AutoGluon

Mikołaj Gałkowski **Kacper Kurowski** **Hubert Bujakowski**

June 2, 2022

Abstract

(ZOSTANIE UZUPEŁNIONY)
The article describes the popular AutoML framework AutoGluon, which has been tested via benchmark.

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 [1] 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 is going 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 using the AutoML framework AutoGluon [2]. We will test the capabilities offered by the framework and create a pipeline with which we can throw in raw data and get satisfactory results. As a guideline, we have used the benchmark [3], in which the authors have tested multiple AutoML frameworks on a subset of datasets available on the Open ML website. See [4] for the description of the website and its mission.

2 Framework description

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

- Simplicity – A user can train a model on the raw data directly without knowing the details about the data and ML models.
- Predictable Timing – It returns the results within the time-budget specified by users.
- Fault Tolerance – The training can be stopped and resumed at any time.
- Robustness – The 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[1] the necessary preprocessing. The following types of columns are allowed: `boolean`, `numerical`, `categorical`, `datetime`, `object`.

2.1 Tabular

2.1.1 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).

2.1.2 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.

3.1 Preprocessing

Our implementation includes creating the pipeline, which divides variables into categorical and numeric ones. Then according to the given type pipeline does:

- imputating missing data – KNNeighbours for categorical, Mean for numerical variable
- one hot encoding of categorical variables
- removing extreme numerical values – numerical variables
- scaling variables – numerical variables

3.2 Models

Our framework consists of 5 models:

- Logistic Regression
- Random Forest Classifier

[1]See 4.1

- XGBClassifier
- SVC
- Voting Classifier (based on previous models)

3.3 Metrics

Metrics saved by the framework:

- ROC AUC score
- f1 score
- recall
- precision
- accuracy

Framework during fitting trains and saves all of them into the leaderboard with all metrics. Best model is selected basen on ROC AUC score. Framework does not provide hyperparameter optimization and cross-validation.

4 Benchmark

4.1 Description

In order to do the benchmark, we have used 20 datasets present in [3] for the binary prediction task which was carried out via a pipeline. 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 [3].

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.

Based on it, we created our own benchmark based on 16 binary classification datasets.

4.2 Results

The effectiveness of the framework was tested on 10 folds using the AUC metric. The obtained values were then averaged. The final results are presented in Figures 1 and 2. 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.

5 Conclusion

In conclusion, AutoML is a very fast growing field that is both beginner and expert friendly due to its simplicity.

Simple implementation solutions can sometimes perform at a similar level to advanced frameworks.

However, keep in mind that the power lies in human hands, so educated data scientists will still not be replaced by AutoML.

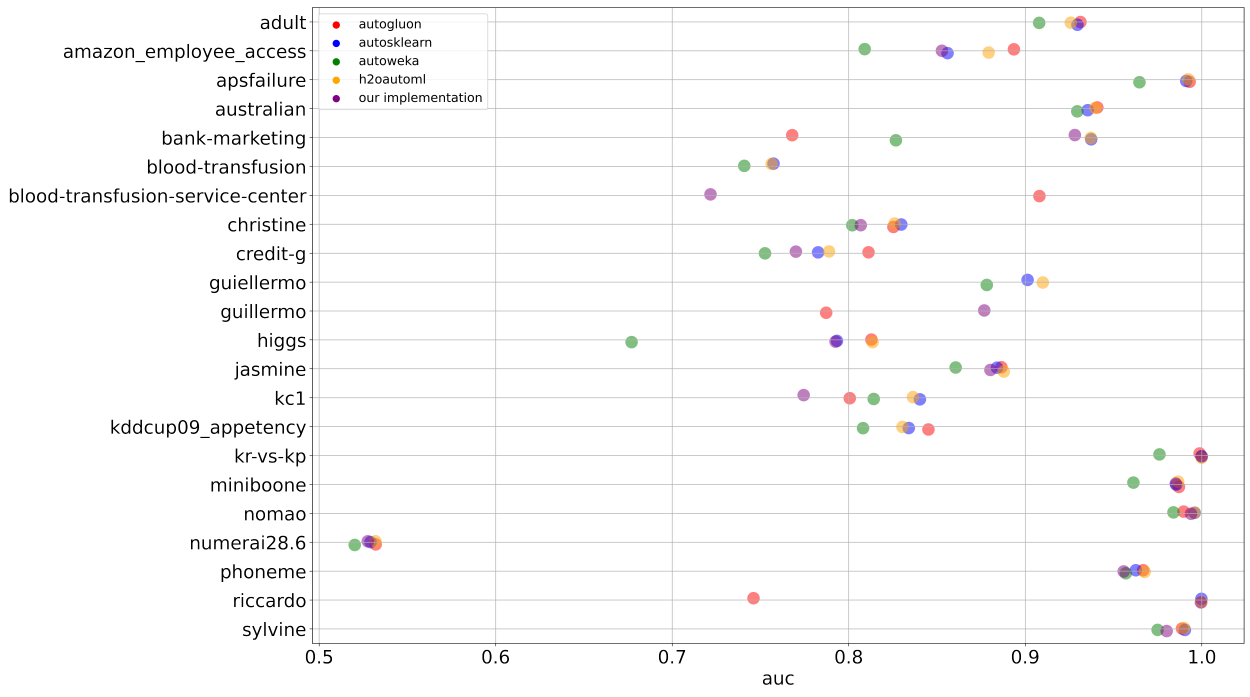

Figure 1: Average values of ROC_AUC metric for the binary prediction task including 3 OpenML frameworks, AutoGluon and our framework

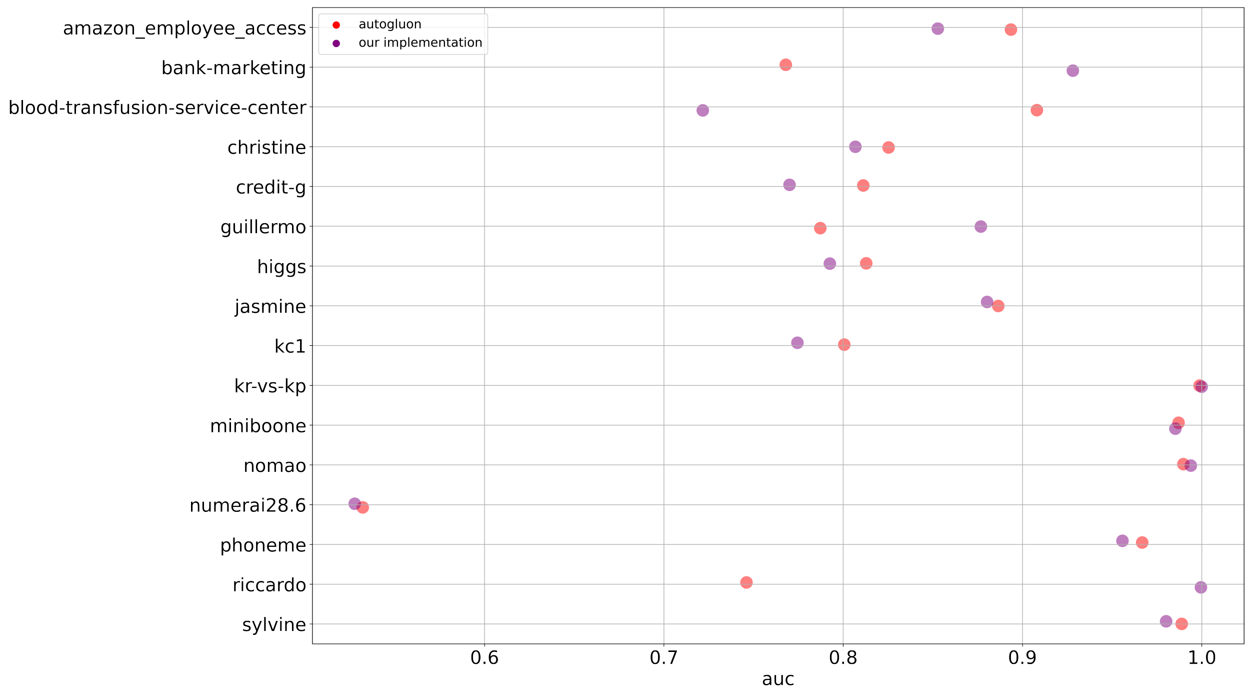

Figure 2: Average values of ROC_AUC metric for the binary prediction task achieved using 10-fold cross-validation by AutoGluon and values of ROC_AUC acheived by our framework. It is worth noting the Riccardo set, for which our framework got even great results, while AutoGluon obviously worse

References

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

2. Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola.

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

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