Machine Learning Engineer Nanodegree

Capstone Report: Targeted Advertising (Arvato)

Shuo Cheng Jan 21th, 2020

Overview

Domain Background

Arvato Bertelsmann is an International media company. One of the firm's key focuses is to provide market analysis for its clients to target and acquire new customers more accurately and efficiently. [1] Levering creative data science providing robust conviction and timing, a company could find the right potential target customers in the most cost/time efficient manner. The research and models applied in the study would include Exploratory Data Analysis[2], unsupervised learning (PCA & K-means) and supervised learning (Logistic Regression, Ada Boost and Gradient Boosting Classifiers).

Project Origin & Problem Statement[3]

This project is to aid a mail-order sales company to acquire new German clients for their mail-out campaign based on analyzing and comparing the attributes of general population and potential target Germen population. The end goal is to identify and predict a group of target audience of the campaign that could bring the highest return for the company. This study would point out a general direction for the company to move forward with higher return on investment.

Datasets and Inputs[4]

There are four data files associated with this project:

- **Udacity_AZDIAS_052018.csv:** Demographics data for the general population of Germany; 891 211 persons (rows) x 366 features (columns).
- **Udacity_CUSTOMERS_052018.csv:** Demographics data for customers of a mail-order company; 191 652 persons (rows) x 369 features (columns).
- **Udacity_MAILOUT_052018_TRAIN.csv:** Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns).
- **Udacity_MAILOUT_052018_TEST.csv:** Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

Each row of the demographics files represents a single person, but also includes information outside of individuals, including information about their household, building, and neighborhood.

The "CUSTOMERS" file contains three extra columns ('CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP'), which provide broad information about the customers depicted in the file. The original "MAILOUT" file included one additional column, "RESPONSE", which indicated whether each recipient became a customer of the company.

Solution Statement

We will utilize the information from the first two files to figure out how customers ("CUSTOMERS") are similar to or differ from the general population at large ("AZDIAS"), then use your analysis to make predictions on the other two files ("MAILOUT"), predicting which recipients are most likely to become a customer for the mail-order company.

Benchmark Models

The benchmarks for this project would be Logistic Regression model and Ada Boost Classifier. We will work all the models on the same dataset and record the ROC_AUC scores as well as time taken to find the most efficient candidate. We then compare the benchmark models with default settings, then pick the ones with higher ROC_AUC, fine-tuned models and conclude the final model used for the test. Beating the benchmark models could imply that the project has adequately solve the problem.

Evaluation Metrics [5,6]

For the classification problem, the evaluation metric is AUC - Area under the ROC (Receiver Operating Characteristic) curve. The axes of plane divides data into 4 segments:

- FP: False Positive; The population mistakenly grouped as potential customers
- TP: True Positive; The target customers predicted that are actual customers
- FN: False Negative; The real customers missed by the model prediction
- TN: True Negative; The population that is not potential clients

ROC is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The curve gradually increases the threshold of Positive Rate, from 0 to ALL. The AUC represents validity of the model, ranging from 0.5 to 1.

- **Metrics Justifications:** Because we are predicting which recipients are most likely to become a customer for the mail-order company, the result is a binary answer. The AUC - ROC model is ideal for this type of problems.

Analysis

Data Exploration & Exploratory Visualization

- Data loading and content exploration

In the Datasets and Inputs section, we've covered that the demographics data for the general population of Germany contains 891,211 persons and 366 features while the demographics data for customers of a mail-order company contains 191,652 persons and 369 features.

The column headers of the data files are not self-explanatory:

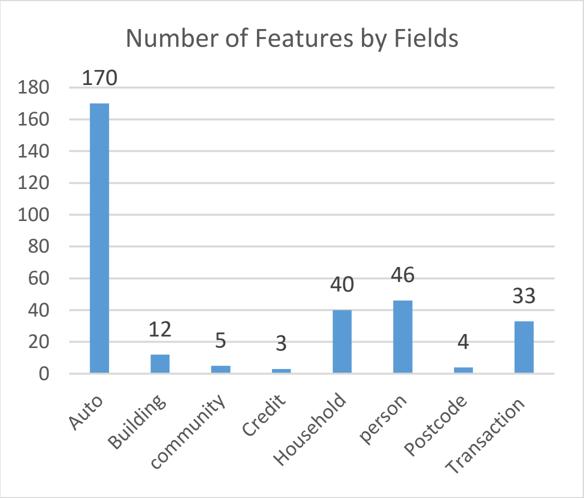
In order to learn more about the columns depicted in the files, we refer to two Excel spreadsheets provided in the workspace.

- **DIAS Information Levels - Attributes 2017.xlsx:** a top-level list of attributes and descriptions, organized by informational category.
- **DIAS Attributes - Values 2017.xlsx:** a detailed mapping of data values for each feature in alphabetical order.

The detailed features cover a wide range of parameters such as:

- Household (ex. members in a household)
- Building (ex. number of holders of an academic title in the building)
- Auto (ex. share of 5 seaters)
- Transaction activity (ex. transactional activity based on BOOKS and CDs)
- Personality (ex. affinity indicating in what way the person is cultural minded)

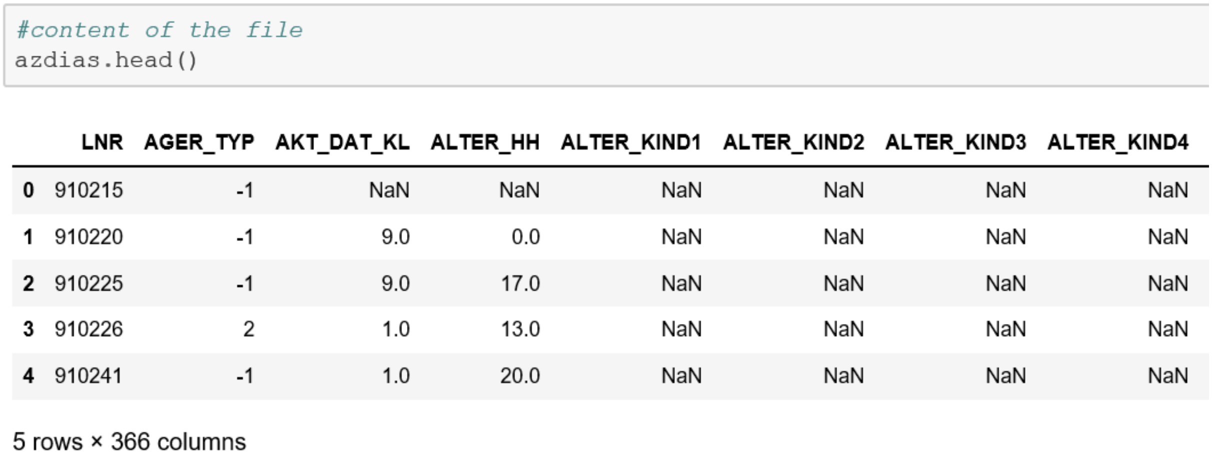
And so on…



Each of the genre has the same Prefix. For instance, transaction activities mostly start with D19; car shares mostly start with KBA. So, a lot of the features are redundant. We can use PCA to reduce the repetitiveness by minimizing features with high absolute value of correlation. (close to 1)

- Data value analysis

The data sheet contains a lot of missing values (represented by NaN in the table), and some of the values invalid such as (-1, 0, -9) also representing the missing information.



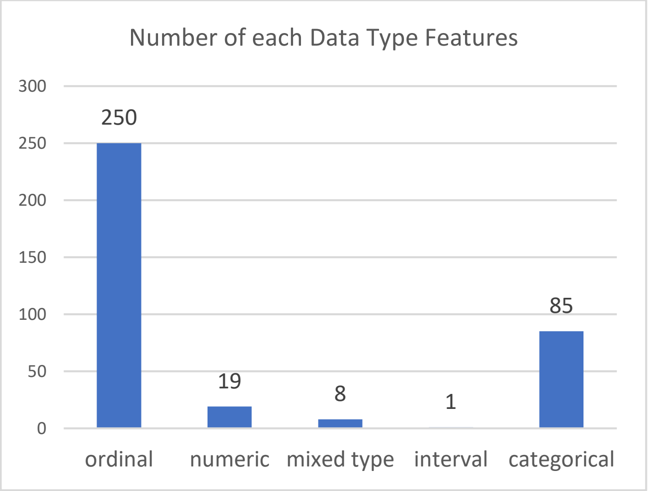
```
#content of the file
azdias.head()
```

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4
0	910215	-1	NaN	NaN	NaN	NaN	NaN	NaN
1	910220	-1	9.0	0.0	NaN	NaN	NaN	NaN
2	910225	-1	9.0	17.0	NaN	NaN	NaN	NaN
3	910226	2	1.0	13.0	NaN	NaN	NaN	NaN
4	910241	-1	1.0	20.0	NaN	NaN	NaN	NaN

5 rows × 366 columns

- Data type/category analysis
  a. For numeric and interval data, these features can be kept without changes.
  b. Special handing may be necessary for the remaining varies such as categorical and 'mixed'

We list out below all the different types of values that the datasets own:



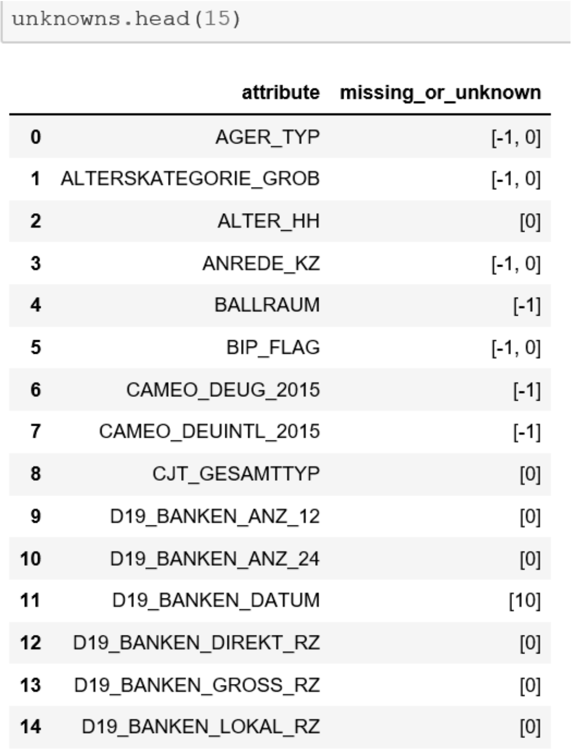
- Ordinal and Interval Features
  - Nothing special
    - need to decide what to do with the missing values and scaling
- Numerical Features
  - Nothing special
    - need to decide what to do with the missing values and scaling.
- Mixed-Type Features
  - There are a handful of features that are marked as "mixed" in the feature summary that require special treatment before we can include then in the analysis. There are two that deserve attention:
    - PRAEGENDE_JUGENDJAHRE combines information on three dimensions: generation by decade, movement (mainstream vs. avantgarde), and

nation (east vs. west). While there aren't enough levels to disentangle east from west

- **CAMEO_INTL_2015** combines information on two axes: wealth and life stage.

- Categorical Features
  - For categorical features, we encode the levels as dummy variables. Depending on the number of categories, we can perform one of the following:
    - Binary variables with numeric values
      - keep them without needing to do anything.
    - Binary variables that take on non-numeric values
      - re-encode the values as numbers or create a dummy variable.
    - For multi-level categorical features (three or more values)
      - encode the values using multiple dummy variables

- Data Missing Values Analysis



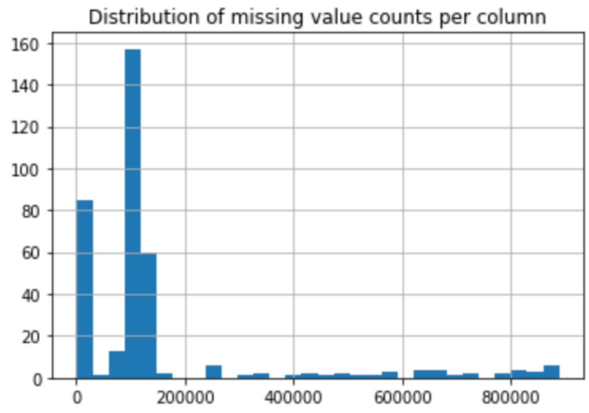
```
unknowns.head(15)
```

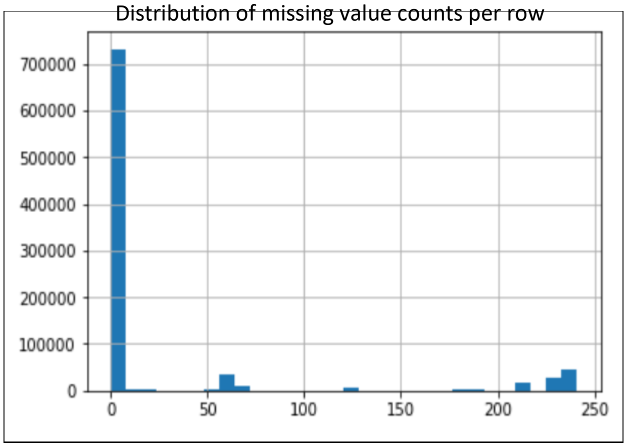
	attribute	missing_or_unknown
0	AGER_TYP	[-1, 0]
1	ALTERSKATEGORIE_GROB	[-1, 0]
2	ALTER_HH	[0]
3	ANREDE_KZ	[-1, 0]
4	BALLRAUM	[-1]
5	BIP_FLAG	[-1, 0]
6	CAMEO_DEUG_2015	[-1]
7	CAMEO_DEUINTL_2015	[-1]
8	CJT_GESAMTTYP	[0]
9	D19_BANKEN_ANZ_12	[0]
10	D19_BANKEN_ANZ_24	[0]
11	D19_BANKEN_DATUM	[10]
12	D19_BANKEN_DIREKT_RZ	[0]
13	D19_BANKEN_GROSS_RZ	[0]
14	D19_BANKEN_LOKAL_RZ	[0]

Based on the DIAS Attributes – Values 2017 file, we've built a summary version of valid properties and their corresponding unknown data to help us make data cleaning decisions.

The third column **missing_or_unknown** of the feature attributes summary, documents the codes from the data dictionary that indicate missing or unknown data. We convert data that matches a 'missing' or 'unknown' value code into a NaN values. The figure below shows how much data is NaN.

We have distribution of missing value counts where we can see that there are a few columns that are outliers with large number of missing values. We see for rows, (right figure) there are also some groups of points that have very different number of missing values.





Algorithms and Techniques

There are 2 parts of this project:

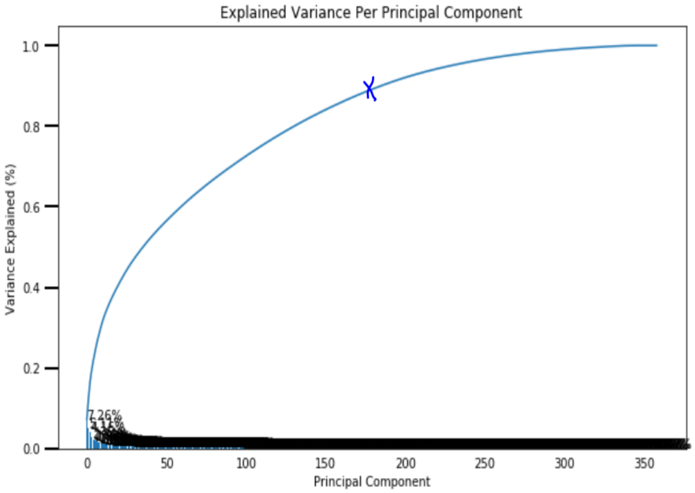
1. Unsupervised Learning:

The main bulk of our analysis is in this part of the project. Here, we use unsupervised learning techniques to describe the relationship between the demographics of the company's existing customers and the general population of Germany. By the end of this part, we should be able to describe parts of the general population that are more likely to be part of the mail-order company's primary customer base, and which parts of the general population are less so.

Customer Segmentation Report - Unsupervised Learning Model [2]

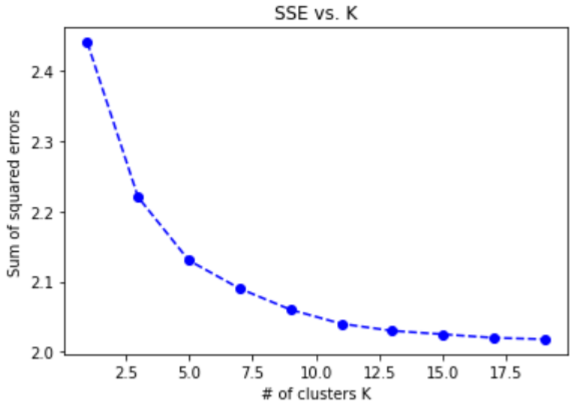
We begin the project by using unsupervised learning methods to analyze attributes of established customers and the general population in order to create customer segments.

i. Dimensionality reduction with PCA



PCA attempts to reduce the number of features within a dataset while retaining the "principal components", which are defined as weighted, linear combinations of existing features that are designed to be linearly independent and account for the largest possible variability in the data. We decide on the number of components that explain at least 90% of the variance in data, which is 180.

ii. **Use k-means, to segment customers using their PCA attributes for Clustering**
  - We create **KMeans** models with clusters from 2 to 15.
  - We use clustering where each data point is grouped into a subset or a cluster which contains similar kind of data points.
    i. We decide on the optimal number of clusters based on **the Elbow Plot** - 8 Clusters.



8 Seems like a kinky point so we use 8 clusters

iii. We then exploring the resultant clusters & visualization for research purpose

2. **Supervised Learning**

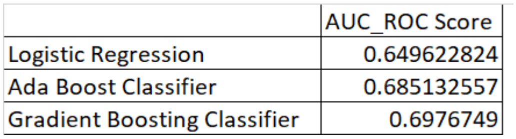
To predict the probability of a person to reply to the mailing campaign, we create a Gradient Boosting Classifiers which we use to predict this probability.

Before training the model, we decide on a resampling technique for the data, and after we start by searching the best hyperparameters for models using all available features by using a Bayesian search.

Once we have a list of optimized hyperparameters, we use them for training a model on the resampled data. After training the model is used to predict on the TEST dataset.

- **Benchmark**

Model evaluation is the process of objectively measuring how well machine learning models perform the specific tasks they were designed to do.



	AUC_ROC Score
Logistic Regression	0.649622824
Ada Boost Classifier	0.685132557
Gradient Boosting Classifier	0.6976749

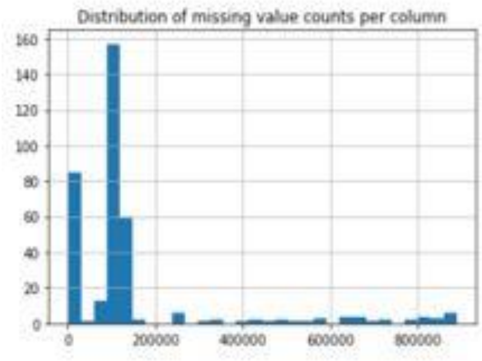
We use the AUC scores to benchmark the performance of the models. A model with the highest AUC is considered as the best performer.

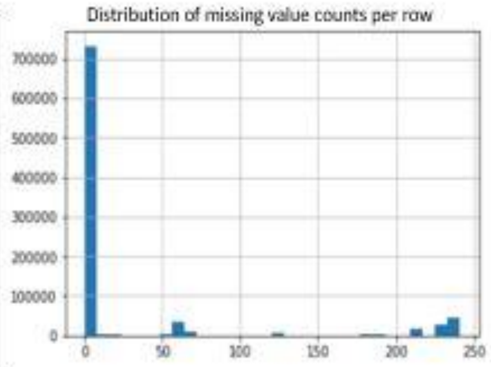
i. Train a **Logistic Regression model, Ada Boost Classifier as well as Gradient Boosting Classifier** with default parameters and use it to predict the probabilities for the train and test datasets.
ii. The obtained AUC score is our base score used for comparison.

Methodology

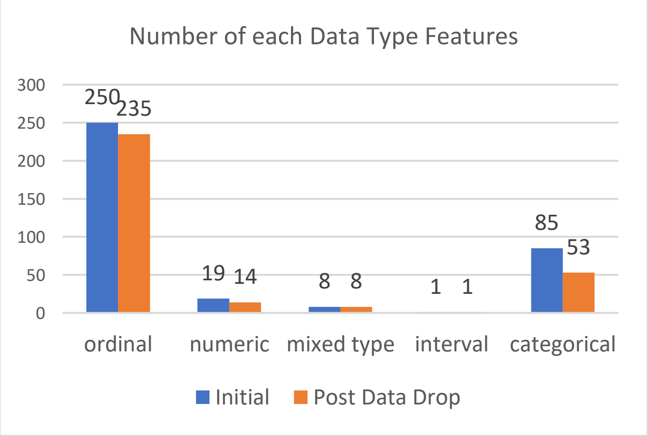
Data Pre-processing

- Data cleaning and pre-processing
    a. EDA -exploratory data analysis
        i. Encode missing values as NaNs
        ii. Based of the distribution below, drop any incomplete rows of data
            1. Remove columns that have > 200 000 missing values
            2. Remove rows that have >10 missing values
            3. If a categorical variable has more than 10 levels, drop for simplicity
            4. Re-encode categorical variables by creating dummy variables





    b. Create a new Data Frame, and drop unnecessary columns
    c. Re-code the numbers for text descriptions so the model can read them
    d. Impute the missing values using the mean
    e. Normalize the data using **Standard Scaler** to transform the numerical values so that they all fall between 0 & 1
    f. Figure illustrate the number of features dropped



- Generate a cleaning function
    a. performs all the above steps so that it can be run on the customers dataset as well

Implementation

Unsupervised Learning Model

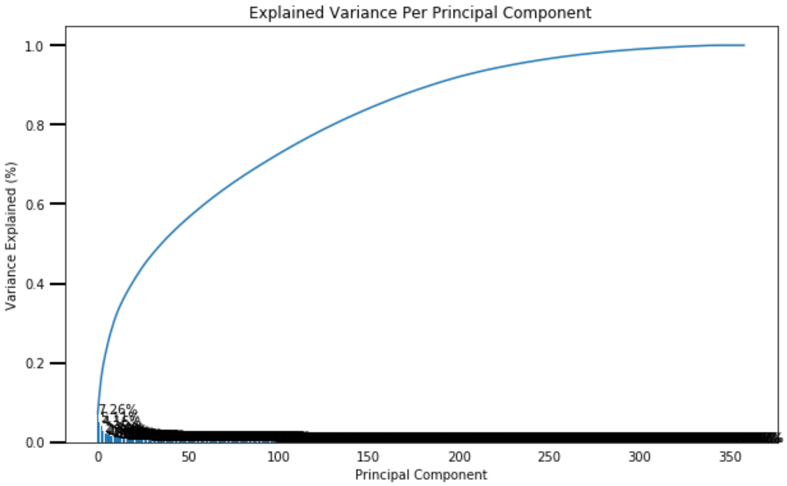
- Perform Dimensionality Reduction (PCA)

On our preprocessed data, we are now ready to apply dimensionality reduction techniques.

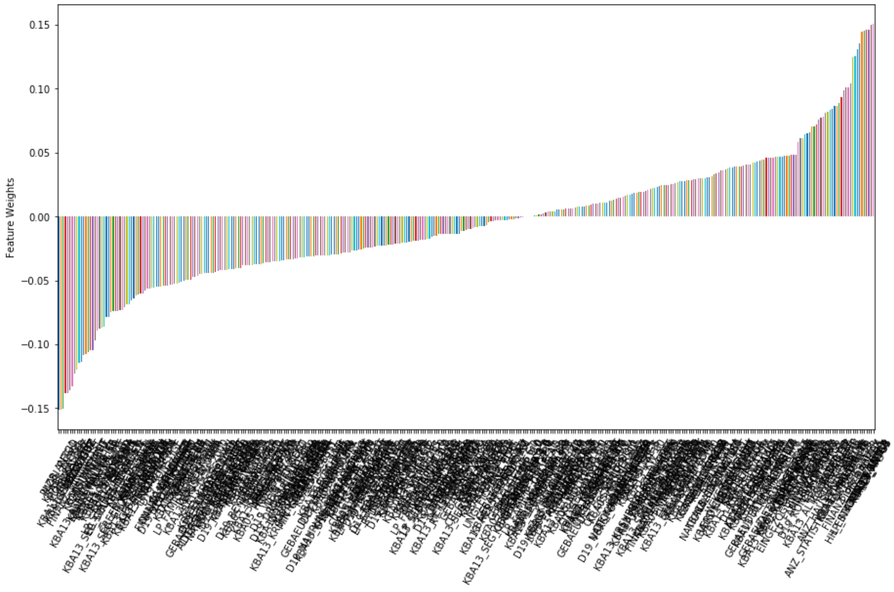
We use Sklearn's PCA class to apply principal component analysis on the data, thus finding the vectors of maximal variance in the data.

We start by fitting a PCA on 685 dimensions (our initial dataset has 366 dimensions but increases to 685 after one-hot encoding the categorical features). You can find the results for the PCA in figure.

We check out the ratio of variance explained by each principal component as well as the cumulative variance explained. Based on the results from the PCA fitted previously, we decide to keep the first 180 reduced dimensions, that explain 90% cumulative variance in data.

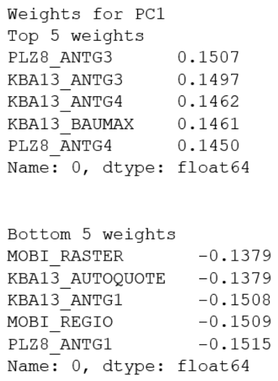


Now that we have our transformed principal components, we check out the weight of each variable on the first few components to see if we can interpret them some fashion.

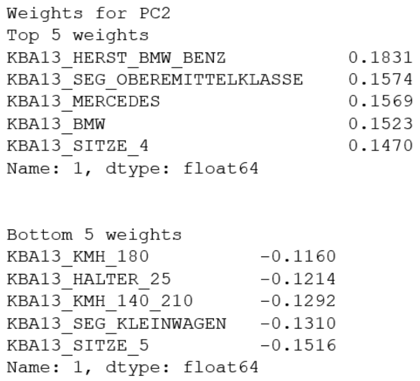


Each principal component is a unit vector that points in the direction of highest variance (after accounting for the variance captured by earlier principal components). The further a weight is from zero, the more the principal component is in the direction of the corresponding feature. If two features have large weights of the same sign (both positive or both negative), then increases in one tend to expect to be associated with increases in the other. To contrast, features with different signs can be expected to show a negative correlation: increases in one variable should result in a decrease in the other.

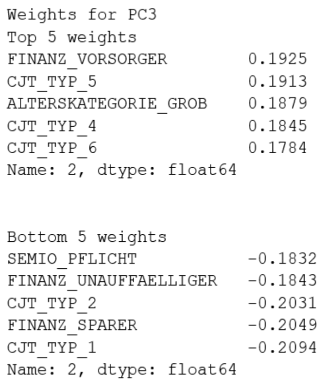
To investigate the features, we map each weight to their corresponding feature name, then sort the features according to weight. The most interesting features for each principal component, are those at the beginning and end of the sorted list.



```
Weights for PC1
Top 5 weights
PLZ8_ANTG3      0.1507
KBA13_ANTG3     0.1497
KBA13_ANTG4     0.1462
KBA13_BAUMAX    0.1461
PLZ8_ANTG4      0.1450
Name: 0, dtype: float64


Bottom 5 weights
MOBI_RASTER        -0.1379
KBA13_AUTOQUOTE    -0.1379
KBA13_ANTG1        -0.1508
MOBI_REGIO         -0.1509
PLZ8_ANTG1         -0.1515
Name: 0, dtype: float64
```

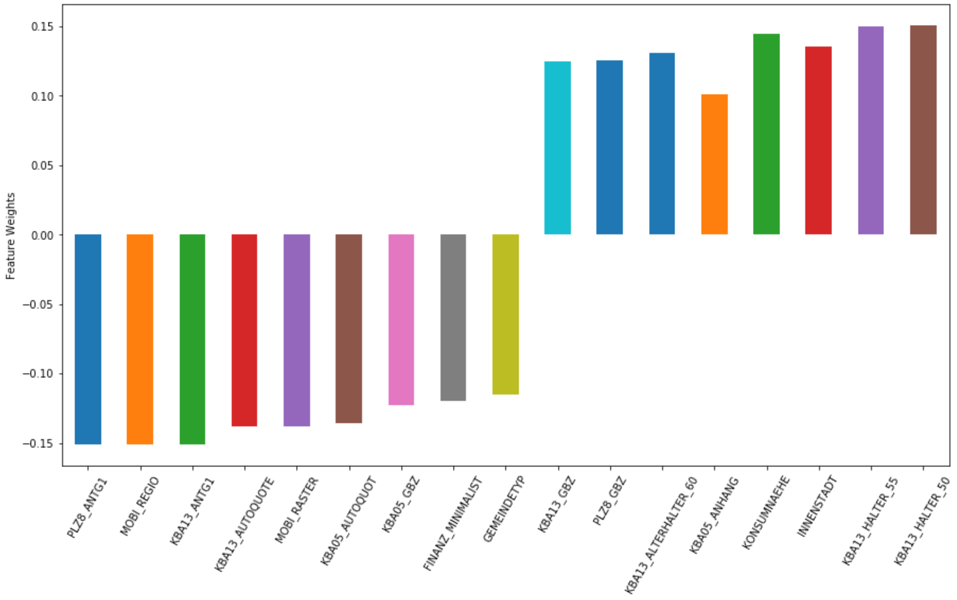


```
Weights for PC2
Top 5 weights
KBA13_HERST_BMW_BENZ            0.1831
KBA13_SEG_OBEREMITTELKLASSE     0.1574
KBA13_MERCEDES                  0.1569
KBA13_BMW                       0.1523
KBA13_SITZE_4                   0.1470
Name: 1, dtype: float64


Bottom 5 weights
KBA13_KMH_180           -0.1160
KBA13_HALTER_25         -0.1214
KBA13_KMH_140_210       -0.1292
KBA13_SEG_KLEINWAGEN    -0.1310
KBA13_SITZE_5           -0.1516
Name: 1, dtype: float64
```



```
Weights for PC3
Top 5 weights
FINANZ_VORSORGER        0.1925
CJT_TYP_5               0.1913
ALTERSKATEGORIE_GROB    0.1879
CJT_TYP_4               0.1845
CJT_TYP_6               0.1784
Name: 2, dtype: float64


Bottom 5 weights
SEMIO_PFLICHT            -0.1832
FINANZ_UNAUFFAELLIGER    -0.1843
CJT_TYP_2                -0.2031
FINANZ_SPARER            -0.2049
CJT_TYP_1                -0.2094
Name: 2, dtype: float64
```

We present here the investigation of feature associations from the first five principal components:



Example explained:

Weights for PC1

Top 5 weights: **PLZ8_ANTG3,** number of 6-10 family houses in the PLZ8, explaining most of the component. **KBA13_ANTG3,** number of cars in PLZ8, which is logically directly related to the number of households; the more houses, the more cars. **KBA13_ANTG4;** number of family in households and **PLZ8_ANTG4,** number of family in households in PLZ8, are roughly the same factor with different scope, the more family the more houses. And **KBA13_BAUMAX,** most common building-type within the PLZ8; more houses cover more common building types. Thus, the features are all plausibly positively correlated.

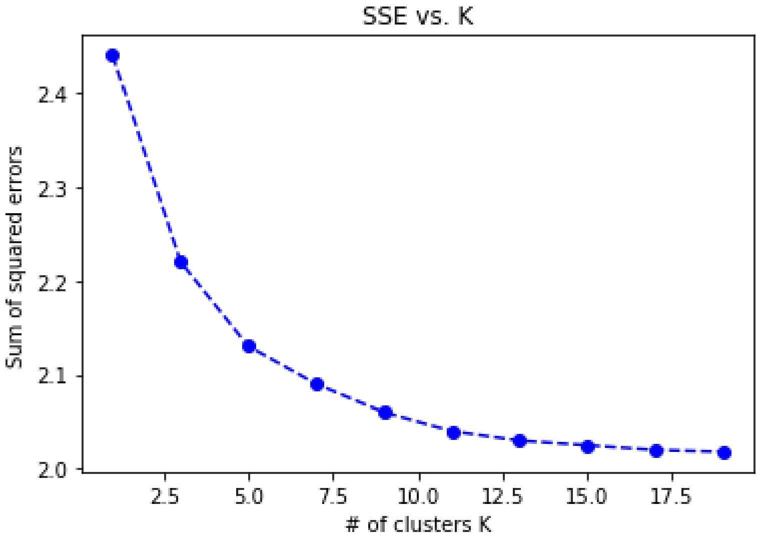
If we look at the Bottom 5 weights: the most negatively correlation is **PLZ8_ANTG1,** number of 1-2 family houses in PLZ8, makes sense because this is directly opposite the large number of households; same applies to **KBA13_ANTG1,** number of 1-2 family houses in the cell. They basically mean the same thing. **MOBI_RASTER,** industrialized areas are not suited for residential houses with large family, thus negatively related to big households. **KBA13_AUTOQUOTE,** share of cars per household within the PLZ8, because the number of households are divided by large number of households, because the large number of the top weights are now denominators, this feature is at the bottom, which makes sense. **MOBI_REGRIO**, high mobility with smaller households.

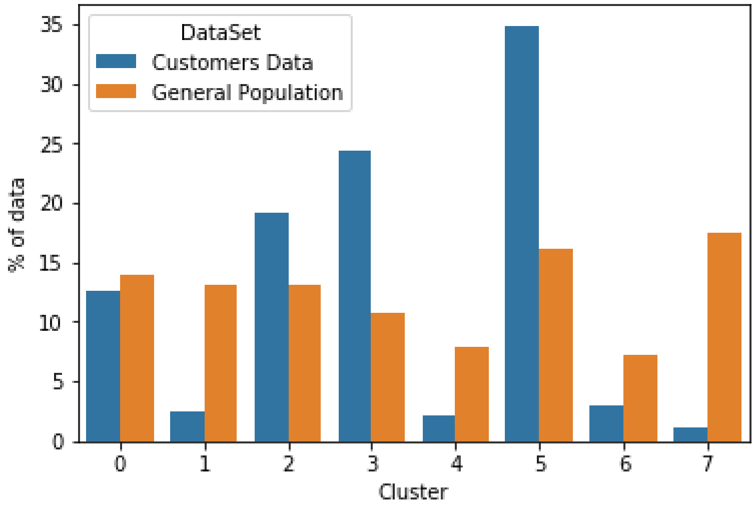
- Clustering using K-Means method

We apply k-means clustering to the dataset and use the average within-cluster distance to decide the number of clusters to keep. We use sklearn's **K-Means** class to perform k-means clustering on the **PCA-transformed data**.

We fit a **K-Means** model on the 180 reduced dimensions and observe the change within-cluster distance across for a range of clusters between 1 and 20.

Based on Figure, we can see that a good number of clusters is 8. We refit the k-means model with the selected number of clusters and obtain cluster predictions for the general population demographics data.





We compare the ratio of data in each cluster for the customer data to the proportion of data in each cluster for the general population:

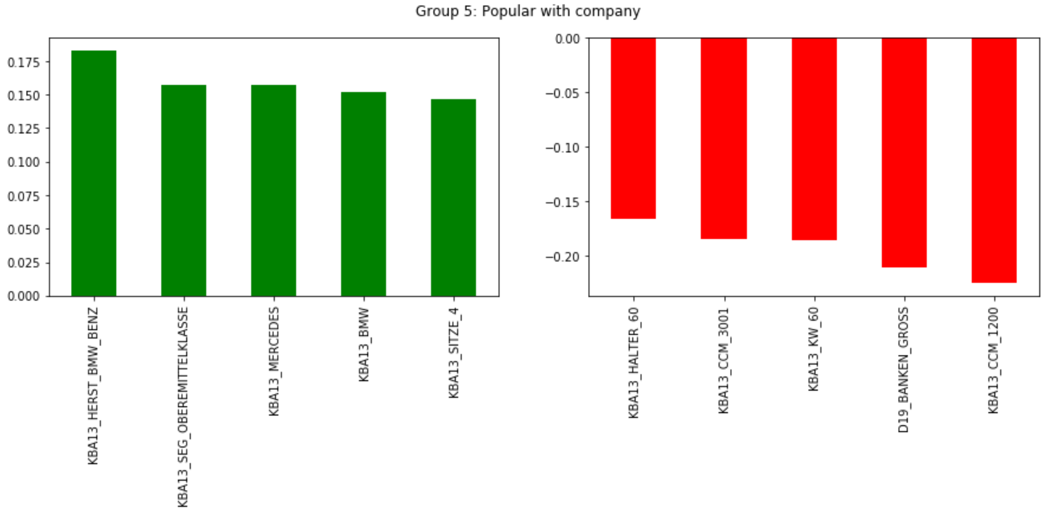
We inspect what kind of people are part of a cluster that is over-represented in the customer data compared to the general population (cluster 5)

We inspect what kind of people are part of a cluster with low representation in the customer data compared to the general population (cluster 7)

Compare Customer Data to Demographics Data

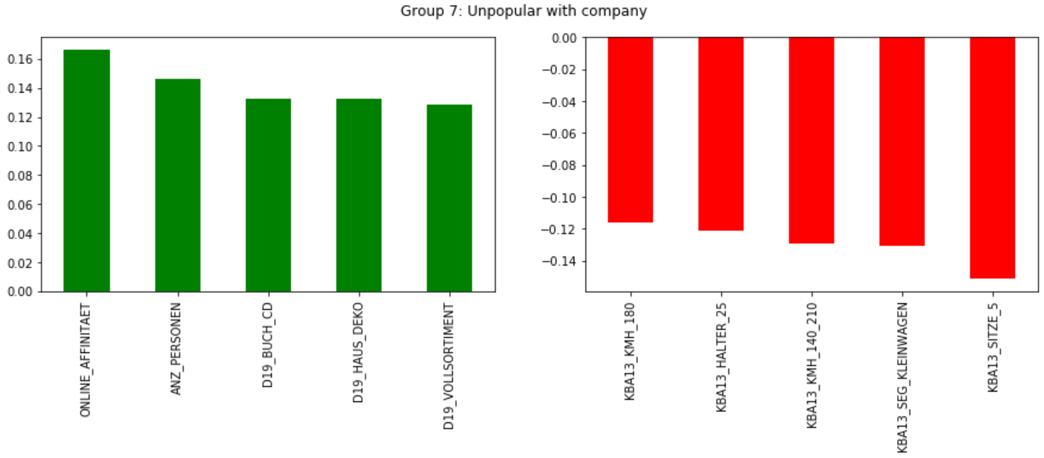
Popular with company - Cluster 5

- 'KBA13_HERST_BMW_BENZ', share of BMW & Mercedes Benz within the PLZ8
- 'KBA13_SEG_OBEREMITTELKLASSE', share of upper middle class cars and upper class cars (BMW5er, BMW7er etc.)
- 'KBA13_MERCEDES', share of MERCEDES within the PLZ8
- 'KBA13_BMW', share of BMW within the PLZ8
- 'KBA13_SITZE_4', number of cars with less than 5 seats in the PLZ8


Group 5: Popular with company

Unpopular with company - Cluster 7

- 'ONLINE_AFFINITAET', online affinity
- 'ANZ_PERSONEN', number of adult persons in the household
- 'D19_BUCH_CD', transactional activity based on the product group BOOKS and CDS
- 'D19_HAUS_DEKO', transactional activity based on the product group HOUSE DECORATION
- 'D19_VOLLSORTIMENT', transactional activity based on the product group COMPLETE MAIL-ORDER OFFERS


Group 7: Unpopular with company

Supervised Learning Model

- Benchmarking

We now build a customer conversion prediction model using the demographic information from each individual to decide if it will be worth it to target that person in the campaign.

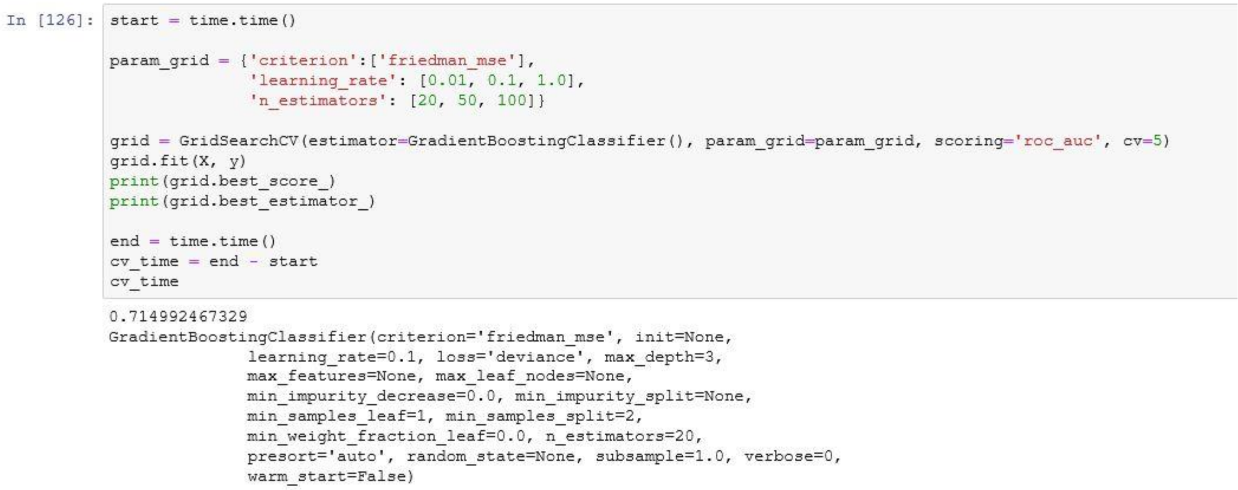
The training dataset includes 42,982 rows with 367 features with a binary "response" row of either 0's or 1's depending on the person's tendency of becoming a customer and a test dataset with 42,833 people. Because the dataset is quite imbalanced (only about 1% conversion), ROC AUC was used as a metric for choosing the best model, and for scoring the Kaggle competition.

After pre-processing the datasets, we tested 3 base machine learning models:

- As shown in the table below, the GradientBoostingClassifier had the highest ROC AUC of 0.70, followed closely by the AdaBoostClassifier with an ROC AUC score of 0.69.
- I chose the default classifier models as the benchmarks and fine-tuned the AdaBoostClassifier and the GradientBoostingClassifier because they were slightly better.
- The AdaBoostClassifier was 3 times faster, but GradientBoostingClassifier is still doable.
- The ROC AUC score increased from 0.68/0.69 to 0.71 and 0.715 for the fine-tuned models.

	AUC_ROC Score
Logistic Regression	0.649622824
Ada Boost Classifier	0.685132557
Gradient Boosting Classifier	0.6976749
Tuned ADA	0.709199188
Tuned GradientBoosting	0.714992467

The methodology would be choosing the final model to test the results on Kaggle. I first chose **the fine-tuned GradientBoostingClassifier** because it has the highest AUC_ROC Score for the training data. But then the results says otherwise.



```
In [126]: start = time.time()

          param_grid = {'criterion':['friedman_mse'],
                        'learning_rate': [0.01, 0.1, 1.0],
                        'n_estimators': [20, 50, 100]}

          grid = GridSearchCV(estimator=GradientBoostingClassifier(), param_grid=param_grid, scoring='roc_auc', cv=5)
          grid.fit(X, y)
          print(grid.best_score_)
          print(grid.best_estimator_)

          end = time.time()
          cv_time = end - start
          cv_time
```

```
0.714992467329
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=20,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm_start=False)
```

Results

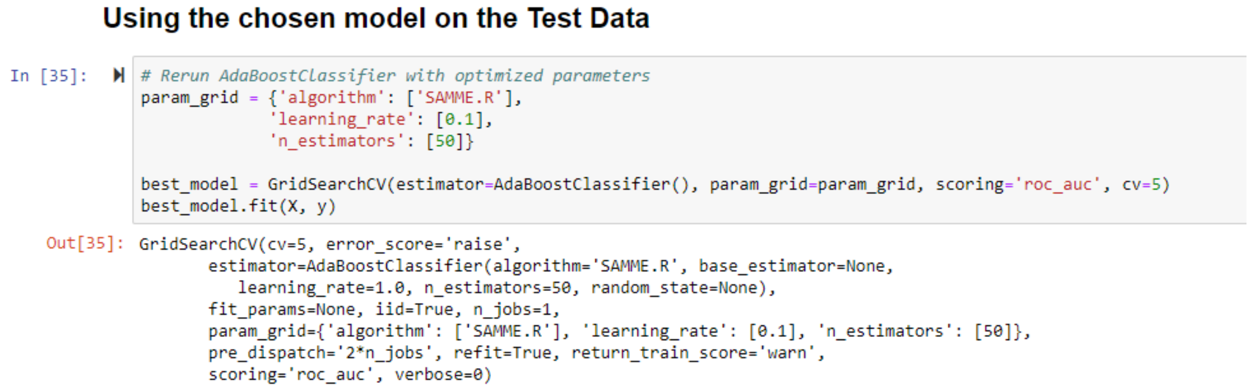
Model Evaluation and Validation

We evaluate the models through

- Benchmarks
- Grid Search function to tune the hyperparameters

We chose the best performing Gradient Boosting model with learning rate 0.1 and n-estimators 20. As shown in the figure above, the AUC ROC score increased from the 0.697 in the table to 0.715. However, after running the test data through the optimized model and submitting the results on Kaggle, we can see that the model performed worse than the training dataset, with a ROC AUC of **0.59**. Thus, the previously chosen fine-tuned GradientBoostingClassifier might not be optimal for this project.

To solve this problem, I built a notebook dedicated to improving the model. I switched back to the fine-tuned AdaBoostClassifier with the higher AUC_ROC Score 0.7091. Similarly, this best model has the learning_rate turned to 0.1 and n_estimators at 50. Validated by the grid search method.

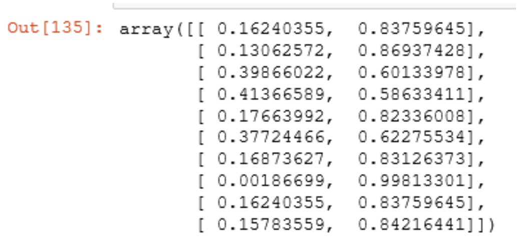


Using the chosen model on the Test Data

```
In [35]:   # Rerun AdaBoostClassifier with optimized parameters
           param_grid = {'algorithm': ['SAMME.R'],
                         'learning_rate': [0.1],
                         'n_estimators': [50]}

           best_model = GridSearchCV(estimator=AdaBoostClassifier(), param_grid=param_grid, scoring='roc_auc', cv=5)
           best_model.fit(X, y)

Out[35]: GridSearchCV(cv=5, error_score='raise',
                estimator=AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
                   learning_rate=1.0, n_estimators=50, random_state=None),
                fit_params=None, iid=True, n_jobs=1,
                param_grid={'algorithm': ['SAMME.R'], 'learning_rate': [0.1], 'n_estimators': [50]},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='roc_auc', verbose=0)
```

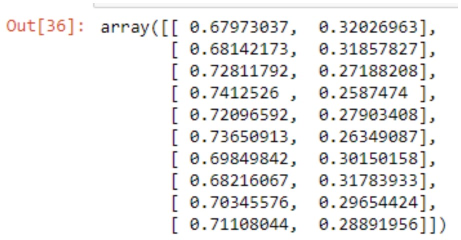
Justification

Below you can see the difference in the Kaggle.csv submission. The Figure on the right is better.



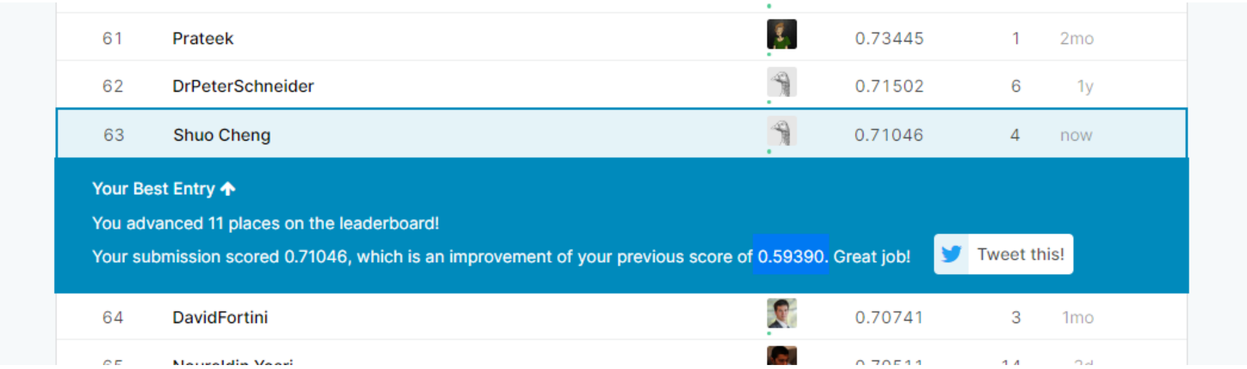
```
Out[135]: array([[ 0.16240355,  0.83759645],
                 [ 0.13062572,  0.86937428],
                 [ 0.39866022,  0.60133978],
                 [ 0.41366589,  0.58633411],
                 [ 0.17663992,  0.82336008],
                 [ 0.37724466,  0.62275534],
                 [ 0.16873627,  0.83126373],
                 [ 0.00186699,  0.99813301],
                 [ 0.16240355,  0.83759645],
                 [ 0.15783559,  0.84216441]])
```

Submitting to the Kaggle Competition



```
Out[36]: array([[ 0.67973037,  0.32026963],
                [ 0.68142173,  0.31857827],
                [ 0.72811792,  0.27188208],
                [ 0.7412526 ,  0.2587474 ],
                [ 0.72096592,  0.27903408],
                [ 0.73650913,  0.26349087],
                [ 0.69849842,  0.30150158],
                [ 0.68216067,  0.31783933],
                [ 0.70345576,  0.29654424],
                [ 0.71108044,  0.28891956]])
```

Submitting to the Kaggle Competition



61	Prateek	0.73445	1	2mo
62	DrPeterSchneider	0.71502	6	1y
63	Shuo Cheng	0.71046	4	now

Your Best Entry 🡅

You advanced 11 places on the leaderboard!

Your submission scored 0.71046, which is an improvement of your previous score of 0.59390. Great job!

64	DavidFortini	0.70741	3	1mo
65	Nouraldin Yaari	0.70511	14	2d

I decided to stop here because the fine-tuned AdaBoostClassifier increased my ROC AUC score by more than 0.10, beating the benchmarks, thus justified. Many people above me were able to reach a score of 0.8, therefore there is room for improvement which is discussed below.

Conclusion

Reflection

There are many areas for improvement:

- In the data of the popular and unpopular, there were still a lot of overlapping features clearing with high correlations. I could dive deeper in cleaning up the features so that the number of highly-related features would be further deducted
- I dropped the mixed data types for the ease of analysis
- For the supervised machine learning part, I could've used more models and more sophisticated models. Because this is my first machine learning application, I'm not particularly adept in some more complex models to compare.
    - Should Leveraging more of what we learnt from the Unsupervised learning and apply to the supervised model
- Could do more in adjusting the weights of the imbalanced data to resample the data.

For a real-life marketing campaign, it is essential to correctly identify the customers who will respond to a campaign.

In this project, we explored the demographic data for customers of a mail-order sales company in Germany, compared it against the general population. Exploratory Data Analysis was performed to understand and preprocess the data. Unsupervised learning methods such as

Principal Component Analysist with Clustering using K-means were used to perform customer segmentation, identifying core customer base of the company. Then, supervised models such as logistic regression, Ada boost classifier as well as the Gradient boosting classifier on a third dataset with demographic information for targets of a marketing campaign for the company and use a model to predict which individuals are most likely to convert into becoming customers for the company.

This project was an excellent opportunity to apply and learn new techniques primarily related to imbalanced data problems.

References

1. Bertelsmann, Arvato. *About.*2020. URL:
2. Tukey, John W. (1977). Exploratory Data Analysis. Pearson. ISBN 978-0201076165.
3. Udacity, Bertelsmann/Arvato Project Overview. 2019. URL: https://classroom.udacity.com/nanodegrees/nd009t/parts/2f120d8a-e90a-4bc0-9f4e-43c71c504879/modules/2c37ba18-d9dc-4a94-abb9-066216ccace1/lessons/4f0118c0-20fc-482a-81d6-b27507355985/concepts/8400bad9-69b4-4455-826c-177d90752f00
4. Udacity, Bertelsmann/Arvato Project Workspace. 2019. URL: https://classroom.udacity.com/nanodegrees/nd009t/parts/2f120d8a-e90a-4bc0-9f4e-43c71c504879/modules/2c37ba18-d9dc-4a94-abb9-066216ccace1/lessons/4f0118c0-20fc-482a-81d6-b27507355985/concepts/e9553619-113b-4565-a34e-a9ef450659de
5. China Science Communication, 2019. URL: https://baike.baidu.com/item/AUC/19282953?fr=aladdin#3
6. Fawcett, Tom (2006). "An Introduction to ROC Analysis" (PDF). Pattern Recognition Letters. 27 (8): 861–874. doi:10.1016/j.patrec.2005.10.010.
7. Udacity, Solution: Simple Neural Network, 2019. URL: https://classroom.udacity.com/nanodegrees/nd009t/parts/670d5990-4694-47a7-887d-c84710e15a45/modules/6712751e-f328-4956-8881-bfe0c7e6cd7b/lessons/c461a06f-883b-4a29-bc43-edc1ffba821b/concepts/ed1a320d-db94-4609-aaba-f02b1f473466
8. Udacity, Notebook: Fraud Detection, Exercise, 2019. URL: https://classroom.udacity.com/nanodegrees/nd009t/parts/670d5990-4694-47a7-887d-c84710e15a45/modules/6712751e-f328-4956-8881-bfe0c7e6cd7b/lessons/cf391ace-c3c9-476b-b357-83ef349eb800/concepts/c30267d5-ac17-47de-a6e1-38fd07709a0d