[Expert Systems With Applications 243 (2024) 122877](https://doi.org/10.1016/j.eswa.2023.122877)





|  |
| --- |
|  |

Contents lists available at [ScienceDirect](https://www.elsevier.com/locate/eswa)

Expert Systems With Applications

journal homepage: [www.elsevier.com/locate/eswa](http://www.elsevier.com/locate/eswa)

|  |  |  |
| --- | --- | --- |
| AutoMLBench: A comprehensive experimental evaluation of automated machine learning frameworks  Hassan Eldeeb [a](#bookmark1),[b](#bookmark1) ,[∗](#bookmark2), Mohamed Maher [a](#bookmark1), Radwa Elshawi [a](#bookmark1), SherifSakr [a](#bookmark1) a *Institute of Computer Science, University of Tartu, Tartu, 51009, Estonia*  b *Computers and Automatic Dept., Faculty of Engineering, Tanta University, Tanta, Egypt* | | [check for updates](http://crossmark.crossref.org/dialog/?doi=10.1016/j.eswa.2023.122877&domain=pdf) |
| A R T I C L E I N F O | A B S T R A C T  With the booming demand for machine learning applications, it has been recognized that the number of knowledgeable data scientists cannot scale with the growing data volumes and application needs in our digital world. In response to this demand, several automated machine learning (AutoML) frameworks have been developed to fill the gap of human expertise by automating the process of building machine learning pipelines. Each framework comes with different heuristics-based design decisions. In this study, we present a comprehensive evaluation and comparison of the performance characteristics of six popular AutoML frameworks, namely, AutoWeka, AutoSKlearn, TPOT, Recipe, ATM and SmartML across 100 data sets from established AutoML benchmark suites. Our experimental evaluation considers different aspects for its comparison, including the performance impact of several design decisions, including *time budget*, *size of search space*, *meta-learning*, and *ensemble construction*. The results of our study reveal various interesting insights that can significantly guide and impact the design of AutoML frameworks. | |
| Dataset link: [https://datasystemsgrouput.githu](https://datasystemsgrouput.github.io/AutoMLBench/datasets)  [b](https://datasystemsgrouput.github.io/AutoMLBench/datasets).i[o/AutoMLBench/datasets](https://datasystemsgrouput.github.io/AutoMLBench/datasets) |
| *Keywords:*  Automated machine learning  Meta-learning Search space Ensemble  Time budget |

**1. Introduction**

We are witnessing tremendous interest in artificial intelligence ap- plications across governments, industries and research communities with a yearly cost of around 12.5 billion US dollars ([International](#bookmark3) [Data Corporation](#bookmark3), [2017](#bookmark3)). The driver for this interest is the advent and increasing popularity of machine learning (ML) and deep learning (DL) techniques. The rise of generated data from different sources, processing capabilities, and ML algorithms opened the way for adopting ML in a wide range of real-world applications ([Zomaya & Sakr](#bookmark4), [2017](#bookmark4)). This situation is increasingly contributing towards a potential *data science crisis*, similar to the software crisis ([Fitzgerald](#bookmark5), [2012](#bookmark5)), due to the crucial need to have an increasing number of data scientists with solid knowledge and good experience so that they can keep up with harnessing the power of the massive amounts of data produced daily. Thus, we are witnessing a growing interest in automating the process of building ML pipelines where the presence of a human in the loop can be dramatically reduced. Research in the area of AutoML aims to alleviate both the computational cost and human expertise required for developing ML pipelines through automation with efficient algorithms.

In particular, AutoML techniques enable the widespread use of ML techniques by domain experts and non-technical users.

Applying ML to real-world problems is a multi-stage process and highly iterative exploratory process. It aims to automatically produce the optimal ML pipeline that maximizes the predictive performance over the validation set of a dataset within a fixed computational budget (See [Fig.](#bookmark6) [1](#bookmark6)). The problem of AutoML for supervised task can be formally stated as follows: For *i* = 1*,* … *, n*′ +*m*′ , let *xi* ∈ **R** denote a feature vector and *yi* ∈ *Y* the corresponding target value. Given a training dataset *Dtrain* = {(*x*1*, y*1)*,* … *,* (*xn*′ *, yn*′ )} and the feature vectors *xn*′ +1*,* … *, xn*′ +*m*′ of a test dataset *Dtest* = {(*xn*′ +1*, yn*′ +1)*,* … *,* (*xn*′ +*m*′ *, yn*′ +*m*′ )} drawn from the same underlying data distribution, as well as a resource budget *b* and a loss metric *L*(*., .*), the AutoML problem is to automatically produce test set predictions *n*′ +1*,* … *, ̂yn*′ +*m*′ while minimizing the loss (*L*). The

loss of a solution *n*′ +*m*′ to the AutoML problem is given by  ∑1

*L*(*n*′ +*j, yn*′ +*j*.

The budget *b* would comprise computational resources (e.g., CPU and/or wallclock time, memory usage). In particular, solving the Au- toML problem aims to select and tune an ML algorithm from a defined search space to achieve (near)-optimal performance in terms of the

The code (and data) in this article has been certified as Reproducible by Code Ocean: (<https://codeocean.com/>). More information on the Reproducibility Badge Initiative is available at <https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals> .

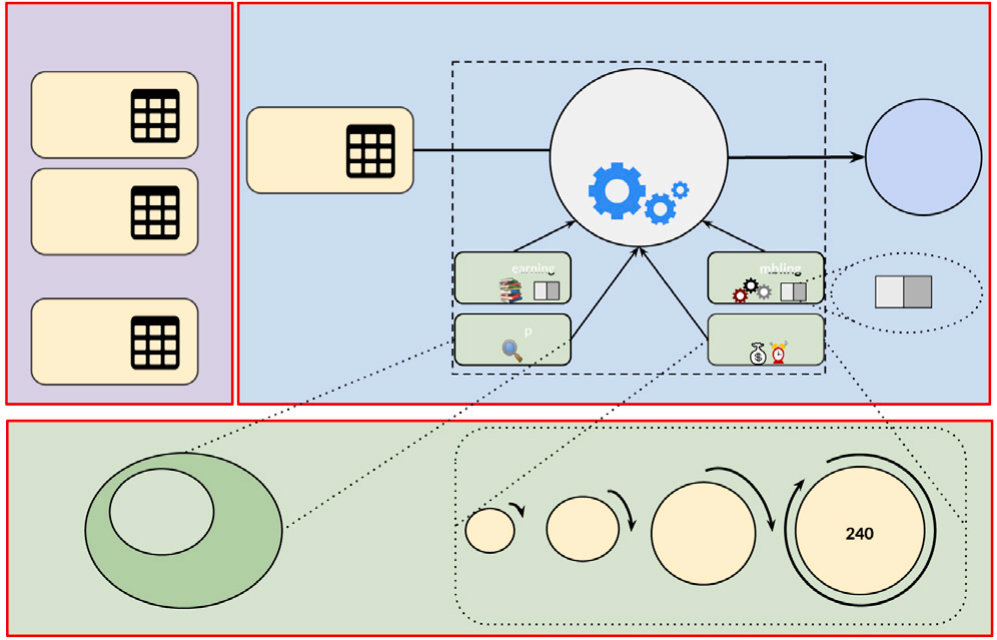
∗ Correspondence to: Data Systems Group, Institute of Computer Science, University of Tartu, Tartu, 51009, Estonia.

*E-mail addresses:* [hassan.eldeeb@ut.ee](mailto:hassan.eldeeb@ut.ee) (H. Eldeeb), [mohamed.abdelrahman@ut.ee](mailto:mohamed.abdelrahman@ut.ee) (M. Maher), [radwa.elshawi@ut.ee](mailto:radwa.elshawi@ut.ee) (R. Elshawi), [sherif.sakr@ut.ee](mailto:sherif.sakr@ut.ee) (S. Sakr).

<https://doi.org/10.1016/j.eswa.2023.122877>

Received 13 July 2023; Received in revised form 6 December 2023; Accepted 6 December 2023 Available online 9 December 2023

0957-4174/© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).



AutoML

Framework

optimization Algorithm

Data

Data 2



Meta-Learning

OFF

search space

Data 100

portfolio

240

search space

minutes



Tuned

ML

pipeline

100 Datasets



Time Budget

Ensembling

Data 1

ON



30



**Fig. 1.** The general Workflow of the benchmark design and AutoML process.

user-defined evaluation metric (e.g., accuracy, sensitivity, specificity, F1-score) within the user-defined budget for the search process, as shown in [Fig.](#bookmark6) [1](#bookmark6). Additionally, different AutoML frameworks consider various design decisions. For example, SmartML ([Maher & Sakr](#bookmark7), [2019](#bookmark7)) adopts a *meta-learning* based mechanism to improve the performance of the automated search process by starting with the most promis- ing classifiers that performed well with similar datasets in the past. Another example, AutoSKlearn ([Feurer et al.](#bookmark8), [2015](#bookmark8)) employs an option to take a weighted average of the predictions of an ensemble composed of the top trained models during the optimization process. Auto-Tuned Models (ATM) ([Swearingen et al.](#bookmark9), [2017](#bookmark9)) restricts the de- fault search space into only three classifiers, namely, decision tree, K-nearest neighbors, and logistic regression. Nevertheless, there is no clear understanding of the impact of various design decisions of the dif- ferent AutoML frameworks on the performance of the output pipeline. In this work, we aim to answer the following four questions:

(1) What is the impact of the time budget on the performance of different AutoML frameworks? Given more time budget, can AutoML frameworks guarantee consistent performance improvement?

(2) What is the impact of the search space size of the AutoML framework on the performance? How does limiting the search space to a predefined portfolio affect the predictive performance?

(3) Does meta-learning always yield a consistent performance im- provement across different time budgets? Is there a relationship be- tween the characteristics of the datasets and the improvement caused by employing the meta-learning version of the AutoML framework?

(4) Does ensemble construction yield better performance than single learners across different time budgets? Is there a relationship between the characteristics of the datasets and the improvement caused by employing the ensembling version of the AutoML framework?

This work is an extension of our initial work ([Eldeeb et al.](#bookmark10), [2021](#bookmark10)) that mainly focused on studying the impact of different design decisions on the performance of AutoSKlearn. More specifically, in this work, we follow a holistic approach to design and conduct a comparative study of six AutoML frameworks, namely AutoWeka ([Kotthoff et al.](#bookmark11), [2017](#bookmark11)), AutoSKlearn, TPOT ([Olson & Moore](#bookmark12), [2016](#bookmark12)), Recipe ([de](#bookmark13) [Sá et al.](#bookmark13), [2017](#bookmark13)), ATM and SmartML, focusing on comparing their general performance under various design decisions including *time budget*, *size of search space*, *meta-learning* and *ensembling*. For ensuring reproducibility as one of the main targets of this work, we provide access to the source codes and the detailed results for the experiments of our studies.[1](#bookmark14)

The remainder of this paper is organized as follows. The related work is reviewed in Section [2](#bookmark15). Section [3](#bookmark16) provides an overview of the

1 <https://datasystemsgrouput.github.io/AutoMLBench/>

evaluated frameworks included in our study. Section [4](#bookmark17) describes our benchmark design. The evaluation of the general performance of the benchmark frameworks and the evaluation of the different design deci- sions on the performance of the benchmark frameworks are presented in Section [5](#bookmark18). We discuss the results and future direction in Section [6](#bookmark19) before we finally conclude the paper in Section [7](#bookmark20).

**2. Related work**

Recently, few research efforts have attempted to tackle the chal- lenge of benchmarking different AutoML frameworks ([Gijsbers et al.](#bookmark21), [2019](#bookmark21); [He et al.](#bookmark22), [2019](#bookmark22); [Shawi et al.](#bookmark23), [2019](#bookmark23); [Truong et al.](#bookmark24), [2019](#bookmark24); [Zöller](#bookmark25) [& Huber](#bookmark25), [2021](#bookmark25)). In general, most experimental evaluation and com- parison studies show no framework always performed the best, as some trade-offs always need to be considered and optimized according to user-defined objectives. For example, [Gijsbers et al.](#bookmark26) ([2022](#bookmark26)) con- ducted a study to compare the performance of 9 AutoML frameworks, namely, Autogluon-tabular ([Erickson et al.](#bookmark27), [2020](#bookmark27)), AutoSKlearn, AutoSKlearn 2 ([Feurer et al.](#bookmark28), [2020](#bookmark28)), FLAML ([Wang et al.](#bookmark29), [2021](#bookmark29)), GAMA ([Gijsbers & Vanschoren](#bookmark30), [2020](#bookmark30)), H2O AutoML ([LeDell & Poirier](#bookmark31), [2020](#bookmark31)), LightAutoMLs ([Vakhrushev et al.](#bookmark32), [2021](#bookmark32)), MLjar ([Płońska &](#bookmark33) [Płoński](#bookmark33), [2021](#bookmark33)), and TPOT, across 71 classification and 33 regression tasks. The study includes techniques for comparing AutoML frame- works, including final model accuracy, inference time trade-offs, and failure analysis. Autogluon has a consistently higher average perfor- mance in this benchmark. Additionally, an interactive visualization tool is supported to explore further the results and reproducibility of the analyses performed. [Gijsbers et al.](#bookmark21) ([2019](#bookmark21)) have conducted an experi- mental study to compare the performance of 4 AutoML frameworks, namely, AutoWeka, AutoSKlearn, TPOT and H2O on 39 datasets across two time budgets (60 min and 240 min). The results showed that no single AutoML framework outperformed others across all time budgets. Surprisingly, on some datasets, none of the frameworks out- performed the Random Forest model within 4 h time budget. [Truong](#bookmark24) [et al.](#bookmark24) ([2019](#bookmark24)) compared the performance of 7 AutoML frameworks, namely, H2O, Auto-keras ([Jin et al.](#bookmark34), [2019](#bookmark34)), AutoSKlearn, Lud- wig,[2](#bookmark35) Darwin,[3](#bookmark35) TPOT and Auto-ml[4](#bookmark35) on 300 datasets across different time budgets. The results showed that no single framework outper- formed 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 frameworks. In particular, H2O slightly outperformed other frameworks for binary classification and regression tasks while achieving poor performance on multi-class classification tasks. Auto-keras showed a stable performance across all tasks and slightly outperformed other frameworks on multi-class classification tasks while achieving poor performance on binary classification tasks.  [Zöller and Huber](#bookmark25) ([2021](#bookmark25)) compared the performance of different optimization techniques, namely, *Grid Search*, *Random Search*, *RObust Bayesian Optimization* (ROBO) ([Klein et al.](#bookmark36), [2017](#bookmark36)), *Bayesian Tuning and Bandits* (BTB) ([Smith et al.](#bookmark37), [2020](#bookmark37)), *hyperopt* ([Bergstra et al.](#bookmark38), [2013b](#bookmark38)), *SMAC* ([Hutter et al.](#bookmark39), [2011](#bookmark39)), *BOHB* ([Falkner et al.](#bookmark40), [2018](#bookmark40)) and *Optu- nity* ([Smith et al.](#bookmark37), [2020](#bookmark37)). The results showed that all optimization techniques achieved comparable performance, and a simple search algorithm such as random search did not perform worse than other techniques. Thus, the study suggested that ranking optimization tech- niques on pure performance measures are not reasonable, and other aspects like scalability should also be considered. The study also com- pared the performance of 5 AutoML frameworks, namely, TPOT, hp- sklearn ([Komer et al.](#bookmark41), [2014](#bookmark41)), AutoSKlearn, ATM, and H2O on 73 real datasets. The study considered AutoSKLearn once with the default optimizer *SMAC* and once replacing *SMAC* with the random

2 <https://github.com/uber/ludwig>

3 <https://www.sparkcognition.com/product/darwin/>

4 <https://github.com/ClimbsRocks/auto_ml>

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

**Table 1**

Comparison table of the functionality of the AutoML frameworks considered in this study as of Jul 9, 2023.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Release  date | Popularity  (#of stars  on GitHub) | Optimization  technique | ML tool box | Meta-Learning | Post-processing | GUI | Data pre-processing |
| AutoWeka | 2013 | 314 | Bayesian  optimization | Weka | × | × | ✓ | ✓ |
| AutoSKlearn | 2015 | 7k | Bayesian  optimization | Scikit-Learn | ✓ | Ensemble selection | × | ✓ |
| TPOT | 2016 | 9.1k | Evolutionary optimization | Scikit-Learn | × | × | × | × |
| Recipe | 2017 | 50 | Grammar- based  genetic  algorithm | Scikit-Learn | × | × | × | ✓ |
| ATM | 2017 | 522 | Distributed Random  search  &  Tree-Parzen estimators | Scikit-Learn | × | × | × | ✓ |
| SmartML | 2019 | 23 | Bayesian  optimization | mlr, RWeka & other R packages | ✓ | Voting ensembles | × | ✓ |

search while ensemble building and meta-learning options are disabled. The comparison results showed that, on average, all AutoML frame- works performed quite similar with a maximum performance difference of 2.2%.

To the best of our knowledge, our study is the first to investigate the impact of different AutoML design decisions on predictive performance. We benchmark six open-source, centralized, and distributed AutoML frameworks, namely, AutoWeka, AutoSKlearn, TPOT, Recipe, ATM and SmartML on 100 datasets from established AutoML bench- mark suites. Differently from the previous benchmark studies focused

only on comparing the performance of different AutoML frameworks,

we take a holistic approach to studying the impact of various de- sign decisions, including the size of the search space, time budget, meta-learning, and ensembling construction on the performance of the AutoML frameworks.

**3. AutoML frameworks**

This section provides an introduction to the evaluated AutoML frameworks used in this study in terms of popularity (measured in terms of the number of stars on GitHub), ML tool-box used, optimization technique, whether they use meta-learning to learn from previous experience, whether they perform post-processing (e.g., ensemble con- struction), whether they use Graphical User Interface (GUI), or whether they perform pre-processing. [Table](#bookmark42) [1](#bookmark42) briefly summarizes the compar- ison across the AutoML frameworks considered in this study. More detailed comparisons between these frameworks follow in the rest of this section.

AutoWeka is implemented in Java on top of Weka, a popular ML library with a wide range of ML algorithms. AutoWeka em- ploys Bayesian optimization using SMAC ([Hutter et al.](#bookmark39), [2011](#bookmark39)) and TPE ([Bergstra et al.](#bookmark43), [2013a](#bookmark43)) for algorithm selection and hyperpa- rameter tuning. In particular, SMAC draws the relationship between algorithm performance and a given set of hyperparameters by esti- mating the predictive mean and variance of their performance along with the trees of a random forest model. TPE is a robust technique that separates low-performing parameter configurations from the best- performing ones.

AutoSKlearn is a tool for automating the process of building ML pipelines for classification and regression tasks. AutoSKlearn

is implemented on top of Scikit-Learn ([Buitinck et al.](#bookmark44), [2013](#bookmark44)), a popular Python ML package, and uses SMAC for algorithm selec- tion and hyperparameter tuning. AutoSKlearn uses meta-learning to initialize the optimization procedure. Additionally, ensemble selec- tion is implemented by combining the best pipelines to improve the performance of the output model. AutoSKlearn supports different execution options including the *vanilla* version (AutoSKlearn-v), the meta-learning version (AutoSKlearn-m), the ensembling selection version (AutoSKlearn-e), and the full version (AutoSKlearn), where all options are enabled.

TPOT is an AutoML framework for building classification and re- gression pipelines based on a genetic algorithm. ML pipelines can be ex- pressed as a computational graph, with different branches representing different preprocessing pipelines. These pipelines are then optimized using a multi-objective optimization technique to minimize pipeline complexity while optimizing for performance to reduce overfitting caused by the large search space ([Olson et al.](#bookmark45), [2016](#bookmark45)).

Recipe is an AutoML framework for building machine learning pipelines for classification tasks. Recipe follows the same optimiza- tion procedure as TPOT, exploiting the advantages of a global search. TPOT suffers from the unconstrained search problem in which re- sources can be spent on generating and evaluating invalid solutions. Recipe handles this problem by adding a grammar that reduces the generation of invalid pipelines and hence accelerates the optimization process.

ATM is a collaborative service for optimizing ML pipelines for clas- sification tasks. In particular, ATM supports parallel execution through multiple nodes/cores with a shared model hub storing the results out of these executions and improving the selection of pipelines that may outperform the currently chosen ones. ATM is based on a hybrid Bayesian and multi-armed bandit optimization technique to traverse the search space and report the target pipeline.

SmartML is the first AutoML R package for classification tasks. In the algorithm selection phase, SmartML employs a meta-learning approach to identify the best-performing algorithms on similar datasets. The hyperparameter tuning of SmartML is based on SMAC. SmartML maintains the results of the new runs to continuously enrich its knowl- edge base to further improve the performance and robustness of future runs. SmartML supports two execution options which are the base version SmartML-m that employs meta-leaning for warm-starting, and the ensemble version SmartML-e that additionally employs a voting ensemble mechanism.

**4. Benchmark design**

Each benchmark task consists of a dataset, a metric to optimize, and design decisions made by the user, including a specific time budget to use. We will briefly explain our choice for each.

**Datasets** We used 100 datasets collected from the popular OpenML repository ([Vanschoren et al.](#bookmark46), [2013](#bookmark46)), allowing users to query data for different use cases. Detailed descriptions of the datasets used in this study are given in [Table](#bookmark47) [A.9](#bookmark47) in [Appendix](#bookmark48) [A](#bookmark48). To evaluate the AutoML frameworks on a variety of dataset characteristics, we selected multiple datasets according to different criteria, including the number of classes, number of features, number of instances, number of categorical features per sample, number of instances with missing values, and the class entropy. The datasets represent a mix of binary (50%) and multiclass (50%) classification tasks, where the size of the largest dataset is 643MB.

**Performance metrics** The benchmark can be run with a wide range of measures per user’s choice. The reported results in this paper are based on F1-score. AutoML frameworks are optimized for the same metric they are evaluated on. The measures are estimated withhold-out validation; each dataset is partitioned into two parts, 70% for training and 30% for testing. All AutoML frameworks are applied to the same training and testing splits on all datasets. To eliminate the effects of non-deterministic factors, the performance reported in each experiment is based on an average of 10 trials. We report a performance of 0 for any framework if the number of failed trials exceeds or is equal to 5.

**Frameworks and design decisions** The frameworks considered in this paper are selected based on ease of use, variety of underlying optimization techniques and ML toolboxes, popularity measured by the number of stars on GitHub, and citation count. All frameworks considered in this work are open source. A reference to the source code of each framework is given in [Table](#bookmark49) [B.10](#bookmark49) in [Appendix](#bookmark50) [B](#bookmark50). We do plan to include more frameworks in future work. For AutoSKlearn, we consider four execution options; AutoSKlearn-v, AutoSKlearn- m, AutoSKlearn-e and AutoSKlearn. For SmartML, we consider two execution options including SmartML-m, and SmartML-e. We examined different design decisions, including the size of the search space, meta-learning, and ensemble construction as a post-processing step. We study the impact of these design decisions for only AutoML frameworks that support configuring these decisions. It is important to highlight that the optimization technique is not consistent among all the frameworks, which prevents drawing a clear conclusion about this point in this benchmark. We consider the following versions of the frameworks: AutoSKLearn 0.11.0, AutoWeka 2.5, TPOT 0.11.6, Recipe 1.0, ATM 0.2.2, and SmartML 0.2.

**Baseline method** To assess the effectiveness of the different Au- toML frameworks included in this work, we use a baseline method which is a simple pipeline consisting of an imputation of missing values and a random forest model ([Pedregosa et al.](#bookmark51), [2011](#bookmark51)).

**Time budget choice** All AutoML frameworks were used with four different time budgets. Each framework is limited by a soft time budget (10, 30, 60, and 240 min) and a hard one (10% more than soft time budget). If a framework exceeds the hard time budget, the run is termi- nated and considered failed. Setting a time budget for all experiments is not straightforward. While it is more favorable to un-set a time limit to guarantee the best performance for each framework, however, doing so for all the six evaluated frameworks, with different configurations, across the 100 datasets, with ten trials for each, is very time-consuming. Therefore we used four time budgets which led to more than 40000 experiments to run for a total of more than 88366-hour EC2 run-time. To keep the experiment run-time and cost to practical limits, we tested the maximum cut-off timeouts of 4 and 8 h on 14 randomly selected datasets. The results are reported in [Table](#bookmark52) [C.11](#bookmark52) in [Appendix](#bookmark53) [C](#bookmark53). Addi- tionally, the Wilcoxon signed-rank test was conducted to determine if a statistically significant difference in performance exists between the AutoML frameworks over the two-time budgets (See [Table](#bookmark52) [C.11](#bookmark52)). The

 AutoskLearn-e  AutoskLearn-m

 Autoweka  TPOT

 AutoskLearn

 AutoskLearn-V

 smartML-e  ATM

 Recipe

 smartML-m

80

70

50

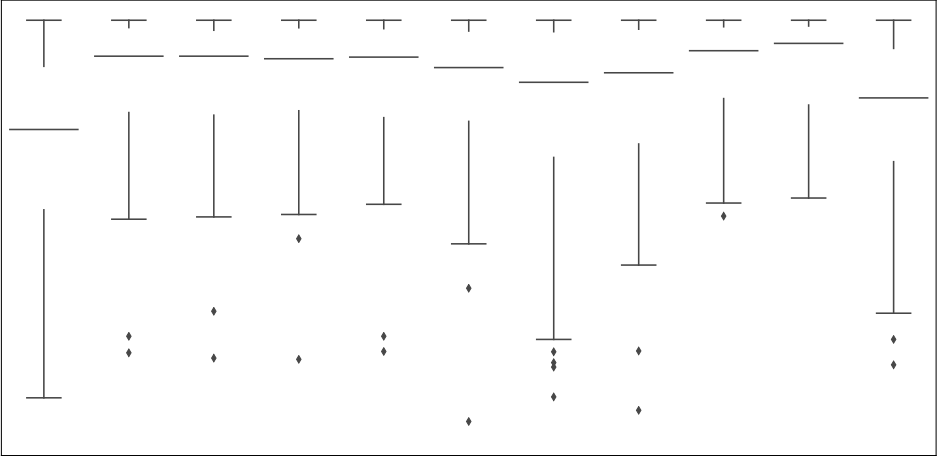
40

30

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

10 Min 30 Min 60 Min 240 Min Time Budget

(a) Number of successful runs .





|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

8

7

|  |
| --- |
|  |



5

4

3

2







Automl Framework

(b) performance of the fnal pipeline per AutoML framework for 240 minutes.

**Fig. 2.** General performance trends of the benchmark AutoML frameworks.

results confirm that the difference is not necessarily towards the 8-hour budget, and not statistically significant. Hence, the 8-hour budget is not further considered.

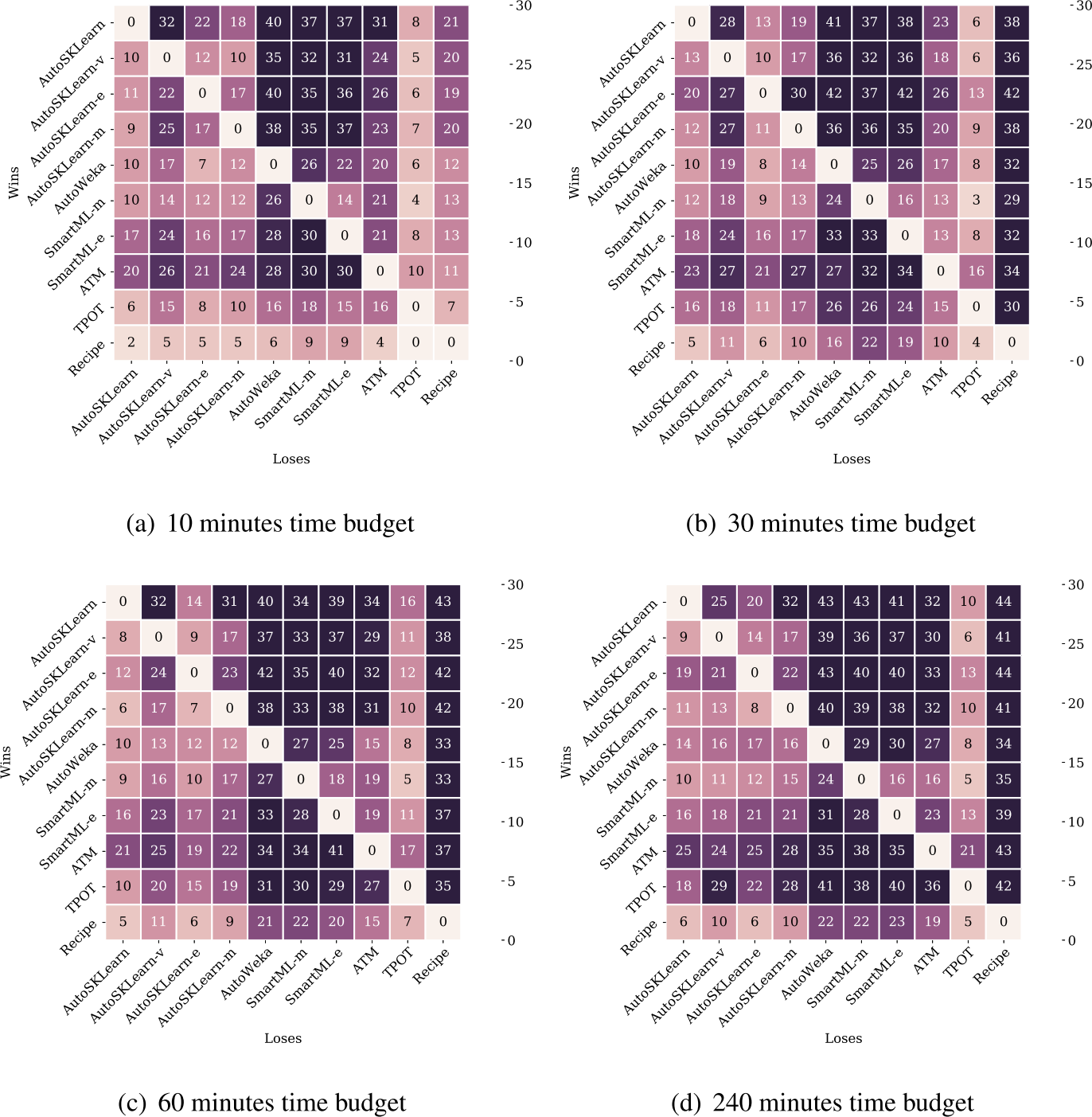
**Hardware choice and resource specifications** Our experiments were conducted on Google Cloud machines; each machine is configured with 2 vCPUs, 7.5 GB RAM and ubuntu-minimal-1804-bionic. Each ma- chine uses Python 2.7.15, Python 3.6.8, scikit-learn 0.21.3, R 3.4.4, and Java 1.8. To avoid memory leakage, we have rebooted the machines after each run to ensure that each experiment has the same available memory size.

**5. Experimental evaluation**

This section provides empirical evaluations of the different AutoML frameworks. We first compare the general performance of the different AutoML frameworks in Section [5.1](#bookmark55). Next, we examine the impact of various design decisions on the performance of the different AutoML frameworks in Section [5.2](#bookmark56).

*5.1. General performance evaluation*

In this section, we focus on evaluating and comparing the general performance of the benchmark frameworks. Our evaluation consid- ers different aspects for its comparison, including (a) the number of successful runs, (b) the average performance of the final pipeline per AutoML framework across all datasets, (c) the significance of the per- formance difference between different frameworks across different time budgets, (d) the robustness of the benchmark frameworks, and (e) the quality of the machine learning pipelines generated by the benchmark frameworks.



**Fig. 3.** Heatmaps show the number of datasets a given AutoML framework outperforms another in terms of predictive performance over different time budgets. Two frameworks are considered to have the same performance on a task if they achieve predictive performance with *<* 1% difference.

[Fig.](#bookmark57) [2(a)](#bookmark57) shows the number of datasets with successful runs of each framework on different time budgets. If an AutoML framework could not generate a model for a particular dataset 5 times or more, it is considered a failed experiment. Generally, the results show that increasing the time budget for the AutoML frameworks increases the number of successful runs. AutoSKlearn achieves the largest number of successful runs across all time budgets, as shown in [Fig.](#bookmark57) [2(a)](#bookmark57). Each of the different versions of AutoSKlearn successfully ran on 99 datasets across different time budgets. SmartML-e comes in second place in terms of the number of successful runs, followed by AutoWeka and SmartML. The genetic-based frameworks, TPOT and Recipe come in the last place, as shown in [Fig.](#bookmark57) [2(a)](#bookmark57). For Recipe and TPOT, the number of successful runs achieved in the longest time budget, 240 min, is almost double that achieved for the smallest time budget of 10 min. Hence, larger budgets are preferable for Recipe and TPOT.

[Fig.](#bookmark54) [2(b)](#bookmark54) reports the performances of all AutoML frameworks av- eraged over all datasets over 240 min budget. It is apparent that all frameworks are able to outperform the random forest baseline on average. However, single results vary significantly. [D.9](#bookmark59),[D.10](#bookmark60),[D.11](#bookmark61) in [Appendix](#bookmark62) [D](#bookmark62) report the performance of all AutoML frameworks and the baseline across 10, 30, 60 min, respectively. We investigate pair-wise ‘‘outperformance’’ by calculating the number of datasets for which one framework outperforms another across different time budgets, shown in [Fig.](#bookmark58) [3](#bookmark58). One framework outperforms another on a dataset if it has at least a 1% higher predictive performance, representing a minimal threshold for performance improvement. In terms of ‘‘outperformance’’, it is worth mentioning that no single AutoML framework performs best across all 100 datasets on all-time budgets. For example, for the 10 min time budget, there are 2 datasets for which Recipe performs

better than AutoSKlearn, despite being the overall worst- and best- ranked algorithms, respectively, as shown in [Fig.](#bookmark58) [3(a)](#bookmark58). On average, the results show that AutoSKlearn framework comes in the first place, outperforming other frameworks on the most significant number of datasets for different time budgets, followed by ATM framework, while Recipe comes in the last place, as shown in [Fig.](#bookmark58) [3](#bookmark58). The Wilcoxon signed-rank test ([Gehan](#bookmark63), [1965](#bookmark63)) was conducted to determine if a statis- tically significant difference in performance exists between the AutoML frameworks including the baseline over different time budgets, the results of which are summarized in [Table](#bookmark64) [2](#bookmark64). The results show that all AutoML frameworks except Recipe statistically outperform the base- line across all time budgets with a significant difference. The results of the Wilcoxon test confirm the fact that there is no dominating winner, and the statistical significance in the performance difference among the AutoML frameworks can vary from one-time budget to another. The ensembling version and the full version of AutoSKlearn statistically outperform most of the other frameworks across all time budgets. The results show that SmartML-m, SmartML-e, and AutoWeka are statistically outperformed by the majority of the frameworks, as shown in [Table](#bookmark64) [2](#bookmark64). For longer time budgets of 60 and 240 min, TPOT signif- icantly outperforms AutoWeka, Recipe, SmartML-m, SmartML-e, AutoSKlearn-m, and AutoSKlearn-v.

We investigate the performance of the different AutoML frameworks based on the various characteristics of datasets and tasks. [Fig.](#bookmark65) [4](#bookmark65) reports the mean performances of the AutoML frameworks on multi-class and binary-class classification tasks across 240 min budget. Notably, the improvement achieved by all AutoML on multi-class datasets is less sig- nificant than the average improvement on whole datasets. The second subgroup of datasets where autoML frameworks struggle to boost their

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

**Table 2**

Wilcoxon pairwise test p-values for AutoML frameworks over different time budgets. Bold entries highlight significant differences (*p* ≤ 0*.*05). Highlighted entries in each row represent a given AutoML framework (row) outperforms another AutoML framework (column).

**10Minutes**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Baseline | ATM | AutoWeka | Recipe | AutoSKLearn-e | AutoSKLearn-m | AutoSKLearn-v | AutoSKLearn | SmartML-m | SmartML-e | TPOT |
| Baseline |  | 0.0 | 0.0 | 0.15 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ATM | **0.0** |  | **0.008** | 0.088 | 0.768 | 0.516 | 0.299 | 0.587 | 0.064 | 0.062 | 0.879 |
| AutoWeka | **0.0** | 0.008 |  | 0.156 | 0.0 | 0.0 | 0.004 | 0.0 | 0.748 | 0.096 | 0.06 |
| Recipe | 0.15 | 0.088 | 0.156 |  | 0.002 | 0.002 | 0.004 | 0.0 | 0.417 | 0.248 | 0.013 |
| AutoSKLearn-e | **0.0** | 0.768 | **0.0** | **0.002** |  | 0.569 | **0.014** | 0.001 | **0.023** | 0.203 | 0.492 |
| AutoSKLearn-m | **0.0** | 0.516 | **0.0** | **0.002** | 0.569 |  | **0.009** | 0.009 | **0.004** | 0.1 | 0.33 |
| AutoSKLearn-v | **0.0** | 0.299 | **0.004** | **0.004** | 0.014 | 0.009 |  | 0.0 | **0.042** | 0.663 | **0.026** |
| AutoSKLearn | **0.0** | 0.587 | **0.0** | **0.0** | **0.001** | **0.009** | **0.0** |  | **0.001** | **0.035** | 0.258 |
| SmartML-m | **0.0** | 0.064 | 0.748 | 0.417 | 0.023 | 0.004 | 0.042 | 0.001 |  | 0.014 | 0.022 |
| SmartML-e | **0.0** | 0.062 | 0.096 | 0.248 | 0.203 | 0.1 | 0.663 | 0.035 | **0.014** |  | 0.452 |
| TPOT | **0.0** | 0.879 | 0.06 | **0.013** | 0.492 | 0.33 | 0.026 | 0.258 | **0.022** | 0.452 |  |

**30Minutes**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Baseline | ATM | AutoWeka | Recipe | AutoSKLearn-e | AutoSKLearn-m | AutoSKLearn-v | AutoSKLearn | SmartML-m | SmartML-e | TPOT |
| Baseline |  | 0.0 | 0.0 | 0.346 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ATM | **0.0** |  | **0.034** | **0.0** | 0.898 | 0.408 | 0.159 | 0.85 | **0.009** | **0.015** | 0.902 |
| AutoWeka | **0.0** | 0.034 |  | **0.004** | 0.0 | 0.0 | 0.003 | 0.0 | 0.92 | 0.195 | 0.003 |
| Recipe | 0.346 | 0.0 | 0.004 |  | 0.0 | 0.0 | 0.0 | 0.0 | 0.134 | 0.007 | 0.0 |
| AutoSKLearn-e | **0.0** | 0.898 | **0.0** | **0.0** |  | **0.015** | **0.0** | 0.694 | **0.001** | **0.005** | 0.94 |
| AutoSKLearn-m | **0.0** | 0.408 | **0.0** | **0.0** | 0.015 |  | **0.03** | 0.152 | **0.006** | 0.075 | 0.316 |
| AutoSKLearn-v | **0.0** | 0.159 | **0.003** | **0.0** | 0.0 | 0.03 |  | 0.0 | 0.064 | 0.112 | 0.005 |
| AutoSKLearn | **0.0** | 0.85 | **0.0** | **0.0** | 0.694 | 0.152 | **0.0** |  | **0.002** | **0.014** | 0.337 |
| SmartML-m | **0.0** | 0.009 | 0.92 | 0.134 | 0.001 | 0.006 | 0.064 | 0.002 |  | 0.015 | 0.002 |
| SmartML-e | **0.0** | 0.015 | 0.195 | **0.007** | 0.005 | 0.075 | 0.112 | 0.014 | **0.015** |  | 0.065 |
| TPOT | **0.0** | 0.902 | **0.003** | **0.0** | 0.94 | 0.316 | **0.005** | 0.337 | **0.002** | 0.065 |  |

**60Minutes**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Baseline | ATM | AutoWeka | Recipe | AutoSKLearn-e | AutoSKLearn-m | AutoSKLearn-v | AutoSKLearn | SmartML-m | SmartML-e | TPOT |
| Baseline |  | 0.0 | 0.0 | 0.201 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ATM | **0.0** |  | **0.017** | **0.0** | 0.075 | 0.358 | 0.424 | 0.149 | **0.005** | **0.004** | 0.064 |
| AutoWeka | **0.0** | 0.017 |  | **0.015** | 0.0 | 0.0 | 0.0 | 0.0 | 0.59 | 0.083 | 0.0 |
| Recipe | 0.201 | 0.0 | 0.015 |  | 0.0 | 0.0 | 0.0 | 0.0 | 0.054 | 0.003 | 0.0 |
| AutoSKLearn-e | **0.0** | 0.075 | **0.0** | **0.0** |  | **0.046** | **0.003** | 0.319 | **0.003** | **0.011** | 0.198 |
| AutoSKLearn-m | **0.0** | 0.358 | **0.0** | **0.0** | 0.046 |  | 0.474 | 0.0 | **0.012** | 0.052 | 0.067 |
| AutoSKLearn-v | **0.0** | 0.424 | **0.0** | **0.0** | 0.003 | 0.474 |  | 0.0 | **0.039** | 0.201 | 0.01 |
| AutoSKLearn | **0.0** | 0.149 | **0.0** | **0.0** | 0.319 | **0.0** | **0.0** |  | **0.001** | **0.015** | 0.86 |
| SmartML-m | **0.0** | 0.005 | 0.59 | 0.054 | 0.003 | 0.012 | 0.039 | 0.001 |  | 0.047 | 0.0 |
| SmartML-e | **0.0** | 0.004 | 0.083 | **0.003** | 0.011 | 0.052 | 0.201 | 0.015 | **0.047** |  | 0.007 |
| TPOT | **0.0** | 0.064 | **0.0** | **0.0** | 0.198 | 0.067 | **0.01** | 0.86 | **0.0** | **0.007** |  |

**4Hours**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Baseline | ATM | AutoWeka | Recipe | AutoSKLearn-e | AutoSKLearn-m | AutoSKLearn-v | AutoSKLearn | SmartML-m | SmartML-e | TPOT |
| Baseline |  | 0.0 | 0.0 | 0.039 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ATM | **0.0** |  | **0.046** | **0.0** | 0.637 | 0.943 | 0.969 | 0.754 | **0.002** | 0.061 | 0.153 |
| AutoWeka | **0.0** | 0.046 |  | **0.027** | 0.0 | 0.001 | 0.002 | 0.0 | 0.773 | 0.389 | 0.0 |
| Recipe | **0.039** | 0.0 | 0.027 |  | 0.0 | 0.0 | 0.0 | 0.0 | 0.024 | 0.004 | 0.0 |
| AutoSKLearn-e | **0.0** | 0.637 | **0.0** | **0.0** |  | **0.015** | **0.021** | 0.447 | **0.001** | **0.007** | 0.152 |
| AutoSKLearn-m | **0.0** | 0.943 | **0.001** | **0.0** | 0.015 |  | 0.852 | 0.0 | **0.006** | **0.043** | 0.001 |
| AutoSKLearn-v | **0.0** | 0.969 | **0.002** | **0.0** | 0.021 | 0.852 |  | 0.001 | **0.004** | 0.06 | 0.0 |
| AutoSKLearn | **0.0** | 0.754 | **0.0** | **0.0** | 0.447 | **0.0** | **0.001** |  | **0.0** | **0.002** | 0.119 |
| SmartML-m | **0.0** | 0.002 | 0.773 | **0.024** | 0.001 | 0.006 | 0.004 | 0.0 |  | 0.031 | 0.0 |
| SmartML-e | **0.0** | 0.061 | 0.389 | **0.004** | 0.007 | 0.043 | 0.06 | 0.002 | **0.031** |  | 0.001 |
| TPOT | **0.0** | 0.153 | **0.0** | **0.0** | 0.152 | **0.001** | **0.0** | 0.119 | **0.0** | **0.001** |  |

performance contains datasets with a relatively large number of fea- tures and asmall number of instances as shown in [D.13](#bookmark66),[D.14](#bookmark67),[D.15](#bookmark68),[D.12](#bookmark69) in [Appendix](#bookmark62) [D](#bookmark62). These figures report the mean performance of the different AutoML frameworks on datasets with various characteristics, including a large number of instances and features, a small number of features and instances, a small number of features and a large number of instances, and a large number of features and a small number of instances. Additionally, we report the mean performance of all frameworks on binary classification tasks (See [Fig.](#bookmark70) [4(b)](#bookmark70) in [Appendix](#bookmark62) [D](#bookmark62)).

We test the robustness of the AutoML frameworks evaluated by the ability of the framework to achieve the same results across different runs on the same input dataset. For a randomly selected dataset, we run each AutoML framework for 10 different times on 10 min time budget. [Fig.](#bookmark71) [5](#bookmark71) shows the robustness of the AutoML frameworks. The results show that the four versions of AutoSKlearn have the most stable runs, and Recipe and AutoWeka comes second. on contrast, the two versions of SmartML achieve the least stable runs.

In our evaluation of machine learning (ML) pipelines generated by AutoML frameworks, we employ a set of well-defined evaluation metrics to provide a comprehensive understanding of the factors that significantly affect the practicality and utility of ML pipelines produced by AutoML frameworks. Following the approach outlined in ([Ali et al.](#bookmark72), [2017](#bookmark72)), we establish quality metrics with the goal of offering a new perspective on the assessment of AutoML frameworks. These met- rics particularly focus on evaluating the frameworks’ efficiency during training, performance during inference, model stability, and robustness to noise. In [Table](#bookmark73) [3](#bookmark73), we present the average performance of AutoML frameworks using these metrics across all datasets. Definitions for these metrics are provided below.

• **Training Efficiency:** assesses the training efficiency of an ML model, measuring the CPU wall-clock time required for model training. This metric provides insights into the computational resources consumed during the training process. [Table](#bookmark73) [3](#bookmark73) shows

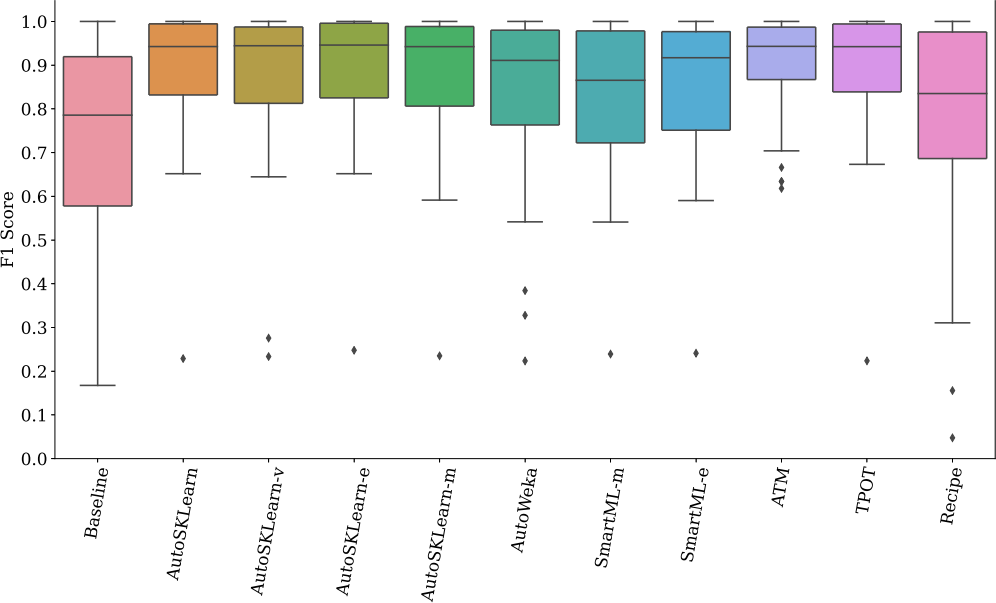
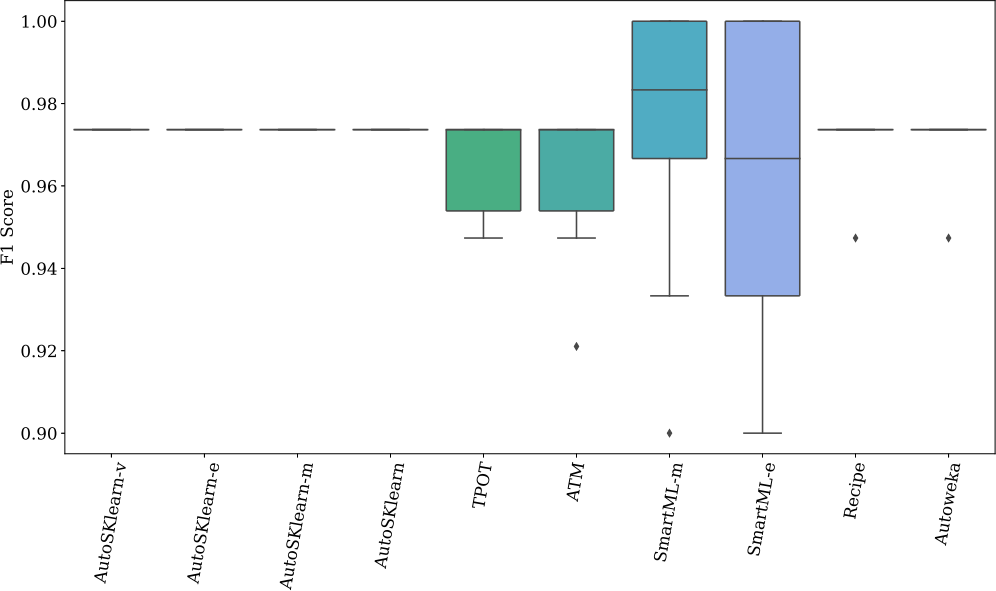
*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

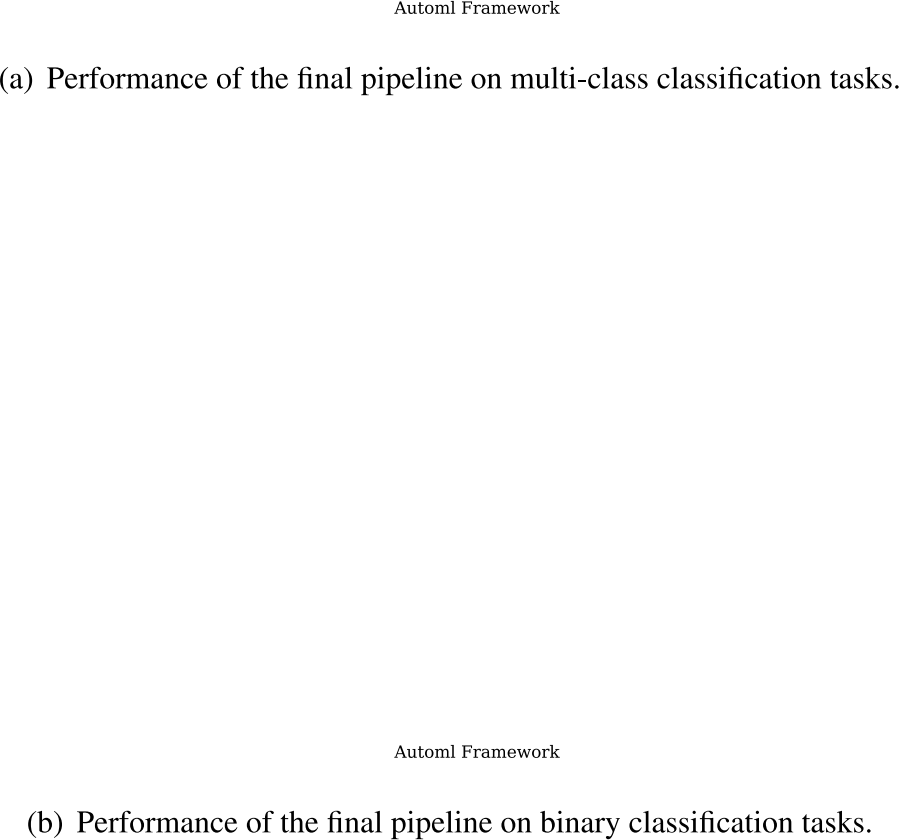
**Table 3**

Performance evaluation of benchmark AutoML frameworks using key metrics: training efficiency, inference performance, robustness to noise and model stability.

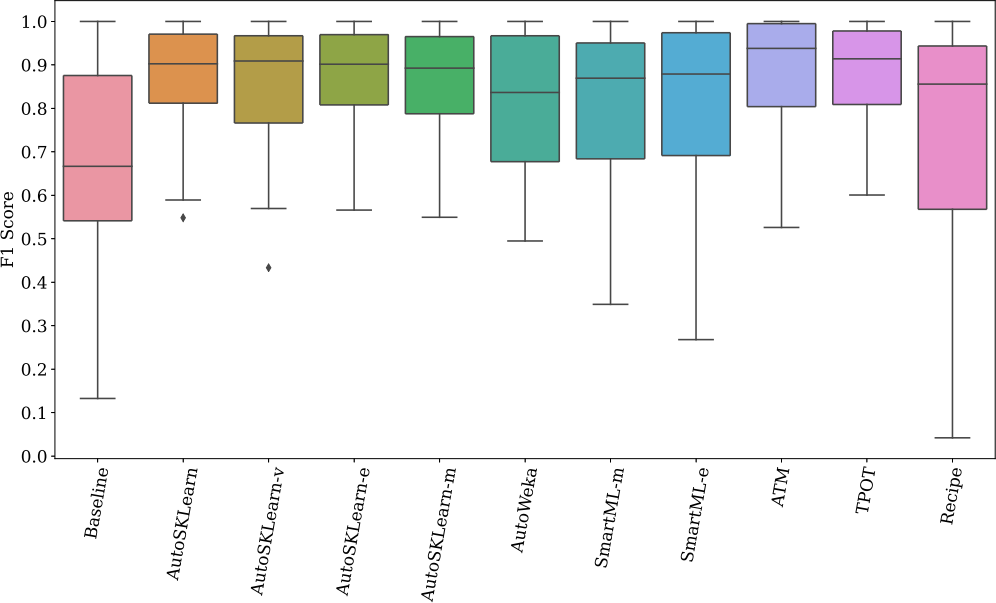
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Framework | Training  efficiency (s) | Inference  performance (s) | Robustness to noise | Model  stability |
| ATM | 43.25 ± 1.17 | 0.100 ± 0.007 | 0.07 | 0.0014 |
| AutoWeka | 46.85 ± 1.28 | 0.103 ± 0.008 | 0.05 | 0.0016 |
| Recipe | 43.10 ± 0.95 | 0.098 ± 0.007 | 0.04 | 0.0013 |
| AutoSKLearn-e | 66.50 ± 1.11 | 0.106 ± 0.007 | 0.10 | 0.0017 |
| AutoSKLearn-m | 45.20 ± 0.98 | 0.099 ± 0.007 | 0.10 | 0.0015 |
| AutoSKLearn-v | 44.80 ± 1.15 | 0.101 ± 0.008 | 0.06 | 0.0018 |
| AutoSKLearn | 67.20 ± 1.02 | 0.105 ± 0.007 | 0.07 | 0.0016 |
| SmartML-m | 53.00 ± 0.93 | 0.098 ± 0.007 | 0.04 | 0.0014 |
| SmartML-e | 73.50 ± 1.10 | 0.107 ± 0.008 | 0.07 | 0.0017 |
| TPOT | 48.80 ± 1.19 | 0.104 ± 0.008 | 0.09 | 0.0015 |







**Fig. 5.** Evaluation of AutoML frameworks for robustness on (dataset\_61\_iris).



predictions are necessary, such as online recommendation sys- tems or autonomous vehicles. Conversely, SmartML-e lags behind, as it records the longest inference time.

• **Model Stability:** assesses the consistency of an ML model by calculating the standard deviation when the model is retrained on the same dataset. Notably, Recipe, SmartML-m, and ATM exhibit the most stable performance among all frameworks, as evidenced by their lower standard deviation.

• **Robustness to Noise:** evaluates the resilience of an ML model to noise introduced into the data. Robustness is assessed by observing how the model’s behavior changes when noise is in- troduced to the test set. Greater variations in a model’s behavior in response to noise indicate lower robustness. The difference in behavior is quantified using the Cohen’s Kappa metric ([Landis](#bookmark74) [& Koch](#bookmark74), [1977](#bookmark74)). More specifically, we use Kappa as a measure of similarity between a model’s predictions on the original and perturbed test sets. Instances in the test sets are perturbed by modifying their feature values within a range of possible values. For continuous features, we introduce Gaussian noise with level 10 ([Beerenwinkel & Siebourg](#bookmark75), [2012](#bookmark75)). For categorical features, random uniform noise is added, implying that all categories, except the original one, have an equal probability (1∕*n* − 1) of being selected as noise, where *n* is the number of categories.

**Fig. 4.** Performance of the different AutoML frameworks based on the various characteristics of datasets and tasks over 240 min.

that ATM and Recipe offer the shortest training periods among the benchmarked frameworks. This is particularly advantageous for applications with limited computational resources or for scenarios where rapid model development is essential. On the other hand, AutoSKLearn and SmartML-e require significantly more time for training.

• **Inference Performance:** measures the inference speed of an ML model by quantifying the CPU wall-clock time required to produce a single prediction. The results in [Table](#bookmark73) [3](#bookmark73) highlight that SmartML-m and Recipe stand out with shorter inference times. This information is vital for use cases where quick decisions or

The results in [Table](#bookmark73) [3](#bookmark73) reveal that AutoSKLearn-e and Auto- SKLearn-m exhibit the highest levels of robustness among the frameworks, while Recipe and SmartML-m demonstrate compar- atively lower levels of robustness.

*5.2. Performance evaluation of different design decisions*

In this section, we study the impact of different design decisions in- cluding time budget (Section [5.2.1](#bookmark76)), size of search space (Section [5.2.2](#bookmark77)),

7

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

**Table 4**

Mean*Succ* , Mean and standard deviation of the predictive performance of AutoML frameworks per time budget. Bold entries highlight highest Mean*Succ* , mean and lowest standard deviation.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Time budget | Framework | *MeanSucc* | Mean | SD | Time budget | Framework | *MeanSucc* | Mean | SD |
| **10 Min** | **ATM** | 0.664 | 0.886 | 0.126 |  | **ATM** | 0.700 | **0.886** | **0.132** |
| **AutoWeka** | 0.724 | 0.842 | 0.165 |  | **AutoWeka** | 0.743 | 0.835 | 0.167 |
| **Recipe** | 0.252 | 0.764 | 0.221 |  | **Recipe** | 0.568 | 0.748 | 0.247 |
| **AutoSKLearn-e** | **0.859** | 0.868 | 0.145 |  | **AutoSKLearn-e** | **0.870** | 0.879 | 0.138 |
| **AutoSKLearn-m** | 0.855 | 0.864 | 0.153 |  | **AutoSKLearn-m** | 0.861 | 0.870 | 0.144 |
| **AutoSKLearn-v** | 0.853 | 0.862 | 0.151 | **60 Min** | **AutoSKLearn-v** | 0.861 | 0.870 | 0.142 |
| **AutoSKLearn** | **0.859** | 0.868 | 0.152 |  | **AutoSKLearn** | 0.868 | 0.877 | 0.137 |
| **SmartML-m** | 0.806 | 0.799 | 0.212 |  | **SmartML-m** | 0.790 | 0.816 | 0.194 |
| **SmartML-e** | 0.711 | 0.831 | 0.176 |  | **SmartML-e** | 0.726 | 0.832 | 0.0172 |
| **TPOT** | 0.383 | **0.890** | **0.121** |  | **TPOT** | 0.620 | 0.885 | 0.137 |
| **30 Min** | **ATM** | 0.665 | **0.899** | **0.121** |  | **ATM** | 0.768 | **0.893** | **0.124** |
| **AutoWeka** | 0.747 | 0.839 | 0.166 |  | **AutoWeka** | 0.771 | 0.838 | 0.166 |
| **Recipe** | 0.516 | 0.748 | 0.254 |  | **Recipe** | 0.645 | 0.759 | 0.248 |
| **AutoSKLearn-e** | **0.866** | 0.875 | 0.141 |  | **AutoSKLearn-e** | 0.874 | 0.883 | 0.132 |
| **AutoSKLearn-m** | 0.859 | 0.868 | 0.152 |  | **AutoSKLearn-m** | 0.864 | 0.873 | 0.141 |
| **AutoSKLearn-v** | 0.858 | 0.867 | 0.149 | **240 Min** | **AutoSKLearn-v** | 0.850 | 0.867 | 0.156 |
| **AutoSKLearn** | 0.862 | 0.871 | 0.148 |  | **AutoSKLearn** | **0.875** | 0.884 | 0.132 |
| **SmartML-m** | 0.804 | 0.808 | 0.199 |  | **SmartML-m** | 0.798 | 0.826 | 0.169 |
| **SmartML-e** | 0.727 | 0.838 | 0.159 |  | **SmartML-e** | 0.735 | 0.840 | 0.165 |
| **TPOT** | 0.518 | 0.878 | 0.144 |  | **TPOT** | 0.790 | 0.888 | 0.131 |

meta-learning (Section [5.2.3](#bookmark79)), and ensembling (Section [5.2.4](#bookmark80)) on the performance of the different AutoML frameworks across different time budgets. For each framework, the performance reported in each exper- iment is based on an average of 10 runs.

*5.2.1. Impact of time budget*

Tuning the time budget is a crucial and challenging task in AutoML, as it requires a balance between the available computational resources and the desired level of performance. It is a task that involves careful consideration of trade-offs between generalization and over-fitting of the AutoML frameworks. We investigate the impact of time budget on the performance of various AutoML frameworks, examining the speed at which they can generate ML pipelines and their ability to consistently improve performance given more time. We assess each framework’s performance on successful runs under four different time budgets: 10, 30, 60, and 240 min. [Table](#bookmark78) [4](#bookmark78) presents the mean (Mean) and standard deviation (SD) of performance for all successful runs at each time budget. Furthermore, we report the mean predictive performance weighted by the percentage of successful runs (*Meansucc*).

*Meansucc* = *Mean* ×  (1)

where *N* is the number of successful runs and *T* is the total number of runs.

The results show that for the 10 and 240 min budgets, AutoSK- learn and AutoSKlearn-e have comparable *Meansucc*, while AutoSKlearn-e has the highest *Meansucc* over the rest of time budgets. In contrast, Recipe achieves the lowest mean performance and Meansucc overall time budgets, as shown in [Table](#bookmark78) [4](#bookmark78). Notably, the performance of genetic-based tools, i.e., Recipe and TPOT, improves over time as the *Meansucc* values show. [G.18](#bookmark81), [G.19](#bookmark82), [G.20](#bookmark83), [G.21](#bookmark84), [G.22](#bookmark85), [G.23](#bookmark86), [G.24](#bookmark87), [G.25](#bookmark88), [G.26](#bookmark89), [G.27](#bookmark90) in [Appendix](#bookmark91) [G](#bookmark91) show the impact of increasing the time budget for each AutoML framework on 100 datasets.

Extended time budgets do not necessarily lead to better perfor- mance, in contrast to the prior assumptions, as shown in [Table](#bookmark92) [5](#bookmark92). We report the gain (*g*) or loss (*l*) in the predictive performance of the frameworks when increasing the time budget. The gain is measured by the mean and maximum predictive performance improvement over all improved/declined datasets. When increasing the time budget from 10 to 30 min, Recipe achieves the highest mean gain of 19.3 on

2 datasets, followed by SmartML-m, while AutoSKlearn comes in

the last place achieving a mean gain of 3.5 on 17 datasets. It is noticeable that Recipe has the smallest number of datasets that wit- nessed performance improvement and performance degradation when increasing the time budget. AutoSKlearn-v have the largest number of datasets that witnessed performance improvement when increas- ing the time budget from 30 to 60 min and from 60 to 240 min, while AutoSKlearn-e witnessed performance improvement across the largest number of datasets when increasing the time budget from 10 to 30 min. In contrast, ATM has the most significant number of datasets with performance degradation when increasing the time budget from 10 to 30 min and from 30 to 60 min.

The Wilcoxon signed-rank test is conducted to determine if the average performance difference when extending the time budget is statistically significant, as shown in [Table](#bookmark93) [6](#bookmark93). The impact of increasing the time budget varies from one framework to another. For example, AutoSKlearn-m, Recipe, ATM, SmartML-m and SmartML-e does not witness significant performance difference. While in most of the cases of AutoWeka, TPOT and all versions of AutoSKlearn except AutoSKlearn-m, the differences are statistically significant. These re- sults show that end-users should always carefully consider the trade-off between time budget and performance for the benchmark frameworks based on their specific goals.

*5.2.2. Impact of the size of search space*

Search space defines the structural paradigm that the different optimization methods can explore; thus, designing a good search space is a vital but challenging problem. [Fig.](#bookmark94) [6](#bookmark94) provides an overview of the most frequent ML models commonly used by the different AutoML frameworks. By analyzing the returned best-performing models, it is notable that there is no single ML algorithm that dominates all AutoML frameworks; however, it is apparent the tree-based models are the most frequent across all frameworks for all time budgets. For example, the returned pipelines by AutoWeka, AutoSKlearn-v, and SmartML- m show that *random forest* is the most frequently used classifier, as shown in [Figs.](#bookmark95) [6(a)](#bookmark95), [6(c)](#bookmark96), and [6(e)](#bookmark97). The most frequent classifier for AutoSKlearn-m, TPOT, and Recipe is *gradient boosting*, as shown in [Figs.](#bookmark98) [6(d)](#bookmark98), [6(f)](#bookmark99), and [6(g)](#bookmark100), respectively. To efficiently utilize the time budget, ATM limits its default search space to only three classifiers, namely, *k-nearest neighbors*, *decision tree*, and *logistic regression*, while *decision tree* is the most frequently used one, as shown in [Fig.](#bookmark101) [6(b)](#bookmark101).

Finding an optimal solution to the time-bounded optimization prob- lem of AutoML requires defining the underlying search space and

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

**Table 5**

Summary of the impact of increasing the time budget. Bold entries highlight the highest mean gain, highest maximum gain, smallest mean loss, smallest maximum loss, and maximum and minimum number of datasets with gain *>* 1 and loss *>*1, respectively.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Time Budget in minutes | Framework | Gain (*g*) | | #datasets with | | | Loss (*l*) | |
| Mean | Max | *g >* 1% | *g* ≈ 0% | *l >* 1% | Mean | Max |
| 10 → 30 | **ATM**  **AutoWeka Recipe**  **AutoSKLearn-e AutoSKLearn-m AutoSKLearn-v AutoSKLearn**  **SmartML-m SmartML-e TPOT** | 4.3 5.8  **19.3**  3.6 3.6 3.9 3.5 8.0 7.9 6.2 | 21.0 20.1 36.6 13.5 13.3 22.1 15.2 33.3  **85.2**  17.0 | 19  16  2  24  21  23  17  13  18  7 | 33  62  17  67  65  63  72  64  67  29 | 15  7  2  8  13  13  10  11  11  4 | −4.7 −3.8 −4.4 −2.7 −2.6 −3.1 −2.6 −3.6 −**6.3** −2.2 | −**21.4** −11.9 −6.2  −6.8 −10.7 −7.4 −4.8 −8.3 −16.9 −3.7 |
| 30 → 60 | **ATM**  **AutoWeka Recipe**  **AutoSKLearn-e AutoSKLearn-m AutoSKLearn-v AutoSKLearn**  **SmartML-m SmartML-e TPOT** | 5.0 9.1 4.7 4.9 4.9 4.1 4.1  **12.2**  6.5 3.7 | 16.5  **66.6**  17.2 19.6 23.4 14.2 32.0 40.0 18.2 8.7 | 15  14  6  16  16  23  22  10  21  6 | 34  61  58  66  67  62  65  73  54  42 | 20  13  3  17  16  14  12  6  20  8 | −6.7  −9.6  −**19.2** −2.2  −4.1 −4.6 −2.4 −6.2 −9.0 −2.8 | −28.6 −56.7 −29.1 −5.7  −13.3 −13.9 −6.8  −18.3 −**84.3** −7.7 |
| 60 → 240 | **ATM**  **AutoWeka Recipe**  **AutoSKLearn-e AutoSKLearn-m AutoSKLearn-v AutoSKLearn**  **SmartML-m SmartML-e TPOT** | 5.6 4.1 13.5 4.3 4.0 3.6 4.8  **10.6**  9.1 2.6 | 31.1 8.7 38.2 39.0 13.3 12.5 36.5  **59.6**  22.3 5.6 | 21  17  4  21  20  22  22  19  23  18 | 39  61  69  62  59  63  63  59  53  47 | 17  8  2  16  20  13  14  10  19  5 | −3.4  −3.8  −**20.5** −3.8  −2.7 −8.7 −3.2 −6.6 −6.9 −4.1 | −12.0 −11.5 −**40.0** −12.7 −6.0  −25.3 −9.9 −19.4 −18.8 −7.7 |

searching for well-performing ML pipelines as efficiently as possible. Often these search spaces are chosen arbitrarily without any validation, sometimes leading to bloated spaces and the inability to find optimal results ([Zöller & Huber](#bookmark25), [2021](#bookmark25)). In the following, we examine the impact of a budget allocation strategy as a complementary design decision for AutoML frameworks. The strategy is based on using a static portfo- lio ([Kotthoff,](#bookmark102) [2016](#bookmark102)) – a set of configurations that covers as many diverse datasets as possible and minimizes the risk of failure when facing a new task. So, we construct a portfolio consisting of the top three performing classifiers over the 100 datasets and supported by all AutoML frame- works. These classifiers are *support vector machine*, *random forest*, and *decision tree*. Then for a dataset at hand, all algorithms in this portfolio based on different hyperparameters are evaluated. For the AutoML frameworks included in this work that allows configuring the search space, ATM, AutoSKlearn, and TPOT, we compare the performance of using the full search space including all available classifiers (*FC*) to the performance when using the static portfolio (3*C*) on 30 min time budget, the results of which are summarized in [Fig.](#bookmark103) [7](#bookmark103).

For AutoSKlearn, the results show that the performance of the *FC* outperforms 3*C* on 28 datasets with an average predictive per- formance gain of 3.3%. However, the performance achieved using the 3*C* outperforms that achieved using the *FC* on 21 datasets by 5.9%, as shown in [Fig.](#bookmark104) [7(a)](#bookmark104). This performance discrepancy is attributed to the AutoML framework’s focus on tuning classifiers that have yielded good performance, thereby evaluating more hyperparameters of these classifiers. Hence, AutoML frameworks concentrate on promising re- gions in the search space while disregarding unimportant ones. The performance of both *FC* and 3*C* is comparable on 50 datasets, with pre- dictive performance differences of less than 1%. For TPOT, 23 datasets failed to run using the 3*C*, while 20 failed using the *FC*. Both search spaces failed to produce results for 12 datasets, as shown in [Fig.](#bookmark105) [7(b)](#bookmark105).

For successful runs, the *FC* outperformed the 3*C* on 21 datasets, with an average predictive performance improvement of 9.6%. In contrast, the performance of both search spaces was comparable on 18 datasets. Notably, the 3*C* search space achieved better performance than the *FC* search space on six datasets, with an average predictive performance difference of 8.8%. For ATM, the 3*C* outperformed the *FC* search space on 17 datasets, with an average predictive performance improvement of 4%. In contrast, the *FC* search space outperformed the 3*C* search space on 15 datasets, with an average performance improvement of 9.3%. Both search spaces achieved comparable performance on 22 datasets, as depicted in [Fig.](#bookmark106) [7(c)](#bookmark106). Notably, the *FC* failed to produce results for 19 datasets in which the 3*C* search space succeeded. In contrast, the 3*C* failed for ten datasets that the *FC* was successful. The Wilcoxon signed-rank test was conducted to determine if a statistically significant difference in performance exists between the *FC* and 3*C* across all datasets. For TPOT, the test results show that the difference in performance between the two search spaces is statistically significant with more than 95% level of confidence (p-value=0.003). However, no statistically significant difference in performance exists between the two search spaces for AutoSKlearn and ATM.

*5.2.3. Impact of meta-learning*

One way to define meta-learning is the process of learning from pre- vious experience gained during applying various learning algorithms on different ML tasks, reducing the time needed to learn new tasks ([Van-](#bookmark107) [schoren](#bookmark107), [2018](#bookmark107)). In the following, we study the impact of meta-learning on the performance of AutoML frameworks. The only framework that supports configuring meta-learning is AutoSKlearn. Furthermore, we investigate the relationship between the characteristics of the different datasets and the improvement caused by employing the vanilla version or the meta-learning version of the AutoSKlearn.

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

**Table 6**

Wilcoxon test p-values for all the AutoML frameworks over different time budgets. Bold entries highlight significant difference.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Framework | Time budget 1 | Time budget 2 | Avg. acc. diff | *P* value | Framework | Time budget 1 | Time budget 2 | Avg. acc. diff | *P* value |
| **AutoWeka** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.008 0.003 0.000 0.009 0.005 0.005 | **0.016 0.034**  0.885  **0.000 0.039 0.042** | **AutoSKLearn** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.003 0.010 0.007 0.016 0.013 0.006 | 0.338  **0.001 0.034 0.004 0.018**  0.129 |
| **TPOT** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.009 0.008 0.001 0.013 0.006 0.004 | 0.117 0.388 0.428  **0.016 0.035 0.008** | **AutoSKLearn-v** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.005 0.008 0.003 0.005 0.000 -0.003 | 0.175  **0.001**  0.088  **0.000 0.040**  0.099 |
| **Recipe** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.014 -0.002 -0.004 0.023 0.003 0.002 | 0.866 0.955 0.535 0.093 0.067 0.345 | **AutoSKLearn-e** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.008 0.012 0.004 0.015 0.007 0.003 | **0.000 0.000**  0.904  **0.000 0.038**  0.291 |
| **ATM** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.001  −0.007 −0.008 0.003  −0.001 0.008 | 0.583 0.254 0.499 0.585 0.799 0.394 | **AutoSKLearn-m** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.004 0.006 0.002 0.009 0.004 0.003 | 0.156 0.105 0.873 0.210 0.920 0.660 |
| **SmartML-m** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.007 0.009 0.009 0.026 0.025 0.015 | 0.636 0.832 0.597 0.121  **0.050**  0.071 | **SmartML-e** | 30  60  60  240  240  240 | 10  10  30  10  30  60 | 0.008  0.003  −0.004 0.011  0.004 0.008 | 0.521 0.589 0.672 0.092 0.182 0.305 |

AutoSKlearn applies a meta-learning mechanism based on a knowledge base storing the meta-features of datasets as well as the best-performing pipelines on these datasets. AutoSKlearn uses 38 meta-features, including statistical, information-theoretic and simple meta-features. In an offline phase, the meta-features and the empir- ically best-performing pipelines are stored for each dataset in their repository (140 datasets from the OpenML repository). In an online phase, for any new dataset, the framework extracts the meta-features of the new dataset and searches for the most similar datasets in the knowledge base. It returns 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 optimization process. To assess the impact of the meta-learning mechanism, we compare the per- formance of AutoSKlearn-vand AutoSKlearn-mon 100 datasets across different time budgets, as shown in [Fig.](#bookmark108) [8](#bookmark108). The results show that using meta-learning is not necessarily associated with performance improvement. On average, the performance of the vanilla and the meta- learning versions is very comparable on the 4 time budgets. In particu- lar, both versions perform similarly on 64, 55, 65, and 69 datasets for 10 min, 30 min, 60 min and 240 min, respectively. [Table](#bookmark109) [7](#bookmark109) summarizes the performance of both of AutoSKlearn-m and AutoSKlearn- v, in addition to the number of datasets achieved improvement in performance by employing AutoSKlearn-m over AutoSKlearn-v on different time budgets. The improvement achieved by employing the meta-learning version decreases for extended time budgets. For exam- ple, the number of datasets that achieved performance improvement by using meta-learning dropped from 28 for the 30 min budget to 14 for the 240 min budget, as shown in [Table](#bookmark109) [7](#bookmark109). We use Wilcoxon statistical test to assess the significance of the performance difference between the vanilla version and the meta-learning version. The results show that the impact of the meta-learning is statistically significant only for the tiniest time budget of 10 min with more than 95% level of confidence (pvalue=0.004).

In the following, we explore the relationship between the charac- teristics of datasets and the improvement achieved by utilizing the meta-learning version of AutoSKlearn over different time budgets. We train a model that takes as input the meta-features of datasets, i.e., their characteristics, and predicts whether meta-learning can im- prove the performance. To develop this model, we label each dataset as Class 1 if utilizing meta-learning improves performance over the vanilla version and Class 0 otherwise. We implement a total of 42 meta-features from the literature, including simple, information- theoretic, and statistical meta-features ([Kalousis](#bookmark110), [2002](#bookmark110); [Mitchell et al.,](#bookmark111) [1990](#bookmark111)), such as statistics about the number of data points, features, and classes, as well as data skewness and entropy of the targets. All meta-features are listed in [Appendix](#bookmark112) [E](#bookmark112), [Table](#bookmark113) [E.12](#bookmark113). Using the extracted information from the knowledge base, for our 100 datasets, we fit a shallow decision tree of depth 4 using the meta-feature variables as pre- dictors. We considered decision tree classifier due to its interpretable nature, that allows rules to be derived from a root–leaf path in the tree. Given a new dataset, we compute its meta-features and use the decision tree model to recommend whether meta-learning is likely to improve performance or not. Our model achieves the following performance metrics: Recall = 0.85 and F1 Score = 0.85. Rules for Class 1 and Class 0 can be represented as follows, where ∧ is the logical AND:

• **R1:** *min*() *>* 0*.*5 ⟹ Class1

• **R2:** *min*() *<* 0*.*27∧ *noise*-*signal ratio >* 8*.*57∧*p <* 845 ⟹ Class1

• **R3:** 0*.*1 *< min*() *<* 0*.*27 ∧ *noise*-*signal ratio <* 8*.*57 ⟹ Class1

• **R4:** 0*.*27 *< min*() *<* 0*.*5 ⟹ Class0

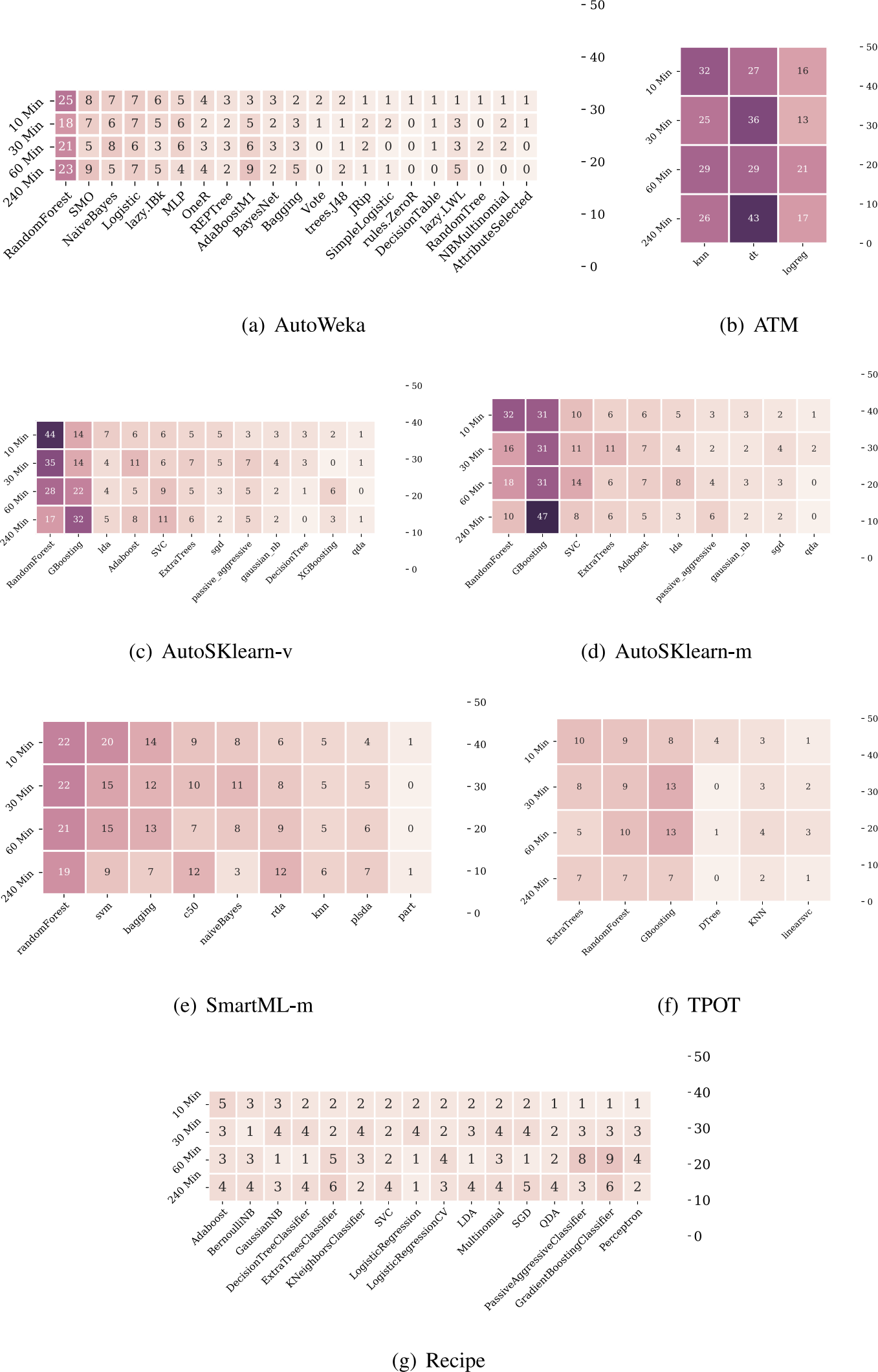
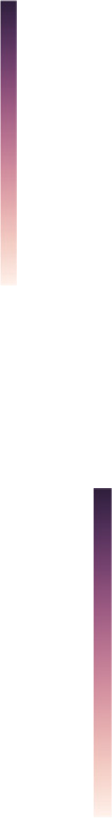
• **R5:** *min*() *<* 0*.*27∧ *noise*-*signal ratio >* 8*.*57∧*p >* 845 ⟹ Class0

• **R6:** *min* *<* 0*.*1 ∧ *noise*-*signal ratio <* 8*.*57 ⟹ Class0

It is clear from the extracted rules that the number of features *p*, the

percentage of the minority class to the number of instances (*min*()),

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*



**Fig. 6.** The frequency of using different machine learning models by the different AutoML frameworks.

**Table 7**

The performance of AutoSklearn-v and AutoSklearn-m and the gain in performance achieved by employing the meta-learning on 100 datasets over different time budgets.

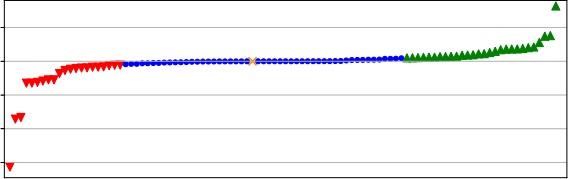
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Time budget | Framework | Predictive performance | | Performance gain | | | #datasets with gain *>* 1% |
| Mean | SD | Min | Mean | Max |
| 10 | AutoSKlearn-m | 0.864 | 0.153 | 1.1% | 2.9% | 7.1% | 28 |
| AutoSKlearn-v | 0.862 | 0.151 | 1.1% | 5.4% | 15.5% | 12 |
| 30 | AutoSKlearn-m | 0.868 | 0.152 | 1.1% | 3.1% | 20.6% | 28 |
| AutoSKlearn-v | 0.8867 | 0.149 | 1.1% | 5.1% | 16.7% | 15 |
| 60 | AutoSKlearn-m | 0.870 | 0.144 | 1.1% | 3.3% | 18.8% | 20 |
| AutoSKlearn-v | 0.870 | 0.142 | 1.1% | 4.3% | 14.0% | 17 |
| 240 | AutoSKlearn-m | 0.873 | 0.141 | 1.1% | 7.7% | 31.6% | 14 |
| AutoSKlearn-v | 0.867 | 0.156 | 1.1% | 2.7% | 8.4% | 20 |

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

performance Difference between fc and 3c for 30 minutes - AutoskLearn

M

I



 Negative  same  positive  Failed



2 3

2 3 4 5 7 8

Data set

(a) Autosklearn

performance Difference between fc and 3c for 30 minutes - TPOT





5 -





5 -

|  |
| --- |
|  |
|  |



2 3 4 5 7 8

Data set

(b) TPOT

performance Difference between fc and 3c for 30 minutes - ATM

  Negative  same  positive  Failed

5

5

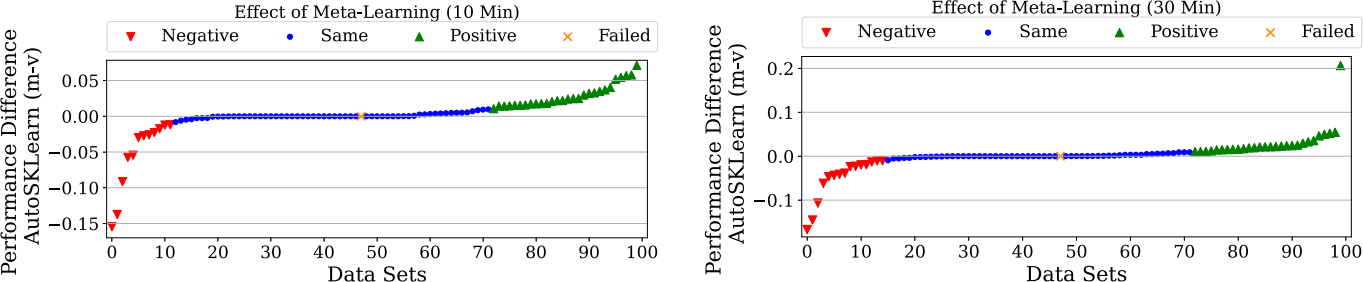
|  |  |
| --- | --- |
|  | |
| △ |  |
|  | |
|  | |
|  | |

2 3 4 5 7 8

Data set

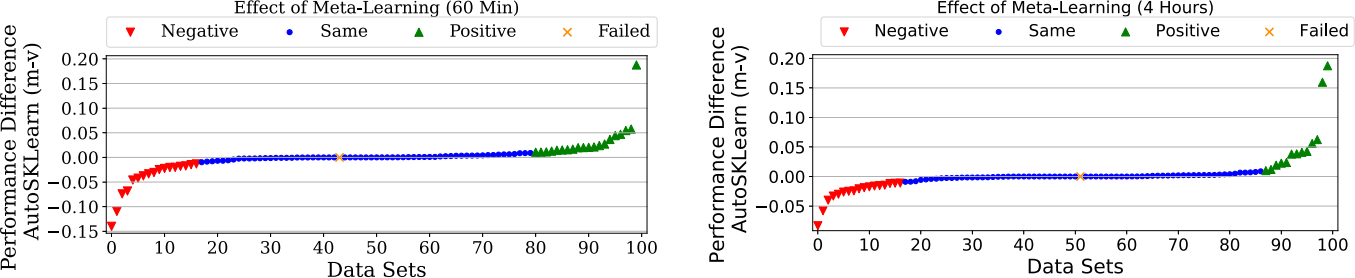
(C) ATM

**Fig. 7.** The impact of using a static portfolio on each AutoML framework. Green markers represent better performance with *FC* search space, blue markers represent comparable performance with a difference less than 1%, red markers represent better performance with 3*C* search space, yellow markers on the left represent failed runs with *FC* but successful with 3*C*, yellow markers on the right represent failed runs with 3*C* but successful with *FC*, and yellow markers in the middle represent failed runs with both *FC* and 3*C*.













**Fig. 8.** The impact of meta-learning over all time budgets. Green markers represent better performance with AutoSKlearn-m, blue markers represent comparable performance with a difference less than 1%, red markers represent better performance using AutoSKlearn-v, and yellow markers represent failed runs with both runs with both FC and 3C.

and noisiness of data (*noise*-*signal ratio*) are quite important features for the prediction.

*5.2.4. Impact of ensembling*

Ensembling ([Dietterich](#bookmark114), [2000](#bookmark114)) is the process of combining multiple ML base models for the same task to produce a better predictive model. These base models can be combined in different techniques, including simple voting (averaging), weighted voting, bagging, and boosting ([Dietterich](#bookmark114), [2000](#bookmark114)). In the following, we explore the impact of ensembling on the performance of the AutoML frameworks allowing enabling and disabling post-processing ensemble. Such frameworks include AutoSKlearn and SmartML-m. Furthermore, we investi- gate whether there is a relationship between the characteristics of the different datasets and the improvement caused by employing the vanilla version or the ensembling version of the AutoML framework. During the optimization process of AutoSKlearn and SmartML, the frameworks store the generated models instead of just keeping the best- performing one. These models are used in a post-processing phase to construct an ensemble model. This automatic ensemble construction

avoids relying on a single hyperparameter setting which makes the generated model more robust to overfitting. AutoSKlearn uses the ensemble selection methodology introduced by [Caruana et al.](#bookmark115) ([2004](#bookmark115)), while SmartML uses majority voting ([Lam & Suen](#bookmark116), [1997](#bookmark116)). Ensemble selection is a greedy technique that starts with an empty ensemble and iteratively adds base models to the ensemble in a way that maximizes the validation performance. The technique uses uniform weights; how- ever, it allows repetitions. Majority voting is considered the simplest scheme. It adheres to democratic principles, i.e., the class with the most votes wins. We kept the default setting of AutoSKlearn and SmartML using 50 and 5 base models in the ensemble, respectively.

To assess the impact of the ensembling, we compare the mean performance of vanilla/base version of each of AutoSKlearn and SmartML to their ensembling versions across different time budgets, as shown in [Table](#bookmark117) [8](#bookmark117). More detailed performance comparisons over all datasets across all time budgets are given in [Figs.](#bookmark118) [F.16](#bookmark118) and [F.17](#bookmark119) in [Appendix](#bookmark120) [F](#bookmark120).

**AutoSKlearn**: The results show that ensembling does not always contribute to better performance than the vanilla version. However, it

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

**Table 8**

Performance comparison between vanilla/base version vs ensembling version of AutoSKlearn and SmartML different time budgets.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Time budget | Framework | Predictive performance | | Performance gain | | | #datasets with gain *>* 1% |
| Mean | SD | Min | Mean | Max |
| 10 | AutoSKlearn-e AutoSKlearn-v | 0.868 0.868 | 0.145 0.151 | 1.1% 1.1% | 3.9% 3.2% | 16.7% 8.9% | 24  14 |
| SmartML-e SmartML | 0.831 0.176 | 0.176 0.176 | 1.1% 1.1% | 12.6% 10.2% | 64.2% 36.1% | 30  14 |
| 30 | AutoSKlearn-e AutoSKlearn-v | 0.875 0.867 | 0.141 0.149 | 1.1% 1.1% | 3.2% 2.9% | 13.9% 11.1% | 32  11 |
| SmartML-e SmartML | 0.838 0.838 | 0.159 0.199 | 1.1% 1.1% | 12.5% 10.6% | 73.2% 39.4% | 33  16 |
| 60 | AutoSKlearn-e AutoSKlearn-v | 0.879 0.870 | 0.138 0.142 | 1.1% 1.1% | 4.7% 2.6% | 12.7% 6.1% | 25  13 |
| SmartML-e SmartML | 0.832 0.816 | 0.172 0.194 | 1.1% 1.1% | 11.2% 10.5% | 55.8% 31.1% | 28  18 |
| 240 | AutoSKlearn-e AutoSKlearn-v | 0.883 0.867 | 0.132 0.156 | 1.1% 1.1% | 8.0% 3.1% | 69.7% 8.4% | 24  14 |
| SmartML-e SmartML | 0.842 0.826 | 0.165 0.169 | 1.1% 1.1% | 10.2% 11.9% | 34.5% 37.2% | 28  16 |

achieved mean improvement of 3.9%, 3.2%, 4.7%, and 8.0% on 24, 32, 25 and 24 datasets over 10, 30, 60, and 240 min budgets, respectively, as shown in [Table](#bookmark117) [8](#bookmark117). We use Wilcoxon statistical test to assess the significance of the performance difference between AutoSKlearn- e and AutoSKlearn-v. The results show that ensembling enhances the performance with a statistically significant gain of more than 95% level of confidence (*p* value *<* 0.05) on the 4 time budgets. The level of confidence is almost 99% over all the time budgets combined.

**SmartML**: SmartML-e slightly improved the performance over the SmartML-m by average performance of 12.6%, 12.5%, 11.2%, and 10.2% on 30, 33, 28, and 28 datasets for 10, 30, 60, and 240 min time budgets, respectively, as shown in [Table](#bookmark117) [8](#bookmark117). We also use Wilcoxon statistical test to assess the significance of the performance differ- ence between the base (meta-learning) and the ensembling versions of SmartML. The results show that the ensembling version enhance the performance with a statistically significant gain of more than 95% level of confidence (*p* value *<* 0.05) on the 4 time budgets.

In the following, we explore the relationship between the charac- teristics of the datasets and the improvement achieved by utilizing the ensembling version of AutoSKlearn over different time budgets. To this end, we followed the same approach in Section [5.2.3](#bookmark79) and trained a decision tree of depth 3 that takes the meta-features of 100 datasets as input and provides prediction to whether using ensembling can improve the performance (Class 1) or not (Class 0). So,given a new dataset, we compute its meta-features and use the decision tree model to recommend whether ensembling will likely improve performance. Our model for AutoSKlearn has achieved the following performance: Recall = 0.70 and F1 Score = 0.70. AutoSKlearn rules for Class 1 and Class 0 can be represented as follows:

• **R1:** ( ) *>* 0*.* 13 ∧ ( ) *>* 0*.*27 ∧ *max*( ) *>* 0*.*98 ⟹ Class1

• **R2:** ( ) *>* 0*.* 13 ∧ ( ) ≤ 0*.*27 ∧ *min*( ) *>* 0*.*44 ⟹ Class1

• **R3:** ( ) ≤ 0*.* 13 ∧ ( *i*) *>* 3*.* 11 ∧ (*n* ) *>* 0*.*03 ⟹ Class1 *p*

• **R4:** ( ) ≤ 0*.* 13 ∧ ( *i*) ≤ 3*.* 11 ∧ *min*(*Mutual inform.*) *>* 0*.*03 ⟹ Class1

• **R5:** ( ) *>* 0*.* 13 ∧ ( ) *>* 0*.*27 ∧ *max*( ) ≤ 0*.*98 ⟹ Class0

• **R6:** ( ) *>* 0*.* 13 ∧ ( ) ≤ 0*.*27 ∧ *min*( ) ≤ 0*.*44 ⟹ Class0

• **R7:** ( ) ≤ 0*.* 13 ∧ ( *i*) *>* 3*.* 11 ∧ (*n* ) ≤ 0*.*03 ⟹ Class0 *p*

• **R8:** ( ) ≤ 0*.* 13 ∧ ( *i*) ≤ 3*.* 11 ∧ *min*(*Mutual inform.*) ≤ 0*.*03 ⟹ Class0

Clearly, the following features are important to the prediction of the model; the mean of the pairwise correlation between features ( ( )),

standard deviation of the pairwise correlation between features ( ( )), maximum of the pairwise correlation between features (max( )), mini- mum of the pairwise correlation between features (min( )),the standard deviation of the ratio between number of instances and the number

of features ( (*n* )), the minimum of the mutual information between *p*

features and class (*Mutual inform*.), and the mean of the unique categorical values of features ( ( *i*)).

For SmartML, we trained multiple models; however, none of the models could capture the relation of the meta-features and the perfor- mance improvement caused by employing ensembling.

**6. Discussion and future direction**

The global average performance, weighted by the percentage of successful runs, shows that Auto-SKlearn-e and AutoSKlearn achieve the highest performance, while Recipe comes in the last place. Overall, AutoSklearn achieves the highest number of suc- cessful runs across different time budgets and witnessed performance improvement over the most significant number of datasets when in- creasing the time budget. Our analysis reveals that the impact of meta- learning declines over longer time budgets (i.e., 60 mins, 240 mins). In contrast, ensembling achieves consistent performance improvement across all time budgets. For AutoSKlearn, the analysis reveals a relationship between the characteristics of the datasets (e.g., number of features, noisiness of data, mutual information between features and class) and the improvement achieved by utilizing meta-learning or ensembling. Generally, AutoML frameworks considered in this work build pipelines with an average length of 2. TPOT yields the shortest pipelines with an average length of 1.5. A possible explanation could be that TPOT generates pipelines that optimize both the pipelines’ performance and complexity. Additionally, AutoSKlearn, ATM, and TPOT achieve the highest performance on multi-class classification tasks. For datasets with a large number of instances and a small number of features, ATM is a clear winner.

Our comprehensive evaluation, guided by a set of well-defined metrics, offers a comprehensive perspective on AutoML framework performance. Notably, training efficiency emerges as a vital factor, particularly in resource-constrained environments and scenarios that demand rapid model development. Here, ATM and Recipe excel, mak- ing them compelling choices for applications characterized by limited computational resources or a need for swift model iteration. In terms of inference times, SmartML-m and Recipe emerge as standout perform- ers, with better efficiency. Moreover, Recipe, SmartML-m, and ATM



*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

demonstrate exceptional model stability, underscoring their reliability.

These attributes make them particularly well-suited for applications where consistent model performance is paramount. Furthermore, the robustness to noise metric reveals valuable insights. AutoSKLearn-e and AutoSKLearn-m exhibit high levels of robustness, making them well-suited for scenarios dealing with unpredictable, noisy real-world data.

For some datasets, the performance of the different versions of AutoSKlearn varies significantly across different iterations. These datasets are characterized by having far fewer instances than features. Analyzing the pipelines of the different versions of AutoSKlearn on these datasets across multiple iterations shows that data preprocessing component is responsible for the significant performance variance be- tween the different pipelines. For example, the performance difference between AutoSKlearn-v and AutoSKlearn-m on phpdo58hj varies significantly between 6% to 13% across different iterations. The two generated pipelines for Auto- SKlearn-vand AutoSKlearn- m used the same model (LDA) with the same set of hyperparameters but different preprocessors. For large datasets, meta-learning shows significant performance improvement. For example, AutoSKlearn- m achieves significantly better performance than AutoSKlearn-v on CovPokElec. A possible explanation is that meta-learning warm-starts the optimization process and increases the chances of finding a well- performing configuration in the limited attempts during the defined time budget.

Specifying the time budget needs to be considered carefully as sig- nificantly increasing the time budget for the search process (e.g., from 60 min to 240 min) may not significantly improve the predictive performance. This decision varies from one scenario/application to another. For some applications, spending a long time to achieve an additional predictive performance of 1% could be crucial while less important for other applications. However, more extended time budgets may lead to over-fitting. Carefully selecting a small search space with few top-performing classifiers can lead to a comparable performance with a search space that includes many classifiers, which is the case for AutoSKlearn and ATM frameworks.

Intuitively, an extensive systematic search for a well-performing machine learning pipeline should bear a high risk of over-fitting, and previous AutoML frameworks have confirmed this intuition ([Thornton](#bookmark121) [et al.](#bookmark121), [2013](#bookmark121)). AutoML tools are on the right extreme of the bias– variance spectrum as they choose among all learners and even construct new and arbitrary large ones using ensemble methods ([Mohr et al.](#bookmark122), [2018](#bookmark122)). Notably, SmartML and AutoWeka witnessed performance degradation when increasing the time budget from 30 to 60 min. One possible explanation is that the data available for the search process is not sufficiently substantial and representative of ‘‘real’’ data. Hence, the danger of over-fitting is higher than for basic learning algorithms. This insight calls for developing novel and more efficient mechanisms to prevent over-fitting.

While AutoML frameworks optimize predictive performance, many exceed the specified time budget by more than 10%. This violation of the time constraints caused many runs to be terminated and con- sidered as failed. This problem is observed in all frameworks except for AutoSKLearn, which calls for a robust implementation and careful consideration of the time constraint.

Most of the current work on AutoML considered automating the preprocessing, algorithm selection and hyperparameter tuning while ignoring the feature engineering part. In practice, the feature engineer- ing part consumes most of the Engineer’s time to build ML pipelines and significantly affects the performance. The proper feature engi- neering phase could turn the feature space into a linearly separable space, so even naive classifiers could achieve relatively high predictive performance. On the other hand, skipping this phase or using the wrong feature engineering preprocessors makes it harder to achieve relatively high predictive performance, even for the most efficient classifiers. Hence, further research in this area can improve the overall performance of the resulting AutoML pipelines.

**7. Conclusion**

In this paper, we present a comprehensive evaluation and compar- ison of the performance characteristics of six AutoML frameworks on 100 datasets from OpenML. Our analysis reveals that no single winning framework outperforms others over all time budgets. Across various evaluations, AutoSklearn, ATM, and TPOT are the top-performing frameworks. The results also show that genetic-based frameworks (TPOT and Recipe) have high frequent failure rates for short time budgets while their success rates are steadily increasing as the time budget increases. We also find that meta-learning has a significant impact on small-time budgets, and such impact declines as the time budget increases. In contrast, ensembling consistently improves per- formance significantly across all time budgets. Furthermore, carefully selecting a small search space with few top-performing classifiers can lead to a comparable performance with a search space that includes many classifiers. Furthermore, increasing the time budget does not necessarily improve predictive performance. We believe that the results of our analysis are beneficial for guiding and improving the design process of future AutoML techniques.

**CRediT authorship contribution statement**

**Hassan Eldeeb:** Conceptualization, Methodology, Software, Valida- tion, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Mohamed Maher:** Software, Investi- gation, Data curation, Writing – review & editing. **Radwa Elshawi:** Validation, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Sherif Sakr:** Conceptualization, Methodology, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

**Declaration of competing interest**

The authors declare that they have no known competing finan- cial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Data availability**

The datasets generated during and/or analyzed during the current study are available in the AutoMLBench repository, [https://datasystem](https://datasystemsgrouput.github.io/AutoMLBench/datasets) [sgrouput.github.io/AutoMLBench/datasets](https://datasystemsgrouput.github.io/AutoMLBench/datasets).

**Acknowledgments**

We would like to acknowledge support for this project. This work was supported by European Social Fund via ‘‘ICT programme measure’’ . The authors would like to thank the students Oleh Matsuk, Abdelrah- man Aldallal for their involvement in some of the experiments of this work.

**Appendix A. Evaluated datasets**

[Table](#bookmark47) [A.9](#bookmark47) shows the datasets used in evaluating all the AutoML frameworks included in this work.

**Appendix B. Framework and source code**

[Table](#bookmark49) [B.10](#bookmark49) lists the Github repositories of all the open-source Au- toML frameworks considered in this work. Some frameworks are still under active development and may differ from the evaluated versions.

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

**Table A.9**

List of all tested datasets including information about (abbreviated) name and OpenML id for each data set together with the number of classes, the number of features, the number of instances, how many values are missing in total (Missing values), number of categorical features per sample, and the class entropy.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Dataset Name (openml id) | Nr features | Nr instances | Nr classes | Nr missing values | Nr categorical features | Class entropy |
| AirlinesCodrnaAdult (1240) | 30 | 1 076 790 | 2 | 11 896 | 1 | 1,00 |
| Amazon (1457) | 10 001 | 1500 | 50 | 0 | 0 | 0,93 |
| analcatdata\_authorship (458) | 71 | 841 | 4 | 0 | 0 | 0,99 |
| AP\_Breast\_Lung (1150) | 10 937 | 470 | 2 | 0 | 0 | 0,99 |
| AP\_Omentum\_Ovary (1156) | 10 937 | 275 | 2 | 0 | 1 | 0,93 |
| AP\_Prostate\_Ovary (1152) | 10 937 | 267 | 2 | 0 | 0 | 2,18 |
| arrhythmia (1017) | 263 | 452 | 2 | 0 | 1 | 1,58 |
| audiology (999) | 70 | 226 | 2 | 0 | 1 | 3,70 |
| avila-tr (42932) | 11 | 20 867 | 12 | 114 | 10 | 2,27 |
| churn (40701) | 21 | 5000 | 2 | 0 | 1 | 1,00 |
| cifar-10 (40927) | 3073 | 60 000 | 10 | 0 | 70 | 0,81 |
| connect-4 (1591) | 43 | 67 557 | 3 | 0 | 1 | 0,82 |
| CovPokElec (149) | 65 | 1 455 525 | 10 | 0 | 1 | 0,86 |
| dataset\_183\_adult (179) | 15 | 48 842 | 2 | 0 | 0 | 2,21 |
| dataset\_185\_yeast (181) | 9 | 1484 | 10 | 0 | 1 | 2,19 |
| dataset\_186\_satimage (182) | 37 | 6430 | 6 | 1668 | 3 | 1,00 |
| dataset\_187\_abalone (183) | 9 | 4177 | 28 | 0 | 1 | 0,94 |
| dataset\_189\_baseball (185) | 18 | 1340 | 3 | 0 | 0 | 3,32 |
| dataset\_194\_eucalyptus (188) | 20 | 736 | 5 | 816 | 1 | 0,99 |
| dataset\_24\_mushroom (24) | 22 | 8124 | 2 | 0 | 1 | 0,84 |
| dataset\_26\_nursery (26) | 9 | 12 960 | 5 | 0 | 0 | 4,20 |
| dataset\_28\_optdigits (28) | 63 | 5620 | 10 | 0 | 0 | 4,28 |
| dataset\_31\_credit-g (31) | 21 | 1000 | 2 | 0 | 1 | 2,58 |
| dataset\_36\_segment (36) | 19 | 2310 | 7 | 0 | 36 | 3,84 |
| dataset\_39\_ecoli (39) | 8 | 336 | 8 | 32 | 1 | 0,93 |
| dataset\_40\_sonar (40) | 61 | 208 | 2 | 896 | 6 | 2,26 |
| dataset\_42\_soybean (42) | 36 | 683 | 19 | 0 | 1 | 1,79 |
| dataset\_44\_spambase (44) | 58 | 4601 | 2 | 0 | 1 | 2,00 |
| dataset\_54\_vehicle (54) | 19 | 846 | 4 | 0 | 4 | 0,44 |
| dataset\_59\_ionosphere (59) | 34 | 351 | 2 | 0 | 14 | 0,88 |
| dataset 6 letter (6) | 17 | 20 000 | 26 | 0 | 0 | 4,65 |
| dataset\_60\_waveform-5000 (60) | 41 | 5000 | 3 | 0 | 1 | 0,83 |
| dataset\_61\_iris (61) | 5 | 150 | 3 | 0 | 0 | 0,92 |
| dataset 9 autos (9) | 26 | 205 | 6 | 2792 | 4 | 2,99 |
| devnagari (40923) | 785 | 92 000 | 46 | 0 | 0 | 3,17 |
| electricity-normalized (151) | 9 | 45 312 | 2 | 0 | 0 | 1,00 |
| eye\_movements (1044) | 28 | 10 936 | 3 | 40 | 2 | 0,54 |
| GCM (1106) | 16 064 | 190 | 14 | 0 | 1 | 2,49 |
| gina\_agnostic (1038) | 971 | 3468 | 2 | 0 | 1 | 5,64 |
| hiva\_agnostic (1039) | 1618 | 4229 | 2 | 0 | 0 | 3,32 |
| ipums\_la\_99-small (378) | 60 | 8844 | 9 | 0 | 0 | 6,64 |
| jm1 (1053) | 22 | 10 885 | 2 | 0 | 0 | 1,71 |
| jungle\_chess\_2pcs (40997) | 45 | 4704 | 3 | 0 | 0 | 6,64 |
| KDDCup99 (1113) | 40 | 494 020 | 23 | 0 | 0 | 6,64 |
| kin8 nm (189) | 9 | 8192 | 2 | 0 | 1 | 2,41 |
| leukemia (1104) | 7130 | 72 | 2 | 0 | 1 | 0,47 |
| lymphoma\_2classes (1101) | 4027 | 45 | 2 | 0 | 1 | 2,81 |
| MagicTelescope (1120) | 11 | 19 020 | 2 | 0 | 0 | 0,34 |
| mfeat-pixel (20) | 241 | 2000 | 2 | 0 | 0 | 1,00 |
| mnist\_784 (554) | 720 | 70 000 | 10 | 0 | 0 | 1,53 |
| openml\_phpJNxH0q (15) | 10 | 699 | 2 | 0 | 0 | 1,00 |
| page-blocks (30) | 11 | 5473 | 2 | 0 | 1 | 3,60 |
| php0FyS2T (1492) | 65 | 1600 | 100 | 0 | 0 | 0,22 |
| php3CTpvq (1509) | 5 | 149 332 | 22 | 0 | 0 | 0,97 |
| php5OMDBD (40971) | 23 | 1000 | 30 | 0 | 6 | 1,16 |
| php5s7Ep8 (40982) | 28 | 1941 | 7 | 0 | 0 | 1,86 |
| php7KLval (1547) | 21 | 1000 | 2 | 0 | 0 | 0,59 |
| phpB0xrNj (300) | 618 | 7797 | 26 | 0 | 0 | 1,58  (*continued on next page*) |

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

**Table A.9** (*continued*).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| phpbL6t4U (1476) | 129 | 13 910 | 6 | 0 | 1 | 0,11 |
| phpchCuL5 (40966) | 81 | 1080 | 8 | 0 | 0 | 1,71 |
| phpCsX3fx (1491) | 65 | 1600 | 100 | 0 | 1 | 0,48 |
| phpdo58hj (1562) | 4703 | 64 | 2 | 0 | 0 | 3,32 |
| phpdReP6S (1487) | 73 | 2534 | 2 | 0 | 0 | 2,30 |
| phpEZ030X (1561) | 3722 | 64 | 2 | 0 | 0 | 2,48 |
| phpfLuQE4 (1485) | 501 | 2600 | 2 | 0 | 0 | 1,00 |
| phpfrJpBS (1568) | 9 | 12 958 | 4 | 0 | 0 | 1,00 |
| phpGReJjU (40985) | 4 | 45 781 | 20 | 0 | 1 | 4,70 |
| phpGUrE90 (1494) | 42 | 1055 | 2 | 0 | 22 | 1,00 |
| phphQEck0 (1502) | 4 | 245 057 | 2 | 0 | 1 | 1,00 |
| phpHyLSNF (1515) | 1083 | 571 | 20 | 0 | 26 | 0,48 |
| phpkIxskf (1461) | 17 | 45 211 | 2 | 0 | 1 | 2,19 |
| phpmcGu2X (1468) | 857 | 1080 | 9 | 0 | 1 | 0,94 |
| phpmPOD5A (4135) | 10 | 32 769 | 2 | 50 | 0 | 0,71 |
| phpn1jVwe (310) | 7 | 11 183 | 2 | 0 | 0 | 1,57 |
| phpN4gaxw (1477) | 130 | 13 910 | 6 | 0 | 0 | 0,16 |
| phpNevWWL (40477) | 27 | 2800 | 5 | 0 | 0 | 1,71 |
| phpoOxxNn (1493) | 65 | 1599 | 100 | 0 | 9 | 1,72 |
| phpoW7Dbi (1566) | 101 | 1212 | 2 | 0 | 0 | 2,55 |
| phpPbCMyg (1475) | 52 | 6118 | 6 | 0 | 0 | 2,55 |
| phprAeXmK (4535) | 42 | 299 285 | 2 | 0 | 1 | 0,94 |
| phpSZJq5T (1514) | 1088 | 360 | 10 | 0 | 1 | 4,70 |
| phptd5jYj (1501) | 37 | 5100 | 2 | 0 | 1 | 2,64 |
| phpTJRsqa (40498) | 257 | 1593 | 10 | 0 | 0 | 0,32 |
| phpvcoG8S (1169) | 12 | 4898 | 7 | 0 | 9 | 0,52 |
| phpVeNa5j (1497) | 8 | 539 383 | 2 | 0 | 1 | 0,98 |
| phpvtdNPU (1079) | 25 | 5456 | 4 | 0 | 0 | 4,25 |
| phpWfYmlu (1496) | 21 | 7400 | 2 | 0 | 9 | 0,79 |
| phpxijhaP (1507) | 22 278 | 95 | 5 | 0 | 0 | 0,96 |
| phpYLeydd (4538) | 21 | 7400 | 2 | 0 | 0 | 3,32 |
| phpZrCzJR (40900) | 33 | 9873 | 5 | 0 | 0 | 1,22 |
| pokerhand-normalized (155) | 11 | 829 201 | 10 | 0 | 0 | 3,32 |
| schizo (466) | 14 | 340 | 2 | 0 | 1 | 5,52 |
| shuttle (40685) | 10 | 58 000 | 7 | 0 | 0 | 3,99 |
| solar-flare\_1 (40686) | 13 | 315 | 5 | 0 | 0 | 0,74 |
| synthetic\_control (377) | 61 | 600 | 6 | 0 | 29 | 0,34 |
| tumors\_C (1107) | 7130 | 60 | 2 | 0 | 4 | 1,56 |
| umistfacescropped (41084) | 10 305 | 575 | 20 | 0 | 3 | 0,99 |
| vowel (307) | 14 | 990 | 2 | 0 | 0 | 1,42 |
| wine-quality-red (40691) | 12 | 1599 | 6 | 0 | 11 | 0,99 |
| aaaData\_for\_UCI\_named (43007) | 14 | 10 000 | 2 | 0 | 0 | 1,59 |

**Table B.10**

Source code repositories for all used AutoML frameworks.

|  |  |
| --- | --- |
| AutoML framework | Source code |
| AutoSKlearn | <https://automl.github.io/auto-sklearn/> |
| TPOT | <https://github.com/EpistasisLab/tpot> |
| ATM | <https://github.com/HDI-Project/ATM> |
| Recipe | <https://github.com/laic-ufmg/Recipe> |
| AutoWeka | <https://github.com/automl/AutoWeka> |
| SmartML | <https://github.com/DataSystemsGroupUT/SmartML> |

**Appendix C. Cut-off time budget**

We tested the cut-off timeouts of 4 and 8 h on 14 randomly selected datasets. [Table](#bookmark52) [C.11](#bookmark52) shows the mean performance difference between the 8 and 4 h (Avg. diff) over the 14 datasets. Additionally, we report the results of the Wilcoxon signed-rank test to determine if a statistically significant difference in performance exists between the AutoML frameworks over the two-time budgets.

**Appendix D. General performance evaluation**

[D.9](#bookmark59),[D.10](#bookmark60),[D.11](#bookmark61) show the average performance of all frameworks for time budgets 10, 30, and 60 min compared the average performance

**Table C.11**

Performance comparison between the 8 and 4 h budgets on 14 randomly selected datasets.

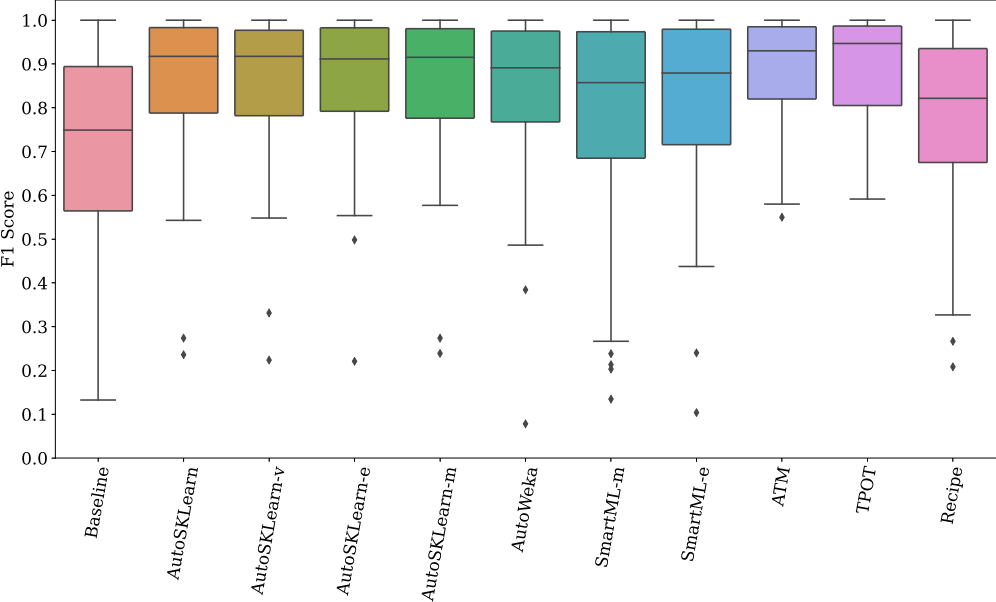
|  |  |  |
| --- | --- | --- |
| Framework | P value | Avg. diff |
| AutoSKlearn | 0.084 | −0.026 |
| AutoSKlearn-e | **0.039** | −0.025 |
| AutoSKlearn-m | 0.382 | −0.031 |
| AutoSKlearn-v | 0.272 | −0.008 |
| AutoWeka | 0.133 | −0.005 |
| Recipe | 0.480 | 0.007 |
| SmartML | 0.594 | −0.003 |
| SmartML-e | 0.753 | −0.009 |
| TPOT | 0.092 | −0.050 |

of the baseline. [4(b)](#bookmark70),[D.13](#bookmark66),[D.14](#bookmark67),[D.15](#bookmark68) show the AutoML frameworks’ average performance on subsets of the datasets with special characteris- tics, namely binary-class, large number of features and instances, small number of features and instances, and small number of features and large number of instances.

**Appendix E. Impact of meta learning**

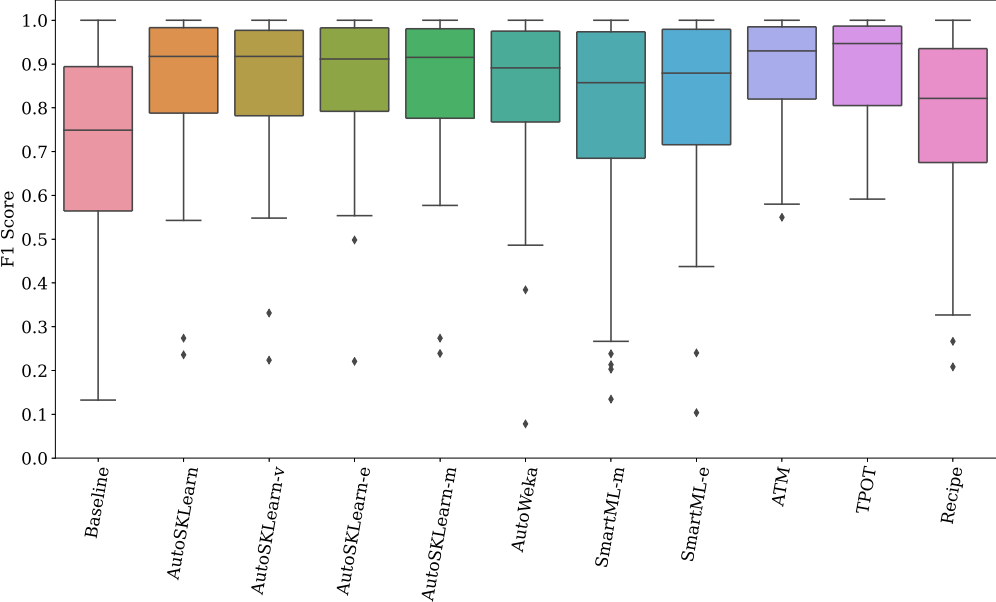
[Table](#bookmark113) [E.12](#bookmark113) lists a total of 42 meta-features including simple, infor- mation-theoretic and statistical meta-features.

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*



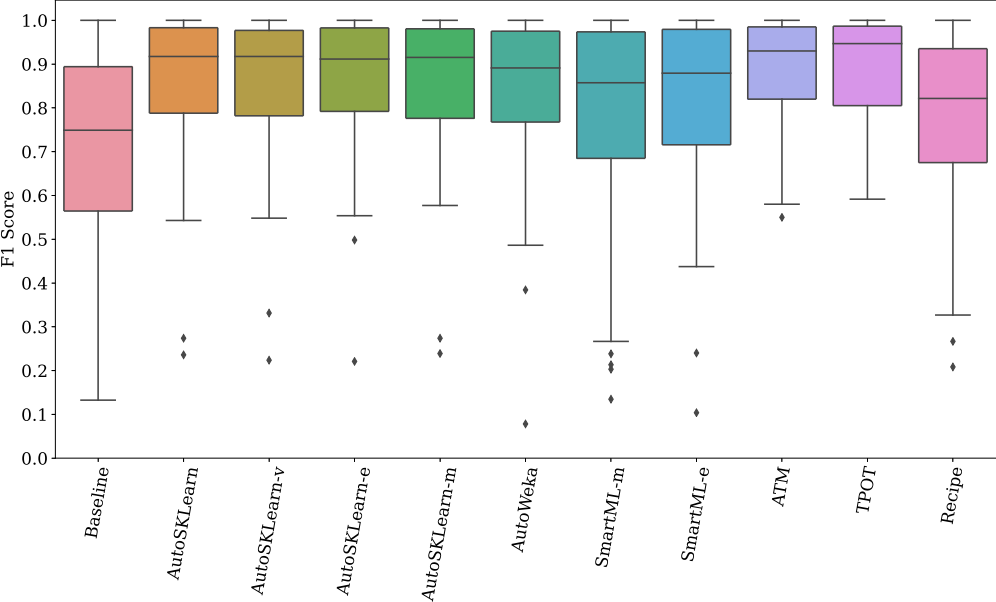


**Fig. D.9.** Average performance of all frameworks (10 Min) compared to the baseline.





**Fig. D.10.** Average performance of all frameworks (30 Min) compared to the baseline.

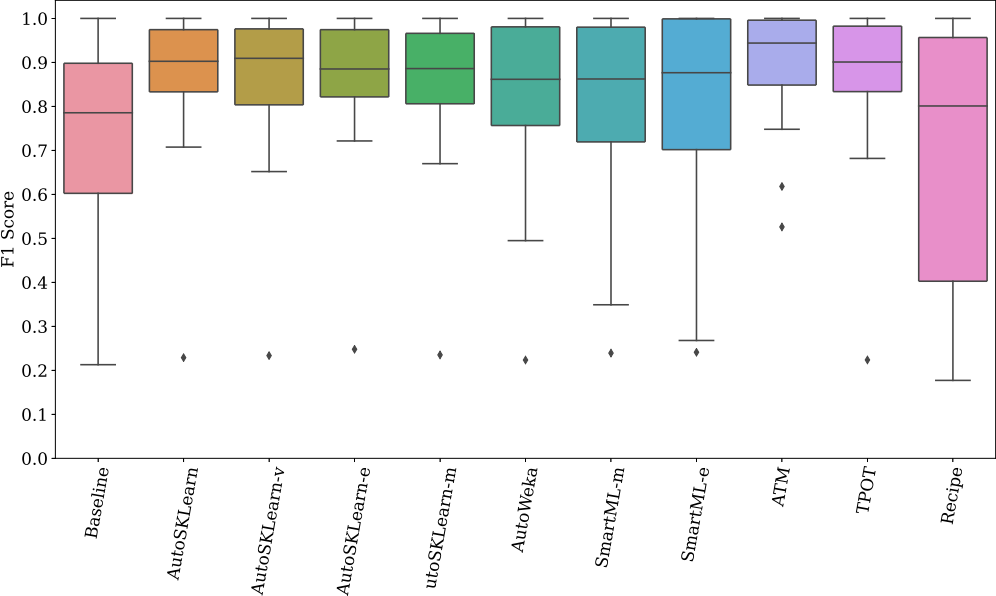




**Fig. D.11.** Average performance of all frameworks (60 Min) compared to the baseline.

17

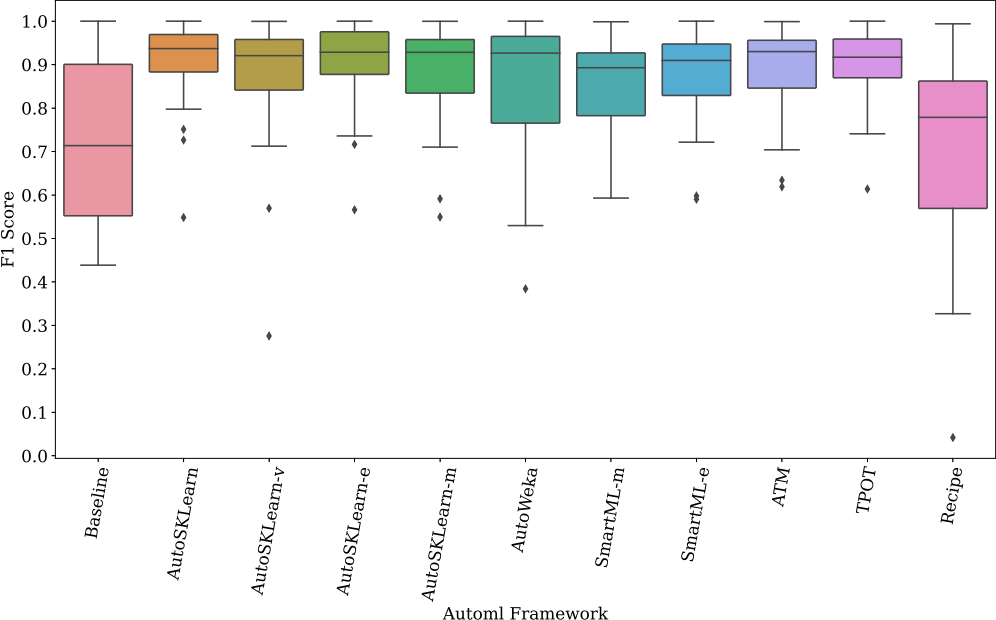
*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*



**Fig. D.12.** Performance of the final pipeline for datasets with large number of features and small number of instances.



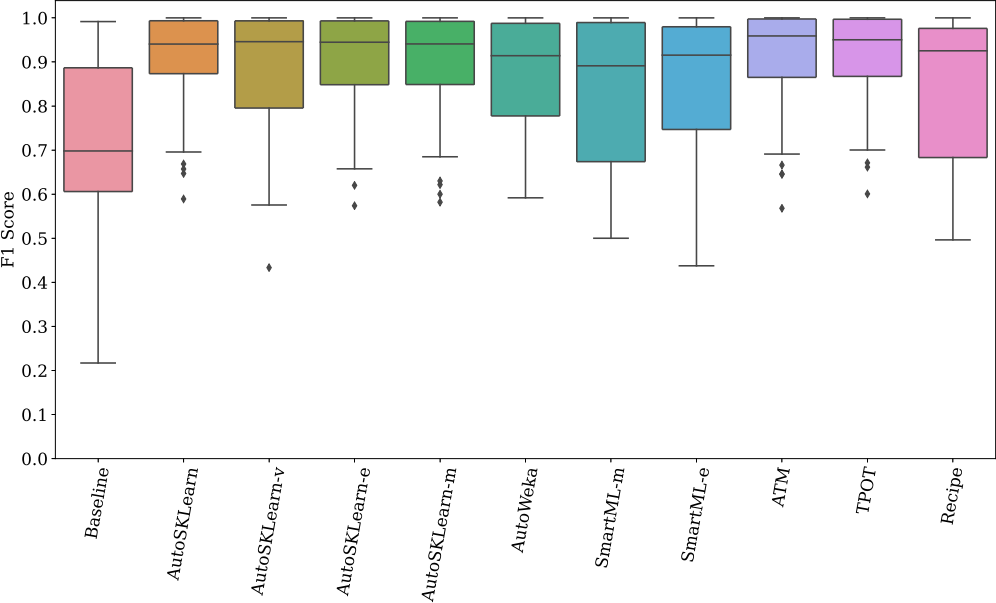
**Fig. D.13.** Performance of the final pipeline for datasets with large number of features and large number of instances.



**Fig. D.14.** Performance of the final pipeline for datasets with small number of features and small number of instances.

18

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*



Automl Framework

**Fig. D.15.** Performance of the final pipeline for datasets with small number of features and large number of instances.

**Table E.12**

Overview of the used meta-features. Groups from top to bottom: simple, statistical, and information-theoretic. Continuous features *X* and target *Y* have

mean *X* , stdev *X* , variance . Categorical features X and class C have categorical values *i*, conditional probabilities *i*∣*j*, joint probabilities *i,j* ,

marginal probabilities *i*+ =  *ij* , entropy *H*(X) = − ∑*i i*+*log*2 ( *i*+). [Vanschoren](#bookmark107) ([2018](#bookmark107)).

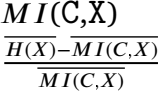
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Formula | Rationale | Additional variants | |
| Nr instances Nr features | *n p* | Speed, Scalability ([Michie et al.](#bookmark123), [1994](#bookmark123)) Curse of dimensionality ([Michie et al.](#bookmark123), [1994](#bookmark123)) | *p*∕*n*, *log*(*n*) *log*(*p*), Nr/ | ∕ ( *i*) |
| Nr classes | *c* | Complexity, imbalance ([Michie et al.](#bookmark123), [1994](#bookmark123)) | min/max/ | ( *c* )  *n* |
| Nr missing values | *m* | Imputation effects ([Kalousis](#bookmark110), [2002](#bookmark110)) |  |  |
| Skewness Kurtosis | *E*(*X* − *X* )3 | Feature normality ([Michie et al.](#bookmark123), [1994](#bookmark123)) Feature normality ([Michie et al.](#bookmark123), [1994](#bookmark123)) | min,max, min,max, | , , *q*1*, q*3 , , *q*1*, q*3 |
| Correlation | *X*1*X*2 | Feature interdependence ([Michie et al.](#bookmark123), | min,max, | , |
|  |  | [1994](#bookmark123)) |  |  |

Class entropy Norm. entropy

Mutual inform.

Noise-signal ratio

*H*(C) 



Class imbalance ([Michie et al.](#bookmark123), [1994](#bookmark123))

Feature informativeness ([Castiello et al.](#bookmark124),

[2005](#bookmark124))

Feature importance ([Michie et al.](#bookmark123), [1994](#bookmark123))

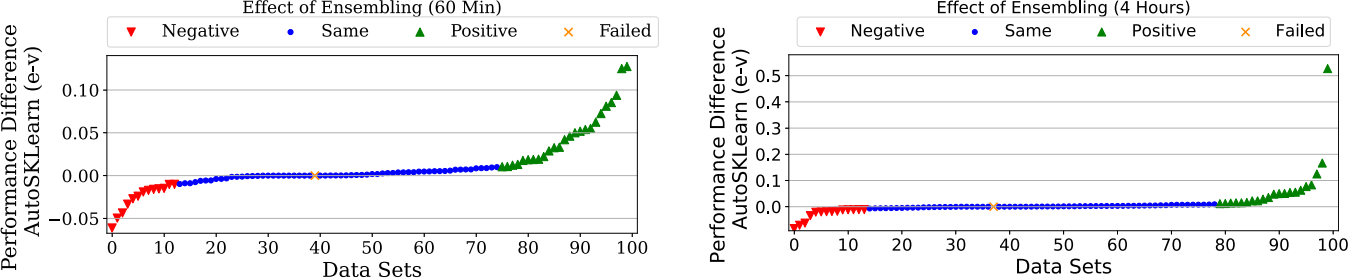
Noisiness of data ([Michie et al.](#bookmark123), [1994](#bookmark123))

*H*(C)/ (*MI*(C,X)) min,max, ,

min,max, ,







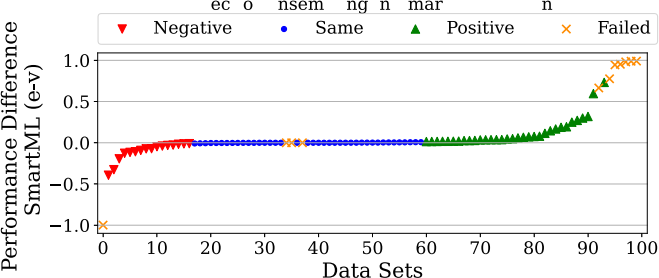
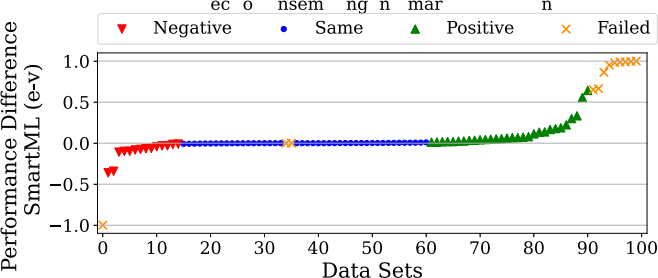




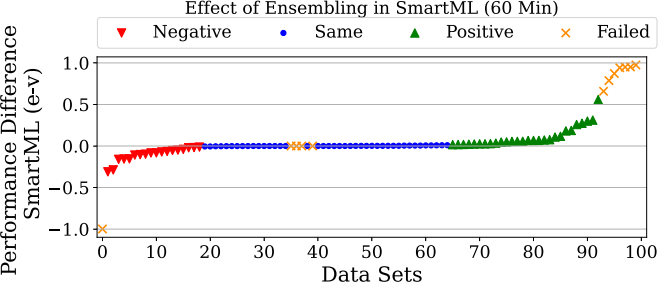
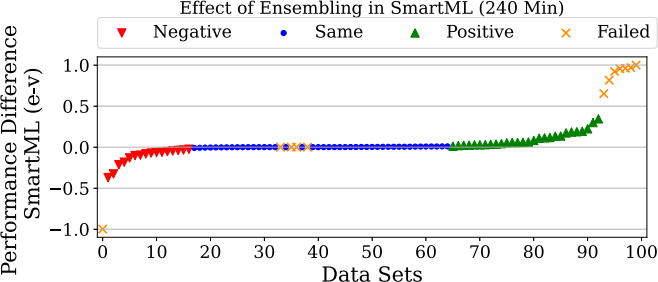
**Fig. F.16.** The performance difference between the AutoSKlearn-e and AutoSKlearn-v over different time budgets. Green markers represent better performance with AutoSKlearn-e, blue markers represent comparable performance with a difference less than 1%, red markers represent better performance with AutoSKlearn-v, and yellow markers represent failed runs on both versions.

19

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

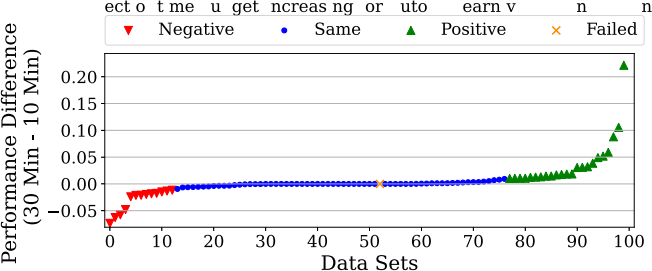
Eff t fE bli i S tML (10 Mi ) Eff t f E bli i S tML (30 Mi )

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

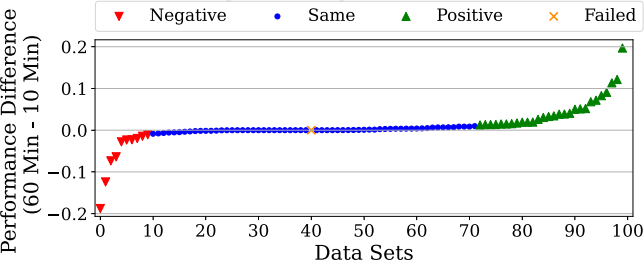


(c) 60 Min. (d) 240 Min.

**Fig. F.17.** The performance difference between the SmartML-m and SmartML-e. Green markers represent better performance with SmartML-e, blue markers represent comparable performance with a difference less than 1%, red markers represent better performance with SmartML, yellow markers on the right represent failed runs with SmartML-m but successful with SmartML-e, yellow markers on the left represent failed runs with SmartML-e but successful with SmartML-m and yellow markers in the middle represent failed runs with both SmartML-m and SmartML-e.

Effect of time budget Increasing for AutoskLearn-v (60 Min- 10 Min)

Eff f i b d I i f A SKL - (30 Mi - 10 Mi )

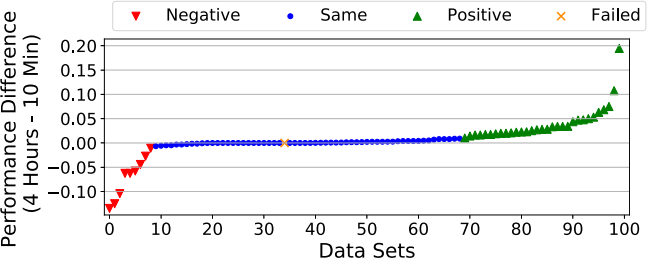
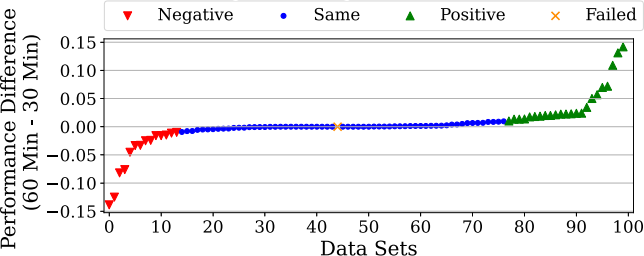


(a) 10- 30 Min.

(b) 10- 60 Min.

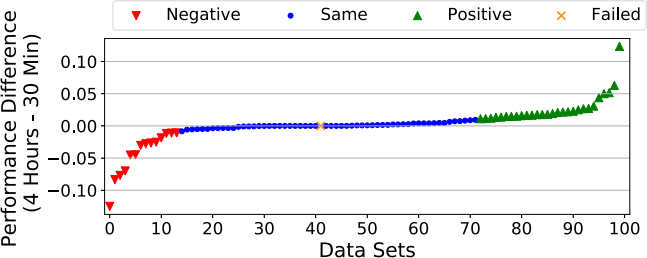
Effect of time budget Increasing for AutoskLearn-V (4 Hours- 10 Min)

Effect of time budget Increasing for AutoskLearn-V (60 Min- 30 Min)

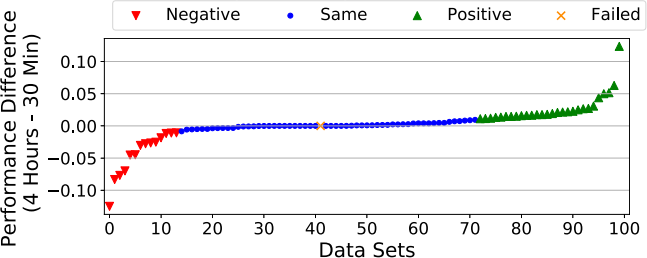


(c) 10-240Min.

(d) 30- 60 Min.

Effect of time budget Increasing for AutoskLearn-V (4 Hours- 30 Min)

Effect of time budget Increasing for AutoskLearn-V (4 Hours- 30 Min)



(e) 30-240Min.

(f) 60-240 Min.

**Fig. G.18.** The impact of increasing the time budget on AutoSKlearn-v performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

20

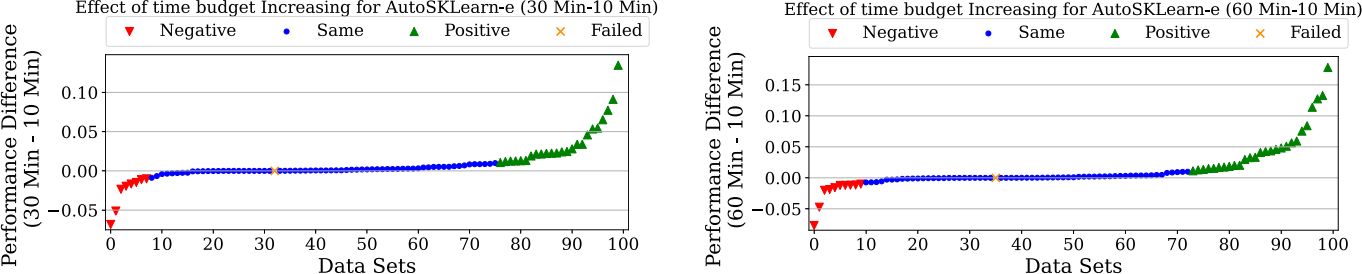
*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*



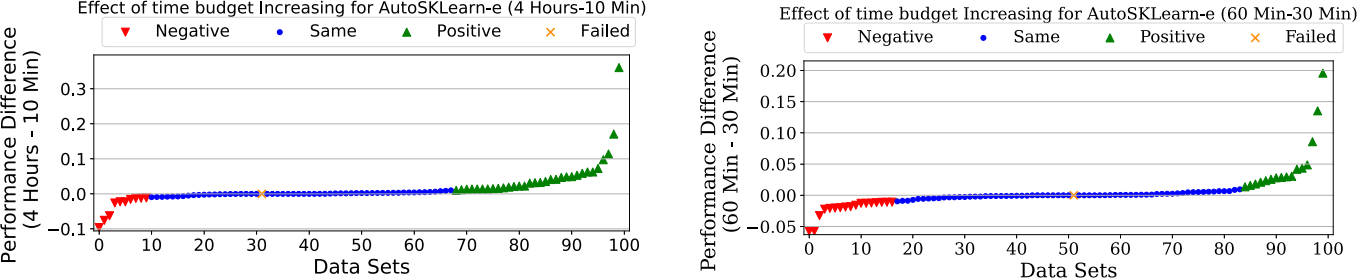




**Fig. G.19.** The impact of increasing the time budget on AutoSKlearn-m performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

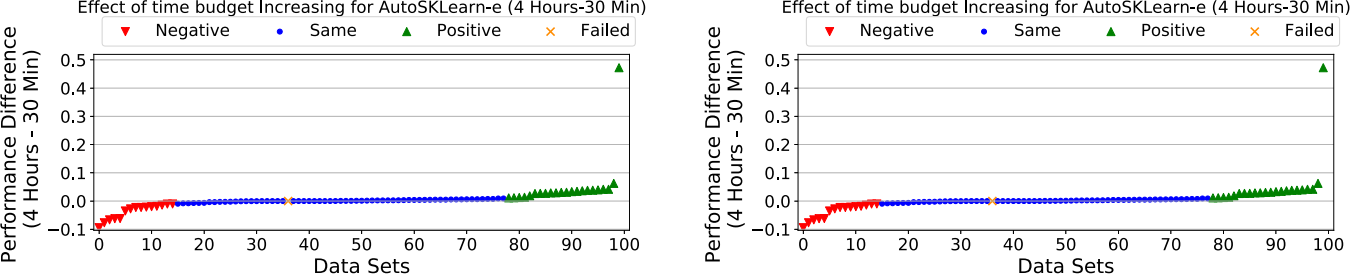










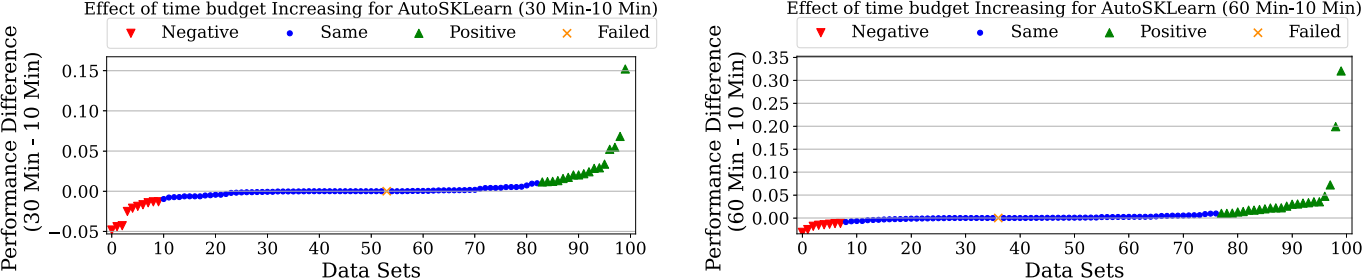






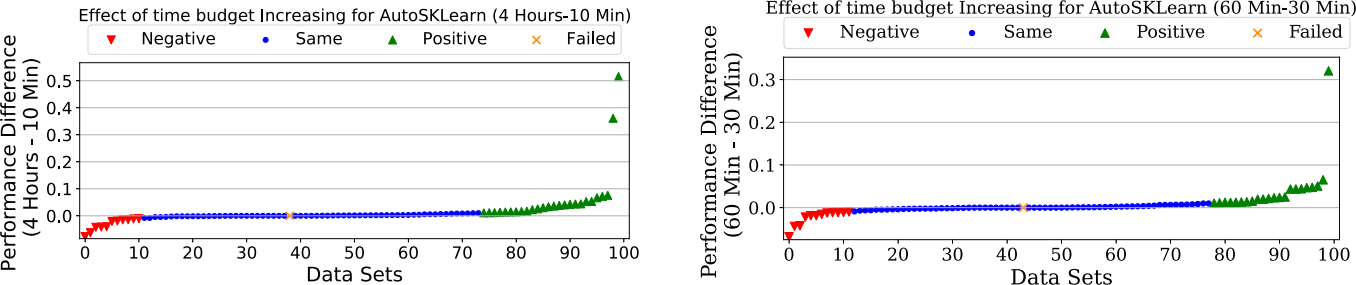
**Fig. G.20.** The impact of increasing the time budget on AutoSKlearn-e performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*



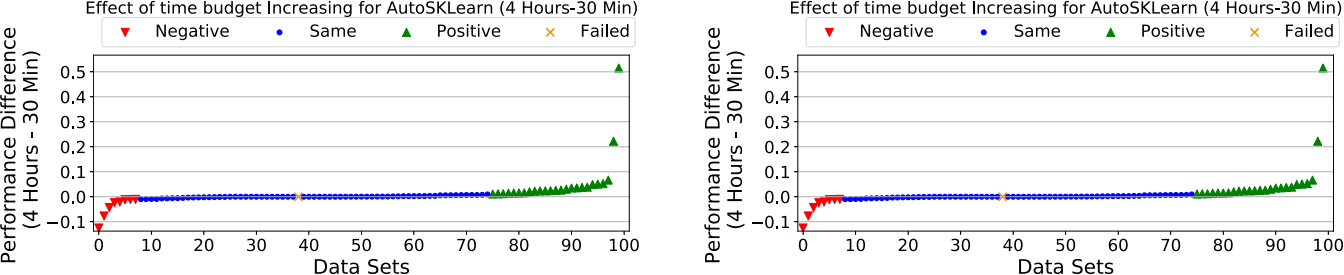








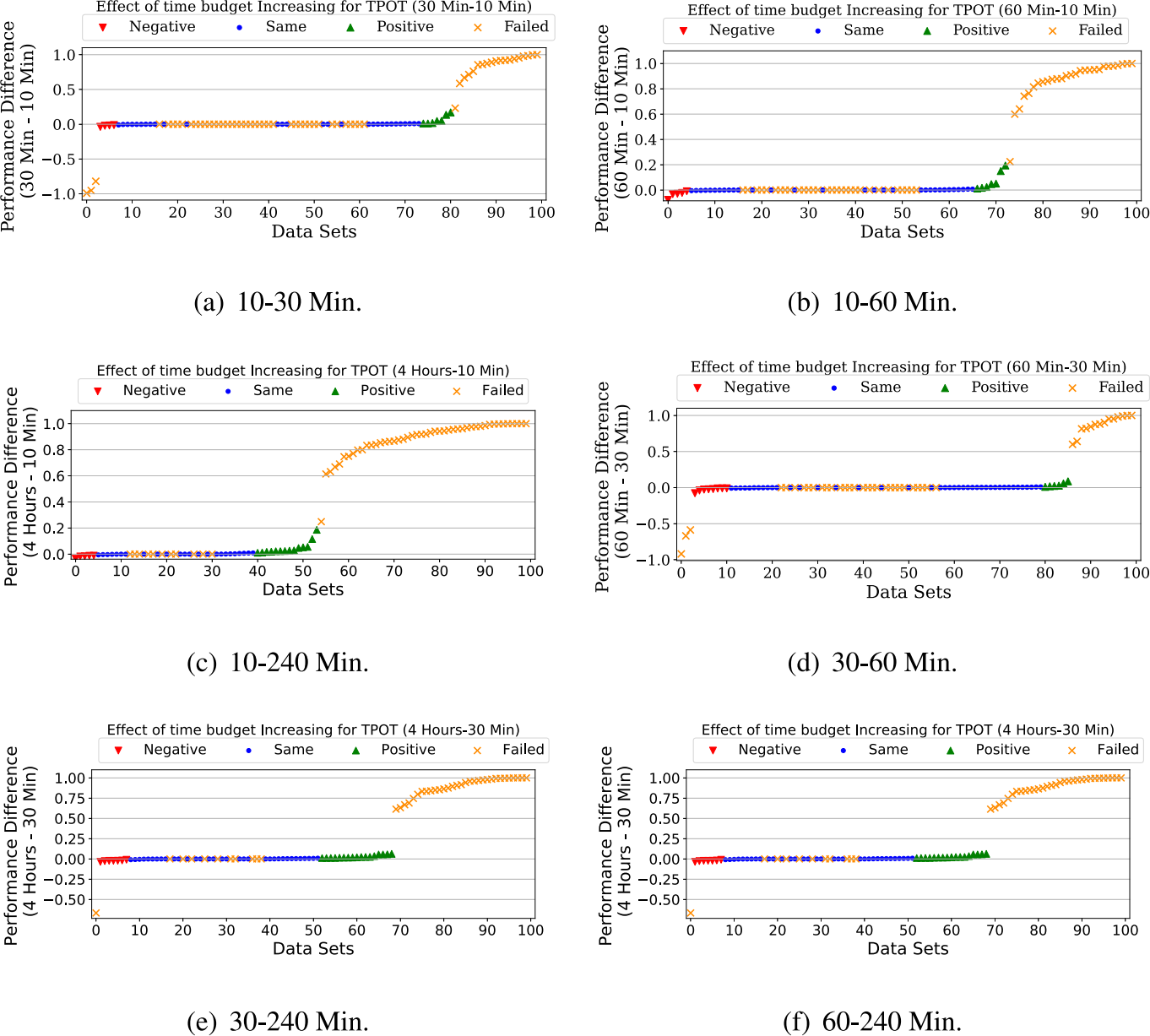








**Fig. G.21.** The impact of increasing the time budget on AutoSKlearn performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

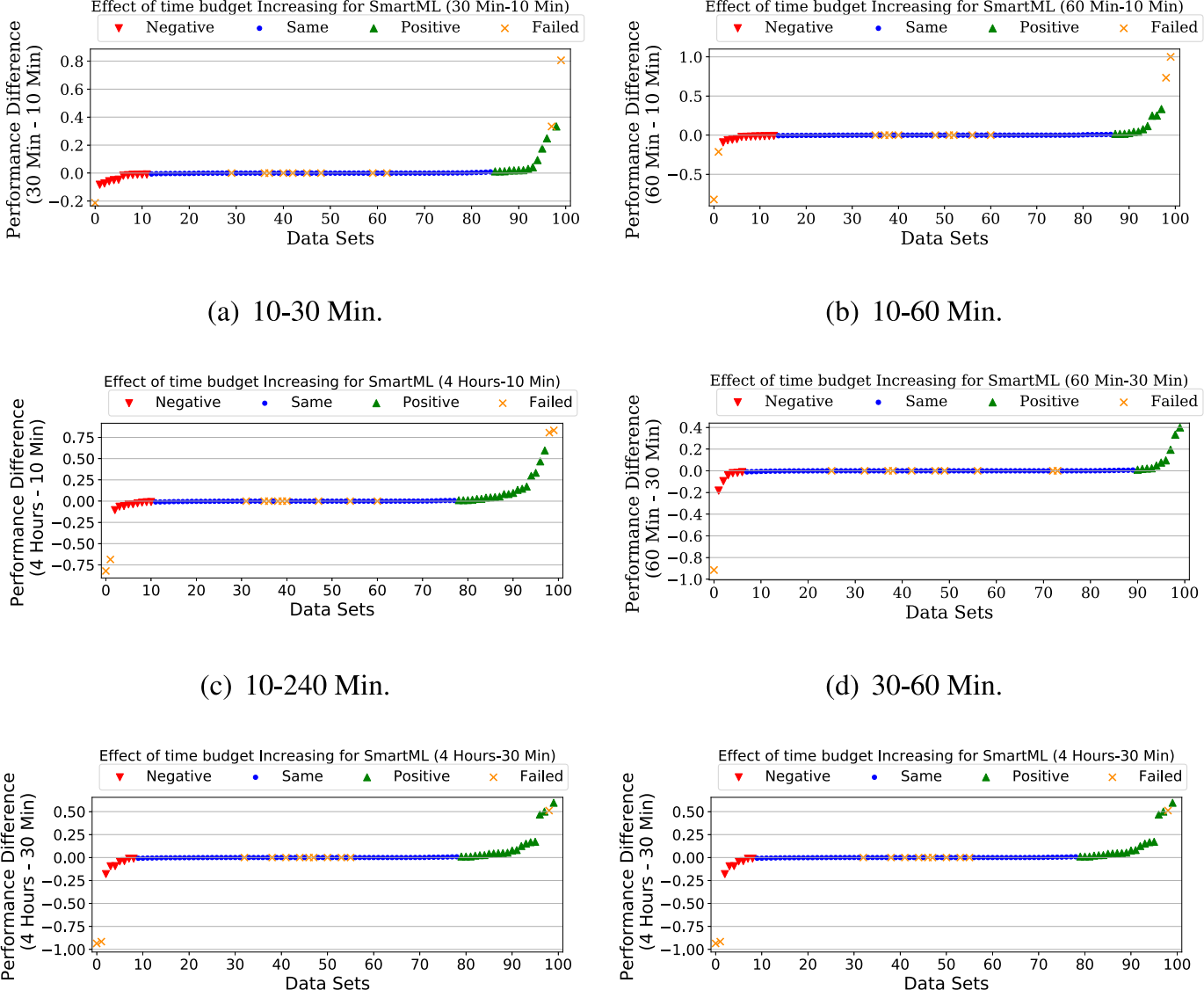


**Fig. G.22.** The impact of increasing the time budget on TPOT performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*



**Fig. G.23.** The impact of increasing the time budget on ATM performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

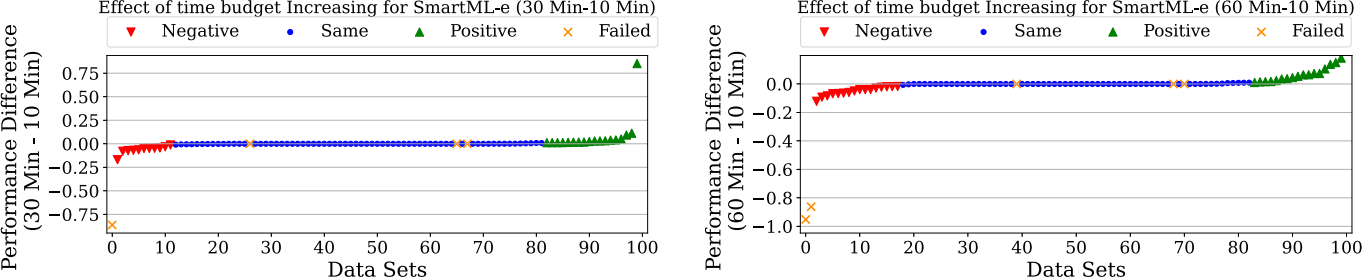






**Fig. G.24.** The impact of increasing the time budget on SmartML-m performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

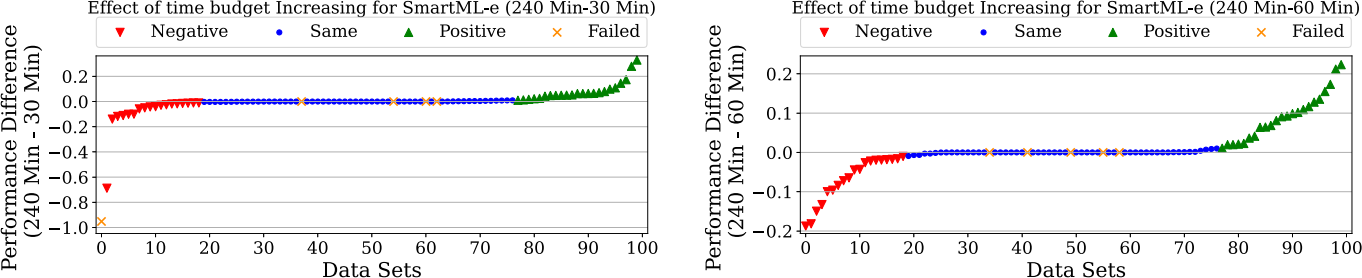








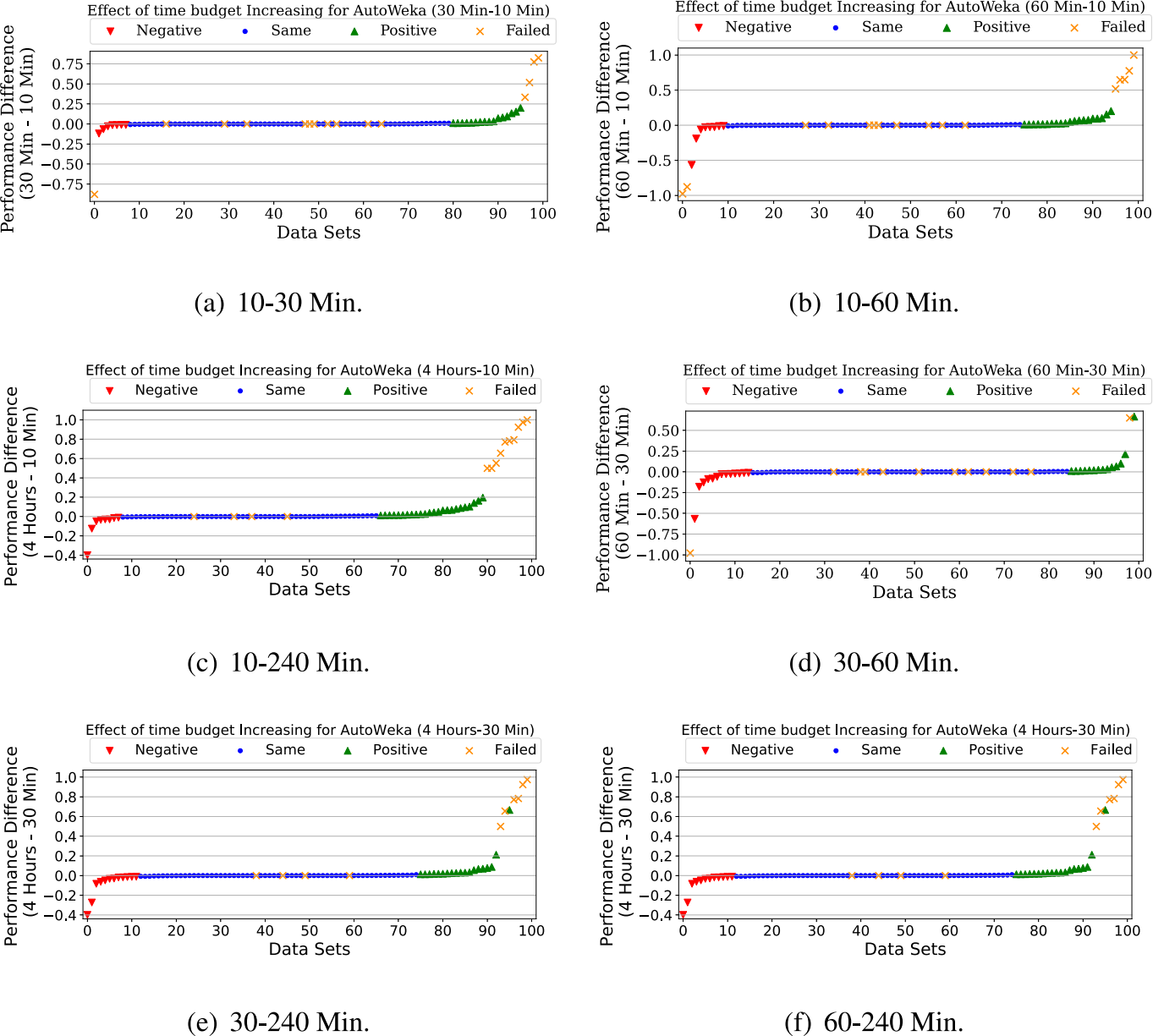






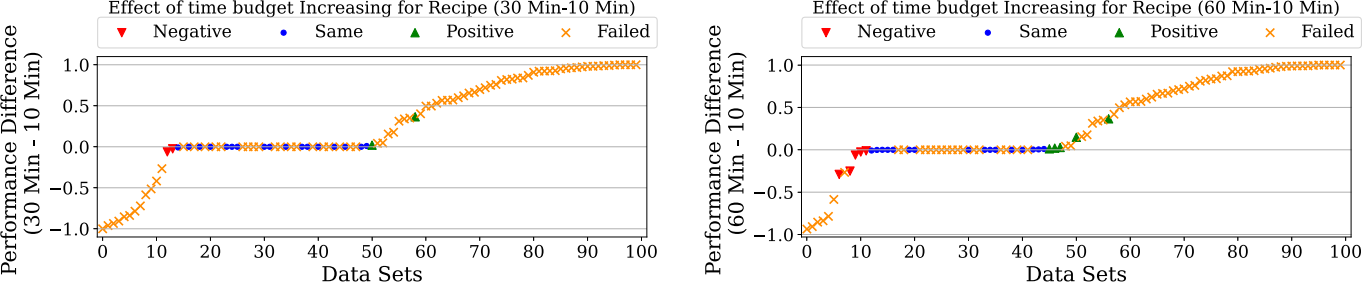


**Fig. G.25.** The impact of increasing the time budget on SmartML-e performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

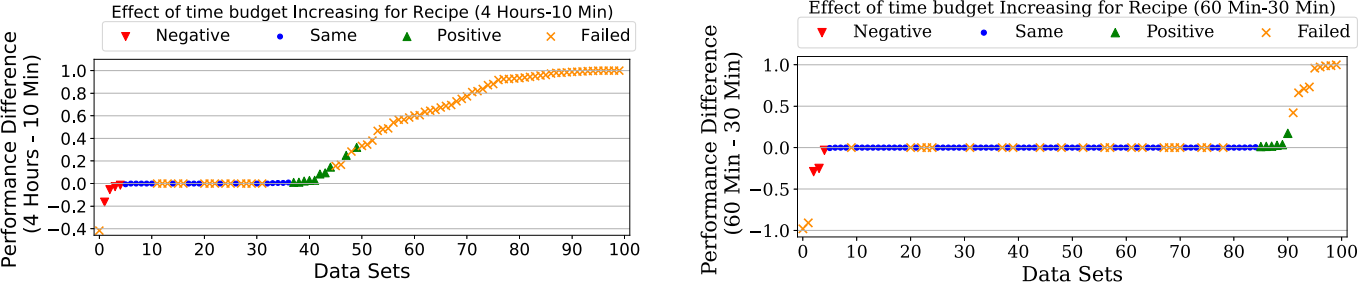


**Fig. G.26.** The impact of increasing the time budget on AutoWeka performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

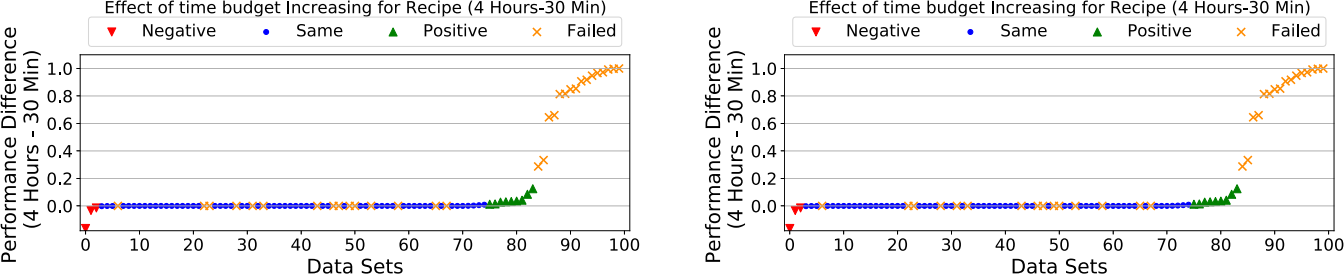
















**Fig. G.27.** The impact of increasing the time budget on Recipe performance from *x* to *y* minutes (x-y). Green markers represent better performance with *y* time budget, blue markers means that the difference between *x* and *y* is *<* 1. Red markers represent better performance on *x* time budget.

**Appendix F. Impact of ensembling**

[Figs.](#bookmark118) [F.16](#bookmark118) and [F.17](#bookmark119) shows the performance difference between the ensembling version and the vanilla/base version of AutoSKlearn and SmartML, respectively over 10, 30, 60 and 240 min time budgets.

**Appendix G. Impact of time budget**

[G.18](#bookmark81),[G.19](#bookmark82),[G.20](#bookmark83),[G.21](#bookmark84),[G.22](#bookmark85),[G.23](#bookmark86),[G.24](#bookmark87),[G.25](#bookmark88),[G.26](#bookmark89),[G.27](#bookmark90) show the im-

pact of increasing the time budget on the performance of the all the

AutoML frameworks considered in this work.

**References**

[Ali, R., Lee, S., & Chung, T. C. (2017). Accurate multi-criteria decision making](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb1) [methodology for recommending machine learning algorithm. *Expert Systems with*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb1)[*Applications*, *71*, 257–278.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb1)

[Beerenwinkel, N., & Siebourg, J. (2012). Probability, statistics, and computational](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb2) [science. *Evolutionary Genomics: Statistical and Computational Methods, Volume 1*,](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb2) [77–110.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb2)

[Bergstra, J., Yamins, D., & Cox, D. D. (2013a). Hyperopt: A Python library for](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb3) [optimizing the hyperparameters of machine learning algorithms. In *Proceedings of*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb3)[*the 12th python in science conference* (pp. 13–20). Citeseer.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb3)

[Bergstra, J., Yamins, D., & Cox, D. D. (2013b). Making a science of model search:](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb4) [Hyperparameter optimization in hundreds of dimensions for vision architectures.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb4) [*Journal of Machine Learning Research*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb4)

[Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V.,](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb5) [Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A.,](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb5) [Holt, B., & Varoquaux, G. (2013). API design for machine learning software:](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb5) [experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb5)[*Data Mining and Machine Learning* (pp. 108–122).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb5)

[Caruana, R., Niculescu-Mizil, A., Crew, G., & Ksikes, A. (2004). Ensemble selection from](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb6) [libraries of models. In *ICML*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb6)

[Castiello, C., Castellano, G., & Fanelli, A. M. (2005). Meta-data: Characterization of](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb7) [input features for meta-learning. In *International conference on modeling decisions for*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb7)[*artificial intelligence* (pp. 457–468). Springer.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb7)

[de Sá, A. G. C., Pinto, W. J. G. S., Oliveira, L. O. V. B., & Pappa, G. L. (2017). RECIPE:](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb8) [A grammar-based framework for automatically evolving classification pipelines. In](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb8) [*Lecture notes in computer science*: *vol.10196*, *EuroGP* (pp. 246–261).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb8)

[Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb9)[*workshop on multiple classifier systems*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb9)

[Eldeeb, H., Matsuk, O., Maher, M., Eldallal, A., & Sakr, S. (2021). The impact of](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb10) [auto-sklearn’s learning settings: Meta-learning, ensembling, time budget, and search](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb10) [space size. In *EDBT/ICDT workshops*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb10)

[Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2020).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb11) [Autogluon-tabular: Robust and accurate automl for structured data.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb11)

[Falkner, S., Klein, A., & Hutter, F. (2018). Bohb: Robust and efficient hyperparameter](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb12) [optimization at scale.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb12)

[Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2020).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb13) [Auto-sklearn 2.0: The next generation 24.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb13)

[Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb14) [Efficient and robust automated machine learning. In *Advances in neural information*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb14)[*processing systems* (pp. 2962–2970).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb14)

[Fitzgerald, B. (2012). Software crisis 2.0. *Computer*, *45*(4).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb15)

[Gehan, E. A. (1965). A generalized Wilcoxon test for comparing arbitrarily](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb16) [singly-censored samples. *Biometrika*, *52*(1–2), 203–224.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb16)

[Gijsbers, P., Bueno, M. L., Coors, S., LeDell, E., Poirier, S., Thomas, J., Bischl, B., &](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb17) [Vanschoren, J. (2022). AMLB: An AutoML benchmark.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb17)

[Gijsbers, P., LeDell, E., Thomas, J., Poirier, S., Bischl, B., & Vanschoren, J. (2019). An](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb18) [open source AutoML benchmark.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb18)

[Gijsbers, P., & Vanschoren, J. (2020). GAMA: A general automated machine learning](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb19) [assistant. In *Joint european conference on machine learning and knowledge discovery*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb19)[*in databases* (pp. 560–564). Springer.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb19)

[He, X., Zhao, K., & Chu, X. (2019). AutoML: A survey of the state-of-the-art.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb20)

[Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimiza-](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb21) [tion for general algorithm configuration. In *International conference on learning and*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb21)[*intelligent optimization*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb21)

[International Data Corporation (2017). *Worldwide semiannual cognitive/artificial*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb22)[*intelligence systems spending guide*: *Technical report*, International Data Corporation.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb22)

[Jin, H., Song, Q., & Hu, X. (2019). Auto-keras: An efficient neural architecture search](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb23) [system. In *ACM KDD*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb23)

[Kalousis, A. (2002). *Algorithm selection via meta-learning* (Ph.D. thesis), University of](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb24) [Geneva.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb24)

[Klein, A., Falkner, S., Mansur, N., & Hutter, F. (2017). Robo: A flexible and robust](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb25) [bayesian optimization framework in Python. In *NIPS 2017 bayesian optimization*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb25)[*workshop*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb25)

*H. Eldeeb et al. Expert Systems With Applications 243 (2024) 122877*

[Komer, B., Bergstra, J., & Eliasmith, C. (2014). Hyperopt-sklearn: Automatic hy-](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb26) [perparameter configuration for scikit-learn. In *ICML workshop on autoML, vol.9*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb26) [Citeseer.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb26)

[Kotthoff, L. (2016). Algorithm selection for combinatorial search problems: A survey.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb27) [In *Data mining and constraint programming* (pp. 149–190). Springer.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb27)

[Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2017). Auto-](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb28) [WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb28) [*JMLR*, *18*(1).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb28)

[Lam, L., & Suen, S. (1997). Application of majority voting to pattern recognition: An](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb29) [analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb29)[*Cybernetics-Part A: Systems and Humans*, *27*(5), 553–568.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb29)

[Landis, J. R., & Koch, G. G. (1977). An application of hierarchical kappa-type statistics](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb30) [in the assessment of majority agreement among multiple observers. *Biometrics*,](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb30) [363–374.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb30)

LeDell, E., & Poirier, S. (2020). H2O autoML: Scalable automatic machine learning. In *7th ICML workshop on automated machine learning*. URL [https://www.automl.org/](https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf) [wp-content/uploads/2020/07/AutoML\_2020\_paper\_61.pdf.](https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf)

[Maher, M., & Sakr, S. (2019). SmartML: A meta learning-based framework for](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb32) [automated selection and hyperparameter tuning for machine learning algorithms.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb32) [In *EDBT: 22nd international conference on extending database technology*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb32)

[Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine learning, neural and*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb33)[*statistical classification*. Citeseer.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb33)

[Mitchell, T., Buchanan, B., DeJong, G., Dietterich, T., Rosenbloom, P., & Waibel, A.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb34) [(1990). Machine learning. *Annual Review of Computer Science*, *4*(1), 417–433.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb34)

[Mohr, F., Wever, M., & Hüllermeier, E. (2018). ML-Plan: Automated machine learning](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb35) [via hierarchical planning. *Machine Learning*, *107*(8–10), 1495–1515.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb35)

[Olson, R. S., Bartley, N., Urbanowicz, R. J., & Moore, J. H. (2016). Evaluation of a](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb36) [tree-based pipeline optimization tool for automating data science. In *Proceedings of*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb36)[*the genetic and evolutionary computation conference 2016* (pp. 485–492).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb36)

[Olson, R. S., & Moore, J. H. (2016). TPOT: A tree-based pipeline optimization tool](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb37) [for automating machine learning. In *Workshop on automatic machine learning* (pp.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb37) [66–74). PMLR.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb37)

[Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O.,](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb38) [Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A.,](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb38) [Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn:](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb38) [Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb38)

Płońska, A., & Płoński, P. (2021). MLJAR: State-of-the-art automated machine learning framework for tabular data. Version 0.10.3. Łapy, Poland: MLJAR Sp. z o.o., URL <https://github.com/mljar/mljar-supervised>.

Shawi, R. E., Maher, M., & Sakr, S. (2019). Automated machine learning: State-of-the-art and open challenges. CoRR abs/1906.02287, [arXiv:1906.02287](http://arxiv.org/abs/1906.02287).

[Smith, M. J., Sala, C., Kanter, J. M., & Veeramachaneni, K. (2020). The machine](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb41) [learning bazaar: Harnessing the ml ecosystem for effective system development.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb41) [In *Proceedings of the 2020 ACM SIGMOD international conference on management of*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb41)[*data* (pp. 785–800).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb41)

[Swearingen, T., Drevo, W., Cyphers, B., Cuesta-Infante, A., Ross, A., & Veeramacha-](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb42) [neni, K. (2017). ATM: A distributed, collaborative, scalable system for automated](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb42) [machine learning. In *2017 IEEE international conference on big data (big data)*.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb42)

[Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-WEKA:](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb43) [Combined selection and hyperparameter optimization of classification algorithms.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb43) [In *ACM KDD* (pp. 847–855). ACM.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb43)

[Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, C. B., & Farivar, R. (2019).](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb44) [Towards automated machine learning: Evaluation and comparison of AutoML](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb44) [approaches and tools. In *2019 IEEE 31st international conference on tools with*](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb44)[*artificial intelligence* (pp. 1471–1479). IEEE.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb44)

[Vakhrushev, A., Ryzhkov, A., Savchenko, M., Simakov, D., Damdinov, R., & Tuzhilin, A.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb45)

[(2021). LightAutoML: AutoML solution for a large financial services ecosystem.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb45) [Vanschoren, J. (2018). Meta-learning: A survey.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb46)

Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2013). OpenML: Networked science in machine learning. *SIGKDD Explorations*, *15*(2), 49–60. [http://dx.doi.org/](http://dx.doi.org/10.1145/2641190.2641198) [10.1145/2641190.2641198](http://dx.doi.org/10.1145/2641190.2641198).

[Wang, C., Wu, Q., Weimer, M., & Zhu, E. (2021). FLAML: A fast and lightweight automl](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb48) [library. *Proceedings of Machine Learning and Systems*, *3*, 434–447.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb48)

[Zöller, M.-A., & Huber, M. F. (2021). Benchmark and survey of automated machine](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb49) [learning frameworks. *Journal of Artificial Intelligence Research*, *70*, 409–472.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb49)

[Zomaya, A. Y., & Sakr, S. (2017). *Handbook of big data technologies*. Springer.](http://refhub.elsevier.com/S0957-4174(23)03379-1/sb50)