Capstone Project Report
Udacity Machine Learning Engineer Nanodegree

Felix Hauer

March 28, 2020

Contents

1 Definition

1.1 Project Overview

A business should know its customers. Not only to adapt the supply of the kind of goods the company produces or develops but also to use advertising right to get the attention of potential customers. Here it is important to speak out to the people who are most likely to buy a certain product and avoid those who are not interested at all. This will lead to a maximized return of each dollar that is put into the marketing budget. In this project, the business is a mail-order sales company in Germany. First, a data set of the customer base of the company is given to compare it to a data set that is representative of the population of Germany. A second data set of the company provides information about whether a person becomes a customer or not.

1.2 Problem Statement

The first part of the project is to determine similarities of the demographics of the general population of Germany with the demographics of customers of the mail-order company. This will be done with the help of a technique called *principal component analysis* or short *PCA*. The PCA allows us to perform a dimensionality reduction. The transformed data will be used as an input for the *k-nearest neighbors* algorithm. This method is used to inspect the similarities between the two data sets. In the second part, the problem that was already mentioned in the first chapter of the proposal is to deliver the advertising to the right kind of customer. This is done with the help of a machine learning model that needs to predict a probability of how likely a customer will buy a product. I will choose this model from different tree-based models, as these models perform best on tabular data. The success of this model is quantified with the metrics that will be introduced in the *Metrics* section.

1.3 Metrics

In this project, there will be two metrics to evaluate the performance of the model and one additional method to describe the predictive power of the model.

The first metric will be the accuracy which is defined as follows:

$$\mathrm{acc} = \frac{TP + TN}{N}$$

where N is the number of predicted observations, TP (true positive) is the number of right predictions that are in the positive class and TN (false negative) is the number of the right predictions that are in the negative class. The accuracy is the metric in which one of the benchmark models is also evaluated, so it will be used to compare the two models. As the model will give us a score or a probability for each observation, one will need a *threshold* to assign a class. This problem leads to a metric that can be calculated without a threshold.

The second metric is the *area under the ROC curve* or short *AUC*. The ROC curve is generated by plotting the true positive rate against the false-positive rate. The AUC is usually a number between $\frac{1}{2}$ and 1 corresponds with the U-statistic from the Wilcox-onMannWhitney test[1]. It is interpreted as the probability that a randomly drawn observation from the positive class has a higher score than a randomly drawn observation from the random class. The AUC is used in the Kaggle competition.

I also want to evaluate if the predicted probabilities are well-calibrated. This is important to do a cost calculation of a commercial campaign. The AUC is not suitable for this task, because is invariant to the absolute values of the class prediction. For example, you could add 0.1 to each prediction and the AUC would stay the same. I will use a so-called *calibration plot* to visualize this.

[1] $AUC = \frac{U}{n_1 n_2}$ where n_i, $i = 1, 2$ is the number of observations in class n_i.

2 Analysis

2.1 Data Exploration

The data that is used throughout this project consists of four datasets and was provided by Bertelsmann Arvato Analytics. The

- *AZDIAS* set consists of $891,211$ observations and 366 features and the
- *CUSTOMERS* set consists of $191,652$ observations and 369 features.

These two are datasets that are used for the first part of the project, which is a customer segmentation report. It is an unsupervised machine learning task, so there are no features that act as a target. The

- *MAILOUT* training set consists of $42,982$ observations and 367 features and the
- *MAILOUT* test set consists of $42,833$ observations and 366 features.

These two are datasets are used for the second part of the project, which is a supervised learning model. The 367th feature is the *response*, that needs to be predicted. In the test set this not available and it will be evaluated within the Kaggle competition. In addition to that, there are two Excel spreadsheets available that contain information about the data including attributes and the range of the data values. There are also a lot of missing values. That issue is discussed in section *Data Preprocessing.*

2.2 Algorithms and Techniques

In this section, I describe the algorithms and techniques that are used in the project.

2.2.1 Principal component analysis

Let n be the number of observations and p the number of variables $X_1, X_2, ..., X_p$. Assume that the mean of all X_i, $i = 1, ..., p$ is 0. One can always do this by subtracting the mean of the column from the column values. This does not change the variance and covariance of the column. The task is now to find a linear combination of the form

$$Z_i = \sum_{i=1}^{p} a_i X_i$$

under the constraint that the empirical variance is maximal and for the coefficients it is required to be normalized:

$$\sum_{i=1}^{p} a_i^2 = 1.$$

Now you do not keep all of the p linear combinations but only $k < p$, with k chosen that it is enough to represent a certain amount of the variance. Often it is recommended to keep around 80% of the variance.

2.2.2 K-means

The technique in this section is meant to solve an unsupervised learning problem. The goal of cluster analysis is to partition observations into groups, also called clusters, to that the pairwise dissimilarities between those assigned to the same cluster tend to be smaller than those in different clusters. As written in [James et al., 2013] the *K-means* algorithm is a iterative descent clustering method. It uses the squared Euclidean distance as the dissimilarity measure. One has to choose a decent number of clusters $k \in \mathbb{N}$. To do that, one can run the algorithm for several k and than choose the best k with the help of an elbow plot.

2.2.3 Tree based ensemble models

An ensemble model for such trees is a sum of such trees

$$f_M(x) = \sum_{m=1}^{M} \eta_m T(x; \theta_m)$$

where $T(x; \theta_m)$ is a decision tree in which splits and weights are expressed in terms of the parameter θ_m. η_m is the step length, which is usually constant for all m. While training, the next tree is fitted to the residuals of the latest model $f_{M-1}(x)$. Steps of this form are called boosting. Models of that form are the state of the art for tabular data. Simple decision and regression trees are introduced in [Kuhn and Johnson, 2013]. For details about boosted trees see [James et al., 2013].

2.3 Benchmark

As a baseline to check whether a model produces reasonable results, the first benchmark is a model that assigns every observation a positive label. This corresponds to the relative number of positive classes in the dataset. A developed model should at least beat this. The second benchmark will be a vanilla model. This is a very simple model. I will use a simple decision tree, which can be fitted which a small number of hyperparameters and is highly interpretable. The third benchmark will be the *Kaggle* leaderboard. Here I can compare my model with a community of the best data scientists of the world. I think a realistic expectation is to reach the top 25 %.

As already mentioned I will quantitatively compare the baseline model with the accuracy and the Kaggle competition with the AUC.

3 Methodology

3.1 Data Preprocessing

As a first step, I eliminated all predictors that are not contained in one of the two Excel spreadsheets available that contain information about the data. There is justification to use data points that are not well defined. After that I convert values that encode unknown or missing values to the NumPy data type:

```
np.nan
```

For example the predictor *AGER TYP* has the value -1 encoded as unknown. In figure 3.1 you see the predictors with the most missing values. After that, I remove all predictors where the percentage of missing values is smaller than 5%. This reduces the number of variables to 59. All categorical variables are then transformed as the *categorial* data type of *pandas* package and after that dummy variable are created. This is necessary because the unsupervised algorithms only work with numerical variables. I impute the numerical variables with the mean of the predictor and the categorical variable with the most common value. Finally, the variables are scaled and standardized.

All these step are performed for the azdias and the customers data set. The customers data set has 3 more columns that are also removed. Also, the *LNR* column is dropped because is an index.

3.2 Implementation

To implement the project I use the programming language python along with the following packages:

- NumPy
- pandas
- matplotlib
- scikit-learn

For the scikit-learn reference see [Pedregosa et al., 2011]. As an IDE I used a jupyter notebook that was also a good choice to present the code and results together along with this report.

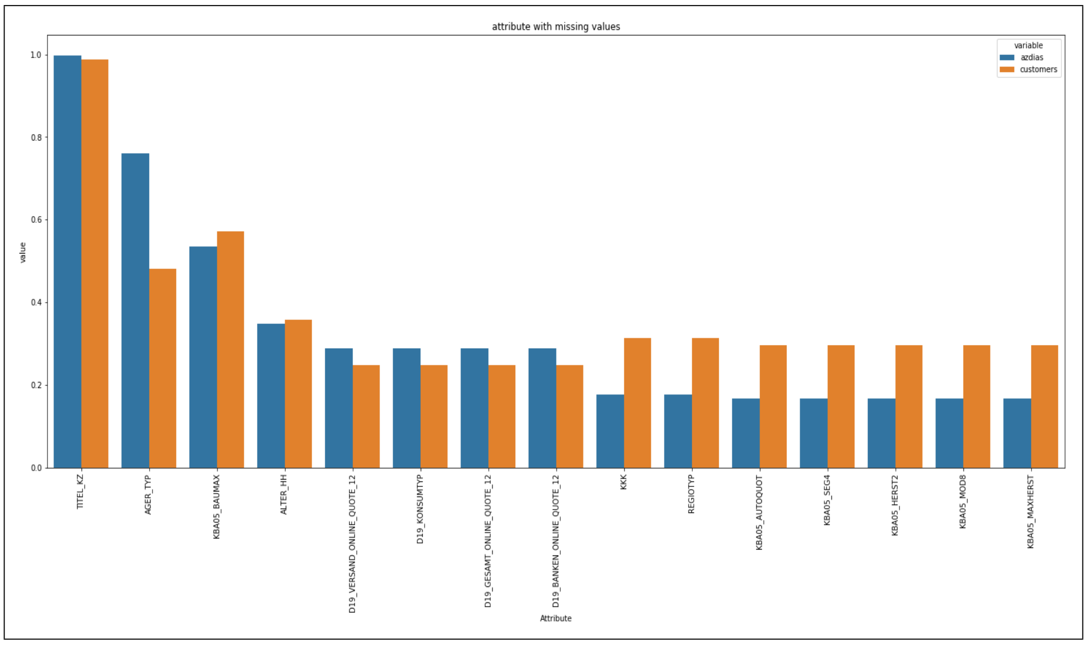


Figure 3.1: missing values

One problem I encountered, was that the data sets were quite big and it took some minutes to load the data and also some computations where time-consuming. In the development stage of the code, I only loaded some party of the data because of that.

3.3 Refinement

A common data science workflow is documented in [Wickham and Grolemund, 2017]:

- Import
- Tidy
- **Transform**
- **Visualize**
- **Model**
- Communicate

Especially those steps in bold are the main steps and should be repeated iteratively. Due to the time restriction of the project, I took the single steps atomary and refined them on the go. For example, I tuned the models with the data I tidied and transformed before that step and kept it like it was. If I had more time, I would go back to the tiding and transformation of the data to see how this improves the performance of the model. Because of this, there are no intermediate solutions, only the final one is available.

4 Results

4.1 Model Evaluation and Validation

4.1.1 Unsupervised Model

With the cleaned data set a pca is fitted for every data set. In figure 4.1 you see the plot of the azdias pca variance. In figure 4.2 you see the plot of the customers pca variance.

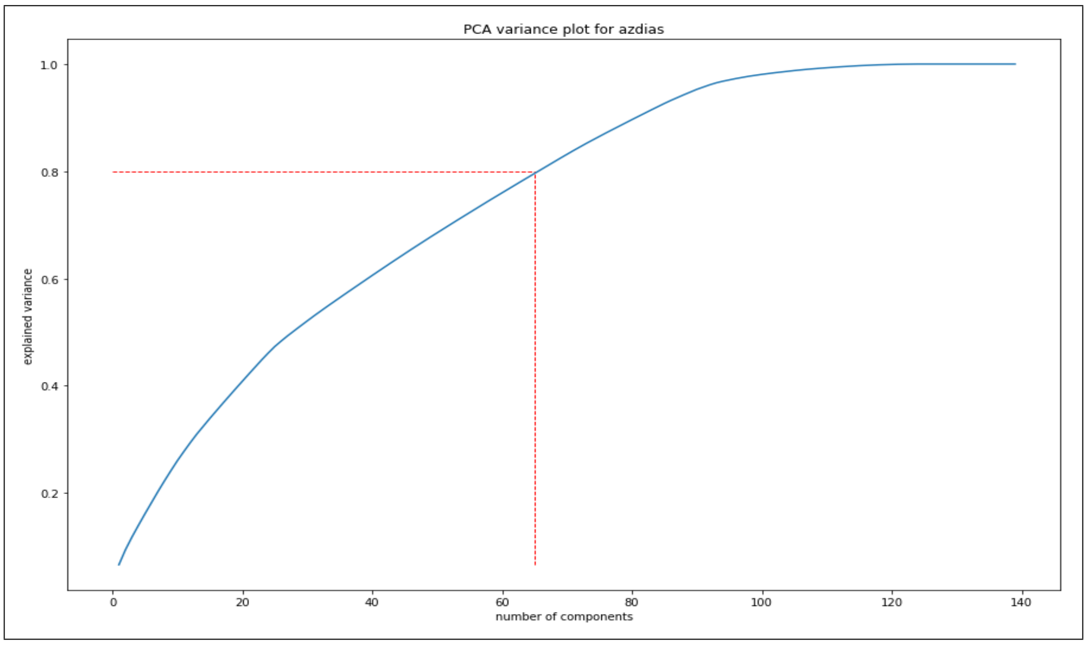


Figure 4.1: variance plot for pca azdias

From these plots, one can see that about 65 components are required to maintain 80% of the variance. With these components, I fit a K-means clustering for $k = 1...10$. In figure 4.3 you see the knn elbow plot of the azdias pca. In figure 4.4 you see the knn elbow plot of the customers pca.

With the help of these elbow plots, I choose $k = 7$ for the number of clusters. I fit a k-means model with $k = 7$ with the azdias data set and predict it on the azdias and customers data. The result of this clustering is seen in figure 4.5.

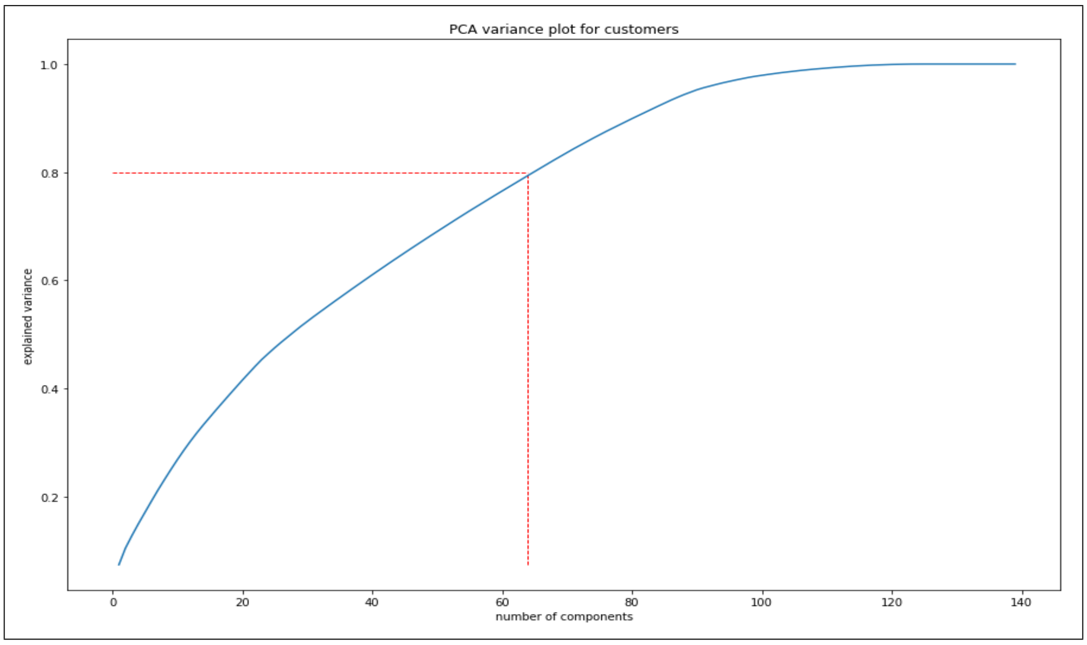


Figure 4.2: variance plot for pca customers

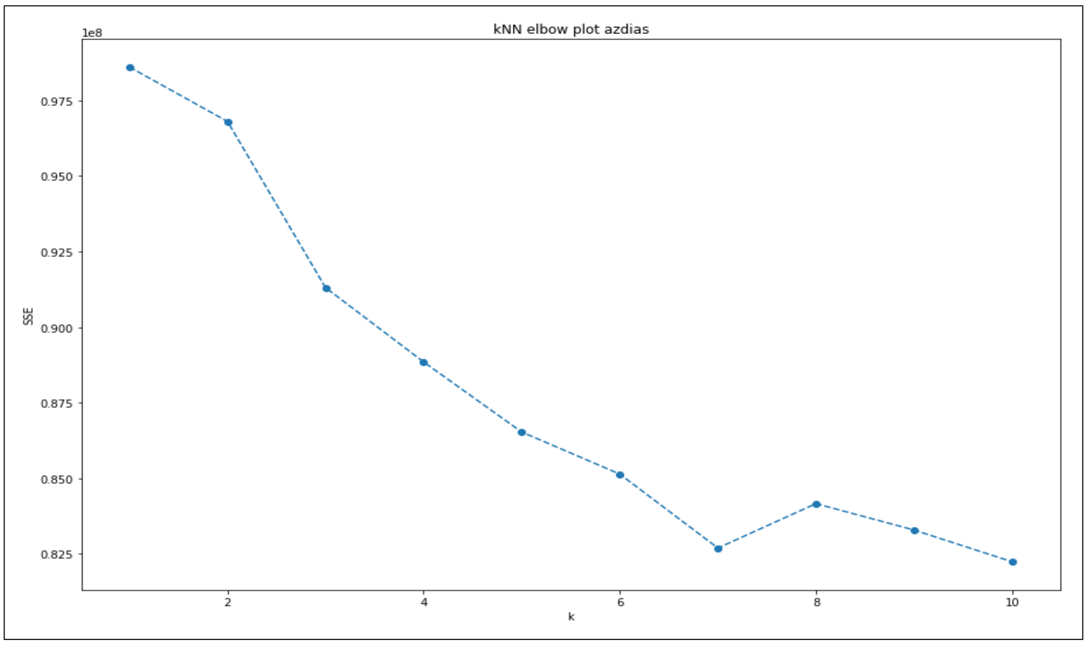


Figure 4.3: kNN elbow plot for the azdias pca

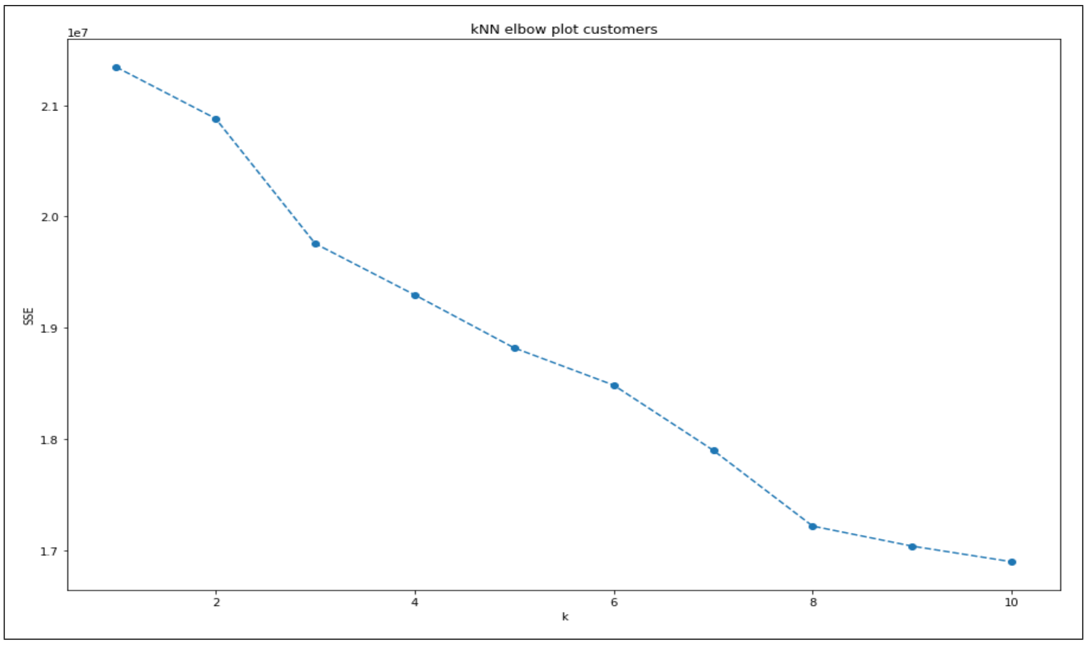


Figure 4.4: kNN elbow plot for the azdias pca

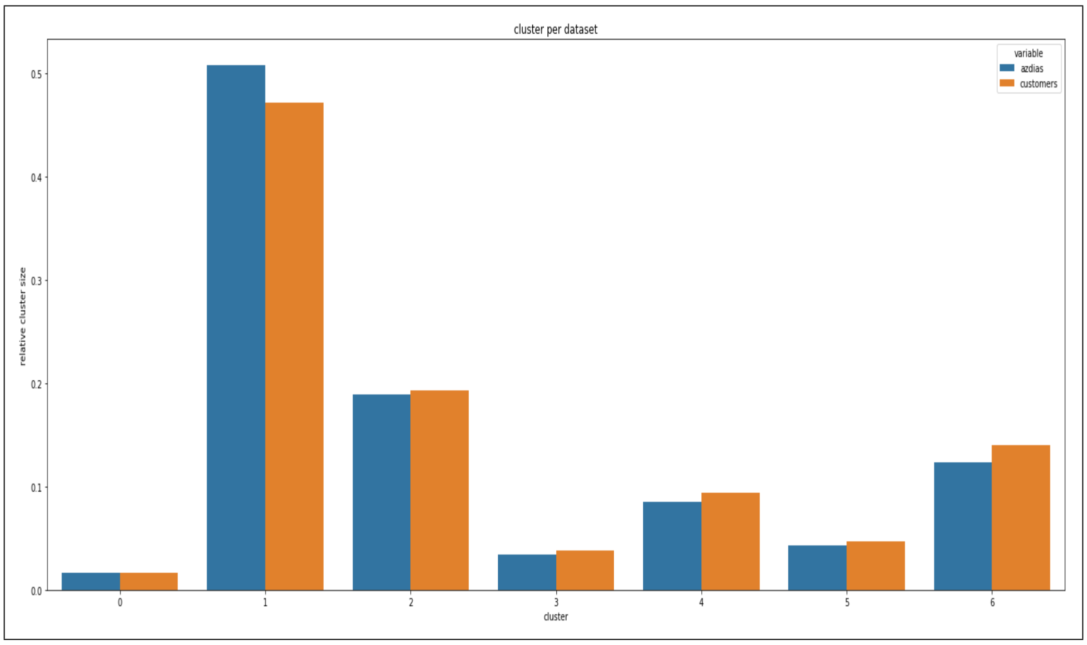


Figure 4.5: results of clustering

Now I select the clusters that are the most equal the most unequal. I take a relative measure regarding the relative cluster size to the whole population to determine this:

$$\frac{|cluster_{azdias} - cluster_{customers}|}{cluster_{azdias} + cluster_{customers}}$$

I select those clusters, where this number is the biggest and the smallest. This is clusters 1 and 6 respectively. Inspecting the components of the cluster that is the most equal. I look at the first two components as they describe the most variance and from these, I look at the five attributes with the largest magnitude. These are shown in figure 4.6 and 4.7.

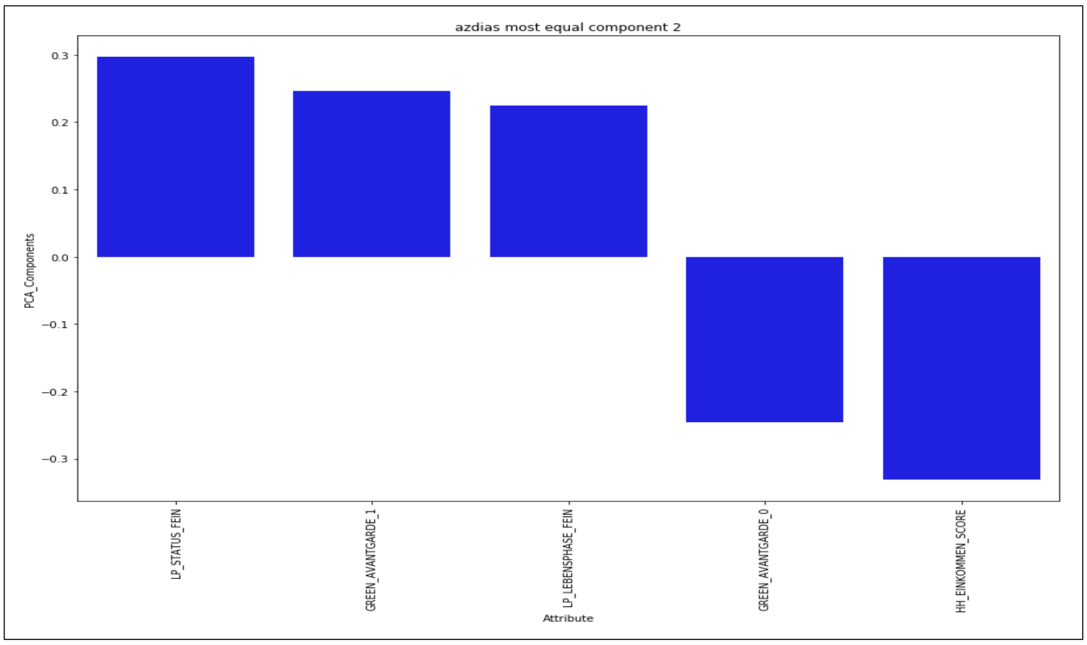


Figure 4.6: attributes of first equal component

For the cluster with the most unequal the attributes of the components are shown in figure 4.8 and 4.9.

The most common attributes are: 'LP STATUS FEIN', 'GREEN AVANTGARDE 1', 'LP LEBENSPHASE FEIN', 'GREEN AVANTGARDE 0', 'HH EINKOMMEN SCORE', 'D19 GESAMT ONLINE DATUM 1', 'D19 VERSAND ONLINE DATUM 1', 'D19 VERSAND DATUM 9', 'D19 GESAMT ONLINE DATUM 9', 'D19 VERSAND ONLINE DATUM 9'

and the most uncommon attributes are: 'ONLINE AFFINITAET', 'D19 BANKEN DATUM 10', 'D19 VERSAND DATUM 10', 'D19 VERSAND ONLINE DATUM 10', 'D19

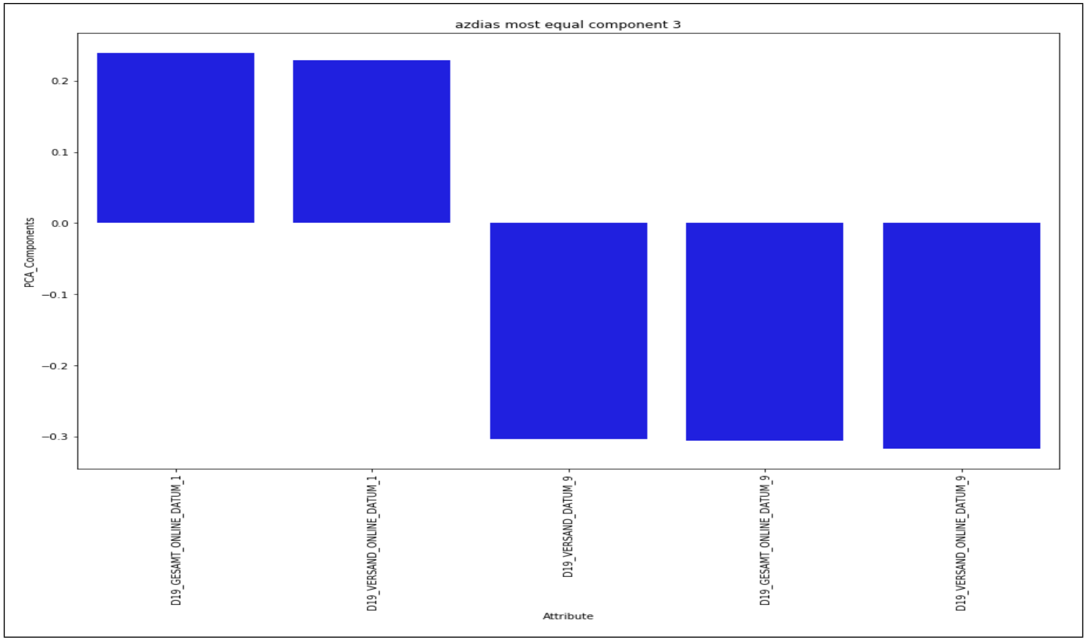


Figure 4.7: attributes of second equal component

GESAMT ONLINE DATUM 10', 'D19 GESAMT ONLINE DATUM 1', 'D19 VERSAND ONLINE DATUM 1', 'D19 VERSAND DATUM 1', 'D19 GESAMT ONLINE DATUM 5', 'D19 VERSAND ONLINE DATUM 5'

4.1.2 Supervised Model

For the supervised part of the project, two new data sets were given. One is intended for training containing an additional target columns and one is intended for testing, as it is the file for the Kaggle competition. I performed all preprocessing steps as described in the section *Data Preprocessing*.

The simple model that assigns every observation a positive label has an accuracy of 98.76%. A simple decision tree with a maximal depth of 6 has an accuray of 56.95%. I used the built-in option for adjustment of unbalanced classes from *scikit-learn*. Quote from the documentation of [Pedregosa et al., 2011]: *The balanced mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data.* The drop of 40% accuracy shows how important it is, to address the class imbalance.

To optimize for the Kaggle competition I use a train test split of the data with a ratio of 75% for training and 25% for testing. Here it is also important to take the class imbalance into account with a stratified split. I perform a grid search with cross-validation of the

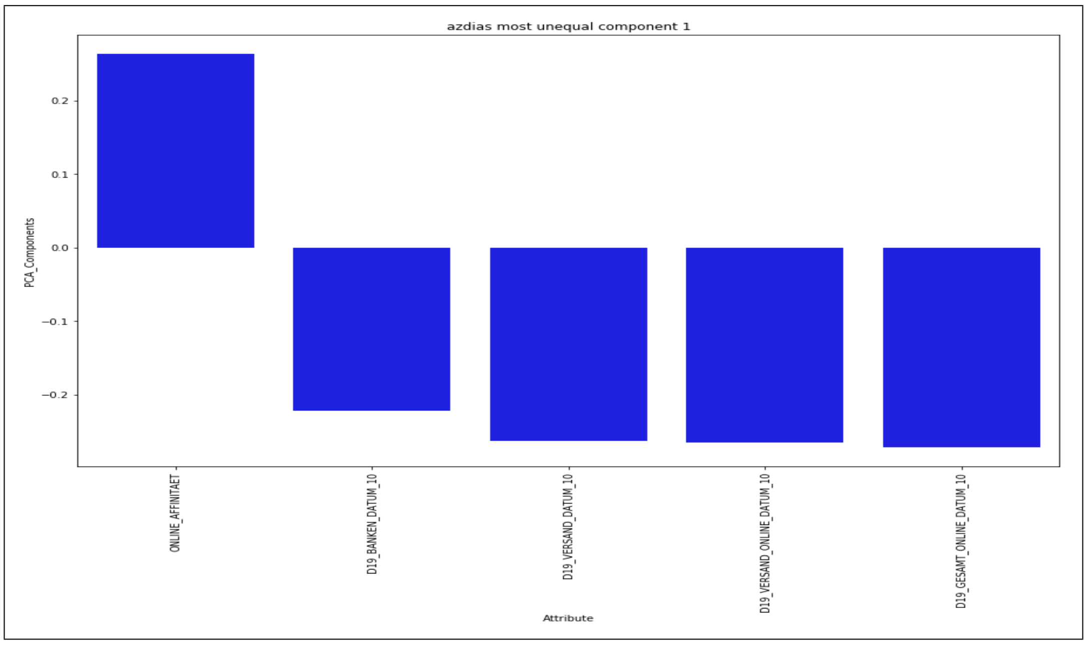


Figure 4.8: attributes of first unequal component

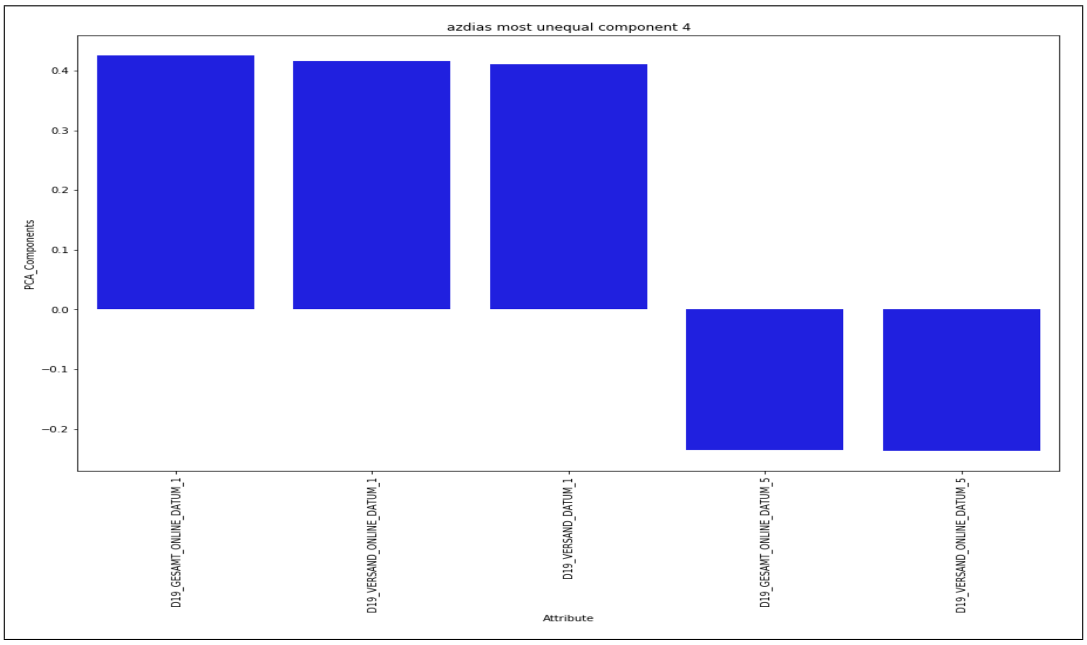


Figure 4.9: attributes of second unequal component

following models:

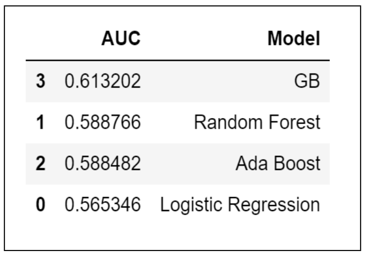
- logistic regression
- random forest
- ada boost
- gradient booster

The AUC of the each best hyperparameter set is shown in figure 4.10. In figure 4.11 and 4.12 the calibration plot for the training set and test set is shown respectively. Here the data is binned in steps of 10% For every bin the observed event rate is calculated and spread along the $y-$axis. On the $x-$axis there is mean predicted value of the bin. Because of the unbalanced classes, the plot is not as smooth as one would expect normally. This is especially the case in the test set, where the plot is quite useless. Nevertheless, the gradient booster has also the best calibration.

The best model is a gradient booster with the following hyperparameters:

```
{'learning_rate': 0.1,
'max_depth': 2,
'n_estimators': 100,
'random_state': 0,
'subsample': 0.75}
```

With this model, I predict the test to provide a solution to the Kaggle competition. The result is shown in figure 4.13. I reached an AUC of 0.6187%. At the time of submitting my solution there were 125 participants in the competition, so I reached the top 81%.



	AUC	Model
3	0.613202	GB
1	0.588766	Random Forest
2	0.588482	Ada Boost
0	0.565346	Logistic Regression

Figure 4.10: AUC of grid search

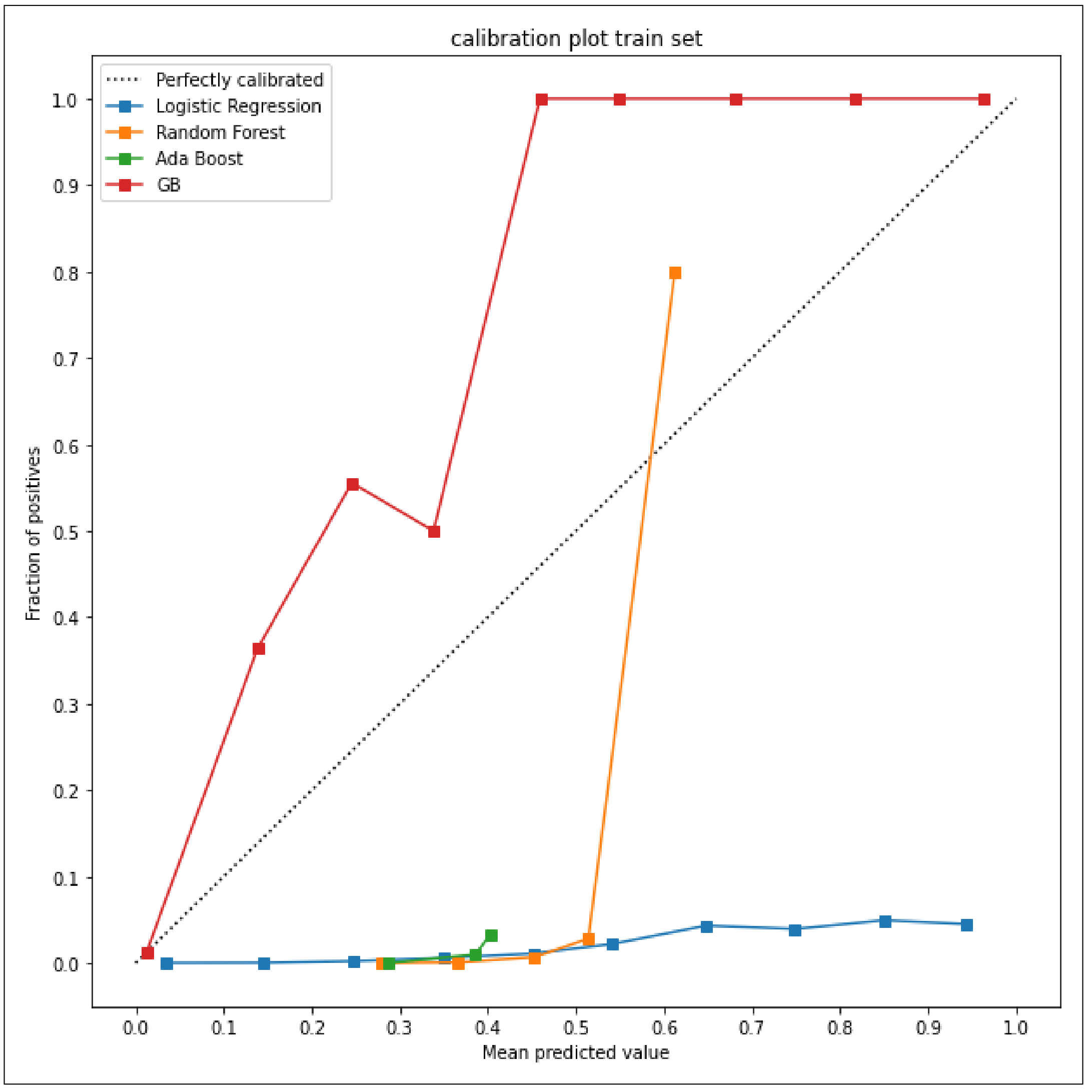


Figure 4.11: calibration plot of the train set

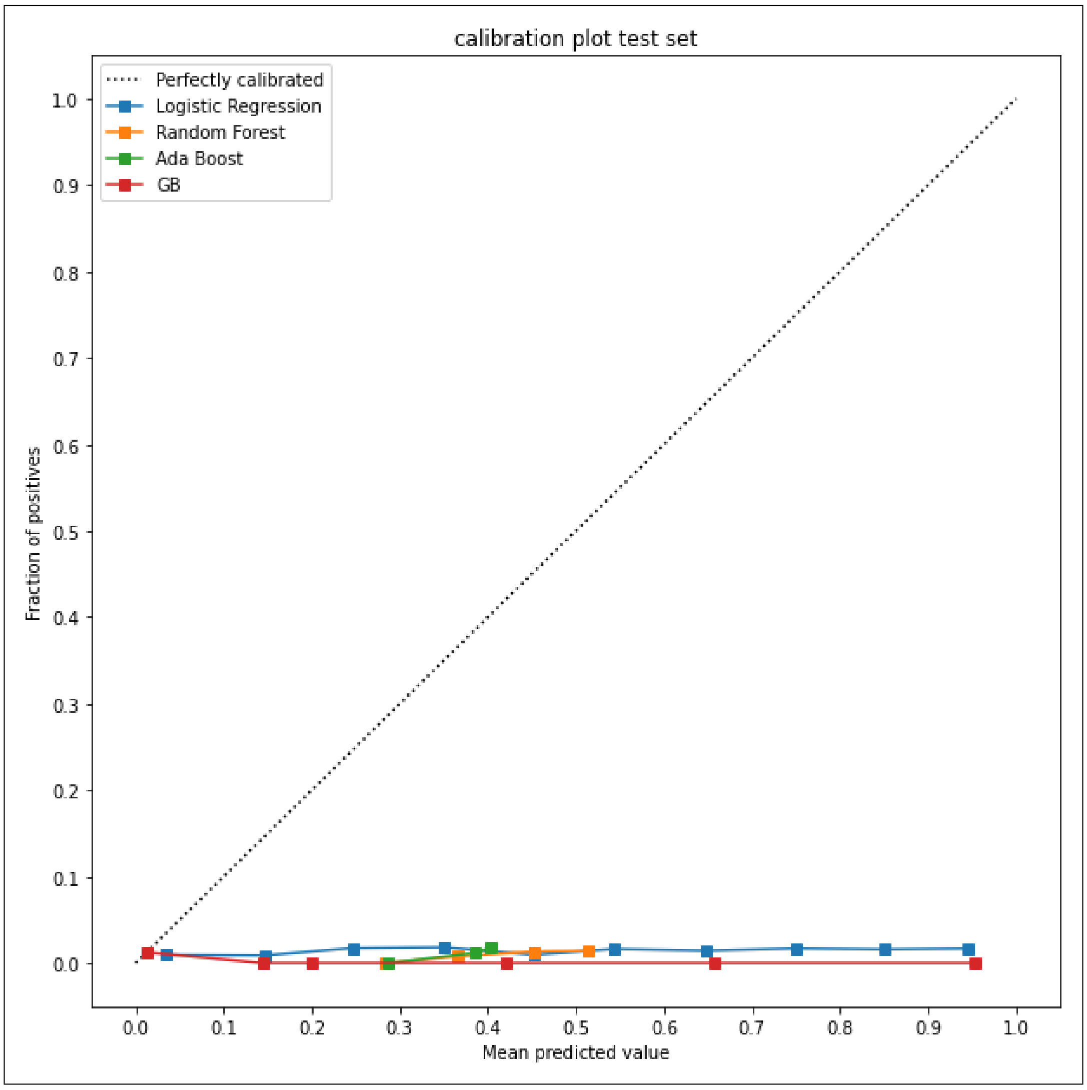
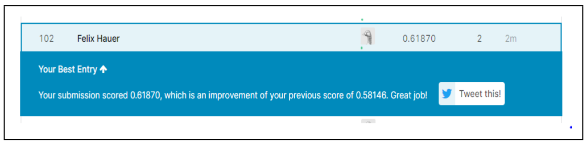


Figure 4.12: calibration plot of the test set


102 Felix Hauer 0.61870 2 2m

Your Best Entry

Your submission scored 0.61870, which is an improvement of your previous score of 0.58146. Great job!

Figure 4.13: results of kaggle

4.2 Justification

The result of the Kaggle competition was unfortunately not satisfying. As I explained in section *Refinement*, a data science project needs to implement a lot more iterations to obtain really good results. But from an educational point of view, this final model has sufficiently solved the task that Udacity gave me. I implemented all steps of a machine learning project that I learned during the machine learning engineering nano degree. Finally, I demonstrated my communication skills with this report.

Bibliography

[James et al., 2013] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer New York.

[Kuhn and Johnson, 2013] Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Wickham and Grolemund, 2017] Wickham, H. and Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. OReilly Media, Inc., 1st edition.

List of Figures