DSTARS: A multi-target deep structure for tracking asynchronous regressor stacking

Saulo Martiello Mastelini [a,*], Everton Jose Santana [b], Ricardo Cerri [c], Sylvio Barbon Jr. [b]

[a] Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos 13566-590, Brazil
[b] Department of Computer Science, State University of Londrina (UEL), Londrina 86057-970, Brazil
[c] Department of Computer Science, Federal University of São Carlos (UFSCar), São Carlos 13565-905, Brazil

ARTICLE INFO

Article history:
Received 27 February 2018
Received in revised form 21 October 2018
Accepted 4 March 2020
Available online 13 March 2020

Keywords:
Machine learning
Multi-target
Multi-output
Regression analysis

ABSTRACT

Several applications of supervised learning involve the prediction of multiple continuous target variables from a dataset. When the target variables exhibit statistical dependencies among them, a multi-target regression (MTR) modelling permits to improve the predictive performance in comparison to induce a separate model for each target. Apart from describing the dependencies among the targets, the MTR methods could offer better performance and less overfitting than traditional single-target (ST) methods. A group of MTR methods have addressed this demand, but there are still many possibilities for further improvements. This paper presents a novel MTR method called Deep Structure for Tracking Asynchronous Regressor Stacking (DSTARS), which overcomes some existing gaps in the current solutions. DSTARS extends the Stacked Single-Target (SST) approach by combining multiple stacked regressors into a deep structure. In this sense, it is able to boost the predictive performance by successively improving the predictions for the targets. Besides, DSTARS exploits the dependency of each target individually by tracking an asynchronous number of stacked regressors. Additionally, our proposal explores the inter-targets dependencies by exposing and measuring them through a nonlinear metric of variable importance. We compared DSTARS to SST, Ensemble of Regressor Chains (ERC) and Multi-objective Random Forest (MORF). Also, the ST strategy with different algorithms was used to compute independent regressions for each target. We used Random Forest (RF) and Support Vector Machine (SVM) as base-learners to investigate the prediction capability of algorithms belonging to different machine learning paradigms. The experiments carried out on eighteen diverse datasets showed that the proposed method was significantly better than the other compared approaches.

1. Introduction

The achievement of a model which provides the best understanding of a given dataset towards predicting precise outcomes from new data is highly desirable. Looking forward to fulfilling this requirement, many single-target (ST) methods were proposed [1–4]. These methods aim at predicting a single target, response, or output y based on a set $\mathbf{X}$ of m input variables.

Recently, the interest in predicting simultaneously multiple outputs related to the same explaining set of real-life problems is increasing [5–11]. In these cases, the problems are called multi-target (MT). MT brings an additional challenge since, besides modelling the input to output dependencies, some relationships among the responses may exist. MT methods offer better performance by exploring the inter-target influences and reducing the overfitting in comparison to collections of ST models [5,7,12].

For definition, consider an input (description space) consisting of tuples with primitive data types (only discrete or continuous values) in the form $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$. Also, consider a target or output space $\mathbf{Y}$, with tuples in the form $\mathbf{y} = \{y_1, y_2, \ldots, y_d\}$, being d the number of the problem's outputs. In addition, take a set of examples E comprising elements from $\mathbf{X}$ and $\mathbf{Y}$, i.e., $E = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}$, being n the number of examples. Lastly, take into consideration a quality criterion q that rewards prediction models which present high predictive performance and low complexity [5]. A MT problem can be defined as the task of finding a function $h : \mathbf{x} \mapsto \mathbf{y}$ such that h maximizes q. This prediction problem is solved upon the set of known examples, E. The sought function h can either refer to a single prediction model or a set thereof. Specifically, when the elements of $\mathbf{Y}$ are continuous, with the possibility of being statistically correlated, the related predictive problem is characterized

* Corresponding author.
E-mail addresses: mastelini@usp.br (S.M. Mastelini), evertonsantana@uel.br (E.J. Santana), cerri@dc.ufscar.br (R. Cerri), barbon@uel.br (S. Barbon Jr.).

as multi-target regression (MTR) [6–8,13,14], the focus of this work.

Many real-life problems were reported as MTR tasks, including the prediction of vegetation, soil, water and river flow properties, the estimation of monthly online product sales, forecasting of the future price of multiple products in supply management chains and air-tickets in different companies, the prediction of solar flares parameters, air pollutants concentration, wheat flour quality, among others [7–9,12,15,16].

The state-of-the-art MTR methods present some limitations regarding the explicit measurement and modelling of the target dependencies, providing black-box solutions. None of the existing methods presupposes different levels of correlation among the outputs. Indeed, most of the existing MTR solutions rely on statically inserting target predictions as new input features or randomly exploring the possible existing chains of influence [8, 13,17]. Lastly, to the best of our knowledge, the existing solutions do not take into consideration the premise that some of the targets may have no relationships with the others. In this case, an effective MTR method should treat these uncorrelated outputs as apart ST tasks.

In this work, our new proposal called DSTARS (*Deep Structure for Tracking Asynchronous Regressor Stacking*) is compared with the Single-target (ST) strategy, Stacked Single-target (SST) [8], Ensemble of Regressor Chains (ERC) [8], and the Multi-objective Random Forest (MORF) [5,18]. DSTARS was motivated by preceding approaches which stack targets' predictions as additional input features, but we focused on overcoming the presented gaps.

Our method employs multiple stacked regressors for each target, and asynchronously tracks the ones who contributed to reduce the prediction error. Each new added regressor uses extra input features which are derived from the previously added predictors. DSTARS successively adds stacked regressors under the hypothesis that the prediction error for the most dependent targets will be gradually reduced as new and improved response estimations are stacked as descriptive features. As a matter of visualization, the multiple stacked predictors for each target can be referred as layers of regression models (or regressor layers). Thus, DSTARS can potentially create deep layers of regressors for some targets, depending on the task. By employing a different number of regressor layers for each response, DSTARS considers the specific characteristics of each target individually. In other words, a target can have different dependency levels (or none) in relation to the other responses, and thus, it demands a particular treatment. Lastly, DSTARS explicitly measures the dependency among the targets by employing a variable importance measure. In this work, we chose to employ a nonlinear importance measure which is based on the Random Forest algorithm. In this sense, the possibly nonlinear relationships among the targets can be measured, and only the correlated targets will be subjected to the stacking process; the non-correlated ones are modelled as separated ST tasks.

This paper was built upon our previous work which was presented in Mastelini et al. [19]. However, this research upgrades and extends the previous one in the following main aspects:

- An expanded set of 18 benchmark datasets is employed in the experiments;
- DSTARS now employs a nonlinear metric of variable importance to select which targets are subjected to the process of stacking regressors;
- The target importance measurements are calculated with ST predictions of the targets, which are taken in validation sets, to offer more realistic measurements on the inter-response relationships;
- The presentation of DSTARS, the experimental setup, and the discussion of the results were expanded;
- The Multi-objective Random Forest (MORF) is also compared;
- Detailed results regarding the obtained errors per target, and the number of regressor layers that DSTARS generated for each output in the evaluated datasets are also presented.
- The MTR methods are also compared regarding their running time;
- Supplementary materials are offered in the Appendices to support and detail the experiments performed.

This paper is organized as follows: Section 2 explores existing MTR methods, presenting in details the ones compared with DSTARS. Section 3 presents the complete description of DSTARS. The experimental setup is described in Section 4, followed by the results and the analysis of them in Section 5. The conclusions and future research directions are presented in Section 6. Lastly, Appendices with additional detailed information about the performed experiments are presented.

2. Related work

A first straightforward strategy to predict multiple outputs is predicting each target variable separately, as a set of ST problems. Although simple, in many cases, committed by weakly correlated targets, this strategy outperformed some MTR methods [7,8,14]. Clearly, when each target is modelled separately, the targets correlations are missed. Still, the ST strategy has been used as a performance baseline in MTR works [6,8,13,14].

Differently, when a given machine learning algorithm is applied to deal with a MT problem at once, some adjusts in the original modelling method must be performed in order to deal with multiple targets. These modifications include changes in the optimization function (SVMs) [20–22], or the node splitting criteria (regression trees) [5,18].

As described by Borchani et al. [7], MTR tasks have been conducted by two approaches: algorithm adaptation and problem transformation. Kocev et al. [5] refer to the mentioned approaches as global and local, respectively. The global or algorithm adaptation approach refers to adapting a ST regression algorithm to deal with multiple outputs or even creating a new one to perform this task. In this case, the resulting method must deal with all the underlying inter-target dependencies within the created model, which can be very challenging. Diversely, the local or problem transformation approach manipulates the training data to explore the statistical dependencies among the outputs and employ multiple ST regressors to predict the targets. Although multiple predictors are employed, increasing the computational costs, this strategy also increases the modularity and flexibility of the solution.

Some global methods were proposed in past years [7], being applied in several tasks [7,12,23–25]. In fact, the global-based methods have achieved good prediction performances, bringing the advantage of generating a unique model and, thus, consuming less resources. Among the global-based methods, the Multi-objective Random Forest (MORF) has received increasing attention. This method, proposed by Kocev et al. [5,18], was explored in many works jointly with the ST for performance comparison [6,8,13,14]. Taking into consideration its relevance, MORF was also compared to DSTARS in our experiments, and it is described in details in Section 2.3.

Local-based methods were also proposed to treat MTR problems as enhanced single output tasks, while exploring inter-targets properties. Indeed, the use of more than one predictor to solve a MTR problem leads to decrease the interpretability and increase the computational costs of the generated solution [7].

However, this kind of approach offers some advantages. As previously mentioned, the solution's modularity and conceptual simplicity are increased [7,8]. In addition, the possibility of using any class of base learner, even a hybrid set, tends to result in better predictive performance and enables the exploration of particular characteristics of the dealt problems. A set local-based methods were proposed in the past years, exploring different data manipulation strategies [8,9,13,14,17,21,26].

Some of the recent methods have been adapted from multi-label classification [7,8]. Spyromitros-Xioufis et al. [8] proposed SST (Stacked Single-Target) and ERC (Ensemble of Regressor Chains), which inspired some other methods, including ours. Their idea is the application of targets' approximations as explaining features, as proposed in the Cascade Generalization [27]. As done in their the proposal and extended to ours, this strategy will be from here onward referred simply as *stacking*. SST and ERC are compared to DSTARS, hence more details about them are exposed in the Sections 2.1 and 2.2.

2.1. Stacked single-target

The SST method consists of training ST models and using their outputs as additional prediction features. In this way, taking into consideration an input set composed by m features, $\mathbf{X}$, and d continuous target variables, $\mathbf{Y}$, SST uses the ST predictions $\hat{\mathbf{Y}}$, in the form $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_d\}$, as new inputs, forming an augmented training set $\mathbf{X}'$ with tuples in the form $\mathbf{x}' = \{x_1, x_2, \ldots, x_m, \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_d\}$. The transformed data is used along with $\mathbf{Y}$ to train another set of ST predictors, whose outcomes are the final predictions.

In this sense, SST explores the possible existing inter-target dependencies by statically adding predictions for all targets as new inputs. However, there are no guarantees that all targets influence and are influenced by the others in all cases. In fact, this method makes no distinction about the existing correlation levels among the targets.

Regarding the computational complexity, consider b as the cost of the chosen base regressor technique. The SST time complexity is equal to $O(2\,d\,b)$, which is twice the number of models employed by the ST strategy. Consequently, the ST time complexity is $O(d\,b)$.

2.2. Ensemble of regressor chains

The ERC method consists of using a set of randomly ordered target groups, referred as chains, to build ST regressors sequentially. For each chain, initially, an ST model is induced using the first output variable of the sequence. New models are trained following the regressor chain (RC) order and the stacking strategy: each new regressor uses an augmented input dataset formed by the combination of the original input variables and the past models' predictions. This process is repeated until the end of all chain sequences. After training all models, new income values are subjected to the set of RCs and their responses are the average predicted values for each target.

Since the output predictions are composed of values from different chains positions, multiple levels of combinations and inter-dependence between targets are explored, albeit in an exhaustive approach influenced by randomness. Given the high number of potential combinations among the outputs, ERC creates all possible target permutations if this number is less than 10 (when $d \leq 3$), otherwise exactly ten random combinations are selected.

Regarding the ERC cost, there are two possibilities. The first scenario corresponds to use all the possible target permutations, which results in a complexity of $O(d^2\,(d-1)!\,b)$, since d regressors are induced for each RC. Nevertheless, by following the ERC's authors recommendation of employing ten random combinations when $d > 3$, the resulting time complexity is $O(10\,d\,b)$.

2.3. Multi-objective random forest

Traditionally, decision and regression trees aim at finding the underlying relationship among the input features to explain a single output variable. In regression tasks, for instance, the split decisions are often made intending to minimize the variance of the output data in the induced partitions [1]. When dealing with multiple outputs, some different or adapted strategy must be employed.

Blockeel et al. [28] proposed a framework for top-down induction of predictive clustering trees (PCTs). In their approach, each data partition begins to be seen as a cluster which holds the cases separated by a decision made over a feature variable. The root node represents the cluster that contains all the problem's instances. The splits are made with the goal of maximizing the homogeneity of the induced sub-clusters. For Multi-target regression Trees (MTRT), the maximization of homogeneity is defined as the reduction in output data variance [28]. Similarly, the average value of the leaf clusters are the responses in these tree structures.

Kocev et al. [18] employed the PCT framework to generate ensembles of MTRT trees. The Bagging and Random Forest (RF) strategies were evaluated, being the latter the best evaluated one. The RF ensemble was called Multi-objective Random Forest (MORF). In the following years, MORF was evaluated in multiple MT problems [5,12], and it was frequently used as a comparison in MTR works [6–8,13,14].

As discussed in Kocev et al. [5], the time complexity to build MORF is $O(k\,(m'N \log^2 n + dm'n \log n))$. In this expression, k represents the number of trees in the forest, m' is the size of the subset of features which are sampled for building each tree, d is the number of targets, and n the number of training instances.

3. Deep structure for tracking asynchronous regressor stacking

DSTARS explores the available data seeking to find the best composition of stacked regression models to decrease the prediction error. For convention, the amount of regressors used to predict a single response is from here onward referred as the number of *regressor layers*. In this sense, ST uses a single layer of regressors for each target, considering only the original input set as descriptors. On the other hand, SST is composed of two regressor layers: the second layer employing the outputs of the first one as additional features. The most suitable number of regressor layers is computed by DSTARS, improving the predictive performance for each target individually.

Our method can be divided into five steps: *Data Partition*, *ST Induction*, *Filtering*, *Tracking*, and *Modelling* (Fig. 1). Firstly, the input and output spaces are partitioned in f training and validation groups (Step 1) towards reducing possible bias and overfitting when searching for the best regressor layers composition. This is performed by employing a consensus scheme to select whether a specific layer integrates into the final predictive model. The data partition is performed using a sampling strategy such as cross-validation or bootstrap sampling. Step 2 is grounded on creating a first layer of single-target regressors, since in a scenario of independence among the targets, just a regressor layer would be needed. Conversely, in a scenario of inter-dependent targets, DSTARS takes advantage of those relationships by fashioning deep regressor layers. This is explored in the next steps.

The *Filtering* phase (Step 3) measures how each target is influenced by the others, calculating the correlation level among them on different partitions, as defined in Step 1. This information enables choosing whether a target's prediction will be stacked when predicting other responses in the next steps. The *Tracking* phase

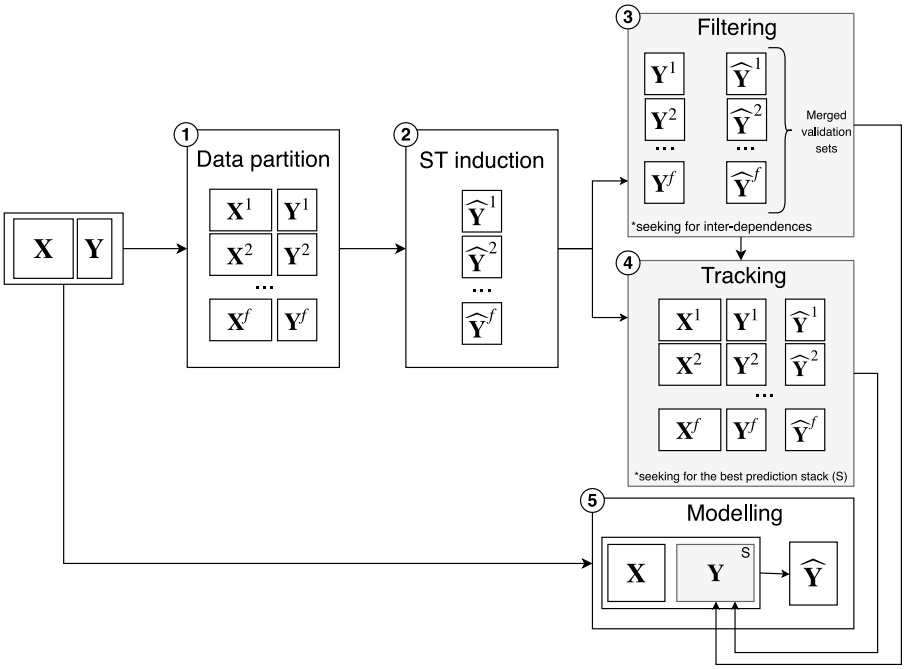


Fig. 1. General overview of the DSTARS method.

(Step 4) evaluates the number of regressor layers able to reduce the prediction error for each target, exploring the training and validation subsets, defined in Step 1. This phase is grounded on hyperparameters ε and ϕ. The minimum expected error reduction by adding a new regressor is defined by ε. The ϕ hyperparameter is related to the minimum percentage of contribution from a given layer in the total error reduction. The *Filtering* and *Tracking* phases determine the structure of regressor layers that is used to train the final DSTARS prediction model in the *Modelling* phase (Step 5).

Algorithm 1 shows how the DSTARS' steps interact with each other, based on the hyperparameters: input and output sets ($\mathbf{X}$ and $\mathbf{Y}$), ε, ϕ and the sampling strategy (specified in Step 1).

Algorithm 1 Complete DSTARS procedure

1: **function** DSTARS($\mathbf{X}$, $\mathbf{Y}$, ϕ, ε, `sampling_strategy`)
2: Separate $\mathbf{X}$ and $\mathbf{Y}$ using `sampling_strategy` in f partitions
3: Induce regressors h_t^p for each target t and partition $(\mathbf{X}^p, \mathbf{Y}^p)$, $p \in [1..f]$
4: Save the predictions for each partition: $\hat{\mathbf{Y}}^p = \{\hat{y}_1^p, ..., \hat{y}_t^p, ..., \hat{y}_d^p\}$
5: Compute the prediction error for each target and partition: e_t^p
6: Get the filtering information: $F \leftarrow Filtering(\mathbf{Y}^{1..f}, \hat{\mathbf{Y}}^{1..f})$
7: Search for the best regressor layer structure: $T \leftarrow Tracking(\mathbf{X}^{1..f}, \mathbf{Y}^{1..f}, \hat{\mathbf{Y}}^{1..f}, e_{1..d}^{1..f}, \varepsilon, \phi)$
8: Create the final prediction model: $M \leftarrow Modelling(\mathbf{X}, \mathbf{Y}, F, T)$
9: **return** M
10: **end function**

We employ the Root Mean Squared Error (RMSE) to calculate the regressors' prediction error throughout the DSTARS phases. RMSE is calculated according to Eq. (1), where n represents the number of test instances, y_i the true responses for the ith test case in the evaluated target, and $\hat{y}_i$ the predictions obtained for the same example.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1}$$

3.1. Filtering

The *Filtering* phase aims at determining the level of influence of the targets among themselves. Herewith, only the responses which are correlated to a given target are stacked as additional features for it. Uncorrelated targets with the others generate independent ST tasks. Previous researches employed the linear correlation metric to measure how the targets influence each other [14,17], but assuming only linear relationships is a non-realistic proposition. Thus, DSTARS uses a nonlinear metric to compute the target influence towards improving the description of the task's intrinsic characteristics.

Looking forward to fulfilling this requirement, the Random Forest Variable Importance [2,29] was chosen. This metric, here referred as `RFimp`, can handle both classification and regression tasks. Besides, `RFimp` does not presuppose a linear relationship between the explaining features and the evaluated response. Despite our choice, any preferred variable importance metric could be employed to describe the inter-target relationships.

Although using the targets' true values for calculating the `RFimp` is possible, the result tends to be too optimistic and biased. In practical applications, the obtained prediction error for a target could be very high. Thus, the estimations for this response might not present the same statistical dependencies as those measured from the real observed values. For that reason, DSTARS measures the `RFimp` between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$. DSTARS only stacks target predictions with non-zero and positive `RFimp` considering all the f data partitions. Thus, $\hat{\mathbf{Y}}$ represents the merged ST predictions over the validation groups of the original output set $\mathbf{Y}$.

Algorithm 2 presents the DSTARS `Filtering` function. This function receives $\hat{\mathbf{Y}}$ and $\mathbf{Y}$ as inputs, and computes the `RFimp` of each element in former set when explaining the targets in the latter.

Algorithm 2 DSTARS Filtering step

function FILTERING($\hat{\mathbf{Y}}$, $\mathbf{Y}$)
  Let F be a $d \times d$ matrix to store the inter-target correlations
  for each target y_t in $\mathbf{Y}$ **do**
    Train a RF regressor h using $\hat{\mathbf{Y}}$ and $\mathbf{y}_t$
    Measure the RFimp values with h and save them in $F[t, 1..d]$
  end for
  return F
end function

3.2. Tracking

The *Tracking* phase searches for the adequate number of regressor layers using the previously defined training and validation sets. Its central idea is to induce successive regressor layers and account whether each created model was able to reduce the prediction error. Each new regressor created for a target y_t, $t \in [1, 2, \ldots, d]$, uses the input set $\mathbf{X}$ augmented with predictions of the most accurate regressors created so far for all the targets related to $\mathbf{y}_t$ accordingly to F (obtained in the Filtering phase). By the most accurate regressor, we mean the predictor related to a regressor layer which was able to obtain the smallest error for a target among all the evaluated models until that moment. Initially, the best predictors are those defined at the beginning of DSTARS (see Step 2 of Fig. 1).

We consider that a newly added regressor improves upon the previous ones when it reduces the prediction error by at least a minimum error improvement criterion ε. The exploratory process stops when no gains greater than ε are observed in any targets. Hence, ε could also be seen as a convergence criterion for DSTARS. Note that a regressor layer might not decrease the error (in at least ε) for a given target, but its successors could achieve this by using improved stacked predictions. Our method asynchronously selects the best current predictors for each target to compose an additional set of features, i.e., the best predictions may come from different regressor layers.

The combination of target and regressor layer that reduced the error is accounted in each case when evaluating all the defined data partitions. Normalizing these occurrences by the total number of data partitions, we can apply a threshold ϕ to select for each target only the regressor layers that brought improvements in at least ϕ percent of the occurrences. After applying ϕ, some regressor layers may be bypassed, creating a regressor layer discontinuity. In this case, no model is created for that specific regressor layer and target combination. The final regressor layers for each target compose the final predictive model of DSTARS.

It is important to highlight the generally expected behaviour for different ε and ϕ choices. Using greater values for ε leads to explore fewer regressor layers during the Tracking since only predictors which decrease the prediction errors in great extents will be considered. This solution can be sub-optimal. On the other hand, setting ε closer to zero enables gradual improvements to be obtained, but could also lead to DSTARS overfitting in the training data. Regarding ϕ, values close to 1 mean that only the regressor layers selected in almost all data partition compose the final DSTARS model. In fact, setting ϕ to 1 or a value very close to it leads the final DSTARS models behaving very similar to the ST or the SST method, or a hybrid of them. Likewise to ε, choosing ϕ values close to zero enables more regressor layers to be used, but also could lead to overfitting the solution to the training data.

The `Tracking` function is presented in Algorithm 3. This function receives the data partitions combined with the ST predictions and their respective errors, the threshold ϕ, the error convergence criterion ε, and the previously defined filtering values F.

Algorithm 3 DSTARS Tracking step

function TRACKING($\mathbf{X}^{1..f}$, $\mathbf{Y}^{1..f}$, $e^{1..f}_{1..d}$, ϕ, ε, F)
  Let T be a list of applicable regressor layers
  Initialize T:
    $T[1, 1..d] \leftarrow f$
  for each $(\mathbf{X}^p, \mathbf{Y}^p, \hat{\mathbf{Y}}^p) \in (\mathbf{X}^{1..f}, \mathbf{Y}^{1..f}, \hat{\mathbf{Y}}^{1..f})$ **do**
    Let l be the regressor layers count, starting with 2
    while error decreases for some target **do**
      for $t \leftarrow 1$ **to** d **do**
        Let $\mathbf{X}'$ be the input set $\mathbf{X}^p$ augmented with $\hat{\mathbf{Y}}^p_v$ such that $v \in [1..d]$ and $F[t, v] > 0$
        Create regressor h_t using $\mathbf{X}'$ and $\mathbf{y}_t$
        Compute the error for h_t: $\tilde{e}$
        if $\tilde{e} + \varepsilon < e^p_t$ **then**
          Update e^p_t and $\hat{\mathbf{Y}}^p_t$
          Increment $T[l, t]$ by one
        end if
      end for
      Increment l by one
    end while
  end for
  Normalize T by f
  $T = \begin{cases} \text{True}, T \geq \phi \\ \text{False}, T < \phi \end{cases}$
  return T
end function

3.3. Modelling

The final DSTARS phase consists of inducing the regressor layers which were defined in the Tracking over the original input and output variables. The inter-target relations defined in the Filtering phase are also employed to isolate the possible ST tasks and to choose only the relevant responses to be stacked when estimating each target. Algorithm 4 presents the DSTARS Modelling phase. The function receives the original input ($\mathbf{X}$) and output ($\mathbf{Y}$) variables, the filtering (F) and the regressor layer tracking (T) information, which were defined in the previous DSTARS phases.

Algorithm 4 DSTARS Modelling step

function MODELLING($\mathbf{X}$, $\mathbf{Y}$, T, F)
  Let L be the maximum number of regressor layers in T
  Let M be an empty set to store the regression models
  Let $\hat{\mathbf{Y}}$ be an empty matrix to store targets' predictions
  for l $\leftarrow$ 1 **to** L **do**
    for t $\leftarrow$ 1 **to** d **do**
      if $T[l, t] = True$ **then**
        Let $\mathbf{X}'$ be the input set $\mathbf{X}$ augmented with $\hat{\mathbf{Y}}_v$ such that $v \in [1..d]$ and $F[t, v] > 0$
        Induce regressor model h^l_t using $\mathbf{X}'$ and $\mathbf{y}_t$
        Update $\hat{\mathbf{Y}}$
        Save h^l_t in M
      end if
    end for
  end for
  return M
end function

Given a trained DSTARS model, new instances will be sequentially subjected to the trained predictors following the regressor layers order. Each predictor provides target estimations which

Table 1
Name, amount of examples, number of input and output variables, and description of the datasets used in the experiments.

Dataset	#Examples	#Input	#Outputs	Description
atp1d	337	411	6	Minimum air ticket price in the following day for different airline options and number of stops.
atp7d	296	411	6	Minimum air ticket price over the successive seven days day for different airline options and number of stops.
oes97	334	263	16	Surveys concerning occupational employment that contains the estimated number of employees in different jobs in 1997.
oes10	403	298	16	Surveys concerning occupational employment that contains the estimated number of employees in different jobs in 2010.
rf1	9125	64	8	River flows after 48 h in the Mississippi River network.
rf2	9125	576	8	River flows after 48 h in the Mississippi River network with precipitation forecast information included as input variables.
scm1d	9803	280	16	Prices of several products in the next day in a supply chain management.
scm20d	8966	61	16	Mean price of different products over the next 20 days in a supply chain management.
edm	154	16	2	Parameters of electrical discharge machining defined by a human operator during .
sf1	323	10	3	Number of times common, moderate and severe solar flares were observed in a 24 h interval.
sf2	1066	10	3	Number of times common, moderate and severe solar flares were observed in a 24 h interval.
jura	359	15	3	Concentration of heavy metals in the top soil of Jura (Switzerland).
wq	1060	16	14	Relative representation of plant and animal species in Slovenian rivers.
enb	768	8	2	Heating and cooling load specifications for energy efficient buildings.
slump	103	7	3	Slump, flow and compressive strength concrete characteristics based on the quantity of concrete elements.
andro	49	30	6	Water quality parameters in Thermaikos Gulf of Thessaloniki, Greece.
osales	639	413	12	Online monthly sales of a product during its first twelve months.
scfp	1137	23	3	Number of views, clicks and comments on online issues.

will act as new input features to the subsequently stacked regressors. In all cases, the last regressor layer for a target will provide the final predictions.

3.4. Complexity analysis

The total time complexity of DSTARS relies on certain choices which must be made before its execution. These aspects include the sampling strategy employed to separate the data and the hyperparameters ϕ and ε which must be set up. Regarding the different DSTARS phases, the Filtering step has $O(d\,r)$ time complexity, where d represents the number of targets and r the cost to calculate the chosen importance metric for one of the responses. The Tracking step has a time complexity of $O(f\,L\,d\,b)$, where f represents the number of employed data partitions, L the maximum number of regressors which were trained until all targets converged, and b the time complexity of the chosen regression technique. The modelling step has $O(L\,d\,b)$ time complexity.

Therefore, by putting $L\,b$ in evidence for the Tracking and Modelling phases, and also, separating the d term in all presented costs, the total time complexity of our proposal is $O(d\,[r+(L\,b)\,(f+1)])$. Thus, the number of targets and induced data partitions greatly influence the final time complexity of DSTARS. Moreover, the number of regressor layers obtained during the experiments, presented in Section 5.6 and in the Appendix A, and also the measured running time (Section 5.5) could offer more intuition about the cost of inducing DSTARS models in real scenarios. Last, these results could also give insights on the memory requirements of DSTARS, since the memory complexities of the local-based methods directly depend on the chosen regressor techniques costs.

4. Experimental setup

This Section presents the benchmark datasets used to compare DSTARS with other MTR methods, together with the employed base regression algorithms and software libraries. Lastly, we present the evaluation metrics which were used to measure the prediction performance of each evaluated MTR method. All experiments were performed in the same 64-bit Linux machine with Processor Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40 GHz with two sockets of 14 cores and 128 GB of RAM.

4.1. Datasets

Eighteen MTR benchmark datasets,[1] already used in former studies [7,8,30] and widely explored in the research field, were adopted in this work. The characteristics of these datasets are registered in Table 1. The datasets comprehend diverse kind of multi-variate regression problems and application areas, as

[1] The datasets can be found in: http://mulan.sourceforge.net/datasets-mtr.html.

described in Table 1. All the datasets were re-scaled in the [0, 1] range prior their manipulation by the MTR approaches.

4.2. Regression techniques

We employed two base regression techniques combined with the compared local MTR methods: Support Vector Machine (SVM) with the Radial Basis Function (RBF) kernel and Random Forest (RF). All regression algorithms used in this work were implemented in the R programming language,[2] and used with their standard hyperparameter settings. he only exception was MORF, which was performed using the Clus Framework.[3]

4.2.1. Support vector machine

SVM is a classification and regression method that is part of the general category of kernel-based methods [3]. SVMs aims at finding the hyperplane which has the minimum possible distance to the data, accordingly to a defined loss function. To model nonlinear relationships among the input and output variables, SVM employs a kernel space transformation, which is known for kernel trick. In this kernel application, a nonlinear transformation function is chosen to map the original data into a space of possibly greater dimension, where the regressor is able to solve the transformed problem with a linear fitting function. Through the mentioned kernel space transformation, this technique has the flexibility to model varied data sources [3], increasing the input dimensional space but also data separability. SVM was performed through the `e1071` R package.

4.2.2. Random forest

RF is a tree-based ensemble algorithm proposed by Breiman [2], which builds multiple regression trees over the training data. Each new tree is induced using subsets of the original input set which are formed by sampling with repetition the problem instances. This strategy to compose ensemble predictors is called Bagging. Besides, when creating a new tree, only a subset of the features are employed, being this group selected by random sampling without replacement. The RF predictions for new cases are formed by taking the average result over all trees in the Forest [2]. The elements which were not selected by an individual tree for training are called out-of-bag (OOB) cases. In this sense, each tree in the forest has a set of OOB samples which can be used to calculate an unbiased internal metric of out-of-bag error (OOBE), and to assess the importance of each explaining variable. The RF variable importance is a metric calculated by comparing the difference on the OOBE before and after randomly permuting the values of each of the features on the OOB cases [2,29]. We employed the `ranger` R package in our experiments for computing the RF predictors.

4.3. DSTARS hyperparameters

During our experiments, we used 10-fold cross-validation as the `sample strategy` setting of DSTARS. Aiming at verifying whether would exists a default hyperparametrization, adequate for most cases, we evaluated multiple possible settings for our proposal using a grid search.

The error improvement hyperparameter, ε, assumed the values 10^{-2}, 10^{-3} and 10^{-4}. In addition, we computed DSTARS models varying the ϕ hyperparameter in the [0, 1] interval with a 0.1 step. Thus, all the possible ϕ threshold values for selecting the regressor layers were explored, according to the chosen data partition strategy. We discuss in Section 5.1 how the different hyperparameter settings impacted the predictive performance of DSTARS.

[2] Available in: Multi-Target Regression Framework (R Language): https://github.com/smastelini/mtr-toolkit.

[3] Available in: https://dtai.cs.kuleuven.be/clus/.

Table 2
aRRMSE results regarding all evaluated datasets, MTR methods and regressor algorithms.

Dataset	Regressor	ST	SST	ERC	DSTARS	MORF
atp1d	RF	0.3920	0.3904	**0.3902**	**0.3902**	0.4474
	SVM	**0.4397**	0.4402	0.4398	0.4404	
atp7d	RF	0.5164	0.5169	0.5178	**0.5146**	0.5593
	SVM	**0.6404**	0.6414	0.6410	0.6417	
oes97	RF	0.5164	0.5135	**0.5133**	0.5147	0.5966
	SVM	0.6118	0.6123	**0.6110**	0.6110	
oes10	RF	0.4070	0.4081	**0.4070**	0.4079	0.4641
	SVM	0.5464	0.5456	**0.5451**	0.5456	
rf1	RF	0.0782	0.0582	0.0731	**0.0561**	0.1206
	SVM	0.1215	0.107	0.1151	**0.1015**	
rf2	RF	0.0847	**0.0784**	0.0852	0.0821	0.4076
	SVM	0.1095	**0.1064**	0.1084	0.1067	
scm1d	RF	0.2871	0.2758	0.2823	**0.2722**	0.3741
	SVM	0.3309	0.3232	0.3258	**0.3214**	
scm20d	RF	0.3648	0.3313	0.3347	**0.3129**	0.5171
	SVM	0.3972	0.3522	0.3474	**0.3316**	
edm	RF	0.6721	**0.6631**	0.6661	0.6766	0.6791
	SVM	0.7699	0.7714	**0.7667**	0.7829	
sf1	RF	1.0051	1.128	1.0161	**1.0047**	0.8798
	SVM	0.9390	0.9477	**0.9250**	0.9604	
sf2	RF	**0.8487**	0.941	0.8617	0.8577	0.7689
	SVM	0.7825	0.7787	0.7827	**0.7764**	
jura	RF	0.6061	0.5974	0.5969	**0.5908**	0.7003
	SVM	0.6409	0.6413	**0.6391**	0.6410	
wq	RF	0.9066	0.9392	**0.9059**	0.9104	0.9271
	SVM	0.9630	**0.9535**	0.9581	0.9576	
enb	RF	0.1504	0.1173	0.1304	**0.1149**	0.2136
	SVM	0.2499	0.2173	0.2404	**0.1678**	
slump	RF	0.8365	0.8324	**0.8222**	0.8338	0.8169
	SVM	0.6924	0.6862	**0.6819**	0.6889	
andro	RF	0.7941	0.7349	0.7614	**0.6584**	0.7068
	SVM	1.1348	0.9243	1.0089	**0.7899**	
osales	RF	0.7577	**0.7275**	0.7332	0.7289	0.9516
	SVM	1.1726	**1.1685**	1.1702	1.1717	
scfp	RF	0.9263	0.9379	**0.8778**	0.9017	0.8459
	SVM	0.8242	0.8256	**0.8151**	0.8251	

4.4. Evaluation metrics

Intending to evaluate the models constructed during the experiments, four different metrics were used: Relative Root Mean Squared Error (RRMSE), average Relative Root Mean Squared Error (aRRMSE), Relative Performance (RP), and the running time of the evaluated approaches. Besides that, the MTR methods were executed using the 10-fold cross-validation strategy.

The RRMSE (Relative Root Mean Squared Error) is obtained from the squared error measured from a target, divided by the squared error which is obtained when always predicting the average value of this response. This last behaves like a baseline in the metric, allowing to compute the improvement over a shallow predictor. This metric has been used in various MTR works [7,8,14] to compare non-homogeneous targets distributions. The aRRMSE is computed by averaging the d targets' RRMSE, as presented in Eq. (2). In the equation, $\hat{y}$ and $\bar{y}$ represent, respectively, the predicted values for a target y and its mean value, while N_{test} represents the number of test cases.

$$\text{aRRMSE} = \frac{1}{d}\sum_{t=1}^{d}\sqrt{\frac{\sum_{k=1}^{N_{\text{test}}}(y_t^k - \hat{y}_t^k)^2}{\sum_{k=1}^{N_{\text{test}}}(y_t^k - \bar{y}_t)^2}} \tag{2}$$

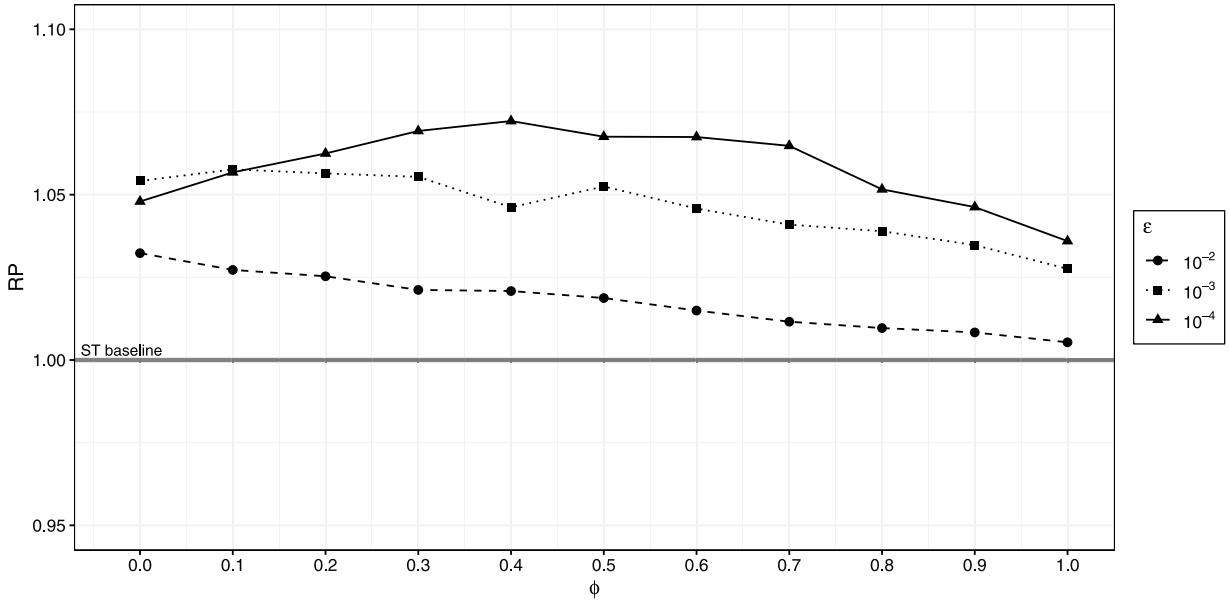


Fig. 2. Mean RPs achieved for each DSTARS' (ϕ, ε) evaluated combination.

The Relative Performance (RP) compares the aRRMSE of ST strategy with the aRRMSE of another local-based MTR method M, in our case, with SST, ERC, and DSTARS. In this sense, it can measure whether there was an increase (RP $>$ 1) or a decrease (RP $<$ 1) relatively to the simple ST strategy [8]. The RP calculation is presented in Eq. (3).

$$\text{RP}(M) = \frac{\text{aRRMSE}(ST)}{\text{aRRMSE}(M)} \tag{3}$$

In addition, the presented error metrics support the comparison of possible method superiority through the application of the Friedman statistical test and the Nemenyi post hoc test with the Critical Difference (CD) diagram, as previously proposed in [31]. We employed these tests to verify the possible statistical differences among the compared MTR approaches, regarding their predictive performance.

Relatively to the measured running time, only ST, SST, ERC, and DSTARS were considered in our computations. Firstly, the mentioned MTR methods belong to the same approach, problem transformation, as previously presented in Section 2. Therefore, it is expected that the global-based method (MORF) would be faster than the local-based ones since it creates a single predictor for all targets, instead of multiple ST regressors. However, our primary motivation to only consider the local-based methods was because all of them were implemented in the same programming platform (R, an interpreted programming language), whereas MORF was performed using the original Clus implementation (Java, a compiled programming language). Thus, it would not be fair to compare methods implemented in such different platforms concerning their running times. Anyhow, we measured the running times when using RF as regressor averaged between 30 repetitions for each dataset and MTR approach.

5. Results and discussion

This section describes and discusses the obtained results when comparing the performance of the five considered methods for MTR problems (ST, SST, ERC, MORF and DSTARS). Firstly, we perform a tuning of DSTARS parameters, as described in Section 4.3, aiming at observing the obtained errors and choosing a candidate to standard set of hyperparameters (ϕ, ε) for our proposal. After, we compare the performance of our proposal using this default configuration with the state-of-the-art solutions for MTR problems. Following, we present a comparison of the local MTR methods with the simpler ST approach. Statistical tests comparing the performance of all compared methods are also presented. We also expose the obtained running time for ST and the compared local MTR methods. Lastly, we analyse aspects of the generated DSTARS models, including the number of trained regressors and the discontinuity in the generated regressor layers.

5.1. Tuning the DSTARS hyperparameters

We followed the approach defined in Section 4.3 to seek for a set of hyperparameters (ε, ϕ) suitable for most of the cases. Thus, if such a set exists, we could adopt it as a default for DSTARS, using these parameters from here onward when comparing our approach with the others. In this sense, we also could provide a fair comparison, since no adjustment step was performed for the other methods.

Therefore, we ran multiple experiments varying the values of ϕ and ε accounting for the resulting aRRMSEs of each parameter combination. Our complete observations are presented in Appendix A, where we summarized the error variation per dataset and regression technique using line plots. To provide global useful insights on how the ϕ and ε combinations impacted in the obtained results, we comprised all the results by mean RP of each hyperparameter set combination among all datasets and regressors. This was done to enable us comparing different ranges of error in the same analysis, presented in Fig. 2.

The ST baseline was employed to calculate the RPs, as indicated in the figure. Firstly, we can see that the $\varepsilon = 10^{-4}$ curve was the best in almost all cases, except for $\phi = 0.0$. The usage of high ε originated the worst results, as seen in the curve $\varepsilon = 10^{-2}$. Indeed, stacking layers with lower ε criteria led the final DSTARS model to better results.

Regarding ϕ values, the RPs' peak was obtained with the value 0.4 leveraging the combination $(\phi, \varepsilon) = (0.4, 10^{-4})$ as default parametrization for DSTARS, which is going to be used from here onward in our discussions.

Table 3
Obtained running time when comparing the MTR local methods in 30 repetitions with the RF regressor.

Dataset	ST	SST	ERC	DSTARS
atp1d	11.68 ± 5.79	24.62 ± 5.70	96.97 ± 5.34	569.79 ± 12.46
atp7d	11.41 ± 6.90	20.90 ± 6.87	74.09 ± 4.19	554.18 ± 11.75
oes97	24.32 ± 7.85	48.82 ± 6.78	216.62 ± 4.98	1837.15 ± 27.13
oes10	28.40 ± 6.20	60.55 ± 7.24	278.94 ± 2.78	2441.68 ± 25.74
rf1	76.67 ± 8.01	152.49 ± 10.74	726.13 ± 9.00	6719.74 ± 23.50
rf2	140.11 ± 11.69	288.05 ± 8.85	1465.91 ± 4.85	10 400.98 ± 49.32
scm1d	417.35 ± 7.29	914.05 ± 7.95	4974.27 ± 10.05	68 387.79 ± 473.61
scm20d	163.87 ± 4.84	377.17 ± 7.67	1997.11 ± 4.51	56 131.47 ± 388.84
edm	0.97 ± 0.89	2.32 ± 1.89	1.95 ± 0.81	31.51 ± 5.78
sf1	1.27 ± 1.05	2.67 ± 2.06	5.16 ± 1.14	35.83 ± 4.15
sf2	1.59 ± 1.03	3.32 ± 2.92	6.39 ± 1.32	57.45 ± 6.60
jura	1.67 ± 2.12	4.09 ± 2.95	7.05 ± 1.17	89.21 ± 10.58
wq	11.99 ± 4.56	22.47 ± 8.42	114.87 ± 2.48	681.00 ± 11.17
enb	2.13 ± 2.33	2.10 ± 0.73	2.98 ± 2.66	58.89 ± 5.38
slump	1.45 ± 1.26	2.24 ± 1.10	5.85 ± 1.75	64.15 ± 5.56
andro	2.93 ± 2.61	4.20 ± 1.48	20.27 ± 1.52	197.2 ± 8.31
osales	19.32 ± 3.61	41.84 ± 2.44	221.3 ± 3.48	2253.88 ± 14.71
scfp	2.65 ± 2.27	4.25 ± 3.49	8.60 ± 1.39	106.56 ± 9.72

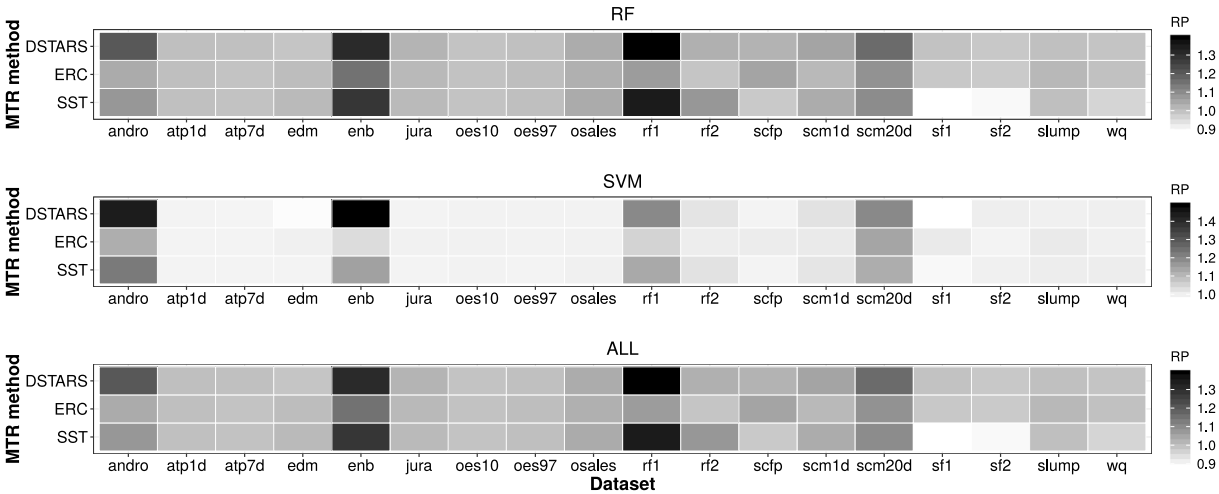


Fig. 3. Comparison of the local MTR methods with ST regarding the RP metric.

5.2. Analysis of prediction error

Table 2 presents the obtained aRRMSE values considering ST and all the compared MTR methods, regression algorithms and datasets. In this table, for each regressor-dataset combination, the method with the smallest aRRMSE is highlighted in bold. Similarly, the smallest prediction error obtained for each dataset is underlined. Considering the 18 datasets and the two regression algorithms, 36 results were reported for each MTR local method. The MORF execution resulted in 18 aRRMSE values since this MTR method belongs to the global category and thus, it is not paired with a regressor technique.

Considering only the local-based methods, the ST strategy reached the smallest aRRMSEs in 3 out of 36 combinations; meanwhile, the SST method was the best choice in 6 out of 36 cases. ERC achieved the best results in 12 cases. Lastly, DSTARS obtained the smallest aRRMSE values in 15 out of 36 combinations. There was one case where DSTARS tied with ERC when using RF as regressor (dataset atp1d).

Regarding only the smallest aRRMSE results per dataset, despite the regressor technique employed by the local-based methods, ST was not ranked as the best choice in any cases. The SST method obtained the smallest errors in three cases (rf2, edm, osales). SST reached these best results when paired with the RF regressor. ERC achieved the smallest aRRMSE values in 6 out of 18 cases, using RF as regressor in four datasets (atp1d, oes97, oes10, and wq), and employing SVMs in the remaining cases (slump and scfp). MORF was the best MTR method in 2 datasets (sf1 and sf2). Lastly, DSTARS resulted in the smallest aRRMSEs in 8 out of 18 cases, all of them using RF as regression technique.

Observing the regressors that were chosen for the local-based methods, the RF technique achieved the best results in 15 datasets (taking into account the tie between ERC and DSTARS) and SVM was the best regressor in 2 cases. In this sense, RF was the best regressor overall. Also, considering that the MORF algorithm corresponds to a multi-output version of the traditional RF technique, it would be possible to fairly compare this method with the local-based methods, as long as they use the RF regressor.

Considering only the decision tree ensembles, as previously stated, SST reached the smallest errors in three cases. ERC resulted in the best results in four datasets. MORF generated the smallest errors in four datasets. Lastly, DSTARS was the best method in eight cases. It is worth mentioning that when not considering SVM, the best MTR ranking changes for the datasets slump and scfp, also altering the overall best MTR methods order. Detailed results for the obtained prediction errors can be found in Appendix B.1 (Table B.5).

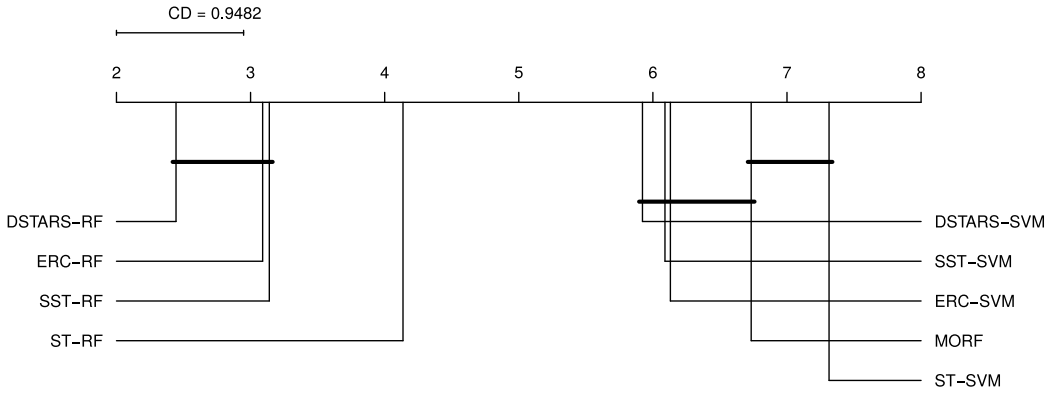


Fig. 4. The Friedman and the post hoc Nemenyi tests results (with $\alpha = 0.05$) when comparing the RRMSE per target of all the evaluated MTR methods and regressors.

5.3. Comparison of the local MTR methods and the ST strategy

Both the ST strategy and the local MTR methods employ traditional regression techniques to deal with MTR problems. However, the local methods manipulate the input space to model the existing inter-target dependencies. In this sense, we analysed whether the more complex MTR methods were able to overcome the more straightforward ST strategy on the evaluated cases. Hence, we employed the RP metric to identify the possible gains.

Fig. 3 presents heatmaps comparing the Relative Performance (RP) of all evaluated local MTR methods over all datasets. The top chart presents the RP results for RF, the middle one for SVM, while the bottom figure shows the metric values averaged among the two evaluated regressors.

Firstly, we can observe that in most of the cases the MTR methods performed better than the ST approach. So, the employment of MTR methods enabled the modelling of inter-target relationships, making possible the obtained gains. In the worst cases, such as sf1 and sf2, almost all MTR methods generated RPs very close to 1, which means there were no notable gains over the ST strategy. In these cases, the additional complexity brought by the MTR variants did not result in any advantage. On the other hand, cases like andro, enb, and rf1 resulted in prominent gains for almost all local MTR methods. It is worthwhile to mention that DSTARS was able to surpass the ST approach even in the sf1 and sf2 datasets.

Regarding the performance of the individual regression techniques, the MTR methods when coupled with RF were able to generate more predictive gains than with SVMs. The authors would like to stress that no tuning was performed for the regressors, as we used their default hyperparameters as implemented in the employed software packages (please refer to Section 4.2). Such adjusts could lead to differences in the prediction errors considering only the regression techniques. However, in the observed scenarios, RF was better than SVM and should be pointed out as the best performer.

Last, observing the aggregated results for both regressors (bottom chart of Fig. 3), it is possible to see that ERC and DSTARS surpassed the ST approach in all cases. The same was not true for SST. Notwithstanding, our proposal was better than ERC by generating more noticeable gains, as again can be seen in datasets andro, enb and rf1, for instance.

5.4. Statistical tests

We performed a set of statistical tests to assess whether the compared MTR methods statistically differed among themselves regarding predictive performance. The Friedman test was first performed to verify possible significant differences (with $\alpha = 0.05$) between the compared approaches. When the obtained p-values were smaller than the significance level, the Nemenyi post hoc was performed to rank the different compared methods. Accordingly to this test, when two methods are not statistically different regarding predictive performance, their ranking differences are smaller than a CD value. In this set of evaluations we considered the errors obtained by target variable (RRMSE) to compare the MTR approaches.

The first study, presented in Fig. 4, observed all MTR methods and regressors. DSTARS appeared in the first position followed by ERC, SST, and ST, all of them employing RF as regression technique. No significative statistical differences were observed among DSTARS, ERC, and SST in this comparison, but the ST strategy was significantly worse than the local-based MTR methods. The cases where SVM was employed as regression technique, in turn, were significantly worse than the ST-RF combination. In this second group of performers, DSTARS again appeared in the first position, being followed this time by SST, ERC, MORF, and the ST approach. Note that SST performed slightly better than ERC when using SVM.

The next comparison exposed the obtained mean RRMSE values when joining the RF and SVM results for each target. MORF was not considered in this comparison since it is a global-based approach. As shown in Fig. 5 DSTARS was significantly better than the other local methods. The second group was composed of ERC and SST. Lastly, ST was significantly worse than the other MTR methods in this comparison.

The last statistical comparison with RRMSE of targets was based on ensemble tree techniques: MORF and local-based MTR methods with the RF regressor, as presented in Fig. 6. DSTARS was the best MTR method, surpassing the other methods by a statistically significant margin. SST and ERC presented very similar ranks, having the first one a slight advantage. Again, the employment of a set of RF regressors significantly surpassed the ensemble of multi-output trees. In fact, MORF was even worse than using the ST approach with RF, as previously observed.

5.5. Running times of the evaluated local MTR methods

The running time was measured for each local MTR method with the RF regressor. MORF was not considered since it is implemented in a different platform. All the methods were performed 30 times and the obtained mean and standard deviation of the execution time are reported in Table 3.

As expected and supported by the asymptotic complexity analysis, the fastest MTR approach was the ST strategy. SST was ranked as the second fastest local MTR in almost all the

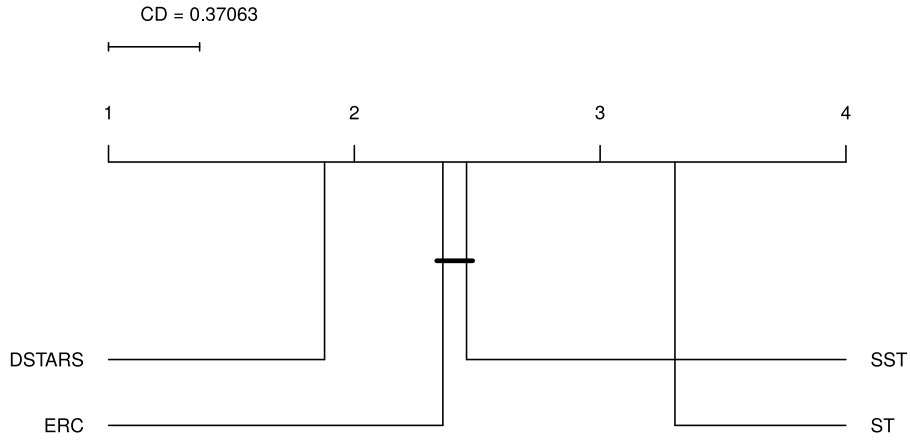


Fig. 5. The Friedman and the post hoc Nemenyi tests results (with $\alpha = 0.05$) when comparing the overall RRMSE per target with only the local-based MTR methods.

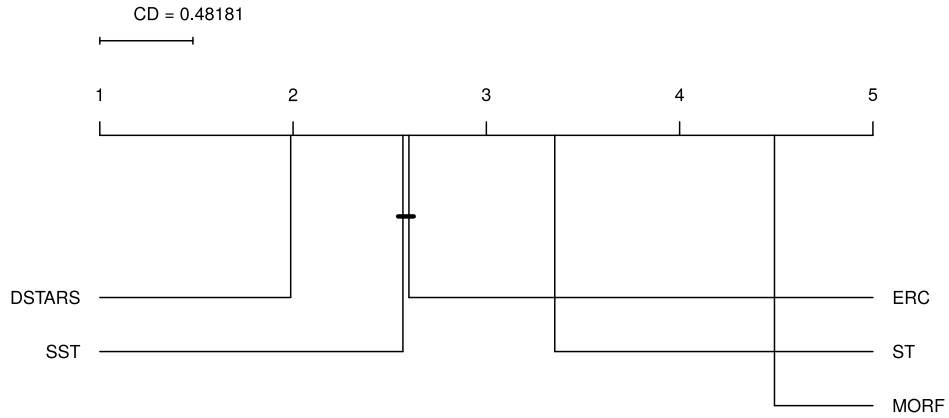


Fig. 6. The Friedman and the post hoc Nemenyi tests results (with $\alpha = 0.05$) when comparing the RRMSE per target with MORF and local-based MTR methods based on RF regressor.

cases (with exception of edm), since it creates twice as many models as ST. ERC was the third fastest method and DSTARS the slowest one. The smallest prediction error of DSTARS presents the trade-off of several operations for describing and modelling the dependencies among the targets. The training costs of DSTARS could be minimized by changing the sampling approach to a more lightweight one. For instance, an internal 5-fold cross-validation could be employed, instead of using a 10-fold cross-validation approach. To further reduce the costs, a single bootstrap sampling or a holdout approach could be also be employed. Nevertheless, once DSTARS is trained, its prediction time overhead (by using multiple regressors) is negligible when compared to the other methods.

5.6. DSTARS: generated models analysis

As discussed previously, the hyperparameter configuration $(\phi, \varepsilon) = (0.4, 10^{-4})$ was selected as default setting for DSTARS. This choice directly impacts on the number of generated models, as well as on the regressor layers structure for each target. Table 4 illustrates the average number of stacked layers in conjunction with the average number of regressor layers discontinuities.

In the majority of cases, DSTARS employed on average less than five regressor layers per target, considering both RF and SVM. An apparent exception was the dataset scm20d and the RF regressor, where an increased number of regressor layers was employed. It is noteworthy that in the same problem, when considering SVM as regressor, fewer layers were employed. Indeed, the number of regression models employed by our method is intimately linked with the choice of the regression technique. This is because the iterative process of Tracking is grounded on the observed errors for the specific regressor chosen. An analysis on how the method performs when using a mixed set of regression techniques, maybe one per target, is an interesting venue for further researches. Such analysis is corroborated by the results discussed in Section 5.2, where different regressors and MTR approaches obtained the best results depending on the evaluated dataset.

Additionally, Table 4 also presents the average number of regressor layers discontinuities. On average, they were more often with RF than with SVM, but rare cases resulted in discontinuities. Interestingly, the case where more discontinuities were observed on average was again the scm20d with RF as the regressor. An analysis of how the obtained errors and unique characteristics of the dealt problems relate to the number of regressor layer discontinuities is also an interesting field for further studies, despite being out-of-the-scope of this study.

Concerning how DSTARS considered the inter-target relations during its construction, we present charts representing the measured `RFimp` values for all datasets and regression techniques in Appendix B.2, as a supplementary material. For all the cases, we highlight, for each target, the responses that were not stacked as additional inputs for it. Lastly, the reader is also encouraged to

Table 4
Average number of regressor layers and the average amount of layer discontinuity observed in the DSTARS models.

Dataset	Layers		Discontinuities	
	RF	SVM	RF	SVM
atp1d	2.9	1.5	0.3	0.0
atp7d	3.0	1.8	0.2	0.0
oes97	2.5	1.5	0.3	0.0
oes10	2.6	1.5	0.3	0.0
rf1	2.5	4.2	0.0	0.1
rf2	2.2	2.2	0.0	0.0
scm1d	5.4	3.0	0.7	0.0
scm20d	13.5	4.6	2.1	0.0
edm	2.2	3.1	0.0	0.0
sf1	1.1	1.6	0.0	0.0
sf2	1.6	2.1	0.0	0.0
jura	3.3	2.0	0.1	0.0
wq	1.1	2.7	0.0	0.1
enb	2.6	5.9	0.0	0.0
slump	3.0	1.6	0.0	0.0
andro	4.0	7.1	0.4	0.1
osales	4.3	1.5	0.5	0.0
scfp	2.1	3.1	0.0	0.0

consult Appendix B.3 for detailed results regarding the number of generated regressor layers and layer discontinuities for DSTARS. There, the observations for each target variable are presented for all datasets and regression techniques.

6. Conclusion

In this paper, state-of-the-art solutions for MTR problems were compared to our new MTR local method, called DSTARS. This new method relies on successive stacked regressor layers to minimize the error for each target, individually. The final DSTARS model comprehends a different number of predictors for each target, which are dynamically and asynchronously defined according to the inter-target dependencies. These dependencies determine whether a target will be subjected to the deep or shallow stacking procedure.

The results showed that DSTARS generated significative smaller RRMSE than the other methods, overcoming the state-of-art algorithms. As future work, we will evaluate DSTARS on other real-life problems, towards the hypothesis that the worst case for DSTARS is to generate ST or SST models in the condition of low statistical dependence among the targets, i.e., our method can mimic the previously mentioned methods by using a fewer number of stacked regressors for each target. We also aim at adapting our method to other multi-output prediction problems, such as multi-label and multi-target classification, and hierarchical classification and regression. Finally, another possible future research direction is the application of local MTR strategies on data stream mining.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to acknowledge to Fundação Araucária (Brazil), CAPES — Coordenadoria de Aperfeiçoamento de Pessoal de Ensino Superior (Brazil, Finance Code 001), CNPq — Conselho Nacional de Desenvolvimento Científico e Tecnológico (Brazil, grants #420562/2018-4 and #400549/2016-6), and FAPESP - São Paulo Research Foundation (Brazil, grant #2018/07319-6), for financial support. The authors also would like to thank Intel AI DevCloud for providing high-performance computational nodes. Additionally, this research was carried out using the computational resources of the Center for Mathematical Sciences Applied to Industry (CeMEAI) funded by FAPESP (grant #2013/07375-0).

Appendix A. Tuning results for DSTARS' hyperparameters

This appendix presents our observations on the aRRMSE variation when varying the ε and ϕ parameters of DSTARS, as described in Section 4.3. As previously discussed in Section 5.1 and presented here, our tuning experiments have shown that the combination $(\phi, \varepsilon) = (0.4, 10^{-4})$ can be pointed out as a satisfactory choice in most of the cases. Therefore, the authors suggest this combination as the standard parametrization for DSTARS when using a 10-fold cross-validation as sampling strategy (Please refer to Algorithm 1).

Fig. A.7 presents the changes in aRRMSE when varying ϕ and ε for both the considered regression algorithms (RF and SVM).

Appendix B. Detailed experiments results

B.1. Observed prediction errors per target

In this appendix we present the obtained errors concerning each target of the evaluated datasets, for all evaluated MTR methods and regressors. The reported metric is the RRMSE. The best results per regression technique are highlighted in bold, while the smallest error obtained by target (independently of the regression technique) is underlined. This analysis is presented in Table B.5. We considered the hyperparameter set $(\phi, \varepsilon) = (0.4, 10^{-4})$ as standard configuration for DSTARS, as discussed in Section 5.1.

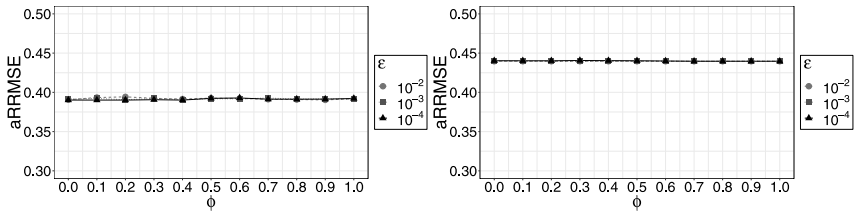
B.2. RF importance during the DSTARS filtering phase

This section presents the `RFimp` measured for each dataset and regressor combination when performing DSTARS. Anytime a negative importance value was observed it was set to zero. The obtained importance values were organized as heatmaps, where each row represent a target and the columns the `RFimp` values measured for it, as described in Section 3.1. Given that the `RFimp` calculation consists in training a RF model and measuring how the input features impact in the OOBE after being randomly permuted, we scaled the observed values between [0, 1].

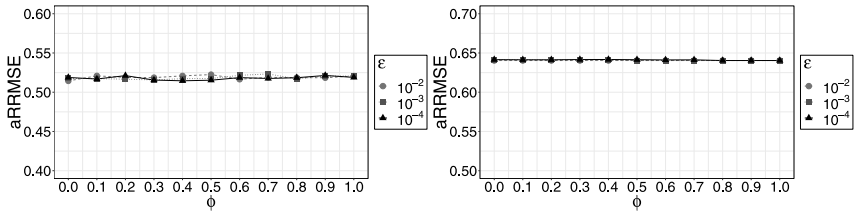
Fig. B.8 presents the `RFimp` observations. Considering that the `RFimp` values were averaged among the iterations of the 10-fold cross-validation employed to evaluate the MTR methods, we marked with an "*" the target combinations where the `RFimp` was smaller or equal to zero at least once. Note that responses with `RFimp` smaller or equal to zero are not added as additional features for the corresponding target.

B.3. Observed number of regressor layers and discontinuities

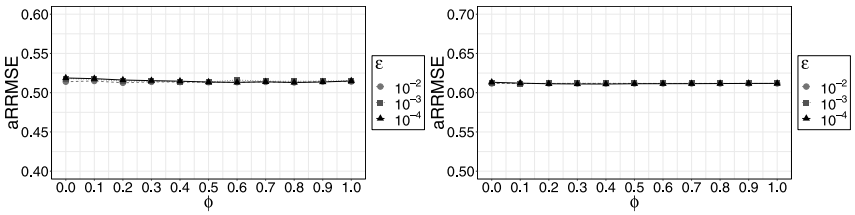
Table B.6 presents the number of regressor layers employed by DSTARS for each target, as well as, the number of observed layer discontinuities. The presented results are averaged among all the cross-validation folds that were used to evaluate the MTR methods. For that reason the number of regressor layers and observed layer discontinuities are in their majority real numbers. In addition, in all the cases, the hyperparameter set $(\phi, \varepsilon) = (0.4, 10^{-4})$ was employed as standard configuration for DSTARS.



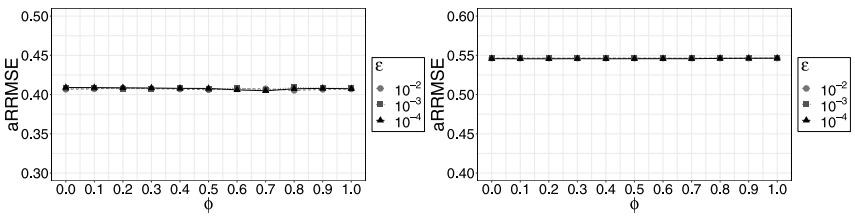
(a) atp1d – RF (b) atp1d – SVM



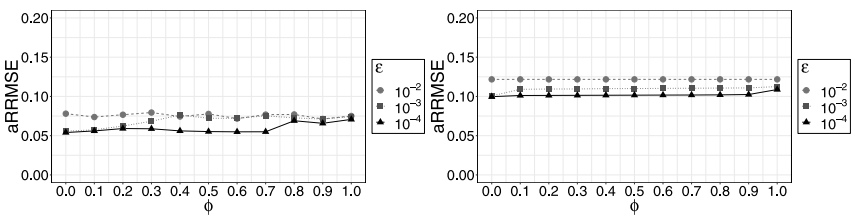
(c) atp7d – RF (d) atp7d – SVM



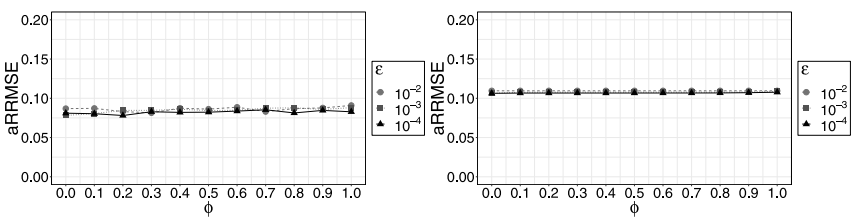
(e) oes97 – RF (f) oes97 – SVM



(g) oes10 – RF (h) oes10 – SVM

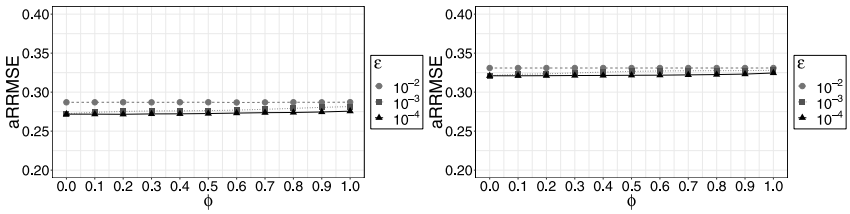


(i) rf1 – RF (j) rf1 – SVM

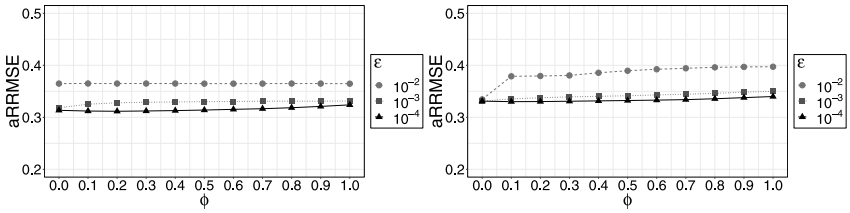


(k) rf2 – RF (l) rf2 – SVM

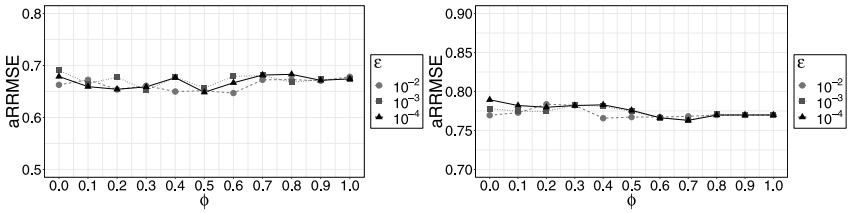
Fig. A.7. Variation of aRRMSE in all evaluated datasets by varying the values of ϕ. Curves for multiple ε are reported in the same charts. Both RF and SVM were considered.



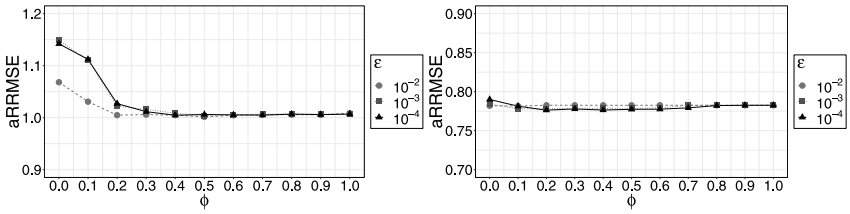
(m) scm1d – RF (n) scm1d – SVM



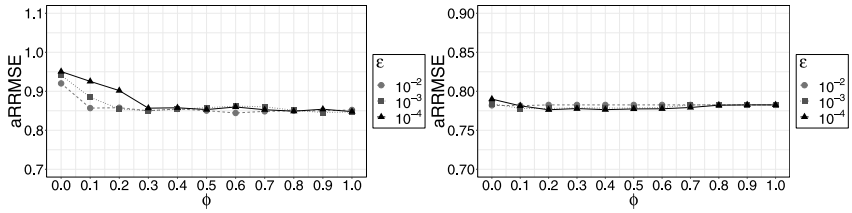
(o) scm20d – RF (p) scm20d – SVM



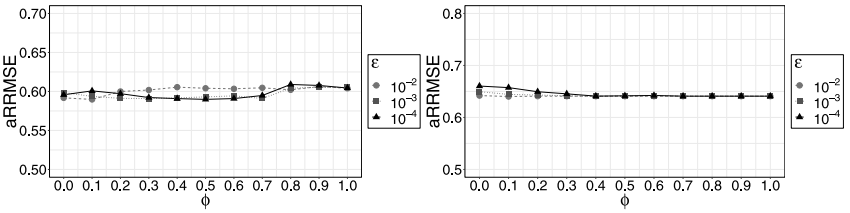
(q) edm – RF (r) edm – SVM



(s) sf1 – RF (t) sf1 – SVM

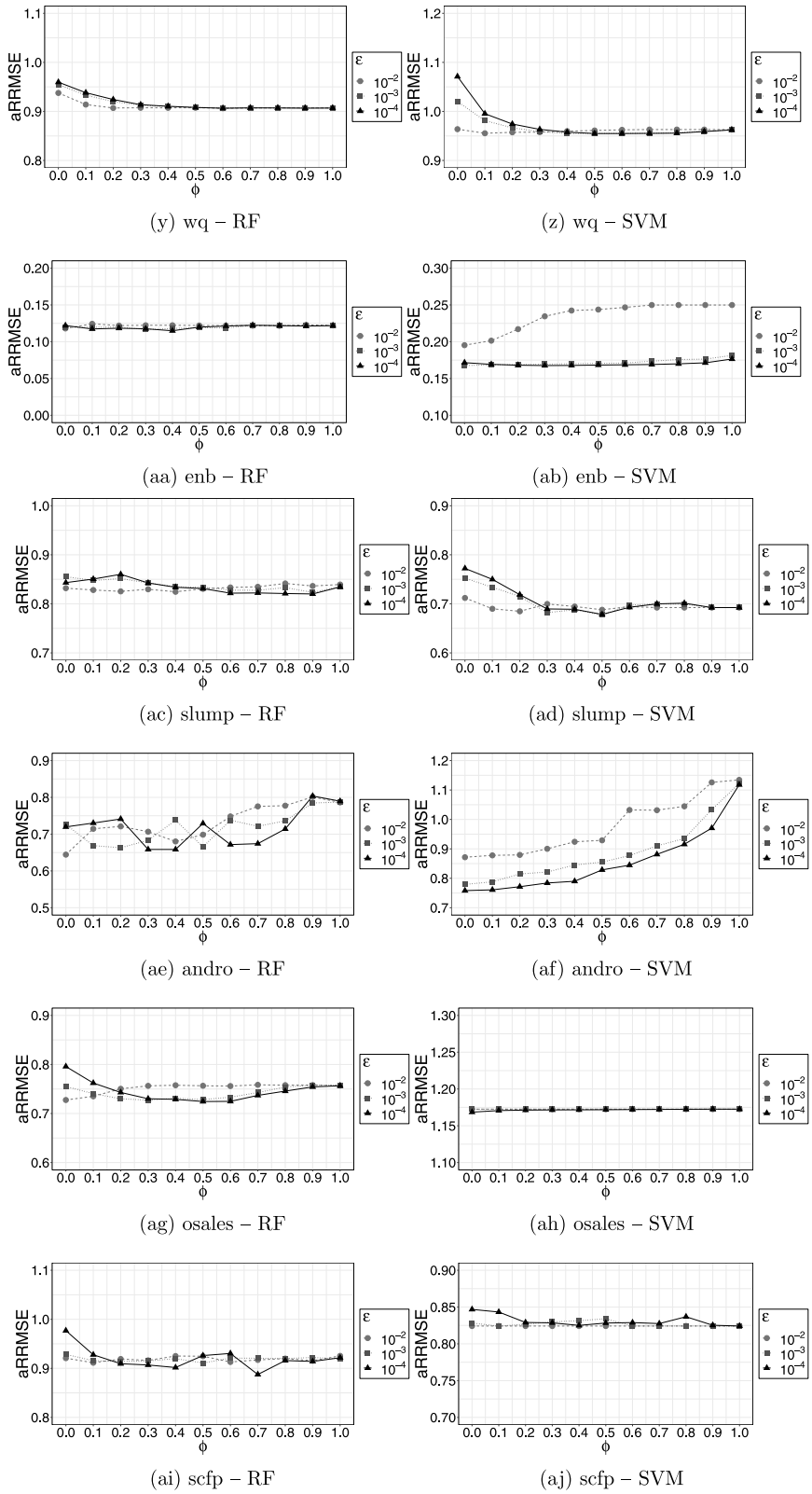


(u) sf2 – RF (v) sf2 – SVM



(w) jura – RF (x) jura – SVM

Fig. A.7. (*continued*).



(y) wq – RF (z) wq – SVM

(aa) enb – RF (ab) enb – SVM

(ac) slump – RF (ad) slump – SVM

(ae) andro – RF (af) andro – SVM

(ag) osales – RF (ah) osales – SVM

(ai) scfp – RF (aj) scfp – SVM

Fig. A.7. (*continued*).

Table B.5
Complete RRMSE results regarding all datasets, MTR methods and regressors. The best results per regressor are highlighted in bold. The smallest errors obtained in each case are underlined. The hyperparameters $(\phi, \varepsilon) = (0.4, 10^{-4})$ were considered for DSTARS.

Dataset	RF				SVM				MORF
Target	ST	SST	ERC	DSTARS	ST	SST	ERC	DSTARS	
atp1d	0.3920	0.3904	0.3902	**0.3902**	**0.4397**	0.4402	0.4398	0.4404	0.4615
ALLminpA	0.4767	0.4788	**0.4766**	0.4788	**0.5435**	0.5444	0.5438	0.5448	**0.4763**
ALLminp0	0.4259	0.4320	0.4273	**0.4241**	**0.4589**	0.4594	0.4592	0.4598	0.5229
aDLminpA	0.4139	**0.4082**	0.4111	0.4102	**0.4336**	0.4342	0.4338	0.4341	0.4601
aCOminpA	0.3091	0.3097	**0.3090**	0.3107	**0.3619**	0.3627	0.3621	0.3625	0.4001
aFLminpA	0.4358	**0.4267**	0.4295	0.4308	**0.4856**	0.4862	0.4857	0.4867	0.5043
aUAminpA	0.2905	0.2869	0.2880	**0.2863**	**0.3543**	0.3544	0.3544	0.3547	0.4051
atp7d	0.5164	0.5169	0.5178	**0.5146**	**0.6404**	0.6414	0.6410	0.6417	0.5794
ALLminpA	**0.5864**	0.5942	0.5914	0.5969	0.7735	**0.7693**	0.7720	0.7694	0.6408
ALLminp0	0.6199	0.6151	0.6201	**0.6072**	0.6906	0.6920	0.6917	**0.6904**	0.6457
aDLminpA	0.5130	0.5036	0.5059	**0.4996**	**0.5833**	0.5835	0.5837	0.5849	0.5387
aCOminpA	0.3931	**0.3869**	0.3936	0.3880	**0.4988**	0.5029	0.5008	0.5034	0.4818
aFLminpA	**0.6108**	0.6246	0.6185	0.6164	**0.7953**	0.7966	0.7955	0.7981	0.6913
aUAminpA	**0.3751**	0.3771	0.3773	0.3792	**0.5008**	0.5041	0.5022	0.5041	0.4779
oes97	0.5164	0.5135	**0.5133**	0.5147	0.6118	0.6123	**0.6110**	0.6110	0.6080
58 028	0.2737	**0.2663**	0.2685	0.2703	0.4175	**0.4157**	0.4159	0.4157	0.3484
15 014	0.3877	0.3841	0.3856	**0.3814**	0.4837	0.4833	**0.4829**	0.4835	0.5551
32 511	0.7751	0.7722	**0.7686**	0.7781	0.8506	0.8549	**0.8504**	0.8506	0.7830
15 017	0.3306	**0.3283**	0.3291	0.3301	0.4112	**0.4077**	0.4093	**0.4077**	0.4717
98 502	0.6827	0.6816	0.6814	**0.6811**	0.8021	**0.8008**	0.8011	0.8017	0.8001
92 965	0.6193	0.6216	0.6169	**0.6162**	**0.6873**	0.6885	0.6885	0.6890	0.7931
32 314	0.5442	**0.5317**	0.5379	0.5377	0.5993	0.5988	**0.5974**	0.5990	0.6968
13 008	0.3072	**0.3056**	0.3070	0.3058	0.4680	**0.4642**	0.4663	**0.4642**	0.3685
21 114	0.2619	**0.2528**	0.2573	0.2562	0.4160	**0.4123**	0.4131	**0.4123**	0.3068
85 110	0.5896	0.5862	**0.5861**	0.5878	**0.6456**	0.6504	0.6476	**0.6456**	0.7176
27 311	0.6335	0.6312	**0.6281**	0.6292	**0.7113**	0.7132	0.7118	**0.7113**	0.7944
98 902	**0.4620**	0.4625	0.4624	0.4642	0.5414	0.5421	**0.5411**	0.5414	0.5276
65 032	0.5475	0.5465	0.5456	**0.5436**	0.6295	0.6305	**0.6291**	0.6296	**0.5320**
92 998	0.7167	0.7214	**0.7145**	0.7230	0.7643	0.7665	**0.7627**	0.7643	0.8240
27 108	0.5782	**0.5733**	0.5750	0.5798	0.6837	0.6860	**0.6833**	0.6836	0.6033
53 905	0.5535	0.5510	**0.5489**	0.5504	0.6770	0.6824	**0.6754**	0.6770	0.6060
oes10	0.4070	0.4081	**0.4070**	0.4079	0.5464	0.5456	**0.5451**	0.5456	0.4914
513 021	0.3843	**0.3816**	0.3871	0.3887	0.5493	0.5495	**0.5486**	0.5493	0.4382
292 071	0.3894	0.3947	0.3894	**0.3889**	0.5443	**0.5409**	0.5415	0.5413	**0.3567**
392 021	0.3742	0.3737	**0.3735**	0.3743	0.5582	0.5590	**0.5578**	0.5591	0.4463
151 131	0.4804	0.4816	0.4788	**0.4743**	0.6073	0.6083	**0.6073**	0.6073	0.5373
151 141	0.3804	0.3780	0.3805	**0.3768**	0.6094	0.6090	**0.6080**	0.6094	0.5365
291 069	0.6260	**0.6145**	0.6177	0.6255	0.6439	0.6466	**0.6427**	0.6439	0.7180
119 032	0.3100	0.3169	0.3115	**0.3099**	0.5560	**0.5535**	0.5540	0.5544	0.4271
432 011	0.3514	**0.3482**	0.3519	0.3558	0.5072	**0.5053**	0.5055	0.5056	0.4015
419 022	0.5999	0.6065	**0.5998**	0.6093	0.6676	0.6675	**0.6657**	0.6681	0.6982
292 037	0.3548	0.3587	0.3544	**0.3535**	0.4938	**0.4914**	0.4918	0.4915	0.3781
519 061	**0.4298**	0.4368	0.4319	0.4335	0.5470	0.5472	**0.5462**	0.5467	0.5727
291 051	0.2254	0.2273	**0.2253**	0.2263	0.4016	0.3989	0.4004	**0.3989**	0.2799
172 141	0.4856	0.4908	0.4867	**0.4856**	0.5902	0.5901	**0.5889**	0.5902	0.7695
431 011	0.2344	0.2294	**0.2280**	0.2292	0.4494	**0.4475**	0.4481	0.4476	0.2775
291 127	0.4831	**0.4802**	0.4842	0.4854	0.5674	0.5670	**0.5666**	0.5675	0.5265
412 021	**0.4033**	0.4111	0.4106	0.4091	0.4489	**0.4483**	0.4485	0.4486	0.4981
rf1	0.0782	0.0582	0.0731	**0.0561**	0.1215	0.1070	0.1151	**0.1015**	0.1206
CHSI2	0.0150	0.0127	0.0135	**0.0119**	0.0781	0.0740	**0.0740**	0.0742	0.0340
NASI2	0.4841	0.3399	0.4495	**0.3248**	0.2852	0.2370	0.2677	**0.2368**	0.6877
EADM7	0.0182	0.0141	0.0157	**0.0137**	0.0909	0.0847	0.0857	**0.0830**	0.0364
SCLM7	0.0190	0.0153	0.0180	**0.0148**	0.1048	0.0914	0.0988	**0.0894**	0.0500
CLKM7	0.0225	0.0209	0.0220	**0.0209**	0.0910	0.0846	0.0885	**0.0812**	0.0302
VALI2	0.0261	0.0237	0.0254	**0.0236**	0.1526	0.1210	0.1381	**0.0869**	0.0355
NAPM7	0.0242	**0.0237**	0.0241	0.0242	0.0746	**0.0741**	0.0743	0.0741	0.0608
DLDI4	0.0167	0.0151	0.0163	**0.0149**	0.0948	0.0891	0.0933	**0.0865**	0.0298
rf2	0.0847	**0.0784**	0.0852	0.0821	0.1095	**0.1064**	0.1084	0.1067	0.4076
CHSI2	0.0180	0.0157	0.0166	**0.0156**	0.0812	0.0797	**0.0796**	0.0798	0.3285
NASI2	0.5151	**0.4774**	0.5252	0.5070	0.2491	**0.2456**	0.2485	0.2490	0.6641
EADM7	0.0207	0.0179	0.0191	**0.0178**	0.0839	**0.0824**	0.0824	0.0824	0.3319
SCLM7	0.0224	0.0200	0.0216	**0.0198**	0.0893	**0.0877**	0.0884	0.0878	0.3421
CLKM7	0.0249	0.0239	0.0245	**0.0239**	0.0842	**0.0827**	0.0830	0.0827	0.3765
VALI2	0.0288	0.0269	0.0279	**0.0269**	0.0959	0.0888	0.0944	**0.0881**	0.3938
NAPM7	0.0277	**0.0267**	0.0273	0.0269	0.0934	**0.0924**	0.0929	0.0924	0.3909
DLDI4	0.0202	0.0187	0.0194	**0.0186**	0.0990	0.0923	0.0983	**0.0916**	0.4326

(*continued on next page*)

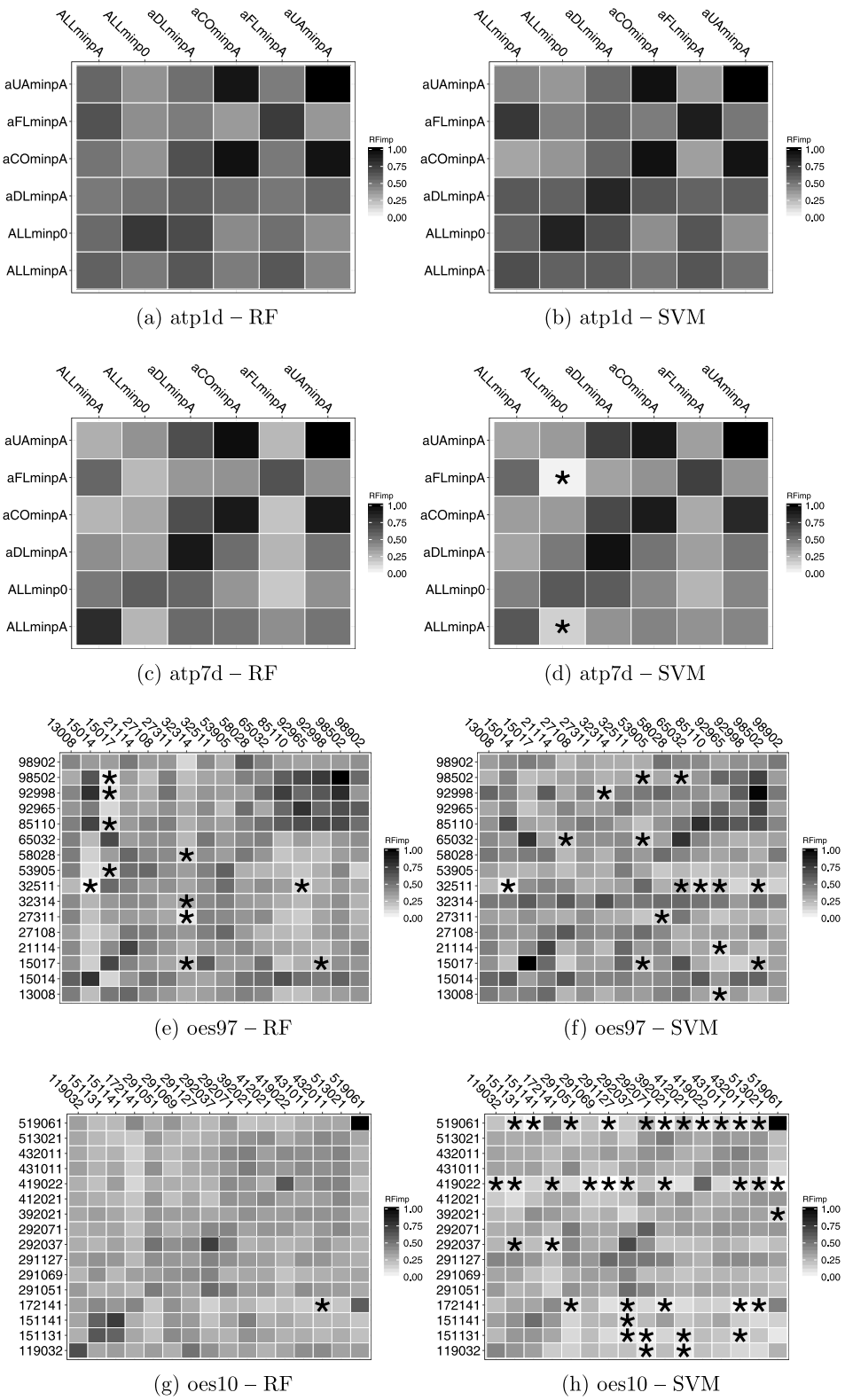
Table B.5 (*continued*).

Dataset / Target	RF				SVM				MORF
	ST	SST	ERC	DSTARS	ST	SST	ERC	DSTARS	
scm1d	0.2871	0.2758	0.2823	**0.2722**	0.3309	0.3232	0.3258	**0.3214**	0.3741
LBL	0.2483	0.2415	0.2462	**0.2396**	0.3054	0.2994	0.3011	**0.2978**	0.3275
MTLp2	0.2691	0.2624	0.2664	**0.2604**	0.3041	0.2981	0.3002	**0.2967**	0.3463
MTLp3	0.2748	0.2646	0.2690	**0.2623**	0.3135	0.3072	0.3085	**0.3059**	0.3430
MTLp4	0.2840	0.2730	0.2797	**0.2691**	0.3186	0.3118	0.3151	**0.3105**	0.3616
MTLp5	0.2901	0.2702	0.2813	**0.2643**	0.3441	0.3307	0.3352	**0.3271**	0.4002
MTLp6	0.3062	0.2861	0.2947	**0.2791**	0.3520	0.3410	0.3434	**0.3366**	0.4197
MTLp7	0.2947	0.2747	0.2868	**0.2677**	0.3348	0.3202	0.3269	**0.3166**	0.4094
MTLp8	0.3023	0.2820	0.2897	**0.2757**	0.3424	0.3275	0.3304	**0.3238**	0.4190
MTLp9	0.2777	0.2676	0.2752	**0.2655**	0.3189	0.3127	0.3153	**0.3112**	0.3565
MTLp10	0.2943	0.2852	0.2911	**0.2823**	0.3334	0.3262	0.3289	**0.3241**	0.3667
MTLp11	0.2921	0.2835	0.2890	**0.2805**	0.3241	0.3185	0.3207	**0.3179**	0.3680
MTLp12	0.3096	0.3018	0.3063	**0.2988**	0.3417	0.3352	0.3368	**0.3346**	0.3840
MTLp13	0.2805	0.2734	0.2786	**0.2714**	0.3466	0.3416	0.3436	**0.3408**	0.3651
MTLp14	0.3083	0.3002	0.3068	**0.2983**	0.3640	0.3590	0.3611	**0.3570**	0.3827
MTLp15	0.2785	0.2707	0.2757	**0.2670**	0.3184	0.3137	0.3152	**0.3136**	0.3630
MTLp16	0.2829	0.2758	0.2808	**0.2732**	0.3330	0.3285	0.3304	**0.3284**	0.3731
scm20d	0.3648	0.3313	0.3347	**0.3129**	0.3972	0.3522	0.3474	**0.3316**	0.5171
LBL	0.3253	0.2991	0.3051	**0.2881**	0.3403	0.3062	0.3088	**0.2954**	0.4680
MTLp2A	0.3334	0.3047	0.3026	**0.2964**	0.3501	0.3166	0.3127	**0.3069**	0.4730
MTLp3A	0.3414	0.3114	0.3129	**0.2976**	0.3634	0.3277	0.3225	**0.3097**	0.4827
MTLp4A	0.3507	0.3217	0.3222	**0.3025**	0.3792	0.3411	0.3356	**0.3272**	0.4967
MTLp5A	0.3748	0.3382	0.3436	**0.3159**	0.4272	0.3709	0.3640	**0.3366**	0.5457
MTLp6A	0.3780	0.3383	0.3364	**0.3125**	0.4230	0.3670	0.3512	**0.3393**	0.5452
MTLp7A	0.3728	0.3315	0.3377	**0.3052**	0.4233	0.3621	0.3631	**0.3289**	0.5384
MTLp8A	0.3796	0.3371	0.3371	**0.3097**	0.4297	0.3617	0.3478	**0.3307**	0.5444
MTLp9A	0.3580	0.3268	0.3301	**0.3138**	0.3895	0.3482	0.3432	**0.3295**	0.5059
MTLp10A	0.3769	0.3457	0.3495	**0.3299**	0.4104	0.3657	0.3625	**0.3482**	0.5201
MTLp11A	0.3724	0.3394	0.3445	**0.3211**	0.4019	0.3571	0.3585	**0.3356**	0.5206
MTLp12A	0.3886	0.3565	0.3630	**0.3442**	0.4262	0.3809	0.3758	**0.3609**	0.5395
MTLp13A	0.3680	0.3362	0.3434	**0.3208**	0.3973	0.3560	0.3512	**0.3364**	0.5204
MTLp14A	0.3868	0.3533	0.3609	**0.3361**	0.4165	0.3796	0.3771	**0.3641**	0.5306
MTLp15A	0.3584	0.3257	0.3336	**0.3046**	0.3874	0.3448	0.3450	**0.3218**	0.5160
MTLp16A	0.3717	0.3350	0.3331	**0.3084**	0.3903	0.3497	0.3400	**0.3343**	0.5260
edm	0.6721	**0.6631**	0.6661	0.6766	0.7699	0.7714	**0.7667**	0.7829	0.6929
DFlow	0.6191	**0.5579**	0.6054	0.5820	**0.7003**	0.7256	0.7038	0.7400	0.6732
DGap	**0.7250**	0.7682	0.7268	0.7711	0.8395	**0.8172**	0.8296	0.8258	**0.7126**
sf1	1.0051	1.1280	1.0161	**1.0047**	0.9390	0.9477	**0.9250**	0.9604	**0.8833**
c.class	1.0842	1.2316	1.1016	**1.0813**	1.0053	1.0019	**0.9932**	1.0277	1.0259
m.class	1.0988	1.2367	1.1145	**1.0986**	1.0838	1.0980	**1.0531**	1.1257	**0.9635**
x.class	0.8322	0.9156	**0.8321**	0.8340	**0.7279**	0.7434	0.7285	**0.7279**	**0.6605**
sf2	**0.8487**	0.9410	0.8617	0.8577	0.7825	0.7787	0.7827	**0.7764**	0.7850
c.class	**0.9881**	1.0944	1.0042	0.9927	1.0067	**0.9924**	1.0098	0.9926	**0.9388**
m.class	**1.1009**	1.3221	1.1583	1.1146	0.9119	0.9150	**0.9103**	0.9124	0.9771
x.class	0.4570	**0.4065**	0.4226	0.4657	0.4291	0.4287	0.4281	**0.4243**	0.4390
jura	0.6061	0.5974	0.5969	**0.5908**	0.6409	0.6413	**0.6391**	0.6410	0.7296
Cd	0.7056	0.7011	0.7018	**0.6981**	0.7103	**0.7052**	0.7059	0.7082	0.7644
Co	0.5272	0.5289	0.5245	**0.5219**	**0.5918**	0.5965	0.5939	0.5925	0.6601
Cu	0.5856	0.5623	0.5645	**0.5524**	0.6208	0.6221	**0.6175**	0.6225	0.7643
wq	0.9066	0.9392	**0.9059**	0.9104	0.9630	**0.9535**	0.9581	0.9576	0.9348
25 400	**0.9154**	0.9638	0.9175	0.9175	0.9892	0.9835	**0.9830**	0.9878	0.9667
29 600	0.9972	1.0631	**0.9915**	0.9953	1.0549	**1.0487**	1.0526	1.0553	1.0006
30 400	0.9672	0.9908	**0.9590**	0.9731	0.9879	0.9852	**0.9842**	0.9946	0.9757
33 400	0.9018	0.9111	**0.8935**	0.9067	0.9394	**0.9067**	0.9345	0.9134	0.9295
17 300	0.8989	0.9270	0.9054	**0.8985**	0.9396	**0.9371**	0.9391	0.9382	0.9463
19 400	**0.8385**	0.8858	0.8394	0.8390	0.8879	0.9024	**0.8863**	0.8975	0.8714
34 500	0.9706	1.0108	**0.9630**	0.9785	1.0076	**0.9940**	1.0009	1.0031	0.9679
38 100	**0.9147**	0.9435	0.9185	0.9258	0.9773	0.9795	**0.9685**	0.9800	0.9294
49 700	**0.7856**	0.8067	0.7934	0.7959	0.8446	**0.8362**	0.8397	0.8428	0.8546
50 390	0.8814	0.9147	0.8836	**0.8795**	0.9811	0.9423	0.9747	**0.9400**	0.9515
55 800	0.9109	0.9466	**0.9078**	0.9112	0.9878	0.9838	**0.9801**	0.9934	0.9339
57 500	0.9166	0.9413	**0.9163**	0.9294	0.9627	**0.9473**	0.9557	0.9545	0.9183
59 300	0.9409	0.9668	**0.9374**	0.9420	1.0299	**1.0057**	1.0209	1.0120	0.9733
37 880	**0.8527**	0.8770	0.8569	0.8537	**0.8924**	0.8964	0.8929	0.8942	0.8686
enb	0.1504	0.1173	0.1304	**0.1149**	0.2499	0.2173	0.2404	**0.1678**	0.2899
Y1	0.1094	0.0519	0.0772	**0.0518**	0.2226	0.1866	0.2120	**0.1312**	0.2798
Y2	0.1914	0.1828	0.1837	**0.1780**	0.2772	0.2479	0.2687	**0.2043**	0.3000
slump	0.8365	0.8324	**0.8222**	0.8338	0.6924	0.6862	**0.6819**	0.6889	0.8419
slump	1.0444	1.0475	**0.9899**	1.0392	0.9295	**0.8797**	0.8868	0.8902	0.9609
flow	**0.8806**	0.9011	0.8809	0.9152	0.8718	0.8761	**0.8612**	0.9006	**0.8275**
cpr_str	0.5846	0.5488	0.5959	**0.5471**	**0.2759**	0.3027	0.2976	**0.2759**	0.7374

(*continued on next page*)

Table B.5 (*continued*).

Dataset / Target	RF				SVM				MORF
	ST	SST	ERC	DSTARS	ST	SST	ERC	DSTARS	
andro	0.7941	0.7349	0.7614	**0.6584**	1.1348	0.9243	1.0089	**0.7899**	0.7704
Target	0.4058	0.3783	0.3897	**0.3652**	0.3603	0.3176	0.3295	**0.2859**	0.5313
Target_2	2.2973	2.1807	2.1899	**1.7572**	4.1015	3.1101	3.5032	**2.3970**	**0.7786**
Target_3	0.4339	**0.3633**	0.4152	0.3634	0.5497	0.4384	0.4979	**0.4137**	0.8027
Target_4	0.4338	0.3713	0.4078	**0.3660**	0.5340	0.4256	0.4721	**0.3962**	0.7828
Target_5	0.5904	0.5592	0.5897	**0.5587**	0.6281	0.6276	0.6222	**0.6206**	0.9027
Target_6	0.6037	0.5566	0.5761	**0.5398**	0.6354	0.6262	0.6284	**0.6262**	0.8245
osales	0.7577	**0.7275**	0.7332	0.7289	1.1726	**1.1685**	1.1702	1.1717	0.9680
M1	0.7019	0.6739	0.6778	**0.6522**	1.6880	**1.6850**	1.6864	1.6880	0.9876
M2	0.7578	0.7206	0.7253	**0.7161**	1.6460	**1.6390**	1.6427	1.6417	0.9709
M3	0.7814	0.7514	0.7537	**0.7390**	0.9752	**0.9717**	0.9734	0.9745	0.9681
M4	0.7510	0.7055	0.7117	**0.6920**	1.1845	**1.1801**	1.1815	1.1837	0.9725
M5	0.7949	**0.7422**	0.7599	0.7753	1.3686	**1.3658**	1.3671	1.3686	0.9790
M6	0.7299	**0.7046**	0.7132	0.7121	0.9761	**0.9721**	0.9740	0.9752	0.9605
M7	0.7682	**0.7403**	0.7449	0.7519	0.9872	**0.9838**	0.9848	0.9866	0.9779
M8	0.7542	**0.7374**	0.7405	0.7387	1.0623	**1.0585**	1.0601	1.0617	0.9569
M9	0.7552	**0.7256**	0.7345	0.7318	1.1696	**1.1662**	1.1680	1.1687	0.9609
M10	0.7570	**0.7339**	0.7411	0.7341	1.1674	**1.1611**	1.1634	1.1658	0.9647
M11	0.7512	0.7328	**0.7300**	0.7355	0.8856	**0.8821**	0.8824	0.8848	0.9556
M12	0.7894	**0.7617**	0.7660	0.7682	0.9605	**0.9566**	0.9579	0.9605	0.9610
scfp	0.9263	0.9379	**0.8778**	0.9017	0.8242	0.8256	**0.8151**	0.8251	0.8620
views	0.8228	**0.7799**	0.7817	0.7914	0.7596	0.7377	0.7462	**0.7258**	0.8262
votes	0.7335	0.7501	**0.7214**	0.7571	0.7304	0.7224	0.7233	**0.7185**	0.7276
comments	1.2226	1.2836	**1.1304**	1.1565	0.9826	1.0167	**0.9758**	1.0309	1.0322



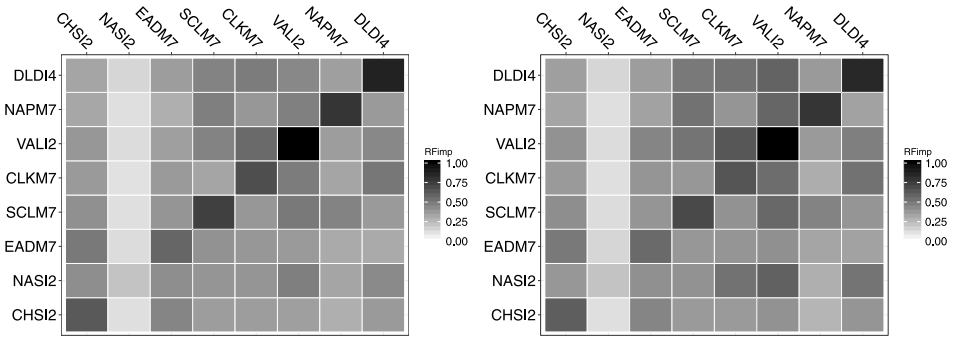
(a) atp1d – RF (b) atp1d – SVM

(c) atp7d – RF (d) atp7d – SVM

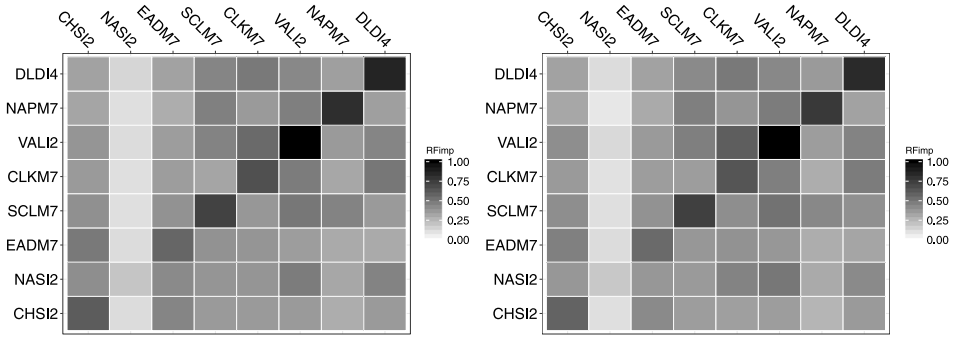
(e) oes97 – RF (f) oes97 – SVM

(g) oes10 – RF (h) oes10 – SVM

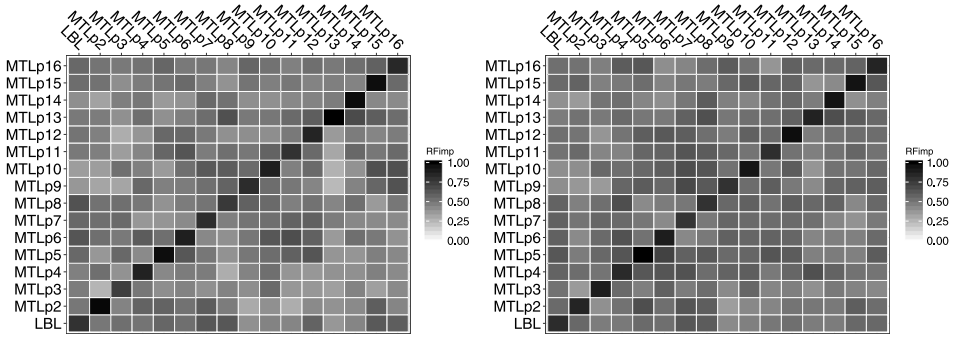
Fig. B.8. Observed values of `RFimp` considering all datasets, regression techniques and the iterations of the 10-fold cross-validation employed to compare the MTR methods. Rows represent the targets and columns the normalized `RFimp` measured for them. The targets combinations which had `RFimp` ≤ 0 in at least one fold are marked with an "*".



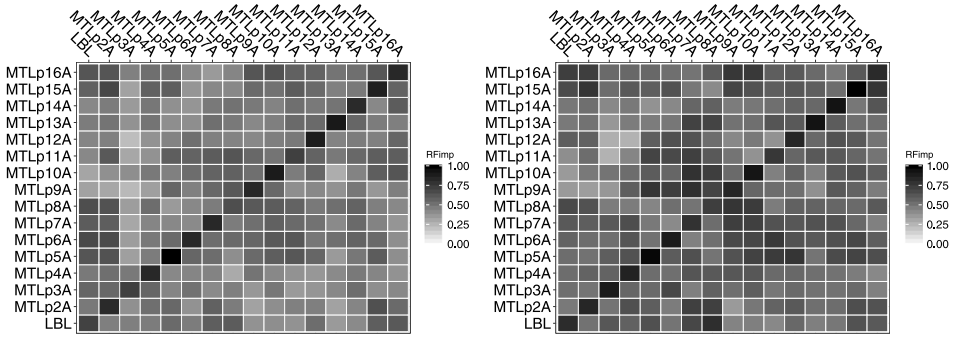
(i) rf1 – RF (j) rf1 – SVM



(k) rf2 – RF (l) rf2 – SVM

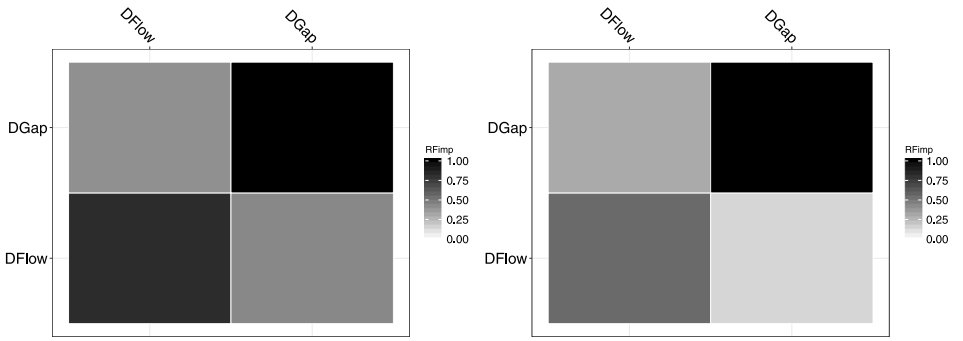


(m) scm1d – RF (n) scm1d – SVM

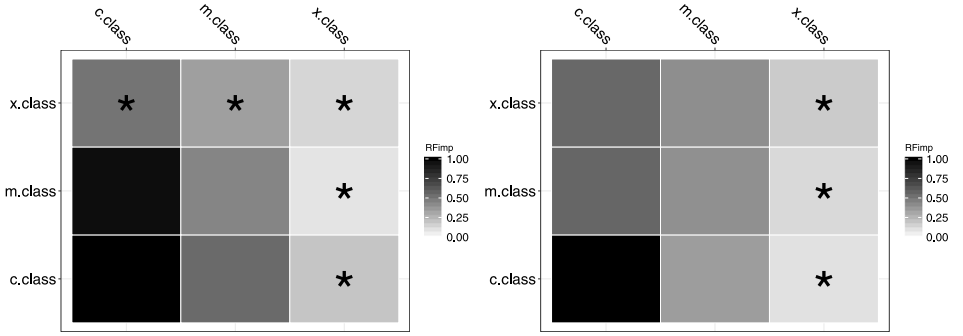


(o) scm20d – RF (p) scm20d – SVM

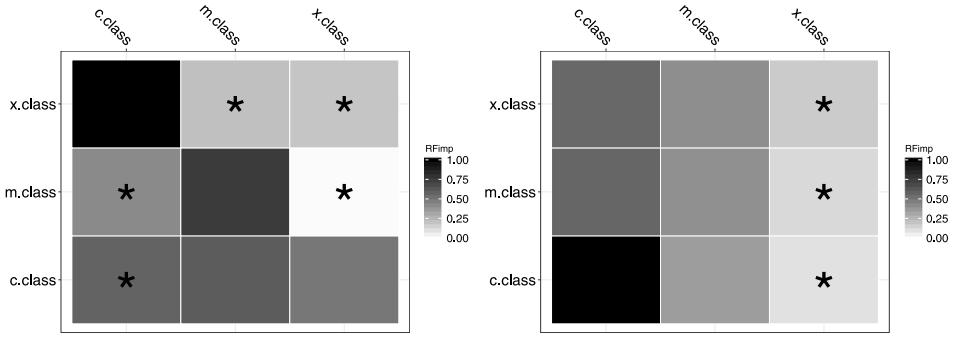
Fig. B.8. (*continued*).



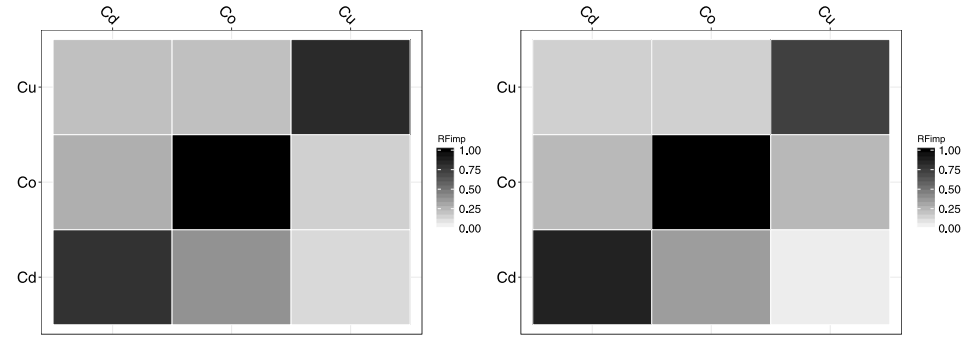
(q) edm – RF (r) edm – SVM



(s) sf1 – RF (t) sf1 – SVM

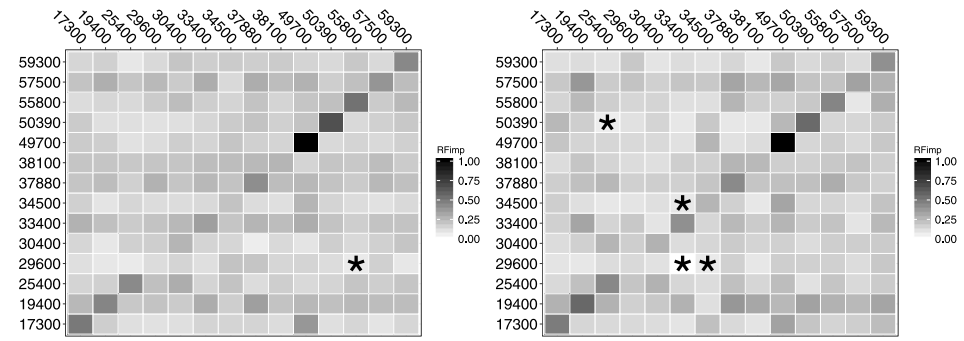


(u) sf2 – RF (v) sf2 – SVM

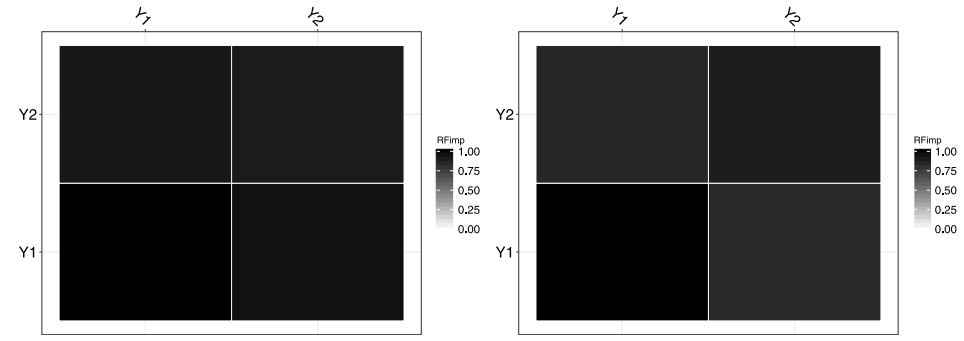


(w) jura – RF (x) jura – SVM

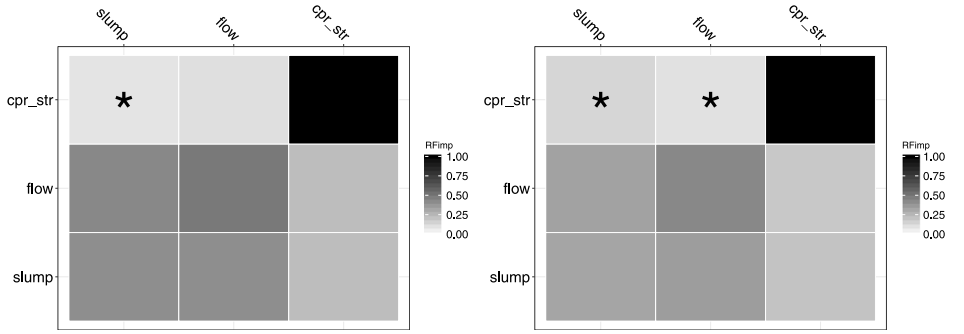
Fig. B.8. (*continued*).



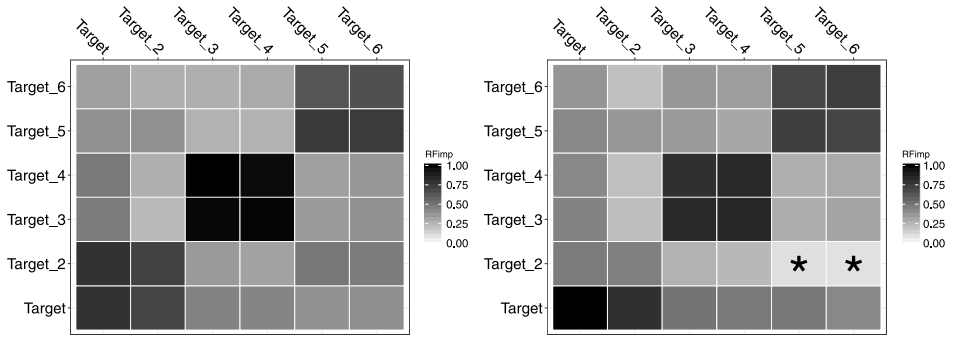
(y) wq – RF (z) wq – SVM



(aa) enb – RF (ab) enb – SVM

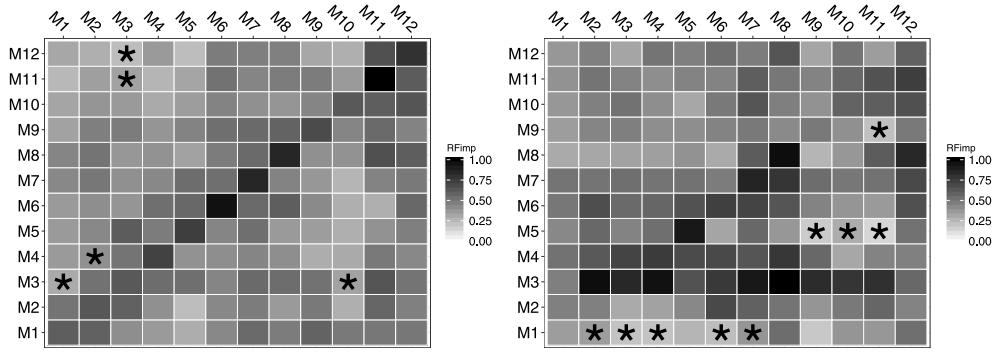


(ac) slump – RF (ad) slump – SVM

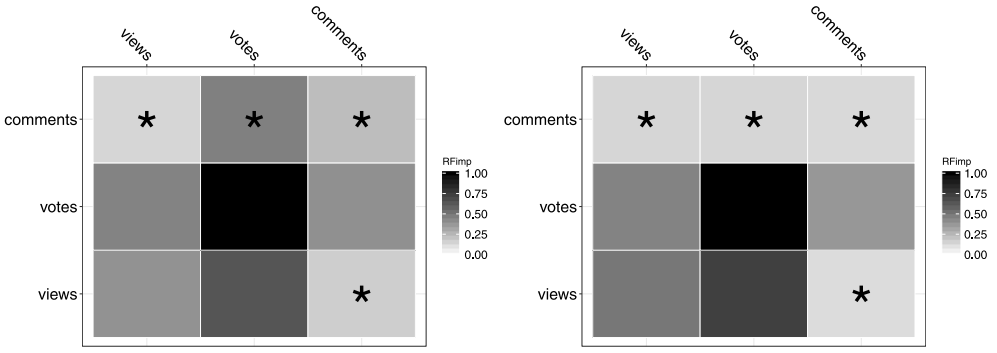


(ae) andro – RF (af) andro – SVM

Fig. B.8. (*continued*).



(ag) osales – RF (ah) osales – SVM



(ai) scfp – RF (aj) scfp – SVM

Fig. B.8. (*continued*).

Table B.6
Number of regressor layers and discontinuities for DSTARS considering all targets, evaluated datasets, and the hyperparameter set $(\phi, \varepsilon) = (0.4, 10^{-4})$.

Dataset Target	Layers		Discontinuities	
	RF	SVM	RF	SVM
atp1d	2.9	1.5	0.3	0.0
ALLminpA	2.6	1.3	0.4	0.0
ALLminp0	2.9	1.7	0.2	0.0
aDLminpA	3.4	1.2	0.3	0.0
aCOminpA	3.1	1.4	0.2	0.0
aFLminpA	2.4	1.6	0.1	0.0
aUAminpA	2.9	1.6	0.3	0.0
atp7d	3.0	1.8	0.2	0.0
ALLminpA	3.2	2.1	0.1	0.0
ALLminp0	3.3	1.1	0.4	0.0
aDLminpA	3.2	1.8	0.2	0.0
aCOminpA	2.9	2.0	0.2	0.0
aFLminpA	2.5	1.7	0.1	0.0
aUAminpA	2.6	2.0	0.1	0.0
oes97	2.5	1.5	0.3	0.0
58 028	2.5	2.0	0.3	0.0
15 014	2.4	1.4	0.1	0.0
32 511	2.5	1.0	0.3	0.0
15 017	2.7	2.0	0.6	0.0
98 502	1.8	2.8	0.1	0.0
92 965	2.2	1.5	0.1	0.0
32 314	2.5	1.9	0.2	0.0
13 008	2.8	2.0	0.6	0.0
21 114	2.1	2.0	0.2	0.0
85 110	2.7	1.0	0.4	0.0
27 311	2.9	1.0	0.3	0.0
98 902	2.8	1.0	0.2	0.0
65 032	3.2	1.1	0.3	0.0
92 998	2.3	1.0	0.4	0.0
27 108	1.9	1.2	0.4	0.0
53 905	2.4	1.0	0.4	0.0
oes10	2.6	1.5	0.3	0.0
513 021	2.8	1.0	0.3	0.0
292 071	2.5	2.0	0.4	0.0
392 021	2.6	1.4	0.3	0.0
151 131	2.4	1.0	0.5	0.0
151 141	2.1	1.0	0.1	0.0
291 069	2.7	1.0	0.3	0.0
119 032	2.1	1.6	0.2	0.0
432 011	2.8	1.9	0.4	0.0
419 022	2.1	1.5	0.1	0.0
292 037	2.4	2.0	0.5	0.0
519 061	2.6	1.3	0.2	0.0
291 051	2.5	2.0	0.1	0.0
172 141	2.9	1.0	0.4	0.0
431 011	2.7	1.9	0.5	0.0
291 127	3.5	1.4	0.6	0.0
412 021	2.3	1.5	0.1	0.0
rf1	2.5	4.2	0.0	0.1
CHSI2	3.1	2.8	0.0	0.0
NASI2	2.2	2.1	0.0	0.0
EADM7	3.0	4.4	0.1	0.0
SCLM7	2.8	4.5	0.1	0.0
CLKM7	2.0	5.6	0.0	0.6
VALI2	2.1	7.0	0.0	0.0
NAPM7	2.0	2.0	0.0	0.0
DLDI4	2.4	5.0	0.0	0.0
rf2	2.2	2.2	0.0	0.0
CHSI2	2.6	2.0	0.0	0.0
NASI2	1.8	1.1	0.0	0.0
EADM7	2.4	2.0	0.0	0.0
SCLM7	2.6	2.0	0.1	0.0
CLKM7	2.0	2.0	0.0	0.0
VALI2	2.1	3.0	0.0	0.0
NAPM7	2.4	2.0	0.0	0.0
DLDI4	2.0	3.4	0.0	0.0
scm1d	5.4	3.0	0.7	0.0
LBL	4.2	3.5	0.4	0.0

(*continued on next page*)

Table B.6 (*continued*).

Dataset Target	Layers		Discontinuities	
	RF	SVM	RF	SVM
MTLp2	5.1	3.0	0.8	0.0
MTLp3	4.9	3.1	0.6	0.0
MTLp4	6.0	3.0	1.1	0.0
MTLp5	7.5	3.2	1.2	0.0
MTLp6	6.0	3.8	0.4	0.0
MTLp7	5.7	3.0	0.2	0.0
MTLp8	6.4	3.0	1.1	0.0
MTLp9	4.7	3.0	0.5	0.0
MTLp10	4.7	3.1	0.4	0.0
MTLp11	5.0	2.8	0.7	0.0
MTLp12	5.0	2.7	0.6	0.0
MTLp13	5.1	2.8	0.7	0.0
MTLp14	4.6	3.0	0.6	0.0
MTLp15	5.2	2.6	0.3	0.0
MTLp16	5.6	2.6	1.2	0.0
scm20d	13.5	4.6	2.1	0.0
LBL	14.5	4.8	3.6	0.0
MTLp2A	11.1	4.0	1.9	0.1
MTLp3A	15.3	5.1	3.4	0.0
MTLp4A	14.0	4.7	1.5	0.0
MTLp5A	15.0	5.9	2.0	0.0
MTLp6A	13.0	4.9	1.2	0.0
MTLp7A	14.6	5.2	2.2	0.0
MTLp8A	14.7	4.0	2.8	0.0
MTLp9A	11.2	4.8	1.1	0.0
MTLp10A	13.7	4.2	2.1	0.0
MTLp11A	12.4	4.8	1.9	0.0
MTLp12A	12.3	4.0	2.2	0.0
MTLp13A	13.2	4.8	1.7	0.0
MTLp14A	13.5	4.0	2.7	0.0
MTLp15A	14.0	4.4	2.3	0.0
MTLp16A	14.0	3.6	1.7	0.0
edm	2.2	3.1	0.0	0.0
DFlow	2.8	2.8	0.0	0.0
DGap	1.5	3.4	0.0	0.0
sf1	1.1	1.6	0.0	0.0
c.class	1.0	1.8	0.0	0.0
m.class	1.0	1.9	0.0	0.0
x.class	1.2	1.0	0.0	0.0
sf2	1.6	2.1	0.0	0.0
c.class	1.1	3.0	0.0	0.0
m.class	1.0	1.2	0.0	0.0
x.class	2.6	2.0	0.0	0.0
jura	3.3	2.0	0.1	0.0
Cd	3.2	2.4	0.2	0.0
Co	3.0	1.4	0.0	0.1
Cu	3.7	2.1	0.2	0.0
wq	1.1	2.7	0.0	0.1
25 400	1.0	2.4	0.0	0.0
29 600	1.0	2.3	0.0	0.0
30 400	1.1	2.6	0.0	0.0
33 400	1.2	3.9	0.0	0.0
17 300	1.0	2.1	0.0	0.0
19 400	1.0	1.5	0.0	0.0
34 500	1.2	3.8	0.1	0.0
38 100	1.1	2.1	0.0	0.0
49 700	1.3	2.6	0.0	0.1
50 390	1.0	4.1	0.0	0.5
55 800	1.0	2.4	0.0	0.0
57 500	1.2	3.3	0.0	0.2
59 300	1.1	3.0	0.0	0.1
37 880	1.1	1.5	0.0	0.0
enb	2.6	5.9	0.0	0.0
Y1	3.1	6.2	0.0	0.0
Y2	2.0	5.6	0.0	0.0
slump	3.0	1.6	0.0	0.0
slump	2.5	2.1	0.0	0.0
flow	2.7	1.8	0.1	0.0
cpr_str	3.9	1.0	0.0	0.0
andro	4.0	7.1	0.4	0.1
Target	5.1	7.1	0.6	0.2

(*continued on next page*)

Table B.6 (*continued*).

Dataset Target	Layers		Discontinuities	
	RF	SVM	RF	SVM
Target_2	3.5	8.2	0.3	0.0
Target_3	3.5	7.4	0.1	0.0
Target_4	3.5	7.2	0.2	0.0
Target_5	3.9	6.6	0.2	0.1
Target_6	4.3	5.8	0.7	0.1
osales	4.3	1.5	0.5	0.0
M1	4.9	1.0	0.3	0.0
M2	5.9	1.8	0.9	0.0
M3	3.9	1.8	0.1	0.0
M4	5.0	1.4	0.4	0.0
M5	4.8	1.0	0.4	0.0
M6	3.5	1.7	0.3	0.0
M7	6.0	1.7	1.7	0.0
M8	4.6	1.5	0.7	0.0
M9	4.1	1.7	0.5	0.0
M10	3.0	1.7	0.1	0.0
M11	3.9	1.9	0.5	0.0
M12	2.5	1.0	0.1	0.0
scfp	2.1	3.1	0.0	0.0
views	2.1	3.8	0.0	0.0
votes	2.1	3.1	0.0	0.0
comments	2.1	2.5	0.0	0.0

References

[1] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, CRC Press, 1984.

[2] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

[3] A. Ben-Hur, J. Weston, A user's guide to support vector machines, in: Data Mining Techniques for the Life Sciences, Humana Press, 2010, pp. 223–239.

[4] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 785–794.

[5] D. Kocev, C. Vens, J. Struyf, S. Džeroski, Tree ensembles for predicting structured outputs, Pattern Recognit. 46 (3) (2013) 817–833.

[6] T. Aho, B. Zenko, S. Dzeroski, T. Elomaa, Multi-target regression with rule ensembles, J. Mach. Learn. Res. 13 (2012) 2367–2407.

[7] H. Borchani, G. Varando, C. Bielza, P. Larrañaga, A survey on multi-output regression, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 5 (5) (2015) 216–233.

[8] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, I. Vlahavas, Multi-target regression via input space expansion: treating targets as inputs, Mach. Learn. 104 (1) (2016) 55–98.

[9] E.J. Santana, S.M. Mastelini, S. Barbon Jr., Deep regressor stacking for air ticket prices prediction, in: Brazilian Symposium of Information Systems, SBSI, 2017, pp. 216–233, 2017.

[10] S.M. Mastelini, V.G.T. da Costa, E.J. Santana, F.K. Nakano, R.C. Guido, R. Cerri, S. Barbon, Multi-output tree chaining: An interpretative modelling and lightweight multi-target approach, J. Signal Process. Syst. (2018) 1–25.

[11] E.J. Santana, B.C. Geronimo, S.M. Mastelini, R.H. Carvalho, D.F. Barbin, E.I. Ida, S. Barbon, Predicting poultry meat characteristics using an enhanced multi-target regression method, Biosyst. Eng. 171 (2018) 193–204.

[12] D. Kocev, S. Džeroski, M.D. White, G.R. Newell, P. Griffioen, Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition, Ecol. Model. 220 (8) (2009) 1159–1168.

[13] G. Tsoumakas, E. Spyromitros-Xioufis, A. Vrekou, I. Vlahavas, Multi-target regression via random linear target combinations, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2014, pp. 225–240.

[14] G. Melki, A. Cano, V. Kecman, S. Ventura, Multi-target support vector regression via correlation regressor chains, Inform. Sci. 415 (2017) 53–69.

[15] S.B. Junior, S.M. Mastelini, A.P.A. Barbon, D.F. Barbin, R. Calvini, J.F. Lopes, A. Ulrici, Multi-target prediction of wheat flour quality parameters with near infrared spectroscopy, Inf. Process. Agric. (2019).

[16] S. Masmoudi, H. Elghazel, D. Taieb, O. Yazar, A. Kallel, A machine-learning framework for predicting multiple air pollutants' concentrations via multi-target regression and feature selection, Sci. Total Environ. 715 (2020) 136991.

[17] J.M. Moyano, E.L. Gibaja, S. Ventura, An evolutionary algorithm for optimizing the target ordering in ensemble of regressor chains, in: Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, 2017, pp. 2015–2021.

[18] D. Kocev, C. Vens, J. Struyf, S. Džeroski, Ensembles of multi-objective decision trees, in: European Conference on Machine Learning, Springer, 2007, pp. 624–631.

[19] S.M. Mastelini, E.J. Santana, R. Cerri, S. Barbon, Dstars: A multi-target deep structure for tracking asynchronous regressor stack, in: 2017 Brazilian Conference on Intelligent Systems (BRACIS), IEEE, 2017, pp. 19–24.

[20] G. Liu, Z. Lin, Y. Yu, Multi-output regression on the output manifold, Pattern Recognit. 42 (11) (2009) 2737–2743.

[21] W. Zhang, X. Liu, Y. Ding, D. Shi, Multi-output LS-SVR machine in extended feature space, in: CIMSA 2012-2012 IEEE Int. Conf. Comput. Intell. Meas. Syst. Appl. Proc., 2012, pp. 130–144.

[22] S. Xu, X. An, X. Qiao, L. Zhu, L. Li, Multi-output least-squares support vector regression machines, Pattern Recognit. Lett. 34 (9) (2013) 1078–1084.

[23] T. Xiong, Y. Bao, Z. Hu, Multiple-output support vector regression with a firefly algorithm for interval-valued stock price index forecasting, Knowl.-Based Syst. 55 (2014) 87–100.

[24] M. Breskvar, D. Kocev, S. Džeroski, Ensembles for multi-target regression with random output selections, Mach. Learn. 107 (2018) 1673–1709.

[25] O. Reyes, S. Ventura, Performing multi-target regression via a parameter sharing-based deep network, Int. J. Neural Syst. 29 (9) (2019) 1950014.

[26] J. Wang, Z. Chen, K. Sun, H. Li, X. Deng, Multi-target regression via target specific features, Knowl.-Based Syst. 170 (2019) 70–78.

[27] J. Gama, P. Brazdil, Cascade generalization, Mach. Learn. 41 (3) (2000) 315–343.

[28] H. Blockeel, L.D. Raedt, J. Ramon, Top-down induction of clustering trees, in: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc, 1998, pp. 55–63.

[29] U. Grömping, Variable importance assessment in regression: linear regression versus random forest, Amer. Statist. 63 (4) (2009) 308–319.

[30] E. Hadavandi, J. Shahrabi, S. Shamshirband, A novel boosted-neural network ensemble for modeling multi-target regression problems, Eng. Appl. Artif. Intell. 45 (2015) 204–219.

[31] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.