A Novel Evolutionary Algorithm for Automated Machine Learning Focusing on Classifier Ensembles

João C. Xavier-Júnior*, Alex A. Freitas†, Antonino Feitosa-Neto* and Teresa B. Ludermir‡
*Digital Metropolis Institute - Federal University of Rio Grande do Norte, Natal, Brazil
Email: jcxavier@imd.ufrn.br, antonino_feitosa@yahoo.com
†School of Computing - University of Kent, Canterbury, United Kingdom
Email: a.a.freitas@kent.ac.uk
‡Center for Information Technology - Federal University of Pernambuco, Recife, Brazil
Email: tbl@cin.ufpe.br

***Abstract*—Automated Machine Learning (Auto-ML) is an emerging area of ML which consists of automatically selecting the best ML algorithm and its best hyper-parameter settings for a given input dataset, by doing a search in a large space of candidate algorithms and settings. In this work we propose a new Evolutionary Algorithm (EA) for the Auto-ML task of automatically selecting the best ensemble of classifiers and their hyper-parameter settings for an input dataset. The proposed EA was compared against a version of the well-known Auto-WEKA method adapted to search in the same space of algorithms and hyper-parameter settings as the EA. In general, the EA obtained significantly smaller classification error rates than that Auto-WEKA version in experiments with 15 classification datasets.**

I. Introduction

Classification is one of the most important tasks in Machine Learning, and decades of research in this area have produced a large number of classification algorithms [20]. Hence, the selection of the best classification algorithm to a given input dataset is a challenging problem, since the predictive performance of an algorithm is strongly dependent on characteristics of the input dataset [10], as well as on the hyper-parameter settings of the algorithm.

Hence, the research area of Automated Machine Learning (Auto-ML) has emerged, in the last 5 years or so, as a promising approach to tackle this problem [4], [6], [24]. This approach is promising because it automatically performs a systematic search in a very large space of candidate algorithms and hyper-parameter settings, relieving the user from the task of performing ad-hoc, tedious and very time-consuming experiments with different algorithms and their settings.

This work follows the Auto-ML approach, but it is mainly focused on a broad type of classification algorithms, called classifier ensembles, which combine the outputs of many base classifiers (e.g. by majority voting). This combination usually improves predictive accuracy by comparison with a single base classifier [1]. We focus on ensembles because they are usually considered one of the best types of classification algorithms regarding predictive accuracy. For instance, a recent study [9] compared the predictive accuracy of 179 classification algorithms divided into 17 families of algorithms, across 121 datasets, and concluded that overall the best algorithms were versions of random forests, a well-known type of ensemble (combining the outputs of many decision trees). Even restricting the choice of classifiers to ensembles, though, there are still many different types of ensembles, and their predictive accuracies also depend on both characteristics of the input dataset and their hyper-parameter settings.

In this context, the main contribution of this work is to propose a new Evolutionary Algorithm (EA) for the Auto-ML problem of automatically selecting the best ensemble method and its best hyper-parameter setting for an input dataset. In general, an EA iteratively generates and evaluates a number of candidate solutions, using an evaluation function to select the most promising ones at each generation, and then uses the information encoded in the selected solutions to generate better ones. Hence, the candidate solutions evolve towards better solutions with time. The proposed algorithm is an Estimation of Distribution Algorithm (EDA) [7], which follows the basic EA principle of evolving candidate solutions, but (unlike a typical EA) it does so by explicitly evolving a probabilistic model of the best solutions and their components, as discussed in Section II-C. Hence, EDAs combine methods and concepts from both EAs and probability theory, which arguably gives them a sounder mathematical basis than conventional EAs.

II. Background

A. Classification and Classifier Ensembles

In the classification task of machine learning, each instance (object) in the input dataset is represented by a set of features (characteristics) and a class attribute. A classification algorithm has access to the class values of instances in the training set, but not in the test set. Hence, the goal is to learn a model from the training set that is able to predict the class value of each instance in the test set (with instances unseen during training), based on the feature values for that instance.

Ensemble methods produce a classification model containing a number of classifiers (a classifier ensemble). Such ensembles have a two-layer structure. A set of base classifiers (first layer) receive input data and their predicted classes are sent to a combination module (second layer), which combines all received predictions into a single predicted class for each instance. Combining the results of different classifiers often outperforms base classifiers [1], [2], since an ensemble's

predictions are usually more robust than the predictions performed by a single classifier.

In classifier ensembles, the two main aspects are the selection of base classifiers and the combination method. Regarding the former, some ensemble methods use multiple base classifiers of the same type (homogeneous ensembles), changing only their parameter settings; whilst other methods use different types of base classifiers (heterogeneous ensembles). Regarding the combination method in the second layer, several methods have been proposed [3], such as: simple majority voting, weighted voting (where the vote for a class is weighted by its estimated probability), using a full classification algorithm – treating the classes predicted by the base classifiers as features for predicting the class at the meta-level, etc.

B. Automated Machine Learning (Auto-ML)

There are a great variety of types of classification algorithms in the literature, such as Decision Trees, Neural Networks, Support Vector Machines, among many others [9]. However, each type of algorithm has its own limitations, and there is no classifier which is the best for all problem domains, since the predictive accuracy of a classifier strongly depends on the characteristics of the input dataset [10]. In addition, in general the predictive performance of any classification algorithm is also strongly dependent on its hyper-parameter settings. This leads to the challenging optimization problem of how to select the best classification algorithm and its corresponding best hyper-parameter settings for each input dataset provided by a user. An emergent approach to solve this problem involves Automated Machine Learning (Auto-ML) methods, which automatically search for the combination of classification algorithm and hyper-parameter settings that maximizes predictive performance in an input dataset.

Recent research on Auto-ML has provided some off-the-shelf Auto-ML tools for researchers and practitioners of ML; such as Auto-sklearn [6] and Auto-WEKA [4]. Here we briefly review only Auto-WEKA, which is used as a strong baseline method in our experiments reported later.

Auto-WEKA is a tool for automatically selecting a machine learning algorithm and its hyper-parameters, as proposed in [4]. Auto-WEKA uses a Bayesian optimization search method called SMAC (Sequential Model-based Algorithm Configuration) to automatically search through the joint space of WEKA's learning algorithms and their respective hyper-parameter settings, in order to maximize predictive performance. Combined with the WEKA tool [5], Auto-WEKA has obtained good results for a wide variety of data sets [11].

C. Estimation of Distribution Algorithms

Evolutionary Algorithms (EAs) are a type of search and optimization technique inspired by the principle of natural selection. As optimization techniques, EAs have the advantages of performing a global search in the space of candidate solutions (less likely to get trapped into a local optimum than a greed search) and being robust to noise [12], [13].

Estimation of Distribution Algorithms (EDAs) [7] are a type of EA which explore the space of potential solutions by building and sampling explicit probabilistic models of promising candidate solutions. EDAs have been applied to several types of machine learning tasks, including feature selection [14], [15], [18] and classification [17].

EDAs iteratively generate and evaluate a population of candidate solutions (individuals) to a problem. The initial population is generated based on the uniform distribution over all possible solutions. Once the fitness of each individual is computed, individuals are ranked based on their fitness values, and a subset of the most promising solutions (usually, 50% of the best solutions) are selected. Then, a probabilistic model is constructed aiming to estimate the probability distribution of the selected solutions. Once the model is constructed, new solutions are generated by sampling the distribution encoded by this model. The fitness of each new individual is evaluated, and so on. This process is repeated until some termination criteria is met, like in other types of EAs.

The crucial difference between EDAs and other EAs is how they generate new individuals at each iteration (generation), as follows. In EDAs the selected individuals are used to update a probabilistic model, from which new individuals will be probabilistically sampled in the next generation. By contrast, in other EAs the next generation's individuals are generated by applying solution-alteration operators like crossover and mutation to the selected individuals of the current generation. EDAs explicitly maintain and evolve a probabilistic model of the best solutions evaluated so far, unlike other EAs. Hence, an advantage of EDAs (versus other EAs) is that they directly use sound concepts of probability theory to guide the evolutionary process. Another advantage of EDAs is that they require fewer parameters than most EAs. In particular, most EAs require the user to choose which type of crossover and mutation operators should be used to create new solutions, as well as choosing the corresponding crossover and mutation probability rates. EDAs relieve the user from such concerns, since they do not use any operator to generate new solutions, and simply sample new candidate solutions from a probability distribution.

The Population-Based Incremental Learning (PBIL) algorithm, proposed in [8] and recently reviewed in [16], is an EDA that evolves a probability vector, where each vector component represents the probability of that component being selected for inclusion in the creation of a candidate solution. The vector components' values are usually initialized with a probability of 0.5. Then, at each generation (iteration), a population of individuals (candidate solutions) are sampled from the probability vector based on its probability values, and each individual is evaluated using a fitness function. A predefined number of the best individuals (based on fitness) in the current generation are selected, and the relative frequency of solution components in those selected individuals is used to update the probability vector, by increasing the probability values for the solution components that occurred most often in the selected individuals. The amount of increase is controlled by a learning rate parameter. Hence, the probability vector

gradually evolves towards components with probability values closer to 1 or 0, depending on whether or not, respectively, the component has been used in the best candidate solutions evaluated along the generations.

The PBIL algorithm has only 3 parameters: (a) the population size, i.e., the number of candidate solutions (individuals) sampled from the probability vector at each generation; (b) the Learning Rate, which specifies how large the steps towards good solutions are; (c) the number of best individuals selected for updating the probability vector at each generation.

III. Related Work

A. Automated Selection of Ensembles Methods

Current Auto-ML methods typically use a search space with many types of classification algorithms, without focusing on ensembles as in this work. However, some studies have proposed different approaches for automating the creation of classifier ensembles (base classifiers and their hyper-parameter settings) [21], [22], [23]. In particular, Wistuba et al. [21] proposed an automatic approach to generate ensembles with several layers, called Automatic Frankensteining, where a Bayesian Optimization method is used to select base classifiers and their settings using a bagging strategy.

In fact, most Auto-ML methods are based on Bayesian optimization. For instance, Lacoste et al. [23] proposes an extension of SMBO (Sequential Model-Based Optimization) to optimize the selection of ensemble members based on their performance on randomly selected subsets of the validation data produced by a bootstrap method. In addition, Levesque et al [22] propose an approach to build fixed-size ensembles, optimizing the configuration of one base classifier of the ensemble at each iteration of the hyper-parameter optimization algorithm, considering the interaction with other models when evaluating performance. In this way, the Bayesian optimization method estimates which prediction model is the best candidate to be added to the ensemble.

It is important to emphasize that all three aforementioned methods for automating the selection and configuration of classifier ensembles use the Bayesian optimization method, whereas our proposed approach uses the PBIL algorithm – a type of Evolutionary Algorithm. Unlike the sequential nature of the search performed by the Bayesian optimization method, PBIL evolves a population of candidate classifier ensembles, performing a more global, broader search (conceptually a parallel search) in the space of candidate solutions.

B. The use of Evolutionary Algorithms for Auto-ML

As mentioned earlier, Evolutionary Algorithms (EAs) have been used in different areas mainly because they are less likely to get trapped into a local optimum (compared to a greedy search), and also robust to noise. These important advantages are taken into consideration in Auto-ML researches [25], [19]. For example, in [25] the authors proposed the use of an EA (a genetic algorithm) for searching a very large search space of many different Multi-Label Classification (MLC) algorithms and their hyper-parameters; whilst in [26] the authors proposed a genetic programming method for automating the selection and configuration of both classification algorithms and data pre-processing methods (classification pipelines). On the other hand, in a very recent work [19], the authors proposed the use of a genetic programming (GP) method to search the space of possible architectures of hierarchical ensembles and to optimize their hyper-parameters. Broadly speaking, the GP method proposed in [19] addresses the same type of problem addressed in our work (the automatic creation of ensembles), but using GP, a type of EA that is very different from the EDA proposed in this work – for a brief review of the differences between EDAs and other types of EAs, see Section II.C. In addition, the work in [19] focuses more on transfer learning and meta-learning, which is not the focus of this current work. Hence, to the best of our knowledge, this is the first work to propose an EDA addressing the Auto-ML task of optimizing the selection and configuration of classifier ensembles.

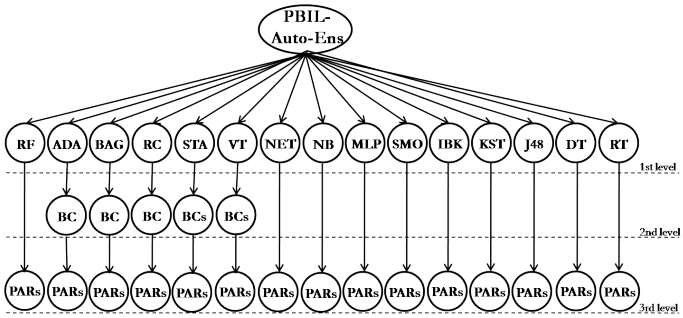

Fig. 1. The General structure of PBIL-Auto-Ens' search space

IV. The Proposed PBIL-Auto-Ens

As mentioned earlier, the main contribution of this work is to propose a new Evolutionary Algorithm (EA), based on the general PBIL algorithm described in Section II.C, for the automated selection and configuration of classifier ensembles. The proposed method is called PBIL-Auto-Ens (PBIL for Auto-ML focusing on Ensembles). The general structure of the search space of PBIL-Auto-Ens is presented in Figure 1.

Each individual (candidate solution) of the PBIL's population can represent either an ensemble method or a base classifier. As shown in Figure 1, 6 ensemble methods and 9 base classifiers can be selected. All possible ensemble methods and base classifiers come from the WEKA Machine Learning tool [5]. The possible options are divided into two types: (a) ensemble methods: Random Forest (RF), AdaBoost (ADA), Bagging (BAG), Random Committee (RC), Stacking (STA), Vote (VT); and (b) base classifiers: BayesNet (NET), NaiveBayes (NB), Multilayer Perceptron (MLP), a Suport Vector Machine algorithm (SMO), k-Nearest Neighbor (IBK), KStar (KST), Decision Tree (J48), Decision Table (DT), and Random Tree (RT).

The second level of the search space is used to generate the possible members (base classifiers) for the ensembles of classifiers. Any one of the 9 base classifiers can be chosen as a

valid member. Stacking and Vote ensembles can have up to 10 different members, whereas AdaBoost, Bagging and Random committee have only one type of member. The Random Forest method is the only exception since it does not have any member choice – its members are fixed to be decision trees. Finally, the third level is used for setting up the parameters of all base classifiers and their corresponding ensemble methods.

Each candidate solution is represented by a dynamic binary vector which can have up to 363 positions. In fact, the binary vector consists of three parts (P1, P2 and P3). P1 specifies which classifier (ensemble method or base classifier) is represented (1st level of Figure 1). P2 specifies each of the members of the ensemble specified in part P1 (2nd level). It is divided in n parts, where n represents the number of base classifiers used by the ensemble in P1. Finally, P3 specifies the parameters of each base classifier used by the ensemble method in P2 (3rd level). It is divided in m parts, where m represents the parameters of each base classifier.

The proposed PBIL-Auto-Ens extends the original PBIL [8] and its more recent variants [16] in two ways. First, it uses a hierarchically structured probability vector, with three levels of solution component probabilities that mirror the three levels of the search space shown in Figure 1. Second, the creation of individuals by sampling solution components from the probability vector is adapted to follow the hierarchical structure of the probability vector, as described next.

At each generation, each individual is created by sampling solution components from the hierarchical probability vector in a top-down fashion, as follows. First, one of the 15 types of classifiers at the first level of Figure 1 is selected. This will define the type of ensemble or the choice of using a single base classifier instead. If the sampled component was a single base classifier or a Random Forest method, the individual creation procedure skips the second level and jumps straight to the third level. Otherwise, the ensemble type chosen at the first level is used to identify which solution components (which types of base classifiers) are candidate for selection at the second level. Finally, at the third level, the candidate parameter settings are determined by the choice of the base classifier at the second level (or at the first level, if that was the case).

At each of the three levels, the selection of one solution component among all candidate components is done via a probabilistic sampling from the corresponding level of the probability vector. At the start of the evolution (generation 0), all solution components in the three levels of the probability vector are initialized with a uniform probability distribution. At the end of each generation, the 50% best individuals of that generation are selected and their solution components are used to update the corresponding component probabilities in the three levels of the probability vector. This updating consists of increasing the probability for each component in proportion to the relative frequency of use of that component among the selected individuals (the 50% best ones), and also in proportion to the learning rate parameter. More precisely, for each i-th solution component, its probability at the end of generation g, denoted by $p[i]_g$, is updated using the formula: $p[i]_g = (1 - LR) * p[i]_{g-1} + LR * RelFreq[i]_g$, where LR is the learning rate and $RelFreq[i]_g$ is the relative frequency of the i-th solution component among the individuals selected at generation g. Hence, by iteratively selecting the best candidate solutions and increasing the probabilities of each of their components in the probability vector, gradually the probability vector evolves to contain higher probability values for the best solution components, leading to the creation of better and better classifier ensembles. This evolutionary process terminates when a predefined runtime limit is reached.

TABLE I
DESCRIPTION OF THE DATASETS.

Id	Dataset	# Instances	# Disc. Attr.	# Cont. Attr.	# Classes
d1	Abalone	4,177	1	7	28
d2	Adult	32,561	8	6	2
d3	Arrhythmia	452	0	260	13
d4	Car	1,728	0	6	4
d5	Ecoli	336	0	7	8
d6	German-Credit	1,000	13	7	2
d7	KR-vs-KP	3,196	36	0	2
d8	Madelon	2,600	0	500	2
d9	Nursery	12,960	8	0	5
d10	Secom	1,567	0	590	2
d11	Semeion	1,593	0	256	10
d12	Sonar	208	0	60	2
d13	Waveform	5,000	0	40	3
d14	Wine-quality	4,898	0	11	11
d15	Yeast	1,484	0	8	10

V. EXPERIMENTAL METHODOLOGY

A. Datasets Used in the Experiments

In order to evaluate the proposed PBIL-Auto-Ens method, we used 15 classification datasets, available for download from the well known UCI machine learning repository. The majority of these datasets have also been used in recent Auto-ML studies [4], [6], [21]. Table I shows the number of instances, attributes (separately for continuous and discrete attributes) and classes in each of the used datasets.

B. Auto-ML Methods Compared in the Experiments

PBIL-Auto-Ens was compared against a strong baseline Auto-ML method, namely Auto-WEKA [4], using a nested version of the well-known cross-validation (CV) procedure. Both methods have been run with an external 5-fold CV, and an internal 10-fold CV. Hence, at the external CV level, the input dataset is randomly divided into 5 folds (each with about 20% of the instances), and then PBIL-Auto-Ens and Auto-WEKA are run 5 times, each using a different fold as the test set and the other 4 folds as the training set. In each of those 5 runs, the training set (80% of the full dataset) is divided into 10 folds, each with about 10% of the training set (i.e., about 8% of the full dataset). Then, whenever a candidate solution (ensemble method or base classifier) is generated by PBIL-Auto-Ens or Auto-WEKA, that solution is evaluated by running its configuration using the internal 10-fold CV, so that each of the 10 runs of that candidate solution uses 9 internal

folds (72% of the full dataset) as a learning set (to learn the classification model) and one internal fold (8% of the full dataset) as a validation set to evaluate the predictive accuracy of the learned model. The quality measure of that candidate solution is given by the mean error rate of the learned model over the 10 internal validation sets. Hence, the evaluation of each candidate solution uses only the training set, not the external test set, which is reserved for measuring the predictive accuracy of the best solution returned by PBIL-Auto-Ens and Auto-WEKA.

Note that both PBIL-Auto-Ens and Auto-WEKA are non-deterministic search methods, i.e. their results depend on a seed number used to randomly initialize the candidate solutions. Each reported result is the mean over the results obtained by a method with two random seeds (the same seeds are used by both methods), running an external 5-fold CV for each seed as explained above – i.e., each reported result is the mean over 10 results (with 10 different test sets).

We used all default parameter settings of Auto-WEKA, including the classification error rate (the proportion of incorrectly classified instances) as the evaluation function to be optimized during training. To make the comparison between PBIL-Auto-Ens and Auto-WEKA fair, we also used error rate as the fitness function of PBIL. Both methods used the same runtime limit, 60 minutes for each run, where one run means one execution of one iteration of the external 5-fold CV (applying the method to the training set of that iteration) for a single value of the random seed, for each dataset. Hence, in total each method was run for 150 hours, considering 5 external CV iterations times 2 seeds times 15 datasets.

PBIL was run with the following parameter settings: population size of 50, learning rate of 0.5, and 50% of the best individuals of the current generation selected for updating the probability model. The latter two parameter settings are relatively standard in the PBIL literature, whilst the population size was set based on preliminary experiments. Note that, unlike most PBIL (and EA) implementations, PBIL-Auto-Ens does not have a parameter for the number of generations, because its stopping criterion is a runtime limit.

PBIL-Auto-Ens has been implemented in Java using the WEKA API. We run the experiments on a desktop PC with Windows 10 professional 64 bit operating system driven by an Intel Core i7-3770S 3.10GHz Processor with 8GB RAM.

VI. Experimental Results

A. An Analysis of the Mean Error Rate

Table II presents the mean error rates for Auto-WEKA (3rd column) and PBIL-Auto-Ens (4th column), and the normalized error rate reductions (5th column), in % values. The latter measure is calculated by substracting the PBIL-Auto-Ens' error rate from the Auto-WEKA's error rate and dividing this result by the greater between those two error rates. This produces a normalized error rate reduction, with possible values varying in the range [-1,1], where the greater the positive (negative) value, the better the result of PBIL-Auto-Ens (Auto-WEKA).

TABLE II
Error Rate on the test set (mean over 10 runs).

Id	Dataset	Auto-WEKA	PBIL-Auto-Ens	Error Rate Reduction (%)
d1	Abalone	0.7371	0.7454	-1.11
d2	Adult	0.1438	0.1388	3.48
d3	Arrhythmia	0.2993	0.2864	2.32
d4	Car	0.0029	0.0012	58.62
d5	Ecoli	0.1370	0.1325	3.28
d6	German-Credit	0.2785	0.2495	10.41
d7	KR-vs-KP	0.0587	0.0058	90.12
d8	Madelon	0.2702	0.2533	6.25
d9	Nursery	0.0038	0.0099	-61.62
d10	Secom	0.0664	0.0660	0.60
d11	Semeion	0.1098	0.0618	43.72
d12	Sonar	0.2265	0.1832	19.12
d13	Waveform	0.1357	0.1495	-9.23
d14	Wine-quality	0.3349	0.3185	4.90
d15	Yeast	0.4064	0.4060	0.10
Number of wins		3	12	—
Mean Error Rate Reduction			—	11.40%

This normalization is important because, as the datasets represent different problem domains, an (unnormalized) error rate reduction could be relatively large in some cases but relatively small in others. As shown in the penultimate row of Table II, PBIL-Auto-Ens achieved a smaller error than Auto-WEKA in 12 out of the 15 datasets. In addition, as shown in the last row of the table, PBIL-Auto-Ens achieved a mean normalized error rate reduction of 11.40% across all 15 datasets.

In order to conduct a more robust analysis of the results, the Wilcoxon Signed-Rank statistical test was used (at the conventional significance level of 5%) to determine whether or not there is a statistically significant difference between the mean error rates of PBIL-Auto-Ens and Auto-WEKA across the 15 datasets. This test was chosen because it is non-parametric (avoiding the assumption of normality), being based on the relative rank of the two methods across the datasets. This test produced a p-value = 0.03572 (two-tailed), so that the difference between the error rates of the two methods can be considered statistically significant.

B. An Analysis of the Returned Best Solutions

Table III presents the 10 best solutions (classifiers) returned by each method (one solution for each run) for each of the 15 datasets. The numbers in brackets right after a classifier's acronym are the number of times that classifier was selected, out of the 10 runs. Due to lack of space, this table shows only the name of the selected classifiers, not their full configuration. Recall that, although the search space of PBIL-Auto-Ens and Auto-WEKA includes mainly classifier ensembles and their configurations (both methods use the same search space), it also includes the option of selecting and configuring only a single base classifier. The latter is a useful option, since it is possible that in some datasets a single classifier can perform very well, without the need for the extra computational cost and complexity of an ensemble.

As shown in Table III, broadly speaking, the classification algorithms selected by PBIL-Auto-Ens are more diverse than the ones selected by Auto-WEKA. Hence, it seems that PBIL-

TABLE III
BEST ALGORITHMS RETURNED BY EACH METHOD.

Dataset	Auto-WEKA	PBIL-Auto-Ens
d1	MLP(5), KST(2), RF(2), VT(1)	RF(4), MLP(2), DCT(2), IBK(1), RC(1)
d2	NET(4), J48(3), RF(3)	DCT(4), NET(3), BAG(2), J48(2)
d3	NET(3), NB(2), J48(1), RF(1), VT(3)	NB(8), DCT(2)
d4	MLP(6), SVM(4)	MLP(2), SVM(7), RC(1)
d5	SVM(4), RF(3), IBK(2), VT(1)	ADA(1), BAG(3), RC(3), RF(3)
d6	MLP(2), RF(3), VT(5)	MLP(3), RF(6), BAG(1)
d7	RF(8), MLP(1), VT(1)	RF(5), ADA(1), MLP(2), SVM(1), DCT(1)
d8	RF(4), J48(4), DCT(2)	RF(10)
d9	MLP(4), SVM(4), RF(1), VT(1)	ADA(5), RF(5)
d10	RF(8), STA(1), IBK(1)	RT(4), SVM(4), DCT(1), RC(1)
d11	RF(9), KST(1)	RF(6), SVM (4)
d12	SVM(4), KST(3), IBK (1), MLP (1)	MLP(2), KST(2), SVM(1), IBK(1), ADA(2), STA(2)
d13	MLP(6), SVM(4)	MLP(5), BAG(2), RC(3)
d14	RF(9), KST(1)	RF(6), KST(3), IBK(1)
d15	RF(6), VT(3), SVM(1)	RF(4), VT(2), NET(1), SVM(1), IBK(1), RC(1)

Auto-Ens is exhibiting more flexibility in selecting the best algorithm for each dataset, which is the core motivation for Auto-ML. In addition, Auto-WEKA selected an ensemble (rather a single base classifier) in 49.3% (74 / 150) of the cases, whilst PBIL-Auto-Ens selected an ensemble a little more often, in 53.33% (80 / 150) of the cases.

VII. CONCLUSION AND FUTURE WORK

This work proposed a new evolutionary algorithm (PBIL-Auto-Ens) for the Auto-ML problem of automatically selecting the best ensemble method and its best hyper-parameter setting for an input dataset. PBIL-Auto-Ens was compared against Auto-WEKA [4], a strong baseline Auto-ML method. Both methods used the same search space of candidate solutions and the same evaluation function (classification error rate) to guide their search; hence the difference in their results reflect mainly their different search methods, as discussed earlier.

The results have shown that PBIL-Auto-Ens obtained a smaller error rate than Auto-WEKA in 12 of the 15 datasets used in the experiments; and on average, across all datasets, PBIL-Auto-Ens' error rate was about 11% smaller than Auto-WEKA's error rate – a normalized error reduction value, which takes into account the differences of scale of the error rates across the datasets. In addition, the difference of error rates between the two methods was statistically significant, according to a Wilcoxon Signed-Rank test (p-value = 0.03572 – two-tailed test).

In this work we used as evaluation function the error rate since it is the default in Auto-WEKA, but in future work we intend to use different evaluation functions, such as the Area Under the Receiver Operating Characteristic curve (AUROC) and the F-measure. Another future work would be to extend the experiments to consider other Auto-ML methods, like the one very recently proposed in [19].

ACKNOWLEDGMENTS

This work has been financially supported by the Brazilian Research Council (CNPq - process no. 150748/2017-5).

REFERENCES

[1] Zhi-Hua Zhou, (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition.

[2] L.I. Kuncheva, (2004). *Classifier Ensembles for Changing Environments*. In: Roli F., Kittler J., Windeatt T. (eds) Multiple Classifier Systems. Lecture Notes in Computer Science 3077, pages 1–15. Springer.

[3] L.I. Kuncheva, (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience.

[4] C. Thornton, F. Hutter, H.H. Hoos and K. Leyton-Brown, (2013). *Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms*. Proc. 19th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pages 847–855. ACM Press.

[5] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, (2009). *The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter*, volume 11(1), pages 10–18.

[6] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, (2015). *Efficient and Robust Automated Machine Learning*. Advances in Neural Info. Processing Systems 28, pages 2962–2970.

[7] P. Larrañaga, J.A. Lozano, (2002). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, Boston, MA.

[8] S. Baluja and R. Caruana, (1995). *Removing the Genetics from the Standard Genetic Algorithm*, Proc. 12th Int. Conf. on Machine Learning, pages 38–46, California, July, 1995.

[9] M. Fernández-Delgado, E. Cernadas, S. Barro and D. Amorim, (2014). *Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?*, J. of Mach. Learn. Res., 15(1), pages 3133–3181.

[10] P. Brazdil, C. Giraud-Carrier, C. Soares and R. Vilalta, (2009). *Metalearning: Applications to Data Mining*, Springer.

[11] L. Kotthoff, C. Thornton, H.H. Hoos, F. Hutter and K. Leyton-Brown, (2017). *Auto-WEKA 2.0: Automatic Model Selection and Hyperparameter Optimization in WEKA*, J. of Mach. Learn. Res., 18(1), pages 826–830.

[12] A.A. Freitas, (2002). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Springer.

[13] A.E. Eiben and J.E. Smith, (2015). *Introduction to Evolutionary Computing*, 2nd edition, Springer.

[14] I. Inza, P. Larrañaga and B. Sierra, (2002). *Feature subset selection by estimation of distribution algorithms*, Estimation of Distribution Algorithms, pages 269–293, Springer.

[15] K. Shelke, S. Jayaraman, S. Ghosh, J. Valadi, (2013). *Hybrid feature selection and peptide binding affinity prediction using an EDA based algorithm*, Proc. IEEE Congress on Evolutionary Computation (CEC), pages 2384–2389.

[16] M. Zangari, R. Santana, A. Mendibury, A.T.R. Pozo, (2017). *Not all PBILs are the same: unveiling the different learning mechanisms of PBIL variants*, Applied Soft Computing, 53, pages 88–96, April 2017.

[17] X. Yang, H. Dong and H. Zhang, (2009). *Naive Bayes Based on Estimation of Distribution Algorithms for Classification*, International Conference on Information Science and Engineering, pages 908-911.

[18] Y. Saeys, S. Degroeve, D. Aeyels, P. Rouz and Y. Van de Peer, (2004). *Feature selection for splice site prediction: A new method using EDA-based feature ranking*, BMC Bioinformatics, 5(64), 11 pages, 2004.

[19] P. Kordík, Jan Černý and T. Frýda, (2018). *Discovering Predictive Ensembles for Transfer Learning and Meta-learning*, Machine Learning, 107(1), pages 177–207, January, 2018.

[20] S. B. Kotsiantis, (2007). *Supervised machine learning: A review of classification techniques*, Emerging Artificial Intelligence Applications in Computer Engineering, pages 3–24, IOS Press, 2007.

[21] M. Wistuba, N. Schilling and L. Schmidt-Thieme, (2017). *Automatic Frankensteining: Creating Complex Ensembles Autonomously*, Proc. SIAM Int. Conf. on Data Mining, pages 741–749, SIAM, 2017.

[22] J. Lévesque, C. Gagné and R. Sabourin, (2016). *Bayesian Hyperparameter Optimization for Ensemble Learning*, Proc. 32nd Conference on Uncertainty in Artificial Intelligence (UAI), pages 437–446. Jersey City, New Jersey, USA, 2016.

[23] A. Lacoste, H. Larochelle, F. Laviolette and M. Marchand, (2014). *Sequential Model-Based Ensemble Optimization*, Computing Research Repository (CoRR), 2014.

[24] R. Olson, R. Urbanowicz, P. Andrews, N. Lavender, L. Kidd, and J.H. Moore, (2016). *Automating Biomedical Data Science Through Tree-Based Pipeline Optimization*, European Conference on the Applications of Evolutionary Computation, pages 123–137. Springer, 2016.

[25] A.G.C. de Sá, G.L. Pappa and A.A. Freitas, (2018). *Automated selection and configuration of multi-label classification algorithms with grammar-based genetic programming*, To appear in Proc. of the 15th International Conf. on Parallel Problem Solving from Nature (PPSN-2018), to be held in Coimbra, Portugal, Sep. 2018.

[26] A.G.C. de Sá, W.J.G.S. Pinto, L.O.V.B. Oliveira, G.L. Pappa, (2017). *RECIPE: A Grammar-based Framework for Automatically Evolving Classification Pipelines*, Proc. of the 20th European Conference on Genetic Programming (EuroGP'17), LNCS 10196, pages 246–261. Springer.