Neurocomputing 551 (2023) 126516

Contents lists available at ScienceDirect

Neurocomputing

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

Regularized boosting with an increasing coefficient magnitude stop criterion as meta-learner in hyperparameter optimization stacking ensemble

Laura Fdez-Díaz [a], José Ramón Quevedo [b], Elena Montañés [b,*]

[a] Global R&D ArcelorMittal, Calle Marineros, 4, 33490 Avilés, Asturias Spain
[b] Artificial Intelligence Center, University of Oviedo, 33204 Gijón, Asturias, Spain

ARTICLE INFO

Article history:
Received 22 July 2022
Revised 29 April 2023
Accepted 30 June 2023
Available online 6 July 2023
Communicated by Zidong Wang

Keywords:
Hyperparameter optimization
Stacking ensemble
Boosting

ABSTRACT

Hyperparameter Optimization (HPO) aims to tune hyperparameters for a system in order to improve the predictive performance. Typically, only the hyperparameter configuration with the best performance is chosen after performing several trials. However, some works try to take advantage of the effort made when training all the models with every hyperparameter configuration trial and, instead of discarding all but one, they propose performing an ensemble of all the models. However, this ensemble consists of simply averaging the model predictions or weighting the models by a certain probability. Recently, some of the so-called Automated Machine Learning (AutoML) frameworks have included other more sophisticated ensemble strategies, such as the Caruana method or the stacking strategy. On the one hand, the Caruana method has been shown to perform well in HPO ensemble, since it is not affected by the issues caused by multicollinearity, which is prevalent in HPO. It just computes the average over a subset of predictions, previously chosen through a forward stepwise selection with replacement. But it does not benefit from the generalization power of a learning process. On the other hand, stacking approaches include a learning procedure since a meta-learner is required to perform the ensemble. Yet, one hardly finds advice about which meta-learner can be adequate. Besides, some possible meta-learners may suffer from problems caused by multicollinearity or need to be tuned in order to mitigate or reduce this obstacle. In an attempt to reduce this lack of advice, this paper exhaustively explores possible meta-learners for stacking ensemble in HPO, free of hyperparameter tuning and able to mitigate the problems derived from multicollinearity as well as taking advantage of the generalization power that a learning process may include in the ensemble. Particularly, the boosting strategy shows promise in this context as a stacking meta-learner, since it satisfies the required conditions. In addition, boosting is even able to completely remove the effects of multicollinearity. This paper provides advice on how to use boosting as a meta-learner in the stacking ensemble. In any case, its main contribution is to propose an implicit regularization in the classical boosting algorithm and a novel non-parametric stop criterion suitable only for boosting and specifically designed for the HPO context. The existing synergy between these two improvements performed over boosting exhibits competitive and promising predictive power performance as a stacking meta-learner in HPO compared to other existing meta-learners and ensemble approaches for HPO other than the stacking ensemble.

1. Introduction

Hyperparameter Optimization (HPO) [1] research rises from the need to find promising hyperparameter configurations in machine learning systems in order to achieve high predictive performance [2]. The hyperparameters determine the structure of the model and how the learning process will take place. They must be tuned before the learning process starts and they must be differentiate from the model parameters. Model parameters are estimated during the learning process, they configure the model itself and must be taken into account for making predictions.

HPO aims to obtain an optimal model that minimizes a prefixed loss function or maximizes a performance measure. Typically, the

* Corresponding author at: Edificio Departamental Oeste, Nº1 Despacho 1.B.12, Campus de Viesques, s/n., 33204 Gijón, Asturias Spain.
E-mail addresses: laura.fernandezdiaz@arcelormittal.com (L. Fdez-Díaz), quevedo@uniovi.es (J.R. Quevedo), montaneselena@uniovi.es (E. Montañés).

https://doi.org/10.1016/j.neucom.2023.126516
0925-2312/

hyperparameter tuning process involves i) defining a model structure, ii) establishing the hyperparameters to be tuned and the domains for their values, iii) designing a hyperparameter value sampling method, iv) establishing an estimation procedure given an evaluation metric and, finally, v) configuring the final model. Among all these steps in hyperparameter tuning, researchers have focused their attention on developing promising strategies for hyperparameter sampling (step iii)) [1,2]. The common procedure in the literature for steps i), ii) and iv) involves respectively: checking several model structures, several hyperparameter domains and several evaluation procedures. Regarding the final configuration of the model (step v)), a typical approach would involve selecting the model with the best hyperparameter configuration based on the averaged evaluation metric estimations and then discarding the models with the rest of the hyperparameter configuration trials. Despite the selected model does provide the best estimation on average, it might not be the best option overall. This means that more than one option might include at least some predictive power and may contribute to better overall performance. Hence, ensembling the models induced by the hyperparameter configuration trials [3] seems like a straightforward favourable strategy to consider (one must not confuse this practice with using a sampling strategy in order to optimize an ensemble [4]). In fact, there is a theoretical study based on ambiguity decomposition [5] that shows that an adequate ensemble guarantees a better performance than the averaged performance of the individual models.

Some existing works have already explored the impact of performing ensemble in HPO [6–8], including the Basic Ensemble Method (BEM) [9], which computes the average of the model predictions, the Generalized Ensemble Method (GEM) [9,10], which performs regularized least squares regression under certain constraints, or weighting by a probability as in the Inverse Expected Error Weighting (IEW) [8]. Also, and more recently, some Automated Machine Learning (AutoML) systems include the option of performing an ensemble within their frameworks. However, only few of them do. In fact, AutoML systems mainly focus on other parts of the process, such as parallelizing or distributing the process, or on improving the performance of exploring encouraging hyperparameter configuration trials, or even on multi-output HPO [11]. The few systems that include ensemble in their scenarios do not perform an exhaustive study on them; instead they just contemplate the option whether to perform ensemble or not. These few AutoML systems either use the Caruana method [12] for the ensemble (as in Auto-Gluon [13], Auto-Sklearn [14,15] or Auto-Pytorch [16]) or adopt stacking ensemble [17] (as in Auto-Weka [18,19] or H2O [20,21]). Despite the Caruana method having shown to perform successfully due to the forward selection with replacement strategy, it only computes the simple average afterwards and does not profit from the generalization power of including a learning procedure. Unlike the Caruana method, the stacking procedure learns the ensemble, but in order for the system to learn the ensemble the meta-learner, must be carefully chosen. In fact, and to the best of our knowledge, the literature provides no guidelines as to which systems may be adequate in HPO. Furthermore, a quite recent survey argues this shortage of studies about stacking ensemble [22] for general purpose ensemble, hence, all the more reason for HPO. Besides, if the system has hyperparameters, they must be carefully tuned in order to get good predictive performance, typically having to use an HPO procedure in turn.

Bagging and boosting [23] are other general-purpose ensemble strategies widely used in the literature. The models to ensemble under these strategies are dynamically generated, and hence not applicable in the context of HPO, where the models to ensemble are defined beforehand and induced by the hyperparameter configuration trials. Evolutionary algorithms [24] are another kind of general-purpose ensemble strategies, but they are usually overfilled with hyperparameters that need to be tuned. Other ensemble methods exist [25], but they are developed specifically for certain cases, such as time series, neural networks, deep learning or multiple kernel learning [25]. Therefore, we discard these ensemble strategies, since they can not be adapted to HPO.

As a result, we shall now focus on stacking ensemble because (and unlike BEM, IEW and Caruana) this method takes advantage of the generalization power of ensemble through a learning process, despite a lack of guidelines about adequate non-hyperparametric meta-learners [22]. As a matter of fact, one of the contributions of this paper is to review possible and adequate candidates as meta-learners for stacking ensemble in HPO; but before we delve into that, let us review the peculiarities of the HPO stacking ensemble context. The first peculiarity is that some of these hyperparameter trials may lead to excessively general models (underfitted models) that turn out highly similar predictions for all instances. The second peculiarity is that the values to ensemble may be excessively similar for some hyperparameter configurations. This happens because in the particular case of HPO the models to ensemble are all learned using the same machine learning system and, therefore, some variations in the values of the hyperparameters may not induce enough different models. These peculiarities give rise to the multicollinearity problem [26]. Muticollinearity emerges in a multivariable regression when the variables in the regression are highly correlated. This situation affects the accuracy in estimating the regression coefficients, producing skewed, misleading and unreliable results [27]. It commonly leads to overfitted models, hence, reducing the statistical power of the regression. In HPO, this problem gets particularly worse. The reason being, as stated before, the models to ensemble may be quite similar, and hence will provide similar predictions, which are in turn the values that must be ensembled.

Ordinal Least Squares (OLS) [28] is an option as stacking meta-learner with no need to be tuned, but the above-mentioned peculiarities of the HPO ensemble context may cause overfitting. In fact, OLS is known to be highly affected by multicollinearity [29], since it takes all the features to perform the regression. Adding constraints to OLS as a way of introducing a certain regularization procedure leads to GEM, which is slightly able to reduce the effects of multicollinearity with regard to the original OLS but not to a satisfactory degree. In fact, this approach has recently been proposed for ensembling in HPO [10]. Some alternatives to OLS are Forward Stepwise Regression (FSR) [30], Principal Component Regression (PCR) [31], Partial Least Squares (PLS) [32] and Boosting (BOOST) [33]. All of these approaches are iterative procedures that require predefining a number of iterations beforehand. However, this hyperparameter can be substituted by a stop criterion and, if this stop criterion is non-hyperparametric, then, the ensemble can be considered as non-hyperparametric. Other methods are able to overcome the peculiarities of the HPO ensemble context but at the cost of tuning real-valued regularization hyperparameters. This is the case, for instance, of methods such as Ridge [34], Support Vector Regression (SVR) [35] or Random Forests Regression (RFR) [36]. In view of that, the first contribution of this paper is to discuss and explore the impact of FSR, PCR, PLS and BOOST as meta-learners for stacking ensemble in HPO, analyzing different non-hyperparametric stop criteria so that the meta-learners become non-hyperparametric.

The second and main contribution of the paper is to include two improvements in BOOST in an attempt to exploit the specific potential of this method as meta-learner for stacking ensemble in HPO. Particularly, BOOST can be promising in this context, since, unlike FSR, PCR and PLS, it carries out several regressions with just one feature each time, rather than including several features in the regression, as FSR, PCR and PLS do. The set of those different one-feature regressions are combined afterwards. On the one hand,

performing a regression with just one feature allows using OLS as the base-learner regressor, removing the problems caused by the existing multicollinearity. On the other hand, the robust way adopted by BOOST to combine the one-feature regressions makes it possible to detect collinear features as redundant, which enables significantly reducing, or even removing, the influence of these features in the ensemble. However, this reduction may not be as promising as expected. This issue prompts one of the improvements to BOOST, since one of the main contributions of this paper consists of including an implicit regularization in BOOST in order to balance the influence of the collinear features in the ensemble. This practice leads to a method that we will call Regularized BOOST (RBOOST). The second improvement stemming from the main contribution of this paper consists of designing a novel stop criterion, which we will call Increasing Coefficient Magnitude (ICM) and which is specifically designed for BOOST, taking advantage of its property about performing several regressions over just one feature. The result is that both novel improvements for BOOST, namely, implicit regularization (RBOOST) and the novel stop criterion (ICM), exert a synergy showing competitive and promising predictive power performance as meta-learner for stacking ensemble in HPO, compared to other existing non-hyperparametric meta-learners and other ensemble strategies different from stacking ensemble.

The rest of the paper is organized as follows: Section 2 describes some related work concerning the main existing AutoML frameworks and the main sampling strategies. Section 3 deals with the ensemble paradigm. First, several ensemble approaches of the literature are discussed more in depth. Then, it focuses on stacking ensemble and discusses non-hyperparametric meta-learners for it, along with several possible non-hyperparametric stop criteria. BOOST as meta-learner in stacking ensemble is deeply detailed in Section 4. In particular, this section also details the novel stop criteria ICM and RBOOST method as the result of carefully including an implicit regularization to BOOST that was specifically designed for HPO. Experimental settings together with the description of the multicollinearity analysis carried out are described in Section 5. Additionally, Section 6 presents the results and discusses the performance of the approaches. Finally, Section 7 draws some conclusions and proposes some lines of research for future work.

2. Related Work

A key step in HPO is hyperparameter sampling. Great efforts have been made in the literature to design promising sampling strategies, becoming the main aspect in the HPO field [1,2]. Some hyperparameter sampling strategies do not take into account model evaluations to obtain different hyperparameter configuration samples; instead, each sample is drawn independently of the rest. This property makes it possible to learn several configuration trials in parallel but incurs the risk of wasting time on exploring poorly performing configurations. This is the case of Grid Search (GS) [55,56] and Random Search (RS) [57]. GS [55,56] is a simple approach that takes a finite set of values for each hyperparameter and computes the Cartesian product of them to configure a grid of trials to be checked. RS [57] randomly draws a predefined number of trials according to certain distribution following a Monte Carlo technique. Unlike those methods that sample each configuration trial independently, others include a guided search involving a model evaluation of the current sample in order to draw the next, as in Bayesian Optimization (BO) [58], Particle Swarm Optimization (PSO) [59] and Hyperband (HB) [60]. BO [58] is a well-known and successful optimization approach [61] that performs a balance between exploration (taking other hyperparameter values) and exploitation (taking information from the hyperparameters already explored) in order to avoid falling into a local minimum. PSO [59] is a population-based method that simulates a biological behaviour among particles that has also been successfully applied in HPO [62]. The particles in PSO just cooperate rather than mutate or crossover. This provides information to guide the search, but it must be properly initialized to minimize the risk of leading to a local rather than to a global optimum. HB [60] is a bandit-based technique that improves on the successive halving method. It does so by dynamically choosing hyperparameter configurations in an attempt to establish a trade-off between the number of configurations and the available resources (such as time). This way, half of the poorly performing configurations are eliminated each time, while the other half are kept.

Nowadays, HPO is one of the core parts of the AutoML frameworks [63]. While HPO tries only to provide a predictive model by optimizing their hyperparameters, AutoML goes further and does something more than this. Particularly, AutoML covers solving all the tasks a researcher must tackle,obtaining a final solution from the data, trying to avoid requiring expertise assistance. Even though this task includes pre-processing, feature selection and extraction before the predictive model is induced it also calls for interpretability and decision making after the predictive model is induced. Combined Algorithms Selection and Hyperparameter Optimization (CASH) [64] is currently a top field of research that goes a step beyond the task of HPO as well, since it also selects a suitable system that provides the model in addition to the hyperparameters. However, CASH environments do not automate as many tasks as an AutoML framework. In fact, it is quite common for AutoML frameworks to be built over a CASH environment.

Some AutoML frameworks have been proposed in the literature. For instance, Auto-Weka [18,19] is an AutoML environment built on top of WEKA models. Auto Tuned Models (ATM) [37], Auto-Sklearn [14,15] and Tree-based Pipeline Optimization Tool (TPOT) [40] are frameworks that use the scikit-learn library [65]. Hyperopt-sklearn [38,39] is based on Auto-Weka applied to scikit-learn. Some frameworks focus specifically on neural networks such as Auto-Pytorch [16] and TPOT-NN [41] (a particular version of TPOT). Auto-Gluon [13] successfully includes a multi-layer combination of models for image, text, time series, and tabular data. H2O [20,21] is an open source, in memory, distributed, fast and scalable commercial platform also suitable to be managed by non-experts in machine learning. Another recent AutoML framework is MANGO [42], which is an open-source Python library able to parallelize HPO on a distributed cluster. Syne-Tune [43] is an open-source Python library as well, but for large-scale distributed hyperparameter and neural architecture optimization. Also, Ray-Tune [46] is specifically designed for distributed model selection. Additionally, ASHA [48] proposes an asynchronous successive halving algorithm in order to improve the efficiency for numerous parallel evaluations. Even more recently, Hyper-Tune [44] has included improvements in regard to optimizing the BO, such as automatic resource allocation, asynchronous scheduling and multi-fidelity optimizer. MFest-HB [49] proposes a new sampling strategy including multi-fidelity learning to HB sampling strategy, which improves the Bayesian Optimization and Hyperband (BOHB) [51]. Finally, other systems are Google Vizier [45] and OpenBox [47], which both include transfer learning and early stopping to improve the hyperparameter search, but Google Vizier only supports traditional black-box optimizations, whereas OpenBox can cope with multiple objectives and constraints. The fact is that all of these AutoML systems focus on improving the configuration trials generation or on parallelizing or distributing the computations, and only some of them include ensemble after learning the models with the different generated configuration trials. Particularly, only Auto-Gluon, Auto-Pytorch and Auto-Sklearn include the Caruana method for ensemble, while only Auto-Weka

and H2O allow the possibility of performing stacking, but without any advice about which meta-learner is adequate to use. Concerning CASH environments, Auto-Sklearn [14,15] and Hyperopt-sklearn [38,39] also deal with CASH for supervised machine learning. Optunity [50], Bayesian Optimization and Hyperband (BOHB) [51], Sequential Model-based Algorithm Configuration (SMAC) [52], Robust Bayesian Optimization (RoBo) [53], Bayesian Tuning and Bandits (BTB) [54] are other popular CASH frameworks. Table 1 summarizes the sampling and ensemble strategies supported by all these AutoML and CASH environments.

3. Ensemble in Hyperparameter Optimization

A general definition of ensemble learning that covers supervised (classification and regression) and unsupervised learning can be the process of integrating a set of models in order to provide a final prediction [3,25]. Formally, this integration process $\mathscr{I}$ can be defined for a given instance x as

$$f(x) = \mathscr{I}\left(f_1(x), \ldots, f_p(x)\right)$$

where p is the number of models to ensemble, $\{f_i(x)\}_{i=1}^{p}$ is the set of models to ensemble and $f(x)$ is the function obtained after the integration process.

The integration process is commonly assumed to involve a linear combination or fusion of the individual models. Hence, $f(x)$ can be rewritten as

$$f(x) = \sum_{i=1}^{p} h_i(x) \cdot f_i(x)$$

where $\{h_i(x)\}_{i=1}^{p}$ is a set of functions that grant weights to the individual models $\{f_i(x)\}_{i=1}^{p}$.

Diversity is a key issue in ensemble learning [66]. However, among the existing kinds of diversity, hyperparameter diversity is the one that fits HPO, since in HPO different hyperparameter configuration trials are the ones that provide the different models to ensemble [25].

The common practice establishes the weighting functions $\{h_i(x)\}_{i=1}^{p}$ as constants, that is, $\{h_i(x)\}_{i=1}^{p} = \{\alpha_i\}_{i=1}^{p}$. Non-constant weighting functions have also been studied. Static methods (methods that define non-constant weighting functions during learning) either split the input space by assigning models to predefined regions [67] or perform static selection defining areas of expertise for the models [68]. dynamic methods (methods that define non-constant weighting functions in prediction time), on the other hand, do it by searching for similar instances in the training set, typically via k-nearest neighbour approaches [69]. In any case, both types require hyperparameter tuning, for subsampling in the case of the static methods and for the k-nearest neighbour based approaches in the case of dynamic methods.

Hence, this paper will focus solely on constant-weighting functions, and only on those that do not require adjusting hyperparameters within them, since one of the goals of this paper is precisely to propose a non-hyperparametric ensemble procedure. Section 3.1 reviews existing ensemble strategies, including stacking. Section 3.2 discusses meta-learners for stacking ensemble, for which a stop criterion must be stated. Finally, Section 3.3 makes a review and discussion of existing stop criteria for meta-learners in stacking ensemble.

3.1. Review of existing ensemble strategies in Hyperparameter Optimization

BEM [9] has been employed for ensemble in HPO and provides constant-weighting functions without tuning hyperparameters, just by computing the simple average of the individual predictions $\{f_i(x)\}_{i=1}^{p}$. Hence, the $\{\alpha_i\}_{i=1}^{p}$ are all equal to the constant $1/p$ for all $i = 1, \ldots, p$. The ensemble function is then $f(x) = \sum_{i=1}^{p} \frac{1}{p} \cdot f_i(x) = \frac{1}{p} \cdot \sum_{i=1}^{p} f_i(x)$. The IEW strategy [70], which consists of establishing the weights $\{\alpha_i\}_{i=1}^{p}$ as inversely proportional to the expected error of $\{f_i(x)\}_{i=1}^{p}$, has also been employed as ensemble method in HPO and also provides constant-weighting functions free of hyperparameter tuning. The Caruana strategy [12] is an appealing approach that goes along the same line as the previous methods, and has shown promising results recently and has been included in some of the few AutoML systems that provide ensemble in their frameworks [13,14,16]. It differs from the above-mentioned methods in that it performs an ensemble selection first, or, in other words, it establishes some weights to be zero beforehand. More in detail, first the best models that will not be weighted by zero are selected with replacement and the simple average is then computed. As a result, the weights of each model depend on the number of times the model was selected.

These strategies do not require hyperparameter tuning. Futhermore, multicollinearity does not affect them. In the case of BEM, this is so because the weights are constant and chosen independently of the prediction values. In the case of IEW, the weight for each prediction only depends on the prediction of this model, hence, it is chosen independently of the rest. Finally, in the Caruana strategy zero weight is implicitly assigned to some predictions, namely, to those that are not involved in the selection under a replacement procedure. However, these ensemble methods, do not include a learning process that may add generalization power to the ensemble. Bagging, boosting and stacking [23] are typical ensemble strategies that include a learning procedure in the process, have been widely used for many applications [71] and have been recently stated as the most promising kind of ensemble approaches regarding data, algorithm or output level manipulation approaches [22]. Among the three strategies, stacking is the only one suitable to be applied in HPO and has been included in some of the few AutoML systems that use ensemble in their frameworks. Bagging and boosting are not suitable for ensemble in HPO, since models for these approaches are dynamically generated, whereas the models in HPO are learned beforehand according to the range of hyperparameter configuration trials. Apart from the conventional methods for ensemble (bagging, boosting and stacking), other ensemble methods specifically designed for certain situations are available [25]. This is the case of decomposition based methods, typically adopted for time series datasets, which can be classified into divide-and-conquer and hierarchical ensemble methods. The main concept is to decompose the time series into a collection of time series motivated by its seasonal properties. Hence, these methods are not applicable to general-purpose datasets. There are also multi-output optimization ensemble methods, which try to optimize several performance measures and typically adopt evolutionary algorithms to find the Pareto front of the individual models. We discard these methods, since optimizing several performance measure falls out of the scope of this work. Besides, evolutionary algorithms have more than plenty of hyperparameters to tune. Negative correlation ensemble method has been specifically designed for neural networks, where all the individual models are trained simultaneously using penalty terms in the respective error functions. This method is not applicable in our context, since the individual models are trained taking into account the configuration trials that the sampling strategies generate. Deep learning and multiple kernel learning based ensemble methods are also available, but they typically require tuning a considerable number of hyperparameters.

Let us now focus on stacking ensemble. Stacking [17], also called stacked ensemble, stacked regression or superlearning, aims to find an optimal combination of the models $\{f_i(x)\}_{i=1}^{p}$, while providing constant-weights, but including a learning process in the ensemble, typically known as a second-level meta-learner. The inclusion of this

Table 1
Summary of sampling and ensemble strategies for the main AutoML and CASH frameworks

	Framework	Sampling	Ensemble
AutoML	ATM [37]	BO, multi-armed bandit	-
	Auto-Gluon [13]	-	Caruana [12]
	Auto-Pytorch [16]	BO, HB	Caruana [12]
	Auto-Sklearn [14,15]	BO, Successive Halving	Caruana [12]
	Auto-Weka [18,19]	BO	Stacking [17]
	Hyperopt-Sklearn [38,39]	BO	-
	H2O [20,21]	GS, RS	Stacking [17]
	TPOT [40]	Genetic Algorithm	-
	TPOT-NN [41]	Genetic Algorithm	-
	MANGO [42]	BO	-
	Syne-Tune [43]	BO, HB, Population-based	-
	Hyper-Tune [44]	Improved BO	-
	Google Vizier [45]	BGPB[a], others	-
	Ray-Tune [46]	GS, RS, BO, HB Blend-search, BO Dragonfly	-
	OpenBox [47]	BO, PRF[b]	-
	ASHA [48]	Succesive Halving	-
	MFest-HB [49]	BO, HB	-
	Framework	**Sampling**	**Ensemble**
CASH	Optunity [50]	PSO	-
	BOHB [51]	Bayesian Hyperband	-
	SMAC [52]	Bayesian	-
	RoBo [53]	Bayesian	-
	BTB [54]	Bayesian Multi-armed Bandit	-

[a] Batched Gaussian Process Bandits
[b] Probabilistic Random Forest

meta-learner provides the ensemble strategy with a promising generalization power. However, the main drawback is establishing an adequate meta-learner for HPO, since it may be affected by the existing multicollinearity. Currently, the literature does not provide advice on this issue. In fact, in a very recent survey of ensemble methods [22], it is stated that stacking has not been extensively studied so far and it is suggested as a future research line. Furthermore, this meta-learner may have hyperparameters to be tuned, as it is the case with Ridge [34], SVR [35] or RFR [36], in order to avoid, or at least mitigate, the problems derived from the multicollinearity. An alternative could consist of using classical OLS, which is highly affected by multicollinearity, or even, classical OLS with constraints such as the weights to be positive ($\{\alpha_i \geqslant 0\}_{i=1}^{p}$) and sum to one ($\sum_{i=1}^{p}\alpha_i = 1$) in order to express the generalized error, which leads to the method called GEM [9]. GEM also encounters multicollinearity, but it deals with it by imposing the constraints of the weights to be positive and sum to one. In fact, it is one of the methods that has been recently applied to ensemble in HPO [10]. Section 3.2 discusses possible meta-learners for stacking ensemble.

3.2. Discussing possible meta-learners for stacking ensemble in Hyperparameter Optimization

As commented before, there is no advise in the literature about which meta-learners may be suited to ensemble stacking and, in particular, to HPO ensemble stacking. Therefore, some possibilities are exposed in this section. Apart from OLS [28] and GEM, which are highly affected by the multicollinearity problem, and discarding methods that require to tune hyperparameters, FSR [30], PCR [31], PLS [32] and BOOST [33] remain possible meta-learners for stacking ensemble. At this point, we shall clearly state that BOOST is only used here as a meta-learner for stacking ensemble, and not as an HPO ensemble [23].

FSR involves starting with no features in the model, testing the addition of one single feature at a time, using a chosen model fit criterion that adds the feature (if any) that contributes the most statistically significant improvement of the fit, and repeating this process until a stop criterion is satisfied. If every feature is included, then FSR becomes OLS. FSR has been recently adopted for ensemble in HPO [72]. However, this work does not focus on the ensemble itself; it just adopts FSR for ensemble and takes the number of iterations as a stop criterion, hence, adding an additional hyperparameter. The work focuses on deep neural networks and on the claims noticeable benefit when combining (ensembling) different hyperparameter values (coming from a RS sampling strategy) together with different possible initializations of the deep neural network.

PCR computes the so-called principal components, which are the eigenvectors of the covariate matrix, which in turn are the directions of the axes of the most variance, and hence, provide the most information. The principal components are uncorrelated and the information of the original features is expected to be squeezed or compressed into the first components. In this way, the first principal component accounts for the largest possible variance, and consequently, for the most information. The second principal component is uncorrelated with the first principal component and accounts for the next highest variance, and so on until a stop criterion is satisfied. Finally, a regression, typically using an OLS, is performed using the first principal components as features instead of using the original features. Despite this strategy may seem to reduce the multicollinearity because the regression is performed over uncorrelated and transformed features, the main drawback is that no information about the target is taken, so the components are taken in an unsupervised way. Therefore, there is no guarantee on whether the principal components will be related to the target. In this sense, PLS and PCR work similarly, but in PLS the principal components try to extract those features that explain as much as possible the covariance between the features and target, rather than the variance between the features. Then, unlike PCR, PLS takes into account the relationship between the features and the target, making the principal components closer to the target. This property makes PLS a stronger meta-learner than PCR.

Finally, BOOST works quite differently from FSR, PCR and PLS, since BOOST performs a regression using just one feature each time, therefore, completely removing the multicollinearity problem. This is a promising property, not only because it makes BOOST free of multicollinearity, more importantly it allows including regularization

strategies and stop criteria that involve just one feature. Section 4 formally explains the BOOST strategy. It also details our novel proposal RBOOST as a meta-learner in HPO stacking ensemble, which consists of adding an implicit regularization in BOOST (see Section 4.2). Finally, our novel stop criterion ICM is exhaustively and specifically built for BOOST and RBOOST (see Section 4.1).

3.3. Discussing several possible stop criteria for the meta-learners for stacking ensemble in Hyperparameter Optimization

All FSR, PCR, PLS and BOOST require a stop criterion as an alternative to the number of iterations (features), or, in the case of PCR and PLS, the number of principal components. Several non-hyperparametric stop criteria are specifically available in the literature for FSR [73], PCR [74] and PLS [75]. The process works as follows. A stop criterion of this kind is computed for all possible values of the number of features (in the case of FSR) or number of components (in the case of PCR or PLS). The possible values in both cases will range from 1 to p, where p is the number of models involved in the ensemble. Then, the best features from FSR, (or components using PCR or PLS) are obtained for each of these possible values of the number of features (or components). Next, an OLS is carried out over these features (or components), which yields a performance score. After that, the stop criterion is computed from the number of features or components, the features or components themselves and the performance score. Finally, the best option will be the number of features or components that provides the best stop criterion value. These stop criteria are Akaike Information Criterion (AIC) [76], Akaike Information Criterion corrected (AICc) [77], Bayesian Information Criterion (BIC) [78], Hannan-Quinn Information Criterion (HQIC) [79] and generalized Minimum Description Length (gMDL) [80]. AIC determines the relative information value of the model using the maximum likelihood estimation and the number of features. The best-fit model according to AIC is the one that explains the greatest amount of variation using the fewest possible features. AICc introduces a correction into AIC in order to avoid overfitting when the number of instances is small in comparison with the number of features. BIC is similar to AIC, but it penalizes more aggressively the number of instances. HQIC introduces a correction over BIC to smooth the influence of the number of instances. Finally, gMDL combines AIC and BIC and tries to adaptively select the best between the two. All these stop criteria are designed for general purpose regression rather than for HPO ensemble. Specifically, they penalize the number of features and tend to discard adding new features if the prediction performance hardly improves. However, one of the challenges of HPO ensemble is to include in the final model the maximum information contained in the models trained with the variety of hyperparameter configuration trials, even if the performance with fewer models may be accurate enough. In this sense, our proposed stop criterion ICM does not penalize the number of features and therefore allows including information coming from both the previous or new features until an overfitting situation is detected. Section 4.1 exposes in detail how this novel stop criterion was deduced.

3.4. Overall process of ensemble in Hyperparameter Optimization

This section summarizes the whole process of HPO with ensemble. Particularly, Fig. 1 illustrates the integration of the sampling strategies mentioned in Section 2 (GS, RS, BO, PSO and HB). These sampling strategies generate several configuration trials $\left\{\lambda_1^i,\ldots,\lambda_j^i,\ldots,\lambda_h^i\right\}_{i=1}^{p}$ for the hyperparameters $\lambda_1,\ldots,\lambda_j,\ldots,\lambda_h$ of certain base-learners (Ridge, SVR and RFR). The base-learners are trained from an (X_{TR},Y_{TR}) dataset taking into account these configuration trials $\left\{\lambda_1^i,\ldots,\lambda_j^i,\ldots,\lambda_h^i\right\}_{i=1}^{p}$, leading to the set of models $\{f_i\}_{i=1}^{p}$. Then, the models $\{f_i\}_{i=1}^{p}$ are applied to X_{TR} (typically using a cross validation strategy) to provide the set of predictions $\widehat{Y}_{TR}=\left\{\widehat{Y}_{TR}^i\right\}_{i=1}^{p}$. Hence, the data set $\left(\widehat{Y}_{TR},Y_{TR}\right)$ feeds an ensemble strategy outlined in Section 3.1 (BEM, IEW, Caruana and staking ensemble). As far as Best is concerned, it applies the classical procedure in HPO of choosing the best model from $\{f_i\}_{i=1}^{p}$ according to a loss function. Focusing on staking ensemble, several possible meta-learners discussed in Section 3.2 (OLS, GEM, FSR, PCR, PLS, BOOST and our novel proposal RBOOST, built on the basis of BOOST) can be applied. Concerning FSR, PCR, PLS, BOOST and even RBOOST, a non-hyperparametric stop criterion must be established in order for the meta-learner to be non-hyperparametric. This stop criterion can be one of those presented in Section 3.3 (AIC, AICc, BIC, HQIC, gMDL and of course our novel stop criterion ICM). Finally, an ensemble model f_e is induced. As a result, the overall model f is formed by the $\{f_i\}_{i=1}^{p}$ models induced by the different configuration trials and the ensemble model f_e. Hence a test dataset X_{TE} is applied to the configuration trial models $\{f_i\}_{i=1}^{p}$, whose predictions are ensembled using the ensemble model f_e providing the prediction $\widehat{Y}_{TE}^{e}$ for X_{TE}.

4. Boosting as meta-learner for stacking ensemble in HPO

This section discusses BOOST and the proposed implicit regularization for BOOST, which leads to the RBOOST method as a meta-learner in stacking ensemble (see Section 4.2). Additionally, this section exposes in detail how the novel stop criterion ICM, specifically designed for BOOST and RBOOST, is derived (see Section 4.1). In fact, it is not applicable to other meta-learners.

We shall begin with detailing BOOST. Let $\mathscr{F}=\left\{f^i(x)\right\}_{i=1}^{p}$ be the set of p features that describe the predictions provided by the model induced from the different hyperparameter configuration trials. Initially, the set of features $\mathscr{S}$ for the ensemble is empty, that is, $\mathscr{S}^{(0)}=\varnothing$, since the algorithm follows a forward-search strategy. In each stage j, a set of regression procedures involving one single feature is performed: one regression per feature in $\mathscr{F}$ (a feature is selected with replacement). A feature $f^{i^*,(j)}(x)$ of $\mathscr{F}$ is selected in stage j according to a certain criterion in terms of a loss function $\mathscr{L}$ and included in $\mathscr{S}$, that is, $\mathscr{S}^{(j)}=\mathscr{S}^{(j-1)}\cup\left\{f^{i^*,(j)}(x)\right\}$. The target for performing the set of regressions in each stage remains constant for the set of regressions performed in each stage, although it does vary from one stage to another. Hence, the bias is also corrected from one stage to another. In the first stage, the target for the set of regressions is the original one, that is, $r^{(0)}=y$. Then, in each stage, the target for the next stage $r^{(j)}$ is computed as the difference between current stage's target $r^{(j-1)}$ and the prediction performed using the regression model that was induced with the feature selected in the actual stage $h_{f^{i^*,(j)}(x)}\left(f^{i^*,(j)}(x)\right)$. More in detail, the set of regressions performed in certain stage j is

$$\left\{h_{f^i(x)}^{(j)}\left(f^i(x)\right)=r^{(j-1)}:f^i(x)\in\mathscr{F}\right\}$$

where the initial residual is $r^{(0)}=y$ and the residual in stage j is defined in terms of the selected feature $f^{i^*,(j)}(x)$ in stage j, that is, $r^{(j)}=r^{(j-1)}-h_{f^{i^*,(j)}(x)}\left(f^{i^*,(j)}(x)\right)$. The process continues until the stop criterion is satisfied. Consequently, BOOST builds a family of functions $\left\{g^{(j)}(f(x))\right\}_{j=1}^{\cdots}$ in a stage-wise rather than in a step-wise procedure, such as

$$g^{(j)}(f(x))=g^{(j-1)}(f(x))+h^{(j)}(f(x))$$

where $g^{(0)}(f(x))=0$.

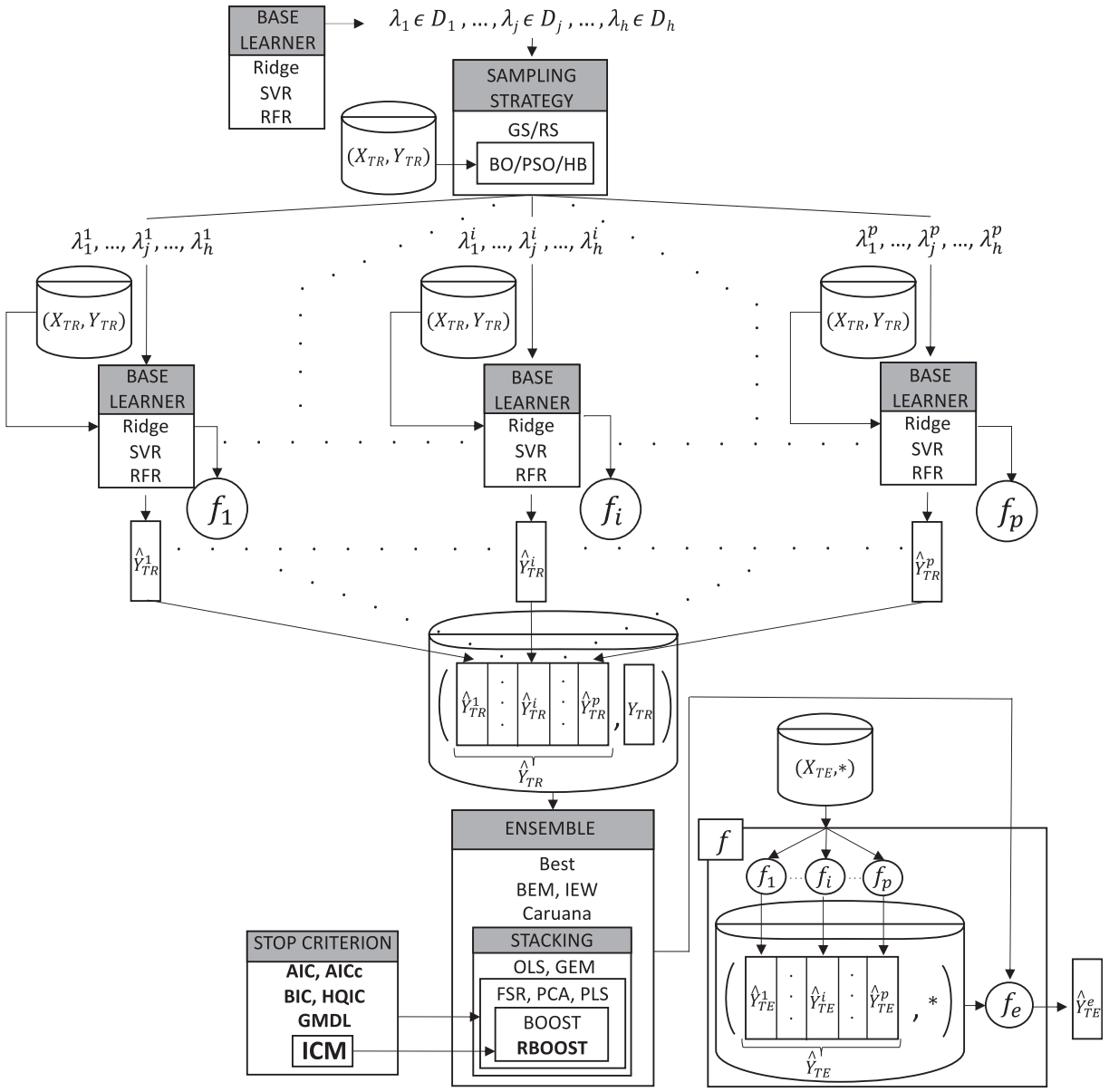


Fig. 1. Overall scheme of the process. A sampling strategy (GS, RS, BO, PSO or HB) generates several hyperparameter configurations for inducing several models using a base-learner (Ridge, SVR or RFR). These models are aggregated using an ensemble method (Best, BEM, IEW, Caruana or stacking). In the case of stacking ensemble, a meta-learner (OLS, GEM, FSR, PCR, PLS, BOOST or RBOOST) is required, and some require a stop criterion (AIC, AICc,BIC, HQIC, gMDL or ICM).

Considering linear regression, the model induced in each stage takes the form

$$h_{f^{i^{*,(j)}}(x)}\left(f^{i^{*,(j)}}(x)\right) = \alpha_{i^{*,(j)}}^{(j)} \cdot f^{i^{*,(j)}}(x) + \beta_{i^{*,(j)}}^{(j)}$$

where $\alpha_{i^{*,(j)}}^{(j)}$ and $\beta_{i^{*,(j)}}^{(j)}$ are the regression coefficients for the $f^{i^{*,(j)}}(x)$ feature taken in stage j. The features in $\mathscr{S}$ after the process ends will be the features taken in the ensemble with the weights $\alpha_{i^{*,(j)}}^{(j)}$ successively computed in the process. As commented before, the process selects adequate features in each stage with replacement, which means that a selected feature in a certain stage might be chosen again in successive stages. The respective weights $\alpha_{i^{*,(j)}}^{(j)}$ for this kind of features are accumulated to provide a unique weight to the feature.

$$\alpha_{f^{i^*}(x)} = \sum_{\substack{f^{i^{*,(j)}}(x) \in \mathscr{S} \\ f^{i^{*,(j)}}(x) = f^{i^*}(x)}} \alpha_{i^{*,(j)}}^{(j)}$$

Bias is included in the procedure since it is not possible to ensure that the features are meaningfully unbiased. All the bias $\beta_{i^{*,(j)}}^{(j)}$ of the regression in each stage are also successively accumulated. Then,

$$\beta = \sum_{f^{i^{*,(j)}}(x) \in \mathscr{S}} \beta_{i^{*,(j)}}^{(j)}$$

Therefore, including the bias in the procedure implicitly alters the expression of the ensemble when compared to the one displayed in the previous section. The new expression for the ensemble will be

$$f(x) = \sum_{f^{i^*}(x) \in \mathscr{S}^*} \alpha_{f^{i^*}} \cdot f^{i^*}(x) + \beta$$

where $\mathscr{S}^*$ is the set of selected features, where $f^{i^*}(x)$ represents a different feature (without replacement).

Algorithm 1: BOOST for stacking ensemble

Algorithm 1 BOOST for stacking ensemble

1: **function** BOOSTING
Input: $\{f^i(x)\}_{i=1}^p$ features of the ensemble (model predictions), y the original target, $\mathcal{L}$ a loss function
Output: $\mathcal{S}$, $\mathcal{A}$, β
2: $\mathcal{A} \leftarrow \emptyset$ ▷ $\mathcal{A}$ set of weights for the features (models) from $\{f^i(x)\}_{i=1}^p$ selected by BOOST
3: $\mathcal{S} \leftarrow \emptyset$ ▷ $\mathcal{S}$ set of features (models) from $\{f^i(x)\}_{i=1}^p$ selected by BOOST
4: $\beta \leftarrow 0$ ▷ Bias of the ensemble obtained by BOOST
5: $r^{(0)} \leftarrow y$ ▷ The initial residual in BOOST is the original target y
6: **while** true **do**
7: $i^{*,(j)} \leftarrow \arg \min_{\{i:f^i(x)\in\mathcal{F}\}} \{l_i^{(j)} : [\alpha_i^{(j)}, \beta_i^{(j)}, l_i^{(j)}] \leftarrow \mathrm{OLS}(f^i(x), r^{(j-1)}, \mathcal{L})\}$
▷ BOOST selects the feature with the lowest value of the loss function $\mathcal{L}$, performing an OLS of each feature isolated from the rest.
8: **if** stop_criterion **then** ▷ Stop criterion: AIC, AICc, BIC, HQIC, gMDL or ICM
9: **break**
10: **end if**
11: $r^{(j)} \leftarrow r^{(j-1)} - (\alpha_{i^{*,(j)}}^{(j)} \cdot f^{i^{*,(j)}}(x) + \beta_{i^{*,(j)}}^{(j)})$
▷ The residual (the target for the next stage) is updated
12: $\mathcal{S} \leftarrow \mathcal{S} \cup \{f^{i^{*,(j)}}(x)\}$ ▷ The selected feature in each stage is added to $\mathcal{S}$
13: $\mathcal{A} \leftarrow \mathcal{A} \cup \{\alpha_{i^{*,(j)}}^{(j)}\}$ ▷ The weight for the selected feature is added to $\mathcal{A}$
14: $\beta \leftarrow \beta + \beta_{i^{*,(j)}}^{(j)}$ ▷ The bias is updated
15: **end while**
16: **return** $\mathcal{S}$, $\mathcal{A}$, β ▷ The stop criterion is satisfied and $\mathcal{S}$, $\mathcal{A}$ and β are returned
17: **end function**

Choosing the feature in each stage, the regressor and the loss function. The criterion applied in order to select a feature in each stage is defined in terms of a loss function $\mathscr{L}$, which will be the same one to be optimized in the HPO process. Hence, the criterion adopted will be the usual one: choosing the feature that produces the lowest value for this loss function [33]. This criterion is expressed as:

$$f_{i^{*,(j)}}^{(j)}(x) = \arg\min_{f(x)\in\mathscr{F}} \mathscr{L}\left(r^{(j-1)}, \alpha_f^{(j)} \cdot f(x) + \beta_f^{(j)}\right)$$

Regarding the regressor employed in the process, OLS is now adequate, since i) it has no hyperparameters to tune and ii) the regression is performed over just one feature each time, so that, the problems derived from multicollinearity disappear. Consequantly, the loss function $\mathscr{L}$ to minimize will be the typical squared-error L_2 loss function [81].

The algorithm. Algorithm 1 displays the pseudocode of the BOOST procedure. Only one feature is involved in each stage (see the first argument of the call to the OLS function in line 7). Hence, just one α-coefficient is provided in each stage. In the end, every α-coefficient computed in each stage is returned (see lines 13 and 16). The values of the β bias computed in all the stages, are added up in order to obtain the final value (see lines 14 and 16). Also, the target varies from one stage to another (see line 5 for the initial target, the second argument of the call to the OLS function in line 7, and the target is updated for the next stage in line 11).

Let us now discuss the novel stop criterion and the proposed implicit regularization included in the process.

4.1. Increasing Coefficient Magnitude as stop criterion

Concerning the stop criterion, when the selected feature in a given stage, despite being the most highly correlated to the target in said stage, is even poorly correlated to the target, the weight of this feature approaches to zero and therefore the residual of this stage will be close to the that of the previous stage. This fact opened the door to including a heuristic to establish a stop crite-

Table 2
Number of instances and features for UCI repository datasets[a]

Dataset	Inst.	Feat.	Dataset	Inst.	Feat.
Abalone	4177	11	Forest	517	13
Airfoil Self Noise	1503	6	Qsar	908	7
Auto MPG	392	8	Servo	167	5
Automobile	158	26	Slump	103	8
Concrete Data	1030	9	Traffic	135	18
Com. and crime	1993	119	Red wine quality	1599	12
Fertility	100	10	White wine quality	4898	12
Flow	103	8			

[a] Flow and Slump refer to the two outputs of the so-called Slump dataset of the UCI repository for multioutput regression. Notice that both have the same number of instances and features.

rion. One may suggest stopping when the loss estimation increases from one stage to another. But this never take place with OLS as the regressor and when L_2 function is taken as the loss function. When this is the case, the loss estimation is always dismissed from one stage to another, since OLS obtains the linear function with precisely the minimum L_2 function value. Instead, our proposal consists of establishing a stop criterion in terms of the selected feature coefficients. The existing stop criteria AIC, AICc, BIC, HQIC and gMDL basically depend on the error value, the number of features and the number of examples. But in this context, the number of examples is a constant. This means that only the error value and the number of features have any influence. In the case of FSR, PCR and PLS, a different feature is added in each iteration. Hence, the error value is the element that conditions when the algorithm stops. Clearly, a feature is highly relevant if the error considerably diminishes. However, if that error is just reduced slightly, these stop criteria may not be able to distinguish whether a feature provides promising information or not, causing an overfitting situation. The same happens with BOOST and RBOOST, but an additional issue affects these ensemble strategies. In fact, both BOOST and RBOOST are capable of taking the same feature more than once. Then, if a feature chosen in a certain stage has already been chosen in a previous stage, these stop criteria will always admit this choice because i) since the feature is not a new one the number of features remains constant and, as stated before, ii) the error value always decreases from one stage to the next. Hence, these stop criteria are not useful under this particular situation. In this sense, the fact that our novel stop criterion includes the selected feature's coefficient proves useful when the feature is selected more than once. This is so because the coefficients vary greatly with each selection of the same feature.

Let us now deduce the novel stop criterion. In OLS, the coefficient of the regression α can be expressed in terms of the feature and target standard deviations $\sigma_{f(x)}$ and σ_y as [82]

$$\alpha = R \cdot \sigma_y / \sigma_{f(x)}$$

where R^2 is the correlation coefficient or coefficient of determination. It represent the share of the variation of y that can be explained through the regression model, and it also satisfies $R^2 = \sigma^2_{f(x),y}$. Taking into account the ANOVA decomposition, the total variability SST is the sum of the variability associated with the model SSM and the variability of the residuals SSR, that is, $SST = SSM + SSR$, where SST, SSM and SSR are expressed as follows

$$SST = \sum_{i=1}^{n}(y_i - \overline{y})^2 \quad SSM = \sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2 \quad SSR = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i are the actual target values, $\hat{y}_i$ are the predictions, $\overline{y}$ is the target value average, and n is the number of instances.

Then, SSR/SST is the proportion of the variation in the target that is not explained by the regression model. Therefore, R^2 can be expressed as

$$R^2 = SSM/SST = 1 - SSR/SST$$

Turning back to the expression of α in terms of the correlation coefficient, one can state that

$$\alpha \cdot \sigma_{f(x)} = R \cdot \sigma_y = \pm\sqrt{\left(1 - \frac{SSR}{SST}\right) \cdot \sigma_y^2}$$

Since $\sigma_y^2 = SST/n$, then

$$\alpha \cdot \sigma_{f(x)} = \pm\sqrt{\left(1 - \frac{SSR}{SST}\right) \cdot \frac{SST}{n}}$$

Taking into account that $SST \neq 0$, then

$$\alpha \cdot \sigma_{f(x)} = \pm\sqrt{\frac{SST - SSR}{n}}$$

Let us notice that SST is invariant regardless of the regression model, since SST is the variability contained in the data. Besides, n is also constant. Considering that SSR is positive, the expression $|\alpha| \cdot \sigma_{f(x)}$ ($\sigma_{f(x)}$ is always positive) is maximum when SSR is minimum. Therefore, as BOOST gets the minimum value for SSR, it also obtains the maximum value for $|\alpha| \cdot \sigma_{f(x)}$. This means that $|\alpha| \cdot \sigma_{f(x)}$ in stage $j-1$ for the selected feature in stage $j-1$ ($f^{i^{*,(j-1)}}(x)$) is greater than or equal to $|\alpha| \cdot \sigma_{f(x)}$ for the rest of the features, including the selected feature in stage j ($f^{i^{*,(j)}}(x)$); otherwise, the selected feature in stage $j-1$ will be a different feature from $f^{i^{*,(j-1)}}(x)$, for instance, the one selected in stage j ($f^{i^{*,(j)}}(x)$). Then, $|\alpha^{(j-1)}_{i^{*,(j-1)}}| \cdot \sigma_{f^{i^{*,(j-1)}}(x)} \geqslant |\alpha^{(j-1)}_{i^{*,(j)}}| \cdot \sigma_{f^{i^{*,(j)}}(x)}$ [1]

On the other hand, $|\alpha| \cdot \sigma_{f(x)}$ typically decreases from one stage to the next. This is because the variability of the residuals (SSR) always decreases in each stage, since the successive target values contain less information attributable for features as the algorithm

[1] Notice that $\alpha^{(j-1)}_{i^{*,(j)}}$ is the coefficient of the regression performed in stage $j-1$ for the feature selected in stage j, which may not be the best option in stage $j-1$.

progresses. Conversely, an increase of $|\alpha| \cdot \sigma_{f(x)}$ from one stage to another is a sign of a poor *SSR* decrease. If this situation takes place, then $|\alpha| \cdot \sigma_{f(x)}$ in stage j for the selected feature in stage j ($f^{i^{*,(j)}}(x)$) is greater than or equal to $|\alpha| \cdot \sigma_{f(x)}$ in stage $j-1$ for the selected feature in stage $j-1$ ($f^{i^{*,(j-1)}}(x)$), that is

$$|\alpha_{i^{*,(j)}}^{(j)}| \cdot \sigma_{f^{i^{*,(j)}}(x)} > |\alpha_{i^{*,(j-1)}}^{(j-1)}| \cdot \sigma_{f^{i^{*,(j-1)}}(x)}$$

Therefore, taking into account the previous inequality leads us to the following one

$$|\alpha_{i^{*,(j)}}^{(j)}| \cdot \sigma_{f^{i^{*,(j)}}(x)} > |\alpha_{i^{*,(j-1)}}^{(j-1)}| \cdot \sigma_{f^{i^{*,(j-1)}}(x)} \geqslant |\alpha_{i^{*,(j)}}^{(j-1)}| \cdot \sigma_{f^{i^{*,(j)}}(x)}$$

Hence,

$$|\alpha_{i^{*,(j)}}^{(j)}| \cdot \sigma_{f^{i^{*,(j)}}(x)} > |\alpha_{i^{*,(j)}}^{(j-1)}| \cdot \sigma_{f^{i^{*,(j)}}(x)}$$

Now, and since $\sigma_{f^{i^{*,(j)}}(x)}$ is positive, the following inequality holds:

$$|\alpha_{i^{*,(j)}}^{(j)}| > |\alpha_{i^{*,(j)}}^{(j-1)}|$$

This means that the coefficient of the same feature in the current stage is greater than its coefficient in the previous stage. At this point, one can only wonder why this feature takes this greater value in the current stage and has not in the previous stage. Hence, the fact that the influence of a feature increases from one stage to another, and taking into account that *SSR* always decreases (sometimes poorly) from one stage to another can be interpreted as a sign that the model overfits the data. Consequently, the proposed stop criterion ICM, aims precisely to prevent this kind of situation. As a result, the algorithm will stop when the following inequality holds

$$|\alpha_{i^{*,(j)}}^{(j)}| \cdot \sigma_{f^{i^{*,(j)}}(x)} > |\alpha_{i^{*,(j-1)}}^{(j-1)}| \cdot \sigma_{f^{i^{*,(j-1)}}(x)}$$

Given that the typical deviations $\{\sigma_f\}_{f \in \mathcal{F}}$ remain constant during the BOOST process, ICM can be expressed as

$$ICM_{\{\sigma_{f(x)}\}_{f(x)\in\mathcal{F}}}\left(\alpha_{i^{*,(j)}}^{(j)}, \alpha_{i^{*,(j-1)}}^{(j-1)}\right) \equiv \left[|\alpha_{i^{*,(j)}}^{(j)}| \cdot \sigma_{f^{i^{*,(j)}}(x)} > |\alpha_{i^{*,(j-1)}}^{(j-1)}| \cdot \sigma_{f^{i^{*,(j-1)}}(x)}\right]$$

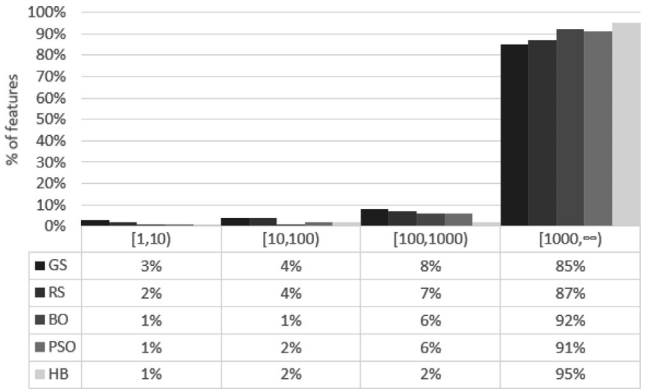
Notice that $\alpha_{i^{*,(j-1)}}^{(j-1)}$ does not exist when $j = 1$. In this case, the stop criterion is defined as $ICM_{\{\sigma_{f(x)}\}_{f(x)\in\mathcal{F}}}\left(\alpha_{i^{*,(1)}}^{(1)}, \alpha_{i^{*,(0)}}^{(0)}\right) \equiv False$. This means that this stop criterion guarantees the selection of at least one feature. Remove line number for one line.

4.2. Implicit regularization

The Caruana method has been shown to perform well in ensemble HPO. This means that combining several features, even if they are highly correlated can be promising. BOOST presents a drawback in this respect because it tries to extract the maximum amount of information from each selected feature in each stage while preventing other highly correlated features from bearing any influence on. An implicit regularization is proposed in order to overcome this drawback and make it possible to include correlated features in the ensemble, which may improve the predictive performance. An a priori idea may consists to be weight the coefficient influence using the probability of this feature being relevant in the ensemble, taking into account that all the features selected before have been included in the ensemble. A probability of 1 for all the features results in the BOOST approach, since all the selected features have the maximum influence on the ensemble. An alternative stems from the well-known sunrise problem formulated by Laplace, which consists of estimating the probability that the sun will rise tomorrow given that it has previously risen $j-1$ times. This estimation was solved by Laplace himself through his own rule of succession [83]. This probability has been stated as:

$$p_L(j) = \frac{(j-1)+1}{(j-1)+2} = \frac{j}{j+1}$$

Initially, the first selected feature will have an influence of $1/2$, allowing in successive stages for other highly correlated features to be selected (including this very same feature). We must remember that each successive target of BOOST in the following stages are the resulting residuals and hence, the regression coefficients using OLS keep decreasing. Taking this into consideration, weighting the coefficients with this probability will avoid having to reduce the influence of those features that are successively selected in more advance stages (even when they are previously selected), since, this probability asymptotically increases to 1. This implicit regularization is then included when it comes to compute the target (residual) of the next stage. In addition to that, this regularization satisfies being independent from every other stage. This condition is mandatory for BOOST because BOOST requires that the regularization procedure may be applied to each selected feature independently of all the features from the remaining stages. This is different from applying the procedure jointly, such as in GEM, which globally adjusts the influence of the features in order to satisfy the constraints of being positive and sum to one. Furthermore, this regularization also satisfies the condition of not including a priori information. This condition is relevant in HPO since there is no a priori available information in this context to be included in the regularization procedure. In this sense, typical regularization procedures either include hyperparameters whose values must be a priori fixed or impose certain conditions to be satisfied beforehand. Algorithm 2 displays RBOOST. Particularly, lines 11 and 13 of Algorithm 1 become, respectively, line 12 and 14 of Algorithm 2, where $p_L(j)$ weights $\alpha_{i^{*,(j)}}^{(j)}$. Line 9 is included in Algorithm 2. This is because if the stop criterion is satisfied in stage j, then the influence of the feature selected in the previous stage $j-1$ will have all the influence. Hence, the correspondent coefficient must go back to its original value.



	[1,10)	[10,100)	[100,1000)	[1000,∞)
GS	3%	4%	8%	85%
RS	2%	4%	7%	87%
BO	1%	1%	6%	92%
PSO	1%	2%	6%	91%
HB	1%	2%	2%	95%

Fig. 2. Percentage of features with the specified VIF values among all features taken for the predictions of Ridge, SVR and RFR varying all the hyperparameter values explored by GS, RS, BO, PSO and HB.

5. Experimental settings and multicollinearity analysis

This section covers the experimental settings (see Section 5.1) and also describes and discusses a multicollinearity analysis carried out (see Section 5.2).

5.1. Experimental settings

This section goes through the settings established for the experiments, namely, the datasets, the base-learners with their hyperpa-

rameters to be tuned, the methods for performing the ensemble and the evaluation loss description. All code was implemented in Python language using the scikit-learn library[2].

sian Optimization (BO) [58], Particle Swarm Optimization (PSO) [59] and Hyperband (HB) [60] are part of the methods that carry out a guided search.

Algorithm 2: RBOOST for stacking ensemble

Algorithm 2 RBOOST for stacking ensemble

function REGULARIZED_BOOSTING

Input: $\{f^i(x)\}_{i=1}^p$ features of the ensemble (model predictions), y the original target, $\mathcal{L}$ a loss function

Output: $\mathcal{S}$, $\mathcal{A}$, β

$\mathcal{A} \leftarrow \emptyset$ ▷ $\mathcal{A}$ set of weights for the features (models) from $\{f^i(x)\}_{i=1}^p$ selected by RBOOST

$\mathcal{S} \leftarrow \emptyset$ ▷ $\mathcal{S}$ set of features (models) from $\{f^i(x)\}_{i=1}^p$ selected by RBOOST

$\beta \leftarrow 0$ ▷ Bias of the ensemble obtained by RBOOST

$r^{(0)} \leftarrow y$ ▷ The initial residual in RBOOST is the original target y

while true **do**

$i^{*,(j)} \leftarrow \arg \min_{\{i:f^i(x)\in\mathcal{F}\}} \{l_i^{(j)} : [\alpha_i^{(j)}, \beta_i^{(j)}, l_i^{(j)}] \leftarrow \text{OLS}(f^i(x), r^{(j-1)}, \mathcal{L})\}$ ▷ BOOST selects the feature with the lowest value of the loss function $\mathcal{L}$, performing an OLS of each feature isolated from the rest.

if stop_criterion **then** ▷ Stop criterion: AIC, AICc, BIC, HQIC, gMDL or ICM

$\mathcal{A} \leftarrow \mathcal{A} \setminus \{p_L(j-1) \cdot \alpha_{i^{*,(j-1)}}^{(j-1)}\} \cup \{\alpha_{i^{*,(j-1)}}^{(j-1)}\}$

break

end if

$r^{(j)} \leftarrow r^{(j-1)} - (p_L(j) \cdot \alpha_{i^{*,(j)}}^{(j)} \cdot f^{i^{*,(j)}}(x) + \beta_{i^{*,(j)}}^{(j)})$ ▷ The residual (the target for the next stage) is updated

$\mathcal{S} \leftarrow \mathcal{S} \cup \{f^{i^{*,(j)}}(x)\}$ ▷ The selected feature in each stage is added to $\mathcal{S}$

$\mathcal{A} \leftarrow \mathcal{A} \cup \{p_L(j) \cdot \alpha_{i^{*,(j)}}^{(j)}\}$ ▷ The weight for the selected feature is added to $\mathcal{A}$

$\beta \leftarrow \beta + \beta_{i^{*,(j)}}^{(j)}$ ▷ The bias is updated

end while

return $\mathcal{S}$, $\mathcal{A}$, β ▷ The stop criterion is satisfied and $\mathcal{S}$, $\mathcal{A}$ and β are returned

end function

Datasets. Datasets coming from the UCI repository[3] were taken for performing the experiments. Table 2 displays the number of instances and features of every dataset. The values for both properties vary from one dataset to another, which enables having different scenarios in order to check the behaviour of each approach. Specifically, the number of instances ranges from 100 to 4898 and the number of features varies from 4 to 119.

Sampling hyperparameter strategies. Several sampling hyperparameter strategies have been included in the experiments. The approaches chosen were those popular and widespread in the AutoML and CASH frameworks. Two of these strategies are Grid Search (GS) [55,56] and Random Search (RS) [57], which are the kind that do not perform a guided search. On the other hand, Baye-

Base-learners and their hyperparameters. Different base-learners were tested in the experiments. Particularly, the approaches taken were Ridge Regression (Ridge)[4], Support Vector Regressor (SVR)[5] and Random Forests Regression (RFR)[6]. Table 3 displays the hyperparameter configuration trials for each sampling strategy. A total of $6 \cdot 6 = 36$ trials for Ridge, $7 + 7 \cdot 4 = 35$ trials for SVR and $7 \cdot 5 = 35$ trials for RFR are explored for the GS sampling method. The same number of trials were taken for the RS sampling approach using a uniform distribution on the specified sets. Finally, the same number of iterations was defined for the search over the specified sets of hyperparameter values that were fixed for BO, PSO and HB.

[2] https://scikit-learn.org/stable/

[3] https://archive.ics.uci.edu/ml/datasets.php

[4] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

[6] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Table 3
Hyperparameter values for the base-learners: Ridge (alpha: Regularization strength and solver: Solver used in the computational routines). SVR (C: Regularization hyperparameter, kernel: Kernel type used and gamma: RBF kernel parameter). RFR (min_samples_leaf: The minimum fraction of samples at a leaf node and max_features: The fraction of features in the best split).

		Hyperparameter Values		
	Hyperparameter name	GS	RS	sBO, PSO and HB
Ridge	alpha	$\{0, 10^{[-4:0]}\}$	U([0, 1])	[0, 1]
	solver	{svd, cholesky, lsqr, sparse_cg, sag, saga}	U({svd, cholesky, lsqr, sparse_cg, sag, saga})	{svd, cholesky, lsqr, sparse_cg, sag, saga}
SVR	C	$\{10^{[-3:3]}\}$	$10^{U([-3,3])}$	$[10^{-3}, 10^{3}]$
	kernel	{linear, RBF}	U({linear, RBF})	{linear, RBF}
	gamma	$\{10^{[-3:0]}\}$	U([0.01, 1])	[0.01, 1]
RFR	min_samples_leaf	$\{2^{[-2:-8]}\}$	$2^{U([-8,-2])}$	$[2^{-8}, 2^{-2}]$
	max_features	{1, 0.8, 0.6, 0.4, 0.2}	U([0.2,1])	[0.2, 1]

Ensemble methods. Several non-hyperparametric stop criteria were checked for Forward Stepwise Regression (FSR), Principal Component Regression (PCR), Partial Least Squares (PLS), Boosting (BOOST) and Regularized Boosting (RBOOST). Specifically, Akaike Information Criterion (AIC) [76], Akaike Information Criterion corrected (AICc) [77], Bayesian Information Criterion (BIC) [78], Hannan-Quinn Information Criterion (HQIC) [79] and generalized Minimum Description Length (gMDL) [80] were compared. The best stop criterion among AIC, AICc, BIC, HQIC and gMDL for FSR, PCR, PLS, BOOST and RBOOST were compared with the proposed stop criterion called Increasing Coefficient Magnitude (ICM), with the Ordinal Least Squares (OLS) and the Generalized Ensemble Method (GEM). Also, a comparison was carried out among the method that just chooses a model (Best), the Basic (BEM) and Generalized (GEM) Ensemble methods [9], the method that provides weights inversely proportional to the expected error (IEW) [8] and the Caruana [12] method.

Evaluation score. The evaluation score was the relative mean squared error computed using a 3-fold cross validation procedure. A higher number of folds would probably provide a lower prediction error. But would also mean training an excessive number of models, which can be computationally heavy and time-consuming. This is especially relevant in this context, since the cross validation is performed along the whole process of Fig. 1. Particularly, in the phase previous to ensemble, a total of 3 base-learners (Ridge, SVR and RFR) were trained. Respectively, a total of 36, 35 and 35 configuration trials (see Table 3) were generated with 5 different sampling strategies (GS, RS, BO, PSO and HB) for 15 different datasets. This leads to $(36 + 35 + 35) \cdot 5 \cdot 15 = 7950$ cross validation experiments. In the ensemble phase, there are 5 ensemble strategies different from stacking (BEM, IEW, GEM, OLS and Caruana). In the case of the stacking ensemble method, there are 5 different meta-learners (FSR, PCR, PLS, BOOST and RBOOST), each one with 5 stop criteria (AIC, AICc, BIC, HQIC and gMDL). Additionally, the meta-learners BOOST and RBOOST also work with the ICM stop criterion, which adds up 2 more ensemble possibilities. In total, there are $5 + 5 \cdot 5 + 2 = 32$ ensemble strategies. All these ensemble strategies were performed for the 3 base-learners (Ridge, SVR and RFR) and the 5 different sampling strategies (GS, RS, BO, PSO and HB), and 15 datasets, which leads to $3 \cdot 5 \cdot 15 \cdot 32 = 7200$ cross validation ensembles. Therefore, the 7950 experiments of the phase previous to ensemble and the 7200 ensembles had to be repeated as many times as the number of folds. Hence, and in the interest to compare the approaches rather than to get optimal predictions, a cross validation of just 3 folds was performed in order to reduce the magnitude of the experiments.

Statistical significant test methods. A Friedman test that rejects the null hypothesis that states that not all learners perform equally [84,85] has been carried out over the evaluation score. The Friedman test is a non-parametric hypothesis test that ranks all algorithms for each data set separately. If the null-hypothesis (all ranks are not significantly different) is rejected, the Nemenyi test [86] is adopted as the post hoc test. According to the Nemenyi test, the performance of two algorithms is considered significantly different if the corresponding average ranks differ by at least the so-called critical difference.

5.2. Multicollinearity analysis

We have argued before how multicollinearity is an issue to avoid, or, at least, mitigate, since it may lead to unstable models under small variations [27]. Despite the predictive performance may be not affected, it does call into question the significance of highly correlated features. This problem looms in HPO, as commented before, in Section 1. This is because certain hyperparameter configurations may produce underfitted models that might give similar prediction values for all the instances, and, in addition, may not induce different models, since the same machine learning system is used, with just little variations in its hyperparameter values.

Several ways of detecting multicollinearity have been studied [87], but the Variable Inflation Factor (VIF) has shown to be the most promising and it is the most widely adopted. This is because VIF is based on calculating the linear regression of a single feature directly against the rest of them. The VIF is the inverse of the tolerance. The tolerance is computed as $1 - R^2$, where R^2 is the coefficient of determination, which measures how well correlated is a certain feature with the remaining ones. R^2 indicates the percentage of the variance in a feature that can be attributed to the set of the remaining features. The VIF represents the factor by which the correlations amongst the remaining features contribute to the variance of the feature for which the VIF is computed. This variance is the error in the coefficient estimation. And this error is taken to establish the confidence intervals of the coefficient estimation. Hence, the higher the error, the wider the confidence interval is. Consequently, coefficient estimation becomes unstable and less accurate. The coefficient of determination R^2 is computed from the residual sum of squares (rss) and the total sum of squares (tss) as $R^2 = 1 - \frac{rss}{tss}$. If R^2 is equal to 0, the variance of the remaining independent features cannot be predicted from the independent feature for which the R^2 is computed. Therefore, when the VIF is equal to 1, the independent feature for which the R^2 is computed is not correlated to the remaining ones, which means multicollinearity does not exist in this regression model. As R^2 becomes close to 1, the independent feature becomes highly correlated with the rest of the features, and multicollinearity tends to infinity. Experience indicates that a VIF greater than 5 or 10 [88] indicates

Table 4
Averaged Friedman ranks for the relative mean squared error over all datasets for FSR, PCR, PLS, BOOST and RBOOST using several stop criteria (AIC, AICc, BIC, HQIC and gMDL), taking into account some base-learners (Ridge, SVR and RFR) and several sampling strategies (GS, RS, BO, PSO and HB). The best averaged rank in each row is bolded.

			FSR			
HPO	MLS	AIC	AICc	BIC	HQIC	gMDL
GS	Ridge	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	SVR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	RFR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	GS	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
RS	Ridge	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	SVR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	RFR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	RS	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
BO	Ridge	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	SVR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	RFR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	BO	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
PSO	Ridge	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	SVR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	RFR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	PSO	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
HB	Ridge	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	SVR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	RFR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	HB	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	→ tal	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
			PCR			
HPO	MLS	AIC	AICc	BIC	HQIC	gMDL
GS	Ridge	2.80	**2.47**	2.93	2.77	4.03
	SVR	3.20	**2.13**	3.30	2.63	3.73
	RFR	2.93	**2.20**	3.47	2.73	3.67
Mean	GS	2.98	**2.27**	3.23	2.71	3.81
RS	Ridge	3.03	**2.60**	3.07	**2.60**	3.70
	SVR	3.13	**2.40**	3.13	2.67	3.67
	RFR	3.03	**1.83**	3.67	2.43	4.03
Mean	RS	3.07	**2.28**	3.29	2.57	3.80
BO	Ridge	3.00	**2.67**	3.10	3.03	3.20
	SVR	3.07	**1.90**	3.53	2.60	3.90
	RFR	3.00	**2.07**	3.57	2.87	3.50
Mean	BO	3.02	**2.21**	3.40	2.83	3.53
PSO	Ridge	2.97	2.73	3.30	**2.63**	3.37
	SVR	3.40	**2.20**	3.03	2.53	3.83
	RFR	3.23	**2.20**	3.67	2.30	3.60
Mean	PSO	3.20	**2.38**	3.33	2.49	3.60
HB	Ridge	3.07	**2.77**	2.97	2.87	3.33
	SVR	3.10	**2.50**	3.30	2.60	3.50
	RFR	2.97	**2.23**	3.50	2.57	3.73
Mean	HB	3.04	**2.50**	3.26	2.68	3.52
Mean	→ tal	3.07	**2.33**	3.33	2.65	3.62
			PLS			
HPO	MLS	AIC	AICc	BIC	HQIC	gMDL
GS	Ridge	3.00	2.97	**2.93**	3.00	3.10
	SVR	2.97	**2.47**	3.03	3.33	3.20
	RFR	3.30	**2.70**	2.87	2.93	3.20
Mean	GS	3.09	**2.71**	2.94	3.09	3.17
RS	Ridge	2.93	3.27	**2.77**	3.10	2.93
	SVR	**2.83**	2.97	3.13	2.87	3.20
	RFR	**2.67**	2.93	3.23	3.00	3.17
Mean	RS	**2.81**	3.06	3.04	2.99	3.10
BO	Ridge	2.93	**2.80**	3.13	2.93	3.20
	SVR	**2.40**	2.63	3.27	2.83	3.87
	RFR	3.23	**2.67**	2.80	3.03	3.27
Mean	BO	2.86	**2.70**	3.07	2.93	3.44
PSO	Ridge	2.97	2.97	**2.80**	2.97	3.30
	SVR	3.03	**2.77**	3.00	3.13	3.07
	RFR	2.80	**2.73**	3.23	2.93	3.30
Mean	PSO	2.93	**2.82**	3.01	3.01	3.22
HB	Ridge	**2.77**	3.10	3.10	**2.77**	3.27
	SVR	2.80	**2.67**	3.27	3.00	3.27
	RFR	3.13	2.90	**2.87**	3.13	2.97
Mean	HB	2.90	**2.89**	3.08	2.97	3.17
Mean	→ tal	2.91	**2.85**	3.04	2.97	3.23
			BOOST			

(*continued on next page*)

Table 4 (*continued*)

			FSR			
HPO	MLS	AIC	AICc	BIC	HQIC	gMDL
HPO	MLS	AIC	AICc	BIC	HQIC	gMDL
GS	Ridge	**2.93**	**2.93**	3.10	**2.93**	3.10
	SVR	2.90	2.90	**2.87**	3.07	3.27
	RFR	**2.93**	**2.93**	3.10	**2.93**	3.10
Mean	GS	**2.92**	**2.92**	3.02	2.98	3.16
RS	Ridge	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	SVR	2.73	**2.43**	3.23	3.10	3.50
	RFR	3.13	3.07	**2.93**	**2.93**	**2.93**
Mean	RS	2.96	**2.83**	3.06	3.01	3.14
BO	Ridge	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	SVR	2.83	2.83	3.07	**2.77**	3.50
	RFR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	BO	2.94	2.94	3.02	**2.92**	3.17
PSO	Ridge	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	SVR	**2.93**	3.10	**2.93**	**2.93**	3.10
	RFR	3.10	3.10	**2.93**	**2.93**	**2.93**
Mean	PSO	3.01	3.07	**2.96**	**2.96**	3.01
HB	Ridge	**2.93**	**2.93**	3.10	**2.93**	3.10
	SVR	2.83	**2.73**	3.13	3.10	3.20
	RFR	2.97	**2.87**	3.10	2.97	3.10
Mean	HB	2.91	**2.84**	3.11	3.00	3.13
Mean	→ tal	2.95	**2.92**	3.04	2.97	3.11
			RBOOST			
HPO	MLS	AIC	AICc	BIC	HQIC	gMDL
GS	Ridge	**2.87**	3.00	3.00	3.00	3.13
	SVR	**2.67**	2.83	3.23	2.97	3.30
	RFR	**2.90**	**2.90**	3.07	3.07	3.07
Mean	GS	**2.81**	2.91	3.10	3.01	3.17
RS	Ridge	**2.97**	**2.97**	**2.97**	**2.97**	3.13
	SVR	3.10	3.10	**2.77**	2.93	3.10
	RFR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	RS	3.02	3.02	**2.91**	2.97	3.08
BO	Ridge	**2.93**	**2.93**	3.13	**2.93**	3.07
	SVR	**2.83**	3.07	2.90	3.03	3.17
	RFR	**2.87**	**2.87**	3.03	3.03	3.20
Mean	BO	**2.88**	2.96	3.02	3.00	3.14
PSO	Ridge	**2.93**	**2.93**	3.07	**2.93**	3.13
	SVR	**2.80**	2.97	3.13	2.97	3.13
	RFR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	PSO	**2.91**	2.97	3.07	2.97	3.09
HB	Ridge	**2.87**	3.00	3.00	3.00	3.13
	SVR	2.77	**2.57**	3.20	3.00	3.47
	RFR	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
Mean	HB	2.88	**2.86**	3.07	3.00	3.20
Mean	→ tal	**2.92**	2.95	3.02	2.99	3.12

multicollinearity [89]. In the case of HPO, the values of VIF drastically exceed these limits, as can be seen in Fig. 2. Particularly, just 1% or 2% of the features, among all the features taken for the predictions of Ridge SVR and RFR varying all the hyperparameter values explored by GS, RS, BO, PSO and HB, have a VIF below 10. Furthermore, between 84% and 95% of the features present values of VIF greater than 1000.

6. Result analysis

This section displays and discusses the performance of the approaches (see Section 6.1) and also analyses the methods in terms of the models involved in the ensemble (see Section 6.3).

6.1. Performance analysis

The relative mean squared error, calculated using a 3-fold cross validation procedure, was computed for each dataset, each base-learner (Ridge, SVR and RFR), each sampling strategy (GS, RS, BO, PSO and HB) and each ensemble method FSR, PCR, PLS, BOOST and RBOOST, using several stop criteria (AIC, AICc, BIC, HQIC and gMDL), each ensemble method OLS, GEM, BEM, IEW, Caruana and the proposed RBOOST with the novel ICM stop criterion. The Friedman ranks of the ensemble methods over each dataset were also computed for all base-learners and for all the sampling strategies. The tables with the relative mean squared error and the corresponding Friedman rank are not displayed in the paper due to lack of space. They are available at https://github.com/laurafernandezdiaz/Ensemble. Instead, Tables 4–6 show the Friedman ranks averaged over all the datasets. These tables also show the average ranks over the base-learner for each sampling strategy. Finally, the total average ranks over the sampling strategies are displayed at the bottom of the tables.

Particularly, Table 4 shows the averaged Friedman ranks over all datasets for FSR, PCR, PLS, BOOST and RBOOST using several stop criteria (AIC, AICc, BIC, HQIC and gMDL), taking into account some base-learners (Ridge, SVR and RFR) and several sampling strategies (GS, RS, BO, PSO and HB). The results reported in this table enable analyzing the behaviour of the stop criteria when used in different meta-learners. The stop criterion that reports the best performance was selected for each meta-learner in order to be

Table 5
Averaged Friedman ranks for the relative mean squared error over all datasets for OLS, GEM and the best stop criteria among AIC, AICc, BIC, HQIC and gMDL, for FSR, PCR, PLS, BOOST and RBOOST, as well as the novel stop criterion ICM for BOOST and RBOOST, taking into account some base-learners (Ridge, SVR and RFR) and several sampling strategies (GS, RS, BO, PSO and HB). The best averaged rank in each row is bolded.

HPO	MLS	OLS	GEM	FSR (*)	PCR (AICc)	PLS (AICc)	BOOST (AICc)	RBOOST (AIC)	BOOST (ICM)	RBOOST (ICM)
GS	Ridge	8.93	5.53	4.13	7.27	4.53	3.77	4.03	3.73	**3.07**
	SVR	8.13	4.60	4.87	7.67	5.00	4.63	4.23	3.33	**2.53**
	RFR	6.27	2.93	5.10	8.00	5.80	5.10	4.93	4.20	**2.67**
Mean	GS	7.78	4.36	4.70	7.64	5.11	4.50	4.40	3.76	**2.76**
RS	Ridge	8.70	5.03	4.27	6.97	4.57	3.97	4.17	3.90	**3.43**
	SVR	8.10	5.17	4.53	7.83	4.17	**3.40**	4.47	3.57	3.77
	RFR	6.53	**3.53**	4.63	7.80	5.47	5.13	4.63	3.67	3.60
Mean	RS	7.78	4.58	4.48	7.53	4.73	4.17	4.42	3.71	**3.60**
BO	Ridge	9.00	6.07	4.10	6.33	4.33	4.10	3.93	**3.20**	3.93
	SVR	7.87	4.13	4.77	7.40	5.40	3.97	4.20	4.00	**3.27**
	RFR	5.53	3.47	5.20	8.33	5.20	5.20	4.87	4.33	**2.87**
Mean	BO	7.47	4.56	4.69	7.36	4.98	4.42	4.33	3.84	**3.36**
PSO	Ridge	9.00	6.33	3.63	6.60	5.20	3.63	3.40	3.87	**3.33**
	SVR	7.93	4.13	4.73	8.27	5.33	4.73	4.47	2.87	**2.53**
	RFR	7.60	4.73	4.23	8.27	6.33	4.33	4.23	**2.60**	2.67
Mean	PSO	8.18	5.07	4.20	7.71	5.62	4.23	4.03	3.11	**2.84**
HB	Ridge	8.93	5.67	3.93	7.47	4.47	3.63	3.83	3.67	**3.40**
	SVR	7.87	5.47	4.57	7.67	5.00	4.27	3.97	3.87	**2.33**
	RFR	5.47	3.67	5.10	8.00	6.33	4.87	5.17	3.47	**2.93**
Mean	HB	7.42	4.93	4.53	7.71	5.27	4.26	4.32	3.67	**2.89**
Mean	→ tal	7.60	4.64	4.52	7.61	5.20	4.33	4.33	3.63	**3.13**

Table 6
Averaged Friedman ranks for the relative mean squared error over all datasets for Best, BEM, IEW, GEM, Caruana and RBOOST(ICM), taking into account some base-learners (Ridge, SVR and RFR) and several sampling strategies (GS, RS, BO, PSO and HB). The best averaged rank in each row is bolded.

HPO	MLS	Best	BEM	IEW	GEM	Caruana	RBOOST (ICM)
GS	Ridge	2.97	4.93	4.47	3.67	2.97	**2.00**
	SVR	3.40	5.07	4.27	3.47	2.60	**2.20**
	RFR	3.80	4.77	3.87	2.90	**2.67**	3.00
Mean	GS	3.39	4.92	4.20	3.34	2.74	**2.40**
RS	Ridge	3.20	4.87	4.53	3.37	2.67	**2.37**
	SVR	3.53	4.97	3.67	3.87	2.80	**2.17**
	RFR	3.33	4.93	3.93	3.00	**2.90**	**2.90**
Mean	RS	3.36	4.92	4.04	3.41	2.79	**2.48**
BO	Ridge	3.10	4.40	4.00	3.67	3.37	**2.47**
	SVR	3.60	5.13	4.40	**2.47**	2.93	**2.47**
	RFR	3.87	4.87	4.07	2.93	2.73	**2.53**
Mean	BO	3.52	4.80	4.16	3.02	3.01	**2.49**
PSO	Ridge	5.20	4.60	3.87	2.93	2.47	**1.93**
	SVR	5.93	4.93	3.93	2.53	2.13	**1.53**
	RFR	6.00	4.53	3.87	2.73	2.07	**1.80**
Mean	PSO	5.71	4.69	3.89	2.73	2.22	**1.76**
HB	Ridge	3.40	4.80	4.47	3.33	2.93	**2.07**
	SVR	3.67	5.30	3.47	3.90	2.53	**2.13**
	RFR	3.93	4.80	3.73	3.20	**2.27**	3.07
Mean	HB	3.67	4.97	3.89	3.48	2.58	**2.42**
Mean	→ tal	4.04	4.84	3.98	3.14	2.65	**2.34**

compared to the new proposed stop criterion ICM. As seen, there is not so much difference in performance among the stop criteria. Particularly, FSR reports the same result independently of the stop criterion, base-learner and sampling strategy taken. Analyzing in detail this curious behavior, we found that FSR just takes one feature, the best one, and then all stop criteria are satisfied. This may happen because the best feature (the best prediction) completely explain the target, and consequently, the rest of the predictions (which are all quite similar) do not seem to provide any relevant information. This is not the case in the other methods (see Section 6.3 about the analysis and study of the features and iterations taken by the methods). In the case of PCR and PLS, the features are combined to obtain the first component, after that, the second component is obtained orthogonally to the first component and so on, which implies that the maximum information of the features is extracted in each step from what had not already been extracted in the previous steps. With BOOST and RBOOST the target varies from one stage to the next. In particular, the information explained by the feature selected in each stage is removed from the target in order to obtain the target for the next stage. As a result, the features are taken successively in each stage according to the information contained in the updated target. Both mechanisms guarantee, in a certain sense, that the maximum remaining information can be collected until certain stage, when the stop criterion is satisfied. The ranks of the different stop criteria for PCR, PLS, BOOST and RBOOST are quite similar. In fact, our own stop criterion behaves quite similarly. AIC is the point of departure and the rest just add some correction factors to consider the balance between instances and features or whether the number of instances is exceeded. In any case, AICc seems to provides the best results for PCR and

PLS. As to BOOST and RBOOST, both AIC and AICc slightly outperform the rest. gMDL is the worst, followed by BIC, so penalizing the number of instances does not seem to be a good practice. There are significant differences, up to 90% and 95%,[7] between AICc and AIC, BIC and HQIC, but only in the case of PCR.

Table 5 displays the averaged Friedman ranks over all datasets for OLS, GEM[8] and for the best stop criteria (according to the results of Table 4) among AIC, AICc, BIC, HQIC and gMDL for FSR, PCR, PLS, BOOST and RBOOST, as well as the novel stop criterion ICM for BOOST and RBOOST, taking into account some base-learners (Ridge, SVR and RFR) and several sampling strategies (GS, RS, BO, PSO and HB). In this table, one can observe that both BOOST and RBOOST outperform the rest of the methods, for the best stop criterion among AIC, AICc, BIC, HQIC and gMDL and also for the novel stop criterion ICM (see the last row of the last four columns of Table 5). Indeed, all of them, BOOST(AICc), RBOOST(AIC), BOOST(ICM) and RBOOST (ICM), present significant differences at the confidence levels of 90% and even 95%[9] when compared to OLS, PCR and PLS. Besides, using ICM as stop criterion makes BOOST and RBOOST significantly better at these confidence levels than GEM and FSR, in addition to OLS, PCR and PLS. The poor performance of OLS may be the result of multicollinearity, which is highly present in HPO. This drawback is corrected by GEM regularizing in OLS by constraining of the weights to be positive and sum to one. In the case of FSR, it just happens that regardless of the stop criteria only one feature is selected (see the comments of Table 4), hence multicollinearity disappears. In fact, GEM and FSR perform considerably better than OLS. The difference in performance between PCR and PLS is caused by the former not taking into account the target in order to build the components, whereas the latter does. These results confirm two well-known conclusions, namely, i) the target contains critical information and ii) regularization helps to alleviate the problems derived from multicollinearity. In this sense, BOOST and RBOOST (more so, because RBOOST includes a regularization procedure) succeed because they squeeze the information contained in the target. In order to achieve this, each stage the information explained by the selected feature is removed from the target, and the remaining information is left to be explained by the features selected in the following stages. Comparing BOOST and RBOOST, RBOOST stands out as the best option because of the regularization element added to BOOST. Hence, smoothing the influence of the features selected in the first stages in order to allow subsequent features to take part of the ensemble clearly improves the overall performance of the ensemble. In addition, the novel stop criterion ICM improves both BOOST and RBOOST. The results state that ICM is a robust criterion, since it is based on the coefficient from the feature selected in each stage, which enables it to discern promising choices of features, both when a feature has already been selected or if a feature is selected for the first time. In any case, it seems that RBOOST benefits from ICM to a higher degree than BOOST. In fact, RBOOST(ICM) works significantly better that BOOST (AICc) and RBOOST(AIC), whereas BOOST(ICM) does not. Hence, the synergy between both the implicit regularization and the ICM helps to improve the predictive performance of the ensemble. Both improvements are independent from each other, which allows applying both of them simultaneously and combining the benefits provided by them.

Finally, Table 6 presents the averaged Friedman ranks over all datasets for Best (not performing ensemble), BEM, IEW, GEM[10], Caruana and RBOOST with the novel stop criterion ICM (the best result in Table 5), again taking into account the same base-learners (Ridge, SVR and RFR) and the same sampling strategies (GS, RS, BO, PSO and HB). Both the Caruana method and RBOOST (ICM) clearly outperform Best, BEM, IEW and GEM. Besides, the differences at s of 90% and 95%[11] are significant. RBOOST(ICM) performs better than Caruana in almost all cases, but there are no significant differences between them. Best, BEM and IEW show the worst results. These are very simple ensemble methods that do not include a learning procedure in the ensemble and they select each feature only once. GEM performs slightly better, it includes a learning procedure, which adds a higher generalization power. Particularly, it performs an OLS with constraints as a regularization procedure, but if a feature is taken, it is taken only once, such as in Best, BEM and IEW. Unlike these methods, both Caruana and RBOOST (as well as BOOST) are able to take the same feature more than once, and then fully exploit the information contained in it. Nevertheless, RBOOST, like GEM, includes a learning procedure which may add more generalization power and a regularization procedure.

6.2. Computational time analysis

This section deals with a comparison of the ensemble strategies in regard to the computational time. Table 7 shows the computational time (in seconds) spent on training the models that would feed the ensemble for all datasets and all the configuration trials generated by each sampling strategy (GS, RS, BO, PSO and HB) and for each base-learner (Ridge, SVR and RFR). It also shows the total time spent by each sampling strategy and each base-learner. As it shows, the total computational time spent on training all the models is almost 50 days (4297456 s). Each sampling strategy spent similar computational time. However, the base-learner SVR spent considerably much more computational time that Ridge and RFR (more than 90% of the computational time). We are reminded that the number of configuration trials for the base-learner are almost equal (36 for Ridge and 35 for SVR and RFR). Table 8 displays the computational time in seconds for the different ensemble approaches. The time is computed adding up the time spent by all base-learners, all datasets and all sampling strategies. Obviously, the least costly approaches are those that do not include a learning procedure in the ensemble, that is, Best, BEM and IEW (note that the Best method performs no ensemble). Conversely, the most costly approaches are GEM and Caruana. Comparing the computational time reported in Table 7 with the computational time reported in Table 8, the computational time spent by the ensemble approaches is considerably lower than the computational time spent on training the models for all the configuration trials. This allows concluding there is a great benefit in performance when carrying out an ensemble procedure, in comparison with the small loss in computational time, since the models that feed the ensemble must be trained in any case.

6.3. Analysis of the models involved in the ensemble

An analysis of the number of models involved in the ensemble was performed. Table 9 displays the averaged (over all datatasets and all base-learners) number of different features (models) taken by the Caruana method, and by BOOST and RBOOST for all the stop criteria studied (see the top of Table 9). It also shows the averaged (again over all datatasets and all base-learners) number of features with replacement taken, which is in fact the number of iterations carried out by the methods (see the bottom of Table 9).

[7] The respective critical differences for confidence levels of 90% and 95% are 0.36 and 0.40

[8] GEM is included here for being a version of OLS with the constraint of the weights to be positive and sum to one.

[9] The respective critical differences for confidence levels of 90% and 95% are 0.73 and 0.80.

[10] GEM is included here for its performance and as an improvement of BEM.

[11] The respective critical differences for confidence levels for 90% and 95% are 0.45 and 0.50.

Table 7
Time (in seconds) of execution for the training of all the 36 (for Ridge) or 35 (for SVR and RFR) models (the sum regarding all datasets) for each base-learner and each sampling strategy

	Ridge	SVR	RFR	total
GS	29904	779056	30725	839685
RS	44275	799202	43934	887411
BO	24230	770431	21950	816611
PSO	25178	766460	22795	814433
HB	62496	812348	64472	939316
total	186082	3927498	183876	4297456

Table 8
Time (in seconds) of execution for the ensemble approaches (the sum regarding all base-learners, all datasets and all sampling strategies)

Best	BEM	IEW	OLS	GEM	Caruana
2.25	4.50	155.25	254.25	4950.00	4050.00
	FSR	PCR	PLS	BOOST	RBOOST
AIC	207.00	560.25	2400.75	150.75	576.00
AICc	405.00	828.00	117.00	310.50	371.25
BIC	2517.75	175.50	274.50	171.00	193.50
HQIC	279.00	594.00	450.00	159.75	542.25
gMDL	623.25	801.00	423.00	162.00	378.00
ICM	-	-	-	571.50	492.75

Table 9
Averaged (over all datatasets and all base-learners) number of different features (models) and number of features with replacement (iterations) for the Caruana method, and for BOOST and RBOOST taking all the stop criteria.

	Different features												
	Caruana	BOOST						RBOOST					
		AIC	AICc	BIC	HQIC	gMDL	ICM	AIC	AICc	BIC	HQIC	gMDL	ICM
GS	11.20	1.17	1.17	1.06	1.14	1.00	3.10	1.19	1.17	1.09	1.15	1.00	3.48
RS	11.65	1.17	1.14	1.05	1.09	1.00	3.19	1.09	1.09	1.07	1.07	1.00	3.46
BO	9.54	1.20	1.20	1.12	1.17	1.03	2.96	1.18	1.15	1.09	1.12	1.03	3.28
PSO	9.33	1.22	1.21	1.19	1.19	1.17	3.93	1.25	1.24	1.20	1.24	1.17	3.57
HB	11.95	1.17	1.16	1.06	1.15	1.00	3.19	1.21	1.17	1.07	1.12	1.00	3.78
Mean	**10.74**	**1.19**	**1.18**	**1.10**	**1.15**	**1.04**	**3.27**	**1.18**	**1.16**	**1.10**	**1.14**	**1.04**	**3.51**
	Features with replacement												
	Caruana	BOOST						RBOOST					
		AIC	AICc	BIC	HQIC	gMDL	ICM	AIC	AICc	BIC	HQIC	gMDL	ICM
GS	25.80	1.17	1.17	1.06	1.13	1.00	285.29	1.18	1.16	1.08	1.12	1.00	5.69
RS	24.97	1.10	1.14	1.03	1.06	1.00	302.65	1.57	1.57	1.55	1.56	1.51	5.72
BO	23.46	1.11	1.11	1.05	1.09	1.00	423.55	1.14	1.11	1.06	1.09	1.00	5.57
PSO	19.51	1.01	1.02	1.01	1.01	1.00	352.93	1.05	1.04	1.02	1.04	1.00	5.43
HB	26.75	1.16	1.21	1.07	1.15	1.00	303.32	1.17	1.13	1.07	1.10	1.00	6.15
Mean	**24.10**	**1.11**	**1.13**	**1.05**	**1.09**	**1.00**	**333.54**	**1.22**	**1.20**	**1.16**	**1.18**	**1.10**	**5.71**

Two conclusions can be drawn for the analysis of the different features considered in the ensemble, and: i) the Caruana method is by far the approach that takes the highest number of different features (around 10) and ii) the common stop criteria just select an average of one feature, which is very similar to using the Best method. Hence, these stop criteria do not allow neither BOOST nor RBOOST to fully exploit the information contained in the models. In this sense, the novel stop criterion ICM makes BOOST and RBOOST behave conservatively as to how many different features are to be taken in the ensemble compared to the Caruana method, but not as restrictive as with the typical stop criteria.

Regarding the number of features considered in the ensemble, and taking into account the replacement procedure, it is quite noticeable the number of iterations that BOOST with ICM stop criterion spends (in the hundreds) to end up taking only about 3 different features. However, the number of iterations drastically falls for RBOOST with ICM. In fact, RBOOST with ICM hardly performs replacement when compared to the Caruana method.

7. Conclusions and future work

This paper proposes an improved boosting approach as a meta-learner in HPO stacking ensemble, which may be included in an Automated Machine Learning (AutoML) system and which gets better predictive performance. In particular, an implicit regularization would be included in the classical boosting (BOOST) method, leading to the method called Regularized Boosting (RBOOST). Besides, a novel non-hyperparametric stop criterion for both BOOST and RBOOST methods called Increasing Coefficient Magnitude (ICM) is also proposed. Both RBOOST and ICM are specifically designed for HPO. The result is a new meta-learner for stacking

ensemble that is shown to be superior to other possible non-hyperparametric (with an adequate non-hyperparametric stop criterion) meta-learners, such as Forward Search (FSR), Principal Component Regression (PCR) or Partial Least Squares (PLS). Unlike these methods, RBOOST with ICM is built on the basis of BOOST, since our proposal works under the hypothesis that BOOST is a promising regressor in HPO stacking ensemble. The reason for this is that it performs a regression with just one feature in each stage and uses the successive residuals as targets. These are promising properties, since they allow the use of least squares (which is free of hyperparameter tuning) without being affected by the problems derived from multicollinearity. ICM has shown to be a robust criterion, since it is based on the coefficient of the selected feature in each stage rather than only on the error value, number of features and instances, which is what other state-of-the-art stop criteria are typically based on. In fact, ICM is able to discern special situations, which other stop criteria are not. The power of RBOOST lies in fully exploiting the information contained in the target. Specifically, it smoothes the weight of the features selected in the first stages in order to provide other features with an opportunity to supply further information.

It is worth noting that AutoML systems hardly include ensemble in their frameworks. Only some of them do it. The ensemble strategies they typically include are the simple (weighted or not) average or, as in the Caruana method, which computes an average with replacement and has been widely used among researchers. Additionally, some AutoML systems include stacking ensemble, where the interest lies in the included learning process, which might provide the ensemble with higher generalization power. The main drawback of stacking ensemble is the choice of an adequate meta-learner, for which there is a lack of advice in the literature in general, and in the AutoML frameworks in particular. Besides, adequate meta-learners may include real-value hyperparameters that need to be tuned in order to avoid the problems derived from multicollinearity. In this respect, the contribution of this paper is not limited to proposing RBOOST with ICM; in addition, it begins by performing an exhaustive study of possible non-hyperparametric (with an adequate non-hyperparametric stop criterion) meta-learners, such as FSR, PCR, PLS and even the original BOOST, a study that, to the best of our knowledge, has not been carried out in HPO so far. In fact, this study has helped to lay down the foundations for developing the novel approach RBOOST with the novel stop criterion ICM. In this respect, all those methods feature of the generalization power of a learning procedure and are non-hyperparametric with an adequate non-hyperparametric stop criterion. Moreover, PCR, PLS, BOOST and RBOOST are able to cope with the problems derived from multicollinearity. However, in the case of HPO, BOOST and RBOOST provide better performance than PCR and PLS. Furthermore, RBOOST, especially using ICM as stop criterion, exhibits superiority, even over BOOST, and also with regard to other state-of-the-art ensemble procedures typically included in AutoML frameworks. A computational time study was carried out, concluding that performing an ensemble process is worth, since the additional time spent on the ensemble is considerably lower than the time spent on training the models in order to feed the ensemble, which must be taken into account since these models must be trained in any case.

As future work, it would be interesting to include the ensemble procedure inside the guided search performed by the sampling strategies such as BO, PSO or HB. Hence, the next hyperparameter configuration trial in the sampling strategy would be chosen according to the best ensemble from the predictions provided by the models induced using the previous configuration trials. Another proposal for a future line of work would be to provide a non-linear ensemble strategy. Finally, this approach could be extended onto data under distribution changes, such as covariate shift.

CRediT authorship contribution statement

Laura Fdez-Díaz: Conceptualization, Methodology, Software, Validation. **José Ramón Quevedo:** Conceptualization, Methodology, Investigation, Software, Supervision. **Elena Montañés:** Investigation, Supervision, Writing - review & editing.

Data availability

I have shared the link to my data/code in the text of the manuscript

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.

Acknowledgments

This research has been partially supported by the Spanish Ministerio de Ciencia e Innovación through the grant PID2019-110742RB-I00.

References

[1] T. Yu, H. Zhu, Hyper-parameter optimization: A review of algorithms and applications, arXiv preprint arXiv:2003.05689.
[2] L. Yang, A. Shami, On hyperparameter optimization of machine learning algorithms: Theory and practice, Neurocomputing 415 (2020) 295–316.
[3] J. Mendes-Moreira, C. Soares, A.M. Jorge, J.F.D. Sousa, Ensemble approaches for regression: A survey, ACM Computing Surveys (csur) 45 (1) (2012) 1–40.
[4] J.-C. Lévesque, C. Gagné, R. Sabourin, Bayesian hyperparameter optimization for ensemble learning, in: Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI'16, AUAI Press, Arlington, Virginia, USA, 2016, p. 437–446.
[5] G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity creation methods: a survey and categorisation, Information Fusion 6 (1) (2005) 5–20, diversity in Multiple Classifier Systems.
[6] H.J. Escalante, M. Montes, E. Sucar, Ensemble particle swarm model selection, in: The 2010 International Joint Conference on Neural Networks (IJCNN), IEEE, 2010, pp. 1–8.
[7] M. Momma, K.P. Bennett, A pattern search method for model selection of support vector regression, in: Proceedings of the 2002 SIAM International Conference on Data Mining, SIAM, 2002, pp. 261–274.
[8] A. Alaa, M. Schaar, Autoprognosis: Automated clinical prognostic modeling via bayesian optimization with structured kernel learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 139–148.
[9] M.P. Perrone, L.N. Cooper, When networks disagree: Ensemble methods for hybrid neural networks, Tech. rep., Brown Univ Providence Ri Inst for Brain and Neural Systems (1992).
[10] M. Shahhosseini, G. Hu, H. Pham, Optimizing ensemble weights and hyperparameters of machine learning models for regression problems, Machine Learning with Applications 7 (2022) .
[11] A. Morales-Hernández, I. Van Nieuwenhuyse, S.R. Gonzalez, A survey on multi-objective hyperparameter optimization algorithms for machine learning (2021).
[12] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, Ensemble selection from libraries of models, in: Proceedings of the Twenty-first International Conference on Machine Learning, 2004, p. 18.
[13] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A. Smola, Autogluon-tabular: Robust and accurate automl for structured data, arXiv preprint arXiv:2003.06505.
[14] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, in: Advances in Neural Information Processing Systems, 2015, pp. 2962–2970.
[15] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, F. Hutter, Auto-sklearn 2.0: Hands-free automl via meta-learning, Journal of Machine Learning Research 23 (261) (2022) 1–61.
[16] L. Zimmer, M. Lindauer, F. Hutter, Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl, IEEE Transactions on Pattern Analysis and Machine Intelligence (2021) 1.
[17] M.J. van der Laan, E.C. Polley, A.E. Hubbard, Super learner, Statistical Applications in Genetics and Molecular Biology 6 (1).

[18] L. Kotthoff, C. Thornton, H.H. Hoos, F. Hutter, K. Leyton-Brown, Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka, The, Journal of Machine Learning Research 18 (1) (2017) 826–830.
[19] L. Kotthoff, C. Thornton, H.H. Hoos, F. Hutter, K. Leyton-Brown, Auto-weka: Automatic model selection and hyperparameter optimization in weka, in: Automated machine learning, Springer, Cham, 2019, pp. 81–95.
[20] P. Pandey, A deep dive into h2o's automl, Tech. rep., Technical report, 2019. (2019).
[21] E. LeDell, S. Poirier, H2o automl: Scalable automatic machine learning, in: Proceedings of the AutoML Workshop at ICML, Vol. 2020, 2020.
[22] S. Kumar, P. Kaur, A. Gosain, A comprehensive survey on ensemble methods, in: 2022 IEEE 7th International conference for Convergence in Technology (I2CT), 2022, pp. 1–7.
[23] P. Bühlmann, Bagging, boosting and ensemble methods, in: Handbook of computational statistics, Springer, 2012, pp. 985–1022.
[24] N. García-Pedrajas, C. Hervás-Martínez, D. Ortiz-Boyer, Cixl2: A crossover operator for evolutionary algorithms based on population features, Journal of Artificial Intelligent Research 24 (2005) 1–48.
[25] Y. Ren, L. Zhang, P. Suganthan, Ensemble classification and regression-recent developments, applications and future directions [review article], IEEE Computational Intelligence Magazine 11 (2016) 41–53.
[26] M.P. Allen, The problem of multicollinearity, Understanding regression analysis (1997) 176–180.
[27] H.A. Kiers, A.K. Smilde, A comparison of various methods for multivariate regression with highly collinear variables, Statistical Methods and Applications 16 (2) (2007) 193–228.
[28] H. Abdi, Least squares, Encyclopedia for Research Methods for the, Social Sciences (2003) 792–795.
[29] L. Breiman, Stacked regressions, Machine learning 24 (1) (1996) 49–64.
[30] M. Efroymson, Multiple regression analysis, Mathematical Methods for Digital Computers (1960) 191–203.
[31] C. Merz, M. Pazzani, A principal components approach to combining regression estimates, Machine Learning 36.
[32] V. Esposito Vinzi, W. Chin, J. Henseler, H. Wang, Handbook of Partial Least Squares: Concepts, Methods and Applications (2010).
[33] P. Bühlmann, T. Hothorn, Boosting algorithms: Regularization, prediction and model fitting, Statistical science 22 (4) (2007) 477–505.
[34] T. Hothorn, K. Hornik, A. Zeileis, Unbiased recursive partitioning: A conditional inference framework, Journal of Computational and Graphical Statistics 15 (3) (2006) 651–674.
[35] H. Drucker, C.J. Burges, L. Kaufman, A. Smola, V. Vapnik, Support vector regression machines, Advances in neural information processing systems 9.
[36] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[37] T. Swearingen, W. Drevo, B. Cyphers, A. Cuesta-Infante, A. Ross, K. Veeramachaneni, Atm: A distributed, collaborative, scalable system for automated machine learning, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 151–162.
[38] B. Komer, J. Bergstra, C. Eliasmith, Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn, ICML workshop on AutoML, Citeseer, Vol. 9, 2014, p. 50.
[39] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, D.D. Cox, Hyperopt: a python library for model selection and hyperparameter optimization, Computational Science & Discovery 8 (1) (2015) .
[40] R.S. Olson, J.H. Moore, Tpot: A tree-based pipeline optimization tool for automating, Automated Machine Learning: Methods, Systems, Challenges (2019) 151.
[41] J.D. Romano, T.T. Le, W. Fu, J.H. Moore, Tpot-nn: augmenting tree-based automated machine learning with neural network estimators, Genetic Programming and Evolvable Machines 22 (2) (2021) 207–227.
[42] S.S. Sandha, M. Aggarwal, I. Fedorov, M. Srivastava, Mango: A python library for parallel hyperparameter tuning, in: ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 3987–3991.
[43] D. Salinas, M. Seeger, A. Klein, V. Perrone, M. Wistuba, C. Archambeau, Syne tune: A library for large scale hyperparameter tuning and reproducible research, in: International Conference on Automated Machine Learning, PMLR, 2022, pp. 16–1.
[44] Y. Li, Y. Shen, H. Jiang, W. Zhang, J. Li, J. Liu, C. Zhang, B. Cui, Hyper-tune: Towards efficient hyper-parameter tuning at scale, Proc. VLDB Endow. 15 (6) (2022) 1256–1265.
[45] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, D. Sculley, Google vizier: A service for black-box optimization, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 1487–1495.
[46] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J.E. Gonzalez, I. Stoica, Tune: A research platform for distributed model selection and training, CoRR abs/1807.05118.
[47] Y. Li, Y. Shen, W. Zhang, Y. Chen, H. Jiang, M. Liu, J. Jiang, J. Gao, W. Wu, Z. Yang, C. Zhang, B. Cui, Openbox: A generalized black-box optimization service, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 3209–3219.
[48] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, A. Talwalkar, A system for massively parallel hyperparameter tuning.
[49] Y. Li, Y. Shen, J. Jiang, J. Gao, C. Zhang, B. Cui, MFES-HB: efficient hyperband with multi-fidelity quality measurements, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021, AAAI Press, 2021, pp. 8491–8500.
[50] M. Claesen, J. Simm, D. Popovic, Y. Moreau, B. De Moor, Easy hyperparameter search using optunity, arXiv preprint arXiv:1412.1114.
[51] S. Falkner, A. Klein, F. Hutter, BOHB: Robust and efficient hyperparameter optimization at scale, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 1437–1446.
[52] J.N. Van Rijn, F. Hutter, Hyperparameter importance across datasets, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2367–2376.
[53] A. Klein, S. Falkner, N. Mansur, F. Hutter, Robo: A flexible and robust bayesian optimization framework in python, in: NIPS 2017 Bayesian Optimization Workshop, 2017.
[54] L. Gustafson, Bayesian tuning and bandits: An extensible, open source library for automl, Massachusetts Institute of Technology, 2018, Ph.D. thesis.
[55] M. Claesen, B. De Moor, Hyperparameter search in machine learning, arXiv preprint arXiv:1502.02127.
[56] M. Sambridge, B. Kennett, A novel method of hypocentre location, Geophysical Journal International 87 (2) (1986) 679–697.
[57] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research 13 (2) (2012) 281–305.
[58] J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms, in: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2, NIPS'12, Curran Associates Inc., Red Hook, NY, USA, 2012, p. 2951–2959.
[59] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: International Conference on Evolutionary Programming, Springer, 1998, pp. 591–600.
[60] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, The Journal of Machine Learning Research 18 (1) (2017) 6765–6816.
[61] D.R. Jones, M. Schonlau, W.J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global optimization 13 (4) (1998) 455–492.
[62] P.R. Lorenzo, J. Nalepa, M. Kawulok, L.S. Ramos, J.R. Pastor, Particle swarm optimization for hyper-parameter selection in deep neural networks, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 481–488.
[63] M.-A. Zöller, M.F. Huber, Benchmark and survey of automated machine learning frameworks, Journal of Artificial Intelligence Research 70 (2021) 409–472.
[64] C. Wang, H. Wang, T. Mu, J. Li, H. Gao, Auto-model: Utilizing research papers and hpo techniques to deal with the cash problem, in: 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020, pp. 1906–1909.
[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, The, Journal of Machine Learning Research 12 (2011) 2825–2830.
[66] E. Tang, P. Suganthan, X. Yao, An analysis of diversity measures, Machine Learning 65 (2006) 247–271.
[67] P. Bühlmann, Bagging, boosting and ensemble methods, Handbook of Computational Statistics.
[68] L.I. Kuncheva, Switching between selection and fusion in combining classifiers: An experiment, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 32 (2) (2002) 146–156.
[69] N. Rooney, D. Patterson, S. Anand, A. Tsymbal, Dynamic integration of regression models, in: International Workshop on Multiple Classifier Systems, Springer, 2004, pp. 164–173.
[70] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 226–235.
[71] Y. Zhang, J. Liu, W. Shen, A review of ensemble learning algorithms used in remote sensing applications, Applied Sciences 12 (2022) 8654.
[72] F. Wenzel, J. Snoek, D. Tran, R. Jenatton, Hyperparameter ensembles for robustness and uncertainty quantification, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Curran Associates Inc., Red Hook, NY, USA, 2020.
[73] H. Banks, M.L. Joyner, Aic under the framework of least squares estimation, Applied Mathematics Letters 74 (2017) 33–45.
[74] Z. Bai, K.P. Choi, Y. Fujikoshi, Consistency of aic and bic in estimating the number of significant components in high-dimensional principal component analysis, The Annals of Statistics 46 (3) (2018) 1050–1076.
[75] T.A. Nengsih, F. Bertrand, M. Maumy-Bertrand, N. Meyer, Determining the number of components in pls regression on incomplete data set, Statistical Applications in Genetics and Molecular Biology 18 (6) (2019) 20180059.
[76] H. Akaike, Information theory and an extension of the maximum likelihood principle (1998) 199–213.
[77] C.M. Hurvich, C.-L. Tsai, Regression and time series model selection in small samples, Biometrika 76 (2) (1989) 297–307.
[78] E.S. Schwartz, The stochastic behavior of commodity prices: Implications for valuation and hedging, The Journal of finance 52 (3) (1997) 923–973.
[79] E.J. Hannan, B.G. Quinn, The determination of the order of an autoregression, Journal of the Royal Statistical Society: Series B (Methodological) 41 (2) (1979) 190–195.

[80] M.H. Hansen, B. Yu, Model selection and the principle of minimum description length, Journal of the American Statistical Association 96 (454) (2001) 746–774.
[81] J.H. Friedman, Greedy function approximation: A gradient boosting machine, The Annals of Statistics 29 (5) (2001) 1189–1232.
[82] N. Bingham, J. Fry, Regression: Linear Models in Statistics, Springer Undergraduate Mathematics Series, Springer London, 2010.
[83] K.L. Chung, F. AitSahlia, Elementary probability theory: with stochastic processes and an introduction to mathematical finance, Springer Science & Business Media, 2006.
[84] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (200) (1937) 675–701.
[85] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1) (1940) 86–92.
[86] P. Nemenyi, Distribution-free multiple comparisons, in: Biometrics, Vol. 18, International Biometric Soc 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005–2210, 1962, p. 263.
[87] D.A. Belsley, E. Kuh, R.E. Welsch, Regression diagnostics: Identifying influential data and sources of collinearity, John Wiley & Sons, 2005.
[88] S. Chatterjee, A.S. Hadi, Regression analysis by example, John Wiley & Sons, 2013.
[89] D. Montgomery, E. Peck, G.G. Vining, Introducción al análisis de regresión lineal, México: Limusa Wiley.

Laura Fdez-Díaz received her degree in Computer Engineering in Information Technologies from the University of Oviedo, Spain, in 2016 and her masters in Artificial Intelligent from UIMP (Menéndez Pelayo International University) and AEPIA (Spanish Association for Artificial Intelligence), Spain, in 2017. She is a researcher in ArcelorMittal Company and a predoctoral student at Artificial Intelligence Center (University of Oviedo).

José Ramón Quevedo received his MSc and PhD degrees in Computer Science from the University of Oviedo, Spain, in 1997 and 2000, respectively. He is currently a Senior Lecturer and a member of the Artificial Intelligence Center, Gijón. His current research is focused on applying machine learning methods to quantification and to domains where the data distribution varies. He has published many papers in peer-reviewed journals placed at top of the Journal Citation Report. Also, he has authored papers in conferences of recognized prestige.

Elena Montañés received her MSc degree in Mathematics in 1998 and her PhD in Computer Science from the University of Oviedo, Spain, in 2003. She is currently a Senior Lecturer and a member of the Artificial Intelligence Center, Gijón. Her research interests are focused on several machine learning problems. She has authored over 30 papers in peer-reviewed journals and conferences, including IJCAI, ECML, Machine Learning, Information Systems and Pattern Recognition.