SZKIC ARTYKUŁU

A PREPRINT

Tomasz Siudalski **Grzegorz Zbrzeżny** **Piotr Marciniak**

1 Introduction

In the introduction we would like to present (below some main thoughts),

- the main differences between traditional approach and autoML approach,
- why AutoML can be beneficial in solving ML problems,
- the different approaches to AutoML,
- introduction to why we choose AutoSklearn2.0

Some ideas considering first bullet point

- autoML gives you opportunity to use automated algorithms to build predictive models,
- traditional approach requires some domain knowledge and time to build and compare over a dozen models.

Another ideas considering why AutoML can be beneficial

- it allows to automates time-consuming, iterative tasks,
- can product models with high scale, efficiency while maintaining quality,
- it doesn't require any significant domain knowledge.

The different approaches to AutoML (citations)

- Gama [Gijsbers and Vanschoren, 2020] - different approach into an AutoML (it uses genetic algorithms to optimize score) (lower exemplary text, which can be used)
  Gama uses different approach to optimize scores. There are three optimization algorithms, which search for optimal machine learning pipelines: random search, an asychronous successive halving algorithm and an asynchronous multi-objective evolutionary algorithm.
- Autosklearn Feurer et al. [2015] - using meta-learning for initializing the Bayesian optimizer and automated ensemble construction. (lower some example text, which can be used)
  It uses meta-features (i.e characteristics of the dataset), which can be computed efficiently, to choose which algorithm to use on a new dataset. These meta-features are complimentary to Bayesian optimization. They can quickly suggest some ML algorithms that are likely to perform well. But they don't work well with the high dimensional configuration spaces.
- Autosklearn2.0 Feurer et al. [2020] - the improvement of autosklearn (lower exemplary text, which can be used)
  Autosklearn2.0 presents two new parts, which improved performance of package. One of them was Portfolio Successive Halving. Another one of them was proposal of model-base policy selector to automatically chose the best optimization policy for an AutoML system for a given dataset.

2 Autosklearn 2.0 description

Auto-sklearn is an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator. It allows its users to automate:

- data preprocessing
- feature preprocessing
- hyperparameter optimization
- model selection
- model evaluation

Auto-sklearn is currently suitable only for classification. It chooses from sklearn-based models. The space of possible models currently spans 16 classifiers. Auto-sklearn performs following preprocessing techniques:

- Balances the dataset using class weights
- For categorical features it chooses between One Hot Encoding and no preprocessing. It also coerces rarely occurring categories, by default these smaller than 0.01 of the observations
- It imputes missing values using mean, median or mode
- Rescales the data, by default using standardization.
- Performs quantile Transformation by defult with 1000 quantiles and uniform output distribution
- Handles otliers removing the median and scaling the data according to the quantile range

After data pre-processing, features may be optionally pre-processed with one or more of the following categories of feature pre-processors:

- Matrix decomposition using PCA, truncated SCV, kernel PCA or ICA
- Univariate feature selection
- Classification-based features selection
- Feature clustering
- Kernel approximations
- Polynomial feature expansion
- Feature embeddings
- Sparse representation and transformation

For Auto-sklearn to find the best performing pipeline in given time from vast searching space the authors constructed a portfolio consisting of high-performing and complementary ML pipelines to perform well on as many datasets as possible offline. Then for a dataset at hand all pipelines in this portfolio are simply evaluated one after the other and if there is time left afterwards, the algorithm continues with pipelines suggested by Bayesian Optimization warmstarted with the evaluated portfolio pipelines. Auto-sklearn uses use an early-stopping strategy inside the whole search space which performs well on large datasets, but it's mostly useful for tree-based classifiers. To improve model selection the framework uses a multi-fidelity optimization method such as BOHB and several different strategies, including holdout and cross-validation. It also builds an automated policy selection on top of the previous improvements to select the best strategy.

3 Our preprocessing made for Auto-sklearn 2.0

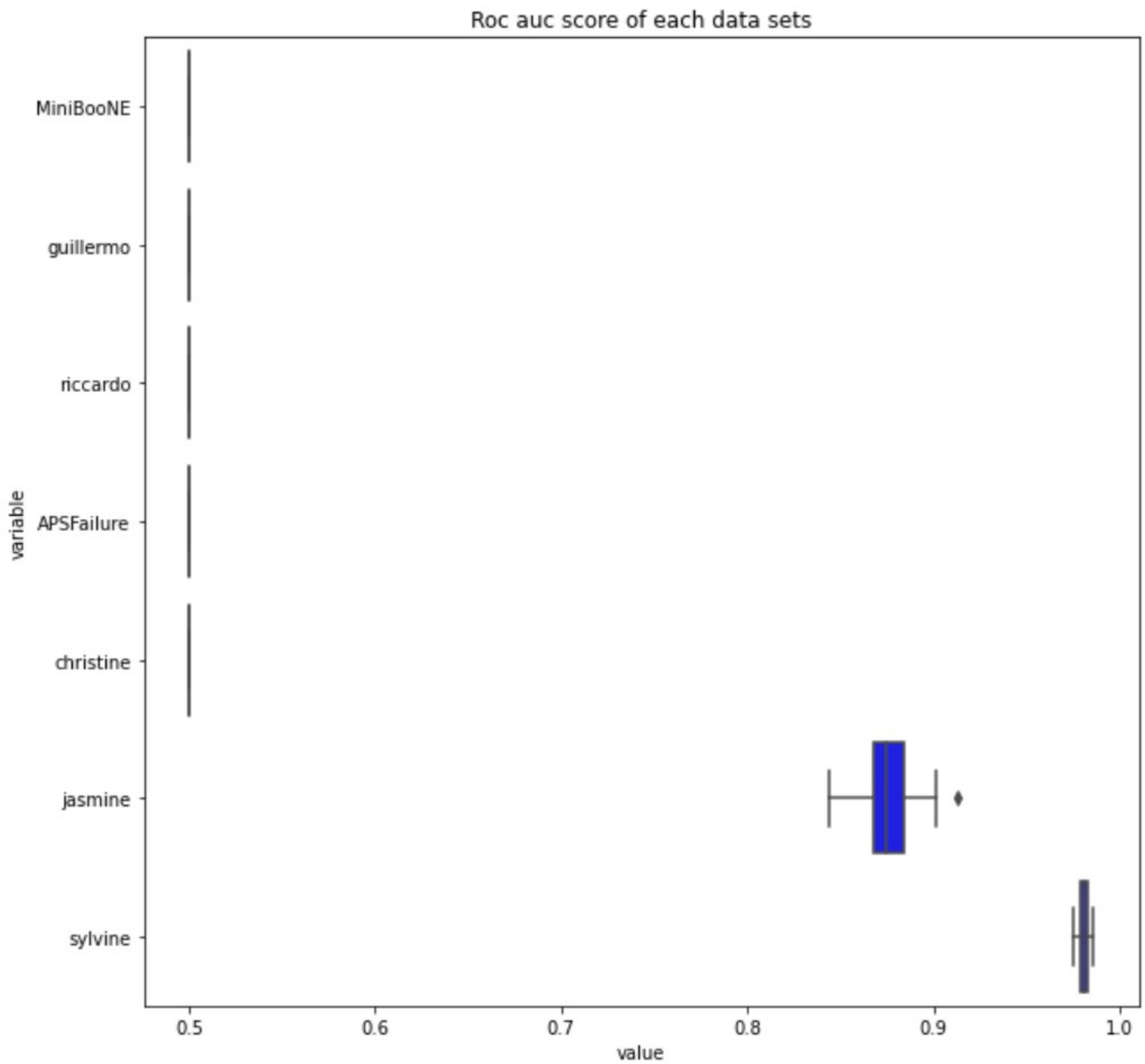
We made some functions responsible for the whole preprocessing and we store them in separate .py file. It is worth mentioning that we assume every column including data connected with dates in given set is in Datetime type. First step of preprocess is to make some changes in Datetime columns. We can not leave them without a change, because Auto sklearn 2.0 would treat them as a categorical column, thus it would be One Hot Encoded. This kind of operation creates many columns, which does not really contribute to any predictions. That is why we divide values in each row into three new columns, first of them is a day, second one is month and last of them is the year of the Datetime object. Then the original column is removed from data set. Second step is to change numerical columns which contains no more than 10 unique values to categorical type. Our framework does not know how to operate on columns with type object, so we decided to convert every object type column into categorical type. Sometimes we can come across a column in a data set which contains only NA or NaN values. It does not give us any information, that is why we remove them before giving the data to our framework.

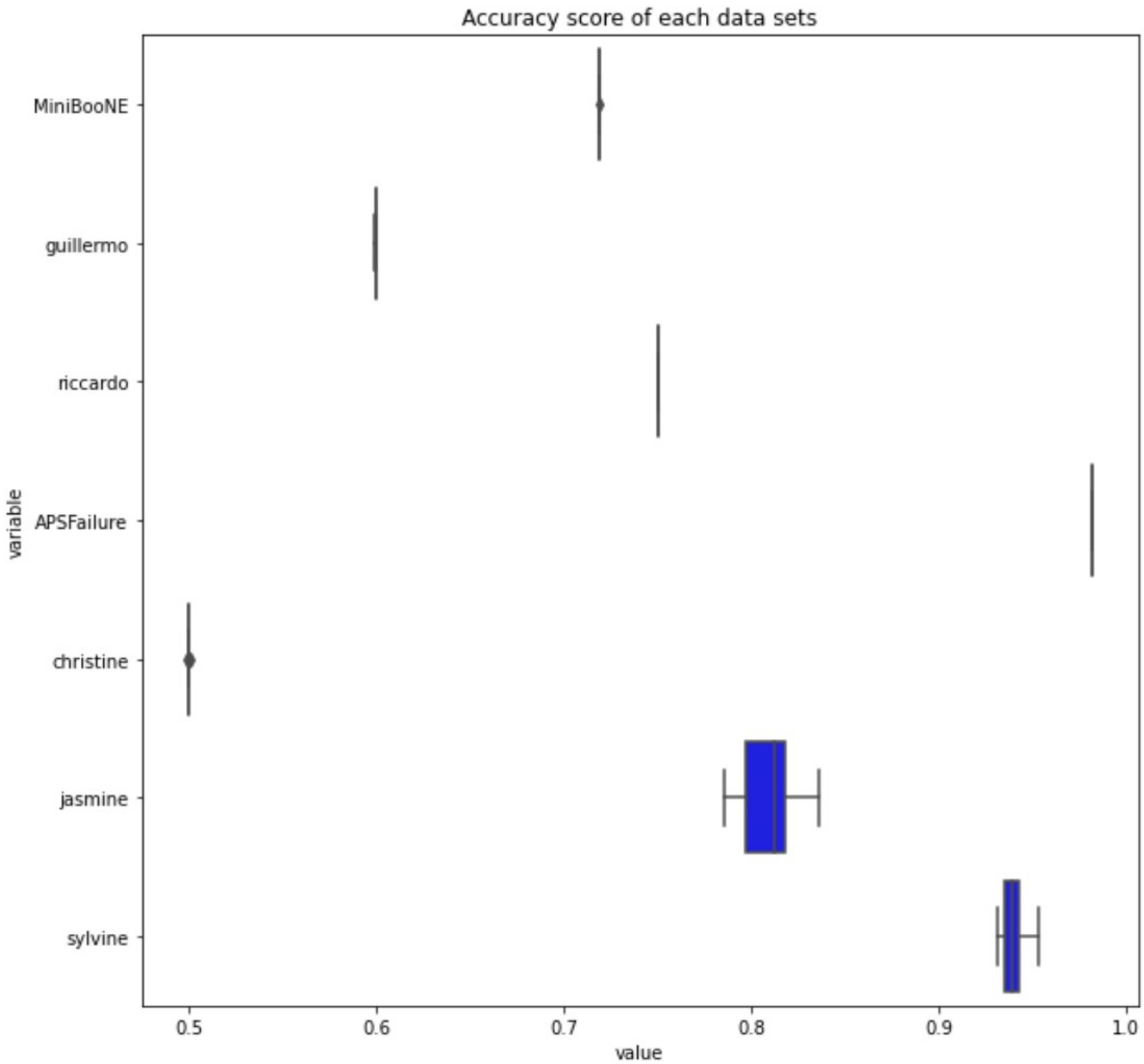
4 Evaluation scores

To test our framework we decided to use data sets from OpenML. We chose twenty-two sets related to binary classification issue. We used ten fold cross-validation on each set to test performance. Our first results are surely not the best one, because we gave our framework only sixty second on each fold due to an *No space left on device* error. To rate the scores received for each set we decided to use two metrics, accuracy score and roc auc score. We did not use f1 score, since we encountered a problem with finding out which value in target column is positive and which one is negative one. We would like to present our mean scores in a table:

	accuracy	auc
kr-vs-kp	0.969020 +- 0.010704	0.997327 +- 0.001657
credit-g	0.713000 +- 0.049810	0.784286 +- 0.039923
kc1	0.830729 +- 0.031422	0.814633 +- 0.032763
KDDCup09_appetency	0.982200 +- 0.000000	0.500000 +- 0.000000
adult	0.829164 +- 0.012293	0.922191 +- 0.004102
phoneme	0.874725 +- 0.015991	0.945183 +- 0.011482
nomao	0.952271 +- 0.002623	0.991082 +- 0.000941
blood-transfusion-service-center	0.759243 +- 0.037808	0.759514 +- 0.040616
bank-marketing	0.865452 +- 0.025790	0.925061 +- 0.007016
Amazon_employee_access	0.795048 +- 0.072844	0.742967 +- 0.014928
higgs	0.709118 +- 0.005331	0.787859 +- 0.006444
Australian	0.866667 +- 0.028102	0.938931 +- 0.022525
numerai28.6	0.519736 +- 0.004422	0.528440 +- 0.004246
MiniBooNE	0.719377 +- 0.000023	0.500000 +- 0.000000
guillermo	0.599850 +- 0.000229	0.500000 +- 0.000000
riccardo	0.750000 +- 0.000000	0.500000 +- 0.000000
APSFailure	0.981908 +- 0.000066	0.500000 +- 0.000000
christine	0.500000 +- 0.000413	0.500000 +- 0.000000
jasmine	0.809307 +- 0.016282	0.876776 +- 0.019175
sylvine	0.939891 +- 0.006401	0.980818 +- 0.003078
airlines	0.554558 +- 0.000006	0.500000 +- 0.000000
albert	0.500000 +- 0.000000	0.500000 +- 0.000000

After we will overcome an error with the lack of memory we will run the framework for every data set once again, this time with few minutes on each fold. But for now we can present a boxplot of scores from some sets from that shortened run.


Roc auc score of each data sets
MiniBooNE
guillermo
riccardo
APSFailure
christine
jasmine
sylvine
variable
0.5
0.6
0.7
0.8
0.9
1.0
value



5 Own implementation

Based on experiences we collected, we have committed ourselves to the task of creating our own AutoML framework. Our goal was to create simple and universal tool that would achieve relatively good results compared to much more advanced frameworks we learned during our classes. In sections below we will describe our solution in detail.

5.1 Preprocessing

To perform preprocessing needed in our implementation of Auto ML we created another .py file, containing few classes and functions responsible for whole task. We made differents preprocessing for different column types. Missing values in numerical columns were imputed with mean and then whole columns were scaled with StandardScaler and then MinMaxScaler. DateTime columns were encoded by our custom class DateTransformer which splits the date into three values: year, month and day. Majority of categorical values had their missing values filled with most frequent value and there whole columns were OneHotEncoded. We treated numerical columns with no more than ten unique values as categorical ones. Moreover some categorical columns were transformed in other way. If categorical column had more than twenty unique values we used our woe pipe on it, which firstly impute all missing values in the same manner as before and then transform whole column using WoEEncoder. Pipes for each categories were stored together in ColumnTransformer.

5.2 Model selection

To find solution that would work quite well in many cases and wouldn't take a lot of time we decided to create seven stacking ensembles of sklearn models. The ensembles differed in size and models used. The models used by as are: LogisticRegression, RandomForestClassifier, AdaBoostClassifier, SVC, GaussianNB and KNeighborsClassifier. We selected these models specifically because they are known to perform well and differ in structure. For each model we created spaces of hyper-parameters. Our framework tests all ensembles on data and performs Bayesian Optimization on them tuning these hyper-parameters. To sum up, you can see a diagram of our solution below.

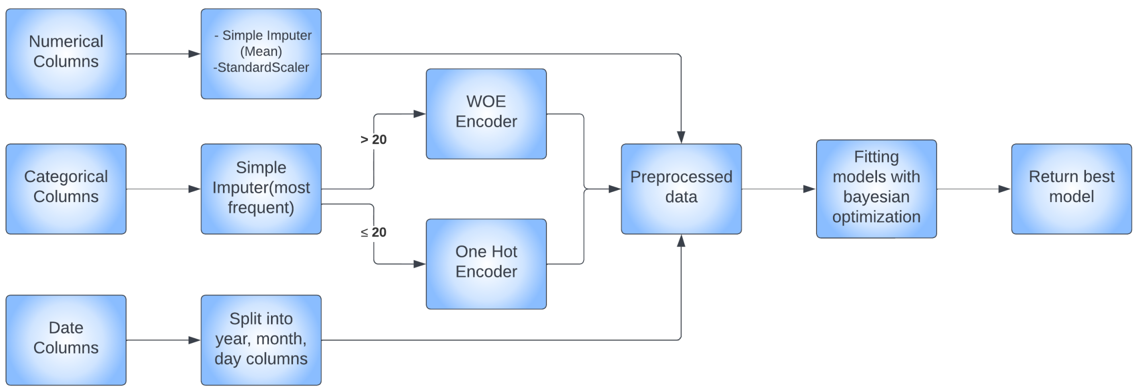


Figure 1: Diagram of our Auto ML solution

References

Pieter Gijsbers and Joaquin Vanschoren. GAMA: a general automated machine learning assistant. *CoRR*, abs/2007.04911, 2020. URL `https://arxiv.org/abs/2007.04911`.

Matthias Feurer, Aaron Klein, Jost Eggensperger, Katharina Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28 (2015)*, pages 2962–2970, 2015.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: Hands-free automl via meta-learning. 2020.