The impact of Auto-Sklearn's Learning Settings

Meta-learning, Ensembling, Time Budget, and Search Space Size

Hassan Eldeeb*
hassan.eldeeb@ut.ee
Data Systems Group
University of Tartu, Estonia

Oleh Matsuk
Data Systems Group
University of Tartu, Estonia

Mohamed Maher
Data Systems Group
University of Tartu, Estonia

Abdelrhman Eldallal
Data Systems Group
University of Tartu, Estonia

Sherif Sakr
Data Systems Group
University of Tartu, Estonia

ABSTRACT

With the booming demand for machine learning (ML) applications, it is recognized that the number of knowledgeable data scientists cannot scale with the growing data volumes and application needs in our digital world. Therefore, several automated machine learning (AutoML) frameworks have been developed to fill the gap of human expertise by automating most of the process of building a ML pipeline. In this paper, we present a micro-level analysis of the AutoML process by empirically evaluating and analyzing the impact of several learning settings and parameters, i.e., *meta-learning*, *ensembling*, *time budget* and *size of search space* on the performance. Particularly, we focus on AutoSklearn, the state-of-the-art AutoML framework. Our study reveals that no single configuration of these design decisions achieves the best performance across all conditions and datasets. However, some design parameters have a statistically consistent improvement over the performance, such as using ensemble models. Some others are conditionally effective, e.g., meta-learning adds a statistically significant improvement, only with a small time budget.

1 INTRODUCTION

Due to the increasing success of machine learning techniques in several application domains, they attract lots of attention from the research and business communities. Hence, a wide range of fields is witnessing many breakthroughs achieved by machine and deep learning techniques [18, 29]. Furthermore, machine learning has significant achievements compared to human-level performance. For example, `AlphaGO` [20] defeated the GO game's champion, and deep learning models excelled in image recognition and surpassed human performance years ago [23].

Nevertheless, the machine learning modeling process is a highly iterative, exploratory, and time-consuming process. Therefore, recently, several frameworks (e.g., `AutoWeka` [22], `AutoSklearn` [7], `SmartML` [17]) are proposed to support automating the Combined Algorithm Selection and Hyper-parameter tuning (CASH) problem [13, 19, 28]. The performance of the automatically generated pipelines, by these `AutoML` frameworks, is perfect for some tasks such that data scientists cannot develop pipelines to beat it; not even `AutoML` designers, as seen in the ChaLearn `AutoML` Challenge 2015/2016 [12].

These AutoML frameworks follow different learning settings, i.e., parameters or options that `AutoML` users have to preset while submitting the input dataset. For example, `AutoSklearn` [7] and `SmartML` [17] adopt a meta-learning-based mechanism to improve the automated search process's performance. That is, they start with the most promising models that have performed well with similar datasets. `ATM` [21] limits the default search space into only three classifiers, namely, *Decision Tree, K-Nearest Neighbours, and Logistic Regression*. `AutoSklearn` offers an ensembling mechanism as a post-hoc optimization instead of reporting only the best-performing model. Additionally, most AutoML frameworks run within a user-determined time budget. Although the user has the option to use different flavors of `AutoML` tools by manipulating these learning settings, it is hard to decide which of them should be used for the input dataset. So, these learning settings are just *hyper-parameters* for `AutoML` tools.

Understanding the impact of these learning settings on real-world datasets is vital, especially when the authors of the `AutoML` frameworks evaluate their contributions using relatively small datasets [22]. Besides, the authors may knowingly or unknowingly select datasets on which their frameworks perform well.

In this study, we present a thorough analysis of the significance of various hyper-parameters (learning settings) of the AutoML process, including *meta-learning, ensembling, length of time budget* and *size of search space*. Since `AutoSklearn` supports different settings for all of these hyper-parameters, we nominated it to be the backbone of this study.

So, the contribution of this paper can be summarized as follows:

- We benchmark 100 datasets on different learning settings (hyper-parameters) of `AutoSklearn`.
- The impact of each hyper-parameter of `AutoSklearn` has been examined using different configurations.
- For each positive/negative impact of hyper-parameter configuration, we validate its consistency using `Wilcoxon` statistical test [27].
- Eventually, we provide a simple guideline for which of these hyper-parameters is expected to improve the performance score based on the input datasets' characteristics.

This analysis is ongoing and extendable. So, we will update it with new datasets and additional frameworks. For ensuring reproducibility, we have released all artifacts (e.g., datasets, source code, log results).[1]

The remainder of this paper is organized as follows. We discuss the related work in Section 2. Section 3 describes our experiment design and defines the target learning settings. Section 4 presents the impact of meta-learning (Section 4.1), ensembling

*Corresponding Author.

[1] https://datasystemsgrouput.github.io/AutoMLDesignDecisions/

(Section 4.2), length of the time budget (Section 4.3), and size of the search space (Section 4.4). The datasets that show significant performance differences towards/against any of these design parameters are further analyzed in Section 4.5. Finally, we conclude the paper in Section 5.

2 RELATED WORK

Recently, several studies have surveyed and compared the performance of various AutoML frameworks [11, 13, 19, 24, 28]. In general, these studies show no clear winner as there are always some trade-offs that need to be considered and optimized according to the context of the problems and the user's goals. For example, Gijsbers et al. [11] have conducted an experimental study to compare the performance of 4 AutoML systems, namely, `AutoWeka`, `AutoSklearn`, `TPOT` and `H2O` using 39 datasets and time budgets of 1 and 4 hours. The study results observed that some AutoML tools perform significantly better or worse on several datasets than others. The authors could not draw clear conclusions about which data properties could explain this behavior.

Truong et al. [24] have conducted a study using 300 datasets to compare the performance of 7 AutoML frameworks, namely, `H2O`, [15], `AutoSklearn`, `Ludwig`[2], `Darwin`[3], `TPOT` and `Auto-ml` using dynamic time budgets. The results of this study showed that no framework managed to outperform all others on a plurality of tasks. Across the various evaluations and benchmarks, `H2O`, `Auto-keras` and `AutoSklearn` performed better than the rest of the tools.

Zöller and Huber [28] have performed a comparison for 8 CASH optimization algorithms, namely, *Grid Search*, *Random Search*, *ROBO* (RObust Bayesian Optimization) [16], *BTB* (Bayesian Tuning and Bandits)[4], *hyperopt* [1], *SMAC*, *BOHB* [6] and *Optunity*[5]. The comparison results showed that all CASH algorithms' performance, except the grid search, perform similarly on average. The authors also noted that a simple search algorithm such as random search did not perform worse than the other algorithms. Besides, the authors compared the performance of 6 AutoML frameworks, namely, `TPOT`, `hpsklearn`[6], `AutoSklearn`, `ATM`, `H2O` in addition to `Random Search`. The results of the frameworks' comparison showed that on average, all AutoML frameworks have similar performance. However, for a single dataset, the performance differs on average by 6% accuracy.

To the best of our knowledge, this study is the first that focuses on analyzing the learning settings and parameters of the AutoML process. The previous studies mainly focus on comparing whole frameworks' performance or comparing different optimization techniques' performance. Thus, a single configuration of the learning settings is used with all the tested datasets. Mostly, it is the default settings to have a fair comparison among the benchmarked frameworks [11, 13, 19, 24, 28]. In contrast, this study focuses on the impact of the learning settings and different configuration of the `AutoML` framework over the accuracy performance. Understanding this relationship can help the domain expert use `AutoML` frameworks and select the learning setting configuration that works well with his dataset.

[2]https://github.com/uber/ludwig
[3]https://www.sparkcognition.com/product/darwin/
[4]https://github.com/HDI-Project/BTB
[5]https://github.com/claesenm/optunity
[6]https://github.com/hyperopt/hyperopt-sklearn/tree/master/hpsklearn

3 EXPERIMENT DESIGN

In this paper, we followed the best practices on how to construct and run good machine learning benchmarks and general-purpose algorithm configuration libraries [2]

3.1 AutoML framework: AutoSKLearn

`AutoSklearn` [7], the winner of two ChaLearn AutoML challenges, is implemented on top of `Scikit-Learn`, a popular Python machine learning package [9]. It uses Sequential Model-based Algorithm Configuration (SMAC) as a Bayesian optimization technique [14]. Beside adopting meta-learning and ensembling design decisions, time budget and search space are also configurable in `AutoSklearn`. The framework uses meta-learning for initializing the search process as a warm start. It also utilizes the ensembling learning setting to improve the performance of output models. Moreover, one of the main advantages of `AutoSklearn` is that it comes with different execution options. In particular, its basic *vanilla* version (`AutoSklearn-v`) applies only the SMAC optimization techniques for the AutoML optimization process. However, `AutoSklearn` also allows the end-users to enable/disable the different optimization options including the usage of meta-learning (`AutoSklearn-m`), ensembling (`AutoSklearn-e`) in addition to the full version (`AutoSklearn`) where all options are enabled.

3.2 Datasets

We used 100 datasets that are collected from `OpenML` repository [26]. `OpenML` datasets are already preprocessed into numerical features. Therefore, they match the input criteria of `AutoSklearn`. The datasets include binary (50%) and multi-class (50%) classification tasks. The sizes of these datasets varies between 5KB and 643MB. The datasets used in this study cover a wide spectrum of various meta-features (e.g., *r_numerical_features, class_entropy, max_prob, mean_prob, std_dev, dataset_ratio, symbols_sum, symbols_std_dev, skew_std_dev, kurtosis_min, kurtosis_max, kurtosis_mean and kurtosis_std_dev*) [5]. Each dataset is partitioned into training and validation split (80%) and a test split (20%) which is used to evaluate the output pipeline.

3.3 Hardware Choice and Resources

The experiments are conducted on Google Cloud machines. Each machine is configured with 2 vCPUs, 7.5 GB RAM, and ubuntu-minimal-1804-bionic. Each experiment is run four times with different time budgets: 10, 30, 60, and 240 minutes.

3.4 Learning Settings Definitions

Learning settings are the parameters or options that `AutoML` users have to preset while submitting the input dataset. This paper focuses on four of them: *meta-learning*, *ensembling*, *time budget*, and *search space size*. In the following, we define each of them in the context of `AutoML` and briefly describe their mechanism used in `AutoSklearn`.

Meta-learning [25] is the process of learning from previous experience gained while applying various learning algorithms on different types of data. In the context of AutoML, the main advantage of meta-learning techniques is that it allows hand-engineered algorithms to be replaced with automated methods designed in a data-driven way. Thus, it is used partially to simulate the machine learning expert's role for non-technical users and domain experts. `AutoSklearn` applies a meta-learning mechanism based on a

knowledge base storing the meta-features of datasets and the best-performing pipelines on these datasets. Thirty-eight statistical and information-theoretic meta-features are used. In the offline phase, the meta-features and the empirically best-performing pipelines are stored for each dataset in their repository (140 datasets from OpenML repository) [7]. For any new dataset in the online phase, the framework extracts its meta-features and searches for the most similar datasets to return the top k best-performing pipelines on these similar datasets. These k pipelines are used as a warm start for the Bayesian optimization algorithm used in the framework.

In principle, the main goal of any meta-learning mechanism is to improve the search process by enabling the optimization technique to start from the most promising pipelines instead of starting from random pipelines. If the suggested pipelines' performance is bad, the Bayesian optimization can recover from these pipelines in the next iterations.

Ensembling is the process of combining multiple ML base models trained on the same task to produce a better predictive model. These base models can be combined using several techniques, including simple/weighted voting (averaging), bagging, boosting, and other techniques [4]. In principle, the main advantage of using ensembling techniques is that it allows the base models to collaborate in generating more generalized predictions than using predictions from an individual base model.

`AutoSklearn` stores the generated models instead of just keeping the best-performing one. These models are used in a post-processing phase to construct an ensemble. `AutoSklearn` uses the ensemble selection methodology introduced by Caruana et al. [3]. It is a greedy technique that starts with an empty ensemble and attractively adds base models to the ensemble to maximize the validation performance.

The time budget represents the available time to examine the search space for identifying a pipeline that maximizes the performance metric. Generally, it is hypothesized that the more time allocated to the search process, the higher the achievable performance [7]. On the other hand, there is another trade-off that should be considered. The longer the budget we allocate, the more computing resources we will consume for the search process and the higher potential that the model overfits the validation set.

The search space of any AutoML process is significantly huge [7]. For example, `AutoSklearn` [7] is designed with over 15 classifiers from the `scikit-learn` library. Assuming that each classifier has only 2 hyper-parameters and each of them has 100 discrete values, the search space contains $15 * 100^2$ different configurations. However, in practice, the real numbers are much bigger.

4 RESULTS AND DISCUSSION

The set of questions aimed at assessing the impact of each learning settings are as follows. Does the parameter improve/decline the performance accuracy? Is the difference statistically significant? And finally, when is it recommended to enable each parameter? We answer these questions for each learning setting.

4.1 The Impact of Meta-Learning

To assess the impact of the meta-learning mechanism, we experimented with comparing the performance of `AutoSklearn-m` and `AutoSklearn-v`. Figure 1 shows the impact of meta-learning over different time budgets. From this figure, the meta-learning mechanism does not always lead to better performance. On average, `AutoSklearn-v` and `AutoSklearn-m` provide a comparable performance, for the four time budgets, as shown in Table 1. In particular, both versions have similar performance in 64, 55, 65, and 69 datasets for the 10, 30, 60, and 240 minutes, respectively. Table 1 summarizes the results for each time budget.

We used `Wilcoxon` statistical test [10] to assess the significance of the performance difference between `AutoSklearn-v` and `AutoSklearn-m`. The results of Table 2 show that the meta-learning mechanism makes a statistically significant gain with 95% confidence (p value < 0.05) only with the 10 minutes time budget. For the 30-minute time budget, the level of confidence decreases to 93.5%. In contrast, for the time budgets of 60 and 240 minutes, there is no statistically significant difference in using the meta-learning mechanism to initialize the search process. Hence, this implies that the longer the time budget, the lower the impact of the meta-learning mechanism. In general, the longer the time budget, the more time available for the AutoML search process to explore more configurations in the search space, and the higher probability of getting a better result. Hence, the impact of the initial configurations is also lower.

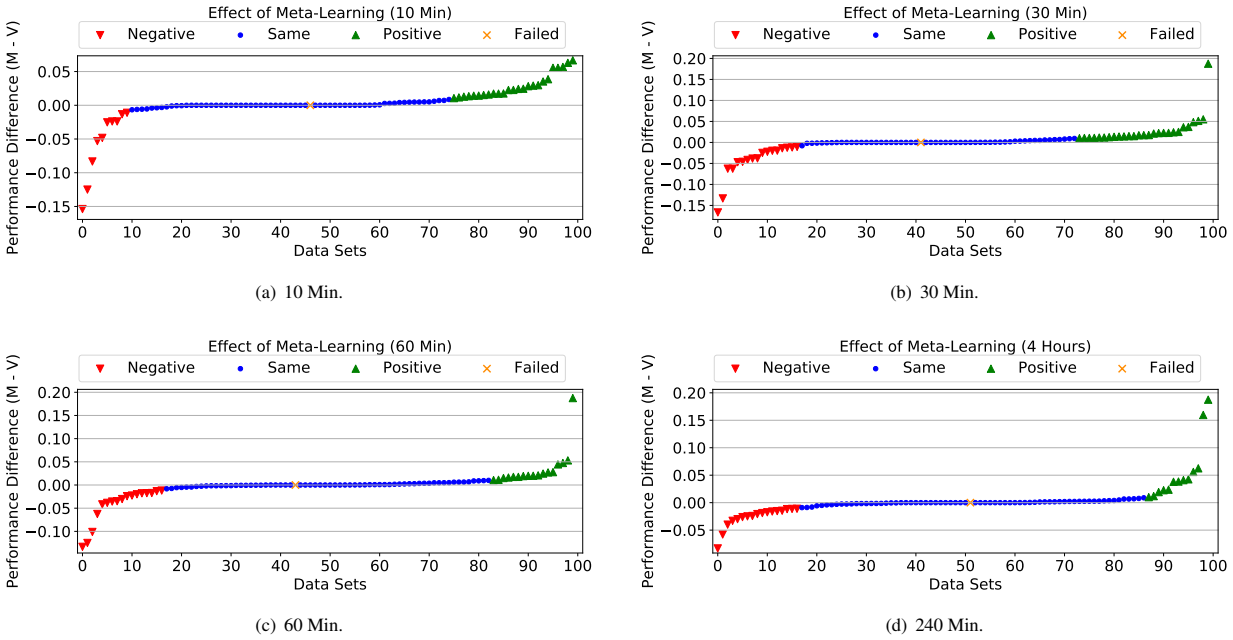
We speculated whether the improvement/deterioration effect of meta-learning is constant for the same datasets across the four time budgets. As shown in Figure 2(a), we found that 25 datasets have better accuracy when using `AutoSklearn-m` than `AutoSklearn-v` in the 10-minute time budget. Out of those 25, the improvement holds in only 13 datasets in the 30-minute time budget. Similarly, among the 13 datasets, only 5 continued to retain this behavior during the 60-minute. Finally, only one dataset is common among the four time budgets.

From Figure 2, the datasets with improved performance scores are not the same in every time budget. By analyzing the meta-features of these datasets, we could not link them over different time budgets. This observation is also confirmed in Figure 2(b), where we could not draw a clear pattern out of these datasets.

4.2 The Impact of Ensembling

To assess the impact of the ensembling, we have experimented with comparing the performance of `AutoSklearn-v` and `AutoSklearn-e`. Figure 3 shows the performance differences between the two versions over the 100 datasets. On average, `AutoSklearn-e` increased the accuracy performance by 0.5%, 0.7%, 1%, and 1.4% over the four time budgets, successively. In particular, the two modes have similar performance in 65, 62, 66, and 63 datasets for 10, 30, 60, and 240 minute time budgets. Table 3 summarizes the results for each time budget.

Table 4 shows that the outcomes of `Wilcoxon` test, which is conducted to assess the statistical significance of the accuracy performance between `AutoSklearn-v` and `AutoSklearn-e`. The table shows that the ensembling techniques enhance the performance with a statistically significant gain with more than 95% confidence (p value < 0.05) on the four time budgets. The level of confidence is almost 99% over all the time budgets combined. Generally, the ensemble model extremely boosts accuracy compared to the individual base models as long as these base models' errors are independent of each other [4]. Although the base classifiers' errors are not completely independent, the ensemble model still enhances the accuracy in a statistically significant manner. It means that the accuracy improvement by the ensemble model, generated by `AutoSklearn-e`, is not a random effect and is expected to be repeated on the new datasets.



(a) 10 Min.

(b) 30 Min.

(c) 60 Min.

(d) 240 Min.

Figure 1: The impact of meta-learning over all time budgets. Upward triangles represent better performance with `Auto-Sklearn-m`, downward triangles represent better performance using `AutoSklearn-v`, and circles mean that the absolute difference is $< 1\%$.

Table 1: Comparison between the performance of `AutoSklearn-v` and `AutoSklearn-m` in terms of accuracy over different time budgets.

Time Budget	Framework	Accuracy		Gain (Accuracy)			#datasets with
		Mean	SD	Min	Mean	Max	gain > 1%
10	AutoSklearn-m	0.870	0.144	1.1%	2.9%	6.7%	25
	AutoSklearn-v	0.868	0.145	1.1%	5.6%	15.6%	10
30	AutoSklearn-m	0.873	0.143	1.1%	2.8%	18.8%	27
	AutoSklearn-v	0.873	0.142	1.1%	4.5%	16.7%	17
60	AutoSklearn-m	0.873	0.141	1.1%	3.4%	18.8%	17
	AutoSklearn-v	0.874	0.137	1.1%	4.4%	13.3%	17
240	AutoSklearn-m	0.877	0.136	1.1%	5.5.%	18.8%	13
	AutoSklearn-v	0.872	0.149	1.1%	2.7%	8.3%	17

Table 2: The results of Wilcoxon test for assessing the statistical significance of the performance difference using `Auto-Sklearn-m` over `AutoSklearn-v`

Mode 1	Mode 2	Time Budget	*P* value
`AutoSklearn-m`	`AutoSklearn-v`	10	**0.004**
		30	0.065
		60	0.434
		240	0.305

The datasets with an enhanced/declined performance using `AutoSklearn-e` within the four time budgets are studied. Figure 4(a) shows the *overlap* among the datasets with better performance using the ensembling mechanism. Figure 4(b) shows the *overlap* among the datasets with better performance using `AutoSklearn-v`. Our analysis shows no strong correlation between the meta-features of the datasets and the probability of having a better/lower performance impact using the ensembling mechanism.

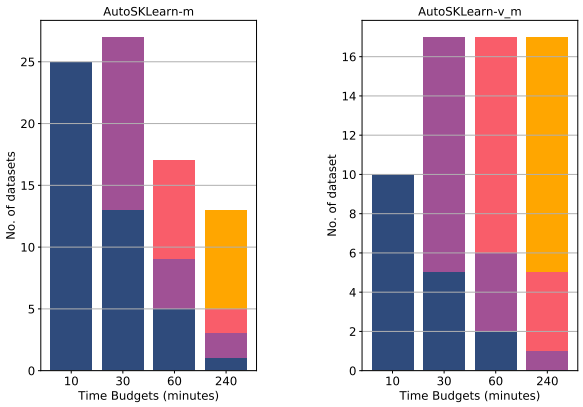
4.3 The Impact of Time Budget

This experiment compares the accuracy gain of all combinations of time budget increases (10/30 Min, 10/60 Min, 10/240 Min, 30/60 Min, 30/240 Min, 60/240 Min). For space limitations, we could not include all comparison figures. However, they are available in the project repository.

On average, the accuracy values on each of the four time budgets are comparable. For example, the base/increased time budgets, i.e., 10/30 Min (Figure 5(a)), 30/60 Min (Figure 5(b)), 60/240 Min (Figure 5(c)) and 10/240 Min (Figure 5(d)), have similar performance in 65, 57, 66, and 60 datasets, respectively. The accuracy of the base time budget was lower than the performance of the increased time budget for 9, 9, 13, and 11 datasets on the four figures, respectively. On the other hand, they have better performance for 17, 21, 22, and 26 datasets. The majority of the datasets

Table 3: Comparison between the performance of `AutoSklearn-v` and `AutoSklearn-e` in terms of accuracy over different time budgets.

Time Budget	Framework	Accuracy		Gain (Accuracy)			#datasets with
		Mean	SD	Min	Mean	Max	gain > 1%
10	AutoSklearn-e	0.873	0.139	1.1%	4.1%	16.4%	22
	AutoSklearn-v	0.868	0.145	1.1%	3.4%	8.3%	12
30	AutoSklearn-e	0.880	0.136	1.1%	3.4	13.5%	27
	AutoSklearn-v	0.873	0.142	1.1%	3.1%	11.1%	10
60	AutoSklearn-e	0.884	0.132	1.1%	4.9%	12.5%	24
	AutoSklearn-v	0.874	0.137	1.1%	3.1%	6.4%	9
240	AutoSklearn-e	0.886	0.130	1.1%	7.1%	52.7%	14
	AutoSklearn-v	0.872	0.149	1.1%	2.9%	8.3%	11



(a) `AutoSklearn-m` **(b) `AutoSklearn-v`**

Figure 2: Overlap among datasets having better performance using `AutoSklearn-m`(a), and `AutoSklearn-v`(b) through each time budget. The new color in each bar represents the number of new datasets with higher performance at this time budget, while the same color represents the same datasets from previous time budgets.

Table 4: The results of Wilcoxon test for assessing the statistical significance of the performance difference using `AutoSklearn-e` over `AutoSklearn-v`

Framework 1	Framework 2	TB	*P* value
`AutoSklearn-e`	`AutoSklearn-v`	10	**0.011**
		30	**0.000**
		60	**0.000**
		240	**0.008**

Table 5: The results of Wilcoxon test for assessing the statistical significance of the performance gain for increasing the time budget

Framework	TB 1	TB 2	Avg. Acc. Diff	*P* value
AutoSklearn-v	30	10	0.005	0.226
	60	10	0.007	**0.004**
	60	30	0.002	0.141
	240	10	0.007	**0.000**
	240	30	0.002	**0.027**
	240	60	0.000	0.110
AutoSklearn-m	30	10	0.004	0.211
	60	10	0.004	0.198
	60	30	0.00	0.956
	240	10	0.008	0.099
	240	30	0.004	0.614
	240	60	0.004	0.398
AutoSklearn-e	30	10	0.007	**0.000**
	60	10	0.011	**0.000**
	60	30	0.004	0.675
	240	10	0.013	**0.000**
	240	30	0.006	**0.038**
	240	60	0.002	0.265
AutoSklearn	30	10	0.003	0.362
	60	10	0.009	**0.000**
	60	30	0.005	**0.019**
	240	10	0.014	**0.001**
	240	30	0.011	**0.002**
	240	60	0.005	0.117

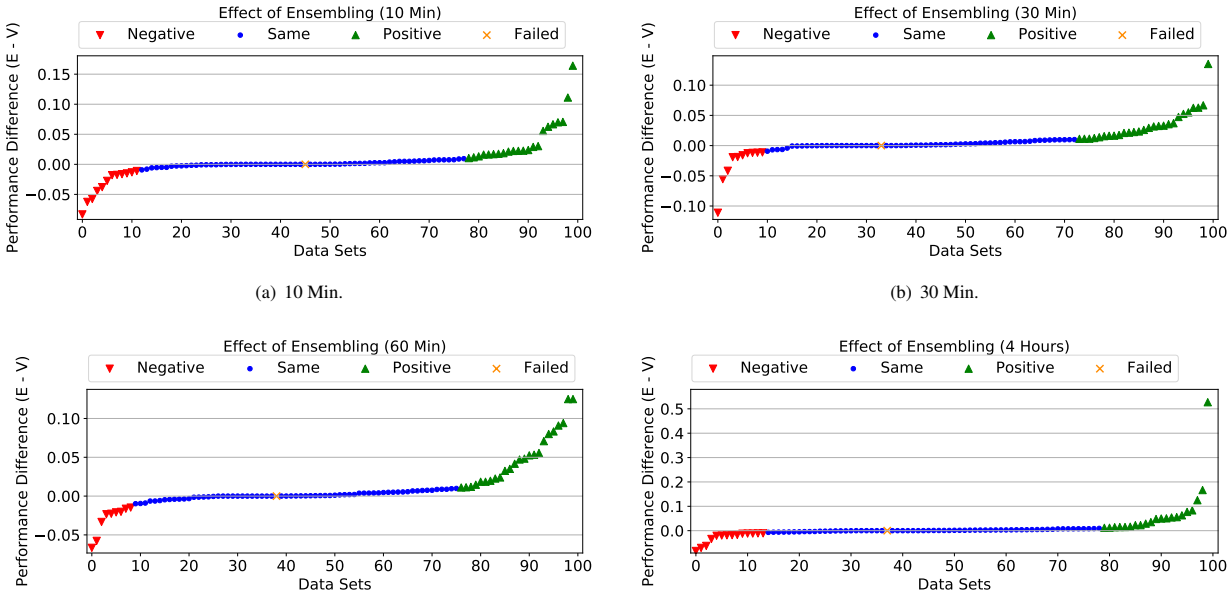
Table 6: Search space effect: result summary

Search Space	Mean	SD
3*C*	0.867	0.139
FC	0.863	0.153

achieve a performance improvement when increasing the time budget, while few datasets witness a performance decline. Thus, offering more time for `AutoSklearn` to search for a better solution *generally* lead to accuracy performance gains as previously established in [7]. Table 5 shows the statistical significance of increasing the time budget [27]. In particular, increasing the time budget from 10 minutes to 30 minutes and from 60 minutes to 240 minutes do not provide a statistically significant performance gain. On the other hand, increasing the time budget from 10 to 60, 30 to 60, 10 to 240, and 30 to 240 provide a statistically significant accuracy gain.

4.4 The Impact of The Size of The Search Space

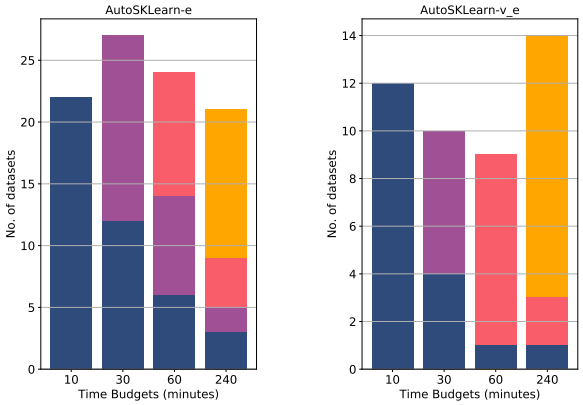
In this experiment, we compare the accuracy using the full search space, with all available classifiers (*FC*), to a subset of search space containing the best-performing classifiers (3*C*). In practice, we selected the top 3 classifiers, i.e., *support vector machine, random forest, and decision trees*, based on the results of the *FC*. Table 6



(a) 10 Min. (b) 30 Min.

(c) 60 Min. (d) 240 Min.

Figure 3: The impact of ensembling over all time budgets. Upward triangles represent better performance with `Auto-Sklearn-e`, downward triangles represent better performance using `AutoSklearn-v`, and circles means that the absolute difference is $< 1\%$



(a) `AutoSklearn-e` **(b) `AutoSklearn-v`**

Figure 4: Overlap among datasets having better performance using `AutoSklearn-e`(a) and `AutoSklearn-v`(b) through each time budget. The new color in each bar represents the number of new datasets with higher performance at this time budget, while the same color represents the same datasets from previous time budgets.

shows that both search spaces have a comparable performance. Figure 6 shows the effect of using the *FC* against using only 3*C* on `AutoSklearn`. The results show that there is no clear winner. In particular, the *FC* exceeds the accuracy of 3*C* in 28 datasets with an average accuracy gain of 3.3%, while using 3*C* achieves better performance on 21 datasets with an average accuracy difference of 5.9%. Besides, 50 datasets have negligible accuracy differences

Table 7: The results of Wilcoxon test for assessing the statistical significance of the performance difference for increasing the search space

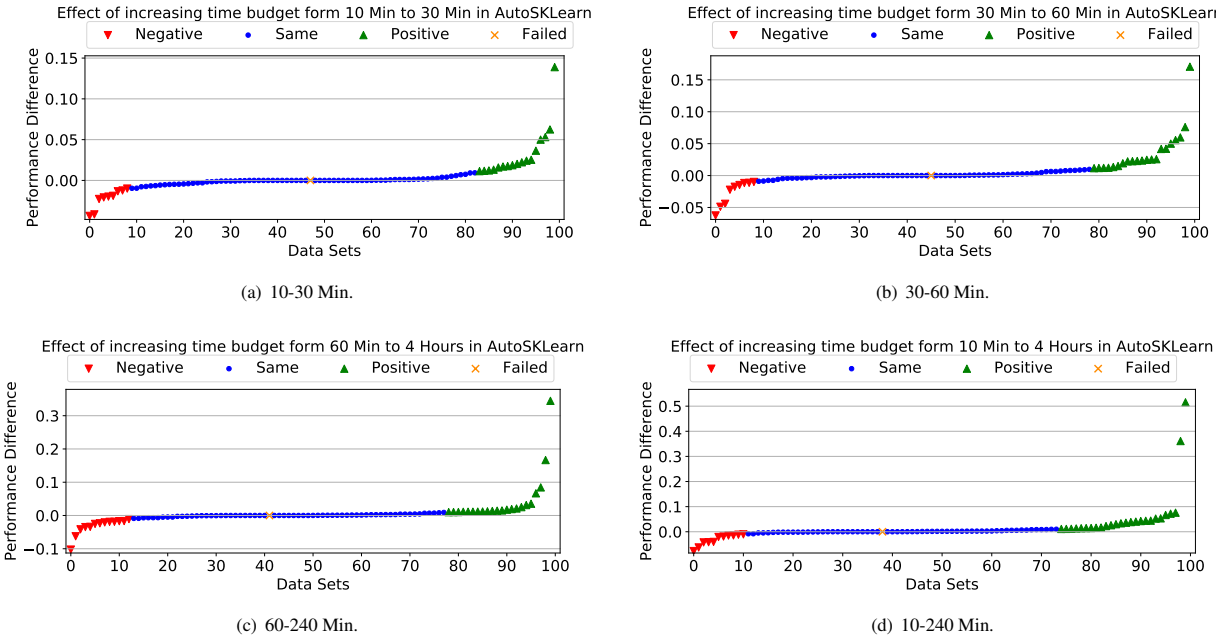
Mode 1	Mode 2	Avg. Acc. Diff	*P* value
FC	3*C*	-0.003	0.618

(less than 1%). The Wilcoxon test (Table 7) shows no statistically significant difference between the two search spaces. Although the *FC* is much larger, it does not reduce accuracy than the exploited search space (3*C*). Therefore, it is better to keep all classifiers in the target search space.

4.5 Special Runs and Discussion

The datasets with substantial performance differences towards/against any of the discussed learning settings are further investigated and reran 3 times per configuration. We noticed that most of these datasets have an order of magnitude fewer instances than features or have significantly few instances (mostly with datasets from medical domains); see Table 8. Generally, the generated pipelines and their accuracy for these kinds of datasets are completely different in each iteration. For instance, 5 different classifiers are selected in 6 unique pipelines for the `dataset_40_sonar` (`sonar`) dataset.

In principle, the importance of the learners' hyper-parameters varies based on their effect on the accuracy [8]. Moreover, the importance of the hyper-parameters depends on the dataset characteristics. For example, the regularization parameter is critical for datasets with fewer instances than features to avoid over-fitting. Hence, the AutoML tool should pay more attention to it for better generalization with the current few instances.



(a) 10-30 Min.

(b) 30-60 Min.

(c) 60-240 Min.

(d) 10-240 Min.

Figure 5: The impact of increasing the time budget on `AutoSklearn` performance from x to y minutes (x-y). Upward triangles represent better performance with y time budget. Downward triangles represent better performance on x time budget. Circles mean that the difference between x and y is $< 1\%$.

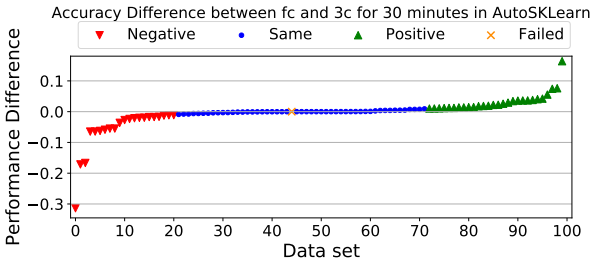


Figure 6: The impact of reducing the search space size on each AutoML framework. Upward triangles represent better performance with *FC* search space. Downward triangles represent better performance with 3*C* search space. circles means that the difference between *FC* and 3*C* is $< 1\%$.

In `AutoSklearn`, the ML pipeline structure consists of three fixed components, i.e., data preprocessor, feature preprocessor, and classifier. `AutoSklearn` tries several options for each stage and selects the one that maximizes the validation accuracy. Since the feature engineering phase is significant, the output pipelines have high-performance differences when they have different feature preprocessors, even if the same classifier is selected for all of them. For example, although `lda` is selected as a classifier in two pipelines for `dbworld-bodies` (`bodies`), their accuracies are very different; i.e., 93.8% for `AutoSklearn-v` since it used `nystroem_sampler` preprocessor compared to 75% for `AutoSklearn-m` without any preprocessors. Additionally, over `AutoSklearn-v`, two pipelines with the same Gaussian naive Bayes `gaussian_nb` classifiers generate 100% and 83.3% accuracy values with different preprocessors. In practice, the feature engineering phase consumes most of the data scientists' time; however, it is not handled very well by any AutoML tool, including `AutoSklearn` [24]. Therefore, there is a huge room of improvement in the automated feature engineering phase.

These results reflect the great importance of the *feature engineering* phase as a crucial step in classical machine learning. The right *feature engineering* phase could turn the feature space into a linearly separable space, so even naive classifiers could achieve relatively high accuracy. On the other hand, skipping this phase or using the wrong *feature engineering* preprocessors makes it harder to achieve relatively high accuracy, even for the most efficient classifiers. Therefore, the image datasets that use many raw pixels as features usually have an oscillated performance based on the preprocessors selected in the feature engineering phase. Consequently, the pipeline that uses more suitable preprocessors to the target datasets relatively achieves better accuracy.

Using `AutoSklearn-e's` greedy implementation of ensembling with datasets having significantly few instances declines the performance since the validation set is expected to contain significantly few instances too. The fitted model is vulnerable to over-fitting on such a small validation set, e.g., `GCM` in Table 8.

We believe that when dealing with really *big* datasets[7], the optimization process would not have the luxury to attempt the same large number of configurations. The reason behind this is the significant costs (e.g., time and computing resources) associated with each configuration attempt. Thus, to tackle the challenge of dealing with big datasets, there is a crucial need for a distributed AutoML search process. For such *big* datasets, the meta-learning mechanism can have a better significant impact on reducing the search space and optimizing the search process with possibly a

[7]The average size of the 100 datasets of our experiments is 21.2MB (relatively small). Relatively big datasets such as `Cifar-10` (643MB) have failed with `AutoSklearn`.

Table 8: A sample of the datasets' characteristics and results from the repeated (special) runs. 'm', 'e', 'v' stands for the version of `AutoSklearn` (A).

.

Dataset	#Feat.	#Inst.	A	Accuracy		
10 minutes						
sonar	61	208	m	0.885	0.827	0.788
			v	0.846	0.827	0.712
bodies	4703	64	m	0.875	0.875	0.75
			v	0.938	0.938	0.875
tumors_C	7130	60	m	0.666	0.666	0.60
			v	0.666	0.466	0.466
micro-mass	1301	517	m	0.881	0.839	0.811
			v	0.947	0.867	0.832
GCM	160064	190	e	0.604	0.604	0.583
			v	0.792	0.708	0.646
30 minutes						
stemmed	3722	64	m	0.812	0.812	0.75
			v	0.875	0.875	0.812
lymphoma	4027	45	m	0.812	0.812	0.75
			v	0.875	0.875	0.812
rsctc2010_3	22278	95	m	0.958	0.875	0.875
			v	1.0	0.833	0.75
240 minutes						
CovPokElec	65	1.4 M	m	0.954	0.888	0.62
			v	0.80	0.572	0.504

lower number of attempts on the defined time budgets (See Table 8 for `CovPokElec` dataset).

5 CONCLUSION

This paper analyzed and presented various learning settings employed and considered by `AutoSklearn` and AutoML frameworks in general. The analysis revealed several insights that can help guiding and improving the design process of future AutoML techniques. For example, no single configuration of the learning settings can *always* guarantee an improved performance for all datasets. Each configuration usually leads to a better performance on some datasets. The meta-learning mechanism pioneered by `AutoSklearn` achieves a statistically significant performance improvement with short time budgets only, and it significantly loses its impact with longer time budgets. Hence, we only recommend using meta-learning with limited time budgets or huge datasets that take a long time to train a single model. Using ensemble models, the results are consistently improved for all time budgets. Thus, ensembling is recommended, especially with datasets with many features and few instances, since it reduces the chances of overfitting the validation split. Increasing the time budget needs to be considered carefully as it does not always lead to a significant improvement of the accuracy. This decision can vary from one scenario/application to another according to the resource-accuracy tradeoff. Deliberately selecting a small search space with a few top-performing classifiers can lead to a very comparable performance with a search space that includes many classifiers. This insight is essential, especially for large datasets that cannot be evaluated using many classifiers. Finally, this study opens the door to adaptively configure the default learning settings for each input dataset based on its characteristics.

ACKNOWLEDGEMENT

This work is funded by the European Regional Development Funds via the Mobilitas Plus programme (grant MOBTT75).

REFERENCES

[1] James Bergstra, Dan Yamins, and David D Cox. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*. Citeseer, 13–20.
[2] Bernd Bischl et al. 2017. OpenML benchmarking suites and the OpenML100. *arXiv preprint arXiv:1708.03731* (2017).
[3] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. In *ICML*.
[4] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*.
[5] Salijona Dyrmishi, Radwa Elshawi, and Sherif Sakr. 2019. A Decision Support Framework for AutoML Systems: A Meta-Learning Approach. In *Proceedings of The 1st IEEE ICDM Workshop on Autonomous Machine Learning (AML)*.
[6] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. Bohb: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774* (2018).
[7] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2020. Auto-sklearn 2.0: The next generation. *arXiv preprint arXiv:2007.04074* (2020).
[8] Matthias Feurer and Frank Hutter. 2019. Hyperparameter optimization. In *Automated Machine Learning*. Springer, Cham, 3–33.
[9] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *NIPS*.
[10] Edmund A Gehan. 1965. A generalized Wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika* 52, 1-2 (1965), 203–224.
[11] Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, and Joaquin Vanschoren. 2019. An open source AutoML benchmark. *arXiv preprint arXiv:1907.00909* (2019).
[12] Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, et al. 2019. Analysis of the AutoML Challenge Series. *Automated Machine Learning* (2019), 177.
[13] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2019. AutoML: A Survey of the State-of-the-Art. *arXiv preprint arXiv:1908.00709* (2019).
[14] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*.
[15] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-keras: An efficient neural architecture search system. In *ACM KDD*.
[16] Aaron Klein, Stefan Falkner, Numair Mansur, and Frank Hutter. 2017. Robo: A flexible and robust bayesian optimization framework in python. In *NIPS 2017 Bayesian Optimization Workshop*.
[17] Mohamed Maher and Sherif Sakr. 2019. SmartML: A Meta Learning-Based Framework for Automated Selection and Hyperparameter Tuning for Machine Learning Algorithms. In *EDBT*.
[18] Sherif Sakr and Albert Y. Zomaya (Eds.). 2019. *Encyclopedia of Big Data Technologies*. Springer. https://doi.org/10.1007/978-3-319-63962-8
[19] Radwa El Shawi, Mohamed Maher, and Sherif Sakr. 2019. Automated Machine Learning: State-of-The-Art and Open Challenges. *CoRR* abs/1906.02287 (2019). arXiv:1906.02287 http://arxiv.org/abs/1906.02287
[20] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.
[21] Thomas Swearingen, Will Drevo, Bennett Cyphers, Alfredo Cuesta-Infante, Arun Ross, and Kalyan Veeramachaneni. 2017. ATM: A distributed, collaborative, scalable system for automated machine learning. In *2017 IEEE International Conference on Big Data (Big Data)*.
[22] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *ACM KDD*.
[23] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. 2020. Fixing the train-test resolution discrepancy: FixEfficientNet. *arXiv preprint arXiv:2003.08237* (2020).
[24] Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan Hines, Bayan Bruss, and Reza Farivar. 2019. Towards automated machine learning: Evaluation and comparison of automl approaches and tools. *arXiv preprint arXiv:1908.05557* (2019).
[25] Joaquin Vanschoren. 2018. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548* (2018).
[26] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations* 15, 2 (2013), 49–60. https://doi.org/10.1145/2641190.2641198
[27] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*. Springer, 196–202.
[28] Marc-André Zöller and Marco F Huber. 2019. Benchmark and Survey of Automated Machine Learning Frameworks. (2019).
[29] Albert Y Zomaya and Sherif Sakr. 2017. *Handbook of big data technologies*. Springer.