Comparison of AutoML frameworks with classic machine learning: case study

AutoML 2023Z

Dawid Płudowski, Antoni Zajko

January 15, 2024

1 Methodology

In this section, we introduce methodology of experiments, explain and introduce notation used in this paper and list models and datasets used in experiments.

We compared AutoML frameworks with manual machine learning. The experiments were performed on one train dataset provided in repository: https://github.com/kozaka93/2023Z-AutoML. The dataset were split into train and validation set with ratio 2:1 for both manual ML and AutoML. For manual ML, we used: *XGBoost*, *lightgbm* and *scikit-learn* packages. We used three frameworks to perform AutoML task: *AutoGluon*, *AutoSklearn* and *MLJar*. For *AutoGluon* and *AutoSklearn*, the target metric was balanced accuracy. As a distribution of the classes in train dataset was uniform, we used accuracy instead of balanced accuracy in *MLJar*.

2 Experiments

In this section, we provide description of the experiments and theirs results.

2.1 Manual machine learning

As a tool for the manual creation of a machine learning model, we decided to use *scikit-learn*, *xgboost*, and *lightgbm* libraries. The first step of this process was a feature selection. As a selection algorithm, we decided to use Recursive Feature Elimination with Random Forest as a feature importance estimator. Specification of hyperparameters of this step is provided in Table 1. Afterward, we decided to try various models - XGBoost, SVMs, Logistic regression, and LightGBM. The best of them was LightGBM with 0.81 balanced accuracy. Next, we decided to concatenate them using stacking with 10-fold cross-validation and logistic regression as a final estimator which significantly improved performance on the test set and increased balanced accuracy to 0.82. We considered this result as decent and let it become our final solution. The performance of all our approaches is in Table 2. Specification of all hyperparameters is in Table 3. We searched for all hyperparameters manually. The final architecture of manual model is in Figure 1

Entry	Hyperparameter	Value
RFE	n_features_to_select	100
	step	10
Estimator - Random Forest	n_estimators	50
	max_depth	4

Table 1: Hyperparameters of feature selection step.

Model	Balanced Accuracy
XGBoost	0.79
SVM	0.75
LightGBM	0.81
Logistic Regression	0.56
Stacking	**0.82**

Table 2: Performance of each tried model.

Entry	Hyperparameter	Value
Stacking	CV folds	10
XGBoost	n_estimators	50
	max_depth	5
SVM	C	100
LightGBM	n_estimators	100
Logistic Regression	penalty	l2
	C	1

Table 3: Hyperparameters used for creating a model.

2.2 Machine learning with AutoML frameworks

We tested all aforementioned frameworks on the same split of the data. The best score and the training time for each framework is presented in the Table 4. For *AutoGluon*, the time of the training was automatically chosen by the framework, while for the rest of the frameworks it was set manually.

Framework	Time	Best accuracy
AutoGluon	1min	0.83
AutoSklearn	1h	0.85
MLJar	**6h**	**0.86**

Table 4: The results of the training of the AutoML frameworks. Best accuracy is calculated based on the validation set that was not seen by the model during the training.

MLJar's best model is the simple random forest trained on selected features. Its hyperparameters are show in Table 5. It was evaluated on 10-fold validation with a stratify. The training process was performed with *Compete* mode.

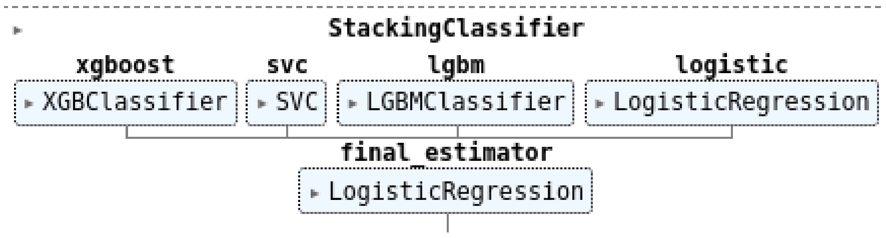

Figure 1: Architecture of the model created manually .

Hyperparameter	Value
criterion	entropy
max_features	1
min_samples_split	40
max_depth	7

Table 5: The hyperparameters of the random forest created by *MLJar*.