Evaluation of AutoML Systems on OpenML Binary-Classification Tasks

Xinchen Yang[1], Jieshi Chen[2] and Artur Dubrawski[2]

***Abstract*— The Automated Machine Learning (AutoML) System has powered data scientists and domain experts with its efficient model discovery capabilities and helped address shortages of qualified data scientists. Widely used AutoML Systems include the Auto^{n}ML developed by the Auton Lab at Carnegie Mellon University (CMU), H_2O AutoML, Auto-Sklearn, etc. Various performance evaluations have been done on these AutoML systems. However, as AutoML systems get updated with improved model searching ability and newly added functionalities, we need to obtain a new map to depict the performance of the AutoML systems. Motivated by this goal, we conducted experiments to evaluate the performance of 4 popular AutoML systems, including Auto^{n}ML, H_2O AutoML, TPOT and AutoGluon, on 177 OpenML binary-classification tasks, using Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve as the evaluation metric. We analyzed the experimental data from various aspects, including the relative rankings of the AutoML systems, training-testing performance discrepancies, relationship between the performance of AutoML systems and dataset characteristics, and the winning algorithms used by each AutoML system.**

***Index Terms*— AutoML, OpenML, model selection**

I. INTRODUCTION

An automated machine learning (AutoML) pipeline is a combination of a series of steps of data preprocessing, feature selection, model and parameter tuning, etc. Each pipeline can be regarded as a well-defined classifier or regressor that takes machine leaning (ML) tasks as input and yield the predication values as the output. An AutoML system has many such pipelines in its search space. When given a ML task, an AutoML system will look for pipelines in its search space, apply these pipelines to the task and rank the performance of the pipelines according to a selected metric (accuracy, AUC, etc). We want to know whether a designed AutoML system can find the best pipelines given a ML task. In this sense, an AutoML system can be regarded as a "huge" machine learning model. For each task given to an AutoML system, its performance is represented by the performance of the best pipeline it returns —- the pipeline with the highest evaluation score on the AutoML's leaderboard of its searched pipelines.

Multiple open-source AutoML systems are available to use now and they are evolving rapidly. Here is a list of a few of them:

(1) Auto^{n}ML: Auto^{n}ML is an open-source AutoML system developed by CMU Auton Lab [1] using DARPA D3M ecosystem, aiming at power data scientists with efficient model discovery and advanced data analytics. Auto^{n}ML takes the training data as input, then conducts several operations including featurization, fitting and prediction, and validation. Auto^{n}ML outputs a leaderboard of ranked pipelines and the pipelines on the leaderboard can be used to make predictions on the testing data.

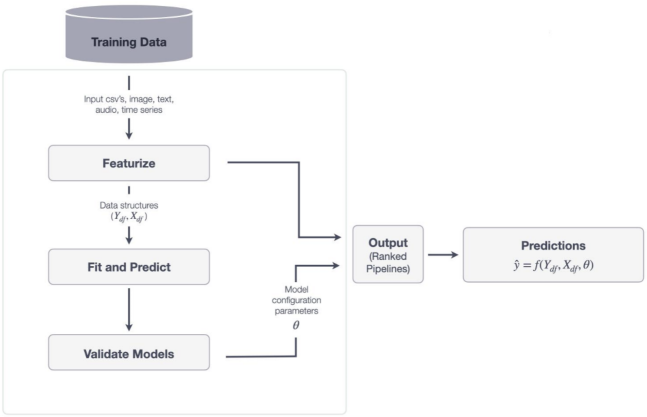

Fig. 1. Auto^{n}ML workflow

(2) H_2O AutoML: Presented by LeDell and Poirier [2], H_2O AutoML is an open-source, highly scalable, fully-automated AutoML framework. H2O AutoML uses a combination of fast random search and stacked ensembles to achieve competitive results. H_2O AutoML has an easy-to-use interface by providing simple wrapper functions that perform a large number of modeling tasks in order to save time for the user.

(3) Tree-Based Pipeline Optimization Tool (TPOT): TPOT is an open-source genetic programming-based AutoML framework introduced by Olson and Moore [3]. The goal of TPOT is to automate the pipeline building process by combining tree representation of pipelines with stochastic search algorithms. TPOT makes use of the Python-based scikit-learn library. .

(4) AutoGluon-Tabular: AutoGluon-Tabular is an open-source AutoML framework that utilizes the technique of ensembling multiple models and stacking them in multiple layers presented by Erickson [4]. The multi-layer combination of many models makes AutoGluon an AutoML that can produce results very quickly with still good results, which can serve the practical uses well.

In this paper, we evaluated the predication performance of 4 AutoML systems: Auto^{n}ML, H_2O AutoML, TPOT and AutoGluon on 177 OpenML binary-classification tasks. Our

[1]Xinchen Yang is with the Department of Electrical and Computer Engineering, New York University, 6 MetroTech Center, Brooklyn, NY, USA xy2332@nyu.edu

[2]Jieshi Chen and Artur Dubrawski, PhD, are with the Auton Lab, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA jieshic@andrew.cmu.edu, awd@cs.cmu.edu

goal is to compare the relative performance of these AutoML systems, identity whether there are relationships between the performance of AutoML systems and certain factors such as time budget and datasets characteristics. We also would like to know whether the top pipelines of those AutoML systems towards the same dataset are different in their selection of core algorithms or not.

II. RELATED WORK

Research has been conducted on the evaluation of AutoML methods and frameworks. Many developers of the existing AutoML systems conducted an evaluation on their AutoML system against other AutoML systems when they introduced their works. For example, the inventors of H_2O AutoML, LeDell and Poirier, evaluated H_2O AutoML against several other AutoML systems on the OpenML AutoML benchmark, which contains 44 classification tasks, proving the effectiveness of the H_2O AutoML [2]. Olson and Moore introduced TPOT and benchmarked TPOT's performance on a set of 150 supervised classification tasks and found that it significantly outperforms a basic machine learning algorithm in 21 tasks and has a minimal degradation only on 4 tasks. It is all accomplished without domain knowledge or tedious manual efforts, which shows a great promise of GP-based AutoML systems [3]. In addition, Erickson, who presented AutoGluon-Tabular, evaluated AutoGluon-Tabular on a suite of 50 classification and regression tasks from Kaggle and the OpenML AutoML Benchmark against several other AutoML platforms, showing the robustness and high performance of AutoGluon-Tabular. AutoGluon-Tabular was also showed to be a time-saving AutoML system in their experiment compared to others [4].

Comprehensive surveys have also been done on the evaluation of the AutoML systems. For example, Zoller and Huber evaluated the performance of a set of methods of algorithm selection and hyperparametr optimization and 6 AutoML frameworks (TPOT, Hyperopt-sklearn, Auto-Sklearn, Random Search, ATM, H_2O AutoML) on 137 OpenML datasets [5]. Ferreira conducted empirical evaluations of 8 AutoML tools on 12 OpenML datasets and compared the best scores achieved by the AutoML tools with the best OpenML public results, confirming the potential of AutoML tools to fully automate the manual efforts on model selection and hyperparameter tuning [6]. Truong evaluated a selected subset of AutoML tools on nearly 300 OpenML datasets, observing that most AutoML tools are able to obtain reasonable results in terms of their performance across many datasets, but there is no "perfect" tool that can outperform all others on a plurality of tasks yet [7].

Tools and platforms have been introduced to facilitate the evaluation and analysis of the AutoML frameworks. For example, Milutinovic introduced an standardized, open-source machine learning framework, D3M, upon which AutoML systems can be evaluated with their strengths and weaknesses exposed. Milutinovic also demonstrated the viability of the D3M framework through the evaluations of 8 AutoML systems upon it [8]. The Auto^{n}ML system developed by the Auton Lab at Carnegie Mellon University (CMU) is built upon the D3M framework.

III. METHODS

Experiments are set up to answer the following questions:

(1) What is the performance difference of the AutoML systems compared to each other?

(2) What is the performance difference of the AutoML systems under different time budgets?

(3) What is the performance difference of the AutoML systems evaluated on datasets with different characteristics?

(4) What is the main contributor to the discrepancies between the performance of AutoML systems, core algorithm selection or other factors?

To answer the questions above, we evaluated 4 AutoML systems: Auto^{n}ML, H_2O AutoML, TPOT and AutoGluon, on 177 OpenML binary classification tasks. Datasets are selected with varied dimensionalities and number of instances. Experiments were run separately on 3 machines, tagged as "Lab Server", "Desktop" and "NYU". All of them are 8-core Linux Machines. First, each dataset is randomly split into training and testing data, with 75% of the original data used as the training data and 25% used as the testing data. Next, the same training and testing data on each task are passed as input to each of the AutoML systems. The metric used to measure the performance of the AutoML systems is AUC. Each AutoML system will look for plausible machine learning pipelines in its search space and rank them according to the training score in AUC. For each AutoML system, the Top 1 pipeline on its leaderboard is used to predict on the testing data, representing the AutoML system that finds it. We set up 3 experimental groups with the time budget to be 60 seconds, 600 seconds and 1200 seconds respectively. The same evaluation process is repeated within the 3 groups only with altered time budget.

IV. RESULTS

After running the experiments, we collected the train and test prediction score of the AutoML systems on the 177 datasets under 3 different time budgets and analyzed the experimental data from several aspects.

A. Performance of AutoML Systems over Time Budget

To illustrate the performance of AutoML systems over different time budget, we compared the relative rankings of the performance of the AutoML systems on the test data under the time budget of 60 seconds, 600 seconds, and 1200 seconds, as shown in Figure 2. For each data point, its X-coordinate denotes the time budget of the experiment and its Y-coordinate is obtained by averaging the performance rankings of its corresponding AutoML system across all datasets, with corresponding 95% confidence intervals.

In Figure 2, AutoGluon shows to be an AutoML system with good performance, getting the first place under the time budget of 60 seconds and 600 seconds, and getting the second place under the time budget of 1200 seconds. The average rank of Auto^{n}ML is rather stable across different

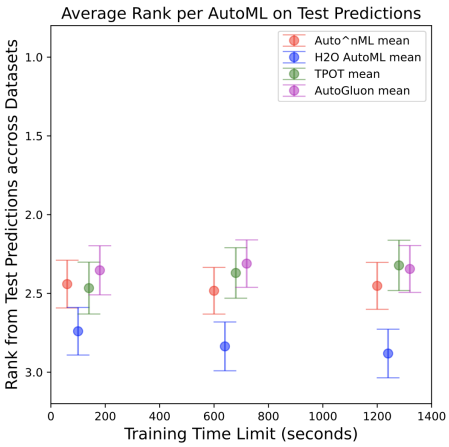

Fig. 2. Average Rank of Test AUCs per AutoML

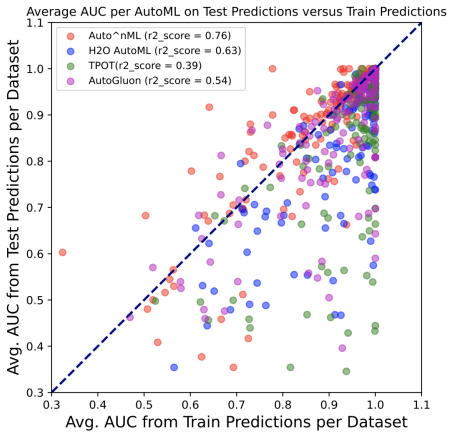

Fig. 3. Average AUC of testing data versus average AUC on Training data per Dataset

time budgets. The average rank of TPOT increases as the time budget increases. In addtion, we can observe that the performance of Auton^{n}ML, TPOT, and AutoGluon are relatively close to each other, while the performance of H_2O AutoML apparently lags behind. As the time budget increases, H_2O AutoML even shows a bigger disadvantage towards other AutoML systems.

TABLE I

RANK STATISTICS ON TEST DATA PER TIME BUDGET, AVERAGED ACROSS N=177 DATASETS, ± STANDARD ERROR OF THE MEAN

	60 seconds	600 seconds	1200 seconds
Auto^{n}ML	2.44±0.15	2.48±0.15	2.45±0.15
H_2O AutoML	2.74±0.15	2.84±0.15	2.88±0.15
TPOT	2.47±0.16	2.37±0.16	2.32±0.16
AutoGluon	2.35±0.16	2.31±0.15	2.34±0.15

B. Training-Testing Performance Discrepancies

To illustrate the relationship between the training and testing performance of the AutoML systems, we make a plot to compare the performance metric (absolute AUC score) from the testing data versus those from the training data as shown in Figure 3. For each data point, its X-coordinate is obtained by averaging the training performance (based on AUC score) of its corresponding AutoML across all time budgets on a specific dataset. Similarly, for each data point, its Y-coordinate is obtained by averaging the test performance (based on AUC score) of its corresponding AutoML across all time budgets on the same dataset. We also include the diagonal line $y = x$ to denote the ideal scenario when the test prediction score is equal to the train prediction score.

We use R-squared (R^2) score as a metric to evaluate the degree of consistency between training AUC scores and testing AUC scores across all datasets per AutoML system. We can observe that Auto^{n}ML, H_2O AutoML, TPOT and AutoGluon achieve a R^2 score of 0.76, 0.63, 0.39, and 0.54 respectively. It shows that, compared to the other AutoML systems, Auto^{n}ML has a stronger correlation between its test prediction score and train prediction score, which is a desired feature. We hope that the data points can get as close to the line $y = x$ as possible, because we want the train performance of the AutoML systems can serve as a good indicator of the test performance of the AutoML systems. Here, we can observe that TPOT have many data points that stray far away from the line $y = x$, with a test performance far worse than the train performance, which indicates that TPOT may suffer from an over-fitting problem on some data tasks.

C. Relationship Between AutoML System Performances and Dataset Characteristics

We are interested in how the characteristics of the datasets, in particular, dimensionality and number of instances, affect the performance of the AutoML systems.

To illustrate the relationship between the performance of the AutoML systems and the dimensionality of the datasets, we divide the whole range of dimensionality into distinct ”buckets” using an interval of 10. Within each dimensionality ”bucket”, we average the test prediction scores over all time budgets over all datasets for each AutoML to obtain a data point. The results are shown in Table II and Figure 4.

Similarly, to illustrate the relationship between the performance of the AutoML systems and the number of instances of the datasets, we divide the whole range of number of instances into distinct ”buckets” using an interval of 1000. Within each ”bucket” of number of instances, we average the test prediction scores over all time budgets over all datasets for each AutoML to obtain a data point. The results are shown in Table III and Figure 5.

Regarding the relationship between the performance of AutoML systems and the dimensionality of datasets, we can see that H_2O AutoML remains in a lagging position almost over all dimensionalities, as shown in Figure 4, which is consistent to its state of falling behind in its average rank across datasets over different time budgets. As the dimensionalities increase, the average rank of Auto^{n}ML climbs up first, reaching its peak when the dimensionality

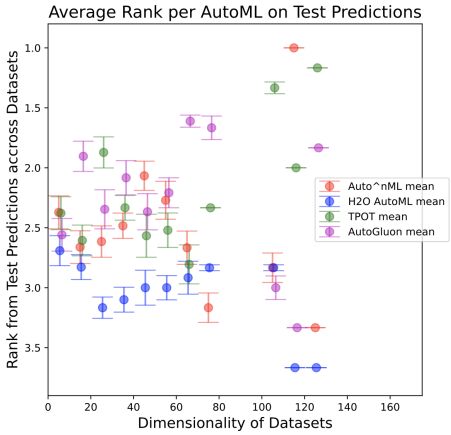

Fig. 4. Average Rank per AutoML on Test Predictions across Datasets within Different Dimensionality Groups

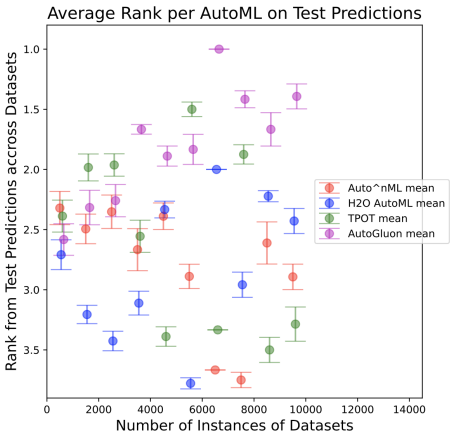

Fig. 5. Average Rank per AutoML on Test Predicitions accross Datasets within Different Groups of Number of Instances

is within the range of [40,50), then falls down, forming the shape of a parabola. However, we are not sure whether this pattern can reveal certain relationships between the performance of Auto^{n}ML and the dimensionality of datasets. We are not certain whether this pattern is representative, either. For TPOT and AutoGluon, no obvious relationships between their performance and the dimensionality of datasets can be found for now.

Regarding the relationship between the performance of AutoML systems and the number of instances, Figure 5 shows that AutoGluon has an upward trend in terms of its relative ranking as the number of instances increases. When the the number of instances is greater than or equal to 6000, AutoGluon apparently prevails over other AutoML systems. TPOT, on the other hand, achieves high relative ranks when the dataset is small (e.g. with number of instances less than 3000), but suffers from a significant drop in terms of its relative ranking on the 17 datasets with more than 8000 instances. No definitive conclusion has been found on the relationship between the relative performance of Auto^{n}ML and number of instances of the dataset, yet. However, H_2O AutoML seems to have a lift in its raltive ranking from its lagging position when the number of instances of the dataset is greater than or equal to 6000.

TABLE II
AVERAGE RANK PER AUTOML ON TEST PREDICTIONS ACROSS N=177 DATASETS WITHIN DIFFERENT DIMENSIONALITY GROUPS

Dimensionality	Number of Tasks	Auto^{n}ML	H_2O AutoML	TPOT	AutoGluon
[0,10)	96	2.37±0.14	2.69±0.12	2.38±0.14	2.56±0.14
[10,20)	33	2.66±0.14	2.83±0.10	2.61±0.13	1.90±0.13
[20,30)	13	2.62±0.13	3.17±0.09	1.87±0.13	2.35±0.16
[30,40)	10	2.48±0.11	3.10±0.10	2.33±0.10	2.08±0.14
[40,50)	5	2.07±0.12	3.00±0.15	2.57±0.18	2.37±0.15
[50,60)	8	2.27±0.16	3.00±0.10	2.52±0.14	2.21±0.13
[60,70)	6	2.67±0.14	2.92±0.14	2.81±0.16	1.61±0.05
[70,80)	2	3.17±0.12	2.83±0.02	2.33±0.00	1.67±0.10
[80,90)	0	NaN	NaN	NaN	NaN
[90,100)	0	NaN	NaN	NaN	NaN
[100,110)	2	2.83±0.12	2.83±0.02	1.33±0.05	3.00±0.10
[110,120)	1	1.00±0.00	3.67±0.00	2.00±0.00	3.33±0.00
[120,∞)	1	1.67±0.00	3.67±0.00	1.17±0.00	1.83±0.00

TABLE III
AVERAGE RANK PER AUTOML ON TEST PREDICTIONS ACROSS N=177 DATASETS WITHIN DIFFERENT GROUPS OF NUMBER OF INSTANCES

Number of Instances	Number of Tasks	Auto^{n}ML	H_2O AutoML	TPOT	AutoGluon
[0,1000)	107	2.32±0.14	2.71±0.12	2.39±0.13	2.58±0.13
[1000,2000)	30	2.49±0.12	3.21±0.08	1.98±0.11	2.32±0.14
[2000,3000)	9	2.35±0.14	3.43±0.08	1.96±0.09	2.26±0.13
[3000,4000)	3	2.67±0.17	3.11±0.10	2.56±0.13	1.67±0.04
[4000,5000)	3	2.39±0.11	2.33±0.07	3.39±0.08	1.89±0.08
[5000,6000)	3	2.89±0.10	3.78±0.05	1.5±0.06	1.83±0.13
[6000,7000)	1	3.67±0.00	2.00±0.00	3.33±0.00	1.00±0.00
[7000,8000)	4	3.75±0.06	2.96±0.10	1.88±0.08	1.42±0.07
[8000,9000)	3	2.61±0.18	2.22±0.05	3.50±0.10	1.67±0.14
[9000,∞)	14	2.89±0.11	2.43±0.10	3.29±0.14	1.39±0.10

D. Pipeline Algorithm Exploration

1) Frequency of Winning Algorithms: In order to illustrate the distribution of winning algorithms of each AutoML system, we collect the core algorithm used by the top AutoML on each data task at each time budget. When there is a tie, we treat all the AutoML systems that can achieve the highest test prediction score as the "top" AutoML system. We make bar plots to show the frequency of the first-place algorithms for each AutoML system in Figure 6 and the percentage frequency of the first-place algorithms for each AutoML system in Figure 7.

We can see from the percentage frequency plots that Auto^{n}ML has very stable winning core algorithms over varied time budgets. The same eight core algorithms have ever made Auto^{n}ML win across varied time budgets and their percentage splits across varied time budgets are close to each other. "gradient_boosting" is the algorithm that helps Auto^{n}ML win most, taking a share between 20% and 30% of the winning algorithms of Auto^{n}ML across all time budgets. There are several other noticeable points as well. For example, for H_2O AutoML, it starts with exploring a few algorithms when the time budget is limited, including XGBoost, DeepLearning, and Gradient Boosting Machine (GBM). As the time budget increases, H_2O AutoML explores more types of algorithms and uses them to win. However, as the time budget continues to increase, H_2O AutoML returns to its original choice of algorithms. For TPOT, we can see that it is the AutoML system that has the largest

number of distinct winning algorithms (over 10 distinct winning algorithms under any time budget), which suggests that TPOT may try a large number of different algorithms, and some of the algorithms may not be in use by other AutoML systems. This diversity may provide TPOT an edge in its searching of good algorithms and pipelines. Actually, TPOT turns out to be a frequent winning AutoML system in this experiment. Together with AutoGluon, TPOT never falls out of the first two places in terms of winning frequency, and they have significantly more winnings than Auto^{n}ML and H_2O AutoML. For AutoGluon, we can observe that its winning algorithms are very stable across varied time budgets. "WeightedEnsemble_L2" and "CatBoost" are the Top 2 most frequent winning algorithms of AutoGluon and each of their shares is significantly larger than the share of any other winning algorithm. "WeightedEnsemble_L2" is the most frequent winning algorithm under any time budget and its edge over "CatBoost" in frequency is significant, which is consistent to the fact that AutoGluon uses the technique of ensembling several other models to produce its own prediction models. Together with TPOT, AutoGluon is a frequent winning AutoML system over Auto^{n}ML and H_2O AutoML.

TABLE IV

ABSOLUTE FREQUENCY AND PERCENTAGE FREQUENCY OF WINNING ALGORITHM FOR EACH AUTOML SYSTEM OVER EXPERIMENTS OVER N=177 DATASETS (TIME BUDGET = 60 SECONDS)

	Auto^{n}ML	H_2O AutoML	TPOT	AutoGluon	Total
gradient_boosting	14 / 24.14%	0	8 / 12.31%	0	22 / 9.61%
extra_trees	13 / 22.41%	1 / 2.50%	7 / 10.77%	0	21 / 9.17%
ada_boost	9 / 15.52%	0	0	0	9 / 3.93%
sgd	0	0	0	0	0
bagging	5 / 8.62%	0	0	0	5 / 2.18%
mlp	5 / 8.62%	0	7 / 10.77%	0	12 / 5.24%
random_forest	4 / 6.90%	0	6 / 9.23%	0	10 / 4.37%
XGBoost	4 / 6.90%	13 / 32.50%	10 / 15.38%	1 / 1.52%	28 / 12.23%
logistic_regression	4 / 6.90%	0	3 / 4.62%	0	7 / 3.06%
DeepLearning	0	8 / 20.00%	2 / 3.08%	2 / 3.04%	12 / 5.24%
GBM	0	14 / 35.00%	8 / 12.31%	0	22 / 9.61%
GLM	0	1 / 2.50%	0	0	1 / 0.44%
DRF	0	2 / 5.00%	0	0	2 / 0.87%
GaussianNB	0	1 / 2.50%	3 / 4.62 %	0	4 / 1.75%
MultinomialNB	0	0	2 / 3.08%	0	2 / 0.87%
BernoulliNB	0	0	0	0	0
XGBClassifier	0	0	1 / 1.54%	0	1 / 0.44%
DecisionTreeClassifier	0	0	2 / 3.08%	0	2 / 0.87%
WeightedEnsemble_L2	0	0	0	43 / 65.15%	43 / 18.78%
LightGBMLarge	0	0	0	0	0%
CatBoost	0	0	0	17 / 25.76%	17 / 7.42%
LightGBM	0	0	0	1 / 1.52%	1 / 0.44%
KNeighborsDist	0	0	6 / 9.23%	0	6 / 2.62%
LightGBMXT	0	0	0	2 / 3.03%	2 / 0.87%
ToTal	58 / 100.00%	40 / 100.00%	65 / 100.00%	66 / 100.00%	229 / 100.00%

TABLE V

ABSOLUTE FREQUENCY AND PERCENTAGE FREQUENCY OF WINNING ALGORITHM FOR EACH AUTOML SYSTEM OVER EXPERIMENTS OVER N=177 DATASETS (TIME BUDGET = 600 SECONDS)

	Auto^{n}ML	H2O AutoML	TPOT	AutoGluon	Total
gradient_boosting	15 / 26.79%	4 / 9.09%	14 / 20.00%	0	33 / 13.69%
extra_trees	9 / 16.07%	1 / 2.27%	21 / 30.00%	0	31 / 12.86%
ada_boost	5 / 8.93%	0	0	0	5 / 2.07%
sgd	0	0	0	0	0
bagging	5 / 8.93%	0	0	0	5 / 2.07%
mlp	6 / 10.71%	0	10 / 14.29%	0	16 / 6.64%
random_forest	6 / 10.71%	0	4 / 5.71%	0	10 / 4.15%
XGBoost	3 / 5.36%	5 / 11.36%	0	2 / 2.82%	10 / 4.15%
logistic_regression	7 / 12.5%	1 / 2.27%	1 / 1.43%	0	9 / 3.73%
DeepLearning	0	9 / 20.45%	0	4 / 5.63%	13 / 5.39%
GBM	0	22 / 50.00%	0	0	22 / 9.13%
GLM	0	0	0	0	0
DRF	0	0	0	0	0
GaussianNB	0	0	3 / 4.29%	0	3 / 1.24%
MultinomialNB	0	0	1 / 1.43%	0	1 / 0.41%
BernoulliNB	0	0	0	0	0
XGBClassifier	0	0	1 / 1.43%	0	1 / 0.41%
DecisionTreeClassifier	0	1 / 2.27%	3 / 4.29%	0	4 / 1.66%
WeightedEnsemble_L2	0	0	0	43 / 60.56%	43/ 17.84%
LightGBMLarge	0	0	0	0	0
CatBoost	0	0	0	20 / 28.17%	20 / 8.30%
LightGBM	0	0	0	0	0
KNeighborsDist	0	1 / 2.27%	12 / 17.14%	0	13 / 5.39%
LightGBMXT	0	0	0	2 / 2.82%	2 / 0.83%
Total	56 / 100.00%	44 / 100.00%	70 / 100.00%	71 / 100.00%	241 / 100.00%

TABLE VI

ABSOLUTE FREQUENCY AND PERCENTAGE FREQUENCY OF WINNING ALGORITHM FOR EACH AUTOML SYSTEM OVER EXPERIMENTS OVER N=177 DATASETS (TIME BUDGET = 1200 SECONDS)

AutoML	Auto^{n}ML	H2O AutoML	TPOT	AutoGluon	Total
gradient_boosting	13 / 22.41%	0	17 / 23.29%	0	30 / 12.66%
extra_trees	11 / 18.97%	0	20 / 27.40%	0	31 / 13.08%
ada_boost	8 / 13.79%	0	0	0	8 / 3.38%
sgd	0	0	0	0	0
bagging	5 / 8.62%	0	0	0	5 / 2.11%
mlp	6 / 10.34%	0	11 / 15.07%	0	17 / 7.17%
random_forest	4 / 6.90%	0	6 / 8.22%	0	10 / 4.22%
XGBoost	2 / 3.45%	11 / 26.83%	0	2 / 3.08%	15 / 6.33%
logistic_regression	9 / 15.52%	0	1 / 1.37%	0	10 / 4.22%
DeepLearning	0	10 / 24.39%	0	4 / 6.15%	14 / 5.91%
GBM	0	20 / 48.78%	0	0	20 / 8.44%
GLM	0	0	0	0	0
DRF	0	0	0	0	0
GaussianNB	0	0	3 / 4.11%	0	3 / 1.27%
MultinomialNB	0	0	1 / 1.37%	0	1 / 0.42%
BernoulliNB	0	0	0	0	0
XGBClassifier	0	0	1 / 1.37%	0	1 / 0.42%
DecisionTreeClassifier	0	0	3 / 4.11%	0	3 / 1.27%
WeightedEnsemble_L2	0	0	0	37 / 56.92%	18 / 15.61%
LightGBMLarge	0	0	0	0	0
CatBoost	0	0	0	19 / 29.23%	19 / 8.02%
LightGBM	0	0	0	0	0
KNeighborsDist	0	0	10 / 13.70%	0	10 / 4.22%
LightGBMXT	0	0	0	3 / 4.62%	3 / 1.27%
Total	58 / 100.00%	41 / 100.00%	73 / 100.00%	65 / 100.00%	237 / 100.00%

2) Performance Discrepancies versus Core Algorithm Discrepancies: In order to explore whether the performance discrepancies is contributed by the difference in the selection of the core algorithm or not, we pick up one AutoML system, Auto^{n}ML, into study in particular. We focus on the data tasks that Auto^{n}ML did not win. In order to investigate the algorithm distribution on these tasks, we make heat maps of the algorithms used by the winning AutoML versus the algorithms used by Auto^{n}ML. If there is a tie over the Top 1 AutoML system, we take all of them and their core algorithms as well into consideration. The result is shown in Figure 8. The count in each grid denotes the frequencies that Auto^{n}ML did not get the first place with its corresponding algorithm and another AutoML system got the first place with its corresponding algorithm.

Several observations are as below. There is a prominent grid: "WeightedEnsemble_L2"-"gradient_bossting". It has a value of 16, 15, 12 under the time budget of 60 seconds, 600

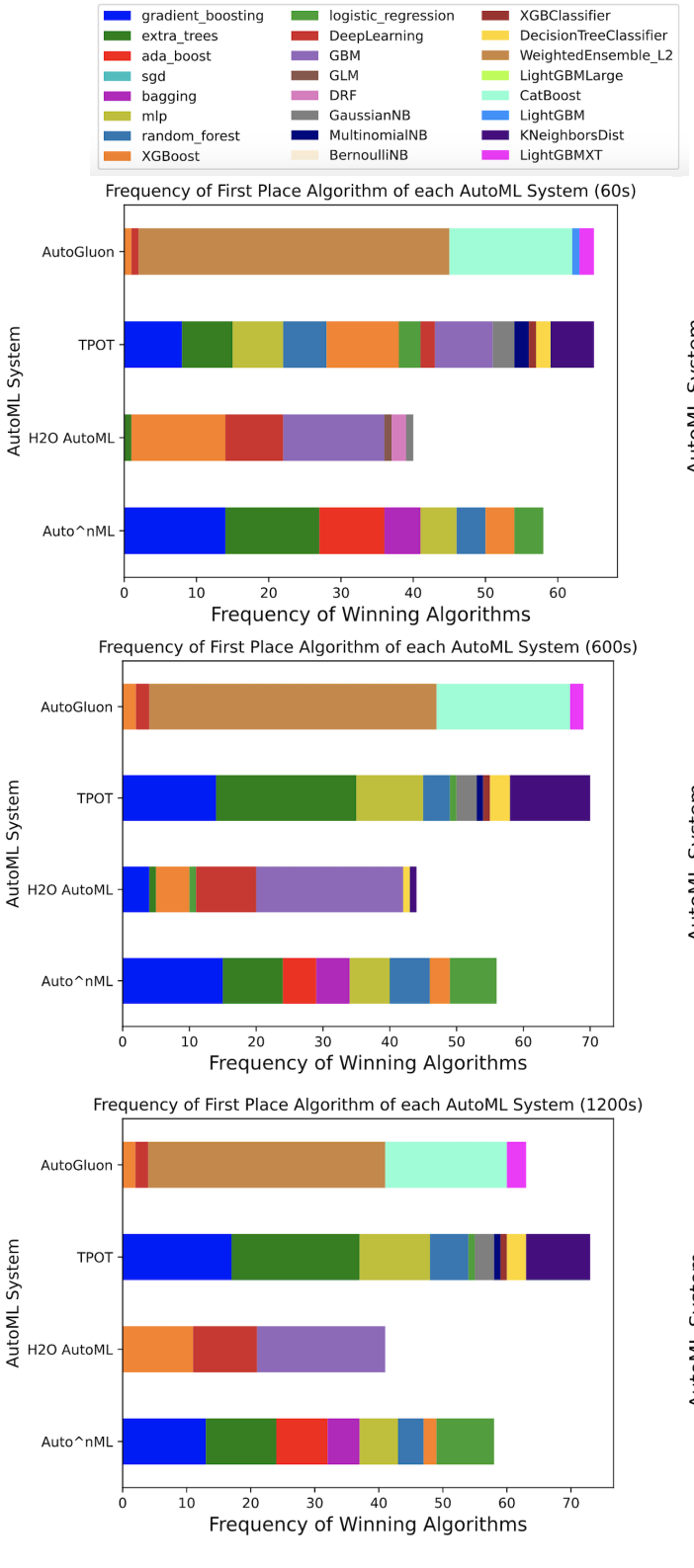

Fig. 6. Frequency of First Place Algorithm of each AutoML

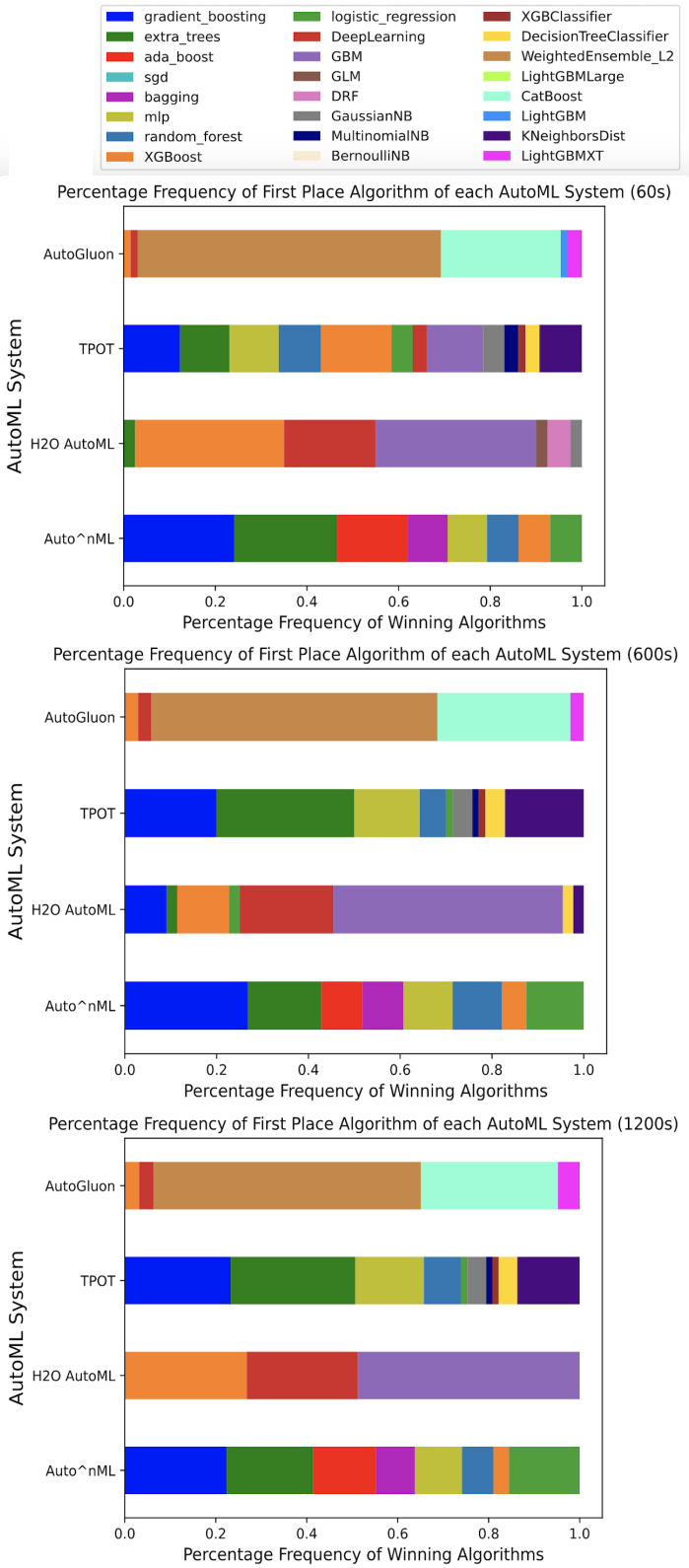

Fig. 7. Percentage Frequency of First Place Algorithm of each AutoML

seconds, and 1200 seconds respectively, which means that there are 16, 15, 12 times when Auto^{n}ML did not win using the algorithm of "gradient_bossting" while another AutoML system won using the algorithm of "WeightedEnsemble_L2" under the time budget of 60 seconds, 600 seconds and 1200 seconds respectively. It shows that Auto^{n}ML has a tendency to utilize the "gradient_boosting" algorithm, but this individual algorithm often failed to beat the ensemble technique. In addition, "WeightedEnsemble_L2" also counts for many times of Auto^{n}ML not getting the first place in total, which are 43, 43 and 37 times respectively under the time budget of 60 seconds, 600 seconds and 1200 seconds. Several other major algorithms that tend to evade Auto^{n}ML as winning algorithms include "DeepLearning", "GBM", "extra_trees" and "gradient_boosting". We noticed that under the time budget of 600 seconds and 1200 seconds, there are 2 additional prominent grids: "extra_trees"-"extra_trees", and "gradient_boosting"-"gradient_boosting". The value of grid "extra_trees"-"extra_trees" are 7 and 5 respectively under the time budget of 600 seconds and 1200 seconds. The value of grid "gradient_boosting"-"gradient_boosting" are 4 and 5 respectively under the time budget of 600 seconds and 1200 seconds. It shows that other AutoML systems may beat Auto^{n}ML by factors other than pipeline core algorithm selection, such as data processing or model parameter selection when the time budget is large.

V. CONCLUSION

We evaluated the performance of 4 AutoML systems on 177 OpenML binary-classification tasks, using AUC as the evaluation metric. We analyzed the experimental data from several aspects, including the relative rankings of the AutoML systems, training-testing performance discrepancies, relationship between the performance of the AutoML systems and the dataset characteristics, and the core algorithms used by each AutoML system that can help it win. We show that AutoGluon achieved the highest average rank among the 4 AutoML systems being evaluated under the time budget of 60 seconds and 600 seconds, while TPOT achieved the highest average rank under the time budget of 1200 seconds. We find that Auto^{n}ML has the strongest correlation between its test prediction score and train prediction score, which indicates better generalization of the predictive models. No definitive conclusion has been found regarding the relationship between the performance of the AutoML systems and dimensionality of the datasets yet. However, there may exist certain relationships between the performance of the AutoML systems and number of instances of the datasets. For example, according to the experimental results on the 177 datasets, the relative performance of AutoGluon goes in an upward trend as the number of instances of the dataset increases, while TPOT generally performs relatively better on smaller datasets than larger ones. Lastly, we showed the absolute frequency and the percentage frequency of winning algorithms of each AutoML system. In addition, we used Auto^{n}ML as an example to study how algorithm selection can contribute to performance discrepancies. We identify

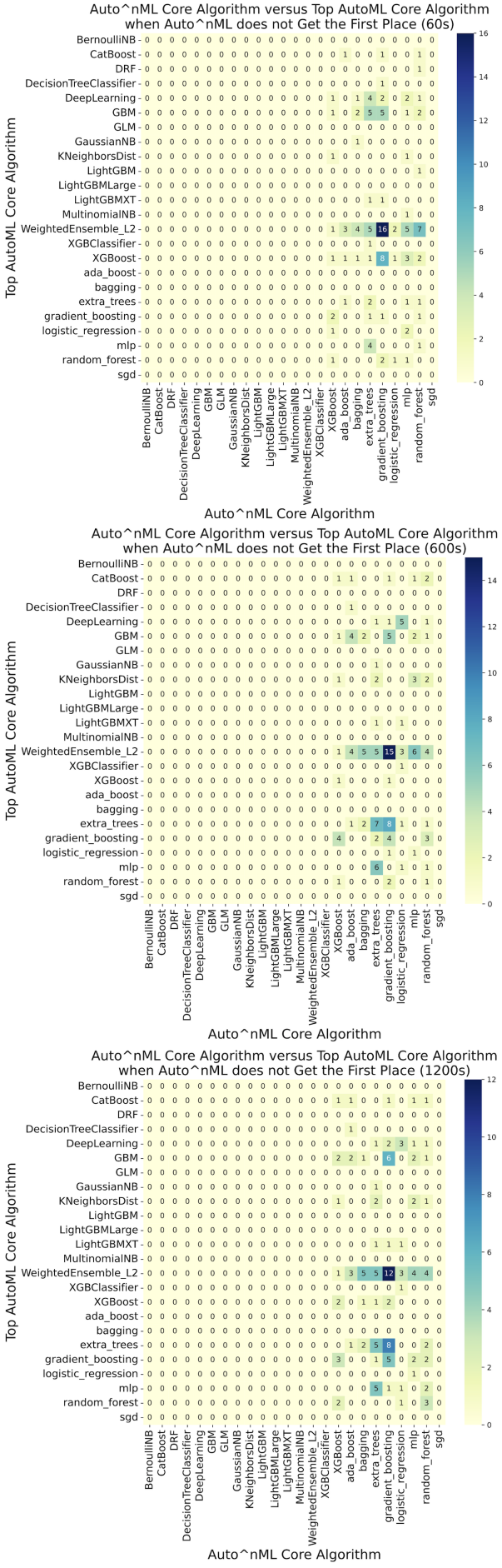

Fig. 8. Heat Maps of Top AutoML Core Algorithm versus Auto^{n}ML Core Algorithm when Auto^{n}ML did not Get the First Place

several algorithms that Auto^{n}ML tend to miss but grabbed by other AutoML systems to get the first place, such as the ensemble technique. We showed that both the pipeline algorithm selection and other factors can contribute to the performance discrepancies among the AutoML systems.

However, our research still has some limitations. For example, our selection of data tasks contains too many datasets with small dimensionalities and number of instances and only a few datasets with large dimensionalities or number of instances. Therefore, the performance lift of AutoGluon over large datasets may not be representative. In the future, we hope to incorporate more datasets with large dimensionality or number of instances into our evaluation process. In addition, we solely focus on OpenML binary-classification tasks for now. In the future, we hope to expand our experiment to multi-class classification tasks and regression tasks to see how the AutoML systems behave in these types of tasks. We also like to test AutoML systems on datasets from sources other than OpenML and observe their behaviors over these tasks.

VI. APPENDIX

Please See Table VII attached for summary of average train prediction scores and test prediction scores of the AutoML systems across time budgets. See Table VIII attached for a description of data tasks that the 4 AutoML systems have been evaluated upon.

VII. ACKNOWLEDGMENT

This work is supported by D3M MILEI funding and Carnegie Mellon Robotics Institute Summer Scholars (RISS) program. The authors would like to thank Rachel Burcin and Dr. John M. Dolan for their organization and support of this program.

REFERENCES

[1] "Auto^{n}ml - taking your ml capacity to the nth power," https://autonlab.org/research/usability.html.

[2] E. LeDell, "H2o automl: Scalable automatic machine learning," 2020.

[3] R. S. Olson and J. H. Moore, "Tpot: A tree-based pipeline optimization tool for automating machine learning," in *Proceedings of the Workshop on Automatic Machine Learning*, ser. Proceedings of Machine Learning Research, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., vol. 64. New York, New York, USA: PMLR, 24 Jun 2016, pp. 66–74.

[4] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "Autogluon-tabular: Robust and accurate automl for structured data," 2020. [Online]. Available: https://arxiv.org/abs/2003.06505

[5] M.-A. Zöller and M. F. Huber, "Benchmark and survey of automated machine learning frameworks," 2019. [Online]. Available: https://arxiv.org/abs/1904.12054

[6] L. Ferreira, A. Pilastri, C. M. Martins, P. M. Pires, and P. Cortez, "A comparison of automl tools for machine learning, deep learning and xgboost," pp. 1–8, 2021.

[7] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of automl approaches and tools," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, pp. 1471–1479.

[8] M. Milutinovic, B. Schoenfeld, D. Martinez-Garcia, S. Ray, S. Shah, and D. Yan, "On evaluation of automl systems," in *Proceedings of the ICML Workshop on Automatic Machine Learning*, 2020.

TABLE VII

AVERAGED TRAIN PREDICTION SCORE AND TEST PREDICTION SCORE OF THE AUTOML SYSTEMS ACROSS TIME BUDGETS

	OpenML ID	Auto^{n}ML Train	Auto^{n}ML Test	H_2O Train	H_2O Test	TPOT Train	TPOT Test	AutoGluon Train	AutoGluon Test
0	1013	0.7254	0.4167	0.7647	0.6919	0.9035	0.4621	0.9	0.505
1	823	0.9984	0.9983	0.9999	0.9986	0.9662	0.967	0.9993	0.9987
2	799	0.9634	0.9647	0.9965	0.9654	0.9903	0.9617	0.9735	0.9701
3	1004	1.0	1.0	0.9998	1.0	1.0	0.9992	1.0	1.0
4	842	0.7298	0.88	0.7618	0.68	0.9467	0.8133	0.85	0.8
5	1006	0.9131	0.9662	0.9618	0.9255	0.9943	0.8407	1.0	0.9676
6	737	0.9255	0.9176	0.9548	0.9163	0.9845	0.9333	0.9279	0.9307
7	740	0.9579	0.9786	0.9954	0.9786	0.9984	0.982	0.9751	0.9812
8	1220	0.6927	0.7053	0.7387	0.7062	0.6733	0.6709	0.7055	0.7159
9	757	0.8819	0.8484	0.981	0.8342	0.9997	0.846	0.974	0.8308
10	792	0.9743	0.9858	0.9945	0.9792	0.9974	0.9888	0.9977	0.9841
11	1011	0.9862	0.9846	0.9997	0.9861	0.9963	0.9799	1.0	0.9801
12	803	0.9774	0.982	0.9988	0.9812	0.9922	0.9836	0.979	0.9848
13	13	0.6388	0.6707	0.9321	0.6651	0.7727	0.7434	0.8602	0.7536
14	15	0.9915	0.9977	0.9933	0.9975	0.9967	0.9977	1.0	0.9981
15	37	0.8203	0.8548	0.836	0.8367	0.8941	0.8619	0.8368	0.8747
16	43	0.6639	0.7668	0.7119	0.6966	0.718	0.7412	0.7966	0.7632
17	50	0.9967	0.9997	1.0	1.0	1.0	1.0	1.0	1.0
18	333	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
19	334	0.9998	1.0	1.0	0.9972	1.0	1.0	1.0	1.0
20	335	0.9934	0.9877	0.9987	0.9755	0.9998	0.9936	0.9959	0.9812
21	346	0.7182	0.8	0.6404	0.6222	0.797	0.8333	0.7333	0.8667
22	444	0.7301	0.863	0.9161	0.8296	0.8863	0.9346	0.8889	0.8704
23	448	0.9378	0.6625	0.9986	0.7392	0.9854	0.6775	1.0	0.59
24	450	0.9951	0.9935	0.9678	0.9738	1.0	0.9869	1.0	0.9803
25	451	0.9914	0.9893	0.9965	0.9943	0.9961	0.9979	0.9978	0.994
26	464	0.9221	0.9645	0.9258	0.9655	0.9666	0.9746	0.9583	0.9746
27	472	0.845	0.775	0.9582	0.7597	0.8792	0.7833	1.0	0.8083
28	476	0.9697	1.0	0.996	0.9889	0.9106	0.9	1.0	0.9667
29	479	0.8833	0.9296	0.9853	0.9	0.9455	0.9222	0.9091	0.9333
30	949	0.8371	0.734	0.8098	0.7099	0.967	0.7974	0.8531	0.7814
31	1037	0.8994	0.9108	0.9439	0.9101	0.9161	0.9071	0.9177	0.9095
32	1566	0.8232	0.9103	0.9984	0.9976	1.0	0.9978	0.9989	0.987
33	744	0.9282	0.9465	0.9989	0.9674	1.0	0.9571	0.93	0.93
34	1558	0.9065	0.8795	0.9966	0.8801	0.959	0.8831	0.9181	0.8934
35	1024	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
36	23499	0.7153	0.6951	0.7968	0.6668	0.8403	0.6638	0.9853	0.6873
37	1167	0.6303	0.6833	0.7449	0.649	0.6531	0.6822	0.6182	0.6829
38	1511	0.9627	0.9673	0.9826	0.9551	0.9895	0.9563	0.9778	0.9704
39	1524	0.9155	0.9289	0.9784	0.8958	0.9419	0.9449	0.9415	0.9067
40	890	0.8823	0.9013	0.8869	0.7632	1.0	0.9013	1.0	0.8092
41	1455	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
42	1473	0.5644	0.5298	0.925	0.6825	0.8971	0.5595	0.8846	0.5833
43	1463	0.9202	0.7833	0.969	0.8122	0.9718	0.6366	0.9773	0.77
44	1495	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
45	40714	0.1806	0.1562	0.5648	0.3542	0.9722	1.0	0.6667	0.8125
46	41538	0.5032	0.6824	0.7632	0.4881	0.9816	0.5528	0.5189	0.5702
47	42638	0.8712	0.835	0.9318	0.8269	0.9672	0.8385	0.9233	0.8436
48	40669	1.0	1.0	0.9994	1.0	1.0	1.0	1.0	1.0
49	40681	1.0	1.0	0.9996	0.9961	1.0	1.0	1.0	1.0
50	40690	0.9999	1.0	0.9999	0.9979	1.0	1.0	1.0	0.9993
51	724	0.9229	0.9325	0.9965	0.8941	0.9965	0.9396	0.9694	0.9502
52	731	0.8201	0.6806	0.802	0.6829	0.8892	0.6875	0.9643	0.6806
53	729	0.9308	1.0	0.9962	0.9762	1.0	1.0	1.0	1.0
54	730	0.9847	0.9934	0.9998	0.9626	0.9997	0.9926	1.0	0.9778
55	726	0.8587	0.8929	0.9393	0.8052	0.9995	0.9416	0.9444	0.8279
56	767	0.9533	0.9372	0.988	0.9048	0.9699	0.9138	0.9984	0.9369
57	764	0.9157	0.9609	0.9909	0.9112	0.9816	0.9316	0.9604	0.9276
58	765	0.9783	0.931	0.9885	0.9439	0.9925	0.9569	0.9344	0.9057
59	790	0.9601	0.8958	0.994	0.9028	0.994	0.9306	1.0	0.9167
60	795	0.5192	0.5002	0.7443	0.5366	0.6927	0.4993	0.632	0.4597
61	865	0.6932	0.3542	0.6836	0.2917	0.8164	0.3542	0.9286	0.3958
62	864	0.7268	0.821	0.8446	0.8766	0.9372	0.3457	0.85	0.7037
63	867	0.8188	0.858	0.9516	0.9115	0.9442	0.8786	0.8906	0.821
64	899	0.9446	0.9077	0.9163	0.8385	0.9807	0.8692	1.0	0.8923
65	905	0.7778	1.0	0.8768	0.8571	0.9555	0.9841	0.875	0.9524
66	900	0.5454	0.516	0.8533	0.5537	0.9411	0.4286	0.6742	0.476
67	942	0.7135	0.5119	0.6364	0.4444	0.8523	0.4683	0.875	0.5476
68	944	0.8297	0.7981	0.9898	0.7691	0.8988	0.7883	0.899	0.7963
69	945	0.8858	0.8846	0.9375	0.7778	0.9979	0.8632	1.0	0.8077

	OpenML ID	Auto^{n}ML Train	Auto^{n}ML Test	H_2O Train	H_2O Test	TPOT Train	TPOT Test	AutoGluon Train	AutoGluon Test
70	946	0.6668	0.4583	0.9128	0.468	0.9324	0.5639	0.4694	0.4625
71	967	0.9763	0.9704	1.0	0.969	0.9999	0.9762	0.9748	0.9733
72	961	0.8487	0.8457	0.8514	0.8066	0.9693	0.8719	0.8489	0.8634
73	968	0.9525	0.8447	0.9495	0.8724	0.9702	0.8773	0.9758	0.8835
74	960	0.3239	0.6029	0.6867	0.4788	0.7291	0.4396	0.625	0.6324
75	996	0.9032	0.9276	0.9999	0.8947	0.999	0.8975	0.95	0.9227
76	997	0.9988	1.0	0.9985	0.9961	1.0	0.9994	0.9995	0.9984
77	1025	0.9703	0.9413	0.9885	0.9575	0.8562	0.8056	0.9836	0.9174
78	739	0.7202	0.7167	0.8919	0.6556	0.9943	0.5389	0.8	0.8167
79	733	0.9835	0.9889	0.9974	0.9947	0.9917	0.9894	1.0	0.9884
80	784	0.8509	0.8881	0.9938	0.8928	0.9749	0.8666	0.9636	0.8462
81	777	0.9065	0.9444	0.9779	0.9259	0.9773	0.8704	0.9167	0.8889
82	782	0.9861	1.0	0.998	1.0	0.9993	1.0	1.0	1.0
83	875	0.9435	1.0	0.9942	0.9854	0.9948	0.9941	1.0	0.8669
84	916	0.9113	0.7564	0.8914	0.7233	1.0	0.8354	0.9815	0.7821
85	895	0.9197	0.9473	0.9655	0.939	0.9723	0.9275	0.9451	0.9395
86	974	0.9317	0.9925	0.9579	0.9883	0.9641	0.9908	0.9394	0.94
87	754	0.9538	0.92	0.8353	0.87	0.9939	0.9444	1.0	0.8733
88	811	0.9823	0.9315	0.9964	0.9353	0.9971	0.8964	1.0	0.9522
89	747	0.9983	0.9653	0.9962	0.9769	0.9998	0.9816	1.0	0.9918
90	714	0.6244	0.377	0.6515	0.5192	0.7269	0.4584	0.7024	0.5397
91	955	0.7779	0.9	0.7088	0.7949	0.9993	0.8082	0.8917	0.7508
92	748	0.8248	0.9106	0.9975	0.8646	0.8386	0.8348	0.881	0.7545
93	719	0.8019	0.6957	0.9956	0.5749	1.0	0.6981	0.8556	0.558
94	1075	0.7375	0.8125	0.9088	0.5417	0.8935	0.2708	0.7778	0.2812
95	814	0.9262	0.9408	0.9998	0.9349	0.9995	0.9502	0.9744	0.9359
96	776	0.9377	0.8976	0.9999	0.9231	1.0	0.9526	0.9111	0.9127
97	911	0.9439	0.9628	1.0	0.9449	0.9997	0.9619	0.9972	0.951
98	886	0.6847	0.6871	0.9899	0.6272	0.9873	0.6494	0.7956	0.6812
99	796	0.9995	1.0	0.9976	0.9989	1.0	1.0	1.0	1.0
100	774	0.5631	0.5652	0.6265	0.5311	0.5243	0.4977	0.5797	0.5251
101	893	0.641	0.9167	0.6116	0.6556	0.884	0.5722	0.7667	0.8778
102	906	0.5293	0.4084	0.9256	0.4671	0.9992	0.433	0.5787	0.5392
103	884	0.9595	0.9262	0.9959	0.9373	0.9998	0.9624	0.9502	0.9654
104	894	0.9931	1.0	0.9989	1.0	0.9948	1.0	1.0	1.0
105	770	0.9995	1.0	1.0	1.0	1.0	1.0	1.0	1.0
106	870	0.9634	0.9612	0.9993	0.9438	0.9953	0.9731	0.9879	0.9449
107	749	0.9652	0.9463	0.9962	0.9764	0.9979	0.9858	0.9915	0.9799
108	1014	0.5554	0.5447	0.7281	0.4906	0.6218	0.4956	0.6293	0.4792
109	947	0.9042	0.869	0.9998	0.9373	0.9892	0.9141	0.9095	0.8142
110	841	0.9938	0.9996	0.9998	0.9987	0.9998	0.9995	0.998	0.9992
111	1005	0.8954	0.9041	0.9984	0.8602	1.0	0.9243	0.9307	0.8612
112	950	0.9252	0.9988	0.9355	0.915	0.9997	0.9946	1.0	0.9975
113	907	0.5073	0.4803	0.8244	0.5547	0.639	0.4565	0.6433	0.5821
114	874	1.0	0.9545	1.0	1.0	1.0	0.9545	1.0	1.0
115	750	0.6023	0.7786	0.7227	0.6032	1.0	0.6641	0.7037	0.6636
116	848	0.898	0.7619	1.0	0.7381	1.0	0.4444	1.0	0.6429
117	1049	0.9381	0.9435	0.9944	0.9332	0.9939	0.9381	0.9676	0.951
118	847	0.943	0.9336	0.9661	0.9363	0.9684	0.9338	0.9621	0.9387
119	316	0.8581	0.9503	0.9922	0.8662	0.9589	0.9269	0.9782	0.903
120	910	0.9734	0.981	0.9988	0.9789	0.9998	0.9819	0.9834	0.9809
121	904	0.9394	0.9653	0.9998	0.9608	0.9983	0.9558	0.9586	0.9628
122	930	0.8049	0.8502	0.9014	0.8346	0.9572	0.8553	0.8582	0.8546
123	958	1.0	0.9998	1.0	0.9982	1.0	0.9997	1.0	0.9999
124	1019	0.9998	0.9998	0.9998	0.9991	1.0	0.998	1.0	0.9997
125	723	0.9558	0.9662	0.9992	0.965	1.0	0.9778	0.9703	0.9602
126	734	0.9562	0.9559	0.9913	0.9593	0.9561	0.9529	0.9596	0.9599
127	4154	0.9586	0.8352	0.9971	0.7659	0.9996	0.8235	1.0	0.6971
128	1056	0.9387	0.8858	0.9849	0.8327	0.9985	0.8941	0.9946	0.9173
129	1002	0.8469	0.863	0.9027	0.8672	0.9141	0.8745	0.8622	0.8754
130	845	0.9522	0.9437	0.9972	0.9434	0.9994	0.9624	0.9753	0.9582
131	1020	0.9979	0.9994	0.9992	0.9981	1.0	0.9998	1.0	0.9997
132	971	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
133	44	0.9851	0.9886	0.9992	0.989	0.9978	0.9869	0.9897	0.9889
134	979	0.9599	0.9654	0.9992	0.958	1.0	0.9667	0.9758	0.965
135	718	0.961	0.9309	0.9993	0.9396	0.9979	0.9631	0.9895	0.9379
136	715	0.9707	0.9672	0.9999	0.9698	0.999	0.9758	0.9875	0.9745
137	761	0.9817	0.9821	0.996	0.9828	0.9985	0.9829	0.986	0.9854
138	1453	0.8137	0.8574	0.9871	0.7733	0.9878	0.858	0.8369	0.8397
139	821	0.948	0.9536	0.9861	0.9576	0.9339	0.9336	0.9599	0.9608

	OpenML ID	Auto^{n}ML Train	Auto^{n}ML Test	H_2O Train	H_2O Test	TPOT Train	TPOT Test	AutoGluon Train	AutoGluon Test
140	31	0.7937	0.7683	0.9925	0.7483	0.9871	0.7363	0.8413	0.7633
141	1504	1.0	1.0	1.0	0.9998	1.0	1.0	1.0	1.0
142	849	0.9511	0.9523	0.9992	0.944	0.995	0.9619	0.9657	0.9596
143	1068	0.8455	0.897	0.871	0.8038	0.9959	0.9118	0.87	0.8849
144	1022	0.999	0.9983	0.9999	0.9973	1.0	0.9998	1.0	0.9997
145	981	0.9484	0.9498	0.9891	0.9581	0.9567	0.9423	0.9637	0.9603
146	1487	0.9165	0.919	0.99	0.9348	1.0	0.9387	0.913	0.9444
147	1471	0.9789	0.988	0.9998	0.9934	1.0	0.9956	0.998	0.9981
148	995	0.9986	0.9998	0.9984	0.998	1.0	0.9994	1.0	0.9978
149	143	0.9951	0.9956	0.9966	0.9964	0.9943	0.9945	0.9972	0.9964
150	3	0.9985	0.9995	1.0	0.9998	0.9999	0.9996	1.0	0.9999
151	1496	0.9961	0.9965	1.0	0.9967	0.9978	0.9977	0.9943	0.9968
152	1461	0.9241	0.9257	0.9672	0.9326	0.9287	0.901	0.9404	0.9366
153	751	0.9629	0.9658	0.9984	0.9458	0.9965	0.9676	0.9822	0.9532
154	1067	0.8164	0.8264	0.9642	0.8085	0.9867	0.8427	0.7992	0.826
155	722	0.9985	0.999	1.0	0.9995	0.9977	0.9936	0.9994	0.9997
156	802	0.8737	0.8971	0.999	0.9143	0.9999	0.9117	0.8875	0.9228
157	1547	0.7126	0.698	0.7162	0.6886	0.8978	0.6772	0.6661	0.7274
158	913	0.9683	0.9872	1.0	0.9859	1.0	0.9888	0.9963	0.9905
159	976	0.9994	0.9998	0.9996	0.9993	1.0	0.9999	0.9997	0.9999
160	953	0.9936	0.9915	0.9999	0.9892	1.0	0.9896	0.9922	0.9908
161	993	0.993	0.9886	0.9998	0.99	0.9979	0.9897	0.9941	0.9902
162	752	0.9571	0.9566	0.9987	0.9602	0.9683	0.9496	0.9678	0.9691
163	1018	0.8942	0.9003	0.9079	0.8999	0.9496	0.8982	0.9015	0.8989
164	1050	0.8425	0.8559	0.9725	0.8293	0.977	0.8485	0.85	0.8354
165	797	0.9586	0.9657	0.9996	0.9571	1.0	0.9646	0.9779	0.9663
166	806	0.9589	0.9642	0.9999	0.9577	1.0	0.9678	0.9565	0.9563
167	866	0.9631	0.9759	0.9985	0.9663	0.9996	0.9784	0.9671	0.9657
168	837	0.9703	0.9779	1.0	0.9785	1.0	0.9819	0.984	0.9804
169	897	0.9996	0.9994	1.0	0.998	1.0	0.9996	1.0	0.9987
170	903	0.978	0.967	0.9995	0.9632	0.9998	0.9764	0.9767	0.9609
171	1494	0.9233	0.9347	0.9887	0.9172	0.9936	0.9346	0.9306	0.9309
172	917	0.9646	0.9749	0.9999	0.9782	0.9989	0.9755	0.982	0.9785
173	983	0.7576	0.7871	0.9074	0.7754	0.8441	0.7892	0.7925	0.7934
174	977	0.9998	0.9998	1.0	0.9999	0.9999	0.9998	1.0	1.0
175	1021	0.9909	0.9921	0.9966	0.9925	0.9988	0.9943	0.9942	0.9956
176	980	0.9983	0.9997	0.9999	0.9986	1.0	0.9999	1.0	0.9999

TABLE VIII

DATA TASKS DESCRIPTION

	OpenML ID	Computing Machine	No. of Features	No. of Instances
0	1013	Lab Server	3	138
1	823	Lab Server	9	20640
2	799	Lab Server	6	1000
3	1004	Lab Server	61	600
4	842	Lab Server	11	60
5	1006	Lab Server	19	148
6	737	Lab Server	7	3107
7	740	Lab Server	11	1000
8	1220	Lab Server	10	39948
9	757	Lab Server	22	528
10	792	Lab Server	6	500
11	1011	Lab Server	8	336
12	803	Lab Server	6	7129
13	13	Desktop	10	286
14	15	Desktop	10	699
15	37	Desktop	9	768
16	43	Desktop	4	306
17	50	Desktop	10	958
18	333	Desktop	7	556
19	334	Desktop	7	601
20	335	Desktop	7	554
21	346	Desktop	5	50
22	444	Desktop	4	132
23	448	Desktop	4	120
24	450	Desktop	5	264
25	451	Desktop	6	500
26	464	Desktop	3	250
27	472	Desktop	4	87
28	476	Desktop	6	50
29	479	Desktop	10	92
30	949	Desktop	5	559
31	1037	Desktop	15	4562
32	1566	Desktop	101	1212
33	744	Desktop	6	250
34	1558	Desktop	17	4521
35	1024	Desktop	35	2796
36	23499	Desktop	10	277
37	1167	Desktop	9	320
38	1511	Desktop	9	440
39	1524	Desktop	7	310
40	890	Desktop	8	108
41	1455	Desktop	7	120
42	1473	Desktop	10	100
43	1463	Desktop	6	100
44	1495	Desktop	7	250
45	40714	Desktop	6	32
46	41538	Desktop	7	246
47	42638	Desktop	8	891
48	40669	Desktop	7	160
49	40681	Desktop	7	128
50	40690	Desktop	10	512
51	724	Desktop	4	468
52	731	Desktop	5	96
53	729	Desktop	4	44
54	730	Desktop	6	250
55	726	Desktop	6	100
56	767	Desktop	4	475
57	764	Desktop	4	450
58	765	Desktop	4	475
59	790	Desktop	3	55

	OpenML ID	Computing Machine	No. of Features	No. of Instances
60	795	Desktop	4	662
61	865	Desktop	3	100
62	864	Desktop	8	60
63	867	Desktop	3	130
64	899	Desktop	6	92
65	905	Desktop	3	39
66	900	Desktop	7	400
67	942	Desktop	4	50
68	944	Desktop	10	130
69	945	Desktop	7	76
70	946	Desktop	3	88
71	967	Desktop	9	406
72	961	Desktop	8	285
73	968	Desktop	4	365
74	960	Desktop	9	90
75	996	Desktop	10	214
76	997	Desktop	5	625
77	1025	Desktop	6	400
78	739	Desktop	8	62
79	733	Desktop	7	209
80	784	Desktop	4	140
81	777	Desktop	8	47
82	782	Desktop	3	120
83	875	Desktop	4	100
84	916	Desktop	6	100
85	895	Desktop	3	222
86	974	Desktop	5	132
87	754	Desktop	6	100
88	811	Desktop	3	264
89	747	Desktop	5	167
90	714	Desktop	5	125
91	955	Desktop	6	151
92	748	Desktop	6	163
93	719	Desktop	8	137
94	1075	Desktop	9	130
95	814	Desktop	3	468
96	776	Desktop	6	250
97	911	Desktop	6	250
98	886	Desktop	8	500
99	796	Desktop	8	209
100	774	Desktop	4	662
101	893	Desktop	6	73
102	906	Desktop	8	400
103	884	Desktop	6	500
104	894	Desktop	6	66
105	770	Desktop	7	625
106	870	Desktop	6	500
107	749	Desktop	6	500
108	1014	Desktop	5	797
109	947	Desktop	5	559
110	841	Desktop	10	950
111	1005	Desktop	10	214
112	950	Desktop	5	559
113	907	Desktop	8	400
114	874	Desktop	6	50
115	750	Desktop	8	500
116	848	Desktop	6	38
117	1049	NYU	38	1458
118	847	NYU	15	6574
119	316	NYU	117	2417

	OpenML ID	Computing Machine	No. of Features	No. of Instances
120	910	NYU	11	1000
121	904	NYU	51	1000
122	930	NYU	34	1302
123	958	NYU	20	2310
124	1019	NYU	17	10992
125	723	NYU	26	1000
126	734	NYU	41	13750
127	4154	NYU	31	14240
128	1056	NYU	39	9466
129	1002	NYU	56	7485
130	845	NYU	11	1000
131	1020	NYU	65	2000
132	971	NYU	77	2000
133	44	NYU	58	4601
134	979	NYU	41	5000
135	718	NYU	101	1000
136	715	NYU	26	1000
137	761	NYU	22	8192
138	1453	NYU	38	1077
139	821	NYU	17	22784
140	31	NYU	21	1000
141	1504	NYU	34	1941
142	849	NYU	26	1000
143	1068	NYU	22	1109
144	1022	NYU	241	2000
145	981	NYU	69	10108
146	1487	NYU	73	2534
147	1471	NYU	15	14980
148	995	NYU	48	2000
149	143	NYU	17	131072
150	3	NYU	37	3196
151	1496	NYU	21	7400
152	1461	NYU	17	45211
153	751	NYU	11	1000
154	1067	NYU	22	2109
155	722	NYU	49	15000
156	802	NYU	19	1945
157	1547	NYU	21	1000
158	913	NYU	11	1000
159	976	NYU	15	9961
160	953	NYU	61	3190
161	993	NYU	61	7019
162	752	NYU	33	8192
163	1018	NYU	57	8844
164	1050	NYU	38	1563
165	797	NYU	51	1000
166	806	NYU	51	1000
167	866	NYU	51	1000
168	837	NYU	51	1000
169	897	NYU	16	1161
170	903	NYU	26	1000
171	1494	NYU	42	1055
172	917	NYU	26	1000
173	983	NYU	10	1473
174	977	NYU	17	20000
175	1021	NYU	11	5473
176	980	NYU	65	5620