Genetic programming for stacked generalization

Illya Bakurov[a], Mauro Castelli[a], Olivier Gau[a], Francesco Fontanella[b], Leonardo Vanneschi[a]

[a] *NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, Lisboa 1070-312, Portugal*

[b] *Dipartimento di Ingegneria Elettrica e dell'Informazione, Università degli Studi di Cassino e del Lazio Meridionale, Cassino (FR), 03043, Italy*

This is the accepted author *manuscript of the following article published by Elsevier:*

Bakurov, I., Castelli, M., Gau, O., Fontanella, F., & Vanneschi, L. (2021). Genetic programming for stacked generalization. Swarm and Evolutionary Computation, 65, 1-14. [100913]. https://doi.org/10.1016/j.swevo.2021.100913

Genetic Programming for Stacked Generalization

Illya Bakurov[a,*], Mauro Castelli[a], Olivier Gau[a], Francesco Fontanella[b], Leonardo Vanneschi[a]

[a]*NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal.*
[b]*Department of Electrical Engineering and Information, Universitá degli Studi di Cassino e del Lazio Meridionale, 03043, Cassino, Italy*

Abstract

In machine learning, ensemble techniques are widely used to improve the performance of both classification and regression systems. They combine the models generated by different learning algorithms, typically trained on different data subsets or with different parameters, to obtain more accu- rate models. Ensemble strategies range from simple voting rules to more complex and effective stacked approaches. They are based on adopting a meta-learner, i.e. a further learning algorithm, and are trained on the predic- tions provided by the single algorithms making up the ensemble. The paper aims at exploiting some of the most recent genetic programming advances in the context of stacked generalization. In particular, we investigate how the evolutionary demes despeciation initialization technique, ϵ-lexicase se- lection, geometric-semantic operators, and semantic stopping criterion, can be effectively used to improve GP-based systems' performance for stacked generalization (a.k.a. stacking). The experiments, performed on a broad set of synthetic and real-world regression problems, confirm the effectiveness of the proposed approach.

Keywords: Genetic Programming, Stacking, Ensemble Learning, Stacked Generalization

*Corresponding author
Email addresses: ibakurov@novaims.unl.pt (Illya Bakurov), mcastelli@novaims.unl.pt (Mauro Castelli), M20170057@novaims.unl.pt (Olivier Gau), fontanella@unicas.it (Francesco Fontanella), lvanneschi@novaims.unl.pt (Leonardo Vanneschi)

1. Introduction

Ensemble learning (EL) is a sub-field of machine learning (ML) inspired by humans' natural tendency to seek and weigh others' opinions before making any important decisions. Under this perspective, EL consists of combining several individual models, called base learners, in a way to produce a model (the ensemble), which is expected to solve a given task better than any of the base learners [26]. Stacked generalization, or simply stacking, consists of training an ensemble from the combined predictions of several ideally heterogeneous base learners. More specifically, it consists of training the base learners to solve the underlying task and subsequently train a meta-learner from their predictions [38].

In this paper, we study Genetic Programming (GP), in the context of regression problems, as a meta-learner. As a case study, we have chosen four different kinds of base learners: Multi-Linear Regression (MLR), Multi-Layer Perceptron Regression (MLPR), Random Forests Regression (RFR), and Support-Vector Regression (SVM). This choice was mainly motivated by the desired properties of the ensembles: the expected diversity of the underlying base learners and, particularly to stacked generalization, the convenience of avoiding to choose an appropriate base learners' type. In this context, we decided to consider four types of ML algorithms, which differ in not only their complexity, but also the regression estimation principles. In such a way, we allow GP to automatically evolve computer programs, having as a terminal set the combined predictions of four heterogeneous base learners. The objective is to improve the ensemble's generalization ability on a given Supervised ML (SML) task. For the sake of simplicity, we will refer to this kind of approach as stacking-GP (S-GP). Our motivation relies on the intrinsic properties of GP. We expect that with properly chosen operators and hyper-parameters, GP is capable of combining base learners in a highly non-linear fashion, which could better exploit their outputs and achieve superior generalization ability.

The idea that has inspired S-GP is not new. To our knowledge, the first related work dates back to 2006 [16] when GP was used to combine 10 Artificial Neural Networks into an ensemble. Since then, several other important contributions were proposed [9, 4, 13, 18]. Nevertheless, we consider that the research in S-GP is still much in demand, mainly for the two following reasons. First, the majority of S-GP contributions are assessed on classification problems, whereas few of the previous works provide a concise

benchmark over regression problems. The second reason has to do with the recent methodological achievements in the GP field: Geometric-Semantic Operators, Semantic Stopping Criterion, ϵ-Lexicase Selection (ϵ-LS) and Evolutionary Demes Despeciation Algorithm (EDDA) are among numerous recent methods that deserve to be attentively studied in the context of S-GP.

The paper is organized as follows: Section 2 introduces the necessary theoretical background; Section 3 discusses existing works on ensemble learning, focusing on stacking, and using GP for ensemble; Section 4 describes the research hypothesis and the proposed approach; and Section 5 presents the studied test problems, our experimental framework, and the obtained results. Finally, Section 7 concludes the work and proposes ideas for future research.

2. Theoretical Background

This section recalls the recent contributions in the field of GP that we considered in this study: Geometric Semantic Genetic Programming, Evolutionary Demes Despeciation Algorithm, ϵ-Lexicase Selection for Regression, and Semantic Stopping Criteria.

2.1. Geometric Semantic Genetic Programming

In the terminology adopted by a considerable number of GP researchers [36], the term *semantics* defines the vector of output values of a candidate solution, calculated on the training observations. Following this definition, a candidate solution obtained by means of GP is a point in a multidimensional space called *semantic space*, where the dimensionality is equal to the number of observations in the training set. Geometric Semantic Genetic Programming (GSGP) is a recently introduced variant of GP where standard crossover and mutation are replaced by the so-called *Geometric Semantic Operators* (GSOs) [22]. These operators induce a unimodal error surface for any SML problem in which the candidate solutions' quality is calculated through a distance metric between target and output values. This work uses the GSOs introduced in [22] and implemented in [10].

The reason for considering GSOs is based on the experimental evidence that GSOs are more effective than standard syntax-based operators in optimizing training data and can automatically constrain overfitting [37].

2.2. Evolutionary Demes Despeciation Algorithm

Initialization plays an important role in any population-based algorithm. In GP, this aspect has particular importance since a wide variety of programs of various sizes and shapes is desirable [19, 25]. With the introduction of GSOs, new initialization techniques have been developed [23, 35]. Here, we focus on the Evolutionary Demes Despeciation Algorithm (EDDA) [35].

In EDDA, the initial GSGP population is generated using the best individuals obtained from a set of independent sub-populations (demes), left to evolve for *few* generations and under different evolutionary conditions [35]. For example, some demes use standard GP operators, while the remaining use GSOs. Besides that, each deme uses distinct search parameters, such as the mutation and crossover probability, the mutation step (in the case of GSM), etc.

EDDA should generate an initial population composed of simultaneously diverse and good quality genetic material. In particular, each individual in the initial population comes from a different evolution history, performed in an independent deme, and evolved under different search parameters. This property makes this initialization technique particularly interesting in the context of this study.

2.3. ϵ-Lexicase Selection for Regression

ϵ-Lexicase Selection [20] is a recently introduced improvement upon already existing Lexicase Selection (LS) of parents in GP [33]. The latter was proposed by Spector in 2012 to provide a simple, representation-independent way to solve multimodal problems without interfering with other GP system components.

The underlying assumption of LS is that the different fitness cases may call for different response modes, i.e., may require the system to respond differently. Under this assumption, extending lexicographic ordering to the fitness cases in randomized order ensures that a good fitness, calculated on a given fitness case, will be rewarded independently on the solution's performance on other fitness cases, while also still rewarding the progress on a larger set of fitness cases (up to the size of the training dataset).

In this study, we considered the ϵ-Lexicase Selection introduced in [20], which demonstrated superior results with respect to other LS variants proposed in the literature.

2.4. Semantic Stopping Criterion

Semantic Stopping Criterion (SSC) is a recently proposed stopping criterion [14] that operates in the context of GSOs and was further extended to neuroevolution [15]. To decide when to terminate the evolution, SSC uses the information gathered from a set of semantic neighbors of the current best solution. The authors proposed two types of SSC: Error Deviation Variation (EDV) and Training Improvement Effectiveness (TIE). The first measures the percentage of those neighbors that, besides being *fitter* than the current-best, have a lower sample standard deviation of the absolute errors. The second measures the percentage of times the underlying semantic variation operator, in our case GSM, can produce a neighbor that is superior to the current best. In this study, we considered both the stopping criteria.

2.5. Stacked Generalization

The field of EL has been extensively researched, and different approaches were proposed so far. In general terms, EL methods differ in the way input data is represented and manipulated within the system, whether the ensemble's base learners can be trained independently from each other, the procedure to make the final prediction, etc. [26, 32, 39]. An extensive review of traditional and state-of-the-art ensemble methods, suitable for both practitioners and beginners on both classification and regression problems, can be found in [30]. There, the authors expose the ensemble methods catego- rized into conventional methods (such as bagging, boosting, and random forest), decomposition methods, negative correlation learning techniques, multi-objective optimization based methods, fuzzy methods, multiple ker- nel learning methods, and deep learning ensemble methods. Moreover, their variations, improvements, and typical applications are discussed along with the underlying theory. Here, we provide a more detailed overview of the Stacked Generalization, that Wolpert [38] proposed in late 1992, because the approach we propose strongly relates to this specific ensemble method. Stacked Generalization, or simply *stacking*, consists of training a meta-learner (a.k.a. level 1 model) from the combined predictions of two or more independently trained base learners (a.k.a. level 0 models). In other words, the predictions obtained from the base learners, are used as inputs for a meta-learner. In practice, the system can have several sequential layers of base learners before executing the final meta-learner. Consequently, stacking is expected to perform at least as good as (if not better than) the best base learner for the underlying problem. Theoretically, there are no restrictions

when defining level 0 and level 1 models - these can belong to any known class of ML models. Moreover, these can either belong to the same model class (but instantiated with distinct hyper-parameters) or can be instances of distinct classes. There is evidence that the most improvement can be observed when stacking together more dissimilar base learners [7].

Breiman's [7] work explores stacking from the perspective of linear models. More specifically, to improve the prediction accuracy, the author used cross-validation data and least squares linear regression under the constraint that all the regression coefficients were non-negative to determine linear combinations of different predictors, CART trees of different complexities, or linear regressions with a different number of variables. The study demonstrated the suitability of linear models as the meta-learner and showed that generally few level 0 models (between two and four) were stacked together.

Continuing Breiman's work, Reid and Grudic [29] explored the regularization penalties, such as Ridge, Lasso and Elastic Net, and showed the methods' appropriateness to handle the highly correlated nature of the base learners, as these try to predict the same output. Another relevant advantage of using regularized models for stacking is the possibility of automatically selecting the most performing subset of the base learners.

Stacking has been successfully applied to solve a wide range of real-world problems. Following the work of Reid and Grudic [29], Larkin and McManus have used stacking by combining 20 distinct types of base learners by means of the Empirical Bayesian Elastic Net to model wine preferences [21]. Alhalaseh et al. [5] applied stacking consisting of one SVR combining the outputs of one MLPR and four SVR machines (with several kernel functions) in the context of electrical load forecasting. In the work of Rahman et al. [27] stacking was applied in the field of information security to improve malicious URLs' classification; in their stacking scheme authors have used the Random Forests, Decision Trees, Multi-Layer Perceptrons, Support Vector Machines, Stochastic Gradient Descent and the Gaussian Naive Bayes as level 0 classifiers, that were subsequently combined with the Extreme Gradient Boosting (level 1 classier).

As can be concluded from the aforementioned examples, the practitioners do not follow particular guidelines regarding level 1 models' selection: some use regularized regression, some use the same type of models as the base learners and some use boosted ensembles. This mainly justified our choice of the meta-learners against which we have compared our approach (see section 5.2.2). What regards the base learners, we conclude from the discussion

the common trend of combining few level 0 heterogeneous models.

3. Previous and Related GP Uses as a Meta-Learning Technique

The idea of using GP as an automatic EL technique is not new. To our knowledge, the first evidence comes from 2006 [16], when GP was used to combine 10 ANNs into an ensemble. In [9], authors compared an equivalent approach against 3 ensemble approaches based on Genetic Algorithms (GAs). The experimental results involving four synthetic and one real-world symbolic regression tasks confirmed the preeminence of GP-based ensemble against not only against the best base learner, but also the 3 different types of GA-based ensembles. In [13], the authors extended using GP to learn ensemble policies by using up to 6 heterogeneous base learners, some of which ensembles themselves (such as the Random Forests). Additionally, in an attempt to increase the synergistic effect of combining different base learners into an ensemble policy, the authors proposed a novel operator entitled "decision expression", which splits the input space into sub-spaces, that can then be handled by different sub-policies. Despite the authors' expectations, the proposed approach performed significantly better than the best base learner only in one of the 10 studied problems. In their work, the authors pointed out several limitations, namely the overfitting perceived by how GP simply selects from the base learners instead of learning a complex and meaningful policy and the absence of real-world benchmark problems. The latter translates in the absence of potentially challenging fitness landscapes for the base learners, a scenario that seems suitable for the method they proposed.

In [28], the authors proposed an extension to GSGP, called Universal-GP (U-GP) in which some of the initial individuals are created by means of other ML techniques. More specifically, the authors demonstrated that it is possible to obtain a significant improvement over the standard ramped half-and-half (RHH) initialization. They did so by producing semantics using fundamentally distinct ML models, in this context seen as the base learners, and including them in the initial population of GSGP along with random initial programs. After identifying the system's sensibility toward overfitting of the base learners, the authors have also introduced the pre-evolutionary selection procedure, referred to as PESP. This procedure attempts to exclude from the evolution those base learners that are unable to individually achieve good generalization performance after training and after a short run of GSGP.

This approach allowed the system to achieve superior performance when compared to standard GSGP on four real-world regression problems.

Karakatič and Podgorelec [17] proposed a novel GP method that combined classification trees' induction with GP and AdaBoost to weigh data instances and individual trees through the evolution. Based on 10 UCI classification benchmarks, the empirical evidences pointed out that the proposed method, called Genetic Programming with AdaBoost (GPAB), achieved superior performance when compared with Random Forest and AdaBoost.

The authors in [31] proposed a novel GP method, called Ensemble GP (eGP) that follows the procedural steps of other GP systems, but with differences in the population's structure and the way it is handled in the system. More specifically, eGP is composed of two sub-populations, trees and forests, such that each sub-population uses its own fitness function and genetic operators. The approach involves the sub-sampling of both observations and features and the authors describe it as co-evolutionary, cooperative and compositional. The experimental evidence, based on eight binary classification problems, points to eGP's superiority over the standard GP both in terms of classification accuracy and the models' size; however, other methods such as M3GP and XGBoost showed better performance than eGP.

In Evans [12], among many other contributions, the authors propose a novel stacking method, based on GP, to automatically evolve heterogeneous ensembles of classification algorithms. Under this approach, the ensembles are represented as tree structures where the base learners and the parameters represent the leaves (i.e, the terminals), and the plurality voting represents the function-set (i.e., the intermediate nodes). This representation means that GP can explore the search space of potential trees since the number of potential trees makes it unfeasible to perform an exhaustive search. The proposed stacking approach was compared against the underlying base learners and 3 other ensemble methods on 11 datasets retrieved from the UCI repository. The empirical evidence shows the proposed approach's superiority on all the datasets. Although the proposed approach shows several similarities with the one presented in this document, it is worth highlighting several significant differences: the function-set is very limited, as it is made of plurality voting only, and the proposed approach concerts classification problems only.

The work of Bhowan et al. [6] proposes to improve ensembles' performance on unbalanced classification problems by using a two-step approach to evolve ensembles using GP. Concretely, authors propose to 1) use the multi-objective GP (MOGP) to evolve an accurate and diverse Pareto-approximated front

of classifiers (the base learners) by trading-off the minority and the majority class during learning, and 2) to use GP to automatically combine the multiple Pareto-approximated front members into a single composite GP solution to represent the ensemble. Authors assessed their approach on 12 real-world (binary) unbalanced data sets showing that it performs as well as, or better than, the state-of-the-art techniques and traditional ensemble approaches, such as bagging and boosting ensemble methods.

The work of Castelli et al. [11] proposed an efficient strategy for building ensembles of GP models, mainly inspired by the work of [35]. In the latter, the authors proposed a technique that blends, through GSGP, the elite individuals obtained from running independent sub-populations that vary in terms of the variation operators and the associated probabilities. In this context, the authors of [11] proposed two different pruning criteria to improve the diversity of the GP models used to constitute the ensembles' base-learners: one based on the correlation, another based on the entropy. The experimental results did not allow the authors to draw a general conclusion on the superiority of the proposed criteria. Nonetheless, it was found that considering a similarity criterion when constructing a GP ensemble can help to maintain the generalization ability of the resulting model while reducing the computational effort in some problems.

All the aforementioned contributions, except [28, 17, 31], can be categorized as stacking, where GP assumes the role of a meta-learner from the combined predictions of several BLs. For this reason and from this moment on, we will entitle these kinds of approaches as Stacking GP (S-GP).

At this point, it is important to highlight that, within the aforementioned contributions, we did not find attentive and focused studies involving recent methodological achievements in the field of GP. Moreover, while the vast majority of studies involves classification problems, only a few works consider regression tasks. Motivated by this niche in the scientific panorama, we decided to perform a broad exploration of some of the recent achievements (such as the EDDA technique, the ϵ-lexicase selection, the GSOs, and the SSC), in the context of S-GP applied to regression problems.

It is important to note that, although the approach of [11] was assessed on regression problems and involves GSOs, the work presents several significant key-dissimilarities which make it incomparable with the approach proposed in this paper. First, the authors tested their method on a limited number of problems, whereas our work considers 11 benchmark problems. Second, the authors built the ensembles involving only GP-derived base-learners, while

in this study distinct base-learners are used. Third, they employed a standard GP approach as the meta-learner, whereas we also consider and analyze GSGP for this role. Fourth, the authors assessed the use of pruning techniques in the context of parallel populations' evolution, which is not the central focus of our work.

4. The Proposed Approach

4.1. Individuals' Representation in S-GP

In the context of S-GP, individuals' representation is similar to traditional GP; the only difference is the set of terminals T, which is composed of the predicted outputs ($\hat{y}$) of the base learners trained to solve a given SML problem. In our experiments, the set of base learners is made of Multi-Linear Regression (MLR), Multi-Layer Perceptron Regression (MLPR), Random Forests Regression (RFR) and Support-Vector Regression (SVR); thus, the formal definition of our terminal set is given by $T = \{\hat{y}_{MLR},\ \hat{y}_{MLPR},\ \hat{y}_{RFR},\ \hat{y}_{SVR}\}$. The function set F includes the traditional arithmetic operators, the average, the minimum and maximum between two input semantic vectors, all taken at each index. All the elements of F admit two operands. Additionally, we have included the so-called decision expression Flasch et al. Flasch et al. [13] introduced. Each decision expression application splits the input space into two sub-spaces (binary split), which will then be approximated by different fusion policies. The input space is split axis-parallel by randomly selecting a threshold at a randomly chosen input feature. Then, each sub-set of the space will be approximated by the respective sub-policies. In this sense, the decision operator resembles the functioning of the Decision Trees' binary split, but with random feature and threshold selection. Thus, our primitive set is given by: $F = \{+,\ -,\ *,\ /\ ,\ D,\ AVG,\ MIN,\ MAX\}$, where D stands for the aforementioned decision expression and AVG, MIN and MAX stand for average, minimum and maximum between two considered semantic vectors taken at each index. Notice that all the elements of F admit exactly two operands. A possible individual resulting from composing such program elements is represented in Figure 1. From the figure, one can see that the base learners, represented with white nodes, are combined in a non-linear fashion. The decision expression, represented as a gray node with inscription $D_{X_1,4.1}$, performs a binary split of the search-space on the input feature X_1 at the

threshold 4.1. This means that those data instances whose values across feature X_1 are below 4.1 will be handled by the sub-tree on the left ($\hat{y}_{MLP}/\hat{y}_{LR}$), whereas those whose values are above 4.1 will be handled by the sub-tree on the right ($\hat{y}_{SVM}$).

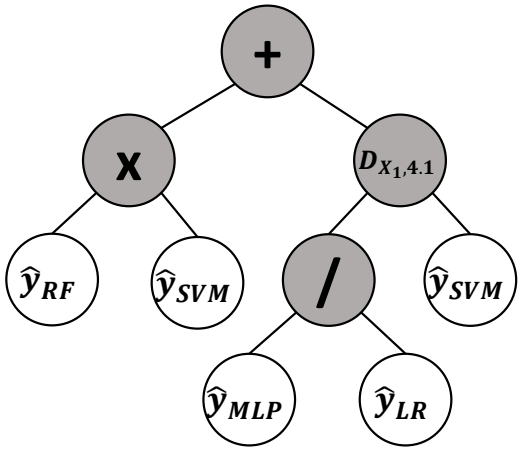

Figure 1: Example of a tree-based representation of an individual in the context of S-GP. The white nodes represent the terminals, whereas gray nodes represent the functions.

4.2. Base learners' manipulation

An important technical pre-requisite of the S-GP system is to train the base learners and to extract their semantics to built T. Under this perspective, in our experiments, T holds four n dimensional vectors, one for each base learner, where n stands for the number of instances in the training data. To assess the system's generalization ability, the evolved trees are provided the base learners' semantics calculated on unseen data. The pseudo-code in Figure 2 provides a detailed explanation of how the approach works.

4.3. Geometric Properties of the Semantic Space

Considering geometric properties of the semantic space, it is clear that using combined outputs of other models as the input terminals of the GSGP system, makes the convex hull of the initial GP population's semantics at least closer to the global optima; in the best case, it will "contain" the

Let $[BLs]$ be the user-specified base learners to stack by means of a GP system, X the input data set and y the respective real-valued target:

1. split X and y into training and unseen partitions: X_{train}, X_{unseen}, y_{train} and y_{unseen}.
2. create two empty lists, X'_{train} and X'_{unseen}, to store the base learners' predicted outputs.
3. *for bl in* $[BLs]$:
   (a) using X_{train} and y_{train}, apply a k-fold cross-validation procedure to search for the best set of the hyper-parameters of *bl*;
   (b) create an instance of *bl* (called *bl'*), under the best-found parameters' set.
   (c) extract the predictions of *bl'* on X_{train} and X_{unseen} and append them to X'_{train} and X'_{unseen}, respectively.
4. compose the set of terminals T for the GP algorithm as the union between the $len([BLs])$ input features and some user-specified constants (a.k.a. the constants' set).
5. execute the GP system on the training data composed of X'_{train} and y_{train} and use the unseen data, composed by X'_{test} and y_{test}, to assess system's generalization ability.

Figure 2: Pseudo-code for the proposed S-GP method.

target's semantics. For these reasons, assessing GSOs, different crossover-mutation probabilities and EDDA initialization is of special concern in this work. We have also included in our study the $\epsilon - Lexicase$ selection as it was specially designed for the regression problems.

5. Experimental Environment

The section aims to provide this study's objectives and to present the experimental parameters and configurations.

5.1. Objectives

The experimental environment is built upon 11 problems, seven of which are synthetic and four are real-world regression problems. The experiments

were conducted to accomplish the following objectives:

1. provide a general recommendation for the hyper-parameters of S-GP;
2. assess the impact of the tuning of the base learners on the S-GP system;
3. compare the performance of S-GP against the underlying base learners;
4. compare the performance of S-GP against other ensemble methods, including other stacked approaches that use different meta-learners.

5.2. Experimental Settings

This section presents the experimental settings. Table 5.2.1 provides a complete enumeration of the experimental parameters of the S-GP system. It is important to highlight that the parameters were equally applied on all the problems, both synthetic and real-world. Section 5.2.1, presents a detailed description of the parameters summarized in the aforementioned table. Section 5.2.3 explains how the data was partitioned and manipulated during the experiments.

5.2.1. Hyper-Parameters of S-GP system

Taking into account the employed algorithms' stochastic nature and that results dependent on the data partitions' volatility, we repeated the experiments 60 times (runs), each with a different seed for the pseudo-random numbers generator, used to partition the data, and to initialize and execute the algorithms. Throughout our experiments, we guaranteed an equal computational effort for each experiment within the S-GP system. Considering the EDDA technique implies significant structural changes in the search procedure and since we aim at comparing the traditional RHH initialization against the EDDA technique, we had to articulate the computational effort differently for each technique, although it was equal for both techniques. For a system that uses the RHH initialization technique, we have specified 500 generations for a population size of 140 (i.e., 500x140 = 70000 fitness evaluations). Whereas an S-GP system initialized with the EDDA technique with maturation of 5 generations ($EDDA_5 - 50\%$), was executed for 200 generations with a population size of 100 (i.e., 100x5x100 + 200x100 = 70000 fitness evaluations). The parameters for EDDA were chosen according to the experimental findings in [35].

In [28], the authors proposed the so-called pre-evolutionary selection procedure to avoid overfitting their system. Thus, we decided to assess the impact of the base learners' tuning before feeding their semantics into S-GP system. A detailed explanation of this process can be found in the Section 5.2.3. Table A.10 enumerates the range of the base learners' hyper-parameters that were explored. To analyze the usefulness of this approach in the context of S-GP, we compared the base learners's tuning against the hyper-parameters that are provided in the computing library by default[1] (from the table $1 BLs\ tuning = False$).

Besides the EDDA initialization, the experiments have also involved the assessing of other methodological achievements in the field of GP: ϵ-LS, GSOs, and SSC; they were compared to traditional Tournament Selection, standard GP variation operators and stopping criteria. We performed the comparison by evaluating the performance of all the possible combinations. It is important to clarify that, by standard GP variation operators we are referring to the Swap Crossover (a.k.a. Sub-tree Crossover) and the Sub-tree Mutation as proposed in [19]. Similarly, by *traditional stopping criteria* we are referring either to a pre-specified maximum number of generations or to the stopping point inferred from the fitness calculated on validation partition (i.e., by estimating the point from which further induction does not allow higher generalization); the two criteria are mentioned in the table as *№generations* and *validation*. The stopping criteria were compared after executing the experiments for the number of generations specified in Table 1. As it traditionally happens in GP, parents' reproduction was not allowed, which means that the probability of applying crossover $P(C)$ is the opposite of the mutation's probability and vice-versa. Notice that the terminal and functional sets were already introduced in Section 4.1.

5.2.2. Other ensemble methods and their parameters

Our study aims to compare the S-GP system not only against the underlying base learners, but also with the state-of-the-art ensemble methods. To validate the proposed approach, we compared it against 3 popular boosting methods: the Adaboost Regression (ABR), the Extreme Gradient Boosting Machine Regressor (XGBR) and the Light Gradient Boosted Machine Regressor (LGBMR). Moreover, we included 8 different meta-learners that

[1] In our experiments, we have used scikit-learn 0.21.0 [24]

Parameters	**Values**
№runs	60
№generations	500_{RHH}, $200_{EDDA_5-50\%}$
Population size	140_{RHH}, $100_{EDDA_5-50\%}$
Function set	+, -, x, /, AVG, MAX, MIN, D
Terminals set	MLR, SVR, RFR, MLPR
BLs tuning	True, False
Initialization	RHH, $EDDA_5 - 50\%$
Selection	TS, ϵ-LS
Variation Operators	{Swap Crossover, Sub-tree Mutation}, {GSC, GSM}
$P(C)$	0, 0.2, 0.8, 1.0
$P(M)$	$1 - P(C)$
Stopping criteria	TIE, EDV, validation, №generations

Table 1: Enumeration of hyper-parameters. Notice that D stands for the decision expression operator proposed in [13]; *BLs tuning* indicates if the base learners' hyper-parameters were tuned or kept at their defaults; ϵ-*LS* represents the selection algorithm proposed in [20]; $P(C)$ and $P(M)$ indicate the crossover and mutation probabilities, respectively, and *validation* stands for the stopping criteria that operates upon the fitness calculated from validation partition by estimating the point from which further induction does not allow higher generalization.

were used for stacking the same base learners as the ones used by S-GP system: stacking with Multi-Linear Regression (S-MLR), Support Vector Regression (S-SVR), Multi-Linear Perceptron Regression (S-MLPR), Elastic Net Regression (S-ENETR), Random Forest Regression (S-RFR), Adaboost Regression (S-ABR), Extreme Gradient Boosting Machine Regressor (S-XGBR) and Light Gradient Boosted Machine Regressor (S-LGBMR). Table A.10 enumerates the hyper-parameters that were explored for each of the presented ensemble methods. Notice that the Random Forests Regression (RFR) is already included in the experiments as it was one of the base learners in S-GP system.

5.2.3. Cross-Validation

In this work, the cross-validation assumes two forms, depending on whether the S-GP system operates with the tuned base learners. In the case when the hyper-parameters of the base learners are not intended to be tuned, the data is operated in terms of the traditional Monte-Carlo cross-validation scheme. In other words, the training dataset is obtained by sampling 70%

of the data at random (without replacement), while the remaining data observations constitute the unseen set. The process is iterated with a different random seed at each new run, meaning that exactly the same data point can appear in the unseen partition zero or *number of runs* times.

In the case when the hyper-parameters of the base learners are intended to be tuned, the data is operated in terms of a two-step cross-validation scheme, similar to the so-called *nested K-Fold cross-validation* Re [28]; we used 10 folds in our experiments.

5.3. Test Problems

In this sub-section, the reader can find a detailed characterization of the experimental problems, which are grouped into two sub-sections: one describes synthetically-generated problems while other covers the real-world problems.

5.3.1. Synthetically-generated problems

Table 2 contains the mathematical formulation for the seven synthetic regression problems used in this study, while Table 3 provides the respective bounds (i.e., the domain) at each dimension. It is worth highlighting that these functions were studied in a two-dimensional input space. For each of these problems, we generated 200 two-dimensional data points under Continuous Uniform Distribution, the parameters which were chosen from Table 3. Then, these points were used as the input for the functions defined in Table 2 to generate the respective target vectors.

5.3.2. Real-world problems

Table 4 summarizes the four real-world regression problems. The *Boston* problem [1] is from the field of real estate analysis and consists of predicting the value of owner-occupied homes as a function of its geographic, socioeconomic, environmental and housing characteristics. The *Diabetes* problem [3] contains blood pressure and demographic data of 442 individuals who have diabetes and the target value is a quantitative measure of the disease's progression one year after the baseline measurement. The *PPB* problem [2] comes from pharmacokinetics field and consists of predicting the percentage of the initial drug dose that binds plasma proteins as a function of its 626 molecular descriptors. Finally, the *Parkinson* problem [34] is mostly composed of biomedical voice measurements from 42 people who, at the time of data-collection, had Parkinson's disease at its early stage. They were

Name	$f(x_1, x_2)$
Branin	$\left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right) cos(x_1) + 10$
Discus	$x_1^2 10^6 + x_2^2$
Griewank	$1 + \frac{1}{4000}\ x_1^2 + \frac{1}{4000}\ x_2^2 - \cos(x_1\ )\ \cos(\frac{1}{2}\ x_2\ \sqrt{2})$
Kotanchek	$\frac{e^{-(x_1-1)^2}}{3.2+(x_2-2.5)^2}$
Mexican Hat	$\frac{1}{\pi\sigma^2}1 - \frac{1}{2}\frac{x_1^2+x_2^2}{\sigma^2}e^{-\frac{x_1^2+x_2^2}{2\sigma^2}}$
Rastrigin	$10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi\ x_i\ )\ ]$
Weierstrass	$\sum_{i=1}^{d}\sum_{k=1}^{kmax}\frac{1}{2}^k \cos\left(2\pi 3^k(x_i + \frac{1}{2})\right) - d\sum_{k=1}^{kmax}\frac{1}{2}^k \cos\left(2\pi 3^k \frac{1}{2}\right)$

Table 2: Mathematical formulation of synthetic regression problems used in this study. The problems are listed in alphabetical order.

recruited to a six-month trial of a telemonitoring biomedical speech recording device for remote symptom progression monitoring. The objective was to predict Parkinson's symptom progression measured through total Unified Parkinsons Disease Rating Scale (UPDRS).

6. Experimental Results

This section presents the experimental results and discusses the main findings.

6.1. The hyper-parameters of the S-GP system

To understand which combinations of hyper-parameters allow the S-GP system to achieve the highest performance, we calculated the average of the performance ranks. More specifically, for each problem, we ranked all the 256 combinations of the hyper-parameters by the average RMSE on the unseen set; then, we computed their ranks' average across all the problems, and the real-world and the synthetic problems separately. Finally, we sorted the average ranks in ascending order and analyzed the 5 top ranked sets of the hyper-parameters. The results are presented in Table 5. We used this approach to quantify the different parameters' combinations because all

	Bound			
	Upper		Lower	
Problem	**x1**	**x2**	**x1**	**x2**
Branin	-5	0	10	15
Discus	-32.786	-32.786	32.786	32.786
Griewank	-600	-600	600	600
Kotanchek	-2	-1	7	3
Mexican Hat	-5	-5	5	5
Rastrigin	-5	-5	-5	5
Weierstrass	-0.5	-0.5	0.5	0.5

Table 3: Enumeration of search space's boundaries (the domain) for each function. Note that columns *Lower* and *Upper* stand for lower and upper bounds of the two dimensional search space respectively.

Dataset (ID)	#Features	#Instances	Field
Boston	13	506	Real Estate
Diabetes	10	442	Medicine
PPB	626	131	Pharmacokinetics
Parkinson	20	5876	Medicine

Table 4: Description of real-world regression problems used in the experiments. For each problem, the name (ID), number of input features (#Features), data instances (#Instances) and field of application (Field) is presented.

the problems have different errors magnitudes. Notice that the smaller the average rank, the higher the average performance.

From analyzing Table 5, it is apparent that, when the system's performance is assessed across all the problems, the S-GP system is expected to achieve the highest performance when the initialization is EDDA, the selection is Tournament, the variation is driven by GSC only, the stopping criteria is TIE and the base learners are tuned. This hyper-parameters set combination empirically confirms our previous speculations about the Convex-Hull and how GP trees made of base learners might surround the global optima. In such a way, through GSO's, one can effectively approximate trees' semantics towards the target semantics. The first four top performing parameter sets only differ in the stopping criterion. The last parameters set significantly deviates from the trend: the S-GP is initialized using RHH and is

Type	Initialization	Selection	Operators	P(C)	Stopping Criteria	Tuning BLs	AVG Rank
All	EDDA	TS	GSOs	100%	TIE	TRUE	20.55
	EDDA	TS	GSOs	100%	Validation	TRUE	25.00
	EDDA	TS	GSOs	100%	EDV	TRUE	27.27
	EDDA	TS	GSOs	100%	№generations	TRUE	28.09
	RHH	TS	GSOs	0%	TIE	TRUE	53.36
Synthetic	EDDA	TS	GSOs	100%	TIE	TRUE	24.00
	EDDA	TS	GSOs	100%	Validation	TRUE	31.43
	EDDA	TS	GSOs	100%	EDV	TRUE	34.86
	EDDA	TS	GSOs	100%	№generations	TRUE	35.71
	EDDA	TS	GSOs	80%	TIE	FALSE	36.00
Real-world	EDDA	TS	GSOs	100%	Validation	TRUE	13.75
	EDDA	TS	GSOs	100%	EDV	TRUE	14.00
	EDDA	TS	GSOs	100%	TIE	TRUE	14.50
	EDDA	TS	GSOs	100%	№generations	TRUE	14.75
	EDDA	TS	GSOs	80%	TIE	TRUE	24.25

Table 5: Average of the performance rank (from the table, *AVG Rank*), computed across all the problems, and real-world and synthetic problems separately. The table reports the five top combinations of the hyper-parameters, grouped by problems' type. The column *Variation* stands for the variation operators used to conduct the search, *P(C)* represents the probability of applying the crossover, which is the opposite of mutation's and *Tuning BLs* symbolizes whether the base learners were tuned (TRUE) or not (FALSE).

driven by GSM; moreover, it also deviates in terms of average performance rank about three times from the best set of the hyper-parameters, which is why we consider it as not deserving significant attention.

When looking at the system's performance on synthetic problems, the previous findings can be confirmed. There is just one set of the hyper-parameters in the top 5 which slightly deviates from the trend: when the initialization is EDDA, the selection is tournament, variation is driven by GSOs with a high crossover rate ($P(C) = 80\%$), the stopping criterion is TIE and the base learners are not tuned. In this case, we could conclude that with some mutation, the system's performance with *default* base learners can be almost as good as when they are tuned. Also in this case, the mutation might play an important role in *approximate* the Convex Hull identified by the population towards global optima.

When looking at the results obtained in real-world problems, the previous findings can be confirmed once again. The S-GS system achieves the highest performance when the initialization is EDDA, the selection is Tournament, variation is driven by GSC only and the base learners are tuned. The only difference consists of the stopping criterion, which is based on based on validation error.

In such a way, we were able to fulfill two of our research objectives: to

provide a general recommendation for the systems' hyper parameters and to verify whether the base learners should be tuned.

6.2. Comparison of S-GP against other SML algorithms

In this section, we provide the results of comparison of S-GP system against the underlying base learners and other ensemble methods, including stacked generalization by means of different meta-learners. Section 6.2.1 provides a visual demonstration of the competitiveness of the proposed system and shows how the output values of the S-GP system correlate with the target. Section 6.2.2, provides the results of statistical assessment. Notice that the proposed of S-GP considered in this section was parameterized after the findings in 5.

6.2.1. Visual analysis

Figures 3- 13 show how the different SML algorithms perform on different data partitions at every considered problem. Every sub-figure regards one different data partition: training on the right, and unseen on the left. The vertical axis shows algorithms' mean squared error (MSE) on a given partition, whereas the horizontal axis identifies the algorithms. The first four boxes of every sub-figure (painted with lavender) regard the four base learners considered in this study, labeled with a prefix BL": the Multi-Linear Regression (BL-MLR), the Support Vector Regression (BL-SVR), the Multi-Linear Perceptron Regression (BL-MLPR) and the Random Forests Regression (BL-RFR). The next three boxes (painted with light sky-blue) regard three state-of-the-art boosting methods: the Adaboost Regression (ABR), the Extreme Gradient Boosting Machine Regressor (XGBR) and the Light Gradient Boosted Machine Regressor (LGBMR). The subsequent eight boxes (painted with royal blue) represent the different meta-learners that were used for stacking the base learners: stacking with Multi-Linear Regression (S-MLR), Support Vector Regression (S-SVR), Multi-Linear Perceptron Regression (S-MLPR), Elastic Net Regression (S-ENETR), Random Forest Regression (S-RFR), Adaboost Regression (S-ABR), Extreme Gradient Boosting Machine Regressor (S-XGBR) and Light Gradient Boosted Machine Regressor (S-LGBMR). Finally, the last box (painted with red) regards the proposed S-GP system.

The problem-wise analysis of the Figures 3- 13 shows a clear pattern: the proposed system tends to outperform the underlying base learners and exhibits at least as good performance as the concurrent methods including

other stacking with other kinds of meta-learners. When analyzing the results on the training set, we must highlight the S-GP's ability to fit the data almost perfectly; however, this is not always reflected on the test data. Nevertheless, it is necessary to highlight the proposed system's high stability when compared to other methods.

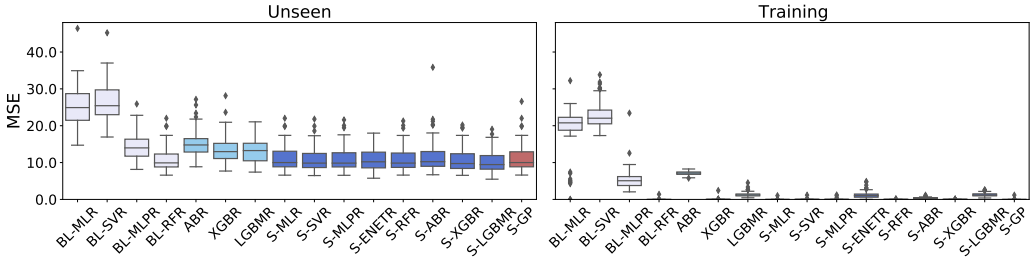

Figure 3: Box-plots on Boston

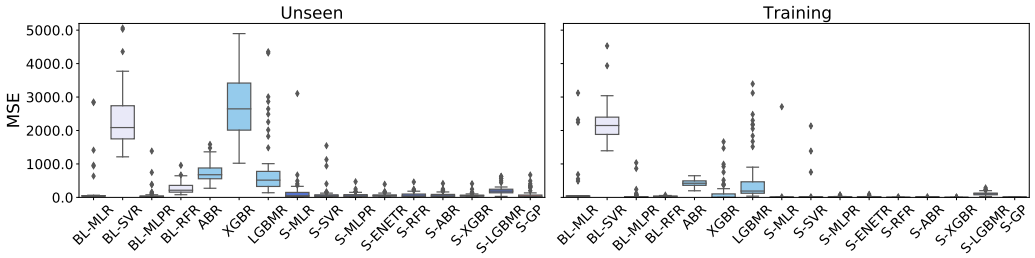

Figure 4: Box-plot on Branin

Table 6 exhibits 11 scatter-plots, one for each problem, where the proposed S-GP system's predicted values are compared against the real ones (y_{pred} and y_{true} on the vertical and horizontal axis, respectively). The comparison is presented for both training and test partitions (gray and red points, respectively). Moreover the plots contain the Pearson's Linear Correlation Coefficient (PLCC) between y_{pred} and y_{true}, calculated at each partition.

From Table 6, we can see that the proposed S-GP system exhibits a *high* correlation with the target on both training and unseen data and that the difference in terms of the coefficients does not vary significantly across the data partitions. The only datasets presenting a different behavior are Diabetes and PPB, for which the PLCC values on the training set are $\tilde{0}.99$ and 1.0, respectively, while on the unseen data the PLCC values are 0.67

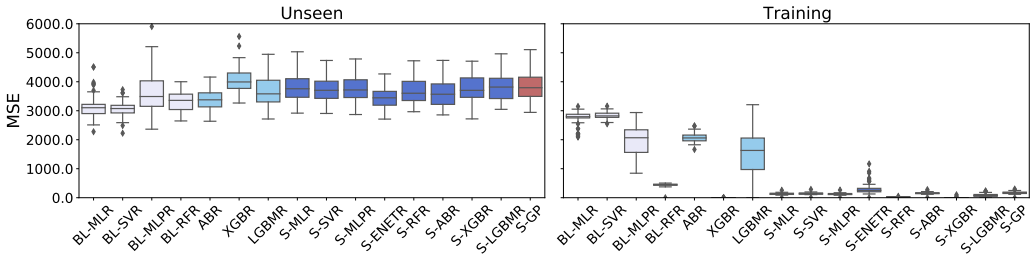

Figure 5: Box-plot on Diabetes

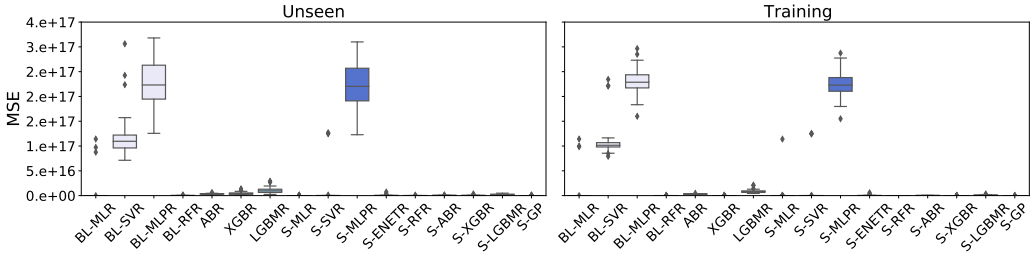

Figure 6: Box-plot on Discus

and 0.359, respectively. Nevertheless, it is worth pointing out that these two datasets can be considered challenging in terms of generalization for not only the S-GP, but also the other SML methods (as confirmed in Figures 5 and 11).

6.2.2. Statistical assessment

In this section, we present the results of the statistical analysis. The median was preferred over the mean because of its higher robustness to the outliers. For all the reported experiments, tests of statistical significance were performed. More specifically, the Kolmogorov-Smirnov test has shown that, the distribution of the differences between pairs cannot be assumed to be normally distributed, as such a rank-based statistic has been used. The Wilcoxon signed-rank test for pairwise data comparison with Bonferroni correction has been used under the alternative hypothesis that the samples do not have equal medians. Due to the extensive Table sizes, we do not report all the results. Nevertheless, we provide the main findings by reporting those test cases that were found to be statistically significant; the full tables can be found by following this link.

Table 8) shows all the combinations of {Problem, BL} (the first two

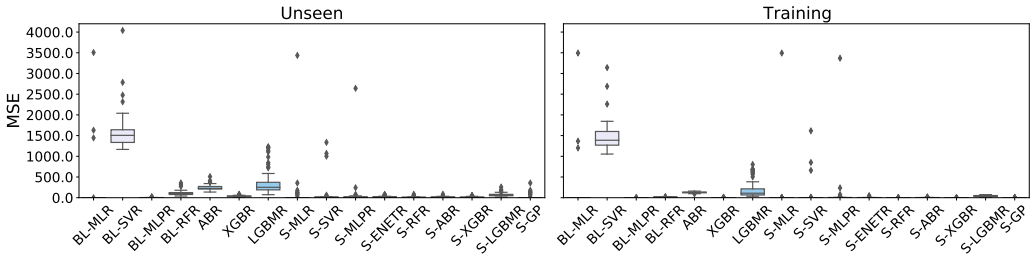

Figure 7: Box-plot on Griewank

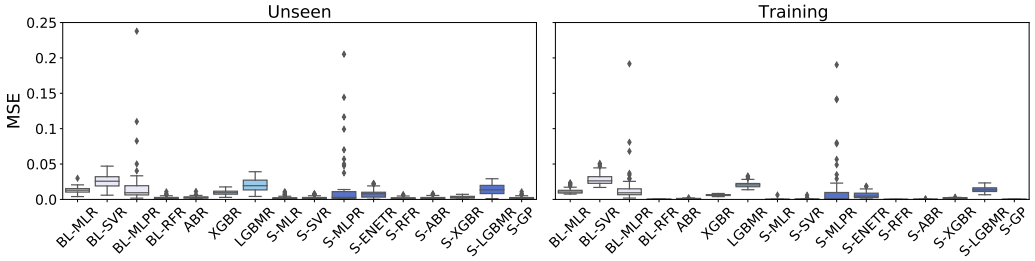

Figure 8: Box-plot on Kotanchek

columns) where a given stacked-generalization's meta-learner was found to be statistically worse than a given base learner (BL) in terms of the generalization ability. The meta-learners are reported in the columns 3-11 and each time a base learner statistically outperformed any of them at a given problem, the Table reports a p-value in the respective cell. Since this statistical procedure involved 396 tests (i.e., 11 problems, nine stacking methods and four base learners), the p-value was adjusted from 0.05 to 1.26E-4. The Tables last row summarizes the number of times an underlying meta-learner was found to be statistically outperformed by the respective base learners across all the problems.

By looking at the counts, we can clearly see that, in general terms, the S-MLR and the proposed S-GP happen to outperform the underlying base learners more often than any other concurrent meta-learning method. Also, we can clearly see that the RFR happens to be the most challenging BL to improve in the context of stacking. In three out of the 11 problems - Diabetes, Griewank and PPB - none of the stacking methods could statistically outperform at least one of the base-learners, which is an interesting finding regarding the problems' difficulty for stacking.

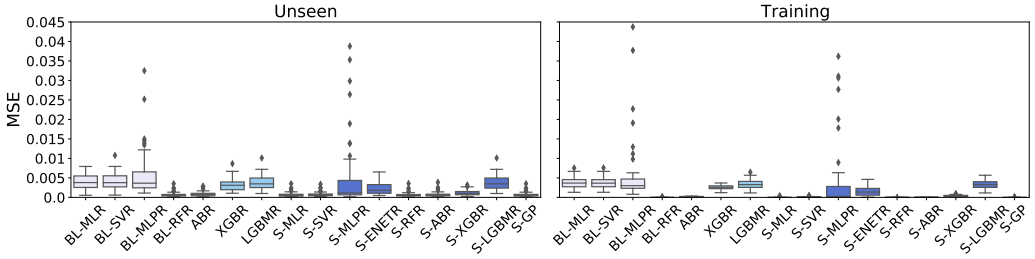

Figure 9: Box-plot on Mexican hat

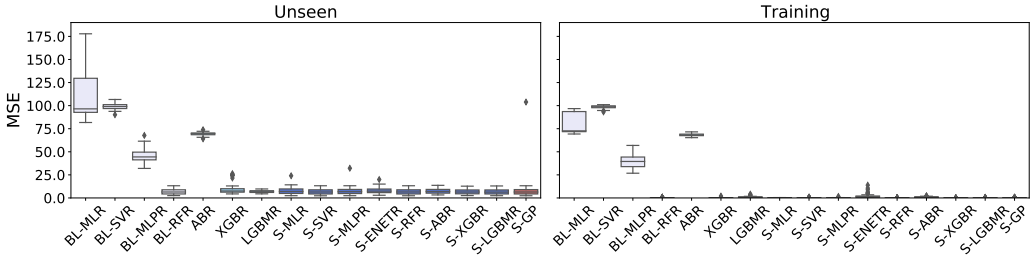

Figure 10: Box-plot on Parkinson

Table 8 reports those cases where the concurrent stacking methods statistically outperformed the S-GP in terms of generalization ability. Since this statistical procedure involved 88 individual tests (i.e., 11 problems and eight concurrent stacking methods), the p-value was adjusted from 0.05 to 5.68E-4. The columns *Problem* and *Sample B* identify the problem and the concurrent stacking method for which the proposed S-GP was found to be statistically worse. The following two columns, $\widetilde{SampleA}$ and $\widetilde{SampleB}$ represent the median of the MSE (calculated on the test partitions across 60 runs), for the proposed S-GP method and the concurrent stacking methods, respectively. The last column reports the p-values for each test.

By analyzing the Table, we can clearly see that there are six out of 11 problems for which at least one of the eight concurrent meta-learners outperformed the S-GP system; in two of the problems (Parkinson and Kotanchek), only one concurrent method outperformed the proposed S-GP system (these were S-LGBMR and S-RFR, respectively). Among all the concurrent methods, S-GP is most frequently outperformed by the S-RFR (in four out of 11 problems). Based on this analysis, it is possible to state that the proposed S-GP method is a viable alternative to the considered stacking methods,

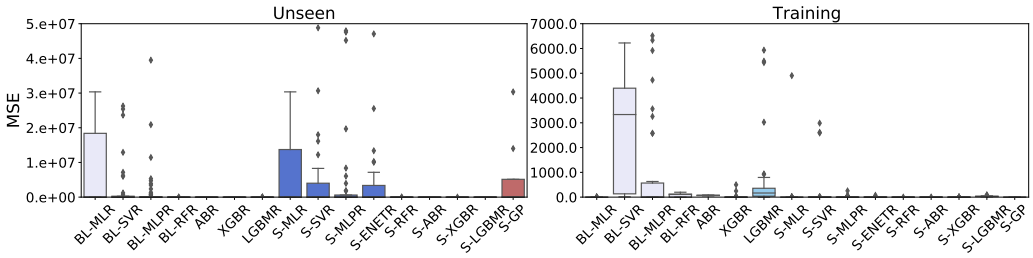

Figure 11: Box-plot on PPB

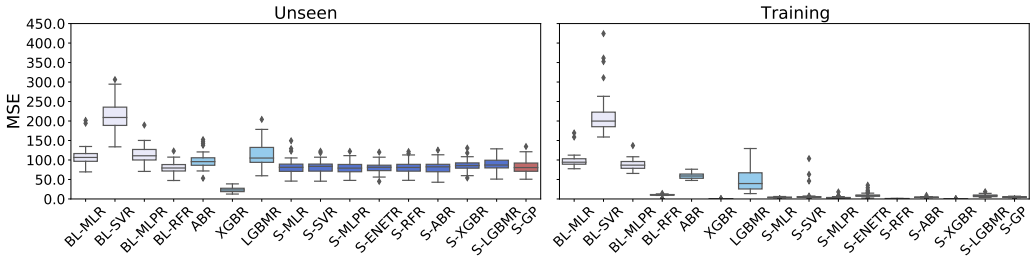

Figure 12: Box-plot on Rastrigin

commonly used in the scientific community.

Table 9 shows, with bold values, all the combinations of the type {Problem, Ensemble} (the first two columns) where a given meta-learner was found to be statistically worse in terms of generalization ability. Since this statistical procedure involved 297 tests (i.e., 11 problems, nine stacking and three ensemble methods), the p-value was adjusted from 0.05 to 1.68E-4. The meta-learners are reported in the columns 3-11 and each time any of them was found to be statistically outperformed by a some base learner at a given Problem, the table contains the p-value of that test in the respective cell. It is worth mentioning that the RFR is not present in this table because, under the light of our experiments, it assumes the role of a base-learner and was already compared to the stacked generalization in Table 7.

By looking at the counts, we can see that, in general terms, the proposed S-GP system generalizes as well (or better than) the majority of the concurrent stacking methods, when compared to the state-of-the-art boosted ensembles. Although the S-RFR, S-ABR, and S-XGBMR present lower counts for the combination {Problem, Ensemble}, the number of problems where S-GP happens to be statistically worse than any of the boosted ensembles is

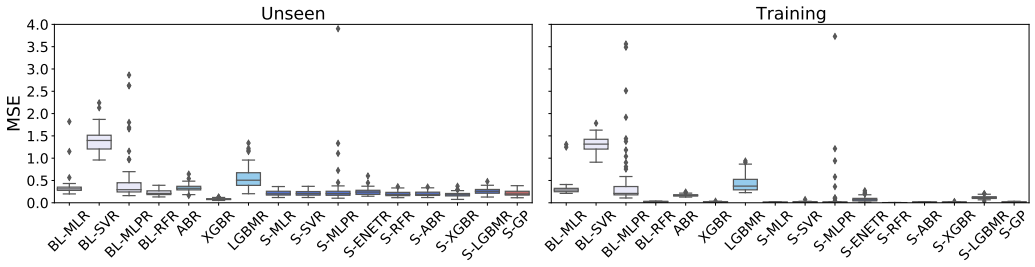

Figure 13: Box-plot on Weierstrass

not smaller.

7. Conclusion

This paper presents a study of Genetic Programming (GP) in the context of Stacked Generalization. More specifically, we have explored GP's role as the meta-learning algorithm that blends in an evolutionary fashion the combined outputs of other Supervised Machine Learning (SML) methods, such as Multi-Linear Regression, Multi-Layer Perceptron, Random Forests and Support-Vector Machines. The contribution of this work is three-fold. First, we assess the impact of recent scientific achievements in the field of GP by comparing them with traditional techniques, something which had not been performed thus far. Second, we provide a broad benchmark over 11 regression problems, an especially important contribution since previous works mostly focused on classification problems. Third, we provide a recommendation for the hyper-parameters of the proposed stacking system and compare it against the underlying base learners, and state-of-the-art ensemble methods (including boosted methods and other stacking approaches). The assessment of experimental results involved statistical validation.

In this work, we have found that in average terms: EDDA initialization, Tournament selection, and GSGP allowed the ensemble system to achieve higher performance when compared against the RHH, $\epsilon - LS$ and GP, respectively. Additionally, we have compared the ensemble system's response towards tuning the base learners' hyper-parameters, before executing the system, and concluded that it achieves superior performance when its hyper-parameters are tuned. Moreover, we provided a formal justification for this to happen, considering the perspective of the semantic space's geometric properties. Furthermore, we have shown that our system can achieve an equivalent

or even better performance than the base learners in isolation, some of which are ensembles themselves (which is the case of Random Forests). Finally, we have shown that the S-GP system is at least as good as the state-of-the-art ensemble approaches, including boosted methods and other stacked generalization approaches.

Further research in the field is still in demand and we consider deepening the cross-fertilization between the fields of Evolutionary Computation and Ensemble Learning. For example, it would be interesting breaking the search-space into sub-spaces that would be better handled by carefully evolved fusion policies focused on a given sub-space.

Acknowledgment

This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia) by the projects PTDC/CCI-INF/29168/2017, DSAIPA/DS/0022/2018 and DSAIPA/DS/0113/2019. Mauro Castelli also acknowledges the financial support from the Slovenian Research Agency (research core funding No. P5-0410).

References

[1] (1980). Statlib datasets archive: Boston house prices dataset. `http://lib.stat.cmu.edu/datasets/boston`. Accessed: 05.09.2019.

[2] (2007). Genetic programming and evolvable machines (8): Genetic programming for computational pharmacokinetics in drug discovery and development. `http://kdbio.inesc-id.pt/œsara/gptp2013/ppb.txt`. Accessed: 05.09.2019.

[3] (2013). Least angle regression, lasso and forward stagewise in r: Diabetes dataset. `https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html`. Accessed: 05.09.2019.

[4] Acosta-Mendoza, N., Morales-Reyes, A., Escalante, H. J., and Gago-Alonso, A. (2014). Learning to assemble classifiers via genetic programming. *International Journal of Pattern Recognition and Artificial Intelligence*, 28.

[5] Alhalaseh, R. and Alhalaseh, K. (2019). Stacked generalization concept for electrical load prediction. In *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–5.

[6] Bhowan, U., Johnston, M., Zhang, M., and Yao, X. (2014). Reusing genetic programming for ensemble selection in classification of unbalanced data. *IEEE Transactions on Evolutionary Computation*, 18(6):893–908.

[7] Breiman, L. (1996). Stacked regressions. *Machine Learning*, 24:49–64.

[8] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

[9] Bukhtoyarov, V. and Semenkina, O. (2010). Comprehensive evolutionary approach for neural network ensemble automatic design. pages 1–6.

[10] Castelli, M., Castaldi, D., Giordani, I., Silva, S., Vanneschi, L., Archetti, F., and Maccagnola, D. (2013). An efficient implementation of geometric semantic genetic programming for anticoagulation level prediction in pharmacogenetics. In *Portuguese Conference on Artificial Intelligence*, pages 78–89. Springer.

[11] Castelli, M., Gonalves, I., Manzoni, L., and Vanneschi, L. (2018). Pruning techniques for mixed ensembles of genetic programming models. *Genetic Programming*, page 5267.

[12] Evans, B. P. (2019). *Population-based Ensemble Learning with Tree Structures for Classification*. PhD thesis, Victoria University of Wellington.

[13] Flasch, O., Friese, M., Zaefferer, M., Bartz-Beielstein, T., and Branke, J. (2015). Learning model-ensemble policies with genetic programming. *Technical report*.

[14] Gonçalves, I., Silva, S., Fonseca, C. M., and Castelli, M. (2017). Unsure when to stop?: Ask your semantic neighbors. In *Proceedings of the Genetic*

and Evolutionary Computation Conference, GECCO '17, pages 929–936, New York, NY, USA. ACM.

[15] Gonalves, I., Silva, S., and Fonseca, C. (2015). Semantic learning machine: A feedforward neural network construction algorithm inspired by geometric semantic genetic programming. In *17th Portuguese Conference on Artificial Intelligence, EPIA 2015 - Coimbra, Portugal*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 280–285. Springer Verlag.

[16] Johansson, U., Lfstrm, T., Knig, R., and Niklasson, L. (2006). Building neural network ensembles using genetic programming. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1260 – 1265.

[17] Karakatič, S. and Podgorelec, V. (2018). Building boosted classification tree ensemble with genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '18, page 165166, New York, NY, USA. Association for Computing Machinery.

[18] Kordk, P. and Cerny, J. (2014). Building predictive models in two stages with meta-learning templates optimized by genetic programming. In *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)*, pages 1–8.

[19] Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

[20] La Cava, W., Spector, L., and Danai, K. (2016). Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, page 741748, New York, NY, USA. Association for Computing Machinery.

[21] Larkin, T. and McManus, D. (2020). An analytical toast to wine: Using stacked generalization to predict wine preference. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 13(5):451–464.

[22] Moraglio, A., Krawiec, K., and Johnson, C. G. (2012). Geometric semantic genetic programming. In *International Conference on Parallel Problem Solving from Nature*, pages 21–31. Springer.

[23] Pawlak, T. P. and Krawiec, K. (2016). Semantic geometric initialization. In Heywood, M. I., McDermott, J., Castelli, M., Costa, E., and Sim, K., editors, *Genetic Programming*, pages 261–277, Cham. Springer International Publishing.

[24] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[25] Poli, R., Langdon, W. B., and McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd.

[26] Polikar, R. (2006). Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE*, 6:21–45.

[27] Rahman, S. S. M., Islam, T., and Jabiullah, M. I. (2020). Phishstack: Evaluation of stacked generalization in phishing urls detection. *Procedia Computer Science*, 167:2410–2418.

[28] Re, A. (2018). *Universal Genetic Programming: a Meta Learning Approach based on Semantics*. PhD thesis, NOVA Information Management School, Universidade Nova de Lisboa.

[29] Reid, S. and Grudic, G. (2009). Regularized linear models in stacked generalization. In Benediktsson, J. A., Kittler, J., and Roli, F., editors, *Multiple Classifier Systems*, pages 112–121, Berlin, Heidelberg. Springer Berlin Heidelberg.

[30] Ren, Y., Zhang, L., and Suganthan, P. N. (2016). Ensemble classification and regression-recent developments, applications and future directions [review article]. *IEEE Computational Intelligence Magazine*, 11(1):41–53.

[31] Rodrigues, N. M., Batista, J. E., and Silva, S. (2020). Ensemble genetic programming. In Hu, T., Lourenço, N., Medvet, E., and Divina, F., editors, *Genetic Programming*, pages 151–166, Cham. Springer International Publishing.

[32] Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33:1–39.

[33] Spector, L. (2012). Assessment of problem modality by differential performance of lexicase selection in genetic programming: A preliminary report. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 401–408, New York, NY, USA. ACM.

[34] Tsanas, A., Little, M. A., McSharry, P. E., and Ramig, L. O. (2009). Accurate telemonitoring of parkinson's disease progression by noninvasive speech tests. *IEEE Transactions on Biomedical Engineering*.

[35] Vanneschi, L., Bakurov, I., and Castelli, M. (2017). An initialization technique for geometric semantic gp based on demes evolution and despeciation. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 113–120. IEEE.

[36] Vanneschi, L., Castelli, M., and Silva, S. (2014a). A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, 15(2):195–214.

[37] Vanneschi, L., Silva, S., Castelli, M., and Manzoni, L. (2014b). Geometric semantic genetic programming for real life applications. In *Genetic programming theory and practice xi*, pages 191–209. Springer.

[38] Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

[39] Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*, volume 14.

Appendix A. Hyper-Parameters of the base learners and other ensemble methods

Table A.10 shows the hyper-parameters that were explored for the base learners and other ensemble-methods. The column *Algorithm* enumerates the considered SML algorithms, the column *Hyper-parameter* stands for the respective hyper-parameters that were assessed during the cross-validation and finally the column *Values range* regards hyper-parameter's values that were considered in the assessment. It is important to highlight that, in the column *Algorithm*, the upper script ^{BL} regards the SML algorithms that were considered as the base learners, ^{S} represents the meta-learners (i.e., level 1

models) for the stacked generalization and ^{X} regards other methods (but the base learners) that were applied on the original feature set.

The hyper-parameter's assessment was performed by mans of a randomized search over the parameters, using RandomizedSearchCV module provided by scikit-learn [8], where each setting is sampled from a distribution over possible parameter values. In contrast to the exhaustive search over the parameter's space, not all the parameter values were assessed for generalization, but rather a fixed number of parameter settings was sampled from the specified distributions. In such a way, the computational effort necessary to perform hyper-parameters tuning can be chosen independent of the number of parameters and their possible values. In our experiments, the number of parameter settings that are sampled equals 10. Following the API, parameters should be sampled through a *dictionary*, where the key represents the hyper-parameter's name, as defined by the underlying regression module, and the value stands either for a distribution over possible values or a list of discrete choices to be sampled uniformly. In this sense, for each base learner, the column *Hyper-Parameter* represents the dictionary key and the column *Values range* represents the respective value. For more information about API's usage and nomenclatures, refer to [8].

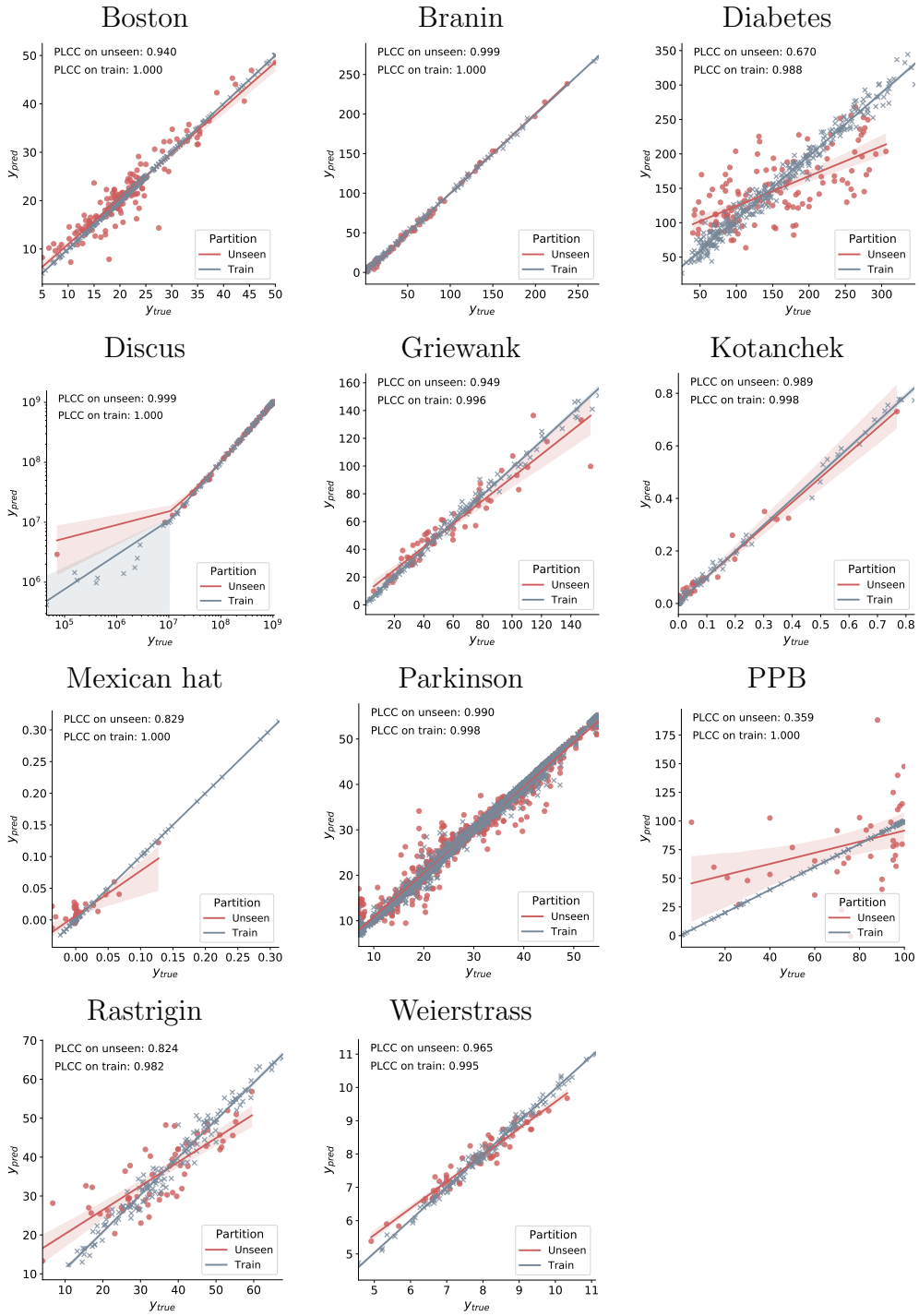

Table 6: Scatter plots and PLCCs between S-GP's y_{true} and y_{pred} on each problem.

		Stacked Generalization's meta-learner								
Problem	**BL**	**S-MLR**	**S-MLPR**	**S-SVR**	**S-ENETR**	**S-RFR**	**S-ABR**	**S-XGBMR**	**S-LGBMR**	**S-GP**
Branin	MLR	0.29	0.73	0.89	0.48	0.68	0.92	0.52	**2.31E-05**	0.06
	MLPR	4.98E-04	0.01	3.90E-03	1.67E-03	7.70E-03	2.60E-03	5.90E-03	**3.23E-08**	0.67
Diabetes	MLR	**1.83E-10**	**3.24E-10**	**2.68E-10**	**1.04E-05**	**8.27E-10**	**1.68E-08**	**3.40E-10**	**7.27E-11**	**1.07E-10**
	SVR	**5.55E-08**	**6.28E-08**	**3.37E-08**	**3.78E-07**	**6.28E-08**	**1.59E-07**	**6.54E-08**	**5.55E-08**	**5.87E-08**
	RFR	**1.63E-11**	**1.71E-11**	**1.71E-11**	**3.05E-06**	**1.63E-11**	**5.15E-11**	**1.90E-11**	**1.63E-11**	**1.71E-11**
Discus	MLR	0.035	**1.63E-11**	**3.82E-08**	**5.55E-08**	**5.55E-08**	**5.55E-08**	**5.55E-08**	**5.55E-08**	0.18
	SVR	3.09E-10	**2.21E-11**	1.63E-11	1.63E-11	1.63E-11	1.63E-11	1.63E-11	1.63E-11	1.63E-11
	RFR	1.25E-09	**1.63E-11**	1.14E-06	0.15	4.22E-11	**4.51E-08**	5.90E-03	**1.63E-11**	3.46E-11
Griewank	MLR	**4.90E-08**	**5.11E-08**	**5.55E-08**	**5.55E-08**	**5.55E-08**	**5.55E-08**	**5.55E-08**	**5.55E-08**	**9.44E-08**
	MLPR	9.00E-04	**3.64E-11**	**1.63E-11**	**1.90E-11**	**2.26E-11**	**1.63E-11**	**3.13E-11**	**1.63E-11**	7.08E-04
Kotanchek	RFR	1.98E-06	**1.25E-04**	0.03	**5.68E-11**	8.02E-08	0.15	**2.23E-05**	**2.32E-11**	1.19E-05
Weierstrass	RFR	6.53E-05	3.23E-03	1.05E-03	0.01	6.92E-11	5.06E-09	6.89E-09	**2.02E-07**	2.07E-04
Mexican hat	RFR	0.98	**3.71E-09**	**9.85E-05**	**3.13E-11**	0.02	0.04	**6.87E-10**	**1.63E-11**	0.17
Parkinson	RFR	0.46	0.92	**8.13E-07**	**2.16E-05**	0.03	**1.63E-11**	0.08	1.02E-08	0.19
PPB	RFR	**1.74E-10**	**2.98E-11**	**4.90E-11**	**9.95E-10**	**3.09E-05**	**9.67E-06**	**1.55E-05**	**6.23E-07**	**3.29E-11**
Rastrigin	RFR	0.72	0.06	0.38	0.16	0.77	0.84	**3.09E-05**	**1.32E-06**	0.58
Count		5	11	9	10	7	9	10	14	5

Table 7: Stacked generalization methods against the underlying base learners. The Table reports the p-values returned by the Wilcoxon's signed-rank test for pairwise data comparison with Bonferroni correction. **Bold** denotes the cases where the underlying base learners outperformed the meta-learners on a given problem.

Problem	Sample B	$\widetilde{SampleA}$	$\widetilde{SampleB}$	p-value
Boston	**S-MLPR**	10.0	9.85	5.15E-05
	S-SVR		9.85	1.44E-10
	S-XGBMR		9.74	2.38E-05
	S-LGBMR		9.45	2.33E-04
Parkinson	**S-LGBMR**	6.45	6.31	9.55E-05
PPB	**S-ENETR**	1852.70	1285.85	1.07E-08
	S-MLPR		1652.69	5.83E-05
	S-SVR		1505.00	2.66E-07
	S-RFR		957.01	1.63E-11
	S-ABR		968.23	2.10E-11
	S-XGBMR		988.80	2.10E-11
	S-LGBMR		997.94	1.58E-10
Diabetes	**S-ENETR**	3794.30	3445.50	1.63E-11
	S-MLR		3757.88	2.33E-04
	S-MLPR		3717.21	3.78E-07
	S-SVR		3705.06	3.64E-07
	S-RFR		3604.16	2.88E-07
	S-ABR		3569.49	1.37E-10
Kotanchek	**S-RFR**	1.65E-03	1.29E-03	8.46E-05
Weierstrass	**S-RFR**	1.97E-01	1.88E-01	5.29E-09
	S-ABR		1.87E-01	2.55E-06
	S-XGBMR		1.80E-01	8.13E-07

Table 8: Statistical assessment of the proposed S-GP approach against other stacking methods. The Table reports only those problems where at least one concurrent stacking method statistically outperformed the S-GP system, after Wilcoxon's signed-rank test for pairwise data comparison with Bonferroni correction.

		Stacked generalization's meta-learner								
Problem	**Ensemble**	**S-MLR**	**S-MLPR**	**S-SVR**	**S-ENETR**	**S-RFR**	**S-ABR**	**S-XGBMR**	**S-LGBMR**	**S-GP**
PPB	ABR	**3.82E-11**	**1.63E-11**	**2.56E-11**	**6.27E-11**	**1.53E-07**	**1.64E-08**	**2.73E-08**	**3.55E-09**	**1.63E-11**
	XGBMR	**3.08E-10**	**1.89E-09**	**8.21E-09**	**7.83E-07**	3.06E-01	8.14E-01	1.43E-03	3.89E-01	**5.97E-11**
	LGBMR	**3.71E-09**	**5.78E-09**	**4.06E-09**	**3.00E-07**	2.63E-01	5.02E-02	5.66E-01	3.90E-03	**4.84E-09**
Diabetes	ABR	**6.92E-11**	**1.24E-10**	**4.10E-10**	2.63E-01	**2.37E-09**	**1.27E-05**	**1.37E-10**	**9.28E-11**	**2.21E-11**
Discus	ABR	3.09E-10	**1.63E-11**	5.55E-08	2.44E-10	1.63E-11	1.63E-11	1.63E-11	5.06E-09	1.63E-11
	XGBMR	3.09E-10	**1.63E-11**	5.55E-08	3.82E-11	1.63E-11	1.63E-11	1.71E-11	7.86E-09	1.63E-11
	LGBMR	3.09E-10	**1.63E-11**	5.55E-08	1.63E-11	1.63E-11	1.63E-11	1.63E-11	1.63E-11	1.63E-11
Griewank	XGBMR	2.19E-04	1.07E-07	2.46E-07	5.57E-06	7.90E-10	4.51E-10	1.04E-09	**3.10E-08**	2.19E-04
Kotanchek	ABR	2.16E-05	1.64E-02	9.49E-04	**8.67E-10**	1.63E-11	2.54E-05	1.22E-01	**1.80E-11**	3.63E-05
	XGBMR	2.32E-11	1.09E-01	1.71E-11	4.84E-04	1.63E-11	1.71E-11	1.63E-11	**2.66E-07**	2.21E-11
Weierstrass	XGBMR	**1.66E-11**	**1.66E-11**	**1.66E-11**	**1.66E-11**	**1.66E-11**	**1.66E-11**	**1.66E-11**	**1.66E-11**	**1.66E-11**
Mexican hat	ABR	1.85E-02	**4.84E-06**	2.82E-01	**2.44E-10**	2.94E-03	6.15E-02	**1.37E-04**	**1.63E-11**	8.77E-03
	XGBMR	1.63E-11	2.45E-01	1.63E-11	5.57E-06	1.63E-11	1.63E-11	1.71E-11	**1.66E-10**	1.63E-11
Rastrigin	XGBMR	**1.63E-11**	**1.63E-11**	**1.63E-11**	**1.63E-11**	**1.63E-11**	**1.63E-11**	**1.63E-11**	**1.63E-11**	**1.63E-11**
Count		6	10	6	7	4	4	5	9	6

Table 9: Stacked generalization methods compared against state-of-the-art boosted ensembles. The Table reports the p-values returned by the Wilcoxon's signed-rank test for pairwise data comparison with Bonferroni correction. **Bold** denotes the cases where stacking did not outperform any of the concurrent boosted ensembles on a given problem.

Algorithm	Hyper-parameter	Values Range
ABR$^{S,\ X}$	learning_rate	$U_{(0.5,\ 1.0)}$
	n_estimators	$U_{\{1,\ 1000\}}$
ENETR^{S}	alpha	$U_{(0.1,\ 1.0)}$
	fit_intercept	[True, False]
	l1_ratio	$U_{(0.0,\ 1.0)}$
	max_iter	$U_{\{1,\ 1000\}}$
LGBMR$^{S,\ X}$	colsample_bytree	$U_{(0.2,\ 0.9)}$
	learning_rate	$U_{(0.05,\ 0.5)}$
	max_depth	$U_{\{2,\ 1000\}}$
	min_split_gain	$U_{(0.1,\ 0.8)}$
	n_estimators	$U_{\{1,\ 1000\}}$
	num_leaves	$U_{\{3,\ 50\}}$
	reg_alpha	$U_{(1.0,\ 1.5)}$
	reg_lambda	$U_{(1.0,\ 1.5)}$
	subsample	$U_{(0.2,\ 0.9)}$
	subsample_freq	$U_{\{5,\ 20\}}$
MLPR$^{BL,\ S}$	alpha	$U_{(0.0,\ 0.5)}$
	hidden_layer_sizes	$U_{\{1,\ 100\}}$
	learning_rate_init	$U_{(0.0,\ 1.0)}$
MLR$^{BL,\ S}$	mlr__fit_intercept	[True, False]
	mlr__normalize	[True, False]
	pf__degree	$U_{\{1,\ 3\}}$
RFR$^{BL,\ S}$	bootstrap	[“best”, “random”]
	max_depth	$U_{\{2,\ 1000\}}$
	max_features	[“n_features”, “sqrt”, “log2”]
	n_estimators	$U_{\{1,\ 1000\}}$
SVR$^{BL,\ S}$	C	$U_{(0.1,\ 0.30)}$
	epsilon	$U_{(0.00001,\ 0.1)}$
	fit_intercept	[True, False]
	loss	[“epsilon_insensitive”, “squared_epsilon_insensitive”]
	max_iter	$U_{\{2,\ 1000\}}$
XGBMR$^{S,\ X}$	colsample_bytree	$U_{(0.2,\ 0.9)}$
	learning_rate	$U_{(0.05,\ 0.5)}$
	max_depth	$U_{\{2,\ 1000\}}$
	n_estimators	$U_{\{1,\ 1000\}}$
	reg_alpha	$U_{(1.0,\ 1.5)}$
	reg_lambda	$U_{(1.0,\ 1.5)}$
	subsample	$U_{(0.2,\ 0.9)}$

Table A.10: SML algorithms' hyper-parameters grid. Notice that $U_{\{a,\ b\}}$ and $U_{(a,\ b)}$ represent discrete and continuous uniform distributions defined [a, b] interval, respectively.