Example Submission Final Project AutoML Lecture WS 2022/2023

Constantin von Crailsheim[1]

[1]LMU Munich, Institute of Statistics

Abstract

1 Introduction

The AutoML system optimizes a ML pipeline consisting of four elements. First, a choice of two imputers replaces missing values. Then, either over- or undersampling was applied to deal with imbalance in the targets. After normalizing all features, three different models were fitted, i.e. a random forest classifier, a gradient boosting classifier and a SVM classifier. The choice of pre-processing and the hyperparameters of the model were optimized by DEHB. Finally, the best pipelines for each model were stacked and the final AutoML system predicts the target using majority voting.

2 Method

Choices of imputation strategy of sklearn.impute:

- The SimpleImputer is an univariate imputer, which completes missing values with a descriptive statistic per feature. I chose the median, since it is less sensitive to outliers than the mean and will most likely still produce sensible imputed values for categorical features.
- The KNNImputer replaces missing values by the mean value for that feature of its 5 nearest neighbors based on Euclidian distance of their non-missing feature observations.

Choices of data-level sampling method to yield balanced dataset of imblearn:

- SMOTE as an oversampling approach generates new samples of the minority class by interpolating between existing observations of the minority class, while not distinguishing between easy and hard samples.
- TomekLinks as an undersampling approach removes samples from the majority class, if they are nearest neighbors to a minority class sample, thus removing noisy borderline examples of the majority class.
- SMOTETomek which combines SMOTE and Tomek links.
- No sampling method, which would allow algorithmic-level methods to deal with the imbalanced data.

Since the above pre-processing methods impute missing values by means or medians and generate new samples by interpolation, values for categorical features could be generated which are not actually categorical anymore. Thus, another layer is added to the pipeline, which rounds all observations of categorical features to an integer.

The last step of pre-processing is the choice of whether to apply the StandardScaler of sklearn.preprocessing to standardize the features. This will be in particular useful for the SVM, since a RBF kernel assumes features centered around zero and similar variance across features.

The first model in the stack is a RandomForestClassifier from sklearn.ensemble. Why? The hyperparameters to be optimized, which will all be sampled uniform, are as following:

Hyperparameter	Data type	Search space	Default	Other
n_estimators	Integer	[50,500]	100	Quantization factor = 50
criterion	Categorical	{Gini, Entropy, Log loss}	Gini	
max_depth	Integer	[5,15]	10	
min_samples_split	Integer	[1, 64]	2	Log scale
min_samples_leaf	Integer	[1, 16]	1	Log scale
max_features	Integer	[0.1, 0.9]	0.5	
class_weight	Categorical	{Balanced, Balanced subsample, None}	None	

Comment on hyperparameter

The second model in the stack is a GradientBoostingClassifier from sklearn.ensemble. Why? The hyperparameters to be optimized are as following:

•

The third model in the stack is a Support Vector Classifier (SVC) from sklearn.svm. Why? The hyperparameters to be optimized are as following:

•

Evaluation strategy:

- 3 fold external cross-validation, i.e. 1800 seconds to fit AutoML system per fold.
- 3 model, so 600 seconds to find best hyperparameter per model with DEHB.
- 4 fold internal cross-validation, i.e. takes 4 times as long per fitting of model.
- Not too many fits, thus no hyperparameter tuning for pre-precessing.

Optimization strategy: DEHB by Awad et al. (2021):

- High speed-up gains.

Refit model for incumbent and and remove unbalanced sampling from pipeline.

Final AutoML system is stacked version of three models with majority voting for final classification.

3 Experiments

External cross-validation performance vs. baseline:
Chosen incumbents with table:

Example of trajectory for CV fold.

4 Conclusion

Improvements: Also optimize hyperparameter of pre-processing methods.

References

Awad, N., Mallik, N., and Hutter, F. (2021). DEHB: Evolutionary hyberband for scalable, robust and efficient hyperparameter optimization. In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2147–2153. ijcai.org.

A Trajectories

Bla Bla

B Incumbent hyperparameter