Information Sciences 584 (2022) 665–684

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Using meta-learning for multi-target regression

Gabriel J. Aguiar [a,*], Everton J. Santana [b], André C.P.F.L. de Carvalho [d], Sylvio Barbon Junior [c]

[a] College of Engineering, Virginia Commonwealth University, Richmond, Virginia, USA
[b] Software Engineering Department, Pontifical Catholic University of Paraná – PUCPR, Londrina, Brazil
[c] Computer Science Department, Londrina State University – UEL, Londrina, Brazil
[d] Institute of Mathematical and Computer Sciences, University of São Paulo – USP, São Carlos, SP 13566-590, Brazil

ARTICLE INFO

Article history:
Received 29 March 2021
Received in revised form 1 November 2021
Accepted 2 November 2021
Available online 14 November 2021

Keywords:
Meta-learning
Multi-output
Machine learning
Regression
Support vector machine
Random forest

ABSTRACT

Choosing the most suitable algorithm to perform a machine learning task for a new problem is a recurrent and complex task. In multi-target regression tasks, when problem transformation methods are applied, this choice is even harder. The reason is the need to simultaneously choose the problem transformation method and the base learning algorithm. This work investigates how to bridge the gap of method/base learner recommendation for problems with multiple outputs. In meta-learning experiments, we use a large number of multi-target regression datasets to investigate whether using meta-learning can provide good recommendations. To do this, we compared the meta-models induced by 3 different ML algorithms, including three variations for each of them, and selected 58 meta-features that we believe are relevant for extracting good dataset descriptions for the meta-learning process. In the experimental results, the meta-models outperformed the baselines (Majority and Random) by recommending the most suitable solution for multi-target regression (for the transformation method and base-learner) with high predictive performance, including real-world applications. The meta-features and the relation between the transformation method and base-learner provided important insights regarding the optimal problem transformation method. Furthermore, when comparing the application of algorithm adaptation and problem transformation methods, our meta-learning proposal was capable of statistically overcoming all competitors, which resulted in a predictive performance using the best choice per problem.

1. Introduction

"Which machine learning (ML) algorithm to choose?" is a recurrent question when deploying a ML-based solution for a new problem [1]. This dilemma is faced even by ML experts, because, according to the 'no free lunch theorem' [2], there is not a method (algorithm) that outperforms the others for every single problem. The traditional approach is to perform a large set of experiments using either previously used or popular ML algorithms. This is a common praxis in different areas, such as sentiment analysis [3] and medical diagnosis citegroup[citekaur2019diagnosis)]. Methodologies that help to answer this question for a new dataset, without the need for massive experimental evaluation, can overcome many restrictions, such

* Corresponding author.
E-mail addresses: aguiargj@vcu.edu (G.J. Aguiar), everton.santana@pucpr.br (E.J. Santana), andre@icmc.usp.br (A.C.P.F.L. de Carvalho), barbon@uel.br (Sylvio Barbon Junior).

https://doi.org/10.1016/j.ins.2021.11.003
0020-0255/

Nomenclature

AA	Algorithm Adaptation
aRRMSE	Average Relative Root Mean Square Error
AutoML	Automated Machine Learning
CD	Critical Difference
COR	Correlation
CV	Cross-Validation
EA	Evolutionary Algorithms
ERC	Ensemble of Regressor Chains
ML	Machine Learning
MLC	Multi-label Classification
MOTC	Multi-output Tree Chaining
MTAS	Multi-target Augmented Stacking
MtL	Meta-learning
MTSG	Multi-target Stacked Generalization
OOB	Out-of-Bag
PT	Problem Transformation
RF	Random Forest
SMO	Smoothness
SST	Stacked Single-target
ST	Single-target
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting

as processing cost and shortage of ML experts [4]. Besides, recommending an appropriate model is an important part of automated ML, also called Automated Machine Learning AutoML, which intends to automate the whole pipeline of end-to-end ML-based solutions by dealing with new problems more efficiently [5,6].

Among the available methodologies for algorithm selection, meta-learning (MtL), which investigates how to learn from previous ML experiences, has been successfully used in many applications. It takes advantage of ML to induce meta-models able to recommend algorithms for a new problem [7–9].

In MtL, usually, a ML algorithm is applied to a meta-dataset, whose predictive features, meta-features, are features extracted from a dataset, describing its main aspects, and the target, called meta-label, can be the predictive performance obtained by one or more ML algorithms when they are applied to the dataset. Other meta-labels used are a ranking of the ML algorithms according to their predictive performance, starting from the top with the best algorithm and the identification of the best algorithm. Thus, each example in the meta-dataset is associated with a conventional dataset. When an ML algorithm is applied to the meta-dataset, it induces a meta-model, which can be used later to recommend the best ML algorithm(s) to be applied to a new dataset.

Regarding predictive tasks, MtL has been used to recommend techniques for single-label classification and regression tasks, and are applied to different domains, such as image recognition [8,10] and text mining [11]. Although the use of MtL has also been investigated for multi-label classification tasks [12,13], the authors found no work encompassing the whole pipeline for a growing relevant task: Multi-target Regression MTR, a regression task whose aim is to predict an array with N continuous values, where $N > 1$. Previous related work showed that MtL is a promising solution [14] for recommending the best MTR problem transformation method, but this research was limited to a fixed base-learner. However, problem transformation methods pose an additional challenge: "Which base-learner to use as base regressor?". These two challenges reflect the complexity of the recommendation of efficient MTR solutions.

MTR methods are able to improve the predictive performance of models that need to predict more than one continuous output value at the same time [15–18]. The main MTR approaches can be divided into two major branches: i) Algorithm Adaptation AA [19,20], which modifies traditional ML algorithms - or creates new MTR algorithms - to output multiple targets concomitantly, and ii) Problem Transformation PT [18,21–23], which transforms the MTR dataset into single-target datasets to use traditional ML algorithms. In the latter branch, the chosen ML algorithm is referred to as a base-learner or regressor. From now on, when we refer to MTR methods, we mean MTR PT methods.

In this study, we investigate the use of MtL for recommending MTR methods (ST, Stacked Single Target, ERC, MOTC, MTAS and MTSG) along with their base regressors. The aim is to maximise the predictive performance, extending the proposal of Aguiar et al. [14]. To do this, we created a meta-dataset with 792 different MTR problems, each one represented by 58 meta-features and 6 possible meta-label values (i.e., MTR methods) combined with 2 base regressors: Random Forest RF and Support Vector Machine SVM. Next, we investigated the application of ML algorithms over this meta-dataset to build an MtL recommendation of one out of 6 MTR methods and the most suitable base regressor. In particular, 5 meta-models induced

with Random Forest, XGBoosting, and SVM (linear, radial, and polynomial kernels) were evaluated for accomplishing the recommendation task. Moreover, we added four AA methods (k-Nearest Neighbours, Decision Tree, Extra Tree, and RF) to the previous PT methods, creating an extended recommending system. This extended MtL system brought some insights regarding challenges and open issues when recommending a wide range of MTR branches.

The main contributions of this work include:

– The recommendation of both method and base-learner for a new MTR task. Despite the relevance of MTR methods, only one previous work [14] investigated the use of MtL to recommended these methods, lacking the important step of base-learner recommendation;

– The evaluation of the recommendation procedures in real-world problems; Meta-learners trained on synthetic datasets were successfully applied to real MTR datasets;

– The study of a diverse set of meta-features with multiple biases and their contribution to the recommendation of the optimal MTR solution.

– A study encompassing Algorithm Adaptation and Problem Transformation methods to the recommendation of the optimal MTR solution.This paper is organised as follows: Section 2 provides a brief background on MTR tasks and discusses the use of MtL for multi-target method recommendation, Section 3 describes the experimental methodology, Section 4 presents and analyses the results of the study, and Section 5 is dedicated to the main conclusions and future work directions.

2. Related Works

This work relates two major areas: MTR and MtL. The main concepts and literature that are related to our work are presented below, starting with MTR.

2.1. Multi-Target Regression

Any MTR task shares the following characteristics [24]:

– An input set X, which contains f descriptive variables, so that each instance i has an input vector $x_i = \left(x_{i_1}, x_{i_2}, \ldots, x_{i_f}\right)$;

– An output set Y, which contains d continuous targets, so that each instance has an output vector $y_i = \left(y_{i_1}, y_{i_2}, \ldots, y_{i_d}\right)$;

– A set of instances E with N pairs of x_i and $y_i, 1 \leqslant i \leqslant N$.For a given quality criterion q, the aim of MTR methods is to find a function h, or a set of functions, that maximises q.

The simplest approach for solving MTR problems is the Single-target ST approach, which consists of treating each output as an independent problem. This results in d individual models, each model representing a function that maps X to one of the d targets. Thus, h_1 is produced considering X and y_1, h_2 is produced considering X and y_2, and so forth, until the model for the last target is produced. However, the approach used by ST to address the problem can be considered naive, since it does not benefit from correlations between Y for improving regression performance.

MTR methods intend to explore the relationships among the targets to better represent the main aspects of the problem, aiming at better performance. As previously mentioned, MTR methods can be divided into two main research areas: Algorithm Adaptation and Problem Transformation. Some examples of the former are Multi-output Learning via Spectral Filtering [25], Fitted Rule Ensembles [16], Multi-output Contour Regression [26], Multi-output Parameter-insensitive Twin Support Vector Regression [27], Ensemble of Randomly Chained Support Vector Regressor [20], Multi-objective Regression Trees [28,29], Ensembles of Multi-objective Decision Trees [19], Multi-target Regression Trees [15] and Deep Multi-target Regression [30]. On the other hand, some examples of Problem Transformation are Stacked Single-target SST [18], Multi-target Augmented Stacking MTAS [21], Multi-target Stacked Generalization MTSG [31], Ensemble of Regressor Chains ERC [18], Multi-output Tree Chaining MOTC [32], Random Linear Target Combinations [33], Deep Regressor Stacking [34] and Deep Structure for Tracking Asynchronous Regressor Stacking [23]. By complementing the method categorisation proposed in [24], we present some examples of methods related to both branches in Fig. 1.

Since we want to explore how the chosen algorithm affects MTR recommendation and *vice versa, exploring AA methods would be out of the scope of this study. However, for completeness, we use a straightforward comparison using AA in Section 4.6 to provide an overview of meta-recommendation achievements when considering both AA and PT.*

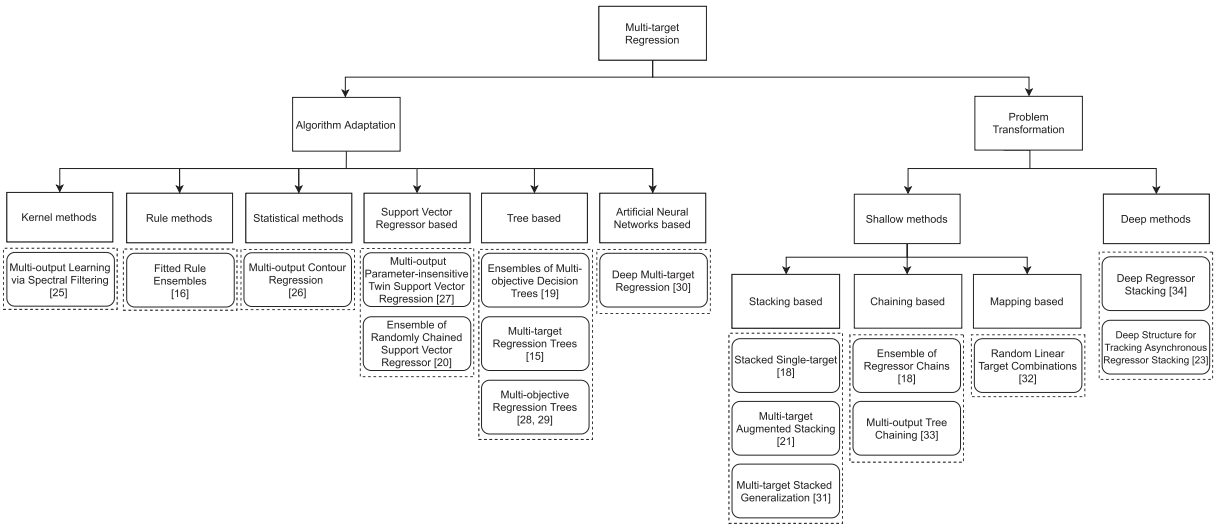


Fig. 1. MTR proposed taxonomy.

Among the problem transformation methods found in the literature, we selected those which followed stacking and chaining strategies and built upon a shallow structure, which reflects in lower computational complexity. In this set of methods, SST and ERC [18] methods are a landmark. In recent years, MOTC [32], MTAS [21] and MTSG [31] were proposed and good predictive results have been obtained.

SST produces d individual models, each one corresponding to an individual target, as in ST. Then, these models are used along with X to obtain predictions for the targets $Y' = \{y'_1, y'_2, \ldots y'_d\}$. These predictions are associated to the original input set, forming $X' = X|Y'$. Afterwards, d new models are trained considering X' and each target. ERC considers different target orders for building these models. In the original version of regressor chains, sequential models are trained for the targets considering the results from the previous models. For instance, y_2 uses as input $X|y'_1, y_3$ uses as input $X|y'_1|y'_2$. In ERC, different regressor chains with different orders are explored and, in the end, the predictions for each target are averaged.

Considering the most recent strategies, MOTC follows the chaining concept of ERC, but uses heuristics to create a tree that represents the dependency among the targets, instead of creating the chain at random. In the first training phase, one tree is built for each target as the root in a top-down approach. The targets most correlated to the root are placed as the level 1 child nodes. Then, the targets most correlated to the level 1 child nodes generate the level 2 child nodes. This process continues until a stopping criterion is reached. In the second phase, using a bottom-up approach, ST models are induced in the leaf node. After that, the previous level nodes consider the original input and the predictions of their child nodes as training input. This process is repeated until the root node builds a model considering the original input set and the prediction of the level 1 child nodes as input. Using the stacking strategy, MTAS takes advantage of multiple families of regressors. To do this, it produces k models for each target during the first training phase, where k is the number of different learning algorithms used to generate these models. Afterwards, only useful predictions are concatenated with X for the next learning phase. In the second phase, one base-learner is chosen to generate the final model for each target.

MTSG also takes into account information about multiple regressor types, such as MTAS. It creates k models for each target in the first training phase. Next, it generates predictions of the models and verifies which of them are relevant for modelling the original problem. In the second phase, only the selected predictions are considered as the training input (which differs from the other methods, in which the predictions are concatenated with the original X), along with a single learning algorithm, producing the final model.

2.2. Meta-Learning

The algorithm recommendation problem was first discussed in [35]. In this text, the author explored the problem of selecting the most suitable algorithm for a given task from a plethora of options. More formally, given a set of algorithms $\mathcal{A}$, a set of datasets $\mathcal{P}$ composed of instances from a distribution $\mathcal{D}$ and a performance measure $\mathcal{M} : \mathcal{P} \times \mathcal{A} \rightarrow \mathbb{R}$, the algorithm recommendation problem is described as how to find a function able to map $m : \mathcal{P} \rightarrow \mathcal{A}$ maximising the expected performance measure for the problems described in $\mathcal{D}$. Different alternatives have been proposed to find this mapping function, one of them is through Meta-learningMtL [36].

MtL uses ML to recommend the most suitable algorithm for a new dataset by using the knowledge acquired from applying different algorithms to several other datasets. To do this, MtL uses ML to map the characteristics of each dataset, which describes them using the performance of a set of algorithms when applied to this dataset. Thus, learning occurs at two levels: *base-level* and *meta-level*. While at the *base-level*, conventional ML algorithms learn a conventional ML task, that is, inducing

models; at the *meta-level* they learn how to learn, inducing meta-models. A more detailed description of MtL can be found in [36].

In the pipeline of MtL experiments, the first step is to select datasets and ML algorithms. Each selected dataset, described by the extraction of a set of meta-feature values from it, becomes a meta-instance in a "*meta-dataset*". In parallel, the selected algorithms are applied to the datasets, and a performance metric (e.g., Accuracy, F-Score, RMSE, *etc.*) is used to define which algorithm presents the best performance. This algorithm will label the meta-instance, becoming a *meta-target*.

MtL has been successfully used in a plethora of works to select the best algorithm, or ranking a set of algorithms [7,37–39].

Concerning multi-output recommendations, there is some related research [12,40] for Multi-label ClassificationMLC, a multi-target research area analogous to MTR for classification tasks [41]. Considering $\mathcal{L}$ the set of binary outputs (labels), in MLC, instances can be simultaneously associated with more than one class label, i.e., an ML algorithm must learn how to associate X to a subset of $\mathcal{L}$. Even though several MLC strategies have been proposed, according to Tsoumakas and Katakis [42], none of them is superior to all others for all MLC datasets.

Chekina et al. [12] proposed MLC method recommendation using MtL. The experiments were carried out on 11 multi-label methods that were grouped in two categories: Single-Classifier Algorithms and Ensemble-Classifier Algorithms. 12 datasets were used to create a recommender of what would be the most suitable MLC method. In general, the results showed that using the recommender model was better than choosing a method randomly.

In another study, Sá et al. [40] used Evolutionary Algorithms (EA) instead of MtL for recommending MLC methods and hyperparameter configurations. 31 MLC methods were evaluated in 3 different datasets showing that, in most of the cases, the predictive performance of the EA selection was either better than or similar to a baseline.

To the best of our knowledge, the previous work [14] was the first work to recommend problem transformation MTR methods using MtL. As an introductory study, RF was fixed as the base-learner for ST, SST, MOTC and ERC and the best recommender achieved 70.83% of accuracy. The results of this work motivated the investigation of different base-learners, with more methods and datasets. Moreover, verifying the adequacy of the recommender to real-world problems is also a missing aspect of [14] and will be addressed in the following sections.

3. Experimental Methodology

The experiments carried out in this work assess the use of MtL to recommend MTR methods and their base-learner. Fig. 2 provides an overview of our experimental methodology.

To induce the recommender (meta-model), a set of MTR methods with different base-learners was applied to several datasets of MTR benchmarking. These datasets were characterised by *meta-feature* values. The evaluation step defines the *meta-targets*, the classes/labels to be predicted by the MtL recommender system. The next subsections describe each one of these steps.

3.1. Meta-Dataset

In the experiments, the meta-dataset comprised 792 benchmarking synthetic datasets[1] (*meta-examples*), generated by following [22]. We used synthetic datasets to overcome the lack of real-world datasets [43] that meet the specific scenario of inter-target dependency and complexity level from input to output relations and cover a different number of input features and targets.

To create a wide possibility of datasets, the parameters of the dataset generator assumed the values presented in Table 1. N, m and d were chosen considering these characteristics in the real-world datasets (that will be shown in Table 3). The g was selected for the case where all relations from input to output are the same or there are 2 different groups and η was included to increase the predictive task and simulate possible noises in real datasets. The numeric targets were built upon math expressions of identity, quadratic, cubic, and quartic functions, and their combinations to cover independent targets and linear/non-linear inter-target correlation.

The inputs of the datasets are generated at random, based on a uniform distribution. The d targets for the g groups are at first generated by random linear coefficients, and therefore each group will share input to output linear mapping properties. The initial d targets can be modified by the mathematical expression to insert inter-target relationships. For instance, if initially Y_1, Y_2 and Y_3 are generated for a synthetic dataset, the final targets might be given by Y_1, Y_1^2 and Y_1^3, or $Y_1, Y_1 + Y_2$ and $Y_1 + Y_3$, among other possibilities. The noise corresponds to random numbers originated from a uniform distribution from -0.01 to 0.01 and is added to the respective instances.

[1] The generated datasets are available for download in: http://www.uel.br/grupo-pesquisa/remid/?page_id=145

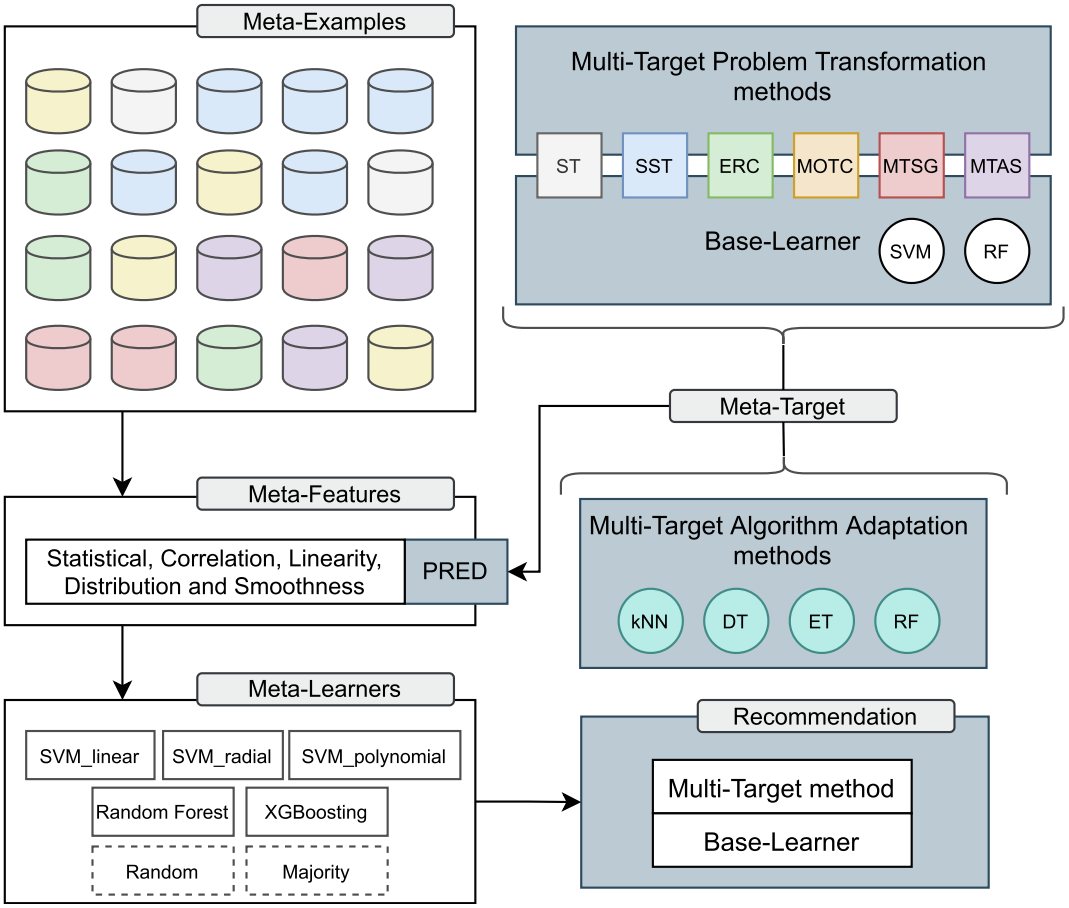


Fig. 2. Experimental Methodology Overview.

Table 1
Hyperparameters used to generate synthetic meta-examples.

Symbol	Hyperparameter	Values
N	Number of instances	$\{500, 1000\}$
m	Number of features	$\{15, 30, 45, 60, 75, 90\}$
d	Number of targets	$\{3, 6\}$
g	Generating groups	$\{1, 2\}$
η	% Instances affected by noise	$\{1, 5, 10\}$

3.2. Meta-Features

Each meta-example is represented by a vector of characteristics (meta-features). In [2], the authors list some requirements to be followed by meta-features: they need to have good discriminative power, their extraction should not have a high computational cost, and the number of meta-features should not be large to avoid overfitting.

In our meta-level experiments, 58 meta-features were extracted from each dataset. They included measures from different categories: statistical information about the dataset (STAT), correlation between attributes and targets (COR), performance metrics related to linear regression (LIN), distribution of the dataset (DIM) and data smoothness (SMO) [44,45].

It is important to mention that some of these meta-features were designed for problems with a single target. Since we are dealing with multi-target problems and we want the meta-features to be valid regardless of the number of targets of the problem, the real value of the meta-features is aggregated for each instance, given that a meta-feature is extracted for each target. To overcome this problem, the meta-features were extracted for each target and the average, standard deviation, maximum, and minimum were added to the set of meta-features [14]. All meta-features, apart from the STAT type, were extracted using the R package `ECoL` [44]. The STAT meta-features, which are straightforward, were extracted from the dataset characteristics.

Table 2
Type, acronym, aggregation function (when applied) and description of meta-features used in the experiments. * indicates the meta-features used in recommendations II and III.

Type	Acronym	Aggregation Functions	Description
STAT	n.samples	-	Number of samples
	n.attributes	-	Number of attributes
	n.targets	-	Number of targets
	target.ratio	-	Ratio between targets and attributes
	pc[1–3]	-	First three components of the Principal Components Analysis
DIM	T2	-	Average number of samples per dimension
	T3	-	Average intrinsic dimensionality per number of examples
	T4	-	Intrinsic dimensionality proportion
COR	cor.targets	{avg,max,min,sd}	Correlation between targets
	C1	{avg,max,min,sd}	Maximum feature correlation to the output
	C2	{avg,max,min,sd}	Average feature correlation to the output
	C3	{avg,max,min,sd}	Individual feature efficiency
	C4	{avg,max,min,sd}	Collective feature efficiency
LIN	regr.L1	{avg,max,min,sd}	Distance of erroneous instances to a linear classifier
	regr.L2	{avg,max,min,sd}	Training error of a linear classifier
	regr.L3	{avg,max,min,sd}	Non-linearity of a linear classifier
SMO	S1	{avg,max,min,sd}	Smoothness of the output distribution
	S2	{avg,max,min,sd}	Smoothness of the input distribution
	S3	{avg,max,min,sd}	Error of a k-nearest neighbour regressor
	S4	{avg,max,min,sd}	Non-linearity of nearest neighbour regressor
PRED*	blr	-	Prediction of Base-Learner
	method	-	Prediction of MTR Method

Table 3
Real-world datasets characteristics and their best method and base-learner considering smallest aRRMSE.

Dataset	# examples	# features	# targets	Best Method	Best base-learner
andro	49	30	6	MTSG	RF
atp1d	337	411	6	MTAS	RF
atp7d	296	411	6	SST	RF
edm	154	16	2	MOTC	RF
enb	768	8	2	MTAS	RF
jura	359	15	3	MTAS	RF
oes10	403	298	16	ERC	RF
oes97	334	263	16	ERC	RF
osales	639	413	12	SST	RF
scfp	1137	23	3	MTAS	RF
scm1d	9803	280	16	MOTC	RF
scm20d	8966	61	16	ERC	RF
sf1	323	10	3	ST	RF
sf2	1066	10	3	ST	RF
slump	103	7	3	MOTC	SVM
wq	1060	16	14	ERC	RF

A complete list of the meta-features used in the experiments is presented in Table 2.

3.3. Recommendation Procedures

When dealing with MTR problem transformation methods, besides choosing a method, it is also necessary to select a base-learner. Each method and base-learner have their own singularities directly affecting the performance of the meta-models. To investigate the combination of these algorithms, we defined 3 recommendation procedures:

– Recommendation I - both algorithm and method recommended independently: The set of 58 meta-features was used to create a model to recommend the method and another model to suggest the base-learner. This recommendation corresponds to the hypothesis that the selection of the method and base-learner can be made separately.

– Recommendation II - first the base-learner, then the method: The set of meta-features was used to first induce a model to recommend the base-learner, and after stacking the recommendations of base-learner as a new feature, to induce the meta-model to recommend the MTR method. This recommendation corresponds to the hypothesis that the characteris-

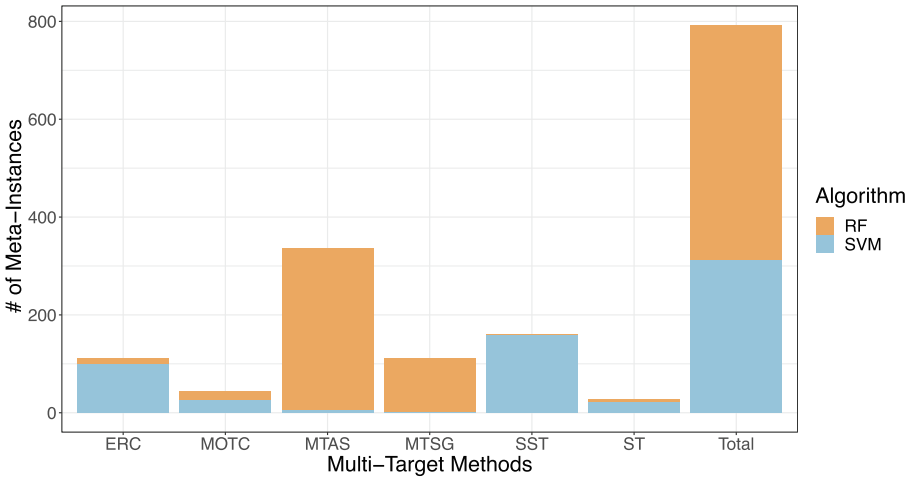


Fig. 3. Class distribution regarding the composition by base-learner.

tics of the base-learner can be used as an advantage to select the most suitable method, taking advantage of its combination.

– Recommendation III - first the method, then the base-learner: Based on the meta-features, we initially recommended the MTR method and after the predicted method was stacked as a new feature, we induced the meta-model to recommend the base-learner. This recommendation corresponds to the hypothesis that the characteristics of the method can be used as an advantage to select the most suitable base-learner, if they are correlated.

3.4. Meta-Target

Along with ST,five MTR methods were explored in this experiment: SST, ERC, MOTC, MTAS and MTSG, described in Section 2. These methods were chosen because they include the most well-known problem transformation MTR methods (SST and ERC) and shallow methods that presented promising results for solving MTR problems (MOTC, MTAS and MTSG). Even though it is the simplest approach, ST was included in the experimental setup because it can perform better than MTR methods in problems with limited inter-target dependency.

These methods were executed for every single meta-example. Their induced models were assessed in terms of lowest Average Relative Root Mean Square ErrorRRMSE, as defined in Eq. 1, where N represents the number of instances, d the number of targets, and $y, \hat{y}$ and $\overline{y}$ represent, respectively, the true, predicted, and mean values of the targets.

$$\text{aRRMSE} = \frac{1}{d}\sum_{t=1}^{d}\sqrt{\frac{\sum_{i=1}^{N}\left(y_t^i - \hat{y}_t^i\right)^2}{\sum_{i=1}^{N}\left(y_t^i - \overline{y}\right)^2}} \tag{1}$$

SVM (with a radial kernel) and RF were used as regressors, performing a k-Fold Cross-ValidationCV resampling validation strategy with $k = 10$. These algorithms were selected because they are used in most MTR studies [34,32,21,22,31,23]. Besides, the method with the lowest RRMSE [24] was chosen as the best multi-target method for every dataset. The experiments were performed using the `mtr-toolkit`[2], implemented in R.

Thus, our meta-dataset comprised two meta-targets: a multi-class meta-label with six different levels indicating the best MTR method or ST regression and a binary meta-label with the best base-learner. The class distribution in the meta-dataset is presented in Fig. 3.

In this distribution, ST delivered the lowest aRRMSE only for 28 meta-instances. This ratifies how the multi-target methods are important for these methods and the relevance of recommending method and base-learner algorithms for a given problem.

3.5. Meta-Learners

Five different meta-models, induced by three ML algorithms, were compared: Random ForestRF [46], Extreme Gradient BoostingXGB [47] and Support Vector MachineSVM [48]. SVM was experimented as meta-learner with linear, radial and

[2] https://github.com/smastelini/mtr-toolkit

polynomial kernels. These algorithms were selected due to their widespread use and capacity of inducing models with high predictive accuracy. The three ML algorithms were implemented using the `R` language[3] and the `mlr` package[4] along with their default hyperparameter values. The k-Fold CV resampling methodology with $k = 10$ folds was also adopted in the meta-level of the experiments to assess the predictive performance of the meta-learners.

3.6. Evaluation Methodology

Six evaluation metrics were used to assess the predictive performance of the induced meta-models: Accuracy, Balanced per class accuracy, Precision, Recall, F-score (f1), and Kappa. They were selected because they are widely used in the literature to evaluate of ML experiments and can also analyse different aspects of the datasets. These measures use TP, TN, FP, FN and *Total*, respectively True Positive, True Negative, False Positive, False Negative and their sum.

Accuracy measures the right predictions of a model. It is obtained by:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

This metric is a suitable measure when the target variable classes in the data are nearly balanced. Otherwise, a variation, balanced per class accuracy, is used.

Precision measures the right predictions in relation to the positive values, i.e.,

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

Recall measures the sensitivity of the classifier to detect positive values.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

F-score gives a balance of precision and recall as it is obtained by the harmonic mean of these metrics, resulting in

$$\text{F} - \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

Kappa measures the agreement between observed and expected accuracy, and is expressed as:

$$\text{Kappa} = \frac{\text{Accuracy} - \text{expAccuracy}}{1 - \text{expAccuracy}} \tag{6}$$

in which

$$\text{expAccuracy} = \frac{(TN + FP) \times (TN + FN) + (TP + FP) \times (TP + FN)}{\text{Total}^2} \tag{7}$$

We used two different baselines from the MtL literature for the comparisons: a model that always recommends the majority class for the whole dataset (`Majority`) and a model that provides random recommendations (`Random`). These baselines are widely used to endorse the need for a recommendation system [2]. Besides, we used an upper-bound as the ground-truth (`Truth`). Thus, we would be able to compare several meta-learners with an ideal one.

After the cross-validation analysis, we also evaluated the recommendation procedures in real-world problems. To do this, we created meta-learners on synthetic datasets and tested their capabilities when recommending the base-learner and method for 16 benchmarking datasets of the MTR literature [18]. They encompass data about water quality (andro), air ticket prices (atp1d and atp7d), electrical machining operation (edm), energy efficiency (enb), soil characteristics (jura), employment surveys (oes10 and oes97), online sales (osales), online engagement (scfp) prices in supply chain (scm1d and scm20d), solar flares (sf1 and sf2), concrete properties (slump) and water quality (wq).

Table 3 contains their major characteristics and shows the method and base-learner which achieved the smallest aRRMSE in each one of them. To determine the two last columns, a 10-fold cross-validation was previously performed at the base-level.

4. Results and Discussion

The main contributions made by this research, supported by the experimental results, will be summarised under four perspectives. First, we present the results considering the predictive performance of the meta-learners (RF, SVM with three kernels and XGboost) and two baselines (`Majority` and `Random`) in the synthetic datasets (Section 4.1). Afterwards, the impact of the recommendation procedures and the need of MtL to recommend the base-learner is discussed in the base-level (Sections 4.2 and 4.3). Afterwards, we evaluated the MtL when recommending the method and base-learner for real datasets

[3] https://www.r-project.org
[4] https://mlr-org.github.io/mlr-tutorial/release/html/index.html

available in the literature (Section 4.4). After that, the importance of each meta-feature is assessed to bring to light the best method for similar problem characteristics (Section 4.5). Finally, we give a broad comparison between problem transformation and algorithm adaptation MTR methods (Section 4.6).

Except for Section 4.5, the analyses were conducted following the three recommendation procedures to verify whether using the method to recommend the base-learner (and vice versa) is more advantageous than recommending both independently. Moreover, we verified how important it is to use one of them (i.e., a method or base-learner) as a hyperparameter to predict the other.

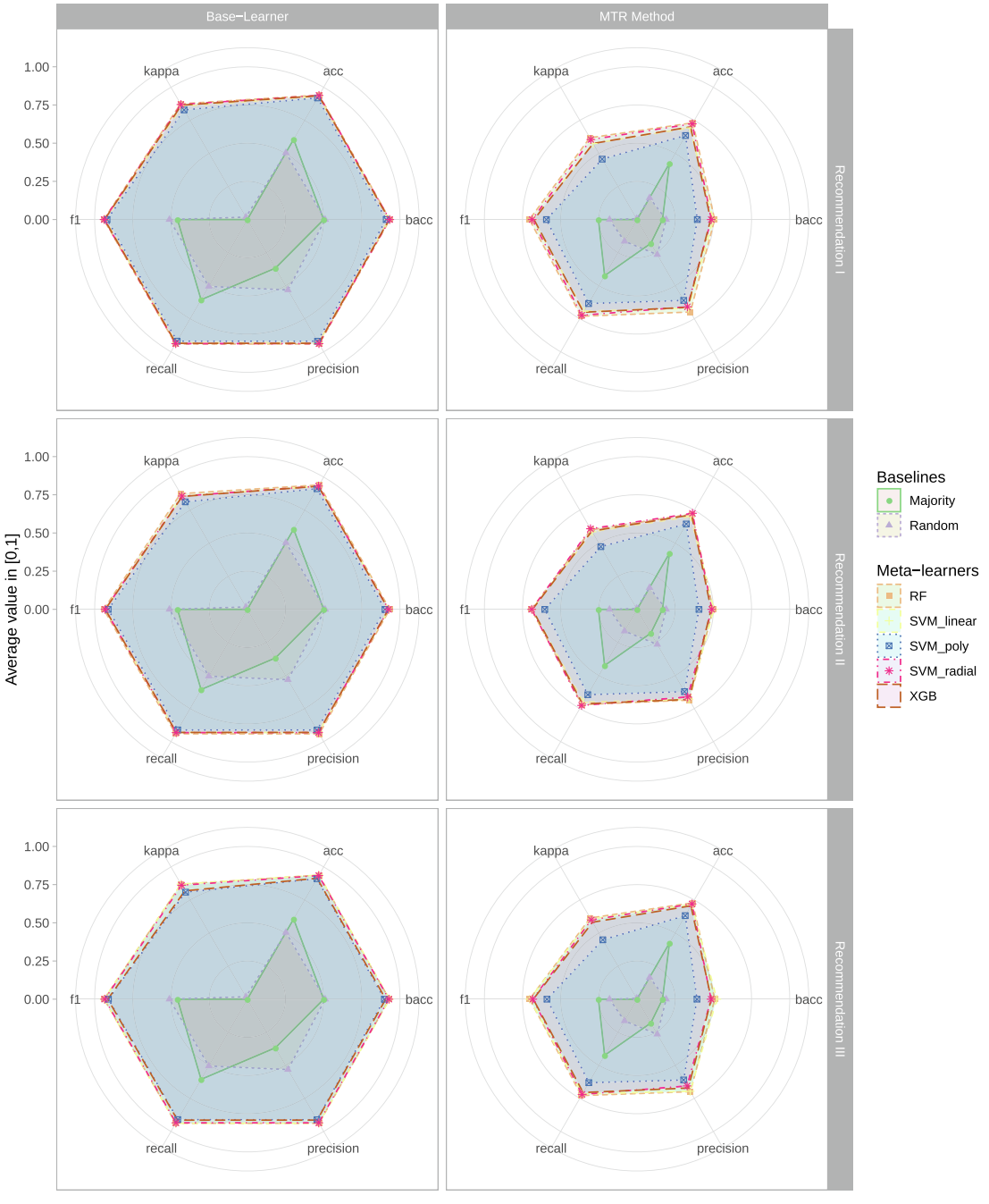


Fig. 4. Performance of the meta-models regarding the base-learner and the method for all recommendations.

4.1. General predictive performance

The predictive performance obtained by the meta-learners and the baselines is presented as radar charts in Fig. 4. Each figure presents the evaluation metrics for the prediction of the MTR methods and base-learners for each recommendation scenario described in Section 3.3. In these figures, each line represents a meta-model and each vertex is related to a different performance measure. The larger the area in the radar chart, the better the meta-model performance is at the meta-level, left side, and at the base-level, right side.

Looking at all radar charts, it can be observed that all meta-learners produced meta-models with superior performance than the `Random` and `Majority` baselines for all metrics and both targets. The superiority of the MtL recommending system over the baselines was statistically significant. To evaluate the statistical significance, we used the Friedman test, with a significance level of $\alpha = 0.05$. The null hypothesis is that the recommendations made by the meta-models and by the baselines are similar. Every time the null hypothesis is rejected, the Nemenyi post hoc test is applied, stating that the performances of the two approaches are significantly different if their corresponding average ranks differ by at least a Critical Difference CD value. When multiple algorithms are compared in this way, a graphic representation can be used to illustrate the results with the CD diagram, as proposed by Demšar [49].

The meta-models (RF, SVM with three kernels and XGB) were compared with `Truth` (expected method), `Majority` (which always predicts the MTAS) and `Random` (the random selection of a method for each dataset), using the aRRMSE as the performance metric. This analysis is shown in Figs. 5 (a), (b) and (c) using the results from the Nemenyi test.

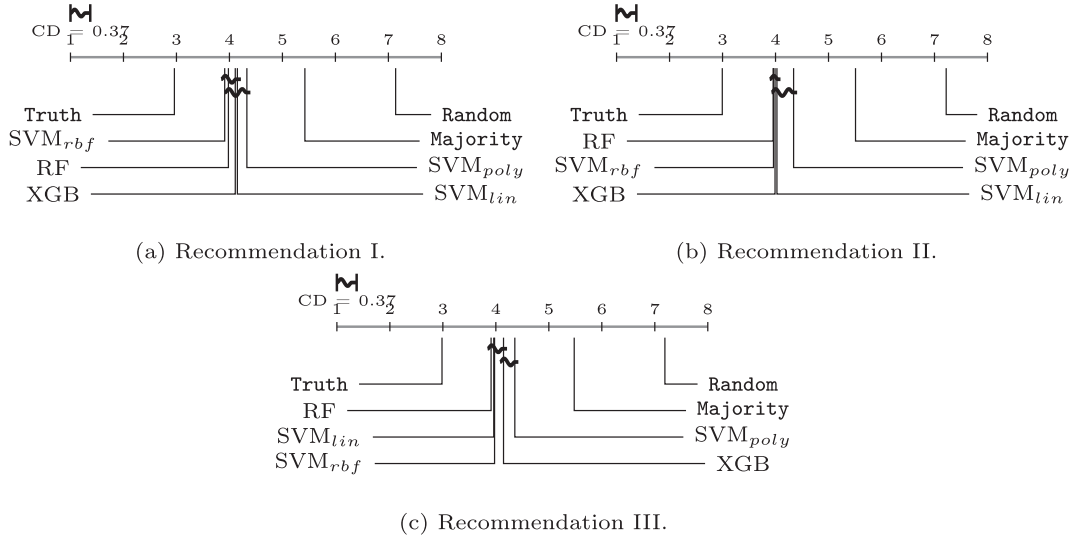
As exposed in the diagrams, no solution was similar to `Truth`. However, the RF, SVM, XGB are connected, which means they were similar and superior to the baselines `Majority` and `Random`. This finding supports the benefit of using MtL recommending system compared to selecting a specific algorithm for every dataset or selecting the algorithm randomly.

4.2. Analysis of recommendation procedures

Regarding the targets, the recommended base-learners presented superior performance in comparison with MTR, since the meta-models obtained high predictive performance from the former recommendation. However, the base-learners recommendation is a binary classification problem, simpler to be solved when compared to MTR, with six different targets.

RF induced a slightly superior meta-model for recommending the MTR method and base-learner in all three recommendation procedures when compared to the other meta-learners (SVM linear, SVM polynomial, SVM radial, and XGBoost).

Regarding Recommendation I, RF obtained 73% of accuracy when predicting the method and 94% when predicting the base-learner. When recommending the base-learner first and the MTR later (Recommendation II), RF obtained 72% of accuracy for the method and 94% for the base-learner. The predictive performance of the RF when using the recommendation in Recommendation III was the same as in Recommendation I. Besides, it can be observed that there was no statistically significant difference between the predictive performance of the recommendation procedures, suggesting that the choice of method and base-learner can be made independently. The tables with all metrics for all targets are presented in the Appendix.



(a) Recommendation I.

(b) Recommendation II.

(c) Recommendation III.

Fig. 5. Comparison of the aRRMSE values obtained by meta-models when recommending MTR methods for procedures I, II and III, according to the Nemenyi test ($\alpha = 0.05$). Groups that are not significantly different ($CD = 0.37$) are connected.

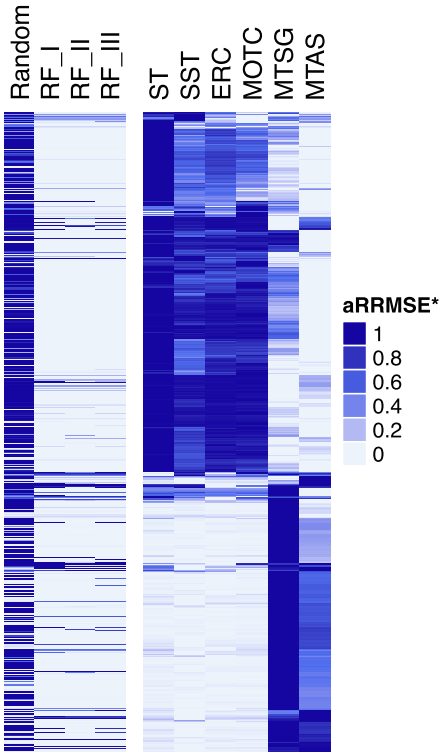


Fig. 6. Error analysis of the methods at base-level and meta-models. The aRRMSE* was obtained using a normalisation procedure for each dataset, in which the minimum aRRMSE for the dataset assumed value 0 and the maximum, 1.

We complementarily built a heatmap (Fig. 6) which shows the error obtained by ten alternatives: the method randomly chosen (Random), RF for Recommendation I (that will be denoted as RF_I), Recommendation II (that will be denoted as RF_II) and Recommendation III (that will be denoted as RF_III), and the error of all methods. The more intense the colour, the greater the error.

Fig. 6 is one more evidence of the 'no free lunch' theorem applied to MTR: none of the methods outperformed all others for all problems. Generally, the datasets in which MTAS and MTSG obtained the worst values were the datasets in which ST, SST, ERC, and MOTC obtained the best predictive performance, and vice versa. This allows us to infer that MTAS and MTSG brought more improvements in datasets where ST had the worst performance. It is a relevant finding, since MTSG and MTAS are the methods with the highest computational complexity among the 6 methods that were compared.

As observed, in most cases, RF_I, RF_II, RF_III recommended methods with low aRRMSE, even though it was not always the method that presented the lowest aRRMSE for the dataset. The colour difference between them and `Random` baseline is very expressive: there is a prevalence of intense colours in `Random`, ratifying the relevance of using the recommendation system over selecting a method randomly. Although MTAS was the majority class of the meta-dataset, when this method presented the highest errors, RF_I, RF_II, and RF_III hardly recommended it. It shows that the meta-model is not highly biased by the difference in class distribution.

Relative to the fact that not always the method that presented the lowest aRRMSE for the dataset was recommended, Table 4 presents a situation where the difference in the aRRMSE value between the best and second-best method was minor. The best method was MTAS and the second best was MTSG. The method recommended by RF in Recommendation I and III was MTSG. Although it is considered a miss (not recommending the best), the difference in aRRMSE between the recommended method and the best method was 0.0050, i.e., less than 1% of error in relation to the best aRRMSE. However, if ST was selected as a method for this dataset, the error would increase to 0.1399, which corresponds to 27.8% of error in relation to the best aRRMSE.

Table 4
aRRMSE of the recommended methods for a given dataset.

	ST	SST	ERC	MTSG	MTAS	MOTC
aRRMSE	0.6430	0.5698	0.6466	0.5056	0.5031	0.6649

4.3. Necessity for base-learner recommendation

In the previous work [14], the use of MtL to predict the MTR method was assessed and showed that using a meta-model was statistically better than selecting the MTR method randomly or using the majority. In continuation, this work evaluates whether it is necessary to induce a meta-model to predict the MTR method and the base-learner. This analysis is presented in Fig. 7.

The results of the `Majority`, `Random` and RF meta-model recommending both method and base-learner meta-targets (indicated as `Majority` + `Majority`, `Random` + `Random` and RF + RF, respectively) are compared with the results RF predicting solely the MTR method combined with `Majority` and `Random` recommending the base-learner (RF + `Majority` and RF + `Random`). The upper-bound `Truth` + `Truth` is used as ground-truth for the scenario where the best method and base-learner are recommended for every problem.

Although none of the combinations were equivalent to the ground-truth, using RF to recommend both the method and the base-learner was ranked as second. Besides, the combinations using RF to predict only the method and not the base-learner were closer to the worst combination in the rank. These results showed that the predictive performance in the base-level is improved when we use a meta-model to predict both the MTR method and its base-learner.

4.4. Recommendation in real-world datasets

Table 5 includes the aRRMSE of the base-learner and method recommended by RF following recommendation procedures I, II, and III in relation to the lowest aRRMSE for each dataset. For comparison purposes, it also shows the relative error corresponding to the selection of the majority or the worst method. The relative aRRMSE of Truth is equal to 1 for all datasets. Besides, we also added the average ranking of each recommendation procedure.

In general, when RF_I, RF_II, and RF_III did not recommend the best option (i.e., presented a value greater than 1), there was a similar observation to Fig. 6: RF_I, RF_II, and RF_III recommended an alternative with aRRMSE very close to the best, reflected by small increases in the average relative aRRMSE (2.07%, 3%, and 2.31%, respectively). For these recommendation procedures, the greatest aRRMSE increase in relation to the best aRRMSE was 13.71% for dataset andro, which presents an extremely small example size (49).

Focusing on RF_I, which delivered the smallest average aRRMSE increase, the combination method/base learner was correctly assigned to 6 datasets. For the other 7 datasets, the error was smaller than 2.5%.

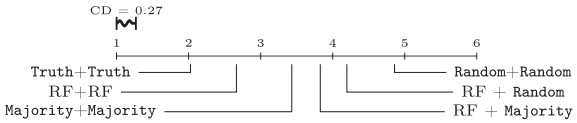


Fig. 7. Comparison of the aRRMSE values obtained by meta-models when recommending MTR methods and its base-leaner according to the Nemenyi test. Groups that are not significantly different ($\alpha = 0.05$ and $CD = 0.27$) are connected.

Table 5
Relative aRRMSE of RF_I, RF_II RF_III, Majority and the worst in relation to the best aRRMSE for each dataset.

Dataset	RF_I	RF_II	RF_III	Majority	Worst
andro	1.1371	1.1371	1.1371	1.1371	2.5378
atp1d	1.0009	1.0009	1.0032	1.0000	1.1450
atp7d	1.0102	1.0102	1.0102	1.0102	1.3449
edm	1.0208	1.0208	1.0208	1.0326	1.3180
enb	1.0000	1.0153	1.0000	1.0000	2.7611
jura	1.0000	1.0000	1.0000	1.0000	1.2318
oes10	1.0000	1.0000	1.0000	1.0071	2.6507
oes97	1.0000	1.0000	1.0000	1.0082	1.8930
osales	1.0067	1.0067	1.0117	1.0067	1.6471
scfp	1.0000	1.0000	1.0684	1.0684	4.0309
scm1d	1.0000	1.0288	1.0288	1.0000	1.4943
scm20d	1.0192	1.0192	1.0192	1.0192	1.8503
sf1	1.0096	1.1152	1.0096	1.1152	1.1840
sf2	1.0712	1.0712	1.0059	1.0712	1.9960
slump	1.0373	1.0373	1.0373	1.0373	1.0844
wq	1.0176	1.0173	1.0176	1.0173	1.1108
Average	**1.0207**	1.0300	1.0231	1.0332	1.8300
Average Ranking	**1.1875**	1.3125	1.375	1.375	2.75

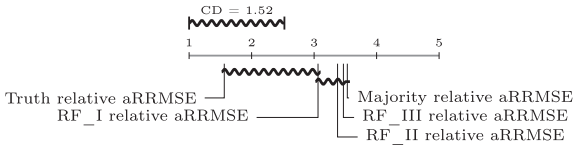


Fig. 8. Comparison of the relative aRRMSE values obtained by meta-models when recommending MTR methods and their base-learner according to the Nemenyi test. Performances that are connected in a $CD = 1.52$ are not significantly different ($\alpha = 0.05$).

It is also notable that selecting the worst method can be very damaging when demanding small aRRMSE values, increasing 83% on average in relation to the best combination. In none of these datasets did the recommender indicate the worst possibility, which is also a positive aspect to show recommendation efficacy.

The Nemenyi post hoc test for these datasets is exhibited in the CD diagram in Fig. 8. Taking into account the relative aRRMSE of Truth, RF_I performance presented no significant difference.

4.5. Meta-features analysis

Taking advantage of the RF induced meta-model, we observed the RF importance given to each meta-feature. This inner RF's metric is calculated by permuting the values of a feature, in this case, a meta-feature, in the Out-of-BagOOB instances and recalculating the OOB Error. Thus, if substituting the values of a feature by random values results in an error increase, this feature was considered important. On the other hand, if the error decreases, the resulting importance is negative, and the feature is considered unimportant. [46].

Fig. 9 shows the meta-feature importance for the meta-dataset in all recommendation procedures. The prediction of the method (in Recommendation III) or the base-learner (in Recommendation II) reached the highest values of feature importance when they were part of the meta-feature space. This is explained by the high correlation between the method and base-learner. As shown in Fig. 3 (Section 3.5), the balancing between base-learner and MTR can be used to distinguish methods.

Additionally, SMO and COR meta-features obtained the highest values of importance for the meta-model used in the recommendation task. Once each MTR method tries to explore the output distribution in different ways and to model the correlation between features and targets, their selection as the best features corroborates our hypothesis. Particularly, the minimum and average smoothness of the output distribution (S1) and error of a k-nearest neighbour regressor (S3) obtained the highest feature importance. This suggests that the distribution of the datasets' targets differs considerably among them and some methods benefit more from higher/lower differences in these distributions than others. For instance, ERC and MOTC deal better with the lowest smoothness values. Table 6 summarises the relationship between the main meta-features and their trend regarding each method.

This study also shows that the number of targets, attributes, and samples had low importance. This might have occurred because these meta-features did not influence the predictive performance, indicating that the MTR methods used in the experiments can deal with different numbers of targets, attributes, and samples in the same way.

4.6. Recommending Problem Transformation and Algorithm Adaptation

The main goal of our research proposal was meta-recommending solutions bounded by Problem Transformation methods using meta-features from the original problems towards identifying the best predictive method coupled with a base-learner. However, it is worth considering the solutions based on Algorithm Adaptation (AA) because they are also able to improve the predictive performance, even requiring structural modifications to fit multiple targets.

We experimented four AA methods: k-Nearest Neighbours (kNN), Decision Tree (DT), Extra Tree (ET), and Random Forest (RF). These algorithms present high predictive performance and low demand for hyperparameter tuning. It is important to mention the low demand for tuning, as it is expected to be a fair comparison to PT methods, which were evaluated without any hyperparameter adjustment. kNN for multi-target is a straightforward extension to multi-label implementation of k-nearest neighbour method [50], adapted for regression tasks. DT and ET for multi-target are based on generalised regression trees for predicting several numeric target attributes simultaneously [28]. Finally, RF for multi-target applies multi-objective decision trees (e.g., DT and ET), which are decision trees that predict multiple target attributes at once [19]. We implemented the methods using Python `scikit-learn`[5] with default hyperparameters.

The four mentioned AA methods were combined with ST and the 5 PT methods coupled with RF and SVM as base regressors, resulting in a total of 16 possibilities of the best method or method and base regressor. We compared the aRRMSE obtained from all 792 datasets to find the lowest error and identify the outperformance.

Fig. 10 presents the best aRRMSE distribution among all possibilities. It can be observed that only 10 among the 16 possibilities are shown, since DT and ET from AA and RF (MOTC), RF (SST), RF (ST), and SVM (MTSG) from PT did not reach the

[5] https://github.com/scikit-learn/scikit-learn

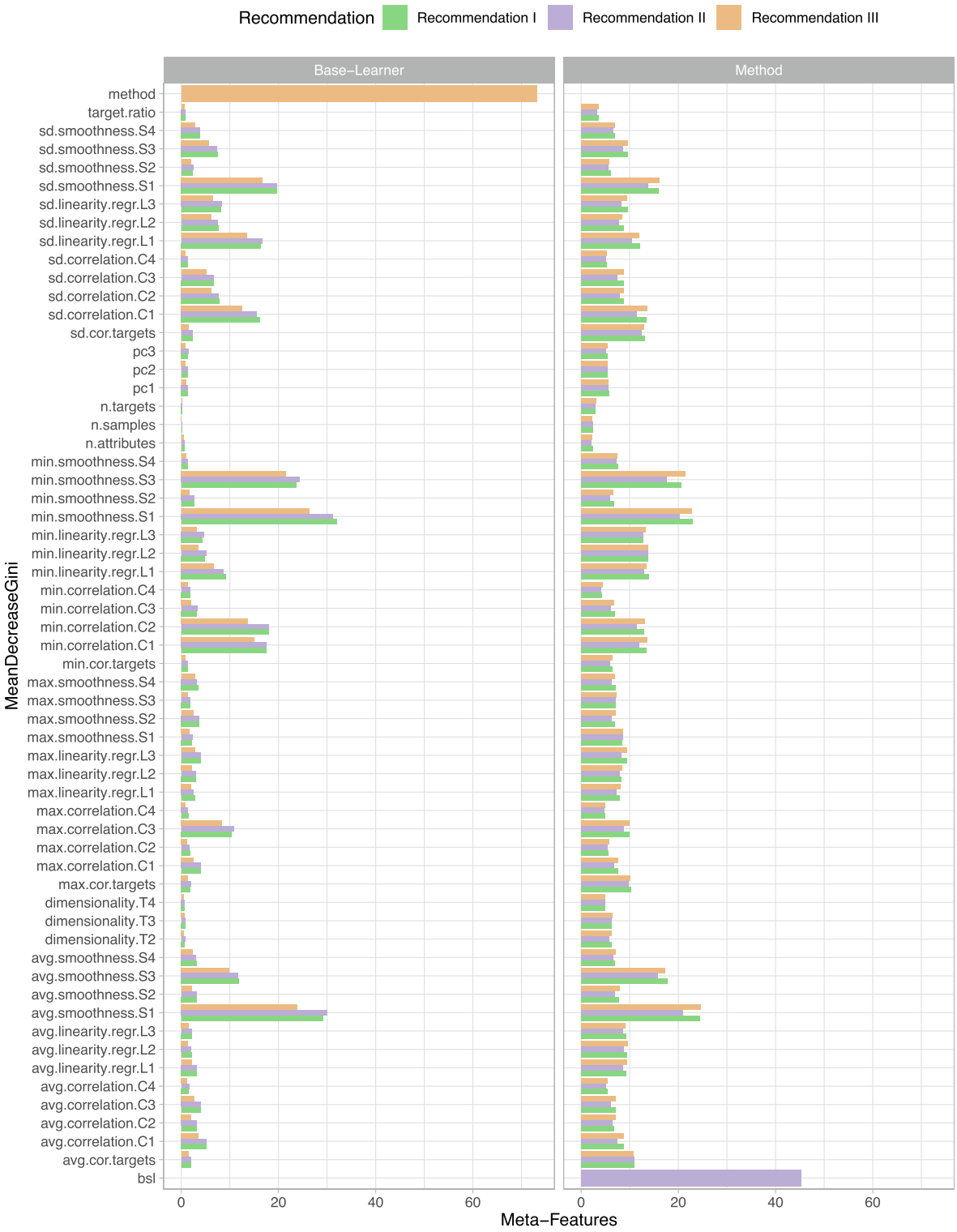


Fig. 9. Average relative importance of the meta-features obtained from the RF importance for both targets when following Recommendation I, II and III. The names of the meta-features in the x-axis follow the acronyms presented in Table 2.

Table 6
Meta-features and methods trends with higher (△) and lower (◆) relationship.

Meta-features meaning	ST	SST	ERC	MTSG	MTAS	MOTC
Miscellaneous output correlation	◆	◆	◆	◆	△	◆
Variation of max feature/output correlation	◆	◆	◆	△	△	◆
Minimal feature/output correlation	△	△	△	△	◆	△
Minimal smoothness of output distribution	◆	△	◆	◆	◆	◆

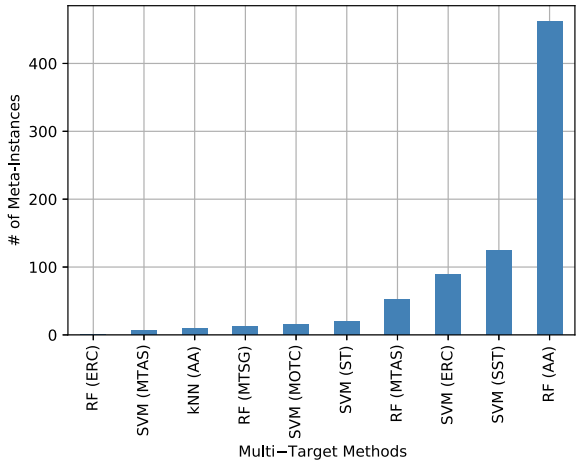


Fig. 10. The frequency of multi-target methods regarding lower aRRMSE among all 16 methods for all 792 datasets (meta-instances). Only 10 methods were chosen since they achieved at least one meta-instance with the lowest error.

lowest error for any meta-instance. RF (AA) was the best-ranked method in 462 meta-instances, achieving the best positions for several methods based on RF as base-learner, i.e., MOTC, SST, and ST. This finding reveals an imbalanced classification problem for meta-recommendation.Fig. 11.

We induced a RF meta-model after removing the minority multi-targets, i.e., RF (ERC), SVM (MTAS), kNN (AA), RF (MTSG). The accuracy obtained for recommending the most suitable among SVM (MOTC), SVM (ST), RF (MTAS), SVM (ERC), SVM (SST) and RF (AA) was 88% (±0.03) and F1-Score 0.69 (±0.03). When comparing the aRRMSE using statistical analysis conducted for the 16 methods, the meta-recommended approach (*meta*) and the best method for each meta-instance (18 populations) with 158 paired meta-instances, we were able to reject the null hypothesis that there are no significant differences between the methods. We used the non-parametric Friedman test to determine if there are differences between the methods and the

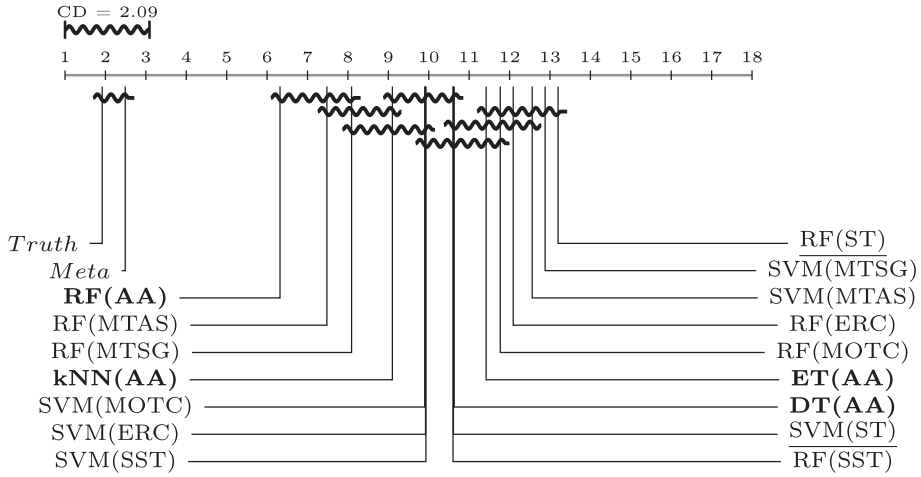


Fig. 11. Nemenyi post hoc test (significance of $\alpha = 0.05$ and critical distance of 2.09) considering the aRRMSE obtained from all Multi-target methods, *best* choice for each meta-instance and the meta-recommendation (*meta*) over small datasets (18 populations with 158 paired samples). The algorithm adaptation and single target methods were highlighted in bold and underlined, respectively.

post hoc Nemenyi test to infer which differences are significant. Based on the post hoc Nemenyi test, we can assume that there are no significant differences within the linked methods. All other differences are significant.

In other words, our meta-model was capable of recommending statically similar results to the best method for each meta-instance (i.e., meta-target problem) on a test set of 158 randomly selected problems. It is notable that the AA Random Forest method and MTAS and MTSG with RF as base-learner were statistically similar. On the other hand, RF with its single-target approach obtained higher aRRMSE, which was similar to several problem transformation and algorithm adaptation methods.

Altogether, what particularly stands out is the 'no free lunch theorem', there is no method that outperforms the others for every single problem. However, we reduced this gap using our proposed meta-learning approach.

5. Conclusion

In this study, we investigated the recommendation of MTR methods and their base-learners using MtL. A meta-dataset, composed of 792 MTR synthetic meta-examples, was used for the induction of meta-models to predict the best tuple of method and base-learner for a given dataset, described by 58 meta-features.

In MtL experiments performed with five meta-learners (RF, SVM with three kernels and XGB), a predictive accuracy around 73% and 94% was obtained for the MTR methods and their base-learners respectively. Most of the time, the best meta-models were induced by RF, regarding the predictive performance at meta-level and at base-level (i.e., regressor). According to the statistical tests, the predictive performance of the meta-models induced by the five MtL alternatives was superior to all baselines. When using the meta-models to recommend the base-learner and method to the 16 real-world datasets, RF_I performance was statistically equivalent to recommending Truth.

The statistical tests also showed that the proposed approach was the best option in three recommendation procedures: i) using only a singular recommendation for method or base-learner ii) selecting the same combination of method and base-learner for all tasks iii) selecting algorithms randomly. The analysis of meta-feature importance revealed that the smoothness of the output distribution and the correlation between targets were the most useful features to predict the performance of an MTR method for a new dataset.

As future works, more problems could be generated to cover a broader space and allow more robustness to the recommender. Another possibility would be a hierarchical recommendation system that could recommend an algorithm adaptation or a problem transformation method, and, in the case of the latter, also the base-learner (base regressor). Finally, the hyper-parameter values of the ML algorithms at the base level could also be meta-recommended, adding a step for an AutoML pipeline.

CRediT authorship contribution statement

Gabriel J. Aguiar: Methodology, Writing - original draft. **Everton J. Santana:** Investigation, Writing - review & editing. **André C.P.F.L. de Carvalho:** Writing - review & editing. **Sylvio Barbon Junior:** Methodology, Writing - review & editing.

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 thank the financial support of Coordination for the Improvement of Higher Education Personnel (CAPES) - Finance Code 001, the Araucária Foundation, the National Council for Scientific and Technological Development (CNPq) of Brazil - Grant of Project 420562/2018–4 and the Scã Paulo Research Foundation (FAPESP) - project 2013/07375–0.

Table 7
Results regarding the recommendation of base-learner following recommendation I.

Meta-Learner	Accuracy	Balanced Acc.	Precision	Recall	F1 Score	Kappa
RF	94%	93%	0.94	0.94	0.94	0.87
SVM_linear	93%	92%	0.93	0.93	0.93	0.85
SVM_poly	91%	90%	0.91	0.91	0.91	0.82
SVM_radial	93%	93%	0.93	0.93	0.93	0.86
XGB	93%	92%	0.93	0.93	0.93	0.86
Random	50%	50%	0.53	0.50	0.51	0.01
Majority	60%	50%	0.36	0.60	0.45	0

Appendix A

The following tables expose detailed results of those expressed in the radar plots in Fig. 4.

Table 8
Results regarding the recommendation of method following recommendation I.

Meta-Learner	Accuracy	Balanced Acc.	Precision	Recall	F1 Score	Kappa
RF	73%	50%	0.69	0.73	0.70	0.61
SVM_linear	69%	49%	0.67	0.69	0.68	0.57
SVM_poly	63%	39%	0.61	0.63	0.59	0.45
SVM_radial	72%	48%	0.65	0.72	0.68	0.60
XGB	70%	48%	0.66	0.70	0.67	0.57
Random	16%	19%	0.26	0.16	0.18	0.01
Majority	42%	16%	0.17	0.42	0.25	0

Table 9
Results regarding the recommendation of base-learner following recommendation II.

Meta-Learner	Accuracy	Balanced Acc.	Precision	Recall	F1 Score	Kappa
RF	94%	93%	0.94	0.94	0.94	0.87
SVM_linear	94%	93%	0.94	0.94	0.94	0.87
SVM_poly	91%	90%	0.91	0.91	0.91	0.81
SVM_radial	93%	93%	0.93	0.93	0.93	0.86
XGB	91%	91%	0.91	0.91	0.91	0.82
Random	51%	51%	0.53	0.51	0.51	0.02
Majority	61%	50%	0.37	0.61	0.46	0.00

Table 10
Results regarding the recommendation of method following recommendation II.

Meta-Learner	Accuracy	Balanced Acc.	Precision	Recall	F1 Score	Kappa
RF	73%	51%	0.70	0.73	0.70	0.61
SVM_linear	72%	52%	0.70	0.72	0.70	0.61
SVM_poly	63%	39%	0.61	0.63	0.59	0.45
SVM_radial	72%	48%	0.66	0.72	0.68	0.60
XGB	71%	49%	0.67	0.71	0.68	0.58
Random	16%	19%	0.26	0.16	0.18	0.01
Majority	42%	17%	0.18	0.42	0.25	0.00

Table 11
Results regarding the recommendation of base-learner following recommendation III.

Meta-Learner	Accuracy	Balanced Acc.	Precision	Recall	F1 Score	Kappa
RF	94%	93%	0.94	0.94	0.94	0.87
SVM_linear	93%	92%	0.93	0.93	0.93	0.85
SVM_poly	91%	90%	0.91	0.91	0.91	0.81
SVM_radial	93%	92%	0.93	0.93	0.93	0.85
XGB	93%	92%	0.93	0.93	0.93	0.85
Random	50%	50%	0.53	0.50	0.51	0.01
Majority	60%	50%	0.36	0.60	0.45	0

Table 12
Results regarding the recommendation of method following recommendation III.

Meta-Learner	Accuracy	Balanced Acc.	Precision	Recall	F1 Score	Kappa
RF	72%	50%	0.69	0.72	0.69	0.60
SVM_linear	70%	50%	0.68	0.70	0.69	0.59
SVM_poly	65%	41%	0.62	0.65	0.60	0.47
SVM_radial	73%	49%	0.66	0.73	0.69	0.61
XGB	71%	48%	0.68	0.71	0.68	0.59
Random	16%	19%	0.26	0.16	0.18	0.01
Majority	42%	17%	0.18	0.42	0.25	0.00

References

[1] S.Y. Yuen, C.K. Chow, X. Zhang, Y. Lou, Which algorithm should i choose: An evolutionary algorithm portfolio approach, Applied Soft Computing 40 (2016) 654–673.
[2] P. Brazdil, C. Giraud-Carrier, C. Soares, R. Vilalta, Metalearning: Applications to Data Mining, 2nd Edition., Springer Verlag, 2009.
[3] S. Kumar, M. Gahalawat, P.P. Roy, D.P. Dogra, B.-G. Kim, Exploring impact of age and gender on sentiment analysis using machine learning, Electronics 9 (2) (2020) 374.
[4] T. Cunha, C. Soares, A.C. de Carvalho, Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering, Information Sciences 423 (2018) 128–144.
[5] J. Vanschoren, Meta-learning, in: Automated Machine Learning, Springer, Cham, 2019, pp. 35–61. .
[6] X. He, K. Zhao, X. Chu, Automl: A survey of the state-of-the-art, Knowledge-Based Systems 212 (2021) 106622.
[7] A.L.D. Rossi, A.C.P. de Leon Ferreira, C. Soares, B.F. De Souza, Metastream, et al, A meta-learning based method for periodic algorithm selection in time-changing data, Neurocomputing 127 (2014) 52–64.
[8] G.J. Aguiar, R.G. Mantovani, S.M. Mastelini, A.C. de Carvalho, G.F. Campos, S.B. Junior, A meta-learning approach for selecting image segmentation algorithm, Pattern Recognition Letters 128 (2019) 480–487.
[9] W. Kong, R. Somani, Z. Song, S. Kakade, S. Oh, Meta-learning for mixed linear regression, in: International Conference on Machine Learning, PMLR, 2020, pp. 5394–5404.
[10] J. Bronskill, J. Gordon, J. Requeima, S. Nowozin, R. Turner, Tasknorm: Rethinking batch normalization for meta-learning, in: International Conference on Machine Learning, PMLR, 2020, pp. 1153–1164.
[11] Z.-Y. Dou, K. Yu, A. Anastasopoulos, Investigating meta-learning algorithms for low-resource natural language understanding tasks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 1192–1197.
[12] L. Chekina, L. Rokach, B. Shapira, Meta-learning for selecting a multi-label classification algorithm, in: 2011 IEEE 11th International Conference on Data Mining Workshops, IEEE, 2011, pp. 220–227.
[13] C. Yu, Y. Ning, Y. Qin, W. Su, X. Zhao, Multi-label fault diagnosis of rolling bearing based on meta-learning, Neural Computing and Applications 33 (10) (2021) 5393–5407.
[14] G.J. Aguiar, E.J. Santana, S.M. Mastelini, R.G. Mantovani, S.B. Júnior, Towards meta-learning for multi-target regression problems, in: 2019 8th Brazilian Conference on Intelligent Systems (BRACIS), IEEE, 2019, pp. 377–382.
[15] 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, Ecological Modelling 220 (8) (2009) 1159–1168.
[16] T. Aho, B. Zenko, S. Dzeroski, T. Elomaa, Multi-Target Regression with Rule Ensembles, J. Mach. Learn. Res. 13 (2012) 2367–2407.
[17] J. Levatić, D. Kocev, M. Ceci, S. Džeroski, Semi-supervised trees for multi-target regression, Information Sciences 450 (2018) 109–127.
[18] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, I. Vlahavas, Multi-target regression via input space expansion: treating targets as inputs, Machine Learning 104 (1) (2016) 55–98.
[19] 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.
[20] G. Melki, A. Cano, V. Kecman, S. Ventura, Multi-target support vector regression via correlation regressor chains, Information Sciences 415 (2017) 53–69.
[21] 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, Biosystems Engineering 171 (2018) 193–204.
[22] S.M. Mastelini, E.J. Santana, V.G.T. da Costa, S. Barbon, Benchmarking multi-target regression methods, in: 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), IEEE, 2018, pp. 396–401.
[23] S.M. Mastelini, E.J. Santana, R. Cerri, S. Barbon Jr, Dstars: A multi-target deep structure for tracking asynchronous regressor stacking, Applied Soft Computing 106215 (2020).
[24] H. Borchani, G. Varando, C. Bielza, P. Larra naga, A survey on multi-output regression, Wiley Interdisciplinary Reviews, Data Mining and Knowledge Discovery 5 (5) (2015) 216–233.
[25] L. Baldassarre, L. Rosasco, A. Barla, A. Verri, Multi-output learning via spectral filtering, Machine learning 87 (3) (2012) 259–301.
[26] Z. Abraham, P.-N. Tan, J. Winkler, S. Zhong, M. Liszewska, et al, Position preserving multi-output prediction, in: Joint European conference on machine learning and knowledge discovery in databases, Springer, 2013, pp. 320–335.
[27] Y. Li, H. Sun, W. Yan, X. Zhang, Multi-output parameter-insensitive kernel twin svr model, Neural Networks 121 (2020) 276–293.
[28] J. Struyf, S. Džeroski, Constraint based induction of multi-objective regression trees, in: International Workshop on Knowledge Discovery in Inductive Databases, Springer, 2005, pp. 222–233. .
[29] H. Blockeel, L. De Raedt, J. Ramon, Top-down induction of clustering trees, in: Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann, 1998, pp. 55–63.
[30] O. Reyes, S. Ventura, Performing multi-target regression via a parameter sharing-based deep network, International Journal of Neural Systems 29 (9) (2019) 1950014.
[31] E.J. Santana, F.R. dos Santos, S.M. Mastelini, F.L. Melquiades, S. Barbon Jr, Improved prediction of soil properties with multi-target stacked generalisation on EDXRF spectra, Chemometrics and Intelligent Laboratory Systems 209 (2021) 104231.
[32] 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, Journal of Signal Processing Systems (2018) 1–25.
[33] 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.
[34] E.J. Santana, S.M. Mastelini, S. Barbon Jr., Deep Regressor Stacking for Air Ticket Prices Prediction, in: XIII Brazilian Symposium on Information Systems: Information Systems for Participatory Digital Governance, Brazilian Computer Society (SBC), 2017, pp. 25–31. .
[35] J.R. Rice, The algorithm selection problem, in: Advances in computers, Vol. 15, Elsevier, 1976, pp. 65–118. .
[36] P. Brazdil, C.G. Carrier, C. Soares, R. Vilalta, Metalearning: Applications to data mining, Springer Science & Business Media, 2008.
[37] P.B. Brazdil, C. Soares, J.P. Da Costa, Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results, Machine Learning 50 (3) (2003) 251–277.
[38] A.L.D. Rossi, C. Soares, B.F. de Souza, A.C.P. de Leon Ferreira, et al, Micro-metastream: Algorithm selection for time-changing data, Information Sciences 565 (2021) 262–277.
[39] I. Gabbay, B. Shapira, L. Rokach, Isolation forests and landmarking-based representations for clustering algorithm recommendation using meta-learning, Information Sciences (2021).
[40] A.G. de Sá, G.L. Pappa, A.A. Freitas, Towards a method for automatically selecting and configuring multi-label classification algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM, 2017, pp. 1125–1132.
[41] F. Serafino, G. Pio, M. Ceci, Ensemble learning for multi-type classification in heterogeneous networks, IEEE Transactions on Knowledge and Data Engineering 30 (12) (2018) 2326–2339.
[42] G. Tsoumakas, I. Katakis, Multi-label classification: An overview, International Journal of Data Warehousing and Mining (IJDWM) 3 (3) (2007) 1–13.
[43] E. Korneva, H. Blockeel, Towards better evaluation of multi-target regression models, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2020, pp. 353–362.

[44] A.C. Lorena, A.I. Maciel, P.B. de Miranda, I.G. Costa, R.B. Prudêncio, Data complety meta-features for regression problems, Machine Learning 107 (1) (2018) 209–246.
[45] R.G. Mantovani, A.L. Rossi, E. Alcobaça, J. Vanschoren, A.C. de Carvalho, A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves svm classifiers, Information Sciences 501 (2019) 193–221.
[46] L. Breiman, Random forests, Machhine, Learning 45 (1) (2001) 5–32.
[47] 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.
[48] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.
[49] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The, Journal of Machine Learning Research 7 (2006) 1–30.
[50] M.-L. Zhang, Z.-H. Zhou, Ml-knn: A lazy learning approach to multi-label learning, Pattern recognition 40 (7) (2007) 2038–2048.