2018 7th Brazilian Conference on Intelligent Systems

Bandit-Based Automated Machine Learning

Silvia Nunes das Dôres, Duncan Ruiz
Escola Politécnica
PUCRS
Porto Alegre, Brazil
Email: silvia.dores@acad.pucrs.br
Email: duncan.ruiz@pucrs.br

Carlos Soares
FEUP
Universidade do Porto
Porto, Portugal
Email: csoares@fe.up.pt

***Abstract*—Machine Learning (ML) has been successfully applied to a wide range of domains and applications. Since the number of ML applications is growing, there is a need for tools that boost the data scientist's productivity. Automated Machine Learning (AutoML) is the field of ML that aims to address these needs through the development of solutions which enable data science practitioners, experts and non-experts, to efficiently create fine-tuned predictive models with minimum intervention. In this paper, we present the application of the multi-armed bandit optimization algorithm Hyperband to address the AutoML problem of generating customized classification workflows, a combination of preprocessing methods and ML algorithms including hyperparameter optimization. Experimental results comparing the bandit-based approach against Auto ML Bayesian Optimization methods show that this new approach is superior to the state-of-the-art methods in the test evaluation and equivalent to them in a statistical analysis.**

I. Introduction

The process of Knowledge Discovery in Databases involves several steps, such as selecting a subset of data, cleaning and transforming data, extracting features, choosing appropriate data mining techniques for pattern extraction, evaluating and interpreting results, and implementing such results [1]. To select methods and algorithms that performs well for a given task it is usually required deep domain expertise, brute-force and/or laborious hand-tuning [2]. Since the demand for Machine Learning (ML) algorithms grows faster than the supply of machine learning experts, there is an increasing need for methods to automatically select, combine and configure algorithms to the task at hand.

Automated Machine Learning (AutoML) is the area that merges research efforts to automation of Machine Learning, which includes algorithm selection, model selection and hyperparameter optimization [3]. Automatic workflow selection is the area of AutoML that groups these various challenges, i.e., it aims to design and recommend an optimized combination of preprocessing methods and machine learning algorithms, including their hyperparameter configuration, to specific learning tasks without much dependency on user knowledge. Figure 1 shows an overview of AutoML tasks.

Currently, the big challenge in automatic workflow selection area is the potentially large size of the search space, especially when it includes hyperparameter optimization. The number of possible hyperparameter combinations increases exponentially with the number of components in a workflow (methods and algorithms) and in most cases it is not computationally feasible to evaluate all of them [4].

In this paper, we address the problem of automatic workflow selection with the application of the multi-armed bandit based algorithm, Hyperband. Initially proposed by [5], Hyperband had as its main focus only the hyperparameter optimization. In this work, we present an extension of this algorithm that supports the optimization of workflows, which includes preprocessing methods, learning algorithms and their respective hyperparameters. The framework generated from this approach, named Auto-Band (AUTOmated Machine Learning using BANDdit-Based Optimization), includes a wide variety of classification algorithms and preprocessing methods from the popular machine learning library Weka [6].

To demonstrate the efficiency of Auto-Band to automatic workflow selection, we present a comparison with the most prominent Bayesian Optimization (BO) methods, Auto-Weka and Auto-Sklearn. Results show that this new AutoML approach performed better than the baselines when compared to the classification error. On the other hand, statistical tests show that there is an equivalence between the approaches. Based on these results, it is also possible to extract insights for the evolution of the proposed approach, which includes the addition of new operators as well as a metalearning strategy.

The paper is organized as follow. Section 2 presents the definition of Automatic Workflow Selection. Section 3 presents the related work. Section 4 present the Hyperband algorithm and details it application in automatic workflow selection. In Section 5, we show the comparison between Auto-Band and the state-of-the-art BO solutions. A general discussion and the conclusions are provided in Section 6.

II. Background

Automatic workflow selection problem was initially formalized by Thornton *et al.* [7], as Combined Algorithm Selection and Hyperparameter optimization (CASH) problem:

Definition 1: Given $A = \{A^{(1)}, ..., A^{(R)}\}$ as a set of learning algorithms with associated hyperparameter spaces $\Lambda^{(1)}, ..., \Lambda^{(R)}$, $D_{train} = \{(\hat{x}_1, y_1), ..., (\hat{x}_n, y_n)\}$ as a training dataset split into *k*-cross validation folds $\{D^{(1)}_{valid}, ..., D^{(k)}_{valid}\}$ and $\{D^{(1)}_{train}, ..., D^{(k)}_{train}\}$ such that $D^{(i)}_{train} = \frac{D_{train}}{D^{(i)}_{valid}}$ for $i = 1, ..., k$ and $L(A^{(j)}_{\lambda}, D^{(i)}_{train}, D^{(i)}_{valid})$ as the result of an

978-1-5386-8023-0/18/$31.00 ©2018 IEEE
DOI 10.1109/BRACIS.2018.00029

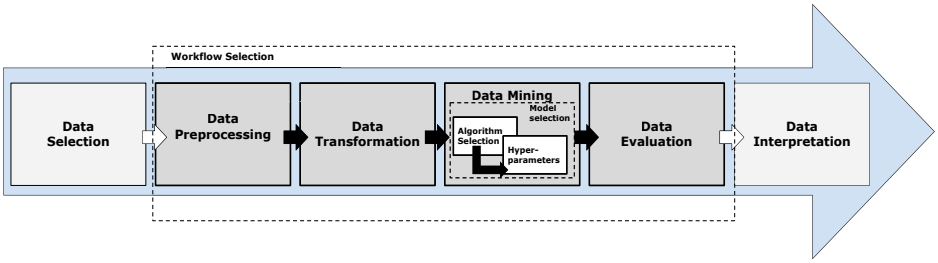

Fig. 1. Automated Machine Learning Tasks

objective function that algorithm $A^{(j)}$ achieves on $D^{(i)}_{valid}$ when trained on $D^{(i)}_{train}$ with hyperparameters λ.

The problem consists of finding the joint algorithm A^* and hyperparameter setting λ_* that optimizes an objective function L (Equation 1 minimizes the k-fold-cross-validation error):

$$A^*_{\lambda_*} = \min_{A^{(j)} \in A, \lambda \in \Lambda^{(j)}} \frac{1}{K} \sum_{i=1}^{K} L(A^{(j)}_{\lambda}, D^{(i)}_{train}, D^{(i)}_{valid}) \quad (1)$$

Address the CASH problem with classical solutions as grid search, i.e. an exhaustive search over all the possible combinations of discretized parameters, can be computationally prohibitive in large search spaces or with big datasets. Instead, a simpler mechanism like random search, where the search space is randomly explored in a limited amount of time, has been shown to be more effective in high-dimensional spaces. Novel approaches gaining popularity in the recent years are based on a Bayesian Optimization methods [7] [8].

III. Related Work

Auto-Weka [7] and Auto-Sklearn [8] are the most prominent AutoML systems to address the automatic workflow selection problem. Both use Bayesian Optimization method, that fits a probabilistic model to capture the relationship between hyperparameter settings and their measured performance; it then uses this model to iteratively select and evaluate the most promising hyperparameter setting, the trade off exploration of new parts of the space vs. exploitation in known good regions [8].

Auto-WEKA [7] was the first AutoML system proposed to address the CASH problem. As optimization strategies it uses two Bayesian Optimization algorithms: Sequential Model-based Algorithm Configuration (SMAC) and Tree-structured Parzen Estimator (TPE). Both algorithms are based in Sequential Model-Based Optimization [9], a stochastic optimization framework that can work explicitly with both categorical and continuous hyperparameters, and that can exploit hierarchical structure stemming from conditional parameters. In the experimental setup they consider feature selection techniques (combining 3 searchers and 8 evaluators methods) and all classification approaches implemented in WEKA, spanning 2 ensemble methods, 10 meta-methods, 27 base classifiers, and hyperparameter settings for each classifier. Kotthoff *et al.* [10] describe the new version of Auto-WEKA, the Auto-Weka 2.0. The improvements with respect to Auto-WEKA 0.5 are: i) support regression algorithms, expanding Auto-WEKA beyond its previous focus on classification; ii) support the optimization of all performance metrics WEKA supports; iii) natively support parallel runs (on a single machine) to find good configurations faster and save the N best configurations of each run instead of just the single best.

Feurer *et al.* [8] introduce a AutoML system based on Scikit-learn, Auto-Sklearn. It improves on existing AutoML methods by automatically taking into account past performance on similar datasets, and by constructing ensembles from the models evaluated during the optimization. First, it searches across datasets to identify instantiations of machine learning frameworks that perform well on a new dataset and warmly start Bayesian Optimization with them. Second, it automatically constructs ensembles of the models considered by Bayesian Optimization. The authors design a parameterized machine learning framework from classifiers and preprocessors implemented in the machine learning framework Scikit-learn [11]. They also perform an empirical analysis using a diverse collection of datasets to demonstrate the performance of Auto-Sklearn system. Auto-Sklearn uses the same Bayesian optimizer as Auto-WEKA, but comprises a smaller space of models and hyperparameters, since Scikit-learn does not implement as many different machine learning techniques as WEKA.

IV. Hyperband to Automatic Workflow Selection

The optimization of the hyperparameters of the operators (methods and algorithms) is the biggest challenge in the workflow selection area. Since the hyperparameters values can effectively affect the final workflow performance, the problem is not only dealing with the size of the search space but also to manage the time needed to evaluate every possible solution [4].

Existing solutions, based on Bayesian Optimization, offer efficiency in this area by adaptively choosing new workflows configurations to train. These solutions focus in the *configuration selection*, i.e., to identify promising workflows configurations more quickly than standard baselines, like random search and grid search. Although these methods have shown efficiency in comparison to the random search, a big challenge

faced by them is the problem of simultaneously fitting and optimizing a high-dimensional space, with include non-convex functions and noise evaluation. To address this problem, the BO methods apply heuristics to model the objective function or speed up resource subroutines. However, adaptive configuration selection methods are intrinsically sequential and difficult to parallelize [5].

In this sense, Li *et al.* [5] proposed Hyperband, a method that formulates the hyperparameter optimization as a pure-exploration adaptive resource allocation problem. In contrast with existing methods, this new approach focuses in speeding up *configuration evaluation*, i.e., it adaptively allocate more resources to promising configurations while eliminating poor ones. These resources can be different elements such as, time of execution, number of features and number of iterations (for iterative algorithms). In the workflow selection context we assume resources as datasets examples (instances). By exploring this perspective, they report better results than the Bayesian Optimization methods, for hyperparameters optimization problems.

The Hyperband approach builds upon Successive-Halving (SH) algorithm [12], which allocates a budget to a set of uniformly sampled configurations, evaluate the performance of all configurations and prunes the worst half, until a single best configuration remains. Thus, the algorithm allocates exponentially more resources to more promising configurations. As the overall performance of Successive-Halving critically depends on the initial allocated computational resources, Hyperband use a infinitely-many armed bandit approach on the space of number of configurations to be considered in parallel.

input : R, η
initialization: $s_{max} = \lfloor \log_\eta(R) \rfloor, B = (s_{max}+1)R$
for $s \in \{s_{max}, ..., 0\}$ **do**
  $n = \left\lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \right\rceil, r = R\eta^{-s}$
  $T = get\_configurations(n)$
  for $i \in \{0, ..., s$ **do**
    $n_i = \lceil n\eta^{-i} \rceil$
    $r_i = r\eta^i$
    $L = \{performance\_evaluation(t, r_i : t \in T)$
    $T = top\_k(T, L, \lfloor \frac{n_i}{\eta} \rfloor)$
  end
end
return Configuration with better performance evaluation seen so far

Algorithm 1: Hyperband Algorithm, adapted from [5]

The Hyperband algorithm is described in Algorithm 1, where: R is the maximum amount of resource that can be allocated to a single configuration, and η is the parameter that controls the proportion of discarded configurations in each round of SH. These inputs dictated the number of SH executions, s. The function $get\_configurations$ returns a set of n sampled configurations according to a fixed distribution. The function $performance\_evaluation$ takes a configuration t and a resource allocation r and returns the validation loss after training the configuration for the resources. Finally, the function top_k returns the top k best configurations.

Although Hyperband was initially designed to address only the hyperparameter optimization problem, we can extend it to deal with workflow selection, that is the aim of this work. For this, we extend the search space, so that the function $get\_configurations$ initially sample workflows (methods of preprocessing + machine learning algorithm), and, for each sampled workflow, the hyperparameters for each component will be selected. We named the Hyperband to AutoML as Auto-Band.

A. Auto-Band Configuration

To design the Auto-Band system we choose Weka [6], because it is one of the most widely applied ML Library and contains a variety of preprocessing methods and ML algorithms that is superior to other libraries, like Scikit-learn [11]. Currently, we use 33 machine learning algorithms, 9 preprocessing methods from Weka.

Table 1 shows a list of the available classifiers algorithms, which has a total of 115 hyperparameters. These algorithms are heterogeneous to represent the different categories of machine learning algorithms, such as nearest neighbor (KNN, KStar), probabilistic (NaiveBayes, BayesNet, NaiveBayes Multinomial), decision trees (HoeffdingTree, J48, REPTree, RandomTree, M5P, DecisionStump, LMT) , optimization-based (MultilayerPerceptron, SGD, SVM, Logistic, Simple Logistic, Voted Perceptron), rule generator (JRip, OneR, ZeroR, Decision Table, PART) and ensemble (AdaBoost M1, RandomForest, RandomSubset, RandomCommitte, Vote, Stacking, Bagging, LogitBoost).

As preprocessing, we choose Feature Selection (FS) and Feature Extraction (FE) methods, because data with extremely high dimensionality has presented one of the most prominent challenges to existing learning methods [13], and these are the most widely employed techniques for reducing dimensionality [14]. The FS methods aims to select a subset of variables from the input data which can efficiently describe these data. To obtain more generalisable results, we evaluate FS methods with different measures of importance such as correlation, information gain and distance. Unlike FS methods, Feature Extraction methods do not select a portion of the original data, but transform this data to a new robust representation. In addition to the methods, we also optimize the hyperparameters of three types of searches, that define how the features will be combined [15]: Best First, Greedy Stepwise and Ranker. Table 2 presents the complete list of preprocessing methods. In total, 28 hyperparameters were optimized, between evaluators (E) and searchers (S).

V. Comparing Auto-Band with Auto-Sklearn and Auto-Weka

To demonstrate the efficiency of the proposed approach to automatic workflow selection problem, we perform a compar-

TABLE I
CLASSIFIERS ALGORITHMS AVAILABLE IN AUTO-BAND AND THE NUMBER OF HYPERPARAMETERS FOR EACH ONE.

name	#hp	categorical	numerical
AdaBoost M1	4	1	3
Bagging	4	1	3
BayesNet	2	2	0
DecisionStump	1	1	0
DecisionTable	4	4	0
HoeffdingTree	6	0	6
J48	8	6	2
JRip	3	2	1
KNN	4	2	2
KStar	3	2	1
LMT	7	5	2
LogitBoost	5	1	4
Logistic	1	0	1
LWL	3	3	0
Multilayer Perceptron	8	4	4
M5P	4	3	1
M5R	4	3	1
NaiveBayesMultinomial	0	0	0
Naive Bayes	2	2	0
OneR	1	0	1
PART	4	2	2
Random Forest	3	0	3
Random Tree	5	1	4
Random Subset	3	0	3
Random Committe	2	0	2
REPTree	4	1	3
Stacking	2	0	2
SimpleLogistic	3	3	0
Stochastic Gradient Descent	5	2	3
SVM	4	2	2
Vote	2	1	1
Voted Perceptron	3	0	3
ZeroR	1	1	0

TABLE II
PREPROCESSING METHODS AVAILABLE IN AUTO-BAND AND THE NUMBER OF HYPERPARAMETERS FOR EACH ONE.

name	Type	#hp	categorical	numerical
CFS Subset Eval	E	3	2	1
Classifier Attribute Eval	E	1	1	0
Gain Ratio Eval	E	1	1	0
Info Gain Eval	E	2	2	0
OneR Attribute Eval	E	3	1	2
PCA	E	3	1	2
RELIEF Eval	E	4	1	3
Symmetrical Uncert	E	1	1	0
Wrapper Sub Eval	E	4	2	2
Best First	S	2	0	2
Greedy Stepwise	S	3	3	0
Ranker	S	1	0	1

ison with two state-of-the-art Bayesian Optimization methods: Auto-Weka and Auto-Sklearn.

We based our experiments on the experimental setup introduced in Auto-Weka paper [7] and replicate in Auto-Sklearn paper [8]. Table 3 shows the detailed description of the 21 benchmark datasets used: 15 datasets from UCI repository [16]; "Convex", MNISTBasic and MNISTRotated from [17]; "KDD09-Appetency" from [18]; Cifar10 and Cifar-10-Small from [19]. For these datasets, we use the original train/test split proposed in [7]: the train data were split in 80% training and 20% validation, and the test data was only used to evaluate

TABLE III
DATASETS USED TO COMPARISON, FROM [7]. NUMBER OF CONTINUOUS ATTRIBUTES, NUMBER OF NUMERIC ATTRIBUTES, NUMBER OF TRAINING SAMPLES AND NUMBER OF TEST SAMPLES

name	#cont.	#num.	#classes	#training	#test
Abalone	7	1	26	2924	1253
Amazon	10000	0	50	1050	450
Car	0	6	4	1210	518
Cifar10	3072	0	10	50000	10000
Cifar-10-Small	3072	0	10	10000	10000
Convex	784	0	2	8000	50000
Dexter	20000	0	2	420	180
Dorothea	100000	0	2	805	345
GermanCredit	7	13	2	700	300
Gisette	5000	0	2	4900	2100
KDD09-Appetency	192	38	2	35000	15000
KR-vs-KP	0	36	2	2238	958
Madelon	500	0	2	1820	780
MNISTBasic	784	0	10	12000	500000
MNISTRotated	784	0	10	12000	500000
Secom	590	0	2	1097	470
Semeion	256	0	10	1116	477
Shuttle	9	0	7	43500	14500
Waveform	40	0	3	3500	1500
Wine Quality White	11	0	7	3429	1469
Yeast	8	0	10	1039	445

the workflows found in the optimization phase.

We impose a 5GB memory limit, 6-minute timeout for each workflow configuration and a one-hour time window to evaluate each strategy on each dataset. Ten trials on each search were performed per dataset. We run Auto-band with $\eta = 3$, i.e. each run of Successive Halving discard $2/3$ of the arms and keeps the remains $1/3$. R is set to equal the training set and $n_{max} = max\{12, R/700\}$. All experiments use Auto-Weka 2.6, Weka 3.9 and Auto-Sklearn 0.3.0.

The experiments were ran on two Intel Core i7-4770 eight-core processors with 3.40GHz and 12 GB of RAM.

A. Experimental Results

Table 4 shows the test set classification error of Auto-Band (AB), Auto-Sklearn (AS) and Auto-Weka (AW). We show the average error across 10 runs of each approach.

All approaches had "memory out" problem for the "KDD09-Appetency" and "Dorothea" datasets, considering the proposed RAM configuration of 5GB, so the results in these datasets are not considered. Auto-Sklearn also had "memory out" problem for the "Cifar10". Our results are slightly different from those presented in [7] and [8], we believe that it is because of the different versions used of Auto-Weka and Auto-Sklearn, since for these experiments we used the current versions of both frameworks.

The results show that Auto-Band performs better than Auto-Sklearn and Auto-Weka in the average classification error, for the 19 datasets computed. Compared with Auto-Sklearn, Auto-Band won in 12/19 datasets, lost against 9 and tied for 1/19. In relation to Auto-Weka, Auto-Band won in 11/19 cases, and lost in 8 cases. In a overview, Auto-Band performs better in 8/19 cases, Auto-Weka in 8/19 and Auto-Sklearn in 4/19, considering the ties. These results show that, on average, Auto-

TABLE IV
TEST SET CLASSIFICATION ERROR OF AUTO-BAND (AB), AUTO-SKLEARN (AS) AND AUTO-WEKA (AW).

	Abalone	Amazon	Car	Cifar10	Cifar10 Small	Convex	Dexter	German Credit	Gisette	KRvsKP
AB	0.768	0.640	0.100	**0.679**	0.765	0.320	0.112	**0.284**	**0.033**	**0.001**
AS	0.963	0.897	0.025	-	0.831	**0.288**	**0.053**	0.295	0.035	0.008
AW	**0.753**	**0.597**	**0.005**	0.681	**0.708**	0.458	0.269	0.290	0.045	0.044
	Madelon	MNIST Basic	MNIST Rot	Secom	Semeion	Shuttle	Waveform	Wine Quality	Yeast	
AB	0.268	0.137	**0.672**	0.079	**0.084**	**0.000**	0.175	0.456	**0.407**	
AS	**0.128**	0.393	0.853	0.085	0.103	**0.000**	0.145	0.449	0.764	
AW	0.269	**0.107**	0.756	**0.001**	0.116	0.078	**0.139**	**0.353**	0.410	

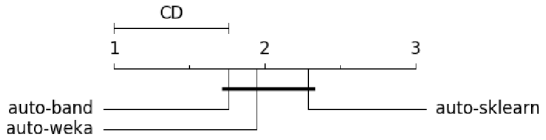

Fig. 2. Critical Difference diagram (with $\alpha = 0.05$) of the comparison between Auto-Band with Auto-Sklearn and Auto-Weka

Band performance is quite close to Auto-weka, although our results are slightly higher.

This is emphasized in a statistical evaluation, where we can see that, although there is a difference between the Auto-Weka, Auto-Sklearn and Auto-Band, this difference is not statistically significant. These tests were carried out using the methodology proposed by Demšar [20]: Friedman rank test with Nemenyi test for *post-hoc* multiple comparisons, and presented in Figure 2. AutoML approaches are sorted by their average ranking (lower is better), and those connected by a horizontal line are statistically equivalent.

In addition, we identified the best workflows generated for each dataset, which are presented in Table 5. To improve the visualization of the information, we did not include the values of the hyperparameters in the table, only the preprocessing methods and the learners.

The workflows generated by Auto-Sklearn contains data preprocessing methods to replace categorical features, imputation of missing values, rescaling features and balancing class. Since Auto-Band only contains feature selection and feature extraction methods, one possibility for improvement would be the inclusion of these methods, which would generate complex chains of operators.

On the other hand, the most current version available of Auto-Weka (2.6) contains only the CFS Sub Eval method as a pre-processing method. Therefore, all workflows are composed of it or only by the learning algorithm.

We can see that Auto-Band found the best workflow for most cases (8/19). In the cases of the datasets Amazon, MNISTBasic and Secom, Auto-Band found the workflow with better performance, but in the average results was exceeded by Auto-Weka. We also note that only one workflow generated by Auto-Band has a preprocessing method.

The original Hyperband starts uniformly sampling the configurations to be evaluated, because there is no previous information about which ones will work well to the current problem. Although it follows a defined distribution (for example, uniform), this sampling of configurations is random, and since the search space grows (in this case the possibilities of combination of operators is of 1,150) many non-promising configurations are tested, which behavior varies largely as the algorithm runs several times. Thus, if there was the possibility of using supplementary information (such as meta-data from previous experiences), Hyperband's choices could be directed to the most promising configurations, making performance more stable.

VI. DISCUSSION AND CONCLUSIONS

In this paper, we present and evaluate the application of Hyperband Algorithm to address the AutoML workflow selection problem, an approach that we named Auto-Band. When we consider that the initial search space of Hyperband only included hyperparameters for a single algorithm, we can see that our proposed work configures a more complex and robust application of the Hyperband algorithm.

Using this multi-armed bandit algorithm with adaptive resource to automatic workflow selection, we obtained equivalent results in comparison with state-of-the-art Bayesian Optimization methods. These results are particularly interesting when we consider that the baseline methods have a superior set of preprocessing methods and ML algorithms, which allows the generation of more robust workflows. As future improvements in Auto-Band, we intend to extend the set of operators to aggregate all preprocessing methods and supervised machine learning algorithms available in Weka, which will allow us to build complex chains of operators and to fulfill the regression task.

An additional improvement to Auto-Band effectiveness is to include a metalearning step. This metalearning approach will use meta-data of previous experiences in similar datasets to suggest which methods can achieve better performance in the current data. With this, we intend to improve the quality of the generated solutions.

REFERENCES

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.

TABLE V
BEST WORKFLOW OPTIMIZED FOR EACH DATASET.

Data	Workflow	Error	Framework
Abalone	prep: None learner: MultilayerPerceptron	0.729	AW
Amazon	prep: None learner: SVM	0.564	AB
Car	prep: CFS Sub Eval learner: SVM	0.000	AW
Cifar10	prep: None learner: SVM	0.600	AB
Cifar-10-Small	prep: Imputation Most Frequent, Feature Agglomeration Linkage learner: K-NN	0.687	AS
Convex	prep: Rescaling Standardize, Fast ICA learner: Extra Trees (Extremely Randomided Trees)	0.213	AS
Dexter	prep: One Hot Encoding, Imputation Mean learner: LibSVM	0.044	AS
GermanCredit	prep: Balancing Weighting, Imputation Most Frequent, Feature Agglomeration learner: Extra Trees	0.273	AS
Gisette	prep: None learner: SVM	0.022	AB
KR-vs-KP	prep: CFS Sub Eval learner: AdaBoostM1	0.003	AW
Madelon	prep: One Hot Encoding, Extra Trees Preprocessors for Classification learner: K-NN	0.112	AS
MNISTBasic	prep: None learner: K-NN	0.055	AB
MNISTRotated	prep: None learner: RandomSubset	0.627	AB
Secom	prep: Wrapper Sub Eval learner: Simple Logistic	0.078	AB
Semeion	prep: None learner: Random Committe	0.079	AB
Shuttle	prep: None learner: Random Forest	0.000	AB
Waveform	prep: CFS Sub Eval learner: Multilayer Perceptron	0.137	AW
Wine Quality White	prep: CFS Sub Eval learner: Random Forest	0.333	AW
Yeast	prep: CFS Sub Eval learner: Bagging	0.373	AW

[2] A. G. de Sá, W. J. G. Pinto, L. O. V. Oliveira, and G. L. Pappa, "Recipe: a grammar-based framework for automatically evolving classification pipelines," in *European Conference on Genetic Programming*. Springer, 2017, pp. 246–261.

[3] I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. K. Ho, N. Macia, B. Ray, M. Saeed, A. Statnikov *et al.*, "Design of the 2015 chalearn automl challenge," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.

[4] M. M. Salvador, M. Budka, and B. Gabrys, "Automatic composition and optimisation of multicomponent predictive systems," *arXiv preprint arXiv:1612.08789*, 2016.

[5] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *arXiv preprint arXiv:1603.06560*, 2016.

[6] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[7] 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 ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 847–855.

[8] 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*, 2015, pp. 2962–2970.

[9] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration." *LION*, vol. 5, pp. 507–523, 2011.

[10] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka," *Journal of Machine Learning Research*, vol. 17, pp. 1–5, 2016.

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

[12] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Artificial Intelligence and Statistics*, 2016, pp. 240–248.

[13] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data Classification: Algorithms and Applications*, p. 37, 2014.

[14] Q. Yang and X. Wu, "10 challenging problems in data mining research," *International Journal of Information Technology & Decision Making*, vol. 5, no. 04, pp. 597–604, 2006.

[15] A. R. S. Parmezan, H. D. Lee, and F. C. Wu, "Metalearning for choosing feature selection algorithms in data mining: Proposal of a new framework," *Expert Systems with Applications*, vol. 75, pp. 1–24, 2017.

[16] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[17] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[18] "Kdd cup 2009: Customer relationship prediction," 2009.

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[20] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.