trees address this shortsightedness by selecting multiple candidate splits when certain conditions are met. Specifically, an option node is introduced when we do not have enough heuristic-based support to split the leaf. In that case, instead of a split node, an option node is created from a leaf in the tree. Each of its children, called options, is a split node with the split corresponding to one of the selected candidate splits. Each of the split nodes is further split into leaf nodes as is the case in regular trees. When enough support is present, a single split is constructed, much in the same way as in a regular tree.

Following from the more complex learning procedure, using an option tree for learning and prediction is also more complex. In a regular tree, each example reaches exactly one leaf. In an option tree, however, an option node can cause an example to reach multiple leaves. Specifically, when an example passes through an option node, it does not choose only one of the paths as in a split node. Instead, the example is copied once for each option. Each copy is then traversed down one of the option nodes. Therefore, for learning, the example is affecting all of the options (and their associated subtrees) of an option node.

In this way, an example can reach one or more leaves. If more than one leaf is reached, this means that the example passed through at least one option node. Each of the leaves reached produces a prediction which is then aggregated through the option node in a bottom-up manner. Specifically, an option node will receive one prediction from each of its options. The prediction of the option node is then an aggregation of the set of predictions. In regression, averaging of the predictions is commonly used. This prediction is then passed further up, where it may be further aggregated in an option node at a higher level.

The property of one example reaching multiple leaves is present also in tree ensembles. There, however, an example reaches the multiple leaves in separate (base) models, while in

an option tree the leaves an example reaches are all part of the same option tree. For this reason we sometimes refer to option trees as a pseudo-ensemble method, i.e., an option tree can be seen as a compact representation of an ensemble. If the option tree was expanded into its “embedded trees”, we would get a proper ensemble, as seen in Fig. 1.

3.3.3 Extension of the iSOUP-Tree method to utilize option nodes

The **iSOUP-OptionTree** method is the option tree extension of the iSOUP-Tree method for the task of multi-target regression. The extension is done in a similar way to the way Online Regression Trees with Options (ORTO) extends the FIMT-DD approach in the single-target scenario (Ikonomovska and Gama 2015).

As in the case of iSOUP-Trees, the Hoeffding bound is used to grow the tree. In addition to it being used to split leaf nodes into split nodes, it is also used as a criterion when to introduce an option node. If, when evaluating splits, we encounter the case where $X_{obs} + \varepsilon < 1$, a split node is grown as in a regular tree. However, if $X_{obs} + \varepsilon > 1$, we haven't enough evidence to split the node according to the best split, i.e., we don't have enough evidence to differentiate between the best and second best splits. Therefore, we introduce an option node, with options for which the following holds

$$\frac{h_j}{h_1} > 1 - \varepsilon,$$

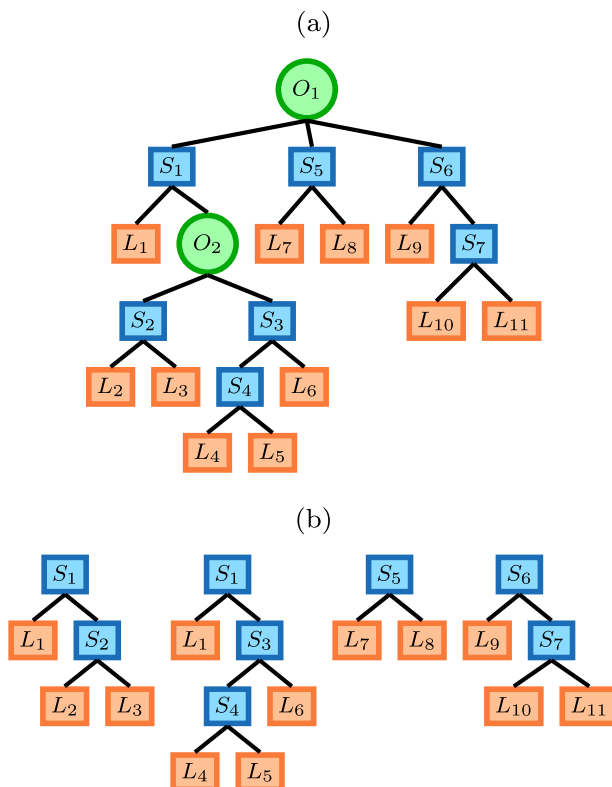


Fig. 1 An option tree (a) and the ensemble of its embedded trees (b). O_i are option nodes, S_j split nodes and L_k leaf nodes

where j enumerates all of the input attributes. The reasoning behind this is the following: all splits for which $X_{obs} + \varepsilon < 1 \sim \frac{h_j}{h_1} + \varepsilon < 1$ does not hold, are approximately equally discriminative, i.e., they are about as discriminative as the best split. The fact that the condition is not met is interpreted as the lack of evidence towards discarding of this split. This concisely determines the candidates for the options in an option node, however, only up to 5 candidates are selected to avoid the combinatorial explosion of the number of option nodes in the tree.

However, we do not (necessarily) select all of the options, i.e., candidate splits. Kohavi and Kunz (1997) have suggested that option nodes are best induced higher in the tree, where they affect more of the data examples. To this end we select only a portion of the candidate splits, in order to decrease the heuristic score, until $K \cdot \beta^{level}$ are selected, where K is the number of candidate splits, $\beta \in (0, 1]$ is the option decay factor and $level$ determines the level of the leaf we are splitting. Option nodes are only induced above level L_{max} .

The parameter β regulates how fast should the induction rate of option nodes be as we descend lower into the tree. The root node is designated level 0, with each split or leaf node contributing 1 to the count. Option nodes do not increase the level count, as they are used only to represent several “parallel” splits. The construction of these parallel splits (contained within the option nodes) is the mechanism through which option trees attempt to address the myopia of the greedy tree building procedure.

As a consequence of the above, each node above level L_{max} is (generally) tested for splitting only once, as one of two things happen. Either there is enough evidence to grow a split node, or there are at least two candidate splits, the best split and the second best split, which satisfy the condition for the split candidates. The only case where this does not happen, is when β^{level} falls below $\frac{1}{K}$. In that case, the leaf node can go through multiple split evaluations. This means that the option tree grows faster than a regular tree.

Note that, this is only one of the possible ways to grow an option tree. Specifically, options are selected through a relatively restrictive statistical test. Instead of using a statistical test, we could also evaluate the merit of candidate splits based on other metrics. For example, we could use the relative value of the heuristic of the candidate split compared to the heuristic of the best candidate split and select all candidates which fall within a certain percentage, e.g., select all candidates whose heuristic values are within 10% of the heuristic value of the best split.

3.4 Online bagging of iSOUP trees (iSOUP–Bagging)

A popular method for learning ensembles in the batch setting is the bootstrap aggregation approach – bagging (Breiman 1996). To introduce variability the constituents of the ensemble, each base models is learned on a bootstrap replicate of the original data. Each bootstrap replicate is in fact a re-sampling with replacement of the original data.

The use of the batch approach for constructing bagging ensembles is not feasible in the streaming setting, as the data is not fully available at any given point in time. To address this problem Oza and Russel (2001) have introduced the online bagging procedure, which maintains several properties from the batch approach. Notably, a given data example in a data set of m examples can appear multiple times in a bootstrap replicate. The probability that it does not appear can easily be calculated as

$$P(\text{given example does not appear}) = 1 - \left(1 - \frac{1}{m}\right)^m.$$

If $m \rightarrow \infty$, this converges to $1 - \frac{1}{e}$. Similarly, when $m \rightarrow \infty$ we can expect $\frac{1}{e}$ repeated instances in each bootstrap replicate. The number of repetitions of a given example in a bootstrap replicate is distributed according to the binomial distribution $B(m, \frac{1}{m})$. When $m \rightarrow \infty$, this distribution tends to the Poisson distribution with $\lambda = 1$, i.e., to Poisson(1).

This motivates the following use of online bagging and using our developed iSOUP-Trees as base models. For each incoming data example and each base model in the ensemble, we sample the number of repetitions k according to the Poisson(1) distribution for that example-model pair. The selected base model then learns from the current data example k times, which introduces variety in the base models.

3.5 Online random forest of iSOUP trees (iSOUP–RandomForest)

Oza and Russel (2001) also introduced an extension of the random forest methodology (Breiman 2001) for data streams. Where online bagging is agnostic to the selection of the base learner, the random forest requires an adaptation of the base tree-based model learner.

In order to develop an online random forest method for our setting, we modified our proposed iSOUP-Tree method, which is used as a base method, as follows. Whenever a leaf node is constructed, i.e., at the beginning of the learning procedure or when a leaf node is split into two leaf nodes, a subset of the input attributes is randomly selected. The statistics are then recorded only for the selected input attributes. Consequently, only the selected attributes can be selected as splits.

The random forest methodology greatly alleviates the stress on the consumption of resources, as only a portion of the statistics are stored, requiring less memory. Additionally, fewer splits are considered when splitting a leaf, making this approach much faster.

There are several suggested values for the portion of input attributes to be considered in a given leaf. The most common are $\sqrt{N_A}$, $\lceil \log N_A \rceil + 1$ and $\alpha \cdot N_A$, where N_A is the total number of attributes and $\alpha \in (0, 1)$. In addition to the randomization of the base model, the online random forest procedure includes the online bagging procedure, i.e., we randomly sample the attribute space as well as the space of data examples.

4 Experimental setup

In this section, we first present the experimental questions that we want to answer in this paper. Next, we describe the datasets and the methods used in the experiments. Finally, we discuss the evaluation measures used in the experiments and present the experimental methodology.

4.1 Experimental questions

Given the introduced methods, we first explore how they compare among themselves in terms of predictive performance. However, another key aspect of the streaming setting are the potential constraints on the available resources. In our case, we observe the performance of different methods through the lens of the use of resources, i.e., the use of time and memory.

We additionally make specific comparisons between some of the introduced methods. In particular, we are interested in how the option tree compare to both single trees and ensembles of trees, as option trees show aspects of both approaches. We are also specifically

interested in how the newly introduced methods, i.e., option trees and online random forests of iSOUP trees, compare to the methods considered in our earlier work.

In a single-target study, Ikonovska and Gama (2015) have shown no particular differences in predictive performance between the basic model tree method and the bagging method (therein referred to as FIMT-DD and OBag, respectively). Additionally, the random forest methodology produced worse results than a single tree, while the option tree variant (ORTO) of FIMT-DD outperformed all of the other methods.

In this work, we wish to investigate whether similar conclusions can be drawn in the multi-target case. To that end, we study the differences in predictive performance between the iSOUP-Tree, iSOUP-OptionTree, iSOUP-Tree bagging and the iSOUP-Tree random forest methods.

As noted before, we are interested in the differences in consumption of resources between the different methods. While the resource consumption of the bagging method in comparison to a single tree is extrapolated trivially, other comparisons are more meaningful. In particular, we are interested in the resource consumption/predictive performance trade-off which occurs in larger, more complex models, i.e., in option trees, bagging and random forests (as compared to the basic single tree approach). The knowledge about this trade-off is especially important when selecting a proper approach for a given real-world application.

4.2 Datasets

For the experiments, we have selected a total of 8 datasets, based on their size, looking for diversity in the number of input and target attributes. We consider the datasets under the assumption of no concept drift, given that these datasets are generally considered as batch datasets. A summary of the datasets and their properties is shown in Table 1.

The *Bicycles* dataset is concerned with the prediction of demand for rental bicycles on an hour-by-hour basis (Fanaee-T and Gama 2013). The 3 targets represent the number of casual (non-registered) users, the number of registered users and the total number of users for a given hour, respectively.

The *EUNITE03*¹ dataset was used for the competition at the 3rd European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems in 2003. The data describes a complex process of continuous manufacturing of glass products, i.e., the input attributes describe various influences (which can or can not be changed by an operator), while the target attributes describe the glass quality.

The data in the *Forestry Kras* dataset was derived from multi-spectral multi-temporal Landsat satellite images and 3D LiDAR recordings of a part of the Kras region in Slovenia (Stojanova et al. 2010). The input attributes and target variables were derived from the Landsat and LiDAR recordings, corresponding to spatial units of 25 meters by 25 meters. For specifics on the data preparation procedure, see Stojanova et al. (2010). The task is to predict 11 target variables which correspond to properties of the vegetation of an observed spatial unit.

The *Forestry Slivnica* dataset was, as in the previous case, constructed from multi-spectral multi-temporal Landsat satellite images and 3D LiDAR recordings of a part of the Slivnica region in Slovenia (Stojanova 2009). In this dataset, the task is to predict only 2 target variables: vegetation height and canopy cover.

¹<http://www.eunite.org/eunite/news/Summary%20Competition.pdf>.

Table 1 Datasets used in the experiments and their properties. N – number of instances, T – number of targets

Dataset	N	Input attr.	T
Bicycles (Fanaee-T and Gama 2013)	17379	12 numeric	3
EUNITE03	8064	29 numeric	5
Forestry Kras (Stojanova et al. 2010)	60607	160 numeric	11
Forestry Slivnica (Stojanova 2009)	6218	149 numeric	2
RF1 (Xioufis et al. 2012)	9005	64 numeric	8
RF2 (Xioufis et al. 2012)	7679	575 numeric	8
SCM1d (Xioufis et al. 2012)	9803	280 numeric	16
SCM20d (Xioufis et al. 2012)	8966	61 numeric	16

The river flow datasets, *RF1* and *RF2*, concern the prediction of river network flows for 48 hours at 8 locations on the Mississippi River network (Xioufis et al. 2012). Each data example comprises observations for each of the 8 locations at a given time point, as well as time-lagged observations from 6, 12, 18, 24, 36, 48 and 60 hours in the past. In *RF1*, each location contributes 8 input attributes, for a total of 64 input attributes and 8 target variables. The *RF2* dataset extends *RF1* with the precipitation forecast information for each of the 8 locations and 19 other meteorological sites. Specifically, the precipitation forecast for 6 hour windows up to 48 hours in the future is added, which nets a total of 280 input attributes.

The *SCM1d* and *SCM20d* are datasets derived from the Trading Agent Competition in Supply Chain Management (TAC SCM) conducted in July 2010. The preparation (preprocessing) of the datasets is described by Xioufis et al. (2012). The data instances correspond to daily updates in a tournament – there are 220 days in each game and 18 games per tournament. The 16 targets are the predictions of the next day and the 20 day mean price for each of the 16 products in the simulation, for the *SCM1d* and *SCM20d* datasets, respectively.

The Bicycles dataset is available at the UCI Machine Learning Repository² and the *RF1*, *RF2*, *SCM1d* and *SCM20d* datasets are available at the Mulan multi-target regression dataset repository.³ The examples with missing values (on some input attributes) in the *RF1* and *RF2* datasets were removed, so the resulting datasets were somewhat smaller than reported in the repository.

4.3 Compared methods

For our experiments, we consider the tree-based methods described in Section 3. Specifically, we consider the local FIMT-DD-based method for MTR (denoted Local), and the global methods, iSOUP-Tree, the iSOUP-OptionTree (iSOUP-OT), iSOUP-Tree online bagging (iSOUP-Bag) and the online random forest of iSOUP-Trees (iSOUP-RF). The parameters of the methods are described in Table 2. All of the considered methods are implemented in the Massive Online Analysis (MOA) framework introduced by Bifet et al. (2010).

The FIMT-DD method is capable of detecting changes in the concept and adapting to them. However, iSOUP-Tree is not. As this study is oriented towards comparing all of the tree-based approaches on equal grounds, the change detection and adaptation mechanisms in FIMT-DD have been disabled for this study.

²<https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset>.

³<http://mulan.sourceforge.net/datasets-mtr.html>.

Table 2 The parameters of the compared methods, their values and descriptions

Shared parameters		
δ	0.0000001	The δ used for the (ε, δ) -estimation of the splits.
N_{min}	200	The number of examples of checking for leaf splitting.
iSOUP-OptionTree		
β	0.9	The interval of checking for leaf splitting.
L_{max}	10	The lowest depth which allows option nodes.
iSOUP-Bagging and iSOUP-RF		
T	100	The number of base models (iSOUP-Trees).
iSOUP-RF		
$f(N_A)$	$\lceil \log N_A \rceil + 1$	The number of attributes to consider in a node.

4.4 Evaluation measures and experimental methodology

To evaluate the predictive performance we define the *average relative mean absolute error* (*RMAE*) measure on an evaluation dataset D as

$$\overline{RMAE} = \frac{1}{n} \sum_{i=1}^n \frac{\sum_{j=1}^{|D|} |y_j^i - \hat{y}_j^i|}{\sum_{j=1}^{|D|} |y_j^i - \bar{y}^i|},$$

where y_j^i is the true values of the target i for example j , \hat{y}_j^i are the predictions of the evaluated model and \bar{y}^i is the average value of the i -th target variable. The use of the average in the denominator normalizes the errors of the predictions for the different target variables, which allows us to average the relative mean absolute errors to obtain \overline{RMAE} . Lower values of the error are desired.

We are using the *prequential* (Gama 2010) approach for evaluating the methods. An incoming instance is first used to make a prediction, which is used in the evaluation. Afterwards, the model is updated using the instance. Since the reported errors on data streams can be volatile if reported on an instance by instance basis, due to, e.g., the sampling of different parts of the input space, we calculate the evaluation measures on the entire dataset. Afterwards, we apply the Friedman test with Nemenyi post-hoc analysis to assess the statistical significance of differences in performance (Demšar 2006).

To evaluate the time consumption, we will consider the running time of the methods. The memory consumption is measured by using the size (in megabytes) of the learned models. We observe the evolution of both the time and memory consumption as the number of processed instances increases, to compare the different methods. Both time and memory consumption are reported at intervals of 1000 examples.

5 Results and discussion

In this section, we present the experimental results and use them to answer our experimental questions. We first look at the predictive performance and the at the use of resources. Finally, we discuss the trade-off between the use of additional resources and obtaining better predictions.

5.1 Predictive performance

The discussion of our results will follow the experimental questions. The results of the experiments in terms of \overline{RMAE} are presented in Table 3. We note that all of the \overline{RMAE} are below 1, which means that each of the observed tree-based methods performs better as the mean predictor, i.e., a regressor that always predicts the mean of the observed target values.

When comparing option trees (iSOUP-OT) with a regular iSOUP-Tree, we notice that using option trees produces better results than regular trees. This can be seen on most of the datasets (excluding of the EUNITE03, RF1 and SCM20d datasets). However, the differences between iSOUP-Tree and iSOUP-OT on these datasets are relatively small. On the other hand, neither regular nor option trees outperform the local method in performance, which is a strong baseline for this problem.

Option trees do not compare favorably to ensemble methods in terms of predictive performance. Specifically, they lose (on almost all datasets) to bagging (iSOUP-Bag) and online random forests (iSOUP-RF), though in the latter case their results are closer. The results of comparing bagging and random forests are as expected, i.e., bagging generally outperforms random forests. This not surprising, however, since we expect the random forest approach to have a better trade-off between the predictive performance and use of resources. Random forests also notably beat the local approach.

Overall, the bagging method outperforms all of the competitors, with random forests coming in second. These results are not unexpected, as these types of ensembles generally bring an increase in predictive performance. Both iSOUP-Tree and iSOUP-OT perform worse than the local FIMT-DD-based approach.

Through the use of the Friedman test and Nemenyi post-hoc analysis, we can also determine that the predictive performance is statistically significantly different between iSOUP-Tree and iSOUP-Bag as seen in Fig. 2. We need further evidence to statistically confirm or deny any of the other observed differences. We also note that the local approach is hard to beat using a single tree approach.

5.2 Use of resources

The results in terms of the use of resources are shown in Figs. 3 and 4. For brevity, only the results on four of the selected datasets (Bicycles, Forestry Slivnica, RF1 and SCM20d) are

Table 3 Predictive performance results on \overline{RMAE}

	Local	iSOUP-Tree	iSOUP-OT	iSOUP-Bag	iSOUP-RF
Bicycles	0.4717 (3)	0.5257 (4)	0.4039 (1)	0.4144 (2)	0.6408 (5)
EUNITE03	0.8199 (4)	0.7014 (2)	0.7504 (3)	0.5976 (1)	0.8916 (5)
RF1	0.1946 (5)	0.1861 (3)	0.1923 (4)	0.1832 (2)	0.1761 (1)
RF2	0.3834 (2)	0.5814 (5)	0.5454 (4)	0.5246 (3)	0.3711 (1)
Forestry Kras	0.6190 (4)	0.6461 (5)	0.5988 (3)	0.4766 (1)	0.4838 (2)
Forestry Slivnica	0.6397 (2)	0.7417 (5)	0.7028 (4)	0.6043 (1)	0.6569 (3)
SCM1d	0.3866 (1)	0.5360 (5)	0.5026 (4)	0.4084 (2)	0.4765 (3)
SCM20d	0.5283 (5)	0.3890 (3)	0.3903 (4)	0.2931 (2)	0.2825 (1)
Avg. rank	3.25	4.0	3.375	1.75	2.625

The table contains the values of \overline{RMAE} (and the rank) of each method on each of the datasets

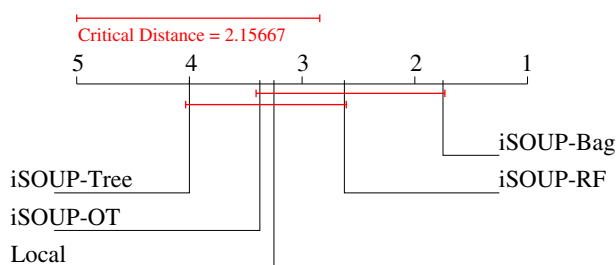


Fig. 2 Average rank diagrams for the \overline{RMSE} measure

presented. The results on the remaining datasets are very similar. Additionally, each dataset is represented on two graphs, as the scales for the single tree and the local approaches are drastically different from those of the ensemble methods. We first discuss the use of memory and then the use of time.

When comparing the memory use of the local approach and the single iSOUP-Tree, we can see that (even when the number of targets is low) iSOUP-Tree uses less memory than the local FIMT-DD approach. This is also the case on the Forestry Slivnica dataset, which only has 2 targets, and as such represents the simplest multi-target regression problem.

Option trees use more memory than both the single tree and the local approach. However, the growth rate of memory usage shows that they do not reach a size (e.g., in terms of the number of leaves) comparable to the size of the models produced by bagging. As compared to the 100 trees in the bagging ensemble, the option tree is composed only of a smaller number of embedded trees.

As expected, random forests use much less memory than bagging. Their memory use is more similar to that of option trees, however, they mostly use more memory than option trees.

When studying results of the use of memory, we notice several steep decreases in the memory used. This occurs when multiple leaves are split in the observed time period. Whenever a leaf is split, the statistics it was storing for use with the Hoeffding bound are forgotten, which frees up a lot of memory.

With regards to time, we again note that iSOUP-Tree generally outperforms (i.e., is quicker) the local approach. Notably, option trees are slower than the local approach. The results of the ensemble methods in terms of time use entirely follow from our observations on memory use.

5.3 Discussion

In the streaming setting, our desire for better predictive performance is often held back by strict constraints on the available resources. The presented methods have different factors of trade-off between the predictive performance and the use of resources.

Specifically, iSOUP-Tree is the quickest, least memory intensive method. However, it also produces comparatively the worst results. iSOUP-OT uses more memory and time and on average achieves better results than iSOUP-Tree, but the additional use of resources could be better “spent” on using the local approach which produces better results with less memory and time. However, if the number of targets increased drastically, e.g., 10- or 100-fold, this result would most likely tip in favor of the option trees.

As we have noted earlier, option trees never really grow to the size of the ensembles, which also explains their lower predictive performance. iSOUP-Bag, the winner in terms of predictive performance, uses a 100-times the resources used by a single iSOUP-Tree.

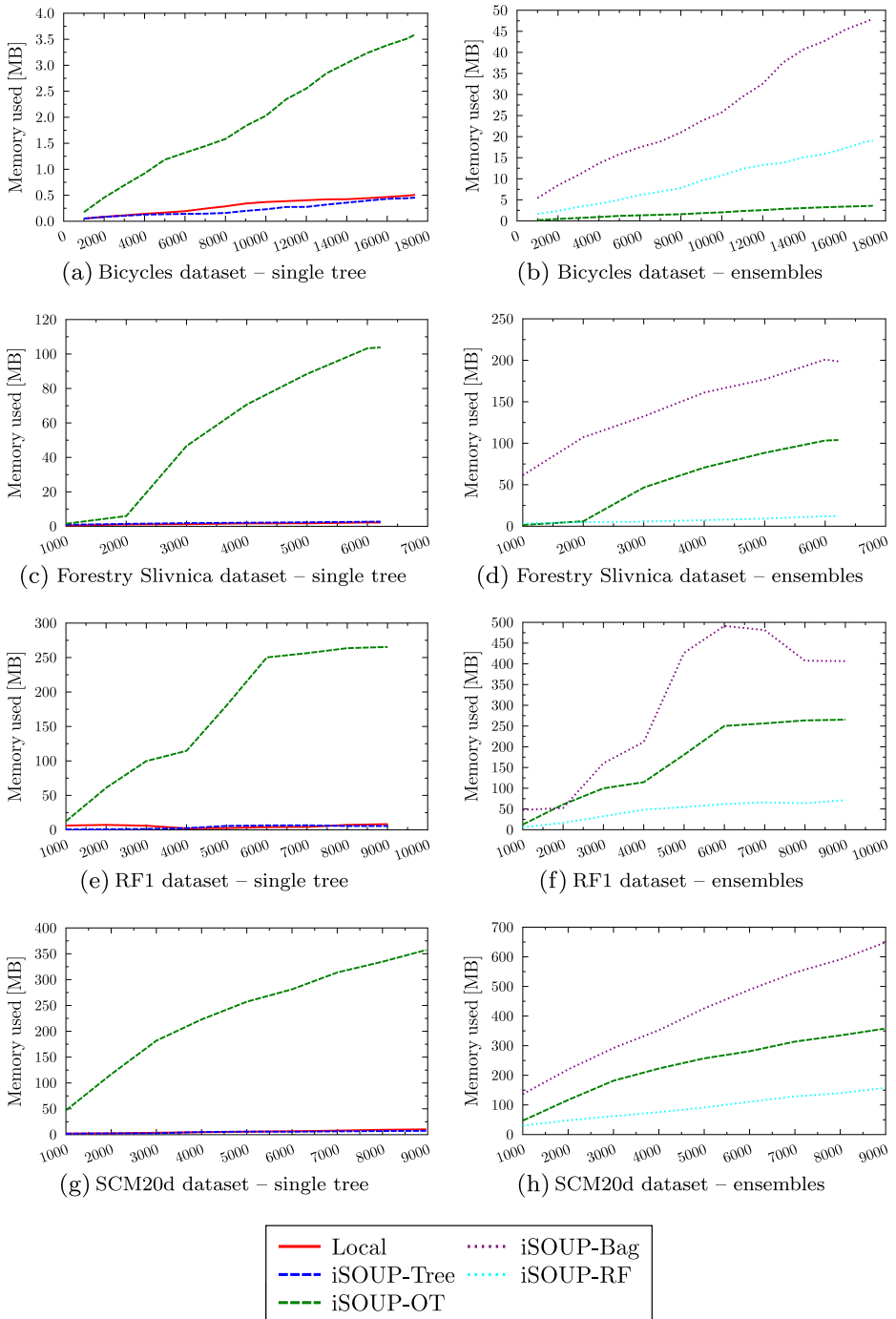


Fig. 3 The memory consumption of the observed methods. Horizontal axes show the numbers of processed examples

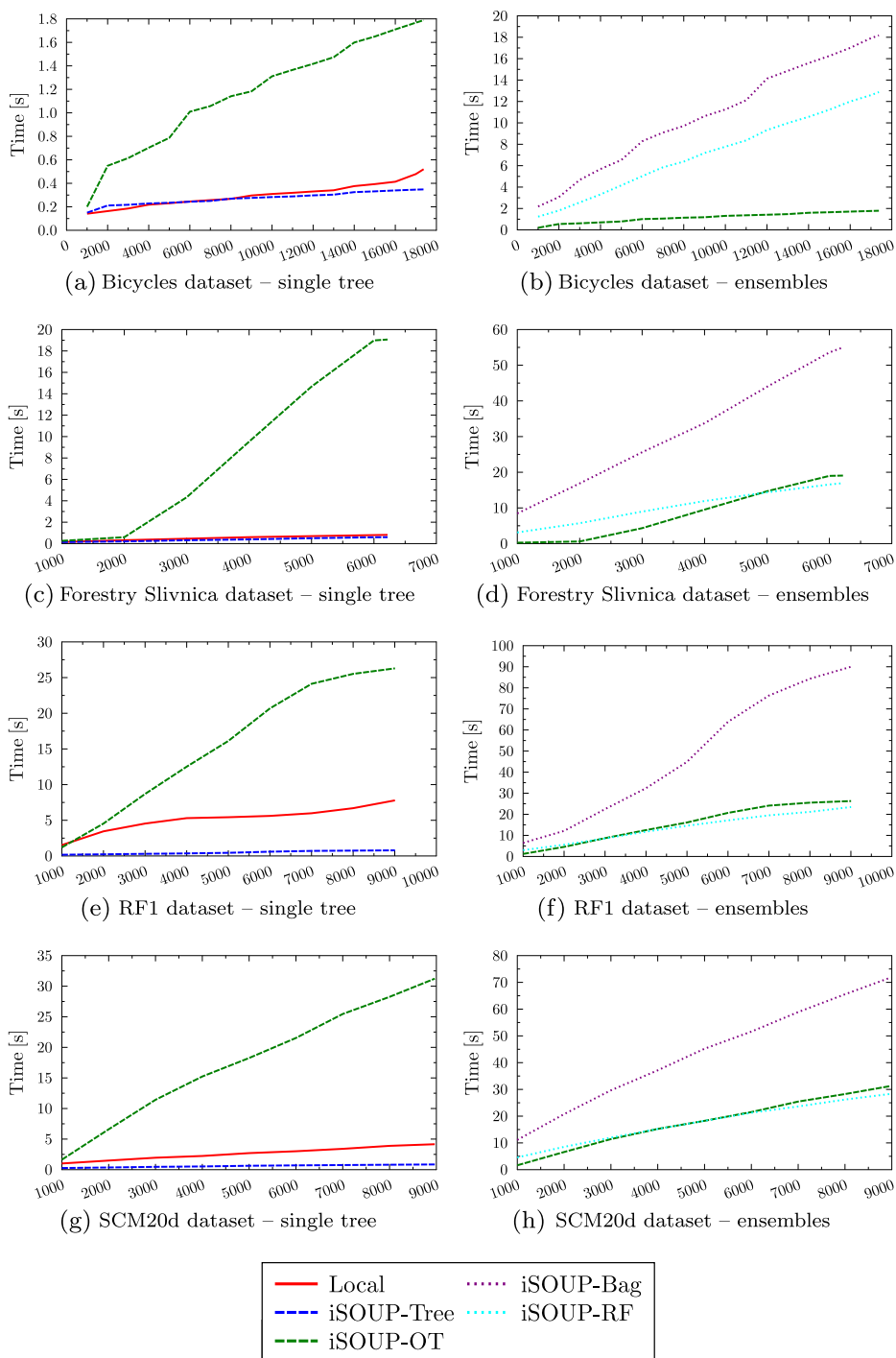


Fig. 4 The time consumption of the observed methods. Horizontal axes show the numbers of processed examples

If resources are unbounded, bagging should be the method of choice. However, if the resources are bounded, random forests provide a good trade-off between the predictive performance and the use of resources. Specifically, they produce reasonable results, while using considerably less resources than the bagging method.

Many of our findings run in opposition to the results and findings of Ikonomovska and Gama (2015) in the single-target scenario. There, both bagging and random forests performed equally to or worse than a single FIMT-DD tree, while option trees (ORTO) clearly outperformed the regular FIMT-DD tree. This discrepancy in the results motivates us to further explore how the characteristics of single-target regression relate to those of multi-target regression.

6 Case studies

In addition to the evaluating the proposed method on benchmark datasets, as described in Sections 4 and 5, we consider two case studies. In the first, we predict the power consumption of the European Space Agency's Mars Express probe, currently in orbit around Mars. In the second, we learn to predict photo-voltaic (PV) power generation in US from the data provided by the National Renewable Energy Laboratory (NREL). For these two case studies, we provide some insights into the practicality of applying predictive modelling in a data streaming setting, in addition to performing the performance comparisons described earlier (Sections 4 and 5).

6.1 Predicting the power consumption of the mars express probe

Dataset The Mars Express dataset is derived from the data provided by the European Space Agency for the Mars Express Power Challenge.⁴ The data describes the operation of the Mars Express Orbiter, a probe that has been in Mars' orbit, for around 12 years (or 4 Martian years). The probe must maintain a proper thermal balance to allow for the operation of its scientific instruments. However, different instruments require different thermal ranges to operate properly. In essence, the amount of power available for scientific experiments is equal to the total amount of energy produced minus the energy requirements of the thermal system and the energy requirements of the satellite platform, i.e., the energy required for maintaining the operation of the satellite itself.

The task of the challenge was to predict the electric current at 33 different thermal heaters in the satellite for each operating hour, corresponding to the power consumption of the thermal regulation system. The data for 3 Martian years was provided as a training set, while the data from the fourth Martian year served as the testing/evaluation set. The raw data⁵ was composed from information about the spatial orientation and alignment of the probe with regards to the Sun, Mars and the Earth including eclipses (umbras), as well as from the probe's flight dynamics and information about the (de)activations of its internal systems. This data also suffers from concept drift, as the probe's different systems degrade in time resulting in different thermal properties.

⁴<https://kelvins.esa.int/mars-express-power-challenge/>.

⁵<https://kelvins.esa.int/mars-express-power-challenge/data/>.

The features used in this dataset are the apparent influx of solar energy for each of the six sides of the approximate cuboid probe, as well as the influx of energy to its solar panels. The features take into account the solar angle of incidence to a particular surface of the probe, the position of the probe in relation to the Sun and the solar constant at a given time point that takes into account the distance to the Sun. Specifically, the features are calculated as

$$\text{feat}(t_i) = \int_{t_i}^{t_{i+1}} A_E(t)c(t)U(t)dt,$$

where $\text{feat} \in \{\text{front, back, left, right, up, down, panels}\}$, t_i and t_{i+1} are the time of the i -th and $(i + 1)$ -th measurement, respectively, $A_E(t)$ is the apparent area of the given surface, $c(t)$ is the solar constant and $U(t)$ is the umbra coefficient. $A_E(t)$ is calculated as

$$A_E = A \max\{\cos \alpha, 0\},$$

where α is the angle of incidence for the given surface and A is the area of the surface. However, since the values of the features are always compared only relative to the values of the same feature and never to the values of the other features, we can, without loss of generality for the learning process, assume that $A = 1$. The umbra coefficient conveys whether the probe is partially or completely in the Mars' shadow (or in the shadow of one of Mars' two moons, Phobos and Deimos), i.e., $U(t) \equiv 0$ when the probe is fully in Mars' umbra (shadow) and $U(t) \equiv 0.5$ when the probe is in Mars' penumbra (partial shadow). Otherwise, $U(t) \equiv 1$.

Additionally, the dataset also contains the sums of the above features, calculated as

$$\text{feat-sum}N(t_i) = \sum_{t=t_{i-1}}^{t_i-N} \text{feat}(t),$$

for $N \in \{4, 16, 32, 64, 128\}$ describing the probe's history. This yields a total of $7 + 7 * 5 = 42$ continuous descriptive features. We calculate the values of the features for each minute of operation, producing a total of around 2.6 million examples. Note that the competition called for hourly predictions. Our dataset is constructed at the one minute granularity to match the actual recordings of the target variables, which were recorded at a rate of about one measurement per minute. In the competition setting, the predictions for 60 consecutive minutes would then be aggregated to obtain the hourly predictions. For the purposes of this paper, we only use the first 100k (of 2.6 milion) examples, as the experimental setup, specifically the measurement of the memory usage (which takes up to 80% of the total experimental time) makes the learning on the entire dataset unfeasible.

Task In this case study, we are concerned specifically with how well the different methods cope with the requirement of producing real-time minute-by-minute predictions. In this case, re-learning a model, e.g., a decision tree, from all data for each new example quickly becomes unfeasible, as we soon reach the point where the learning process itself takes more than the one minute time window in which a prediction must be made.

Results The results for the Mars Express dataset are presented in Fig. 5. In addition to the metrics used in the earlier sections, we also show the average time spent processing an example. Specifically, this time includes the time to make the prediction for the example, as

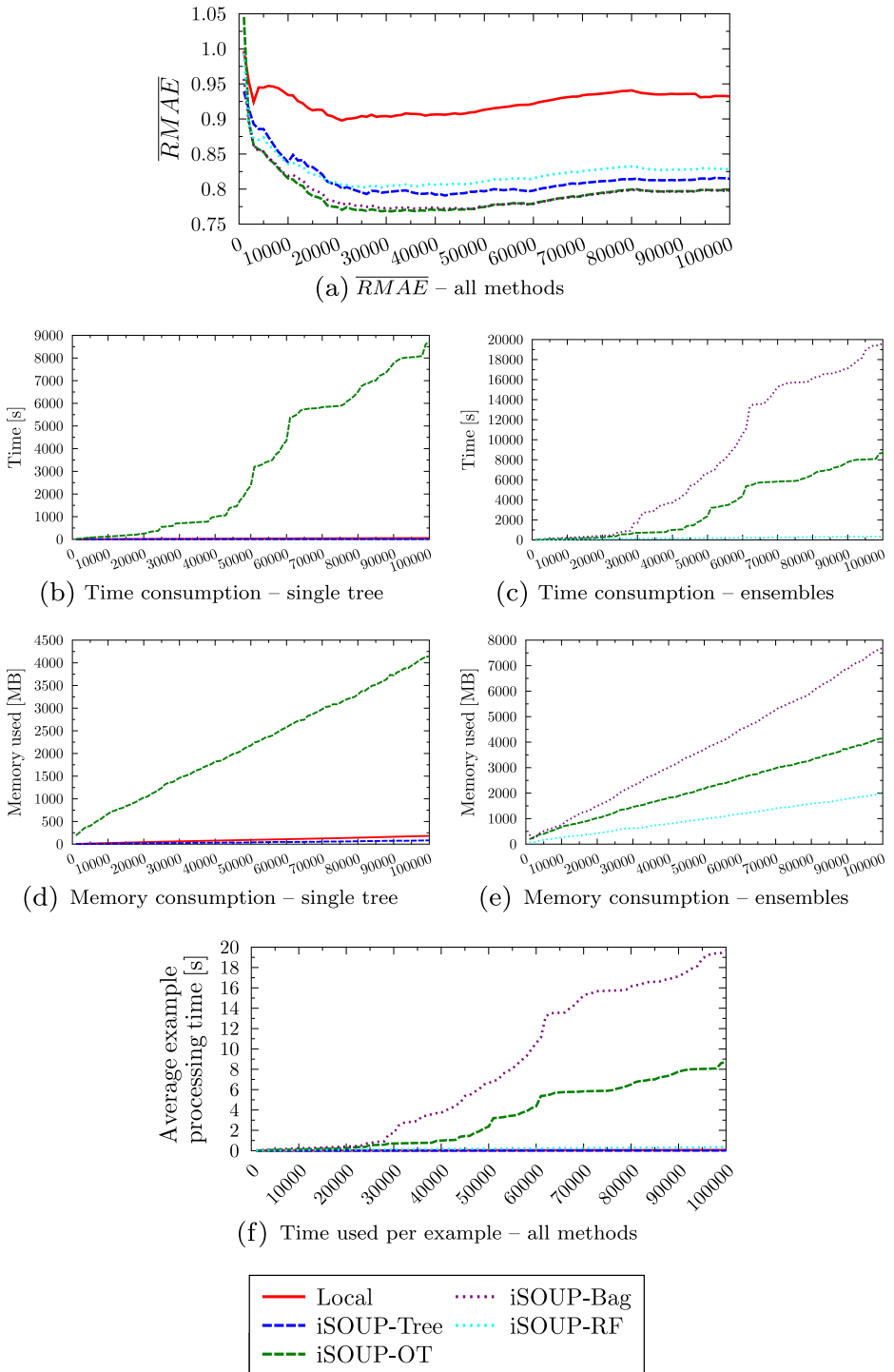


Fig. 5 The results on the Mars Express dataset. The horizontal axes show the numbers of processed examples

well as the time it takes to learn from the example, i.e., the time to update the model. The processing time naturally increases due to the tree growing larger and larger as the model is updated.

The local, FIMT-DD based, approach has the worst performance of all of the observed methods. Given the proximity of the different elements of the heating apparatus, it is reasonable to expect that the targets (electric currents/power consumption at these elements) are quite correlated, which explains the better performance of the multi-target models. Of the observed multi-target iSOUP-Tree based models, iSOUP option trees and bagging of iSOUP trees achieve comparable results, with iSOUP option trees notably outperforming bagging in terms of efficiency.

All of the models use on average less than a minute to process one example, even at the end of the dataset. This would theoretically allow for the models to be used in real-time to predict the electric currents within the Mars Express probe. We expect the processing time to be in a linear relation to the depth of the tree, i.e., $O(\log n)$ where n is the number of processed examples. Note, however, that the results presented are for the first 100k (of the 2.6 million) examples. If the processing time were to exceed one minute, a less complex model would have to be used to reduce the processing time. There are two obvious options; i.e., to learn a less complex model like a single tree which has very low example processing time, or to reduce the complexity of the ensemble/option tree, by discarding some members of the ensemble/options, respectively.

6.2 Predicting photo-voltaic (PV) power generation

Dataset The raw data provided by NREL contains power production forecasts of about 6,000 simulated PV plants. We use the version of the dataset used by Ceci et al. (2016) as a starting point. This dataset has been narrowed down to 48 representative plants and the associated measurements and forecasts. For this version of the dataset the following 24 features are provided for each of the 48 plants: (1) historical data on power production, (2) current weather information, (3) weather forecasts from numerical weather predictions (NWP) models and (4) geographic coordinates of the plant. The features are recorded at hourly intervals, for each hour between 2:00 and 20:00.

Unlike Ceci et al. (2016), who produce predictions for each plant separately in an effort to explore the effects of spatial and temporal correlation, we join all of the data for a single time point, i.e., for a given hour. This reduces the number of examples from around 280k (one per time point per site) to about 7k (one per time point for all sites). Examples joined in this way have a large number of features, as well as targets. Namely, each example contains a total of 1152 features and 48 target values.

In terms of features, this dataset has more than twice the number of features of any other dataset in our experiments. In terms of targets, its number of targets is three times greater than the highest number of targets in the other experiments. This means that each example contains a large quantity of information.

Task We use the NREL dataset as a “stress test” in terms of the size of each individual example. We are interested in how this impacts the time and memory consumption, as well as the average example processing time of the different methods. With the predictions being made at hourly intervals, there is less pressure to achieve a low processing time. Given the relatively large distances between the different PV sites, we also expect the targets to be less correlated than in the Mars Express case study, which should favor the local approach.

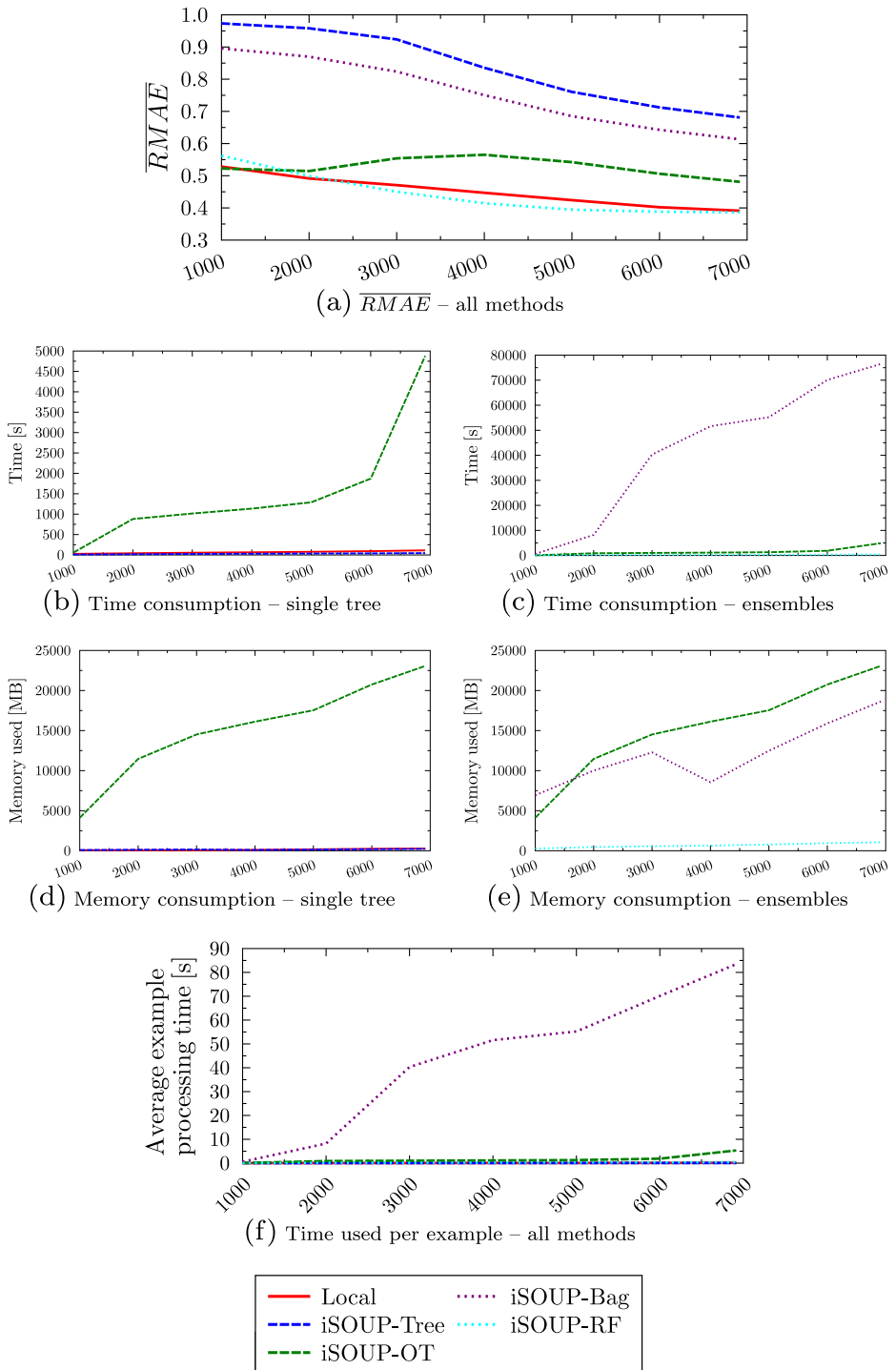


Fig. 6 The results on the NREL dataset. The horizontal axes show the numbers of processed examples

Results The results for the NREL case study are presented in Fig. 6. As expected, the local approach performs very well (i.e., the best) in this setting. Random forests of iSOUP trees achieve a comparable performance, which is likely the result of the large number of features. Option trees are better than bagging, while the single iSOUP tree is worse than both of these.

The local approach, as well as the iSOUP tree and random forest of iSOUP trees process the dataset rather quickly, unlike option trees which require around 90 minutes, while bagging takes more than 22 hours to process the whole dataset. Similarly, option trees and bagging use considerably larger amounts of memory, reaching more than 20 GB of memory use at the end of the dataset.

Since the predictions are made at hourly intervals and neither model uses nowhere near an hour to process an example, the models could also be used alongside a human domain expert, which could analyze the models' predictions and potentially improve the predictive performance. In a purely automated setting, it would be advisable to use either the local approach or the random forest of iSOUP trees, given their better performance, their higher speed and their lower memory usage.

7 Conclusions and further work

In this paper, we have presented several tree-based methods for multi-target regression. Specifically, we introduced the local FIMT-DD-based method, which uses single-target trees for each target separately, the iSOUP-Tree method, which uses a single tree to predict all of the targets, its option tree extension iSOUP-OptionTree, as well as the ensemble methods of online bagging and online random forests, which both use iSOUP-Trees as base models. Our experimental comparison focused on the predictive performance of the methods and the trade-off between the predictive performance and the use of resources.

In terms of predictive performance, we have found that bagging of iSOUP-Trees performs the best, followed by random forests of iSOUP-Trees. The local FIMT-DD-based method outperformed both iSOUP-Tree and iSOUP-OptionTree, while iSOUP-OptionTree performed slightly better than iSOUP-Tree. According to the Friedman test with Nemenyi post-hoc analysis, only the difference between the performance of bagging iSOUP-Trees and a single iSOUP-Tree is statistically significant.

In terms of the use of resources, the results were not surprising, with bagging consuming more time and memory than random forests and single trees. Notably, the single iSOUP-Tree tends to use less resources than the local FIMT-DD-based approach, which produces multiple single-target trees.

Given the results on predictive performance and resource use, we also considered the trade-off between using more time and memory to obtain better performance. Specifically, we have observed the following pairs where trade-offs occur. iSOUP-Tree generally uses less resources than the local FIMT-DD-based method, while performing worse. By design, random forests sacrifice some of the predictive performance of the bagging method to achieve a much faster learning time and lower memory consumption.

Overall, bagging of iSOUP-Trees is the best method in terms of pure predictive performance, while the random forests of iSOUP-Tree provides the best trade-off between predictive performance and use of resources. In a real world scenario, however, any of the selected methods could be appropriate given the constraints of the learning task, e.g., in terms of running time or memory. Learning on, for example, a hand-held device would impose memory constraints on the learning process, while on a data analytics platform we might be constrained by the need for a fast response.

We additionally performed two case studies to evaluate the compared methods in terms of their applicability to real life problems in addition to observing how they deal with large numbers of examples and/or large numbers of input/target features. The studies show that the methods can cope with a large number of examples in a reasonable time frame, with a medium number of features. On the other hand, a large number of input features combined with a large number of targets slows down the methods considerably, requiring longer intervals between examples.

In their study Ikonomovska and Gama (2015) found option trees to (overwhelmingly) have the best predictive performance, while the bagging and random forest performed comparatively badly. As our experimental results are notably different from those of their single-target study, we intend to explore whether this is an intrinsic property of the multi-target learning setting or it is caused by other factors. Note, for example, that Ikonomovska and Gama (2015) use a VFDT implementation of FIMT-DD and ORTO, while we use a MOA implementation of the multi-target extension of these two approaches.

Furthermore, we plan to extend the comparison of multi-target regression methods on data streams by including also non-tree-based methods into the comparison. Additionally, we plan to consider a larger number of datasets, which will allow for additional testing of statistical significance.

We plan to extend and utilize multi-target regression methods for other tasks by employing the proper task transformation methodologies. For example, we have used MTR for the task of multi-label classification on data streams (Osojnik et al. 2016b). Other data stream mining tasks that we plan to consider include (but are not limited to) hierarchical multi-label classification, hierarchical multi-target regression, semi-supervised multi-target regression, feature ranking for multi-target regression, etc.

Acknowledgments The authors are supported by The Slovenian Research Agency (Grant P2-0103 and a young researcher grant) and the European Commission (Grants ICT-2013-612944 MAESTRA and 720270 HBP SGA1).

References

- Appice, A., & Džeroski, S. (2007). Stepwise induction of multi-target model trees. In *18th European conference on machine learning* (pp. 502–509).
- Bifet, A., & Gavalda, R. (2009). Adaptive learning from evolving data streams. In *8th international symposium on advances in intelligent data analysis* (pp. 249–260).
- Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive online analysis. *Journal of Machine Learning Research*, 11, 1601–1604.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). *Classification and regression trees*. New York: Chapman & Hall.
- Buntine, W. (1992). Learning classification trees. *Statistics and Computing*, 2(2), 63–73.
- Ceci, M., Corizzo, R., Fumarola, F., Malerba, D., & Rashkovska, A. (2016). Predictive modeling of PV energy production: How to set up the learning task for a better prediction? *IEEE Transactions on Industrial Informatics*, PP(99), 1–1. doi:[10.1109/TII.2016.2604758](https://doi.org/10.1109/TII.2016.2604758).
- Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning research*, 7, 1–30.
- Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In *6th ACM SIGKDD* (pp. 71–80).
- Duarte, J., & Gama, J. (2014). Ensembles of adaptive model rules from high-speed data streams. In *3rd international workshop on big data, streams and heterogeneous source mining* (pp. 198–213).
- Duarte, J., Gama, J., & Bifet, A. (2016). Adaptive model rules from high-speed data streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(3), 30.

- Fanaee-T, H., & Gama, J. (2013). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2), 113–127.
- Gama, J. (2010). *Knowledge discovery from data streams*. CRC Press.
- Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301), 13–30.
- Ikonomovska, E., Gama, J., & Džeroski, S. (2011a). Incremental multi-target model trees for data streams. In *2011 ACM symposium on applied computing* (pp. 988–993).
- Ikonomovska, E., Gama, J., & Džeroski, S. (2011b). Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, 23(1), 128–168.
- Ikonomovska, E., & Gama, J. (2015). Online tree-based ensembles and option trees for regression on evolving data streams. *Neurocomputing*, 150, 458–470.
- Kocev, D., Vens, C., & Struyf, J. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3), 817–833.
- Kohavi, R., & Kunz, C. (1997). Option decision trees with majority votes. In *14th international conference on machine learning, ICML '97* (pp. 161–169).
- Osojnik, A., Panov, P., & Džeroski, S. (2016a). Comparison of tree-based methods for multi-target regression on data streams, pp 17–31.
- Osojnik, A., Panov, P., & Džeroski, S. (2016b). Multi-label classification via multi-target regression on data streams. *Machine Learning*. doi:[10.1007/s10994-016-5613-5](https://doi.org/10.1007/s10994-016-5613-5).
- Oza, N.C., & Russel, S.J. (2001). Experimental comparisons of online and batch versions of bagging and boosting. In *7th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 359–364).
- Rutkowski, L., Pietruczuk, L., Duda, P., & Jaworski, M. (2013). Decision trees for mining data streams based on the McDiarmid's bound. *IEEE Transactions in Knowledge and Data Engineering*, 25(6), 1272–1279.
- Shaker, A., & Hüllermeier, E. (2012). IBLStreams: a system for instance-based classification and regression on data streams. *Evolving Systems*, 3(4), 235–249.
- Silla, C.N., & Freitas, A.A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2), 31–72.
- Stojanova, D. (2009). *Estimating forest properties from remotely sensed data by using machine learning*. Master's thesis, Ljubljana: Jožef Stefan International Postgraduate School.
- Stojanova, D., Panov, P., Gjorgjioski, V., & Kobler, A. (2010). Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics*, 5(4), 256–266.
- Struyf, J., & Džeroski, S. (2005). Constraint based induction of multi-objective regression trees. In *4th international workshop on knowledge discovery in inductive databases* (pp. 222–233).
- Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: an overview. *International Journal of Data Warehousing and Mining*, 2007, 1–13.
- Xioufis, E.S., Groves, W., Tsoumakas, G., & Vlahavas, I.P. (2012). Multi-label classification methods for multi-target regression. arXiv:[1211.6581](https://arxiv.org/abs/1211.6581).