FOM University for Economics and Management Essen, Germany
Location Münster

Part-time Bachelor of Science Program
Business Information Systems

5th Semester
Big Data & Data Science

Results of Machine Learned Pandas
–
a Prediction Challenge

Mentor:
Sasha Schworm

Authors:
Patrick Schulte-Austum
Matrikel-Nr.: 470057
patrick.schulte-austum@fom-net.de

Münster, February 28, 2020

I. Table of Contents

II. List of Abbreviations

Abbreviation	*Explanation*
GBM	Gradient Boosted Modifier
CV	Cross Validation
SMOTE	Synthetic Minority Oversampling Technique
DFS	Deep Feature Synthesis

III. Table of Figures

IV. Table List

1 Introduction

The Task of this Challenge is to predict weather or not a customer of a bank will churn. Therefore, a dataset (A) is given where the success is featured. With this a machine learning algorithm can be trained to predict on another dataset (B), which doesn't feature the success, if the customer will churn.

To evaluate the model the F1 Score is used. The objective is to achieve a better prediction than 50%.

To fulfill this task, three steps have to be accomplished:

1. Cleaning up the given dataset and detect "leaking" data
2. Feature Engineering
3. Using the correct algorithm for the binary classification problem

In the GitHub-Repository[1] the source code, the submission and the original version of Figure 1 can be found.

2 Software details

For completing the given assignment, we setup our own JupyterLab Environment, where the home server is located and setup by Michael Heichler.

The following packages are not installed over pip but over the package manager Pacman. The chosen Linux distribution is ArchLinux.

Package	***Version***
Pandas	1.0.1
Seaborn	0.10.0
Numpy	1.15.2
Scipy	1.1.0
sklearn	0.19.1-3
Matplotlib	3.0.1
Featuretools	0.13.2
LightGBM	2.3.1
JupyterLab	1.2.6
Jupyter-Core	4.6.1
Jupyter-Client	5.3.4
iPython	7.12.0

Table 1 - Environment - Package Versions

[1] (Heichler & Schulte-Austum, 2020)

3 Structure of the Code

3.0 In a Nutshell

To give a better overview about the code a visualization of the source code was created. A bigger version can be found in the GitHub-Repository[2].

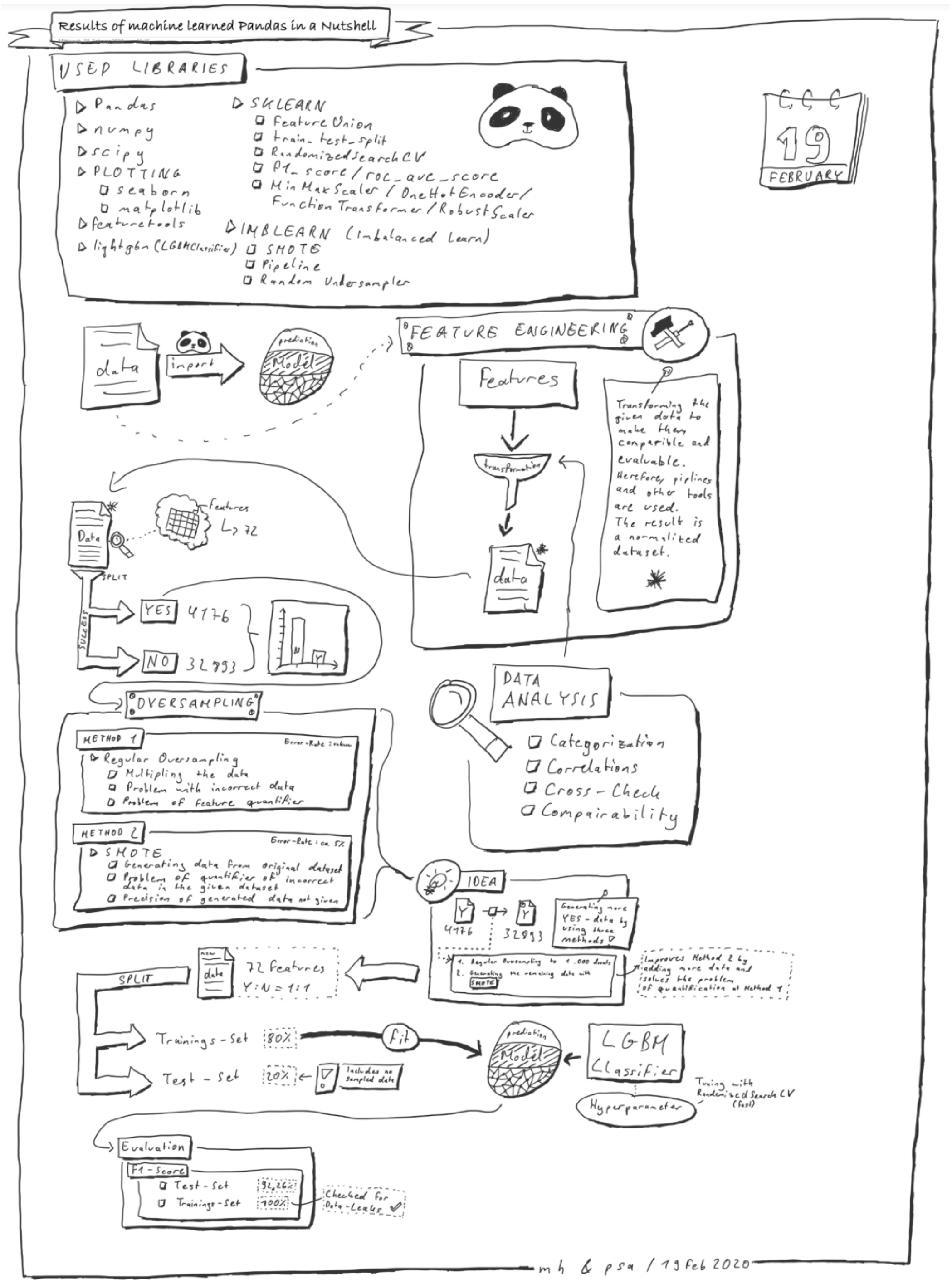
Results of machine learned Pandas in a Nutshell
USED LIBRARIES
Pandas
numpy
scipy
PLOTTING
seaborn
matplotlib
featuretools
lightgbm (LGBMClassifier)
SKLEARN
Feature Union
train_test_split
RandomizedSearchCV
f1_score / roc_auc_score
MinMaxScaler / OneHotEncoder / FunctionTransformer / RobustScaler
IMBLEARN (Imbalanced Learn)
SMOTE
Pipeline
Random Undersampler
19
FEBRUARY
data
import
prediction
Model
FEATURE ENGINEERING
Features
transformation
data
Transforming the given data to make them comparible and evaluable. Herefore, piplines and other tools are used. The result is a normalited dataset.
Data
Features
↳ 72
SPLIT
SUCCESS
YES 4176
NO 32893
N
Y
DATA ANALYSIS
Categorization
Correlations
Cross-Check
Compairability
OVERSAMPLING
METHOD 1
Error-Rate: unknown
Regular Oversampling
Multipling the data
Problem with incorrect data
Problem of feature quantifier
METHOD 2
Error-Rate: ca. 5%
SMOTE
Generating data from original dataset
Problem of quantifier of incorrect data in the given dataset
Precision of generated data not given
IDEA
Y 4176
Y 32893
Generating more YES-data by using three methods
1. Regular Oversampling to 1.000 datas
2. Generating the remaining data with SMOTE
Improves Method 2 by adding more data and solves the problem of quantification at Method 1
new data
72 features
Y:N = 1:1
SPLIT
Trainings-Set 80%
fit
Test-Set 20%
Includes no sampled data
prediction
Model
LGBM Classifier
Hyperparameter
Tuning with RandomizedSearchCV (fast)
Evaluation
F1-Score
Test-Set 92,26%
Trainings-Set 100%
Checked for Data-Leaks ✓
mh & psa / 19 Feb 2020

Figure 1 - In a Nutshell

The following paragraphs are describing the code in addition to Figure 1. A bigger version of the flowchart can be found in the GitHub-Repository.

[2] (Heichler & Schulte-Austum, 2020)

A big help with the analysis was the GitHub repository by Joe Hoeller[3]. The data structure used there differs from data set A only in the features and their count.

3.1 Library Import

At the beginning of the code the import of the needed libraries takes place. Most of the needed packages are mentioned before, the remaining used packages can be found at the top of Figure 1.

3.2 Feature Engineering

3.2.1 Data Analysis

To create a better overview about the given dataset a data analysis is needed. At first, we looked at the bias of the data: with 32,893 sets with "No" result for success and 4,176 sets with "Yes" the given dataset is highly imbalanced.

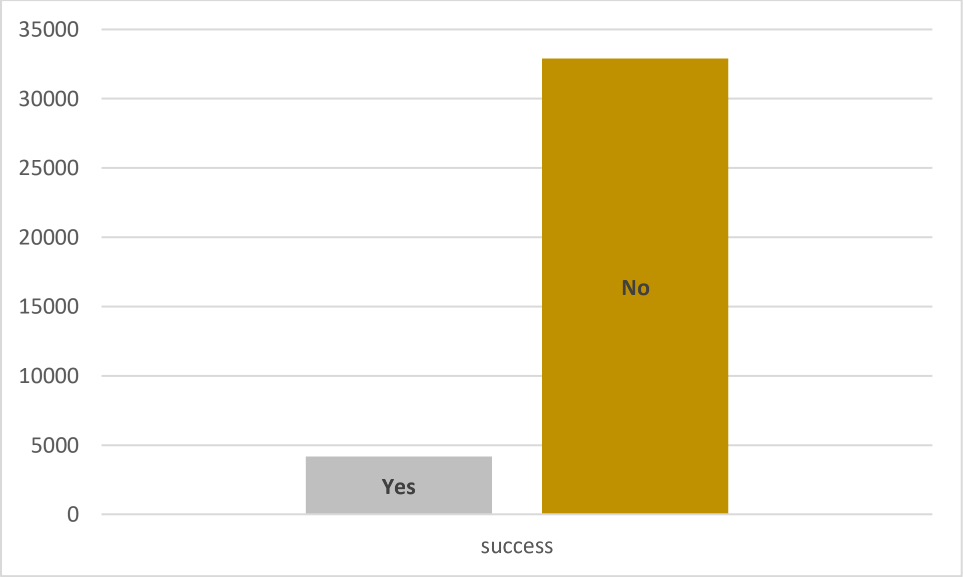

Figure 2 - Data Distribution

Due to this imbalance in the distribution, it is extremely challenging for the model to provide the right predictions.

Subsequently, a more in-depth analysis was conducted regarding the categorical and numerical features.

Categorical Analysis

Before the model can be trained, it must be checked for missing data. Then a visualization of the given values across the categorical features can provide a clear insight which features are important, and which are not relevant for the classification problem.

[3] cf. (Hoeller, 2019)

First of all, it appears that there are many unknown values in general, which can be a huge problem for the training of the model. Only regarding the marital status, the bank has a fully comprehensive database. The low figures for illiterate and defaulted loan are striking. From this it can be deduced that the bank does not want to take on customers with a bad credit.

An in-depth analysis of the categorical features shows that the distribution of customers with and without housing loan is 1:1, which means that this feature will most likely not provide any insights during the training. There are very few entries on personal loan. The high failure rate for previous conversations is striking, which is also reflected in the current campaign. In total, twice as many cellular phone calls were made as landline phone calls.

Finally, the categorical analysis allows us to say how much each feature influences the success of the call. Here, the "precious_conversations" and "n_contacts_before" have the greatest impact on the outcome.

Numerical Analysis

After checking whether data is missing or not, the values and their occurrence are represented in a graphical chart. The y-axis displays the number of occurrences and the x-axis the values of the features.

Hereby it is possible to see, that the age of the customers is most likely in a range between 20 and 60 years with only very few customers age above this range. Often the customers were neither contacted about the ongoing nor last campaign.
The average age of the customer is 38 years and contacts were made at least two times throughout the current campaign. Mostly the days since last contact are marked with -1, which shows that most of the customers were not contacted before or no valid data is available. This manipulates the whole dataset and must be transformed into a zero or to give them a very high number like 999 to suit the purpose of this feature.

3.2.2 Feature Transformation

Like mentioned before the invalid values for "days_since_last_contact" have to be removed. Hereby all values -1 were replaced with 999 to pretend, that the customers were never contacted before. Also, a new feature is derived, which is called "days_since_last_contact_cat" and is a categorical version of the existing feature.

3.2.3 Feature Engineering

To improve the feature engineering "Featuretools" assisted the search. This package uses DFS for automated feature engineering. Based on "days_since_last_contact_cat" and "previous_conversion" the tool generated new features. These were checked about their correlation and discarded when the correlation is below 0.7.

The “duration” column contains leaky data values and was also sorted out, so that the F1-score will not be contaminated.

The existing features were summarized to the extent that the following questions (based on the approach of Joe Hoeller[4]) can be answered:

- *Was the customer contacted about the last campaign?*
- *Was the customer part of the last campaign?*
- *Was the customer contacted more than ten times?*

After a new categorical feature “add_campaign_gte10” was derivate from “n_contacts_before”, where the values are divided into categories above 10 times and below.

To reduce minor observation errors, “Discrete binning” is used in the pre-processing.[5] In closing, the age was binned. In this way groups can be formed without regard to target information. Here the feature age is logarithmized and transformed with “KBInsDiscretizer” to continuous data bins.

3.2.4 Oversampling

To counteract the potential over-fitting caused by the high bias, oversampling is performed. In this process the data of the minor part is amplified. There are two common approaches to do this:

The regular method

In the regular method the minor data is simply multiplied, so every row in the data-table with a positive success will be copied and pasted until 32.893 sets of this data is created. The biggest problem with this method are incorrect records in the given dataset. With 4.176 existing positive records, each incorrect record will appear at least seven times in the generated dataset.
Also, the weighting of the features is strongly influenced by such a large multiplication, which distorts the model.

SMOTE

SMOTE[6] generates new data similar but not identical to the given data to compensate for the imbalance. Hereby, the synthetical generated records can’t be treated as completely accurate. Because the algorithm is feed with only 4.176 records to create 28.717 more the generated records are difficult to trust

[4] cf. (Hoeller, 2019)
[5] cf. (Pitney bowes, 2017)
[6] cf. (Chawla, Bowyer, Hall, & Kegelmeyer, 2002)

→ One solution for the oversampling is a fusion between the regular method and SMOTE. By creating 10.000 records with the first method, the authenticity of the synthetical generated ones from SMOTE is significantly higher and counteracts against the distorted weighting of the features.

3.3 Evaluation

After comparing it with different Gradient Boosting Models like “xgboost”, “LightGBM” was the best fitting model for our task. While relatively new, lightweight and fast this framework generated the best scoring.

Like mentioned before the scoring model for the evaluation was the F1 score, which is depending on precision and recall:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

For splitting up training and test set the so called “StratifiedShuffleSplit” cross-validator was used, which can return stratified and randomized folds. “The folds are made by preserving the percentage of samples for each class.”[7]

4 Conclusion

After analyzing, tuning and transforming the feature as well as using an hybrid oversampling method to ensure a balanced dataset the final test result of 99.98% on the training set and 90.63% on the test set was achieved. To accomplish this an extensive data analysis is essential. Thanks to this, Featuretools and the GitHub-Repository of Joe Hoeller[8] the necessary feature tuning was done.

5 Summary and Outlook

Considering the objective of this challenge (>50% F1-score), a very good result was achieved. With more time an even better hyperparameter tuning would have been possible to improve the model even further. It is very likely that the imbalanced learning parameters of xgboost could have led to an even better result when using that framework. However, due to the general conditions for this challenge, the focus was put on feature engineering and in-depth testing.

[7] (sklearn, 2020)

[8] cf. (Hoeller, 2019)

V. Bibliography

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. In *Journal of Artificial Intelligence Research* (pp. 321-357).

Heichler, M., & Schulte-Austum, P. (2020, February 29). *Prediction Challenge*. Retrieved from GitHub: https://github.com/michaelheichler/predictionchallenge

Hoeller, J. (2019). *Machine Learning for Predictive Lead Scoring*. Retrieved from GitHub: https://github.com/joehoeller/Machine-Learning-For-Predictive-Lead-Scoring

Pitney bowes. (2017). Spectrum Technology Plattform - Machine Learning Handbuch. Stamford CT.

sklearn. (2020, February 28). *StatifiedShuffleSplit*. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

Sworn Declaration

Herewith I assure that the present term paper has been written by me independently and without unauthorized help, that I am taken all places, literally or roughly literally from publications, by quotations have marked as those. I explain that the work in same or similar form hast still been given to no exam authority/ exam place. I agree with the fact that the digital version of this work may become a highly loaded for plagiarism check on the servers of external suppliers. The plagiarism check shows no provision for the public

Patrick Schulte-Aust

Patrick Schulte-Austum