MetaBags: Bagged Meta-Decision Trees for Regression

Jihed Khiari[1]([✉]), Luis Moreira-Matias[1], Ammar Shaker[1], Bernard Ženko[2], and Sašo Džeroski[2]

[1] NEC Laboratories Europe GmbH, Heidelberg, Germany
jihed.khiari@neclab.eu
[2] Jožef Stefan Institute, Ljubljana, Slovenia

Abstract. Methods for learning heterogeneous regression ensembles have not yet been proposed on a large scale. Hitherto, in classical ML literature, stacking, cascading and voting are mostly restricted to classification problems. Regression poses distinct learning challenges that may result in poor performance, even when using well established homogeneous ensemble schemas such as bagging or boosting. In this paper, we introduce `MetaBags`, a novel stacking framework for regression. `MetaBags` learns a set of meta-decision trees designed to select one base model (i.e. *expert*) for each query, and focuses on inductive bias reduction. Finally, these predictions are aggregated into a single prediction through a bagging procedure at meta-level. `MetaBags` is designed to learn a model with a fair bias-variance trade-off, and its improvement over base model performance is correlated with the prediction diversity of different experts on specific input space subregions. An exhaustive empirical testing of the method was performed, evaluating both generalization error and scalability of the approach on open, synthetic and real-world application datasets. The obtained results show that our method outperforms existing state-of-the-art approaches.

Keywords: Stacking · Regression · Meta-learning · Landmarking

1 Introduction

Ensemble refers to a collection of several models (i.e., experts) that are combined to address a given task (e.g. obtain a lower generalization error for supervised learning problems) [24]. Ensemble learning can be divided in three different stages [24]: (i) base model *generation*, where z multiple possible hypotheses $\hat{f}_i(x), i \in \{1..z\}$ to model a given phenomenon $f(x) = p(y|x)$ are generated; (ii) model *pruning*, where $c \leq z$ of those are kept and (iii) model *integration*, where these hypotheses are combined, i.e. $\hat{F}(\hat{f}_1(x), ..., \hat{f}_c(x))$. Naturally, the process may require large computational resources for (i) and/or large and representative training sets to avoid overfitting, since $\hat{F}$ is also learned on the training set, which was already used to train the base models $\hat{f}_i(x)$ in (i). Since the

pioneering Netflix competition in 2007 [1] and the introduction of cloud-based solutions for data storing and/or large-scale computations, ensembles have been increasingly used in industrial applications. For instance, *Kaggle*, the popular competition website, where, during the last five years, 50+% of the winning solutions involved at least one ensemble of multiple models [21].

Ensemble learning builds on the principles of committees, where there is typically never a single expert that outperforms all the others on each and every query. Instead, we may obtain a better overall performance by *combining* answers of multiple experts [28]. Despite the importance of the combining function $\hat{F}$ for the success of the ensemble, most of the recent research on ensemble learning is either focused on (i) model generation and/or (ii) pruning [24].

Model integration approaches are grouped in three clusters [30]: (a) voting (e.g. bagging [4]), (b) cascading [18] and (c) stacking [33]. In voting, the outputs of the ensemble is a (weighted) average of outputs of the base models. Cascading iteratively combines the outputs of the base experts by including them, one at a time, as another feature in the training set. Stacking learns a meta-model that combines the outputs of all the base models. Voting relies on base models to have complementary expertise, which is an assumption that is rarely true in practice (e.g. check Fig. 1(b,c)). On the other hand, cascading is typically too time-consuming to be put in practice, since it involves training of several models in a sequential fashion.

Stacking relies on the power of the meta-learning algorithm to approximate $\hat{F}$. Stacking approaches are of two types: parametric and non-parametric. The first (and most common [21]) assumes a (typically linear) functional form for $\hat{F}$, while its coefficients are either learned or estimated [7]. The second follows a strict meta-learning approach [3], where a meta-model for $\hat{F}$ is learned in a non-parametric fashion by relating the characteristics of problems (i.e. properties of the training data) with the performance of the experts. Notable approaches include instance-based learning [32] and decision trees [30]. However, novel approaches for model integration in ensemble learning are primarily designed for classification and, if at all, adapted later on for regression [24,30,32]. While such adaptation may be trivial in many cases, it is noteworthy that regression poses distinct challenges.

Formally, we formulate a regression problem as the problem of learning a function

$$\hat{f}_\theta : x_i \rightarrow \mathbb{R} \quad \text{such that} \quad \hat{f}(x_i; \theta) \simeq f(x_i) = y_i, \forall x_i \in X, y_i \in Y \qquad (1)$$

where $f(x_i)$ denotes the true unknown function which is generating the samples' target variable values, and $\hat{f}(x_i; \theta) = \hat{y}_i$ denotes an approximation dependent on the feature vector x_i and an unknown (hyper)parameter vector $\theta \in \mathbb{R}^n$. One of the key differences between regression and classification is that for regression the range of f is apriori undefined and potentially infinite. This issue raises practical hazards for applying many of the widely used supervised learning algorithms, since some of them cannot predict outside of the target range of their training set values (e.g. Generalized Additive Models (GAM) [20] or CART [6]).

Another major issue in regression problems are **outliers**. In classification, one can observe either *feature* or *concept* outliers (i.e. outliers in $p(x)$ and $p(y|x)$), while in regression one can also observe *target* outliers (in $p(y)$). Given that the true target domain is unknown, these outliers may be very difficult to handle with common preprocessing techniques (e.g. Tukey's boxplot or one-class SVM [9]). Figure 1 illustrates these issues in practice on a synthetic example with different regression algorithms. Although the idea of training different experts in parallel to subsequently combine them seems theoretically attractive, the above-mentioned issues make it difficult in practice, especially for regression. In this context, stacking is regarded as a complex *art* of finding the right combination of data preprocessing, model generation/pruning/integration and post-processing approaches for a given problem.
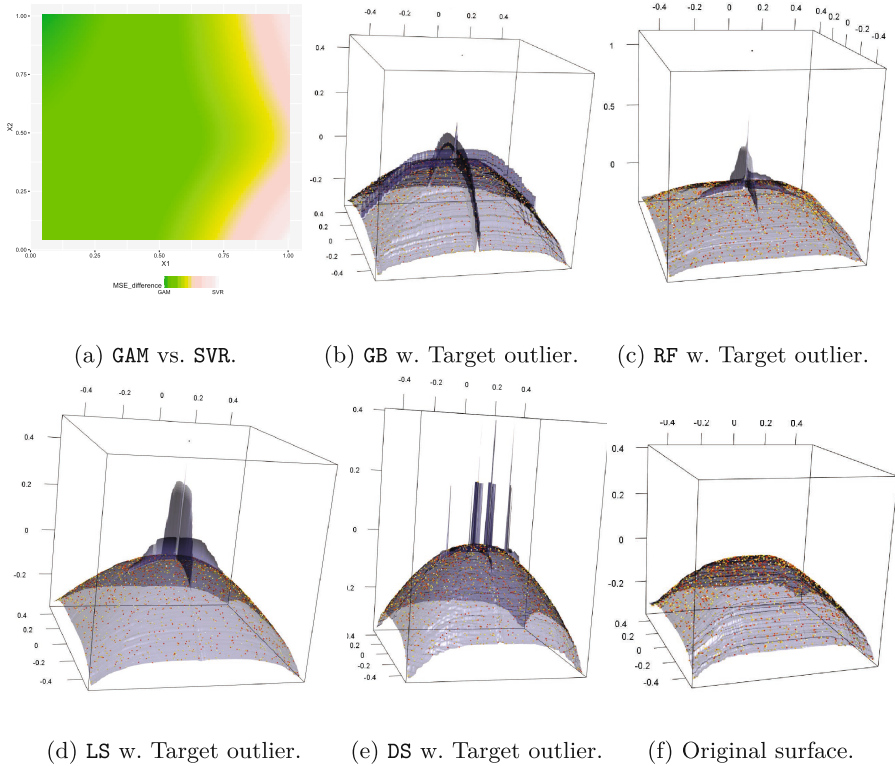
In this paper, we introduce `MetaBags`, a novel stacking framework for regression. `MetaBags` is a powerful meta-learning algorithm that learns a set of meta-decision trees designed to select one expert for each query thus reducing inductive bias. These trees are learned using different types of meta-features specially created for this purpose on data bootstrap samples, whereas the final meta-model output is the average of the outputs of the experts selected by each meta-decision tree for a given query. Our contributions are threefold:

1. A novel meta-learning algorithm to perform non-parametric stacking for regression problems with **minimum user expertise requirements**.
2. An approach for turning the traditional overfitting tendency of stacking into an advantage through the usage of **bagging at the meta-level**.
3. A novel set of **local landmarking meta-features** that characterize the learning process in feature subspaces and enable model integration for regression problems.

In the remainder of this paper, we describe the proposed approach, after discussing related work. We then present an exhaustive experimental evaluation of its efficiency and scalability in practice. This evaluation employs 17 regression datasets (including one real-world application) and compares our approach to existing ones.

2   Related Work

Since its first appearance, meta-learning has been defined in multiple ways that focus on different aspects such as collected experience, domain of application, interaction between learners and the knowledge about the learners [23]. Brazdil et al. [3] define meta-learning as the learning that deals with both types of bias, declarative and procedural. The declarative bias is imposed by the hypothesis space form which a base learner chooses a model, whereas the procedural bias defines how different hypotheses should be preferred. In a recent survey, Lemke et al. [23] characterize meta-learning as the learning that constitutes

(a) `GAM` vs. `SVR`.   (b) `GB` w. Target outlier.   (c) `RF` w. Target outlier.

(d) `LS` w. Target outlier.   (e) `DS` w. Target outlier.   (f) Original surface.

Fig. 1. Illustration of distinctive issues in regression problems on a synthetic example. In all experiments, we generate 1k training examples for the function $y = (x_1^4 + x_2^4)^{\frac{1}{2}}$. In (a), x_1, x_2 training values are sampled from a uniform distribution constrained to $\in [0, 0.8]$, while the testing ones are $\in [0, 1]$. Panel (a) depicts the difference between RMSE between the two tested methods, `GAM` and `SVR`, where the hyperparameters were tuned using random search (60 points) and a 3-fold-CV procedure was used for error estimation. `SVR`'s MSE is significantly larger than `GAM`'s one, and still, there are several regions of the input space where `GAM` is outperformed (in light pink colors). Panels (b,c) depict the regression surface of two models learned using tree-based Gradient Boosting machines (`GB`) and Random Forests (`RF`), respectively, with 100 trees and default hyperparameter settings. To show their sensitivity to *target* outliers, we artificially imputed one extremely high value (in black) in the target of one single example (where the value is already expected to be maximum). In Panels (d, e), we analyze the same effects with two stacking approaches using the models fitted in (a, b, c) as base learners: Linear Stacking (`LS`) in (d) and Dynamic Selection (`DS`) with `kNN` in (e). Please note how deformed the regression surfaces (in gray) are in all settings (b–d). Panel (f) depicts the original surface. Best viewed in color. (Color figure online)

three essential aspects: (i) the adaptation with experience, (ii) the consideration of meta-knowledge of the data set (to be learned from) and (iii) the integration of meta-knowledge from various domains. Under this definition, both ensemble methods bagging [4] and boosting [14] do not qualify as meta-learners, since the base learners in bagging are trained independently of each other, and in boosting, no meta-knowledge from different domains is used when combining decisions from the base learners. Using the same argument, stacking [33] and cascading [18] cannot be definitely considered as meta-learners [23].

Algorithm recommendation, in the context of meta-learning, aims to propose the type of learner that best fits a specific problem. This recommendation can be performed after considering both the learner's performance and the characteristics of the problem [23]. Both aforementioned aspects qualify as meta-features that assist in deciding which learner could perform best on a specific problem. We note three classes of meta-features [3]: (i) meta-features of the dataset describing its statistical properties such as the number of classes and attributes, the ratio of target classes, the correlation between the attributes themselves, and between the attributes and the target concept, (ii) model-based meta-features that can be extracted from models learned on the target dataset, such as the number of support vectors when applying SVM, or the number of rules when learning a system of rules, and (iii) landmarkers, which constitute the generalization performance of diverse set of learners on the target dataset in order to gain insights into which type of learners fits best to which regions/subspaces of the studied problem. Traditionally, landmarkers have been mostly proposed in a classification context [3,27]. A notorious exception is proposed by Feurer et al. [12]. The authors use meta-learning to generate prior knowledge to feed a bayesian optimization procedure in order to find the best sequence of algorithms to address predefined tasks in either classification and regression pipelines. However, the original paper [13] focuses mainly on classification.

The dynamic approach of ensemble integration [24] postpones the integration step till prediction time so that the models used for prediction are chosen dynamically, depending on the query to be classified. Merz [25] applies dynamic selection (`DS`) locally by selecting models that have good performance in the neighborhood of the observed query. This can be seen as an integration approach that considers type-(iii) landmarkers. Tsymbal et al. [32] show how `DS` for random forests decreases the bias while keeping the variance unchanged.

In a classification setting, Todorovski and Džeroski [30] combine a set of base classifiers by using meta-decision trees which in a leaf node give a recommendation of a specific classifier to be used for instances reaching that leaf node. Meta-decision trees (`MDT`) are learned by stacking and use the confidence of the base classifiers as meta-features. These can be viewed as landmarks that characterizes the learner, the data used for learning and the example that needs to be classified. Most of the suggested meta-features `MDT` are applicable to classification problems only.

MetaBags can be seen as a generalization of DS [25,32] that uses meta-features instead. Moreover, we considerably reduce DS runtime complexity (generically, $\mathcal{O}(N)$ in test time, even with state-of-the-art search heuristics [2]), as well as the user-expertise requirements to develop a proper metric for each problem. Finally, the novel type of local landmarking meta-features characterize the local learning process - aiming to avoid overfitting.

3   Methodology

This Section introduces MetaBags and its three basic components: (1) First, we describe a novel algorithm to learn a decision tree that picks one expert among all available ones to address a particular query in a supervised learning context; (2) then, we depict the integration of base models at the meta-level with bagging to form the final predictor $\hat{F}$; (3) Finally, the meta-features used by MetaBags are detailed. An overview of the method is presented in Fig. 2.

3.1   Meta-Decision Tree for Regression

Problem Setting. In traditional stacking, $\hat{F}$ just depends on the base models $\hat{f}_i$. In practice, as stronger models may outperform weaker ones (c.f. Fig. 1(a)), they get assigned very high coefficients (assuming we combine base models with a linear meta-model). In turn, weaker models may obtain near-zero coefficients. This can easily lead to over-fitting if a careful model generation does not take place beforehand (c.f. Fig. 1(d,e)). However, even a model that is weak in the whole input space may be strong in some subregion. In our approach we rely on classic tree-based isothetic boundaries to identify *contexts* (e.g. subregions of the input space) where some models may outperform others, and by using only strong experts within each context, we improve the final model.

Let the dataset $\mathbb{D}$ be defined as $(x_i, y_i) \in \mathbb{D} \subset \mathbb{R}^n \times \mathbb{R} : i = \{1, \ldots, N\}$ and generated by an unknown function $f(x) = y$, where n is the number of features of an instance x, and y denotes a numerical response. Let $\hat{f}_j(x) : j = \{1, .., M\}$ be a set of M base models (*experts*) learned using one or more base learning methods over $\mathbb{D}$. Let $\mathcal{L}$ denote a loss function of interest decomposable in independent bias/variance components (e.g. $L2$-loss). For each instance x_i, let $\{z_{i,1}, \ldots, z_{i,Q}\}$ be the set of meta-features generated for that instance.

Starting from the definition of a decision tree for supervised learning introduced in CART [6], we aim to build a *classification* tree that, for a given instance x and its supporting meta-features $\{z_1, \ldots, z_Q\}$, dynamically selects the expert that should be chosen for prediction, i.e., $\hat{F}(x, z_1, \ldots, z_Q; \hat{f}_1, \ldots, \hat{f_M}) = \hat{f}_j(x)$. As for the tree induction procedure, we aim, at each node, at finding the feature z_j and the splitting point z_j^t that leads to the maximum reduction of impurity. For the internal node p with the set of examples $\mathbb{D}_p \in \mathbb{D}$ that reaches p, the splitting point z_j^t splits the node p into the leaves p_l and p_r with the sets

$\mathbb{D}_{p_l} = \{x_i \in \mathbb{D}_p | z_{ij} \leq z_j^t\}$ and $\mathbb{D}_{p_r} = \{x_i \in \mathbb{D}_p | z_{ij} > z_j^t\}$, respectively. This can be formulated by the following optimization problem at each node:

$$\arg\max_{z_j^t} \quad \omega(z_j^t) \tag{2}$$

$$\text{s.t.} \quad \omega(z_j^t) = [I(p) - P_l I(p_l) - P_r I(p_r)] \tag{3}$$

where P_l, P_r denote the probability of each branch to be selected, while I denotes the so-called *impurity function*. In traditional classification problems, the functions applied here aim to minimize the entropy of the target variable. Hereby, we propose a new impurity function for this purpose denoted as Inductive Bias Reduction. It goes as follows:

$$I(p) = \text{IBR}(p) = \min_{j \in \{1...M\}} E\left[B\big(\mathcal{L}(p, \hat{f}_j)\big)^2 \right] \tag{4}$$

where $B(\mathcal{L})$ denotes the inductive bias component of the loss $\mathcal{L}$.
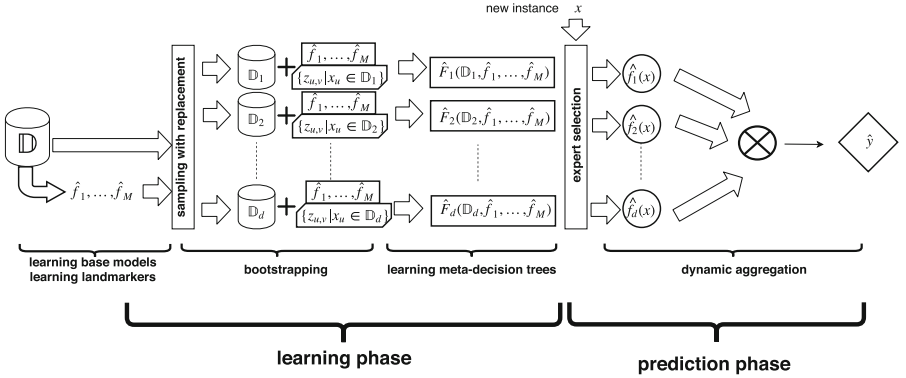
Fig. 2. MetaBags: The learning phase consists of (i) the learning of base models and the landmarkers, (ii) bootstrapping and finally (iii) the learning of the meta decision trees from each bootstrap. The prediction for an unseen example is achieved by consulting each meta decision tree and then aggregating their predictions.

Optimization. To solve the problem of Eq. (2), we address three issues: (i) splitting criterion/meta-feature, (ii) splitting point and (iii) stopping criterion. To select the splitting criterion, we start by constructing two auxiliary equally-sized matrices $a \in \mathbb{R}^{Q \times \phi}$ and $b : z_{i_{\min}} \leq b_{i,j} \leq z_{i_{\max}}, \forall i, j$, where $\phi \in \mathbb{N}, Q$ denote a user-defined hyperparameter and the number of meta-features used, respectively. Then, the matrices are populated with candidate values by elaborating over the Eq. (2, 3, 4) as

$$a_{i,j} = \omega(b_{i,j}), \quad b_{i,j} \sim U(z_{i_{\min}}, z_{i_{\max}}), \tag{5}$$

where $b_{i,j}$ is the jth splitting criterion for the ith meta feature.

First, we find the splitting criteria τ such that

$$\tau = \underset{i \in \{1..Q\}}{\arg\max} \ a_{i,j}, \quad \forall j \in \{1..\phi\}. \tag{6}$$

Secondly, we need to find the optimal splitting point according to the z_τ criteria. We can either take the splitting point already used to find τ or, alternatively, fine-tune the procedure by exploring further the domain of z_τ. For the latter problem, any scalable search heuristic can be applied (e.g.: Golden-section search algorithm [22]).

Thirdly, (iii) the stopping criteria to constraint Eq. (2). Here, like CART, we propose to create fully grown trees. Therefore, it goes as follows:

$$\omega\big(z_\tau^t\big) < \epsilon \ \vee \ |\mathbb{D}_p| < \upsilon : \epsilon \in \mathbb{R}^+, \upsilon \in \mathbb{N} \tag{7}$$

where ϵ, υ are user-defined hyperparameters. Intuitively, this procedure consists in randomly finding ϕ possible *partitioning* points on each meta-feature in a parallelizable fashion in order to select one splitting criterion.

The pseudocode of this algorithm is presented in Algorithm 1.

3.2   Bagging at Meta-Level: Why and How?

Bagging [4] is a popular ensemble learning technique. It consists of forming multiple d replicate datasets $\mathbb{D}^{(B)} \subset \mathbb{D}$ by drawing $s << N$ examples from $\mathbb{D}$ at random, but with replacement, forming bootstrap samples. Next, d base models $\varphi(x_i, \mathbb{D}^{(B)})$ are learned with a selected method on each $\mathbb{D}^{(B)}$, and the final prediction $\varphi_A(x_i)$ is obtained by averaging the predictions of all d base models. As Breiman demonstrates in Sect. 4 of [4], the amount of expected improvement of the aggregated prediction $\varphi_A(x_i)$ depends on the gap between the terms of the following inequality:

$$E\big[\mathcal{L}\big(\varphi(x_i, \mathbb{D}^{(B)})\big)\big]^2 \leq E\big[\mathcal{L}\big(\varphi(x_i, \mathbb{D}^{(B)})\big)^2\big]. \tag{8}$$

In our case, $\varphi(x_i, \mathbb{D}^{(B)})$ is given by the $\hat{f}_j(x_i)$ selected by each meta-decision tree induced in each $\mathbb{D}^{(B)}$. By design, the procedure to learn this specific meta-decision tree is likely to overfit its training set, since all the decisions envisage reduction of inductive bias alone. However, when used in a bagging context, this turns to be an advantage because it causes instability of φ - as each tree may be selecting different predictors to each instance x_i.

3.3   Meta-Features

MetaBags is fed with three types of meta-features: (a) base, (b) performance-related and (c) local landmarking. These types are briefly explained below, as well as their connection with the state of the art.

(a) Base Features. Following [30], we propose to include all base features also as meta-features. This aims to stimulate a higher inequality in Eq. (8) due to the increase of inductive variance of each meta-predictor.

(b) Performance-Related Features. This type of meta-features describes the performance of specific learning algorithms in particular learning contexts on the same dataset. Besides the base learning algorithms, we also propose the usage of landmarkers. Landmarkers are ML algorithms that are computationally relatively *cheap* to run either in a train or test setting [27]. The resulting models aim to characterize the learning task (e.g. is the regression curve linear?). To the authors' best knowledge, so far, all proposed landmarkers and consequent meta-features have been primarily designed for classical meta-learning applications to classification problems [3,27], whereas we focus on model integration for regression. We use the following learning algorithms as landmarkers: `LASSO` [29], `1NN` [10], `MARS` [15] and `CART` [6].

   To generate the meta-features, we start by creating one landmarking model per method over the entire training set. Then, we design a *small* artificial neighborhood of size ψ of each training example x_i as $X'_i = \{x'_{i,1}, x'_{i,2}..x'_{i,\psi}\}$ by perturbing x_i with *gaussian noise* as follows:

$$x'_{i,j} = x_{i,j} + \xi : \xi \sim \mathcal{N}_n(0,1), \forall j \in \{1,..,\psi\} \tag{9}$$

where ψ, is a user-defined hyperparameter. Then, we obtain outputs of each expert as well as of each landmarker given X'_i. The used meta-features are then descriptive statistics of the models' outputs: mean, stdev., 1st/3rd quantile. This procedure is applicable both to training and test examples, whereas the landmarkers are naturally obtained from the training set.

(c) Local Landmarking Features. In the original landmarking paper, Pfahringer *et al.* [27] highlight the importance on ensuring that our pool of landmarkers is diverse enough in terms of the different types of inductive bias that they employ, and the consequent relationship that this may have with the base learners performance. However, when observing performance on a neighborhood-level rather than on the task/dataset level, the low performance and/or high inductive bias may have different causes (e.g., inadequate data preprocessing techniques, low support/coverage of a particular subregion of the input space, etc.). These causes, may originate in different types of deficiencies of the model (e.g. low support of leaf nodes or high variance of the examples used to make the predictions in decision trees).

   Hereby, we introduce a novel type of landmarking meta-features denoted **local landmarking**. Local landmarking meta-features are designed to characterize the landmarkers/models within the particular input subregion. More than finding a correspondence between the performance of landmarkers and base models, we aim to extract the knowledge that the landmarkers have learned about a particular input neighborhood. In addition to the prediction of each landmarker for a given test example, we compute the following characteristics:

- `CART`: depth of the leaf which makes the prediction; number of examples in that leaf and variance of these examples;
- `MARS`: width and mass of the interval in which a test example falls, as well as its distance to the nearest edge;
- `1NN`: absolute distance to the nearest neighbor.

4   Experiments and Results

Empirical evaluation aims to answer the following four research questions:

(Q1) Does `MetaBags` systematically outperform its base models in practice?
(Q2) Does `MetaBags` outperform other model integration procedures?
(Q3) Do the local landmarking meta-features improve `MetaBags` performance?
(Q4) Does `MetaBags` scale on large-scale and/or high-dimensional data?

4.1   Regression Tasks

We used a total of 17 benchmarking datasets to evaluate `MetaBags`. They are summarized in Table 1. We include 4 proprietary datasets addressing a particular real-world application: public transportation. One of its most common research problems is travel time prediction (TTP). The work in [19] uses features such as scheduled departure time, vehicle type and/ or driver's meta-data. This type of data is known to be particularly noisy due to failures in the data collection, which in turn often lead to issues such as missing data, as well as several types of outliers [26].

Here, we evaluate `MetaBags` in a similar setting of [19], i.e. by using their four datasets and the original preprocessing. This case study is an undisclosed large urban bus operator in Sweden (BOS). We collected data on four high-frequency routes/datasets R11/R12/R21/R22. These datasets cover a time period of six months.

4.2   Evaluation Methodology

Hereby, we describe the empirical methodology designed to answer (Q1-Q4), including the hyperparameter settings of `MetaBags` and the algorithms selected for comparing the different experiments.

Hyperparameter Settings. Like other decision tree-based algorithms, `MetaBags` is expected to be robust to its hyperparameter settings. Table 2 presents the hyperparameters settings used in the empirical evaluation (a sensible default). If any, s and d can be regarded as more sensitive parameters. Their value ranges are recommended to be $0\% < s << 100\%$ and $100 \leq d << 2000$. Please note that these experiments did not included any hyperparameter sensitivity study neither a tuning procedure for `MetaBags`.

Testing Scenarios and Comparison Algorithms. We put in place two testing scenarios: A and B. In scenario A, we evaluate the generalization error of MetaBags with 5-fold cross validation (CV) with 3 repetitions. As base learners, we use four popular regression algorithms: Support Vector Regression (SVR) [11], Projection Pursuit Regression (PPR) [17], Random Forest RF [5] and Gradient Boosting GB [16]. The first two are popular methods in the chosen application domain [19], while the latter are popular voting-based ensemble methods for regression [21]. The base models had their hyperparameter values tuned with random search/3-fold CV (and 60 evaluation points). We used the implementations in the R package *caret* for both the landmarkers and the base learners. We compare our method to the following ensemble approaches: Linear Stacking LS [7], Dynamic Selection DS with kNN [25,32], and the best individual model. All methods used $l2$-loss as $\mathcal{L}$.

In scenario B, we extend the artificial dataset used in Fig. 1 to assess the computational runtime scalability of the decision tree induction process of MetaReg (using a CART-based implementation) in terms of number of examples and attributes. In this context, we compare our method's training stage to Linear Regression (used for LS) and kNN in terms of time to build k-d tree (DS). Additionally, we also benchmarked C4.5 (which was used in MDT [30]). For the latter, we discretized the target variable using the four quantiles.

4.3   Results

Table 3 presents the performance results of MetaBags against comparison algorithms: the base learners; SoA in model integration such as stacking with a linear model LS and kNN, i.e. DS, as well as the best base model selected using 3-CV i.e. Best; finally, we also included two variants of MetaBags: MetaReg – a singular decision tree, MBwLM – MetaBags without the novel landmarking features. Results are reported in terms of RMSE, as well as of statistical significance (using the using the two-sample t-test with the significance level $\alpha = 0.05$, with the null hypothesis that a given learner M wins against MetaBags after observing the results of all repetitions). Finally, Fig. 3 summarizes those results in terms of percentual improvements, while Fig. 4 depicts our empirical scalability study.

5   Discussion

The results, presented in Table 3, show that MetaBags outperforms existing SoA stacking methods. MetaBags is never statistically significantly worse than any of the other methods, which illustrates its generalization power.

Figure 3 summarizes well the contribution of introducing bagging at the meta-level as well as the novel local landmarking meta-features, with average relative percentages of improvement in performance across all datasets of 12.73% and 2.67%, respectively. The closest base method is GB, with an average percentage of improvement of 5.44%. However, if we weight this average by using the percentage of extreme target outliers of each dataset, the expected improvement goes up to 14.65% - illustrating well the issues of GB depicted earlier in Fig. 1(b).

Table 1. Datasets summary. Fields denote number of #ATTributes and #INStances, the Range of the Target variable, the number of #Target Outliers using Tukey's boxplot(ranges = 1.5,3), their ORIgin, as well as its TYPe (**P**roprietary/**O**pen) and Collection Process (**R**eal/**A**rtificial).

	Properties					Source and type		
	#ATT	#INS	RT	#TO(1.5)	#TO(3.0)	ORI	TYP	CP
R11	12	17953	[1306,10520]	66	9	BOS	P	R
R12	12	16353	[1507,9338]	154	6	BOS	P	R
R21	12	16280	[1434,6764]	341	27	BOS	P	R
R22	12	16353	[884,6917]	146	10	BOS	P	R
Cal. housing	8	20460	[14999,500001]	1071	0	StatLib	O	R
Concrete	8	1030	[2332,82599]	4	0	UCI	O	R
2Dplanes	10	40768	[−999.709,999.961]	4	0	dcc.fc.up.pt [31]	O	A
Delta Ailerons	6	7129	[−0.0021,0,0022]	107	12	dcc.fc.up.pt [31]	O	R
Elevators	18	16559	[0.012,0,078]	842	344	dcc.fc.up.pt [31]	O	R
Parkinsons Tele.	26	5875	[0.022,0732]	206	17	UCI	O	R
Physicochemical	9	45730	[15.228,55.3009]	0	0	UCI	O	R
Pole	48	15000	[0,100]	0	0	dcc.fc.up.pt [31]	O	R
Puma32H	32	8192	[−0.085173,0.088266]	56	0	DELVE	O	R
Red wine quality	12	1599	[3,8]	28	0	UCI	O	R
White wine quality	12	4898	[3,9]	200	0	UCI	O	R
Computer activity								
CPU-small	12	8192	[0,99]	430	294	DELVE	O	R
CPU-activity	21	8192	[0,99]	430	294	DELVE	O	R

Table 2. Hyperparameter settings used in `MetaBags`.

|	Value	Description
ϕ	10	Number of random partitions performed on each meta-feature;
ϵ	$|I(t_p) \cdot 10^{-2}|$	Min. abs. bias reduction to perform split;
υ	$N \cdot 10^{-2}$	Min. examples in node to perform split;
ψ	100	Size of the artificial neighborhood generated to compute meta-features;
s	10%	Percentage of examples usage to generate the bootstraps;
d	300	Number of generated meta-decision trees;

Figure 4 also depicts how competitive `MetaBags` can be in terms of scalability. Although neither outperforming `DS` nor `LS`, we want to highlight that lazy learners have their cost in test time - while this study only covered the training stage. Moreover, many of its stages (learning of base learners, performance-based meta-features, local landmarking) as well as subroutines of the `MetaBags` are independent and thus, trivially parallelizable. Based in the above discussion, (Q1-Q4) can be answered affirmatively.

One possible drawback of `MetaBags` may be its space complexity - since it requires to train/maintain multiple decision trees and models in memory. Another possible issue when dealing with low latency data mining applications is that the computation of some of the meta-features is not trivial, which may

slightly increase its runtimes in test stage. Both issues were out of the scope of the proposed empirical evaluation and represent open research questions.

Like any other stacking approach, `MetaBags` requires training of the base models apriori. This pool of models need to have some diversity on their responses. Hereby, we explore the different characteristics of different learning algorithms to stimulate that diversity. However, this may not be sufficient.
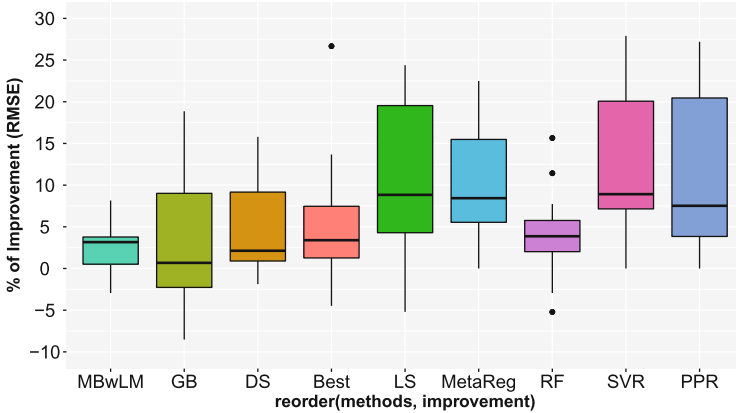
Fig. 3. Summary results of `MetaBags` using the percentage of improvement over its competitors. Note the consistently positive mean over all methods.

Algorithm 1. InduceMetaDecisionTreeRegression

Input: p: root (or current internal node) of the meta decision tree.
$\mathbb{D}_p \subset \mathbb{D}_i$: the subset of examples that reach the root (or internal node) p, $\mathbb{D}_i$ is a bootstrap sampled with replacement from $\mathbb{D}$.
$\{\hat{f}_j | j \in \{1..M\}\}$: the set of base models.
$\{z_{u,v} | x_u \in \mathbb{D}_p \wedge v \in \{1, \ldots, Q\}\}$: the set of meta features for each instance in $\mathbb{D}_p$.
/* check that the current node has the minimum number of supporting instances   */
1 **if** $|\mathbb{D}_p| \leq v$ **then**
2   **return**
    /* create the matrices A and B */
3 $B = [b_{i,j}] \in \mathbb{R}^{Q \times \phi}$ s.t. $b_{i,j} \sim U(z_{i_{\min}}, z_{i_{\max}})$
4 $A = [a_{i,j}] \in \mathbb{R}^{Q \times \phi}$ s.t. $a_{i,j} = \omega(b_{i,j}) = [I(p) - P_l I(p_l) - P_r I(p_l)]$
    /* find the splitting criteria τ */
5 $\tau = \arg\min_{i \in \{1..Q\}} a_{i,j}$,   for all $j \in \{1, \ldots, \phi\}$
6 **if** $\omega(z_\tau^t) \geq \epsilon$ **then**
    /* create the right and left leaf nodes, p_l and p_r */
7      $\mathbb{D}_{p_l} = \{x_i \in \mathbb{D}_p | z_{i\tau} \leq z_\tau^t\}$
8      $\mathbb{D}_{p_r} = \{x_i \in \mathbb{D}_p | z_{i\tau} > z_\tau^t\}$
9   InduceMetaDecisionTree$(p_l, \mathbb{D}_{p_l}, \{\hat{f}_j\}, \{z_{u,v}\})$
10   InduceMetaDecisionTree$(p_r, \mathbb{D}_{p_r}, \{\hat{f}_j\}, \{z_{u,v}\})$
11 **return**

Formal approaches to strictly ensure diversity on model generation for ensemble learning in regression are scarce [8,24]. The best way to ensure such diversity within an advanced stacking framework like `MetaBags` is also an open research question.

Table 3. Detailed predictive performance results, comparing base learners vs. `MetaBags` (top) and SoA methods in model integration vs. `MetaBags` - including variations (bottom). The results reported on the average and (std. error) of RMSE. The last rows depict the wins and losses based on the two-sample t-test with the significance level $\alpha = 0.05$ and the null hypothesis that a given learner M wins against MetaBags (computed after observing the results of all repetitions in a pair-wise manner).

Dataset	SVR	PPR	RF	GB	MetaBags
R11	$232.23_{(5.2)}$	$242.12_{(8.1)}$	$229.18_{(9.8)}$	$225.45_{(6.5)}$	$\mathbf{220.37_{(4.6)}}$
R12	$210.66_{(3.9)}$	$217.34_{(5.4)}$	$205.77_{(2.5)}$	$200.49_{(6.7)}$	$\mathbf{194.36_{(4.2)}}$
R21	$225.81_{(6.8)}$	$240.12_{(5.3)}$	$235.55_{(4.0)}$	$\mathbf{210.45_{(7.7)}}$	$218.87_{(4.9)}$
R22	$260.77_{(5.9)}$	$269.19_{(6.5)}$	$255.91_{(5.3)}$	$\mathbf{248.11_{(5.2)}}$	$253.74_{(2.9)}$
C. Housing	$9.6e^4{(3.1e^3)}$	$10.0e^4{(5.5e^3)}$	$9.9e^4{(5.0e^3)}$	$\mathbf{8.9e^4{(3.9e^3)}}$	$9.2e^4{(3.8e^3)}$
Concrete	$12.19_{(2.6)}$	$15.81_{(2.4)}$	$17.14_{(2.32)}$	$14.54_{(1.7)}$	$\mathbf{11.73_{(0.3)}}$
Delta A.	$3.2e^{-4}{(1.2e^{-4})}$	$3.6e^{-4}{(2.4e^{-4})}$	$4.2e^{-4}{(2.3e^{-4})}$	$2.5e^{-4}{(1.8e^{-4})}$	$\mathbf{2.2e^{-4}{(9.9e^{-5})}}$
2Dplanes	$2.12_{(0.1)}$	$2.57_{(0.2)}$	$2.36_{(0.1)}$	$\mathbf{2.01_{(0.2)}}$	$\mathbf{2.02_{(0.1)}}$
Elevators	$6.2e^{-3}{(3.8e^{-4})}$	$6.3e^{-3}{(5.8e^{-4})}$	$6.4e^{-3}{(5.4e^{-4})}$	$6.7e^{-3}{(4.5e^{-4})}$	$\mathbf{5.6e^{-3}{(5.8e^{-4})}}$
Parkinsons	$5.5e^{-2}{(7.0e^{-3})}$	$8.1e^{-2}{(7.0e^{-3})}$	$7.5e^{-2}{(6.0e^{-3})}$	$6.4e^{-2}{(6.0e^{-3})}$	$\mathbf{4.9e^{-2}{(5.0e^{-3})}}$
Physic.	$3.78_{(0.1)}$	$3.90_{(0.2)}$	$4.99_{(0.2)}$	$3.70_{(0.2)}$	$\mathbf{3.64_{(0.2)}}$
Pole	$24.11_{(7.2)}$	$30.24_{(9.9)}$	$28.90_{(5.9)}$	$\mathbf{18.54_{(9.3)}}$	$20.12_{(3.1)}$
Puma32H	$3.0e^{-2}{(3.0e^{-3})}$	$2.9e^{-2}{(3.0e^{-3})}$	$2.8e^{-2}{(3.0e^{-3})}$	$\mathbf{2.5e^{-2}{(6.0e^{-3})}}$	$2.7e^{-2}{(4.0e^{-3})}$
R. Wine	$\mathbf{0.69_{(0.0)}}$	$0.91_{(0.0)}$	$0.88_{(0.0)}$	$0.79_{(0.0)}$	$0.70_{(0.0)}$
W. Wine	$0.70_{(0.0)}$	$0.86_{(0.0)}$	$0.76_{(0.0)}$	$\mathbf{0.62_{(0.0)}}$	$\mathbf{0.62_{(0.0)}}$
CPU_a.	$\mathbf{5.18_{(0.4)}}$	$5.89_{(0.3)}$	$6.87_{(0.2)}$	$5.99_{(0.3)}$	$5.45_{(0.2)}$
CPU_s.	$6.07_{(0.1)}$	$6.37_{(0.3)}$	$8.71_{(0.4)}$	$6.31_{(0.5)}$	$\mathbf{5.12_{(0.1)}}$
∅ **Rank**	2.82	4.47	4.12	2.12	1.41
Loss/Win	10/0	11/0	4/0	1/0	N/A

Dataset	LS	DS	Best	MetaReg	MBwLM	MetaBags
R11	$230.25_{(9.2)}$	$222.37_{(5.4)}$	$223.20_{(8.9)}$	$240.67_{(15.7)}$	$228.32_{(7.0)}$	$\mathbf{220.37_{(4.6)}}$
R12	$215.32_{(5.9)}$	$\mathbf{190.80_{(3.5)}}$	$201.46_{(4.3)}$	$219.76_{(19.2)}$	$211.63_{(7.9)}$	$194.37_{(4.2)}$
R21	$234.37_{(7.1)}$	$221.98_{(5.8)}$	$226.27_{(4.2)}$	$249.31_{(6.1)}$	$220.01_{(7.5)}$	$\mathbf{218.87_{(4.9)}}$
R22	$260.71_{(4.5)}$	$\mathbf{250.37_{(5.5)}}$	$254.69_{(5.2)}$	$273.12_{(5.4)}$	$261.41_{(4.3)}$	$253.744_{(2.9)}$
C. Housing	$9.8e^4{(4.3e^3)}$	$9.3e^4{(3.2e^3)}$	$9.5e^4{(4.6e^3)}$	$9.8e^4{(4.1e^3)}$	$9.5e^4{(3.7e^3)}$	$\mathbf{9.2e^4{(3.7e^3)}}$
Concrete	$12.14_{(0.3)}$	$\mathbf{11.70_{(0.3)}}$	$12.00_{(0.3)}$	$12.57_{(0.3)}$	$11.91_{(0.3)}$	$11.73_{(0.3)}$
Delta A.	$2.8e^{-4}{(2.3e^{-4})}$	$2.6e^{-4}{(1.8e^{-4})}$	$3.0e^{-4}{(1.8e^{-4})}$	$3.5e^{-4}{(1.7e^{-4})}$	$2.3e^{-4}{(1.8e^{-4})}$	$\mathbf{2.2e^{-4}{(9.9e^{-5})}}$
2Dplanes	$2.35_{(0.2)}$	$2.26_{(0.1)}$	$2.34_{(0.1)}$	$2.39_{(0.1)}$	$2.15_{(0.1)}$	$\mathbf{2.02_{(0.1)}}$
Elevators	$6.2e^{-3}{(5.5e^{-4})}$	$5.5e^{-3}{(7.0e^{-4})}$	$7.4e^{-3}{(6.1e^{-4})}$	$6.5e^{-3}{(4.6e^{-4})}$	$5.5e^{-3}{(6.8e^{-4})}$	$\mathbf{5.7e^{-3}{(5.8e^{-4})}}$
Parkinsons	$5.9e^{-2}{(6.0e^{-3})}$	$6.5e^{-2}{(0.0e^{-3})}$	$6.2e^{-2}{(5.0e^{-3})}$	$5.4e^{-2}{(5.0e^{-3})}$	$5.1e^{-2}{(5.0e^{-3})}$	$\mathbf{4.9e^{-2}{(5.0e^{-3})}}$
Physic.	$\mathbf{3.46_{(0.2)}}$	$3.74_{(0.2)}$	$3.79_{(0.2)}$	$3.80_{(0.2)}$	$3.71_{(0.2)}$	$3.64_{(0.2)}$
Pole	$26.22_{(3.3)}$	$22.15_{(3.6)}$	$21.61_{(3.3)}$	$25.96_{(4.3)}$	$20.91_{(3.6)}$	$\mathbf{20.12_{(3.1)}}$
Puma32H	$4.0e^{-2}{(5.0e^{-3})}$	$3.2e^{-2}{(6.0e^{-3})}$	$3.4e^{-2}{(4.0e^{-3})}$	$3.3e^{-2}{(6.0e^{-3})}$	$2.9e^{-2}{(4.0e^{-3})}$	$\mathbf{2.7e^{-2}{(4.0e^{-3})}}$
R. Wine	$0.87_{(0.0)}$	$0.75_{(0.0)}$	$\mathbf{0.67_{(0.0)}}$	$0.76_{(0.0)}$	$0.70_{(0.0)}$	$0.70_{(0.0)}$
W. Wine	$0.82_{(0.0)}$	$0.63_{(0.0)}$	$0.67_{(0.0)}$	$0.76_{(0.0)}$	$\mathbf{0.61_{(0.0)}}$	$0.62_{(0.0)}$
CPU_a.	$5.92_{(0.2)}$	$5.93_{(5.5)}$	$5.58_{(0.3)}$	$5.54_{(0.3)}$	$5.57_{(0.2)}$	$\mathbf{5.45_{(0.2)}}$
CPU_s.	$5.92_{(0.2)}$	$6.08_{(0.3)}$	$5.90_{(0.2)}$	$6.12_{(0.3)}$	$5.30_{(0.2)}$	$\mathbf{5.12_{(0.2)}}$
∅ **Rank**	4.76	3.06	3.88	5.12	2.65	1.47
Loss/Win	4/0	2/0	4/0	6/0	0/0	N/A
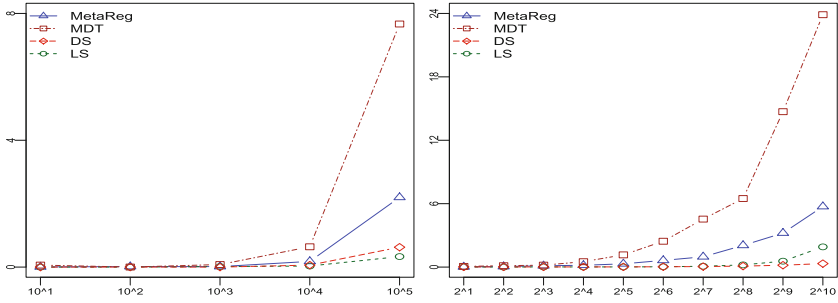
Fig. 4. Empirical runtime scalability analysis resorting to samples (left panel) and features (right panel) size. Times in seconds.

6   Final Remarks

This paper introduces `MetaBags`: a novel, practically useful stacking framework for regression. `MetaBags` uses meta-decision trees that perform on-demand selection of base learners at test time based on a series of innovative meta-features. These meta-decision trees are learned over data bootstrap samples, whereas the outputs of the selected models are combined by average. An exhaustive empirical evaluation, including 17 datasets and multiple comparison algorithms illustrates the ability of `MetaBags` to address model integration problems in regression. As future work, we aim to study which factors affect the performance of `MetaBags`, namely, at model generation level, as well as its time and spatial complexity in test time.

References

1. Bell, R., Koren, Y.: Lessons from the netflix prize challenge. ACM SIGKDD Explor. Newsl. **9**(2), 75–79 (2007)
2. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proceedings of the 23rd ICML, pp. 97–104. ACM (2006)
3. Brazdil, P., Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. Springer, Heidelberg (2008)
4. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
5. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
6. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and regression trees (cart) wadsworth international group, CA, USA, Belmont (1984)
7. Breiman, L.: Stacked regressions. Mach. Learn. **24**(1), 49–64 (1996)
8. Brown, G., Wyatt, J.L., Tiňo, P.: Managing diversity in regression ensembles. J. Mach. Learn. Res. **6**, 1621–1650 (2005)

9. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**(3), 15 (2009)
10. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
11. Drucker, H., Burges, C., Kaufman, L., Smola, A., Vapnik, V.: Support vector regression machines. In: NIPS, pp. 155–161 (1997)
12. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: NIPS, pp. 2962–2970 (2015)
13. Feurer, M., Springenberg, J., Hutter, F.: Initializing Bayesian hyperparameter optimization via meta-learning. In: AAAI, pp. 1128–1135(2015)
14. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
15. Friedman, J.: Multivariate adaptive regression splines. Ann. Stat. **19**, 1–67 (1991)
16. Friedman, J.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**, 1189–1232 (2001)
17. Friedman, J., Stuetzle, W.: Projection pursuit regression. J. Am. Stat. Assoc. **76**(376), 817–823 (1981)
18. Gama, J., Brazdil, P.: Cascade generalization. Mach. Learn. **41**(3), 315–343 (2000)
19. Hassan, S.M., Moreira-Matias, L., Khiari, J., Cats, O.: Feature selection issues in long-term travel time prediction. In: Boström, H., Knobbe, A., Soares, C., Papapetrou, P. (eds.) IDA 2016. LNCS, vol. 9897, pp. 98–109. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46349-0_9
20. Hastie, T., Tibshirani, R.: Generalized additive models: some applications. J. Am. Stat. Assoc. **82**(398), 371–386 (1987)
21. Kaggle Inc.: https://www.kaggle.com/bigfatdata/what-algorithms-are-most-successful-on-kaggle. Technical report (Eletronic, Accessed in March 2018)
22. Kiefer, J.: Sequential minimax search for a maximum. Proc. Am. Math. Soc. **4**(3), 502–506 (1953)
23. Lemke, C., Budka, M., Gabrys, B.: Metalearning: a survey of trends and technologies. Artif. Intell. Rev. **44**(1), 117–130 (2015)
24. Mendes-Moreira, J., Soares, C., Jorge, A., Sousa, J.: Ensemble approaches for regression: a survey. ACM Comput. Surv. (CSUR) **45**(1), 10 (2012)
25. Merz, C.: Dynamical Selection of Learning Algorithms, pp. 281–290 (1996)
26. Moreira-Matias, L., Mendes-Moreira, J., Freire de Sousa, J., Gama, J.: On improving mass transit operations by using AVL-based systems: a survey. IEEE Trans. Intell. Transp. Syst. **16**(4), 1636–1653 (2015)
27. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: ICML, pp. 743–750 (2000)
28. Schaffer, C.: A conservation law for generalization performance. In: Machine Learning Proceedings 1994, pp. 259–265. Elsevier (1994)
29. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Series B (Methodological) **58**(1), 267–288 (1996)
30. Todorovski, L., Dzeroski, S.: Combining classifiers with meta decision trees. Mach. Learn. **50**(3), 223–249 (2003)
31. Torgo, L.: Regression data sets. Eletronic (last access at 02/2018) (February 2018). http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html
32. Tsymbal, A., Pechenizkiy, M., Cunningham, P.: Dynamic integration with random forests. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 801–808. Springer, Heidelberg (2006). https://doi.org/10.1007/11871842_82
33. Wolpert, D.: Stacked generalization. Neural Netw. **5**(2), 241–259 (1992)