Ensembles are all you need: An AutoML method based on ensembles of predefined machine learning pipelines

Juan S. Angarita-Zapata, Antonio D. Masegosa, and Isaac Triguero

Abstract—Nowadays, there are various AutoML frameworks that automatise the construction of machine learning pipelines, that is, the workflow from data preprocessing to model validation. Most of them are based on online optimisation processes to search for the best performing pipeline on a given dataset. This online optimisation could be complemented with meta-learning to warm-start the search phase and/or the construction of ensembles using pipelines derived from the optimisation process. However, given the complexity of the search space and the high computational cost of tuning-evaluating pipelines generated from it, online optimisation is only beneficial when there is a long time period to obtain the final model. In this paper, we propose AutoEn, a simple and efficient method for the automatic generation of ad-hoc multi-classifier ensembles from a predefined set of high-performance machine learning pipelines. We compare its performance against TPOT and Auto-sklearn using the so-called AutoML benchmark, which is composed of binary and multi-class supervised classification problems. Experimental results demonstrate that AutoEn, despite its simplicity, can lead to better or competitive results with respect to the state-of-the-art, becoming a very promising alternative to develop simpler AutoML methods that are potentially more parallelisable and suitable to tackle bigger datasets.

Index Terms—Automated Machine Learning, Ensemble learning, Machine Learning, Supervised Classification.

1 Introduction

Automated machine learning (AutoML) is an emerging area in Machine Learning (ML) that seeks to automate the ML workflow from data preprocessing to model validation [1]. Such automation provides robust AutoML methods that enable people, with either little or no specialised knowledge, to integrate ML solutions into daily activities of business organisations. The latter is known as the democratisation of ML [1] and it is aligned with the actual purpose of Artificial Intelligence: to learn and act automatically without human intervention [2].

Early AutoML applications were merely centred on algorithm selection [3]. Later on, the focus moved forward to include hyperparameter tuning [4], [5]. That is, they only considered the selection and tuning of ML algorithms. However, current ML applications, such as self-driving cars, e-mail spam, and malware filtering or automatic speech recognition, call for the complete automation of the entire Knowledge Discovery in Databases cycle [6], including the data preprocessing stage [7], as it has proven key to gleaning quality data before applying ML techniques [8].

Although recent literature [9], [10], [11] reports a wide variety of AutoML methods, there are only a few that actually automatise the construction of a complete pipeline. Those methods usually generate ML pipelines using an online search strategy, that is, a search strategy that takes place after the input dataset has been provided. In some cases, this online search can be purely based on optimisation approaches that test different promising combinations of pipelines structures from a predefined base of preprocessing and ML algorithms to minimise or maximise a predefined performance measure. Representative methods in this category are Auto-WEKA [12] and TPOT [13].

Alternatively, there are AutoML methods whose online search is complemented with learning strategies [4], [14], [15]. They typically use meta-learning [16]. These techniques first extract meta-features of the input dataset at hand, providing information such as the number of instances, features, classes or entropy, among others. From these meta-features, meta-learning identifies good candidates of pipeline structures from a predefined knowledge base that stores meta-features for different datasets and pipelines that are likely to perform well on them. Then, the candidate pipelines are used for a warm-start of the optimisation approach. In addition, other AutoML approaches use the best performing pipelines found during the search to build an ensemble of models [4], [17]. This ensemble approach has proven to be more robust than only using the best pipeline found via optimisation.

As it can be observed, the online search of AutoML generally uses optimisation as the core engine for generating and tuning pipelines. However, the optimisation of ML pipelines is a difficult task because of mainly two reasons: (1) the complexity of the search space and (2) the evaluation cost. Regarding the search space, its complexity is given by the heterogeneity of the domains of decision variables (e.g. categorical, binary, integer, real, etc.). As for the evaluation cost, it is due to the need to train and validate the model on the dataset to obtain its performance. For these reasons,

- *J. S. Angarita-Zapata and A. D. Masegosa are with the DeustoTech-Faculty of Engineering at the University of Deusto, Av. Universidades, 24, 48007 Bilbao, Spain.*
  E-mail: js.angarita@deusto.es
- *I. Triguero is with the Computational Optimisation and Learning Lab (COL) of the School of Computer Science, University of Nottingham, Nottingham NG8 1BB, United Kingdom.*

.

the optimisation of ML pipelines usually requires a long time to provide good results, specially for big datasets but, a lengthy optimisation process may be prone to overfitting [18], [19], [20]. Furthermore, the great variability of data preprocessing techniques, ML methods and hyperparameters configurations makes the modelling and coding process hard and complex.

Although meta-learning has arisen as a promising strategy to solve some of the previous problems, especially the computational cost of pipelines optimisation, this approach also presents some drawbacks. It is challenging to find a representative set of meta-features good or large enough to characterise very diverse datasets. The latter can lead to the risk that in very different ML tasks to those stored in the knowledge base, the meta-learning component and its meta-features may suggest pipelines that do not work properly on the dataset at hand [21].

To overcome the aforementioned limitations, we propose a simple strategy for AutoML that does not require a pipeline optimisation process (making it more scalable and less prone to overfitting) and does not rely on extracted meta-features. The proposed approach, named *AutoEn*, is a simple yet effective ensemble-based AutoML method for the automatic generation and optimisation of ensembles [22] from a predefined set of pipelines composed of a sequence of preprocessing techniques plus one ML classifier. The novelty of our method lies in automatically generating sets of competitive pipelines without focusing on the fine-tuning of these pipelines. Thus, we conceive a new and much simpler approach that achieves a competitive performance in comparison to the results reported in the state-of-the-art. The main objectives of this paper are:

- To demonstrate the suitability and competitiveness of AutoEn and its simple online search strategy purely based on the construction and optimisation of ensembles of multiple classifiers rather than in the search, construction and tuning of individual pipelines usually done by current AutoML approaches.
- To compare the performance of the proposed approach against state-of-the-art AutoML such as TPOT [13] and Auto-sklearn [4] on the so-called AutoML benchmark [19] using 16 binary and 12 multi-class supervised classification problems.

The rest of this paper is structured as follows. Section 2 presents background and related work about AutoML with special emphasis on methods that automatise complete ML pipelines. Section 4 introduces the AutoML method proposed in this paper. Section 4 exposes the experimental framework and Section 5 analyses the results obtained. Finally, conclusions are discussed in Section 6.

2 Background and Related Work

ML is the field focused on algorithms able of carrying out prediction/classification tasks by means of an automatic learning process without the need of being explicitly programmed by human intervention [2], [23]. Applications like e-mail spam and malware filtering, automatic speech recognition, predictive maintenance in industry, among others, are built upon ML. As a common denominator, all these applications required well-designed and efficient ML pipelines that include data preprocessing as a key factor [7], [8].

According to [11], a pipeline P can be defined as a combination of algorithms A that transforms input data X into target values Y. Let A be defined as

$$A = \{A_{preprocessing} \cup A_{feature} \cup A_{algorithm}\} \quad (1)$$

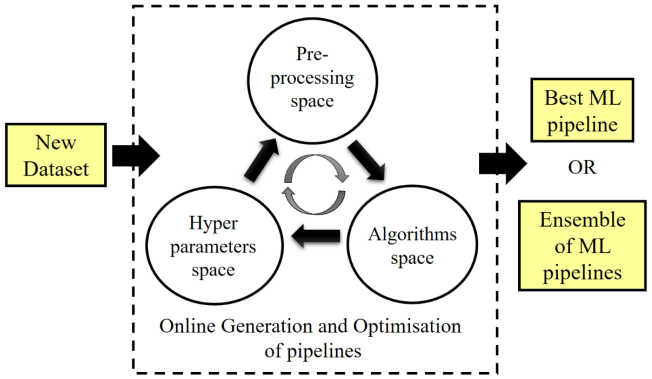
wherein $A_{preprocessing}$ is a subset of preprocessing techniques, $A_{feature}$ a subset of feature engineering methods, and $A_{algorithm}$ a ML algorithm with configuration of hyperparameters $\lambda^i \in \Lambda$. In order to build a ML pipeline with this structure, human effort and high computational capacities are needed because there is no pipeline that can achieve good performance on every learning problem [10], [24]. This is usually done by data scientist who make use of their specialised knowledge for it or non-expert users who tackle the problem by means of a trial and error approach that causes the success of ML to happen at a great cost [10].

AutoML is an emerging area that aims at automatically finding the best combination of preprocessing techniques, ML algorithm and its hyperparameters, according to a certain performance measure on a given dataset without being specialised in the problem domain wherein this data comes from [1]. It allows us to reduce human bias and improving computational costs by making the construction of ML applications more efficiently. More formally, the process consists of identifying the most promising combination P_{A^i,λ^i} that optimise a given performance metric when P_{A^i,λ^i} is trained on training data $D^{(i)}_{train}$ and evaluated on test data $D^{(i)}_{test}$.

Current literature [1], [10], [11] reports a variety of AutoML methods. They differ depending on which stages of a ML pipeline are going to be automated, for example, data preprocessing [25], [26], [27], algorithm selection and hyperparameters [5], [12], [28], [29], or even the entire pipeline. In this paper, we are focused on the automation of ML pipelines composed of data preprocessing techniques and a classifier algorithm with their respective hyperparameter configuration.

In the literature, we can usually find two main categories of AutoML methods to automatise the construction of ML pipelines, based only on optimization or on metalearning + optimization. The fist one use only optimization to generate, tune, and assess pipelines, as represented in Figure 1. In this case, when new data is fed into this type of methods, they start to build multiple possible combinations of pipelines from a predefined search space of preprocessing, ML methods and hyperparameters. Then, after a given time or a fixed number of iterations, they return the best pipeline found or construct an ensemble from a set of competitive pipelines identified during the optimisation. From this perspective, the ML pipeline building problem consists of finding pipeline structures $P_{A,\lambda}$ that minimise a cross-validation loss function (Equation 2). Representative AutoML methods for this category can be Auto-WEKA [12], TPOT [13], H2O [17], among others [30], [31].

Fig. 1. General approach of AutoML methods based on a pure optimisation search strategy.

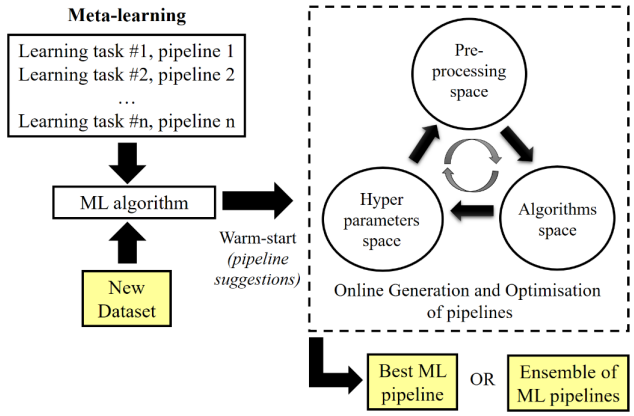


$$P_{A^*,\lambda^*} = \underset{A^{(i)}\in A,\lambda^{(i)}\in\Lambda}{argmin} \frac{1}{K}\sum_{i=1}^{k}\gamma\left(P_{A^{(i)},\lambda^{(i)}}, D_{train}, D_{test}\right) \quad (2)$$

As shown in Equation 2, this search process can be considered as a black-box optimisation problem that it is not easily solvable as the search space can be large and complex. This equation is usually non-smooth and derivative-free, and the convergence speed is a critical problem for building ML pipelines [11]. Some methods to solve this equation are grid search [17], genetic programming [13], and Bayesian optimisation [5].

Regarding AutoML methods that combines optimization with meta-learning [4], [14], [15], their working is shown in Figure 2.

Fig. 2. General approach of AutoML methods with a search strategy of pipelines based on meta-learning and optimisation.



According to Figure 2, in an off-line phase, optimisation is used to find ML pipelines with high performance on a repository of datasets. Simultaneously, a group of meta-features are extracted to characterise each dataset that describe its representative characteristics such as data skewness, entropy of targets, number of instances, variance of the data, among others [15]. Afterwards, the meta-features and the optimised pipelines are stored in a meta-knowledge base wherein each instance contains the set of meta-features describing every dataset and its corresponding optimised pipeline. Then, in the online stage when new input data comes in, the meta-learning component recommends pipelines that are likely to perform well on the input dataset. This selection of pipelines is then used for a warm-start of the optimisation process.

In a nutshell, there are consolidated AutoML methods that automatise the complete ML workflow. As a common factor, the core of their pipeline search strategies is focused on generating and fine-tuning individual pipelines that later can be or not integrated on ensembles. Nonetheless, this online optimisation of pipelines is a demanding process because 1) it involves complex search spaces; 2) the evaluation of the objective function usually is computationally expensive; and 3) to establish a priori the best time budget for the optimisation process is a difficult task since in complex or big datasets collecting good pipelines can require a long time, while in small or simpler datasets an excessive time budget is prone to overfitting [18], [20]. Although AutoML methods that combine meta-learning with optimisation reduce the impact of some of these issues, they also present some drawbacks. The definition of a set of meta-features able to characterise very diverse datasets and to predict pipelines' performance is a difficult task without a solid evidence to guide this design process. In this way, there is a risk that in very different supervised learning tasks to those included in the meta-knowledge base, the meta-learning component may suggest pipelines that are not competitive to warm-start the optimisation process. [21]. In this context, our AutoML enhances the state-of-the art by automatically generate and optimise ensembles during the online search phase of AutoML. Thus, rather than focusing on single pipelines, we approach the finding of strong ensembles of multiple pipelines that is a more simpler and efficient strategy to deal with the aforementioned issues.

3 AutoEn: An AutoML method based on Ensemble Learning

In this section, we introduce AutoEn. First, Section 3.1 motivates the need for the proposed method. Then, Section 3.2 presents the general workflow of AutEn and details of how it does the automated selection and construction of ensembles.

3.1 Motivation

As we stated before, the core the existing AutoML methods is the optimisation of individual ML pipelines, which could be enhanced with a preliminary off-line stage based on meta-learning. However, the current solutions suffer from a number of issues that motivate the design of AutoEn:

- **Optimisation - Overfitting issues**: Optimisation of pipelines is supposed to require long time budgets to find competitive solutions. However, previous research has corroborated that longer execution times may cause overfitting issues due to hyperparameters tuning [18], [19], [20]. Therefore, higher execution times do not always lead to better results as we could expect. In addition, sometimes for medium- and small-size datasets, competitive pipelines can be quickly found without the need of allocating long

time budgets for the optimisation search. Thus, optimisation of pipelines can be an expensive procedure when the goal is simply to make good enough predictions.

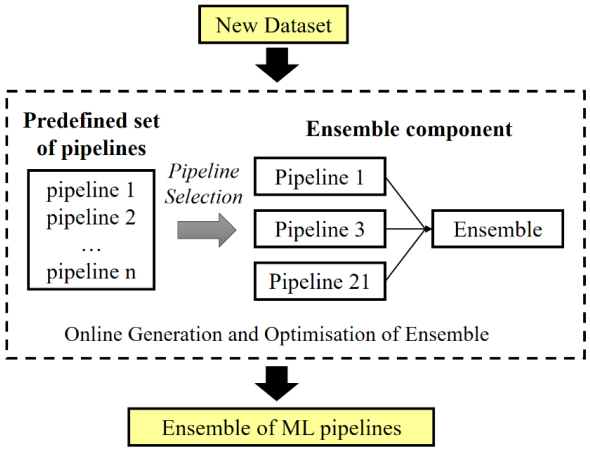
- **Optimisation - Reduced scalability**: Although long time budgets of optimization may be prone to return pipelines with overfitting issues, other times the results can be promising. Particularly, when optimisation deals with small- or medium-size datasets, many diverse pipeline structures can be generated, tuned and test because the complexity of the learning task at hand is influenced by its data size. On the other hand, the latter may stop happening as the data size grows. In this scenario, it is harder to tune and test multiple pipelines, as the optimisation becomes expensive, and the set of candidate pipelines could decrease which ends up affecting the performance of the final solutions.
- **Meta-learning - Representativeness of Meta-features**: Meta-learning can potentially reduce the cost of the optimisation process. Nevertheless, there is a risk that the meta-features can only described the learning tasks in the meta-knowledge base. The latter is understandable because is difficult to find a set of meta-features good enough to characterise very diverse datasets. Therefore, there is a risk that for supervised learning tasks not similar to the ones stored in the meta-knowledge base, the meta-learning component recommends pipelines that do not perform as well as expected in certain supervised learning tasks [21]. Thus, the optimisation process could be warm-started with pipelines that are not competitive on the input data.

Having in mind the motivations presented above, we want to conceive a competitive and simple AutoML method that adjusts to the complexity of the data without the need to define a time budget for its execution. This means that for small- and medium-size datasets, the method is able to find quick solutions; conversely, for large datasets, the method may take longer time to find competitive pipelines. The aforementioned condition could be satisfied not only suggesting single pipelines whose performance could vary drastically from one learning task to another. In this sense, we propose an AutoML method based on the search and optimisation of ensembles of multi-classifiers that does not commit to a single ML workflow. Accomplishing such goal requires the ensemble contains models that individually are different in nature and strong doing individual predictions. Therefore, we introduce an AutoML method based on ensemble learning for combining high-performance instantiations of different pipelines.

3.2 AutoEn design and workflow

Based on the motivation presented above, Figure 3 introduces the architecture of AutoEn. This AutoML replaces the online search and optimisation of individual pipelines by the automated generation of ensembles. Similarly to Auto-sklearn, it has in its inner structure a base of diverse pipelines that were trained on different learning tasks during the construction of AutoEn. The pipelines within this base consist of a sequence of data preprocessing technique and a classifier algorithm. The underlying idea of using a set of predefined pipelines is offering for every learning task, ML workflows different in their nature that can be less prone to overfitting issues. Differently from Auto-sklearn, we will not use meta-features to decide what pipelines are part or not of the ensemble construction.

Fig. 3. General workflow of AutoEns that is composed of two main blocks: 1) a set of predefined pipelines trained on different learning tasks and 2) an ensemble component that generates a multi-classifier system, from the set of pipelines, when a new dataset comes in.



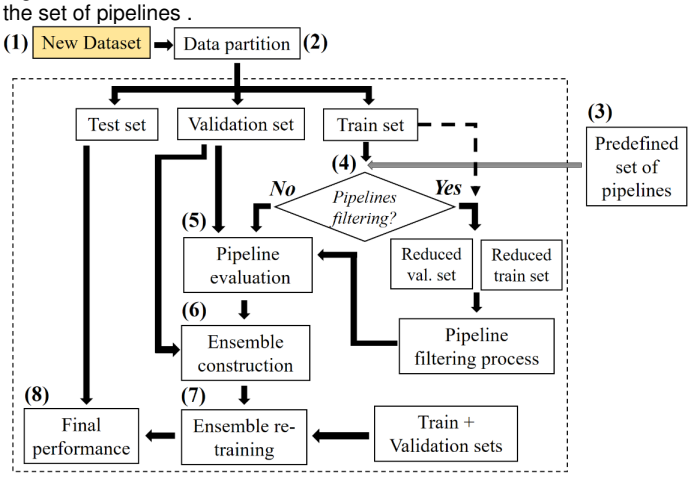
A meta-learning approach should be able to assist us in finding the most promising pipelines to be part of the ensemble. However, we observed in [21] that this idea can only properly work under a good enough set of meta-features able to characterise very different datasets, which unfortunately it is not always the case. To avoid relying on meta-features, when new input data comes in AutoEn, it assesses the performance of the base of pipelines on a validation set. Depending on their validation errors, the pipelines can or cannot be part of the ensemble construction. Thus, we avoid the drawbacks of meta-learning when facing very diverse learning tasks.

Details about the automatic selection and construction of the ensemble based on these pipelines is presented in Figure 4. In step (1), it receives a new and unseen dataset that later in step (2) is split into train, validation and test sets. Then, in step (3), we retrieve the set of available ML pipelines.

Having the list of optimised pipelines at hand, in step (4), it is possible to filter them to improve the computational efficiency or not. The latter decision implies either to directly train the complete list of pipelines on the training set (”No” path) or to reduce the list to have a smaller set (”Yes” path) to enhance the computational efficiency. The former option is the default mode of our AutoML method, whilst the latter implies selecting a sample of the training set (dashed line in Fig. 4), that in turn is divided into training and validation set (reduced train and validation set in Fig. 4), to filter the pipelines and choose the fastest ones.

After selecting the complete list of pipelines or a reduced version of it, the next step is training the available pipelines in the complete train set and ranking them according to their errors in the validation set (step 5). After that, we proceed to build an ensemble from this set of trained pipelines

Fig. 4. Automated selection and construction of the ensemble based on the set of pipelines .



(step 6). We implemented the ensemble selection approach introduced by [32]. The latter is a greedy procedure that starts from an empty ensemble and then iteratively adds the model that maximises ensemble performance in the same validation set in which the pipelines were previously assessed.

For our AutoEn, we initialise the ensemble with one element, which is the pipeline with the highest performance of the validation set. From the one-element ensemble, we start to append new pipelines, with uniform weights, to the one-element ensemble until reaching a predefined ensemble size. It is important to note the inclusion of new pipelines to the ensemble allows repetitions; this means one high-performance pipeline can appear multiple times in the final ensemble. Finally, in step (7), the pipelines that composed the ensemble are re-trained on training+validation partitions to finally make classification on the unseen test set. For this final step, the pipelines that appear multiple times in the ensemble are only re-trained once; thus, we can decrease the computational cost of this final step.

4 Experimental Framework

This paper seeks to answer the question whether it is possible to conceive an AutoML framework based on multi-classifier ensembles and able to make good enough predictions in supervised classification problems. To accomplish such purpose, we compare the proposed method against the results of AutoML methods reported in the benchmark published by Gijsbers et al. [19]. The latter is an open-source AutoML framework that uses public datasets to conduct a thorough comparison of 4 AutoML methods: H2O, AutoWEKA, Auto-sklearn and TPOT.

In this section, we show the factors and issues related to the experimental study. We provide details of the problems chosen for the experimentation (Section 4.1). Then, we introduce the measures employed to evaluate the performance of the methods (Section 4.2). Finally, we present the methods used for comparison (Section 4.3).

4.1 Binary and Multi-class supervised problems

From the AutoML benchmark introduced above [19], we use 16 datasets of binary classification problems and 12 of multi-class problems. The total of 28 datasets were used in previous AutoML papers [12], AutoML competitions [33] and ML benchmarks [34]. The datasets vary in the number of samples and features by orders of magnitude, and vary in the occurrence of numeric features, categorical features and missing values. The current list of datasets is available on OpenML [1] [35]. Table 1 presents a summary of binary datasets used for the experimentation. It shows the number of features and instances (Feat. - Instan.), number of numeric and nominal features (Num. Feat. and Nom. Feat), and number of missing values (Missing Val.) on each dataset.

TABLE 1
Binary datasets

dataset	Feat. - Instan.	Num. Feat.	Nom. Feat.	Missing Val.
adult	15 - 48842	6	9	6465
amazon_employ	10 - 32769	0	10	0
albert	79 - 425240	26	53	2734000
apsfailure	171 - 76000	170	1	1078695
bank-mark	17 - 45211	7	10	0
blood-transf	5 - 748	4	1	0
christine	1637 - 5418	1599	38	0
credit-g	21 - 1000	7	14	0
higgs	29 - 98050	28	1	9
jasmine	145 - 2984	8	137	0
kc1	22 - 2109	21	1	0
kr-vs-kp	37 - 3196	0	37	0
nomao	119 - 34465	89	30	0
numerai28.6	22 - 92320	21	1	0
phoneme	6 - 5404	5	1	0
sylvine	21 - 5124	20	1	0

TABLE 2
Multi-class datasets

dataset	Feat. - Instan	Num. Feat.	Nom. Feat.	Classes
car	7 - 1728	0	7	4
cnae-9	857 - 1080	856	0	9
connect-4	43 - 67557	0	43	3
dilbert	2001 - 10000	2000	1	5
fashion-mnist	785 - 70000	784	1	10
jannis	55 - 83733	54	1	4
jungle	7 - 44819	6	1	3
mfeat-factors	217 - 2000	216	1	10
segment	20 - 2310	19	1	7
shuttle	10 - 58000	9	1	7
vehicle	19 - 846	18	1	4
volkert	181 - 58310	180	1	10

Table 2 introduces an overview of multi-class datasets considered for the experimentation. It summarizes the number of features and instances, number of numeric and nominal features, and the number of classes. In this case, any of these datasets contain missing values.

4.2 Performance measures

In this section, we present the metrics used to measure the performance of methods in binary and multi-class problems, the hardware choice to carry out the experimentation, specifications about how we measure the computational time of AutoEn, and the statistical test considered to make comparison between the performance of the methods.

- **Metrics:** For the results of this paper, we follow the same experimental set-up proposed in the AutoML benchmark in order to make fair comparisons, and therefore, we use the same performance measures.

1. https://www.openml.org/s/218/data

In particular, the area under the receiver operator curve (roc_auc_score) is used for binary classification problems and log loss_score is used for multi-class problems. Besides, both measures are averages of a ten-fold cross-validation.

- **Hardware choice and resource specifications:** We opted to use a cluster based on Centos 7.6 with kernel 3.10.0, and usig QuadCore Intel Xeon processors, with 16 cores and 16 GB of RAM each. Here, would like to highlight that despite the multi-core character of the CPU, the implementation of AutoEn is single-core.
- **Computational time:** To measure how much time AutEn takes since new input data comes in until it makes final predictions, we measure the execution time in seconds extracted from the system clock. Concretely, we start to measure this time when AutoEn receives the input data to make the data partitions until it outputs the final classification of the ensemble built during the online search of the method.
- **Statistical tests:** We made use of non-parametric statistical tests to assess the differences in performance of the methods. Two statistical tests are used following the guidelines proposed in [36]. First, the Friedman's test for multiple comparisons is applied to check whether there are differences among the methods. Then, the Holm's test is used to check whether the differences of the Friedman ranking are statistically significant or not.

4.3 Competitors and baselines

For the experimentation carried out in this paper, AutoEn has used the pipelines list stored in Auto-sklearn's meta-learning component. These pipelines were generated and tunes by means of sequential mode-based optimisation [5] using a search space of 15 classifiers and 18 preprocessing techniques, all of them implemented in scikit-learn ML library. The classifier can be categorised in linear models, support vector machines, discriminant analysis, nearest neighbors, naive Bayes, decision trees and ensembles. Additionally, data preprocessing techniques include rescaling, imputation of missing values, one-hot encoding, feature selection, kernel approximation, feature clustering and polynomial feature expansion. The interested reader can consult [37] in order to know more details about these classifiers and preprocessing techniques.

As we mentioned above, the referenced AutoML benchmark makes comparisons of 4 AutoML state-of-the-art methods: H2O, AutoWEKA, Auto-sklearn and TPOT. Although this comparison offers helpful insights into the differences between the methods, in this paper, we make comparisons against methods based on scikit-learn implementations such as AutoEn. The latter decision seeks to contrast results under the same or similar implementation circumstances. This statement is supported in the differences of their inner implementations. More precisely, for the case of H2O and AutoWEKA, their base of ML algorithms and pre-processing methods are implemented in Java, and H2O and Weka ML libraries, respectively; meanwhile, Auto-sklearn and TPOT are based on python scikit-learn ML library. Additionally, the base of ML algorithms and prepre-processing techniques are not the same for neither of the AutoML methods. For instance, H2O includes neural networks whilst Auto-sklearn does not have.

In addition to a comparison with Auto-sklearn and TPOT, we include the same baselines methods of the referenced AutoML benchmark. They are a constant predictor, which always predicts the class prior (CnstPrd), an untuned Random Forest (RF), and a tuned Random Forest (tuned_RF). We also include as baseline the approach (BestV_ML) proposed in [21]. It uses Auto-sklearn's meta-learning component and extract its pipelines recommendations. Then, all the pipelines are trained and based on their validation errors the best performing pipeline is selected to make classifications on the unseen test set.

Auto-sklearn [4] and TPOT [13] were deployed with their default hyperparameter values and search spaces, since most users will use them in this way, and their time budget per fold was 1 and 4 hours. RF uses scikit-learn 0.20 default hyperparameters and tuned_RF is built with 2000 estimators. For AutoEns and AutoEn_economy (AutoEn_ec), their main parameters are shown in Table 3.

TABLE 3
Initial hyperparameters of AutoEn for its two operative modes

Hyperparameters	AutoEn	AutoEn_ec
Ensemble size	50	50
Data partition per fold	train: 60%, validation: 20%, test: 20%	train: 60%, validation: 20%, test: 20%
Data partition to filter pipelines	-	10% of the original train set
Time left for a pipeline to be trained in the pipelines filtering phase	-	36 seconds

5 Analysis of Results

This section analyses the results obtained from different angles. Specifically, our aims are:

- To compare the competitiveness and the significance of AutoEn performance w.r.t. AutoML competitors and baselines in binary and multi-class learning tasks.
- To investigate the main benefits of an AutoML method without allocating any time budget when dealing with classification datasets of different sizes.
- To contrast the differences in performance and runtime between AutoEns and its economy mode.

Table 4 shows the mean roc_auc_score and log_loss_score values of our AutoML method in its default (AutoEn) and economy modes (AutoEn_ec), the AutoML competitors (AutoSkl and TPOT) and the baseline methods on the test phase. The best result for each dataset is highlighted in bold-face. Note that for binary datasets, the higher the performance value (roc_auc) the better, whilst for multi-class datasets, the lower the loss, the better.

Observing Table 4, we can point out the following:

- As general overview, in binary datasets there are ties of at least 2 methods in 6 datasets: *adult*, *albert*,

TABLE 4
Mean roc_auc (binary problems) and log_loss (multi-class problems) values obtained by AutoEn, AutoEn_ec, the AutoML competitors and the baseline methods. Values highlighted in bold are the highest performance obtained by any of the methods on every dataset.

Type	dataset	AutoSkl_1h	AutoSkl_4h	TPOT_1h	TPOT_4h	ConstPrd	RF	tuned_RF	BestV_ML	AutoEn	AutoEn_ec
Binary	adult	**0.930**	**0.930**	0.927	0.929	0.500	0.909	0.909	0.917	0.920	0.920
	amazon_employee	0.856	0.849	**0.870**	0.868	0.500	0.864	0.863	0.859	0.864	0.863
	albert	**0.748**	**0.748**	0.724	0.724	0.500	0.738	0.738	0.739	0.702	0.704
	apsfailure	0.991	**0.992**	0.990	0.992	0.500	0.991	0.991	0.990	0.991	0.989
	bank-marketing	**0.937**	**0.937**	0.934	0.934	0.500	0.931	0.931	0.931	0.934	0.935
	blood-transfusion	0.757	**0.763**	0.724	0.707	0.500	0.686	0.689	0.730	0.741	0.735
	christine	0.830	**0.831**	0.813	0.820	0.500	0.806	0.810	0.816	0.825	0.811
	credit-g	0.783	0.782	0.786	0.771	0.500	0.795	**0.796**	0.781	0.786	0.795
	higgs	0.793	**0.809**	0.802	0.805	0.500	0.803	0.803	0.798	0.807	0.807
	jasmine	0.884	0.883	0.885	**0.890**	0.500	0.888	0.889	0.878	0.881	0.883
	kc1	0.843	0.839	0.841	0.845	0.500	0.836	0.842	0.840	**0.852**	0.844
	kr-vs-kp	**1.000**	**1.000**	**1.000**	**0.999**	0.500	0.999	**1.000**	**1.000**	0.998	0.998
	nomao	**0.996**	**0.996**	0.995	**0.996**	0.500	0.995	0.995	0.995	**0.996**	0.969
	numerai28.6	0.529	0.530	0.525	0.526	0.500	0.520	0.521	0.529	**0.532**	0.530
	phoneme	0.963	0.962	0.969	**0.973**	0.500	0.965	0.966	0.970	0.972	0.970
	sylvine	0.990	0.991	0.992	**0.995**	0.500	0.983	0.984	0.989	0.989	0.990
Multi-class	car	0.010	0.010	**0.000**	**0.000**	0.836	0.144	0.047	0.105	0.065	0.070
	cnae-9	0.171	**0.168**	0.196	0.189	2.197	0.301	0.297	0.205	0.178	0.182
	connect-4	0.426	0.387	0.400	**0.376**	0.845	0.495	0.478	0.483	0.460	0.461
	dilbert	0.097	0.063	0.217	0.156	1.609	0.328	0.329	**0.033**	**0.033**	0.034
	fashion-mnist	0.354	0.358	0.651	0.700	2.303	0.361	0.362	0.345	**0.314**	0.315
	jannis	0.705	**0.685**	0.732	0.721	1.109	0.728	0.729	0.710	0.702	0.701
	jungle	0.234	0.223	0.198	**0.158**	0.935	0.438	0.402	0.216	0.215	0.215
	mfeat-factors	0.099	0.093	0.138	0.124	2.303	0.234	0.201	0.160	0.093	**0.088**
	segment	0.060	0.063	**0.052**	0.055	1.946	0.084	0.069	0.074	0.061	0.069
	shuttle	0.001	**0.000**	3.444	**0.000**	0.666	0.001	0.001	**0.000**	**0.000**	0.001
	vehicle	0.395	0.379	0.414	0.397	1.386	0.497	0.486	0.352	0.341	**0.332**
	volkert	0.945	0.925	1.011	0.989	2.053	0.980	0.979	0.925	**0.858**	**0.858**

apsfailure, *bank_marketing*, *kr-vs-kp*, and *nomao*. With respect to multi-class datasets, there are 4 ties of at least 2 methods in *car*, *dilbert*, *shuttler* and *volkert* datasets. Summarising , there is no AutoML method that consistently outperforms all AutoML competitors on binary and multi-class; and AutoML scores are relatively close to the performance of tuned_RF. Additionally, any of the AutoML methods outperforms the tuned_RF on the complete list of datasets.

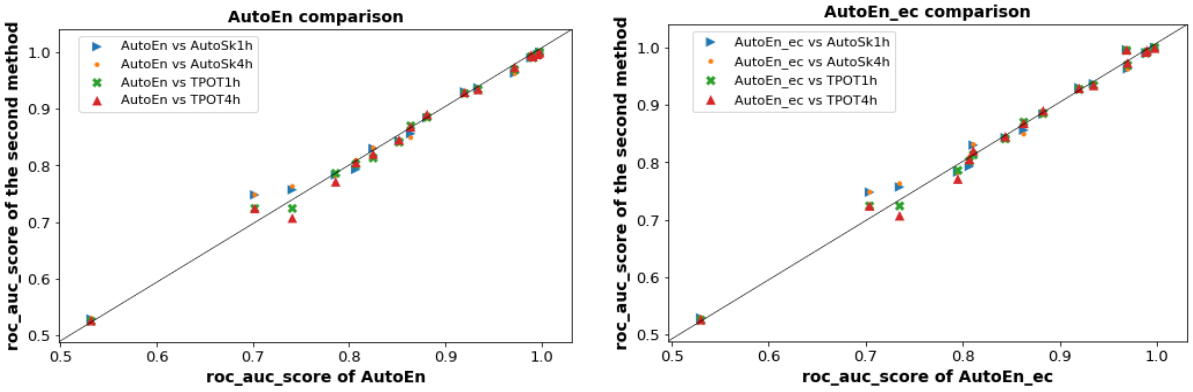
- Another interesting aspect of results in Table 4 is Auto-sklearn and TPOT results are quite similar under the time budgets of 1 and 4 hours per fold. The latter means that the optimisation process carries out by those 2 methods only brings slight score improvements. Particularly for binary-datasets, Auto-sklearn only achieves improvements with a longer execution time in *apsfailure*, *blood-transfusion*, *christine*, *numerai28.6* and *sylvine*. Those improvements range from 0.001 to 0.007. In *amazon_employee* , *credit-g*, *jasmine*, *kc-1* and *phoneme*, Auto-sklearn obtains worse performance with 4 hours execution time than the performance it obtains with only 1 hour. For the case of multi-class datasets, although the improvements are greater, they do not reach at least 10% of enhancement; datasets *cnae-9*, *connect-4*, *dilbert*, *jannis*, *vehicle* and *volkert* exemplified the aforementioned situation. As well as binary datasets, sometimes Auto-sklearn performance decreases with longer execution times in *fashion-mnist* and *segment* datasets. Such worsening under longer time budgets allocated for the its optimisation could be due to overfitting issues such as it has been corroborated by previous research [19], [21].
- For the case of TPOT, the behaviour of improvements from 1 to 4 hours are quite similar w.r.t Auto-sklearn. In binary and multi-class datasets there are datasets wherein overfitting issues also seem to affect the performance of TPOT (*amazon_employee*, *blood-transfusion*, *credit-g*, *fashion-mnist*, and *segment* datasets). Besides, in only one datasets there are significant improvements w.r.t the shortest execution time (*shuttle* dataset).

TABLE 5
Binary datasets: Friedman's average ranking and p-values obtained through Holm post-hoc test using AutoSkl_4 as control method

Methods	Av. Ranking	p-values
AutoSkl_4h	3.4412	-
AutoSkl_1h	3.8529	0.6917
TPOT_4h	3.9118	0.6504
AutoEn	4.5588	0.2818
AutoEn_ec	5.0294	0.1261
TPOT_1h	5.3824	0.0615
tuned_RF	6	**0.0137**
BestV_ML	6.4118	**0.0042**
RF	6.4118	**0.0042**
ConstPrd	10	**0**

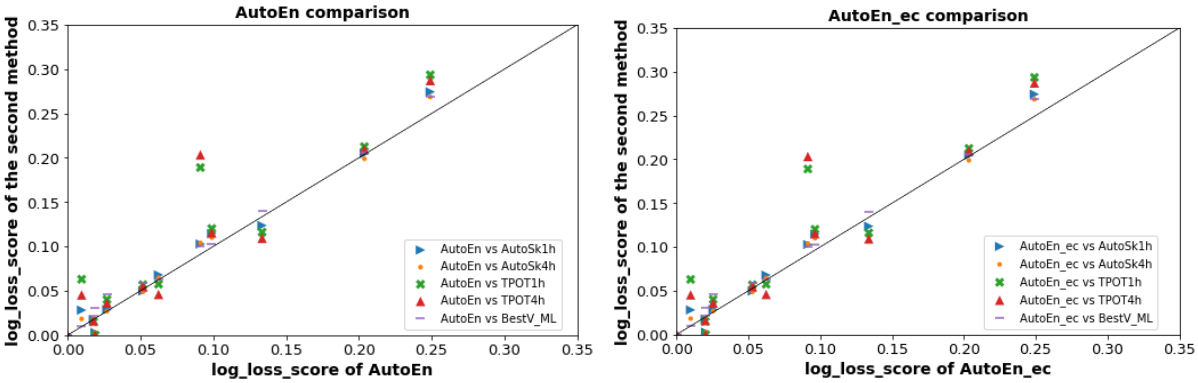
To assess whether the differences in performance observed in Table 4 are significant or not, we made use of non-parametric statistical tests. Two statistical tests have been applied following the guidelines proposed in [36]. First, the Friedman's test for multiple comparisons has been applied to check whether there are significant differences among our AutoEn in its two modes, the AutoML competitors and the baseline methods. Then, the Holm post-hoc test has also been applied to assess the significance of the differences in performance.

Table 5 exposes the test outcomes for binary datasets and again p-values lower than 0.05 are shown in bold. In this case, AutoSkl_4h is the method in the first position of ranking. However, it is interesting to note AutoSkl_4h is only



(a) AutoEn (b) AutoEn_ec

Fig. 5. roc_auc_score over 16 *binary* datasets. Points *below* the $y = x$ line correspond to datasets for which our method performs better than a comparison algorithm



(a) AutoEn (b) AutoEn_ec

Fig. 6. log_loss_score over 12 *multi-class* datasets. Points *above* the $y = x$ line correspond to datasets for which our method performs better than a comparison algorithm

TABLE 6
Multi-class datasets: Friedman's average ranking and p-values obtained through Holm post-hoc test using AutoEn as control method.

Methods	Av. Ranking	p-values
AutoEn	3.25	-
AutoSkl_4h	3.25	1
AutoEn_ec	3.5	1
TPOT_4h	4.25	1
AutoSkl_1h	4.4167	1
BestV_ML	5.0833	0.6900
TPOT_1h	5.9167	0.1858
tuned_RF	7.25	**0.0084**
RF	8.25	**0.0004**
ConstPrd	9.833	**0**

statistical better than the baseline methods. In this sense, it seems that supervised classification of binary problems is quite easily approachable by all the AutoML methods regardless what search strategy they can use.

Table 6 summarises test results for multi-class datasets and significant differences with p-values lower than 0.05 are highlighted in bold. AutoEn is the method in the first position of Friedman's average ranking and such as binary problems, there are no significant statistical differences among any of the AutoML methods and the BestV_ML algorithm.

From Tables 5 and 6, it is possible to observe that despite the Friedman test provides a ranking of the methods evaluated, there were no statistical differences among some of them, particularly between the AutoML methods. Therefore, in order to better assess the performance of AutoEn and AutoEn_ec, we introduce the following analyses.

In Figures 5a and 5b, each point compares AutoEn and AutoEn_ec to a second method on binary problems, respectively. In both Figures, the x-axis position of the points is the roc_auc_score of our method on binary datasets. The y-axis position represents the performance metric of the comparison algorithms that are not statistically better than AutoEn and AutoEn_ec in binary problems. Points below the y=x line correspond to datasets for which our method performs better than a second method.

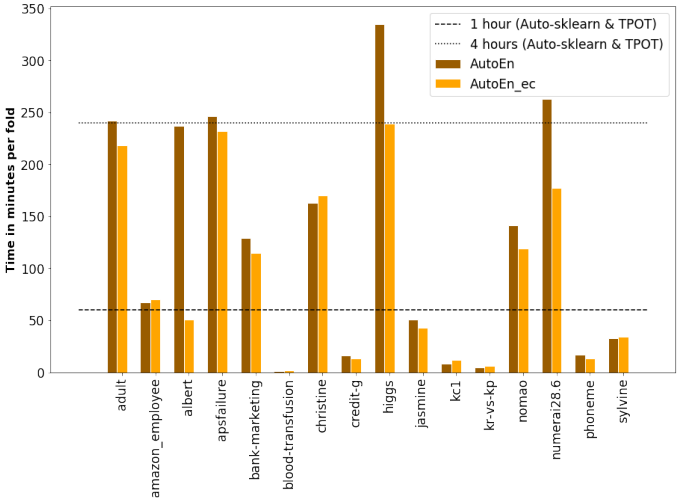
Similarly, Figures 6a and 6b compare AutoEn and AutoEn_ec to a second method in multi-class datasets. Again, the methods for comparison are the ones that were not statistically better than our approach in multi-class problems. Note that log_loss_score is a minimisation metric, therefore, points above y=x are datasets wherein AutoEn and AutoEn_ec have a higher performance than a second method. For this latter case, the values are normalised

between 0 and 1, and outliers were discarded. The reason to normalise the values is because, as was shown in Table 4, there are multi-class problems wherein some methods obtained values greater than 1.

In a general way, it can be observed for binary datasets, AutoEn and AutoEn_ec have in the majority of the cases equal results to the comparison methods. On the other hand, in multi-class problems, our two approaches have a performance generally better than the other methods.

Finally, as the computational cost is also a relevant factor in AutoML, Figure 7 shows the execution time per fold that AutoEn took to make classifications on binary datasets. The Figure also plots the execution time of its economy mode. Additionally, Figure 7 has straight lines that represent the time thresholds of the two optimisation execution times associated with Auto-sklearn and TPOT. Thus, the purpose is to check whether the execution times of AutoEn can be in-between, below or beyond these two thresholds.

Fig. 7. AutoEn execution times in binary datasets under its default and economy modes.

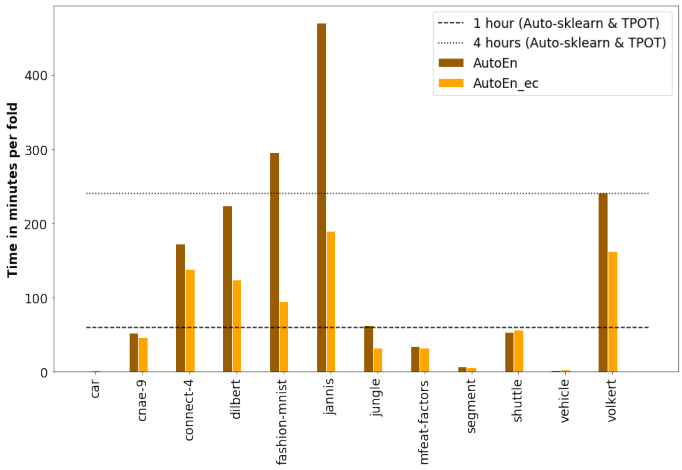


In Figure 7, the execution time of AutoEn in its default mode has three different behaviours. First, in *blood-transfusion*, *credit-g*, *jasmine*, *kc1*, *kr-vs-kp*, *phoneme* and *sylvine* datasets, AutoEn spends less than 60 minutes per fold that is the shortest execution time allocated for Auto-Sklearn and TPOT. Secondly, for *adult*, *amazon_employee*, *albert*, *bank-marketing*, *christine* and *nomao* datasets, its execution time is in-between the the two thresholds. Finally, for the remaining datasets (*apsfailure*, *higgs*, *numerai28.6*), AutoEn takes more than 4 hours of time. W.r.t. AutoEns_ec, its execution time is below 4 hours in all binary datasets, and as it was observed in Figure 5b, its performance remains quite similar to AutoEn.

Figure 8 summarises AutoEn and AutoEn_ec execution times per fold of in multi-class datasets. Besides, it also shows the 2 time thresholds of the two optimisation execution times carried out by Auto-sklearn and TPOT. Significant reductions of time can be seen in *dilbert*, *fashion-mnist*, *jannis* and *volkert* datasets, without affecting the performance as was observed in Figure 6b.

Summarising, there are interesting considerations that can be extracted from the analysis of results presented

Fig. 8. AutoEn execution times in multi-class datasets under its default and economy modes.



above. First, the assumption that only longer execution times for the online phase of AutoML bring high-performance results can be arguable. As was observed, AutoEn is able to generate competitive results for medium- and small-size datasets (binary: *blood-transfusion*, *credit-g*, *kc1*, *kr-vs-kp*, *phoneme* datasets; multi-class: *car*, *segment*, *vehicle* datasets) much faster then the lowest execution time of Auto-sklearn and TPOT. Secondly, the fine-tuning of hyperparameters of individual pipelines, under long execution times in the online phase of AutoML, in the majority of the cases only brings slightly improvements to final results.

6 Conclusions

In this paper, we have proposed an AutoML method for the automated generation of ensembles from a predefined set of ML pipelines.The proposed AutoML method was tested against Auto-sklearn and TPOT, two state-of-the-arts AutoML methods, in multiple binary and multi-class supervised problems. The main conclusions drawn from the analysis of the results are the following:

- The online phase of AutoML should be focused on building strong ensembles rather than individual ML pipelines. In this context, computational effort should be directed to generate high-performance pipelines in the offline phase of AutoML, which later contribute to producing competitive and fast ensembles during the online stage of AutoML.
- Ensembles of multi-classifiers can be more adaptable to datasets of different sizes, which lead to optimising computational efficiency of AutoML. On the one hand, high-performance solutions can be achieved for small- and medium-size dataset faster than traditional optimisation approaches (e.g. bayesian optimisation, evolutionary programming). Contrary, for big datasets, although the time consumption could increase, the computation effort of the online phase is focused on finding competitive combinations of multi-classifiers, which are potentially highly parallelisable and less prone to overfitting issues, compared to generating and fine-tuning individual

pipelines that are even more costly to be evaluated in big datasets.

To sum up, this method opens the path towards AutoML frameworks purely based on ensemble strategies. Further research we would like to explore are: 1) a base of offline optimised pipelines that is able to incorporate new pipelines as it faces a new and seen problem which requires pipelines different from the ones it already contains; 2) design an online ensemble strategy that is able to deal with big datasets without the need of using the complete dataset at hand.

ACKNOWLEDGMENTS

This project has received funding from the European Unions Horizon 2020 research and innovation programme under the grant agreement No. 815069 and the Marie Sklodoska-Curie grant agreement No. 665959.

REFERENCES

[1] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018.

[2] H. Song, I. Triguero, and E. Özcan, "A review on the self and dual interactions between machine learning and optimisation," *Progress in Artificial Intelligence*, pp. 1–23, 2019.

[3] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning - Applications to Data Mining*. Cognitive Technologies - Springer Berlin Heidelberg, 2009.

[4] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and Robust Automated Machine Learning," in *Advances in Neural Information Processing Systems*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970.

[5] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *Learning and Intelligent Optimization*, C. A. C. Coello, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523.

[6] G. Piateski and W. Frawley, *Knowledge Discovery in Databases*. Cambridge, MA, USA: MIT Press, 1991.

[7] S. Garcia, J. Luengo, and F. Herrera, *Data preprocessing in data mining*. *Springer*. Cham, Switzerland, 2015.

[8] I. Triguero, D. García-Gil, J. Maillo, J. Luengo, S. García, and F. Herrera, "Transforming big data into smart data: An insight on the use of the k-nearest neighbors algorithm to obtain quality data," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 2, p. e1289, 2019.

[9] G. Luo, "A review of automatic selection methods for machine learning algorithms and hyper-parameter values," *Network Modeling Analysis in Health Informatics and Bioinformatics*, vol. 5, no. 1, p. 18, 2016.

[10] Q. Yao, M. Wang, Y. Chen, W. Dai, H. Yi-Qi, L. Yu-Feng, T. Wei-Wei, Y. Qiang, and Y. Yang, "Taking Human out of Learning Applications: A Survey on Automated Machine Learning," *CoRR*, 2019.

[11] M.-A. Zöller and M. F. Huber, "Survey on Automated Machine Learning," *arXiv preprint arXiv:1904.12054*, 2019.

[12] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2013, p. 847855.

[13] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 485–492.

[14] P. Nguyen, M. Hilario, and A. Kalousis, "Using meta-mining to support data mining workflow planning and optimization," *Journal of Artificial Intelligence Research pages = 605644,*, vol. 51, no. 1, 2014.

[15] J. Vanschoren, "Meta-learning," F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Springer, 2018, pp. 39–68.

[16] J. R. Rice, "The algorithm selection problem," ser. Advances in Computers, M. Rubinoff and M. C. Yovits, Eds. Elsevier, 1976, vol. 15, pp. 65 – 118.

[17] H2O.ai, *H2O AutoML*, June 2017, h2O version 3.30.0.1. [Online]. Available: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

[18] J. S. Angarita-Zapata, I. Triguero, and A. D. Masegosa, "A preliminary study on automatic algorithm selection for short-term traffic forecasting," in *Intelligent Distributed Computing XII*, J. Del Ser, E. Osaba, M. N. Bilbao, J. J. Sanchez-Medina, M. Vecchio, and X.-S. Yang, Eds. Cham: Springer International Publishing, 2018, pp. 204–214.

[19] P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren, "An open source automl benchmark," *CoRR*, 2019, work presented at AutoML Workshop at International Conference on Machine Learning 2019.

[20] J. S. Angarita-Zapata, A. D. Masegosa, and I. Triguero, *Evaluating Automated Machine Learning on Supervised Regression Traffic Forecasting Problems*. Cham: Springer International Publishing, 2020, pp. 187–204.

[21] ——, "General-purpose automated machine learning for transportation: A case study of auto-sklearn for traffic forecasting," in *Proceeding of the 18th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Cham: Springer International Publishing, 2020.

[22] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761 – 1776, 2011.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[24] Kerschke, Pascal and Hoos, Holger and Neumann, Frank and Trautmann, Heike, "Automated Algorithm Selection: Survey and Perspectives," *CoRR*, 2018.

[25] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *2015 IEEE International Conference on Data Science and Advanced Analytics*, 2015, pp. 1–10.

[26] G. Katz, E. C. R. Shin, and D. Song, "Explorekit: Automatic feature generation and selection," in *2016 IEEE 16th International Conference on Data Mining*, 2016, pp. 979–984.

[27] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. Turaga, "Learning feature engineering for classification," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 2529–2535.

[28] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*. USA: Curran Associates Inc., 2011, pp. 2546–2554.

[29] M. Claesen, J. Simm, D. Popovic, Y. Moreau, and B. D. Moor, "Easy hyperparameter search using optunity," *CoRR*, 2014.

[30] T. Swearingen, W. Drevo, B. Cyphers, A. Cuesta-Infante, A. Ross, and K. Veeramachaneni, "Atm: A distributed, collaborative, scalable system for automated machine learning," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 151–162.

[31] F. Mohr, M. Wever, and E. Hüllermeier, "ML-Plan: Automated machine learning via hierarchical planning," *Machine Learning*, vol. 107, no. 8, pp. 1495–1515, 2018.

[32] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the Twenty-First International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2004, p. 18.

[33] I. Guyon, I. Chaabane, H. J. Escalante, S. Escalera, D. Jajetic, J. R. Lloyd, N. Maci, B. Ray, L. Romaszko, M. Sebag, A. R. Statnikov, S. Treguer, and E. Viegas, "A brief review of the chalearn automl challenge: Any-time any-dataset learning without human intervention," in *Proceedings of the Workshop on Automatic Machine Learning*. New York, USA: PMLR, 2016, pp. 21–30.

[34] B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, and J. Vanschoren, "Openml benchmarking suites," 2017.

[35] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: Networked science in machine learning," *SIGKDD Explor. Newsl.*, vol. 15, no. 2, p. 4960, Jun. 2014.

[36] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of

experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044 – 2064, 2010.
[37] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," *CoRR*, 2018.

Juan S. Angarita-Zapata received his B.Sc. and M.Sc. degrees in Computer Science from Universidad Industrial de Santander (Bucaramanga, Colombia) in 2014 and 2016, respectively. He is currently a Marie Sklodowska-Curie Fellow and a PhD candidate at the Faculty of Engineering of the University of Deusto in Bilbao, Spain. He has published journal and conference papers related to Automated Machine Learning, Complex and Dynamical Systems, Optimisation, Deep Learning, and Transportation. His research interests fall within the intersection of Automated Machine Learning, Supervised Learning, Optimisation and Intelligent Systems.

Antonio D. Masegosa took his University degree in Computer Engineering in 2005 and his PhD in Computer Sciences in 2010, both from the University of Granada, Spain. From June 2010 to November 2014 he was a post-doc researcher at the Research Center for ICT of the University of Granada. From 2014 he is working as IKERBASQUE Researcher in the Deusto Institute of Technology, in Bilbao, Spain. He has published four books, eighteen JCR papers and more than 20 papers in both international and national conferences. He has participated in several research projects at regional, national and European level. His main research interests are Artificial Intelligence, Intelligent Systems, Soft Computing, Hybrid Metaheuristics, Machine Learning, Deep Learning, Intelligent Transportation Systems, Logistic Networks, Travel Demand Estimation and Traffic Forecasting.

Isaac Triguero received his M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2009 and 2014, respectively. He is currently an Associate Professor of Data Science at the University of Nottingham. His work is mostly concerned with the research of novel methodologies for big data analytics. Dr Triguero has published more than 80 international publications in the fields of Big Data, Machine Learning and Optimisation (H-index=25 and more than 2500 citations on Google Scholar). He is a Section Editor-in-Chief of the Machine Learning and Knowledge Extraction journal, and an associate editor of the Big Data and Cognitive Computing journal, and the IEEE Access journal. He has acted as Program Co-Chair of the IEEE Conference on Smart Data (2016), the IEEE Conference on Big Data Science and Engineering (2017), and the IEEE International Congress on Big Data (2018). Dr Triguero is currently leading a Knowledge Transfer Partnership project funded by Innovative UK and the energy provider E.ON that investigates Smart Metering data.