project_report.md

Machine Learning Engineer Nanodegree

Capstone Project

Pelle John August 15th, 2020

I. Definition

Project Overview

The Capstone Project is the final submission for Udacitys Machine Learning Engineer Nanodegree Program. The selected project is based on a real-life example of Arvato, an internationally active services company with a focus on innovations in automation and data/analytics. The requested analysis in about the provided customer data set of a German mail-order sales company to identify customer cluster which can be used to predict the success of targeted marketing campaigns.

The clustering of customers and a respective personalization is a must-have in today's business world. A recent RedPoint Global survey[1] conducted by The Harris Poll that surveyed more than 3,000 consumers in the U.S., U.K., and Canada stated, that

> 63 percent, of consumers expect personalization as a standard of service and believe they are recognized as an individual when sent special offers

It clearly shows that companies need to take into account the characteristics of a customer when choosing the channel, message, and method in a marketing campaign. Multiple studies show, that personalization has a huge impact on the success of a marketing campaign. A study in 2013 by Experian Marketing Services[2] shows that

> personalized promotional emails were shown to lift transaction rates and revenue per email six times higher than non-personalized emails.

The relevance in today's business world, as well as the variety of different machine learning methods to be used in this exercise, excites me the most. it gives the possibility to try different methods and explore different libraries and methods while having a clear goal in mind.

The request is split into three main objectives. First, get an overall understanding of the provided customer and German population database to derive an understanding of the data and clean/transform it in a way that it can be used for the following machine learning exercises.

Secondly, the cleaned data should be used to derive the population cluster which is mapped to the customer database to identify and analyze the relationships between the demographics of the company's existing customers and the general population of Germany. The goal of the second part of the analysis is to describe the parts of the general population that are more likely to be part of the mail-order company's main customer base, and which parts of the general population are less so.

Lastly, the provided understanding should be the basis to train a classification/prediction model based on an existing mailout campaign. In the end, we should be able to use the demographic information from each individual to decide whether or not it will be worth it to include that person in the campaign to and predict the likelihood of response for future mailout campaigns.

The analysis is based on four main datasets, which are all provided by Arvato. The first is a dataset of 891.211 persons (rows) and 366 features (columns) on demographics data for the general population of Germany. It will be used to define a general descriptive cluster for a later mapping of the mail-order company customer base. The customer database consists of 191 652 persons (rows) and 369 features (columns) with demographics data for customers of a mail-order company. It will be used to identify which cluster is in the current customer base of the mail-order company. For classification/prediction model are a train (42 982 persons (rows) with 367 (columns)) and a test dataset (42 982 persons (rows) x 366 (columns)) provided. The train has an additional "RESPONSE" column and will be used to train the classification model for the prediction of success of future targeted marketing campaigns. The test model is without the "RESPONSE" column and can be used in a KAGGLE competition for the success of future targeted marketing campaigns.

Problem Statement

The problem statement can be split into the main two tasks of the analysis: **Population/Customer Segmentation** and **Classification/Prediction**

Population/Customer Segmentation

The key problem to solve is to prepare and cluster the population data in a way, that we can define a clear cluster of interest for the mail-order company, once the customer base is mapped to the trained classificator. Therefore, the key objective is first to transform the dataset, to remove the noise, and to define the major differentiator.

The data transformation will be done in multiple steps. First, we get an overview of the available German population and customer base dataset (columns, data types, shape, unique, distribution, and percentage of the missing values per column). Once, we have a good understanding of the provided dataset, we start to identify missing values and replace them accordingly. Columns/rows, with a high percentage of missing data, will be removed from the dataset, as they don't provide clear differentiators for the later segmentation. Next, we encode the categorical values to be able to further work with them in the segmentation and transformation process.

After a data normalization step, the simplified German population dataset will be used for a PCA (principal component analysis), as it is a fast and flexible unsupervised method for dimensionality reduction in data. It involves zeroing out one or more of the smallest principal components, resulting in a lower-dimensional projection of the data that preserves the maximal data variance[3].

Once we reduced the noise and dimensions of the segmentation, we will train and optimize a **KMeans cluster algorithm** to detect customer segmentations.

The trained cluster model will be used on the customer dataset (which is transformed in the same way as the general population database) to compare the percentages of people per cluster in both datasets. The trained cluster model will show the population cluster, which has a higher/lower percentage as the general Population representation and, therefore, more probably more likely to respond to future marketing/sales campaigns.

Classification/Prediction

The key question of the second part of the analysis is: What are effective data transformation/classification approaches to have a stable and accurate prediction on the response rate of individuals on targeted marketing campaigns based on available potential customer demographic data. **The key challenge will be, that we will develop a classification/prediction engine with a very imbalanced dataset (most of the participants did not respond)**. Therefore, some steps in the data preparation will be done slightly differently to compensate for it.

Similar to the first start of the exercise, the analysis will start with the task to understand the provided datasets. After that, we will do similar data cleansing and transformation steps. The data cleansing and transformation will start to replace the missing data, to remove columns with high missing values percentages, to transform categorical values, to remove rows with high missing values percentages, drop non-important columns, replace NaN values and normalize the dataset for a PCA analysis.

The transformed dataset will be split in train and test dataset and then different classification algorithms (SVM, RandomForest,...) and over- and undersampling techniques will be used to train a sufficient classification/prediction model. The trained and evaluated models will be used on the TEST dataset for the KAGGLE competition.

Metrics

To determine the effectiveness of the used method/model, both parts of the analysis have seperate evalution metrics:

Population/Customer Segmentation: J. Kleinberg[5] defined three properties any clustering algorithm should try to satisfy: The axioms of scale invariance, richness, and consistency. He also proved an an impossibility theorem that shows that no clustering algorithm can simultaneously satisfy all of them. In our exercise we willl focus on finding the right number k (number of cluster) via the Elbow method and validated via Silhouette Coefficient, which is used as ground truth labels are not known and the evaluation must be performed using the model itself.

Classification/Prediction: For the classification and prediction we will use the accuracy, precision, recall and AUC scores[7]:

- **Accuracy**: Computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions $accuracy(y, \hat{y}) = \frac{1}{n_\text{samples}} \sum_{i=0}^{n_\text{samples}-1} 1(\hat{y}_i = y_i)$```
- **Precision**: Precision is the ability of the classifier not to label as positive a sample that is negative `tp / (tp + fp)`
- **Recall**: Recall is the ability of the classifier to find all the positive samples `tp / (tp + fn)`

- **AUC (Area Under the ROC Curve):** To better evaluate the correctness for the prediction of an inbalanced dataset, the incorporate the AUC mteric, which provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example[8, 9]: $\frac{2}{c(c-1)}\sum_{j=1}^{c}\sum_{k>j}^{c}(\mathrm{AUC}(j|k)\mathrm{AUC}(k|j))$

  TP = True Positives; FP = False Positives; FN = False Negatives

II. Analysis

Data Exploration

All four datasets for the analysis of the German mail-order sales company customer have the basis of 366 different columns, which can be split into the following sections:

1. **Personal information (43 columns)**: Information about the respective person in the dataset. This includes, e.g., information on age, gender, and typologies on financial characteristics, life stage, family, social status, and socioeconomic traits.
2. **Household information (32 columns)**: Information about the household the person lives in. This includes, e.g., the number of people, the household structure, transactional activities, and the duration of residence.
3. **Building & postcode and community information (19 columns)**: Information on the building the person lives in. This includes, e.g., the number of households, the type of building, car segments in the neighborhood, the distance from the city center or next metropole, inhabitants, and the share of unemployed persons in the community.
4. **Microcell RR1_ID, RR4_ID & RR3_ID (67 columns)**: Information about a cluster the respective person falls in. This includes the CAMEO typology segmentation and other information like, e.g., the share of car owners in the respective cell, the number of trailers, the number of 1-2 family houses, purchasing power, moving patterns, and online affinity.
5. **AZ Cluster data - 125m x 125m Grid (33 columns)**: Information on transactional data from the mail-order activities for a specific product group for a specific grid. Is based on data from AZ, which has access to 650 Million transaction data.
6. **Postal code related statistics - PLZ8 (114)**: Based on federal German statistics on the postal code. Contains column-like, e.g., the share of car owners per type like BMW, the car engine power, and most-common car types.

Only for customer dataset: The customer set has three additional columns "PRODUCT_GROUP, CUSTOMER_GROUP, ONLINE_PURCHASE" which gives information about the respective product group and online_purchase information.

Only for mailout train dataset: The dataset has one additional column "REPONSE" to support the training of a supervised classificator.

The datasets for Segmentation/Clustering exercise have the following characteristics:

```
Description about data types in population database:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891221 entries, 0 to 891220
Columns: 366 entries, LNR to ALTERSKATEGORIE_GROB
dtypes: float64(267), int64(93), object(6)
memory usage: 2.4+ GB
None

Description about data types in customer database:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 191652 entries, 0 to 191651
Columns: 369 entries, LNR to ALTERSKATEGORIE_GROB
dtypes: float64(267), int64(94), object(8)
memory usage: 539.5+ MB
None
```

Both datasets have categorical columns and a large set of data. One of the key challenges will be to reduce the number of input data so we can process it efficiently. None of the columns has more than one data type and columns are sorted in alphabetical order. A check for uniqueness of the values revealed, that only one column has more than 500 different data entries and is unique:

```
First impression of data LNR:
0    910215
1    910220
2    910225
```

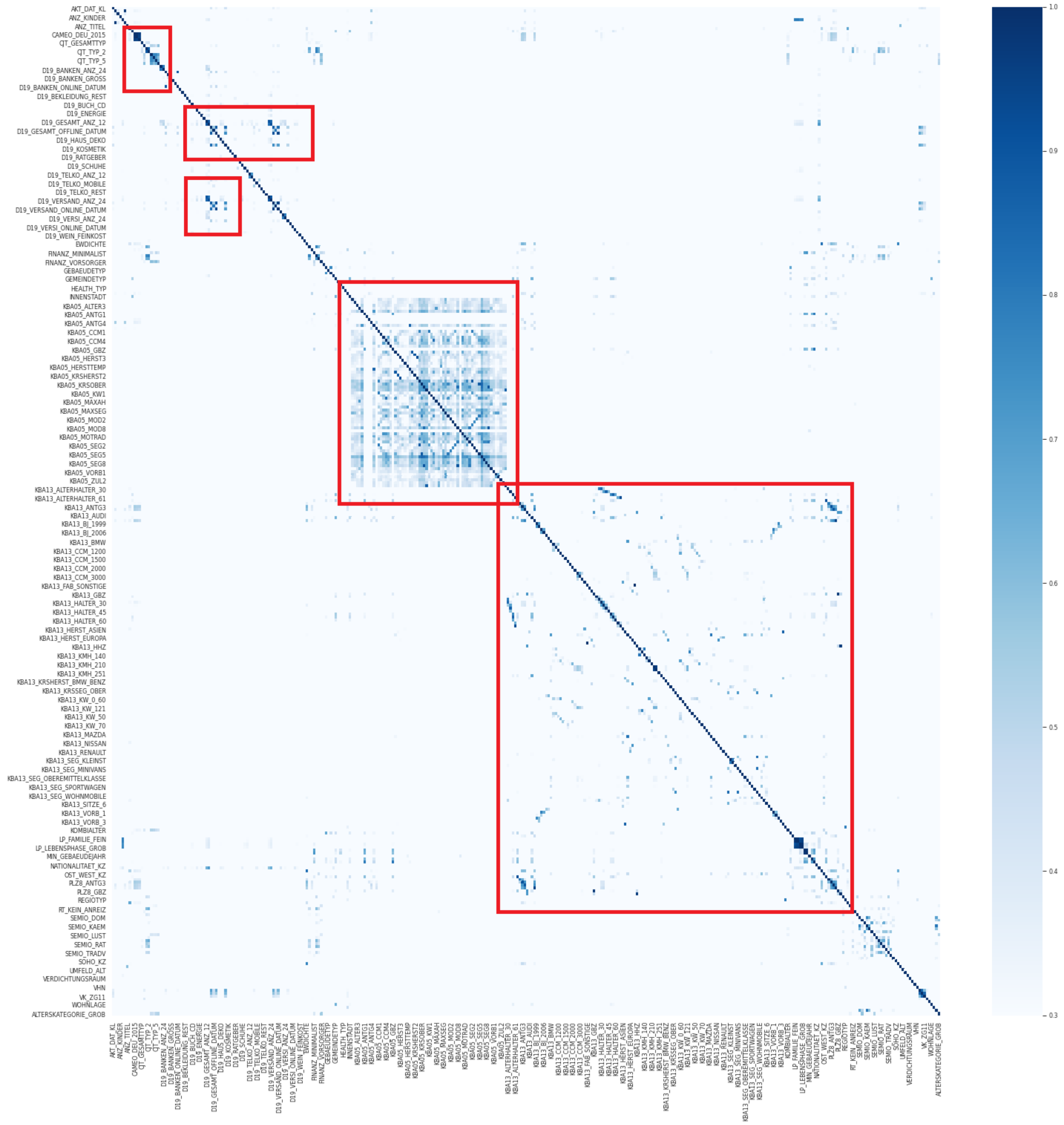
```
3    910226
4    910241
Name: LNR, dtype: int64
Column data types: int64
Sum empty values (absolute/percentage): 0 / 0.0
Rows values unique identifier (Length dataset vs. unique values): True
```

LNR is the index, which we can later use for the KAGGLE competition. A check-in the documentation revealed, that LNR acts as an ID number for each individual in the data partition. The second highest value distribution was in the column ["EINGEFUEGT_AM"]. As this is a piece of plain information on the last update/insert, we also exclude it for further analysis.

Note: Some column information are described in more than one column in the dataset (e.g., no. of kids). Also additional columsn exist for the creation of the row, the last update and a unique identifier. Therefore, the number of columns per category do not exactly add up to 366 columns.

Exploratory Visualization

The number of features is insufficient for a clustering and segmentation algorithm. To better assess the potential, a correlation matrix for the dataset on the German population was composed to identify potentials for improvement.



The axes list the features in the German population dataset. The color-encoding represents the correlation between the respective features. Dark blue represents a high correlation close to 1. White represents a low correlation close to 0. A reduced color-density was used to highlight the areas with a very high correlation.

The correlation matrix shows, that there are multiple feature sets with high correlation. The most important observations are highlighted in red. Especially for the Postal Code related statistics around cars and for the Microcell features. A PCA should be performed to reduce the noise and highlight, extract the most important features in the dataset.

Secondly, to further identify possible improvement potentials, an analysis was performed to identify the columns with more than 25% missing values. The result was the following:

```
['ALTER_KIND4', 'ALTER_KIND3', 'ALTER_KIND2', 'ALTER_KIND1', 'AGER_TYP', 'EXTSEL992', 'KK_KUNDENTYP', 'ALTER_HH',
'ALTERSKATEGORIE_FEIN', 'D19_LETZTER_KAUF_BRANCHE', 'D19_SOZIALES', 'D19_VERSAND_ONLINE_QUOTE_12', 'D19_LOTTO',
'D19_KONSUMTYP', 'D19_TELKO_ONLINE_QUOTE_12', 'D19_BANKEN_ONLINE_QUOTE_12', 'D19_VERSI_ONLINE_QUOTE_12',
'D19_GESAMT_ONLINE_QUOTE_12']
```

```
Number of columns with more than 25% missing values = 18
```

These features are partly redundant based on other features. Therefore, it will be a good step to exclude them from further data cleansing and transformation process. Similar exercises were done for the MAILOUT train dataset for the prediction/clustering algorithm, which resulted in similar results.

Algorithms and Techniques

The following algorithms/techniques were used for the Segmentation/Clustering and for the Classification/Prediction analysis:

1. LabelEncoder() / OneHotEncoder(): Used to transform non-numerical labels (as long as they are hashable and comparable) to numerical labels.
2. MinMaxScaler() / StandardScaler: This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one. The transformation is given by:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

3. df.fillna(df.mean()): Fill NA/NaN values using the medium of the specific column. Similar to: SimpleImputer(strategy='mean').
4. NaN_replace_KNNImputer]: The KNNImputer class provides imputation for filling in missing values using the k-Nearest Neighbors approach.
5. PCA(): Principal component analysis (PCA) is a technique for reducing the dimensionality of large datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance.[3]

Population/Customer Segmentation

6. KMeans(): The KMeans algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares This algorithm requires the number of clusters to be specified. It scales well to a large number of samples and has been used across a large range of application areas in many different fields[4].

The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares criterion:

$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$

Classification/Prediction

For the classification/prediction, the analysis tries two different algorithms:

8. **Ensemble Algorithms**:
   i. GradientBoostingClassifier(): Gradient Tree Boosting or Gradient Boosted Decision Trees (GBDT) is a generalization of boosting to arbitrary differentiable loss functions. GBDT is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems in a variety of areas including Web search ranking and ecology.
   ii. XGBoost():: XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.[10]
9. StratifiedKFold: KFold divides all the samples into k groups of samples, called folds (if k-1, this is equivalent to the Leave One Out strategy), of equal sizes (if possible). The prediction function is learned using k - 1 folds, and the fold left out is used for the test. StratifiedKFold is a variation of k-fold which returns stratified folds: each set contains approximately the same percentage of samples of each target class as the complete set.
10. GridSearchCV(): Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn, they are passed as arguments to the constructor of the estimator classes. Typical examples include C, kernel, and gamma for Support Vector Classifier, alpha for Lasso, etc. It is possible and recommended to search the hyper-parameter space for the best cross-validation score. Any parameter provided when constructing an estimator may be optimized in this manner.

Benchmark

A benchmark is best suited for the classification exercise to predict the success of a targeting marketing campaign. Similar exercises on Kaggle (see below) assume a prediction rate of 75% to 80% ROC score. That is what we will aim for in this exercise. Nevertheless, the analysis will show if we can achieve that with the imbalance of the dataset. The respective Kaggle competition in the exercise suggests between 75% and 80% RUC score.

Similar Kaggle Notebooks/ Competitions:

- https://www.kaggle.com/c/springleaf-marketing-response
- https://www.kaggle.com/c/udacity-arvato-identify-customers/leaderboard

III. Methodology

Population/Customer Segmentation

Data Preprocessing

The following steps have been conducted to preprocess the data for the **Population/Customer Segmentation** exercise for both datasets.

1. The first we need to do with the datasets is to **replace and handle missing values**. Therefore, we need to identify values that represent NaN and replace it. Moreover, we want to focus on dates that are filled more than 70% to be used in the dataset. Before we start the replacement we want to check if the datatypes in the respective column are the same. The following values were identified as wrongly placed NaN values:

```
zero_unkown_list = ['ALTER_HH', 'AGER_TYP', 'CJT_GESAMTTYP', 'HEALTH_TYP']
XX_unkown_list = ['CAMEO_DEU_2015', 'CAMEO_DEUINTL_2015']
overall_replace_values = [-1] #function replaces all values >= 0
```

2. Now that all values have been properly replaced with np. NaN values, we want to exclude the columns which have more than 25% NaN values. We save the values in a string to make sure that we, later on, drop the columns for the customer dataset as well:

```
['EINGEFUEGT_AM', 'LNR', 'ALTER_KIND4', 'ALTER_KIND3', 'ALTER_KIND2', 'ALTER_KIND1', 'AGER_TYP', 'EXTSEL992', 'KK_KUNDENTYP',
'ALTER_HH', 'ALTERSKATEGORIE_FEIN', 'D19_LETZTER_KAUF_BRANCHE', 'D19_SOZIALES', 'D19_VERSAND_ONLINE_QUOTE_12', 'D19_LOTTO',
'D19_KONSUMTYP', 'D19_TELKO_ONLINE_QUOTE_12', 'D19_BANKEN_ONLINE_QUOTE_12', 'D19_VERSI_ONLINE_QUOTE_12',
'D19_GESAMT_ONLINE_QUOTE_12']
```

Note: We added the unique identifier ['LNR'] and the update field ['EINGEFUEGT_AM'], as they have more than 50 categories and produce a lot of unnecessary noise in the overall clustering model.

3. Next, we want to identify the values which are not **float** or **int**, and therefore will need some additional encoding. Looping through the types of columns result in the following dates:

```
['CAMEO_DEU_2015', 'CAMEO_DEUG_2015', 'CAMEO_INTL_2015', 'OST_WEST_KZ']
```

4. The categorical columns will be encoded with the Algorithm described LabelEncoder(). Note: A future improvement of the segmentation could be to encode them with OneHotEncoder, to reduce the possibility of connections between the data in a column by the clustering model. The problem is since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order, 0 < 1 < 2. But this isn't the case at all. To overcome this problem, we use One Hot Encoder.[11]

5. The still-missing values will be filled via the KNNImputer and then normalized for the Principal Component Analysis. Before we start the analysis, a final check was conducted across the General Population dataset:

```
azdias_td_3.head(10)
azdias_td_3.describe()
```

	AKT_DAT_KL	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN	ANZ_STATISTISCHE_HAUSHALTE	ANZ_TITEL	ARBEIT	BALLRAUM
0	0.432021	0.022000	0.002683	0.017087	0.038411	0.020759	0.002167	0.270395	0.524047
1	1.000000	0.029178	0.000000	0.000000	0.044444	0.032698	0.000000	0.250000	0.833333
2	1.000000	0.026525	0.000000	0.000000	0.022222	0.019074	0.000000	0.250000	0.166667
3	0.000000	0.002653	0.000000	0.000000	0.000000	0.005450	0.000000	0.125000	0.500000
4	0.000000	0.007958	0.000000	0.000000	0.088889	0.008174	0.000000	0.375000	0.166667
5	0.000000	0.013263	0.000000	0.000000	0.022222	0.005450	0.000000	0.125000	0.833333
6	1.000000	0.010610	0.000000	0.000000	0.022222	0.008174	0.000000	0.375000	0.833333
7	0.000000	0.015915	0.000000	0.000000	0.022222	0.013624	0.000000	0.125000	0.166667
8	1.000000	0.005305	0.066667	0.000000	0.022222	0.005450	0.000000	0.125000	0.333333
9	0.500000	0.023873	0.000000	0.000000	0.022222	0.019074	0.000000	0.125000	0.833333

10 rows × 346 columns

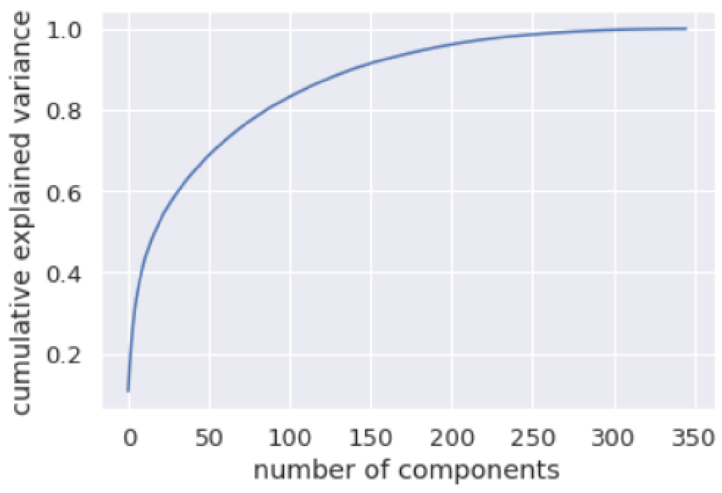
	AKT_DAT_KL	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN	ANZ_STATISTISCHE_HAUSHALTE	ANZ_TITEL
count	891221.000000	891221.000000	891221.000000	891221.000000	891221.000000	891221.000000	891221.000000
mean	0.427744	0.018480	0.002431	0.020848	0.072016	0.020018	0.001549
std	0.435692	0.033360	0.018049	0.066353	0.057137	0.035851	0.025766
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.003731	0.000000	0.000000	0.032258	0.004454	0.000000
50%	0.375000	0.010929	0.000000	0.000000	0.054054	0.011299	0.000000
75%	1.000000	0.021359	0.000000	0.000000	0.090909	0.021869	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 346 columns

The datasets seem correct and can, therefore, be further processed via PCA.

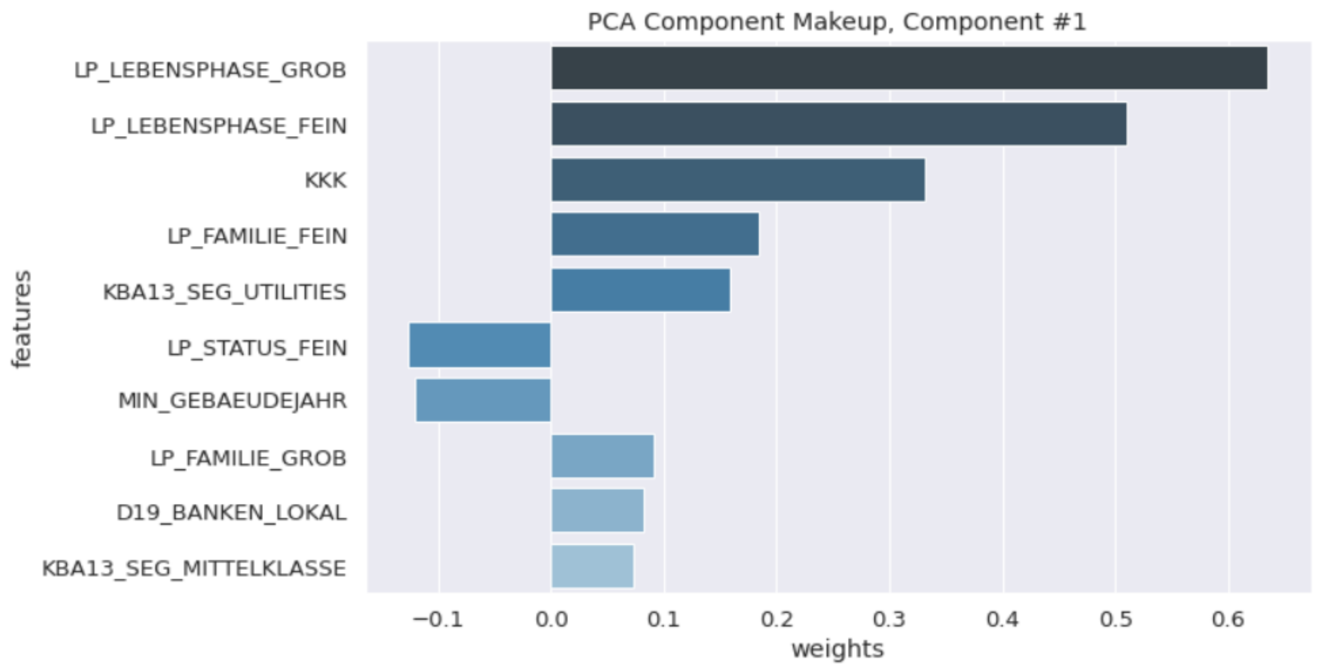
Note: A further optimization potential is to further process columns with a very one-sided skewed data distribution or data with more than 40 categories to further reduce noise in the dataset for the clustering model.

6. Next, a PCA() Principal component analysis was conducted. The following diagram shows the portion of variance explained by the number of components. The analysis suggets t gather around 85% to 95% of the variance. For the further segmentation analysis, 150 variables a composed explaining >90% of the variance of the dataset. The result of the PCA() is, that we could reduce the number of dimensions from 346 to 150 (which also simplifies the training) and were able to reduce the overall noice in the dataset.



```
# test cell
n_top_components = 110 # select a value for the number of top components

# calculate the explained variance
exp_variance = hps.explained_variance(s, n_top_components)
print('Explained variance: ', exp_variance)

Explained variance:  0.8507474950159595
```

Moreover, to get a first insight in the dataset, we display the makeup of the first component, explaining the highest portion of variance:



The composition of the PCA results in the final dataset, ready for the **Population/Customer Segmentation**:

```
azdias_td_cp_3.head(10)
```

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_10	...	c_141	c_142	c_143	c_144
0	-0.940959	-1.292724	-0.599544	0.069410	-0.038011	-0.116565	-0.297340	1.483230	-0.189984	-0.012328	...	-0.016141	-0.141301	0.056076	-0.016263
1	-1.698720	1.406102	-1.091438	0.991934	-0.610289	-0.087393	0.410089	-0.822642	-0.395504	0.442738	...	-0.043998	0.094798	-0.144388	0.352395
2	0.075493	0.259522	-0.803335	1.321752	-0.867229	-0.265853	1.117078	-1.189231	0.452386	-0.474703	...	-0.194478	-0.210145	0.106159	-0.078361
3	1.493806	-1.482566	-0.698831	0.461538	-0.373421	-0.117948	-0.209348	-1.120283	-0.686431	0.484641	...	-0.203564	-0.072872	0.006045	-0.071840
4	1.393217	1.508875	1.959326	0.404456	0.863899	-0.021666	-0.975171	0.118498	-0.789137	-0.477551	...	-0.447260	0.314581	0.070487	0.183362
5	0.557193	-0.613681	-0.574513	0.447376	-0.014921	-0.173497	0.161210	-0.123479	-1.087221	-0.693972	...	-0.097459	-0.134772	-0.028441	-0.140981
6	-0.701115	0.637016	-2.045778	-0.193887	-0.566011	-0.205352	-0.163734	-0.422043	0.976625	-0.394698	...	-0.132642	-0.174270	0.061703	0.017593
7	-0.027647	-0.525865	-0.662858	1.064346	1.748022	-0.176905	0.788511	-0.461813	0.479724	-0.550467	...	0.163525	0.030069	0.097419	-0.000491
8	0.178028	-0.735876	-0.743874	2.147164	0.793837	-0.328023	0.719821	-0.772320	0.677980	-0.339341	...	0.238357	0.119299	0.059404	-0.043180
9	0.943448	0.846092	0.845289	2.897276	-1.051785	-0.193484	1.046799	-0.453695	-0.532542	-1.229785	...	0.109905	-0.183631	0.247604	0.001578

Implementation

The implementation was smooth and no major changes/specifications have been made. The dataset was split up multiple times to be able to process them in the respective workbook without running out of memory. All helper functions to process the data were documented in an additional file.

Refinement

The advancements have been laid out throughout the data processing process. Summarized, the following three improvement potential could be picked up by future students:

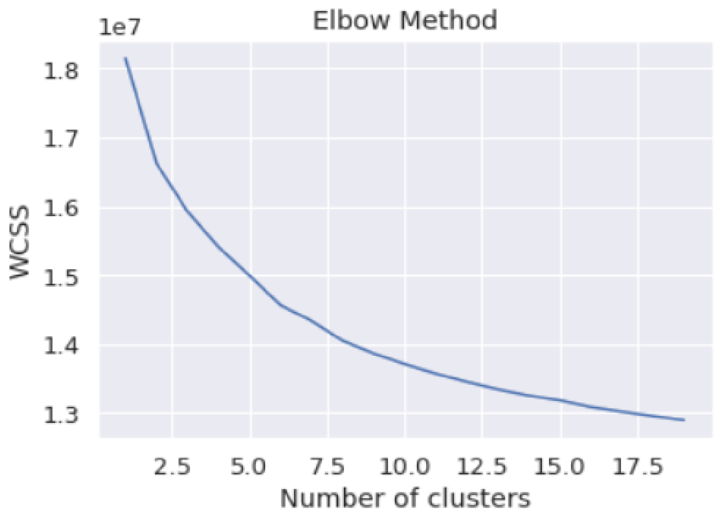
1. Remove the rows with significant NaN values
2. Separate processing strategies for numerical and binary categories
3. Use an OneHoteEncoder for categorical categories
4. Transform columns with one-sided data distributions

IV. Results

Model Evaluation and Validation

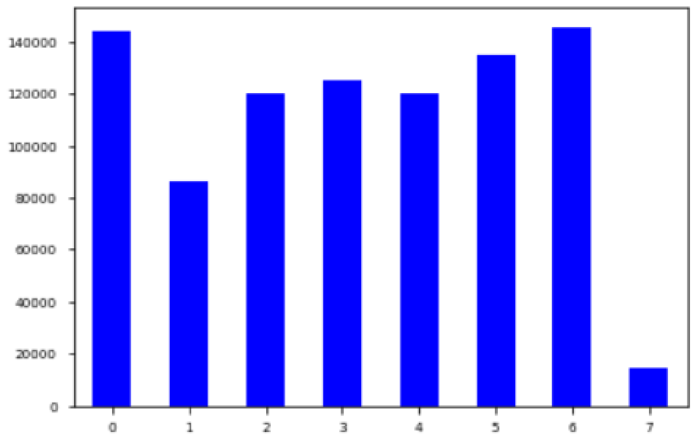
Research suggests, that K-Means is the standard clustering and segmentation algorithm for big datasets[12,13]. Therefore, it was decided to use this algorithm for the exercise. To determine the number of clusters, the elbow method was used which runs k-means clustering on the dataset for a range of values for k (from 1-20) and then for each value of k computes an average score of inertia. Inertia gives the sum of squared distances of samples to their closest cluster center. To determine the optimal number of clusters, we have to select the value of k at the “elbow” ie the point after which the inertia starts decreasing linearly. Thus for the given data, we conclude that the optimal number of clusters for the data is **8**.

```
#Find the right K Cluster
wcss = []

for i in range(1, 20):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(azdias_td_cp_3)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 20), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

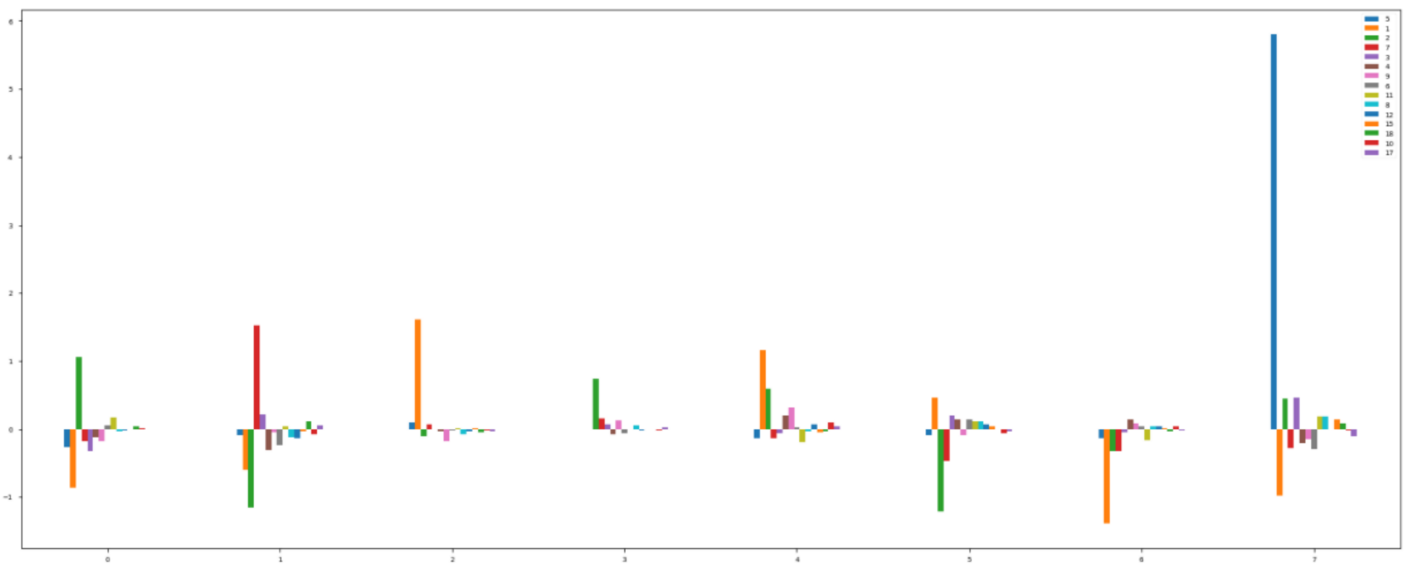


The number of eight clusters is used to fit the model on the German population dataset:

```
# The right value of K is - 8 is the number to go
kmeans = KMeans(n_clusters=8, init='k-means++', max_iter=300, n_init=10, random_state=0)
pred_y = kmeans.fit_predict(azdias_td_cp_3)
ax = cluster_df.plot.bar(rot=0, color='blue')
```

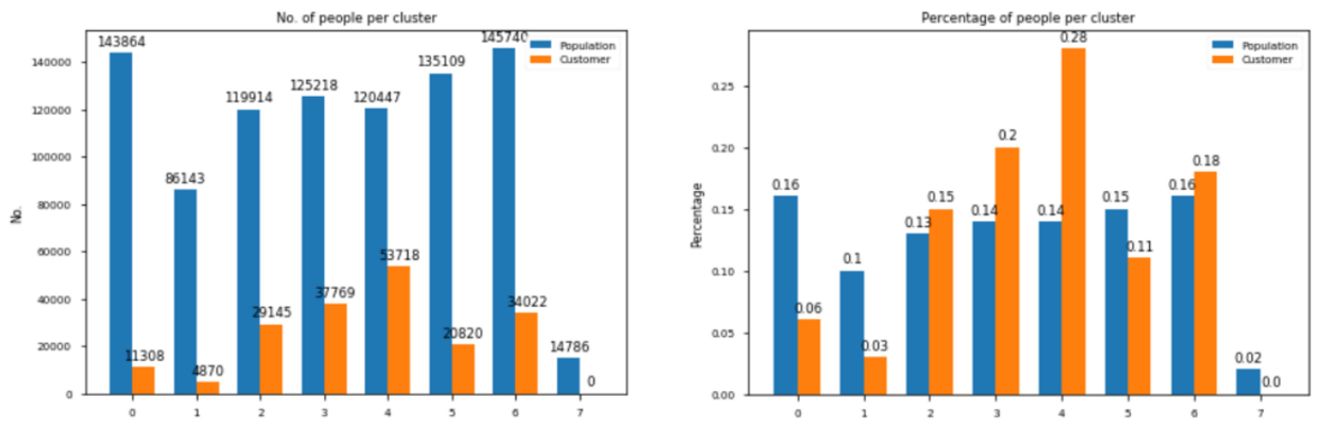


The eight clusters have a fairly even distribution besides the cluster seven with a population size from 80.000 to 140.000. To better understand the composition of a cluster, the component/dimensions wit the highest variance across the eight clusters are displayed:



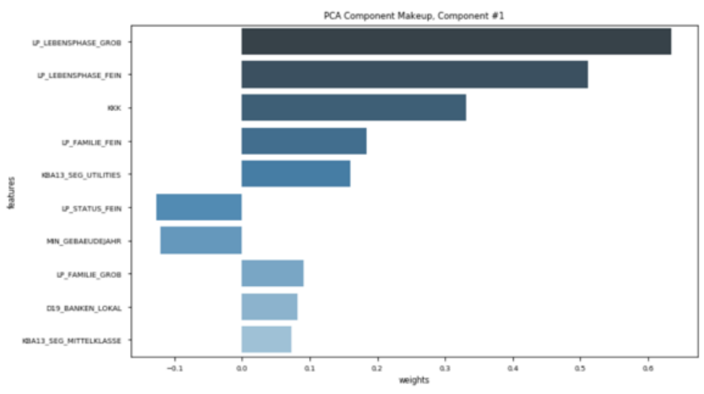
The plot shows the eight clusters on the x-axis and the variance of the respective components on the y-axis. For example, A high correlation with component 5 was the main factor for the distribution to cluster 7. The main 15 dimensions in the dataset were are the ones who explain the most variance among them. Components 5, 1, 2, and 7 play a significant role. They will be used to better explain the customer cluster, once they are mapped.

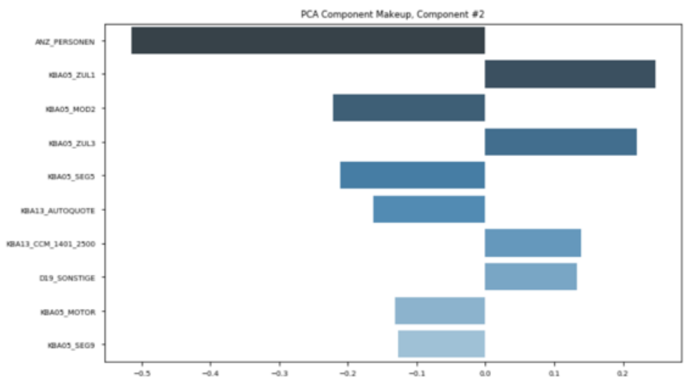
Next, the customer dataset is transformed along with the data processing steps of the German population dataset and the KMeans model was used to determine the cluster for each person in the dataset. The following plot displays the absolute and percentage distribution for both datasets:



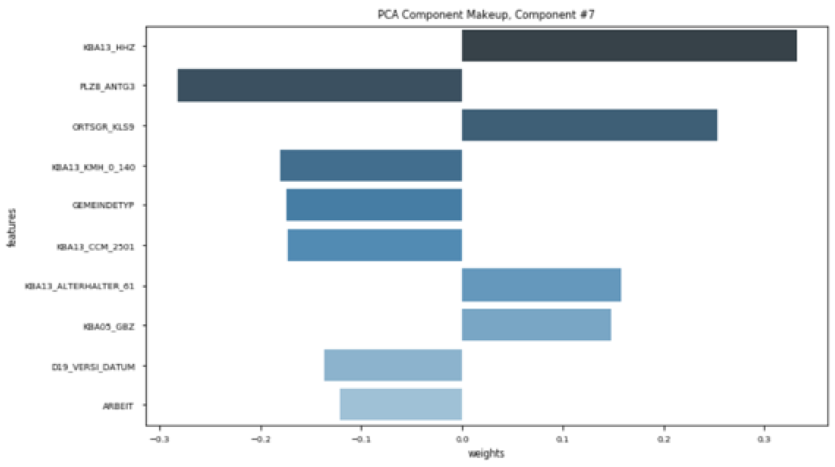
The percentage plot shows that cluster 3 and 4 are significantly overrepresented, while cluster 0 and 1 are underrepresented.

A positive correlation for the components 1 and 2 are the major factor for cluster 3 and 4. The next figure shows the dimensions with the highest correlations with components 1 and 2. The positive correlations lead to the interpretation that the mail-order companies' clients are normally well-off families with multiple persons in their household, a medium to high income. The customer is of the older generation and are normally more traditional/save money.





The mirror in some aspects the opposite of cluster 0 and 1. Looking closer at the additional component 7 for cluster 2, the mail-order company is less represented in areas with significantly more households, which would mean also less income / single house-owner. Moreover, the number of high inhabitants leads to the conclusion that these are more often people from urban areas, with a lower age.



Justification

The first analysis of the segments indicate differences in the customer base of the mail-order company. A further enhancement of the clusters could be used to personalize their marketing activities. Further enhancements, which are listed in the improvement section, are possible.

Classification/Prediction

Data Preprocessing

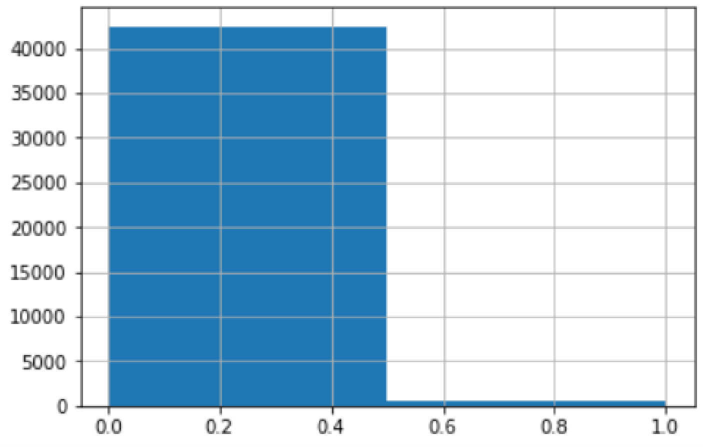
The following steps have been conducted to preprocess the data for the **Classification/Prediction** exercise for both datasets:

1. The first step was to get a general overview of the dataset. We wanted to confirm the size, shape, and types of the mailout_train dataset and detect any main differences to the overall customer and general population dataset.

```
> mailout_train.shape
(42962, 367)

> mailout_train.info()
Description about data types in customer database:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42962 entries, 0 to 42961
Columns: 367 entries, LNR to ALTERSKATEGORIE_GROB
dtypes: float64(267), int64(94), object(6)
memory usage: 120.3+ MB
None
```

Overall the training set seems to have a similar shape and size of the other datasets (except rows). The column which was not in the German general population dataset was the row "RESPONSE". Let's have a look at the respective column. The graphic reveals that the classification will be on an imbalanced data set. Therefore, it focuses on ensemble models and data processing and evaluation techniques that enhance the prediction of rare events (e.g., roc_auc, oversampling, undersampling,...).

```
First impression of data RESPONSE:
0    0
1    0
2    0
3    0
4    0
Name: RESPONSE, dtype: int64
Column data types: int64
Sum empty values (absolute/percentage): 0 / 0.0
Rows values unique identifier (Length dataset vs. unique values): False
0    42430
1      532
Name: RESPONSE, dtype: int64
```



2. Next, similar to the Segmentation, we remove all NaN values for the respective column-types based on the documentation and remove all columns which are more than 25% filled with NaN values. This results in the drop of the following columns:

```
['D19_TELKO_ONLINE_QUOTE_12', 'ALTER_KIND4', 'ALTER_KIND3', 'TITEL_KZ', 'ANZ_TITEL', 'ALTER_KIND2', 'ALTER_KIND1', 'KBA05_ANTG4',
```

3. Moreover, we dropped the columns which do not give a additional information or have a high amount of different values:

```
["EINGEFUEGT_AM", "LNR", "D19_LETZTER_KAUF_BRANCHE"]
```

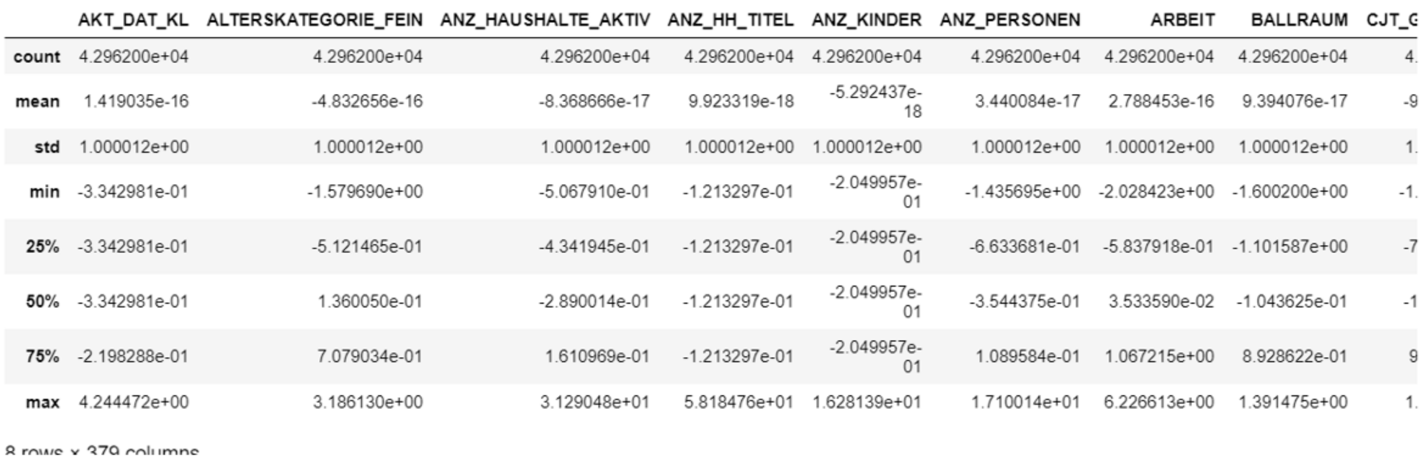
4. The object like columns (['CAMEO_DEU_2015', 'CAMEO_DEUG_2015', 'CAMEO_INTL_2015', 'OST_WEST_KZ']) were first harmonized and then a OneHotEncoder was used to give each date a seperate column. This was done, so a later model would not see the scaled and normalized value as sequence and interprets the value 2 as higher than values 1.

5. Next, we removed all columns with a correlation with one another column higher than 95%. This was done to reduce the noise and simplify the data. No information got lost, as the columns were so similar that the respective correlating columns can carry on the contained information. The correlation matrix revealed a drop of the following columns and a resulted shape of the dataset:

```
['ANZ_STATISTISCHE_HAUSHALTE', 'KBA13_HERST_SONST', 'KBA13_KMH_250', 'LP_FAMILIE_GROB', 'LP_LEBENSPHASE_FEIN', 'LP_LEBENSPHASE_GR

> mailout_train_v6.shape
(42962, 380)
```

6. Before the start of the normalization and scaling, the "RESPONSE" columns were split from the overall database to secure the correctness of it throughout the data normalization process. Then the columns were split into binary categories and numerical categories. The numerical categories were encoded with a KNNImputer, while the Binary categories were encoded with a

SimpleImputer with the strategy "most frequent value". Afterward, the numerical values were normalized with a StndardScaler. The imputer and scaler were saved for the later processing of the mailout_testdata set. This is to secure that both sets undergo the same normalization (e.g., same MinMax ranges).



	AKT_DAT_KL	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN	ARBEIT	BALLRAUM	CJT_G
count	4.296200e+04	4.296200e+04	4.296200e+04	4.296200e+04	4.296200e+04	4.296200e+04	4.296200e+04	4.296200e+04	4.
mean	1.419035e-16	-4.832656e-16	-8.368666e-17	9.923319e-18	-5.292437e-18	3.440084e-17	2.788453e-16	9.394076e-17	-9
std	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.000012e+00	1.
min	-3.342981e-01	-1.579690e+00	-5.067910e-01	-1.213297e-01	-2.049957e-01	-1.435695e+00	-2.028423e+00	-1.600200e+00	-1.
25%	-3.342981e-01	-5.121465e-01	-4.341945e-01	-1.213297e-01	-2.049957e-01	-6.633681e-01	-5.837918e-01	-1.101587e+00	-7
50%	-3.342981e-01	1.360050e-01	-2.890014e-01	-1.213297e-01	-2.049957e-01	-3.544375e-01	3.533590e-02	-1.043625e-01	-1
75%	-2.198288e-01	7.079034e-01	1.610969e-01	-1.213297e-01	-2.049957e-01	1.089584e-01	1.067215e+00	8.928622e-01	9
max	4.244472e+00	3.186130e+00	3.129048e+01	5.818476e+01	1.628139e+01	1.710014e+01	6.226613e+00	1.391475e+00	1.

8 rows × 379 columns

7. The preprocessed dataset was the basis for a PCA and respective reduction. The later training of the models showed early on, that the dataset without PCA performs better than with PCA. Therefore, we left the PCA out of the documentation.

Implementation

The implementation was easy to conduct after the segmentation/clustering analysis. No further points to add.

Refinement

The following refinements could be done in the future to further enhance the data basis for the training of the models:

1. Further feature generation (e.g., use clustering as an additional dimension)
2. Reduce the noise of the dataset and exclude columns with skewed distribution
3. Reduce rows with a high NaN percentage in the training dataset

IV. Results

Model Evaluation and Validation

Both algorithms (GradientBoosting, XGBoost) were tested on the provided dataset. Both algorithms were tested on a dataset with PCA (99% Variance) and one without any decompression. The result showed, that the classifier performed better on the dataset without PCA. For all of the algorithms a cross-validation algorithm was used with the following settings:

```
StratifiedKFold(n_splits=10, shuffle=True, random_state=1)

> XGBoost without PCA - Mean ROC AUC: 0.68153
> GradientBoosting without PCA - Mean ROC AUC: 0.76180

> XGBoost with PCA - Mean ROC AUC: 0.56908
> GradientBoosting with PCA - Mean ROC AUC: 0.58091
```

The first algorithm to optimize is XGBoost classifier. First, we run a fit algorithm with the parameter early_stopping_rounds = 20, to determine the optimal number of boosting rounds. The result indicates the optimum at an early iteration. In this example 3.

The algorithm is trained with the initial parameters:

```
xgb_parameter = {"max_depth" : 3,
                 "subsample": 1,
                 "colsample_bytree" : 0.5,
                 "gamma": 0,
                 "scale_pos_weight": 10,
                 "eval_metric": ['auc'],
                 "objective" : 'binary:logistic'}
```

The cross-validated model with the described set of parameters results in the following scores:

```
Model Report
Accuracy : 0.9876
AUC Score (Train): 0.830278
   train-auc-mean  train-auc-std  test-auc-mean  test-auc-std
0        0.777434       0.001387       0.755182      0.016064
1        0.782917       0.002465       0.750948      0.020557
2        0.791239       0.006767       0.751510      0.027002
3        0.796648       0.004736       0.756218      0.028547
4        0.798753       0.005455       0.750835      0.032309
5        0.807478       0.002928       0.758404      0.024268
6        0.812897       0.002369       0.758292      0.025169
7        0.819175       0.003995       0.758648      0.024433
8        0.822109       0.004266       0.759427      0.021975
9        0.827505       0.002929       0.760905      0.021008
```

The set of parameters are the basis for hyperparameter optimization. First, the parameter max_depth and min_child_weight are optimized. Then subsample and colsample_bytree. The end of the optimization is eta, gamma, and reg_alpha. A description of each of the parameters can be found here.

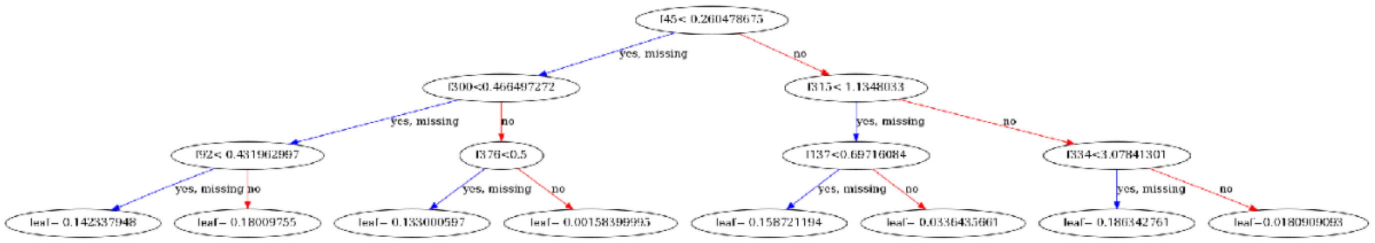
The hyperparameter optimization result in the following dictionary:

```
{'max_depth': 3, 'subsample': 0.6, 'colsample_bytree': 0.7, 'gamma': 0.0, 'scale_pos_weight': 10, 'eval_metric': ['auc'],
'objective': 'binary:logistic', 'min_child_weight': 1, 'eta': 0.1, 'reg_alpha': 0.005}
```

The model achieves the following results with cross-validation, early_stopping and described parameters:

```
Stopping. Best iteration:
[27]    validation_0-auc:0.83979        validation_1-auc:0.76994
```

```
Model Report
Accuracy : 0.9876
AUC Score (Train): 0.817078
```



The above figure shows the depth and structure of the trained XGBoost Classifier.

The optimization of the GradientBoosting Classifier suggests an optimal set of estimator rounds at iteration 23. The model was then optimized with the following set of hyperparameters:

```
parameters = {'learning_rate': [0.1, 0.2], 'n_estimators': [25, 50, 100, 250],
              'max_depth': [3, 5], 'min_samples_split': [2, 4]}
```

The optimized model had the following result and parameters, which were afterward used on the test dataset for the Kaggle competition.

```
Best parameter (CV score=0.762):
{'learning_rate': 0.1, 'max_depth': 3, 'min_samples_split': 2, 'n_estimators': 25}
```

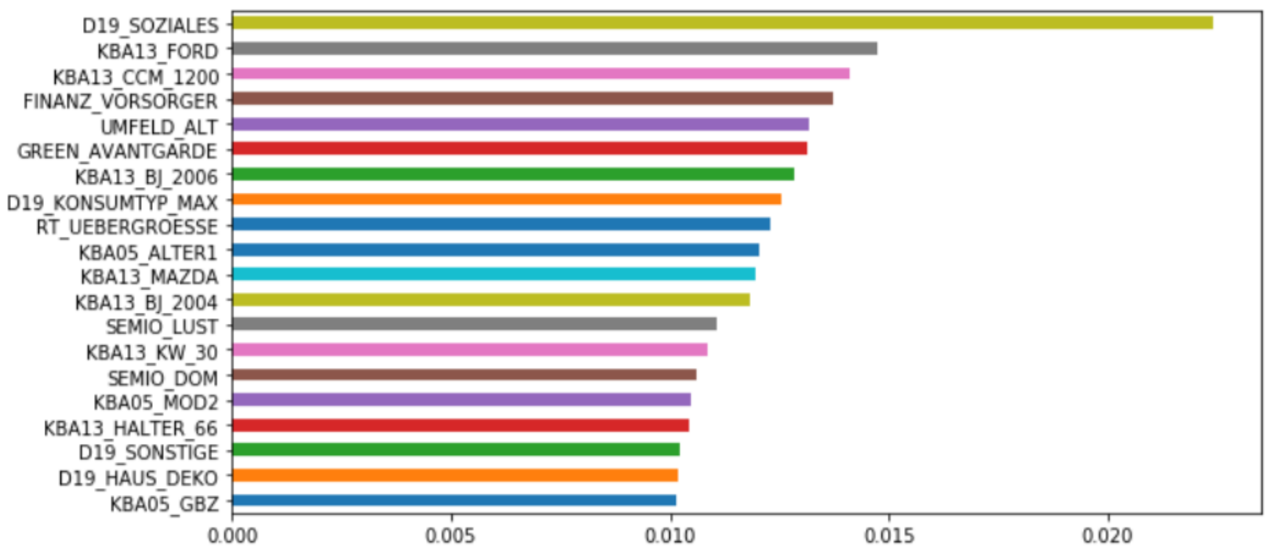
The test set for the Kaggle competition was prepared similar to the train set. The same columns were dropped and the saved weights of the Imputer and Scaler were used to ensure the similarity in the data preprocessing process between train and test set.

The LNR column was used as an identifier and the probability of each person to RESPONSE (=1) was attached as the second column. Both models were submitted with a result of 0.79326 AUC score for the XGBoost algorithm and 0.79672 for the GradientBoosting algorithm, which is slightly lower than the current optimum of 0.81063 AUC score.

V. Conclusion (Population/Customer Segmentation & Classification/Prediction)

Free-Form Visualization

The following figure shows the top 20 leading factors for the trained XGBoost Classifier in ascending order. As shown in the figure, D19_SOZIALES seems to be a major factor to predict the response.



)

Reflection

The exercise included a variety of algorithms, techniques, and optimization challenges for supervised and unsupervised machine learning models. Especially in combination with large datasets and imbalanced data, the exercise was a useful final project to dive deep into the topic and try a bunch of different approaches. Even though the final result of the analysis is compatible and meets the above-described benchmark, the model is improvable and other methods could be used to improve the analysis in the future.

Improvement

There are a variety of improvements one could take for the classifier in the future. One idea would be to use the customer and general population dataset for the classification exercise. Moreover, one could use typical tools to work with imbalanced data such as over- and undersampling. Lastly, the process of feature selection and generation could be enhanced to further reduce the noise in the dataset and stabilize the prediction of probabilities for the test set. This would require further analysis of the feature importance and the setup in each tree-element of the XGBoost classifier.

Sources:

[1] https://www.redpointglobal.com/blog/addressing-the-gaps-in-customer-experience-redpoint-global-harris-poll-benchmark-survey/ [2] https://www.experian.com/assets/marketing-services/white-papers/ccm-email-study-2013.pdf [3] https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html [4] https://scikit-learn.org/stable/modules/clustering.html#k-means [5] https://dl.acm.org/doi/10.5555/2968618.2968676 [6] https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation [7] https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score [8] https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc [9] https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics [10] https://xgboost.readthedocs.io/en/latest/ [11] https://medium.com/@contactsunny/label-encoder-vs-one-hot-encoder-in-machine-learning-3fc273365621 [12] https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html [13] https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68