AutoML benchmark
Minimalistic Implementation vs Popular Frameworks

Mikołaj Gałkowski
Faculty of Mathematics and Information Science
Warsaw University of Technology

Kacper Kurowski
Faculty of Mathematics and Information Science
Warsaw University of Technology

Hubert Bujakowski
Faculty of Mathematics and Information Science
Warsaw University of Technology

June 9, 2022

Abstract

Machine learning has been a very successful field of study with many of its accomplishments already being put to commercial use. The ever-growing demand for ML solutions has inspired many researches to develop AutoML methods, which allow users to create predictive models with minimal expertise and experience.

In this article we explore whether minimalistic framework can achieve satisfactory results, when compared to the well-established ones, especially, *AutoGluon*. We conducted the benchmark on 5 AutoML systems across 15 datasets and analyze the results.

Keywords AutoML · AutoGluon · OpenML

1 Introduction

In recent years, we have seen an exponential growth in the field of machine learning. This is due to people who prepare the data, select the right variables and models, then optimize the models and critically analyze the results. Because the complexity of these tasks is often beyond the reach of non-experts in machine learning, the rapid growth of machine learning applications has created a demand for off-the-shelf machine learning methods that can be used without expertise.

In response to these needs AutoML provides methods that overcome the barrier posed by machine learning to non-professionals/scientists, so that they are able to use ML with little input. It also helps accelerate the research through methods i.e. ensembling, hyperparameter tuning, feature engineering, data preprocessing or data splitting.

Since the proposal in [Thakur and Krohn-Grimberghe, 2015] AutoML has been a quickly-developing research area aimed at the progressive automation of machine learning. Due to the vast amount of already developed frameworks and the need for the researchers to share their results in the subject, in the upcoming months the first international AutoML conference[1] is scheduled to be held in Baltimore, USA. One can think of it as a milestone for the AutoML as a research area.

In this article, we will focus on binary classification in tabular data. We have tested the capabilities of the AutoML framework *AutoGluon* [Erickson et al., 2020] and our own implementation, which we have created based on the gathered experience. We have benchmarked the two against three popular AutoML solutions treating the work [Gijsbers et al., 2019] as a guideline. Therefore, much like in the mentioned paper, we have decided to use a subset of datasets available on the Open ML website[2]. See [Vanschoren et al., 2013] for the description of the website and its mission.

[1]`https://automl.cc/`
[2]`https://www.openml.org/`

2 Framework description

We believe the design of an AutoML framework should adhere to the following principles:

- Simplicity – user can train a model on the raw data directly without knowing the details about the data and ML models
- Predictable Timing – returns the results within the time-budget specified by users
- Fault Tolerance – training can be stopped and resumed at any time
- Robustness – framework can handle a large variety of structured datasets and ensures training succeeds even when some of the individual ML models fail

AutoGluon mostly adheres to all of the aforementioned principles. It should be mentioned though that the training cannot be stopped while the program is running. However, once created, the model can be trained further thus, hopefully, improving its performance.

AutoGluon does almost all the necessary preprocessing.

2.1 The `fit()` API

Consider a structured dataset of raw values stored in a CSV file, `train.csv`, with the label values to predict stored in a column named — `class`. Three lines of code are all that's needed to train and test a model with *AutoGluon*:

Within the call `.fit()`, *AutoGluon* automatically:

- Determines the type of each variable
- Handles common preprocessing problems such as missing data and rescaling the values of individual variables
- Identifies what type of prediction problem this is (binary, multi-class classification or regression)
- Partitions the data into various folds for model-training vs. validation
- Individually fits various models, and finally creates an optimized model ensemble that outperforms any of the individual trained model

AutoGluon also automates the process of hyperparameterization of individual model (which the user would have to define in advance).

2.2 Initial settings

AutoGluon comes with a variety of presets that can be specified in the call to `.fit()` via the presets argument.

- `best_quality` – when accuracy is what matters
- `high_quality` – better than good_quality
- `good_quality` – significantly better than medium_quality
- `medium_quality` – initial prototyping, establishing a performance baseline

The last preset mentioned is the default one. The rest of them require more memory and take much longer time to develop. Therefore, it is recommended to gradually change these settings in order to improve the evaluation and to determine if the model is able to learn within a certain time frame.

3 Our Framework

The framework prepared by us is used to solve binary classification problems. We have decided to follow the *KISS principle*[3]. Therefore, our implementation is minimalistic, while still allowing the user to achieve satisfactory results.

[3] `https://www.wikiwand.com/en/KISS_principle`

3.1 Preprocessing

Our implementation includes creating the pipeline, which divides variables into categorical and numeric ones. Then according to the given type pipeline imputates missing data by using *KNNeighbours* for categorical variables, *Mean* for numerical variables and removes extreme numerical values. Our pipeline also scales numeric variables and does *OneHotEncoding* for categorical ones.

3.2 Models

Our framework consists of 5 models: *Logistic Regression*, *Random Forest Classifier*, *XGBClassifier*, *SVC* and *Voting Classifier*. The last one performs a *soft vote*, i.e. it returns the probability of the given class by averaging the probabilities predicted by the former four.

3.3 Metrics

The framework calculates 5 metrics: *ROC AUC score*, *F1 score*, *recall*, *precision* and *accuracy*. During fitting, framework saves them into the common leaderboard. Best model is selected based on the ROC AUC score. However, using the leaderboard, one can select other model treating different metric as the preferred one.

4 Benchmark

In order to determine the usefulness of our implementation we have decided to compare its results against *Autogluon* and three other frameworks: *autosklearn*, *autoweka* and *h2oautoml*. We have selected 15 datasets from the pool present in [Gijsbers et al., 2019] for the binary prediction task.

4.1 Methodology

We have performed 10-fold cross-validation using the function *StratifiedKFold* from *sklearn* package so that the results of the benchmark could be compared to the ones from [Gijsbers et al., 2019].

While most of the preprocessing was automatically done by *AutoGluon*, we had to transform columns which, as per the framework's claim, contained bytes. We have fixed the issue by decoding any instances of **b'string'** into the usual **string**, so that the previously problematic columns were properly detected as having `object` type.

4.2 Results

The effectiveness of the frameworks was evaluated using the AUC metric. The obtained values were then averaged. The final results are presented in Figure 1. For many datasets *AutoGluon* is one of the leaders. Comparing the results obtained by our framework to those of *AutoGluon* we can see that our framework performs very well, considering its rather simple architecture (Figure 2).

5 Conclusion

In conclusion, AutoML is a very fast-growing field that is beginner-friendly due to usage-simplicity of most of the well-established frameworks. Moreover, experts also may find it useful, because many of the available solutions offer highly customizable interface, which could speed up their work/research. Modern AutoML frameworks often consist of complex architectures which provide hyperparameter optimization, relevant preprocessing and feature selection.

Nevertheless, we have managed to prove that simple implementations can sometimes perform at a similar level to complex frameworks. Still, one can expect that more robust solutions will perform better on advanced tasks.

It is worth keeping in mind though that the power lies in human hands, so educated data scientists will still not be replaced by AutoML. Although the actual tasks of the data scientists may change in the future due to the AutoML, which can potentially automize their work.

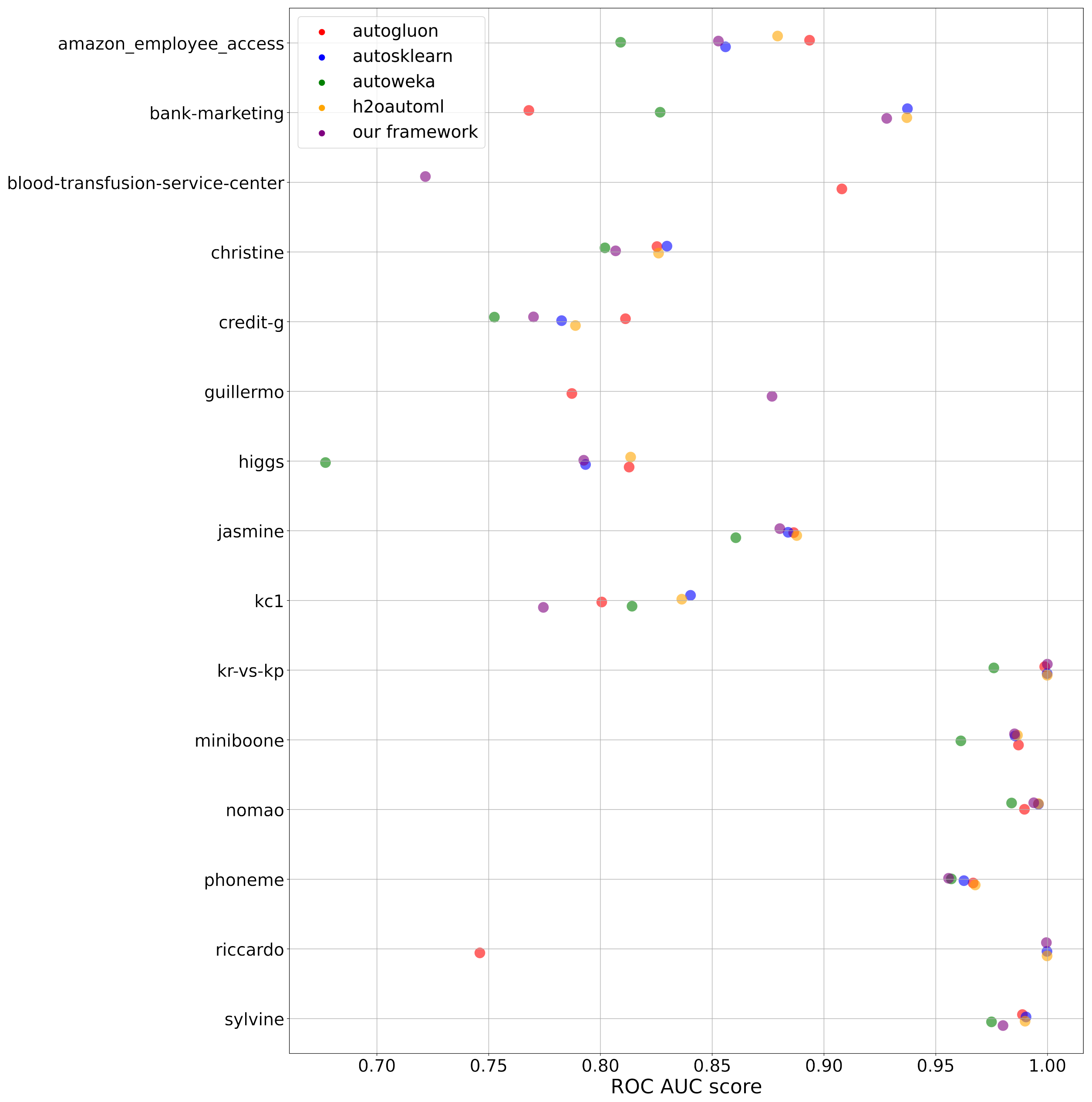

Figure 1: Average values of ROC_AUC metric for the binary prediction task including 3 OpenML frameworks, AutoGluon and our framework.

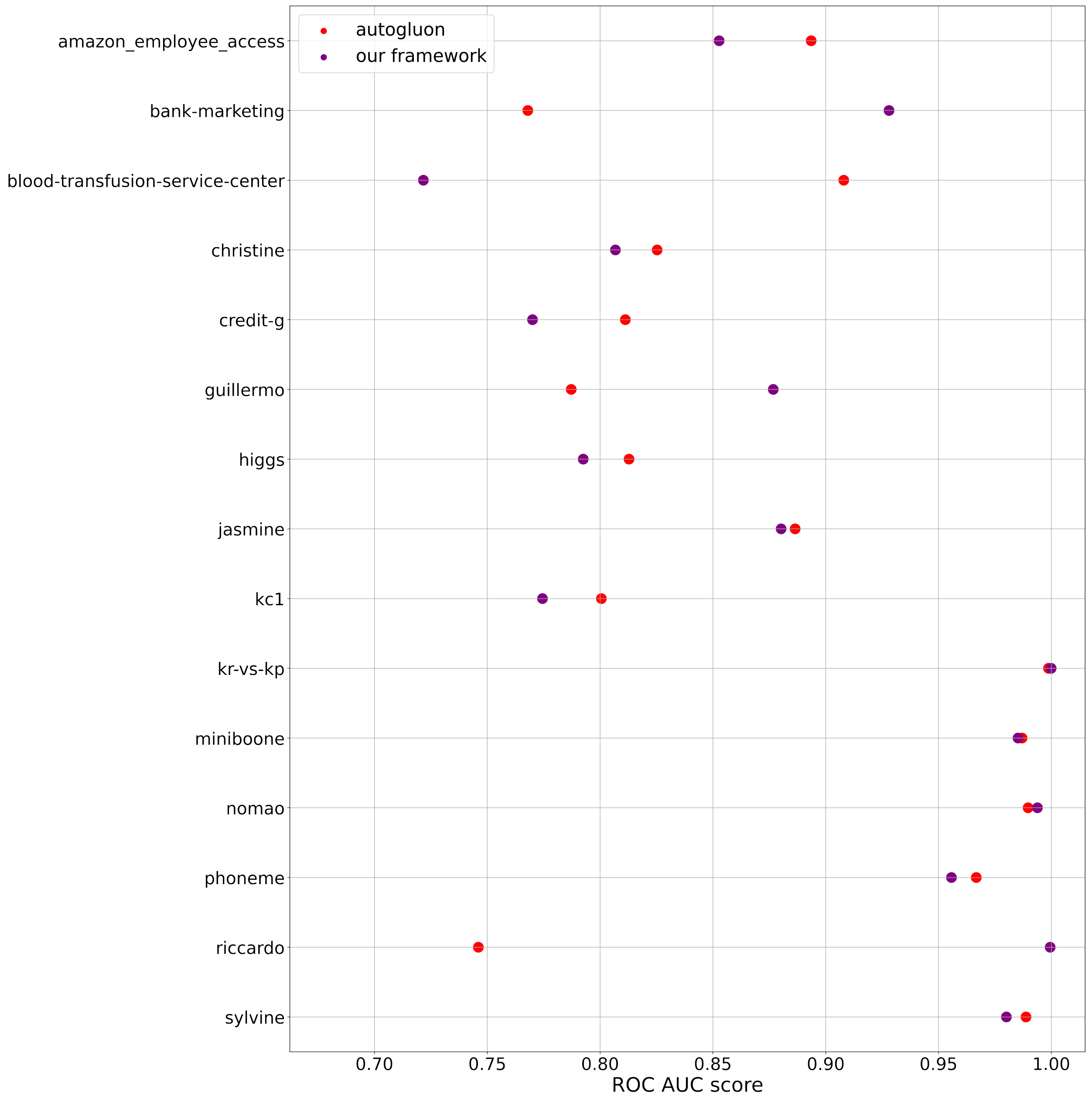

Figure 2: Average values of ROC_AUC metric for the binary prediction task achieved using 10-fold cross-validation by *AutoGluon* and values of ROC_AUC achievied by our framework. It is worth noting the Riccardo set, for which our framework got great results, while *AutoGluon* much worse.

References

Abhishek Thakur and Artus Krohn-Grimberghe. Autocompete: A framework for machine learning competition. *arXiv*, 2015.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv*, 2020.

Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, and Joaquin Vanschoren. An open source automl benchmark. *arXiv*, 2019.

Joaquin Vanschoren, Jan van Rijn, Bernd Bischl, and Luís Torgo. Openml: Networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15:49–60, 2013.