Meta-modeling: A surrogate model selection approach in automated machine learning.

Gascón-Pidevall. Dídac[1], Hidalgo-Gadea. Lorenzo[2]

Abstract—Automated machine learning solutions tend to be implemented as a black-box model, where given a dataset, an accuracy is returned giving no information of the dataset or how to deal with it. This report presents the proof of concept of a white-box approach, giving answers to questions such as: Is the dataset balanced? Which are the important and the redundant features? Which machine learning algorithm should I use? Each component of the code has been implemented to respond these questions. It features a meta-model: Given dataset descriptors, it is capable of selecting the machine learning model that will perform best based on

Index Terms— Machine learning, Automated Machine learning, Python.

1 Introduction

This report, alongside with the presented code, form the final deliver for the first Datathon conducted by Management Soltions.

The Datathon consists in designing and developing an Automated Machine learning model from scratch. The model ought to be capable of making good predictions in various datasets without human assistance. The datasets are constrained to numerical and categorical data and the target, categorical. The performance of the model will be assessed by the AUC metric, provided by the model itself, having the metric as the output. The code execution must not exceed 20 minutes.

Implementing an automated machine learning is a rather complex task; there are many issues that could potentially lead to a biased model. How do you read the data to identify the type of each feature? What happens when the input data is unbalanced or have high variance? How do you deal with High dimensions and unimportant features? What ML model do you choose in each dataset? Why?

The approach here presented is a proof of concept for automated machine learning with the constraints of the Datathon contest. The design of the algorithm is based on the following principles:

1. Data quality and quality assurance are fundamental for a proper model. Therefore, significant computational and time resources will be invested to this end.
2. A single Machine learning method with acurate hyperparameter tunning is preferred over trying a dozens algorithms without proper treatment.
3. A decision tree (the meta-model) performs the machine learning model selection, making it a 'white-box' approach, as opposite to 'blacbox' approaches (where a neural network performs calculations and provides no information about how to deal with the dataset). The features that the decision tree uses are descriptors of the dataset such as number of features, number of observations etc.

The following section details each part of the approach, detailing the decisions that led to the present design.

2 Methodology

2.1 General scheme

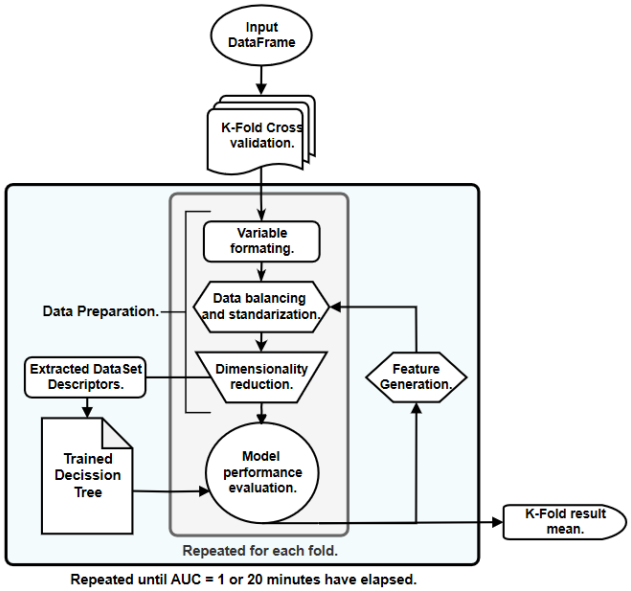

Fig. 1. Automated machine learning training flow chart.

The dataset is formatted, balanced, standardized and dimensionality reduced for each fold. Then the decision tree evaluates which ML model should be applied based on the dataset descriptors. Once the ML model is trained

1. *Energy consultant at Management solutions, Barcelona.*
2. *New technologies consultant at Management solutions, Barcelona.*

and evaluated, more features are generated recursively. The model stops when maximum AUC is achieved or when 20 minutes timeout limit is exceeded, returning the best AUC of the previous iterations.

2.2 Variable formating

Each feature of the dataset is treated independently. Single value features, identificators and dates are droped from the model and the rest are processed as follows. Numeric are formatted and categorical features are encoded with baseN encoding, resulting in a numerical dataset with integers, decimals and Booleans. If target is not a number, is processed with the ordinal encoder.

2.2 Data balancing and standarization

Data with different amount of observations in each classes or not standarized, might bias the model and decrease its predicting power. We have used two algorithms to mitigate this risk.

In order to assess whether a dataset is balanced or not, the entropy is computed. If the entropy appears to be high means that the target is a low probability event and the data is well balanced. On the contrary, if the entropy is low, means that there is a major class that makes the event highly probable, therefore, the probability distribution of the target is skewed and the set unbalanced.

A resampling technique is applied with unbalanced datasets. They consist in adding and dropping observations in de data set in order to achieve a balanced data set among classes. The resampling model applied is Smote Tomek algorithm. SMOTE (Synthetic Minority Oversampling Technique) oversamples the minority class based on the values of the existing ones. Concurrently, Tomek links undersamples the majority class removing instances close to the ones of the opposite side.

Standardization of a dataset is a common requirement for many machine learning estimators. Typically, this is done by removing the mean and scaling to unit variance. However, outliers can often influence the sample mean / variance in a negative way. In such cases, the median and the interquartile range often give better results. That is the reason for the robust scaler to be used in this approach.

2.3 Dimensionality reduction

Once the features are formatted and target-wise consistent, dimensionality reduction ought to be implemented to determine the features that are important for the classification, and remove the ones that are highly correlated and redundant. It is also useful as it reduces computation time in the training of the model. Tipically, Principal component analysis (PCA) has been used to this end. There is a big drawback with PCA, it does not take in account the target, and might delete features that are useful when classifying.

For this reason, an alternative has been used: Recursive feature elimination with cross-validated selection (RFECV). It repeatedly constructs a model and chooses either the best or worst performing feature, setting the feature aside and then repeating the process with the rest of the features. The optimal number of features is selected via cross validation loop.

2.4 Machine learning models and hyperparameter optimitzation

Machine learning models used training the meta-model are:

1 K-nearest neighbors.
2 Logistic regression classifier.
3 Bernoulli naive bayes classifier.
4 Gaussian naive bayes classifier.
5 Passive aggressive classifier.
6 Ridge regression classifier.
7 Logistic regression classifier with stochastic gradient descent learning.
8 Support vector classifier with linear, radial function basis, sigmoid and polynomial kernel.
9 Decision tree classifier.
10 Random forest classifier.
11 Bagging decision tree classifier.
12 Histogram-based gradient boosting classification Tree.

For each model, a search space of hyperparameters has been set. In the training step, Random seach cross validation evaluates combinations of hyperparameters values to maximize the AUC score of the model over the training data. The best performing combination is set and trained with training data.

2.5 Model selector

The criteria for selecting the most suitable ML model for a given dataset, most of the times have the form of a decision tree: (e.i. if the dataset have more than x features use model u, if the model have more than y observations use the model v…). Using this observation as inspiration, an actual decision tree has been developed.

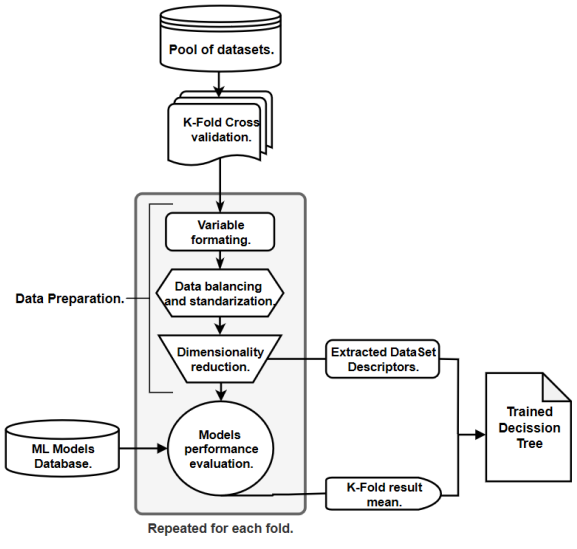

Fig. 2. Flow chart of the training of the meta-model. The pool of

datasets are formatted and all models are trained and evaluated for each dataset. The algorithm with the best performance alongside with the descriptors of the dataset are provided to the decision tree.

A pool of datasets have been gathered and formatted with the processes previously explained (Data preparation). All ML models have been trained and tested with all datasets. For each dataset, the following descriptors are extracted:

1) Nº of observations.
2) Nº of numeric features.
3) Nº of classes in target.
4) Nº of Boolean features.
5) Nº of features.

The meta-model is trained with a dataset consisting of the descriptors as features and the best performing model as target.

In order to determine the most suitable model for a given dataset, it is necessary to extract the descriptors cited above, and the trained decision tree returns the name of the model.

2.6 K-fold Cross validation

Spliting up test and train data is a tricky task, as setting static train and test observations makes the model specially exposed to noise and overfitting. There is a further complication in our model, as the algorithms presented so far (SmoteTomek, REFCV…) may generate diferent datasets among identical executions.

K-fold Cross-validation became the best option, as trains the model using the subset of the dataset and then evaluate using the complementary subset of the data-set. This task is repeated k times, and the output becomes the arithmetic mean of the k outputs. Although it increases significantly the computation costs, it makes the algorithm robust to noise, overfitting and diminished the effect of the inner algorithms.

2.7 Feature generation

Once the first AUC score has been estimated, the feature space is enlarged based on the existing features. Feature generation is executed in order to reveal relations and ratios of the fetaures that could potentially increase the predicting power of the model. The generated features are simple mathematical convinations of the existing features. The combiantions are the following:

1) Additive combination
2) Multiplicative combination
3) Division combination
4) Additive and Multiplicative combination
5) Additive and division combination
6) Multiplicative and division combination
7) Additive, multiplicative and divison combination

Once the new columns are created, the data is again standardized, dimensionality reduced, and the model is trained again giving a new AUC result. This happens recursively for all 7 modalities of feature generation.

Features generation can be time and computationally really costly, especially when the original dataset have a large number of features. This is meant to be this way, as the amount of combinations increase exponentially with the number of original features. Although the big computational cost, it has a great potential as could unveal relevant features that would boost the AUC score. The model has a timeout of 20 minutes, so if the feature generation process exceed the time, the timeout stops the process and shows the biggest AUC estimated.

3 RESULTS

	Filter	Number of sets
1	Gathered Sets	192
2	Fulfil missing and size criteria	171
3	**Used to train meta-model**	**93**

Table 1. Number of tables used for training the meta-model.

In order to have an acurate meta-model, it was needed to gather as much datasets as possible. Some of them had to be removed for the amount of missing values and for exceeding size limit, as the execution time would take to much and would not be representative of the potential datasets with which the model will be tested.

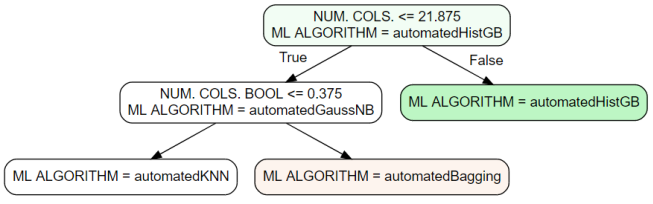

Fig. 3. Graphical representation of the meta-model decision tree classifier.

Figure three presents a graphical representation of the decision tree trained used as model selector. The tree had a severe affectation of class imabalance that had to be fixed manually for the poor number of observations in the minority classes.

Data Set	Algorithm	AUC
yeast1.csv	K-nearest Neighbours	0.68
ring.csv	K-neares t Neighbours	0.5
elephant.csv	Histogram-based gradient boosting	0.85

Table 2. Initial datasets AUC scores of the final model.

The model AUC scores are presented in the table 2 using the initial datasets given by Management solutions.

7 APPENDIX

7.1 Execution of the code

To execute the code it is necessary to install python 3.6 and all library dependencies found in the requiremnt file. This might be done by executing the attached in-

stall_requirements.bat file or by running the following command in the windows console:

pip install –r requirements.txt

After all dependencies have been installed, the config_file.py needs to be updated with the path were the model file is save.

A python console can be started within the same folder the files are located and the code might be imported as another library. For example:

from automl import automl

The main function to be executed is called automl and accepts a pandas *DataFrame* and an optional Boolean as input. If the Boolean variable, named *explicit* is set to *True*, the function will return the AUC score and the name of the algorithm that archived that result. By default, if only a *DataFrame* is passed as input, the function will only return the highest AUC score archived.

Examples:

1. To get the AUC score:
   AUC = automl(df)
2. To get the AUC score and the algorithm used:
   name, AUC = automl(df, explicit=True)

The code is set to execute for a maximum of 20 minutes and will, in most cases, use the whole 20 minutes trying to archive the best AUC possible. This threshold and other parameters can be changed in the confi_file.py.

8 Conclusion and future work

This paper presents a proof of concept for the automated machine learning challenge, based in a white-box structure. The main obstacle that was faced was the computation time that took preparing the data for the meta-model. Once enough information was gathered, the classes appeared to be highly imbalanced.

Future improvements of this work would mainly invonve adding datasets to traing the meta-model and achieving a more complex decision tree that would lead to the usage of more variated machine learning models. Another field of improvement would be focusing the feature generation only to the subset of important features localized with RFECV.

Disclaimer: The poor AUC scores showed in the results section does not represent the actual predicting capacity of this model, which is significantly higher. This model achieves a 0.98 AUC score in the famous iris dataset, and will hopefully perform better in the three datasets on which will be tested for the datathon.

Acknowledgments

We would like to thank the professionals of the Barcelona office firm, sepecially Xavier Martínez, Victor Alicart and Yolanda Pujol for giving us access to facilities and resources without which this project would not have been possible. We really appreciate it.

Link for the python 3.6.0 code

https://github.com/Lorenzohidalgo/AutoML