Machine Learning Nanodegree Program

Capstone Project Report:
Customer Segmentation Report for Arvato Financial Services

Kanarovska Yuliia

September 26th, 2020

Table of Contents

I. Definition

Project Overview

“A Customer Segmentation Report for Arvato Financial Solutions” was one of the proposed projects for the Machine Learning Engineer Nanodegree Program by the Udacity. The main objective of the project is to determine the descriptive portrait of the potential customer and are chances that a new person from the targeted mailout campaign could become a new customer.

Arvato has provided several dataset files that have demographic information about the general population of Germany, current customers of the company, targeted mailout campaign outcomes, and two files with a description of the demographic features.

The project is divided into several subtasks:

1. Data Analysis and Preprocessing;
2. Customer Segmentation Report;
3. Supervised Learning Predictive Model;
4. Kaggle Competition;

The training data is protected under the Terms and Conditions and is unavailable for public sharing.

Domain Background

“Arvato is an internationally active services company that develops and implements innovative solutions for business customers from around the world. These include SCM solutions, financial services and IT services, which are continuously developed with a focus on innovations in automation and data/analytics. Globally renowned companies from a wide variety of industries – from telecommunications providers and energy providers to banks and insurance companies, e-commerce, IT and Internet providers – rely on Arvato’s portfolio of solutions. Arvato is wholly owned by Bertelsmann. [1]”

Arvato efficiently helps its customers with digital transformation, valuable insights, and analysis of the data as well as making better business decisions. “Customer-centric marketing is an approach to marketing that prioritizes customers’ needs and interests in all decisions related to advertising, selling, and promoting products and services. [2]” Understanding correlations and customers' behavior from given data is key to successful customer-centric marketing.

Data analysis techniques and Machine Learning helps to uncover hidden patterns and effectively manipulate large volumes of data with minimum human intervention.

Problem Statement

The formulation of the problem is “Under provided demographic data of the German population and of current customers of the mail-order company, determine the description of the targeted customers' groups

for the given company and verify which new individuals might be acquired as new customers and on which basis."

The proposed solution is divided into 2 subsections. The first step would be to use unsupervised machine learning techniques to segmentize the customers based on the intersection of the data for the current customers and the general population. Secondly, a supervised model will be used on the discovered before data.

Metrics

For the dimensionality reduction algorithm PCA it is better to look at a data variance to decide how many top components to include.

The project belongs to the binary classification problem and it has highly imbalanced data.

```
sns.countplot("RESPONSE", data=mailout_train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x171d08b3b48>
```

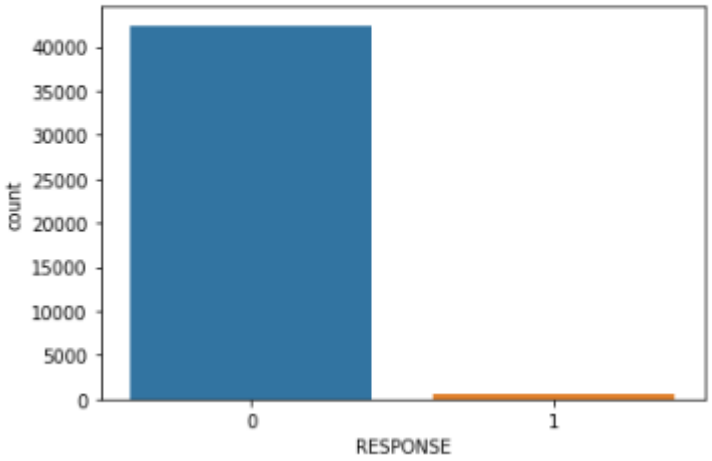

Fig 1 Response balance in the training dataset

Therefore, according to this reason, the chosen evaluation metric is the Area Under the Curve Receiver Operating Characteristics (AUC-ROC).

The AUC-ROC is used to visualize the True Positive Rate against False Positive Rate. Then AUC equals to 1, it means that True Positives and True Negatives are disjoined and perfectly distinguishable, while AUC equals 0 means that the models makes exact opposite classification (all true negatives are classified as positives and vice versa).

Generally speaking, the closer to 1 the result of the AUC the better performance of the model.

II. Analysis

Datasets and Inputs

All the data is provided by Bertelsmann Arvato Analytics and there are given four files for 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).

Additionally, there were 2 more files for describing attributes:

- **DIAS Attributes - Values 2017.xlsx**: Explains values encoding
- **DIAS Information Levels - Attributes 2017.xlsx**: Explains column names meanings

Each row in the demographic data files represents and describes a person as well as his or her environment, such as their household, building, and neighborhood. The general structure of the AZDIAS and CUSTOMERS data files is similar. MAILOUT...TEST and MAILOUT…TRAIN are provided for the development and testing of the supervised model.

Data Exploration

Warning Message

When the data is loaded a warning message appears.

```
C:\Users\Nami_Kaneko\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3063: DtypeWarning: Columns (18,19) have mixe
d types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Fig. 2 The warning message

This warning means that columns 18 and 19 stores the combination of different types. One of the approaches to verify the source of the warning is to extract all the unique values from columns.

```
print('Customers (18, 19): - ', customers.iloc[:, 18:20].columns)
print('Azdias (18, 19): - ', azdias.iloc[:, 18:20].columns)
customers.groupby("CAMEO_DEUG_2015")["CAMEO_DEUG_2015"].count()
```

```
Customers (18, 19): -  Index(['CAMEO_DEUG_2015', 'CAMEO_INTL_2015'], dtype='object')
Azdias (18, 19): -  Index(['CAMEO_DEUG_2015', 'CAMEO_INTL_2015'], dtype='object')

CAMEO_DEUG_2015
1.0     4280
2.0     5910
3.0     4805
4.0     5606
5.0     3042
6.0     4709
7.0     2680
8.0     3333
9.0     1661
1      12498
2      17574
3      13585
4      16458
5       8624
6      14008
7       7878
8       9716
9       4731
X        126
Name: CAMEO_DEUG_2015, dtype: int64
```

Fig. 3 Groups of the unique values from the CAMEO_DEUG_2015 column

The cause of the warning is that there is an actual mix of float, integer, and string types. It is possible to cast an integer to the float and simply transform 'XX' values to the NaN with the help of the NumPy.nan.

Value types for encoding

Because of the warning about mixed-in types in columns 18 and 19, a decision to verify other columns for the 'object' type was made. The verification showed that the dataset contains several columns that store the object-type values.

D19_LETZTER_KAUF_BRANCHE	EINGEFUEGT_AM	OST_WEST_KZ
NaN	NaN	NaN
NaN	1992-02-10 00:00:00	W
D19_UNBEKANNT	1992-02-12 00:00:00	W
D19_UNBEKANNT	1997-04-21 00:00:00	W
D19_SCHUHE	1992-02-12 00:00:00	W

Fig. 4 The snapshot of the table with columns that store object-type values

Because machine learning models cannot process non-numeric values the following columns should undergo the label encoding:

```
CAMEO_INTL_2015, LP_LEBENSPHASE_FEIN, LP_FAMILIE_GROB, LP_STATUS_GROB, EINGEFUE
GT_AM, D19_LETZTER_KAUF_BRANCHE, OST_WEST_KZ, PRODUCT_GROUP, CUSTOMER_GROUP
```

Determine Missing Data

According to the observations in the CUSTOMERS and AZDIAS files were determined 3 distinguishable types of the missing data:

- NaN values that are present in the dataset from the beginning
- X and XX values in columns 18 and 19, that had been handled and converted to NaN values
- In the "DIAS Attributes - Values 2017.xlsx" file in the column Value, there are corresponding encodings for the unknown values represented by numbers.

The next sequence of steps is to collect the labels for unknown values from the "DIAS Attributes - Values 2017.xlsx" and iterate over the entire dataset to replace all unknown values with NaN.

```
Pre-filtered Cusmomer df, no of NaN values:  13864900
Post-filtered Cusmomer df, no of actual NaN values:  14488847
Pre-filtered Azdias df, no of NaN values:  33494042
Post-filtered Azdias df, no of actual NaN values:  37088636
```

Fig. 5 Results of the replacement unknown values with NaN

Determine Outliers

The "DIAS Attributes - Values 2017.xlsx" contains information on the ranges of the proper information for each column, which means that if such a column contains any value that lies beyond the abovementioned range, this value is definitely a mistake and outlier that should be somehow handled. The proposed solution was to store the information on the proper ranges in the form of the dictionary, where the key is the name of the feature and the value is an array with appropriate encodings.

After the actual data was compared with the dictionary, it occurred that the several columns had potential outliers. ANZ_HAUSHALTE_AKTIV, ANZ_HH_TITEL, ANZ_PERSONEN, ANZ_TITEL are should undergo additionally the Inter Quartile Range verification. Although for some columns it was obvious without IQR calculations: in the case with GEBURTSJAHR, which is the Birth Year, 0 makes no sense. For KBA05_MODTEMP, LP_FAMILIE_FEIN, LP_FAMILIE_GROB, LP_LEBENSPHASE_FEIN, LP_LEBENSPHASE_GROB, ORTSGR_KLS9 all other values than NaN are not supposed to exist.

The only question is what would be better: to drop outliers or cap? "Capping can affect the distribution of the data, thus it better not to exaggerate it [3]". Thus some outliers were converted into NaN values, the other was capped by upper and lower limits respectively.

Overcategorized data

While verifying the value ranges, it was a good idea to check the numbers of categories for each column. The calculations showed that some of the features were overcategorized.

```
CAMEO_DEUG_2015, => 10
CAMEO_DEU_2015, => 44
CAMEO_DEUINTL_2015, => 26
CJT_GESAMTTYP, => 7
D19_BANKEN_ANZ_12, => 7
D19_BANKEN_ANZ_24, => 7
D19_BANKEN_DATUM, => 10
D19_BANKEN_DIREKT_RZ, => 8
D19_BANKEN_GROSS_RZ, => 8
D19_BANKEN_LOKAL_RZ, => 8
D19_BANKEN_OFFLINE_DATUM, => 10
```

Fig 6 A snapshot from the categories calculations

The list of columns that are supposed to be overwrited unless they will not be dropped later is:

```
ALTER_HH, CAMEO_DEU_2015, CAMEO_DEUINTL_2015, LP_FAMILIE_FEIN, LP_FAMILIE_GROB,
LP_LEBENSPHASE_FEIN, LP_LEBENSPHASE_GROB, LP_STATUS_FEIN, LP_STATUS_GROB, PRAEG
ENDE_JUGENDJAHRE.
```

Algorithms and Techniques

There are a couple of steps to be performed towards the solution.

1. Features selection (segmentation)
   At this moment the work will be devoted to the appropriate features' selection. By now all features are representing each person, not all of them are required and important for modeling. Therefore, there will be used dimensionality reduction technique - Principal Component Analysis (PCA). "The main basis of PCA-based dimension reduction is
   that PCA picks up the dimensions with the largest
   variances [4]". Then, Elbow Curve will be used to determine the optimal number of clusters for one of the most popular clustering techniques - K-Means.
2. Predictive modeling
   The last step will be devoted to building a supervised model. Because the second part of the project is devoted to binary classification, proposed algorithms were:
   - ❖ KNeighborsClassifier
     "Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point [5]".
   - ❖ DecisionTreeClassifier
     "Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features [6]".
   - ❖ RandomForestClassifier

"A diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers [7]".

- AdaBoostClassifier
  "The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessings, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction [8]".
- GradientBoostingClassifier
  GradientBoosting "is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems in a variety of areas [9]".
- XGBoost Classifier

Benchmark

The first idea for the Benchmark model was Logistic Regression from the scikit-learn library, as it is fast and simple in the implementation.

After fitting our data into the model the following result was obtained:

```
y_pred = lr.predict_proba(X_val)[:,1]
print('Accuracy of logistic regression classifier on validation set: {:.2f}'.format(lr.score(X_val, y_val)))
print('Logistic regression ROC-AUC: {:.2f}'.format(roc_auc_score(y_val, y_pred)))
```

```
Accuracy of logistic regression classifier on validation set: 0.99
Logistic regression ROC-AUC: 0.69
```

Fig 7 The Logistic Regression evaluation

From the observations, we may conclude that a simple accuracy metric indeed has inconsistent performance whereas ROC-AUC is more reliable.

The next step is to test several more complicated models and try to pick the classifiers that can perform better than Linear Regression. Therefore, for this purpose was created a list of benchmark model (that later might undergo the hyperparameters tunning) which included:

- KNeighborsClassifier
- DecisionTreeClassifier
- RandomForestClassifier
- AdaBoostClassifier
- GradientBoostingClassifier
- XGBoost Classifier

The outcome of the training was as follows:

	classifier	score	train_time
0	XGBClassifier	0.784805	14.8941
1	Nearest Neighbors	0.512152	137.117
2	Decision Tree	0.528465	2.09793
3	Random Forest	0.581018	8.72395
4	AdaBoost	0.799279	11.9293
5	GradientBoostingClassifier	0.782638	43.8617

Fig 8 The table of Benchmark model with scores and training time values

Definite leaders are XGBoost, AdaBoost, and Gradient Boosting. However, even though the results of XGBoost and Gradient Boosting are very close, XGBoost is a way faster in training. Therefore, we should proceed with XGBoost and AdaBoost.

III. Methodology

Data Preprocessing

Before fitting data into any of the above-mentioned models, it undergoes the process of refinement and preprocessing. In general, the flow was as follows:

1. Handling the mixed types in columns 18 and 19;
2. Replace encoded unknown values with NumPy.nan;
3. Verify whether `ANZ_HAUSHALTE_AKTIV`, `ANZ_HH_TITEL`, `ANZ_PERSONEN`, `ANZ_TITEL`, `KBA13_ANZAHL_PKW` columns have outliers, if so, then remove;
4. Calculate the percentage of NaN values by columns and rows. Drop columns where the percentage is higher than 30, drop row where the percentage is higher than 50;
5. Calculate the correlation of the features, drop columns where the correlation percentage is higher than 90.
6. Feature engineering;
7. Feature scaling;

Cleaning the missing data

Firstly, the dataset was examined for the quantity of the missing data by each column and every row separately. In some cases, the percentage of the missing values was extremely high (nearly 100%).

```
azdias_missing_cols_df, azdias_missing_rows_df = calculate_missing_data(filtered_azdias)
```

```
Missind data COLUMNs wise:
                         num  percentage
LNR                        0        0.00
AGER_TYP              677503       76.02
AKT_DAT_KL             73499        8.25
ALTER_HH              310267       34.81
ALTER_KIND1           810163       90.90
...                      ...         ...
WOHNDAUER_2008         73499        8.25
WOHNLAGE               93148       10.45
ZABEOTYP                   0        0.00
ANREDE_KZ                  0        0.00
ALTERSKATEGORIE_GROB       0        0.00

[366 rows x 2 columns] (366, 2)
Missind data COLUMNs wise:
         num  percentage
0       259       70.77
11      264       72.13
14      264       72.13
17      264       72.13
24      264       72.13
...     ...         ...
891172  194       53.01
891173  203       55.46
891175  264       72.13
891185  264       72.13
891187  264       72.13

[99969 rows x 2 columns] (99969, 2)
```

Fig. 9 The representation of the missing entries for the AZDIAS dataset

The process of cleaning the missing data was separated into several steps:

1. Calculate the percentage of the missing values by columns for AZDIAS and CUSTOMERS datasets, merge those results.

	index	azdias_%	customers_%
0	LNR	0.00	0.00
1	AGER_TYP	76.02	48.06
2	AKT_DAT_KL	8.25	24.31
3	ALTER_HH	34.81	35.87
4	ALTER_KIND1	90.90	93.86
...	...	...	...
361	WOHNDAUER_2008	8.25	24.31
362	WOHNLAGE	10.45	26.05
363	ZABEOTYP	0.00	0.00
364	ANREDE_KZ	0.00	0.00
365	ALTERSKATEGORIE_GROB	0.00	0.00

366 rows × 3 columns

Fig. 10 Missing data representation per column

It also might be better to understand the situation by visualizing the table.

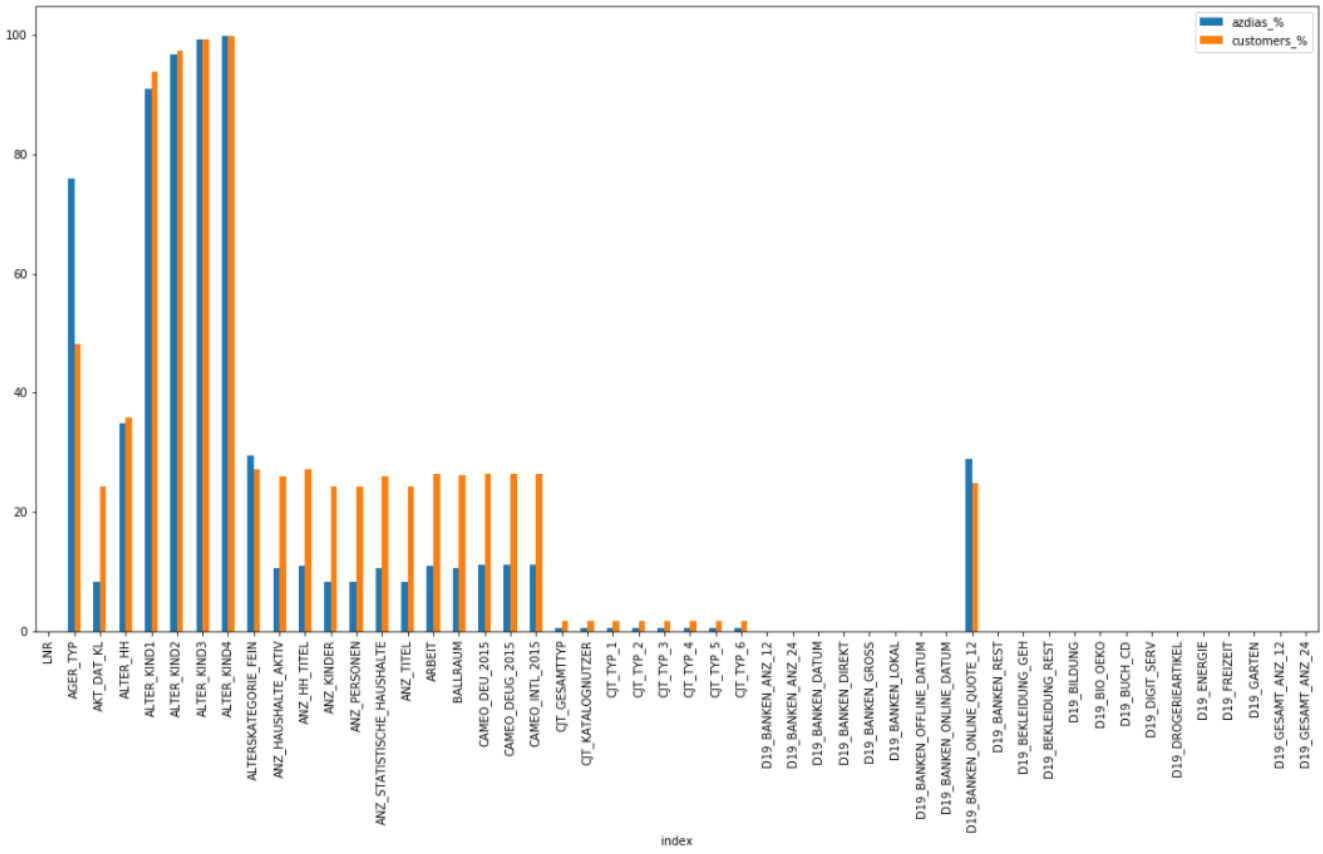

Fig 11 The Bar plot for the missing data from both AZDIAS and CUSTOMERS

2. Find columns where the percentage of missing data was higher than 30% simultaneously in the AZIAS and CUSTOMERS and drop them.

```
cols_to_drop = list(merged.loc[(merged['azdias_%'] > 30) & (merged['customers_%'] > 30)]['index'])
cols_to_drop
```

```
['AGER_TYP',
 'ALTER_HH',
 'ALTER_KIND1',
 'ALTER_KIND2',
 'ALTER_KIND3',
 'ALTER_KIND4',
 'EXTSEL992',
 'GEBURTSJAHR',
 'KBA05_BAUMAX',
 'KK_KUNDENTYP',
 'TITEL_KZ']
```

Fig. 12 Columns common for AZDIAS and CUSTOMERS with a high percentage of NaN values

Now, the shape of datasets changed from

Azdias shape: (891221, 366) => (891221, 355)

Customers shape: (191652, 369) => (191652, 358)

3. The next step was to delete columns that had no description because later it would not be possible to understand the meaning of such columns.
4. By this moment the quantity of missing data has lowered. However, there are still many rows where the overall ratio of NaN values is higher than 50. Therefore, those rows were also dropped.

Feature Engineering and Imputation

First, overcategorized features should be overwritten and divided into new columns. Therefore,

- CAMEO_INTL_2015 was transformed into CAMEO_INTL_2015_wealth_type and CAMEO_INTL_2015_family_type,
- LP_LEBENSPHASE_FEIN was transformed into LP_LEBENSPHASE_FEIN_family_type, LP_LEBENSPHASE_FEIN_earner_type, LP_LEBENSPHASE_FEIN_age_group
- PRAEGENDE_JUGENDJAHRE was transformed into PRAEGENDE_JUGENDJAHRE_movement and PRAEGENDE_JUGENDJAHRE_decade

All the columns that contain non-numeric values were label encoded with the help of the scikit-learn LabelEncoder class. The rest of the missing values that are left after cleaning now can undergo the imputation with the ‘most frequent’ strategy if values in columns have string representation and if they are numerical then imputation with the column mean is used.

Remove highly correlated features

It is known that features that are too highly-correlated may over-inflate the importance of a single feature. Therefore, we should create a correlation matrix and with its help determine the features that correlate higher than the threshold of 90%. Derived columns should be dropped from the dataset.

Columns that possess the abovementioned threshold are

```
CAMEO_DEUG_2015, D19_VERSAND_DATUM, D19_VERSAND_ONLINE_DATUM, D19_VERSAND_ONLIN
E_QUOTE_12, KBA13_HALTER_66, KBA13_HERST_SONST, KBA13_KMH_250, LP_LEBENSPHASE_G
ROB, LP_STATUS_GROB_earner_type, PRAEGENDE_JUGENDJAHRE_movement
```

Feature Scaling

Before moving to the dimensionality reduction, we need to apply feature scaling to the datasets, so that model does not affect by the natural differences in scale of the data and we would not get the misleading components. For the transformation was chosen sklearn.preprocessing.MinMaxScaler, which rescales the data in the range from 0 to 1. “This transformation does not change the distribution of the feature and due to the decreased standard deviations, the effects of the outliers increases. Therefore, before normalization, it is recommended to handle the outliers [10]”.

Implementation

Principal Component Analysis (PCA)

Scaled data is ready to be used for the PCA Algorithm. Since after the cleaned there were 255 features the first step is to understand which features and in what number could represent the variance most efficiently. Therefore the first round of PCA should receive all the data.

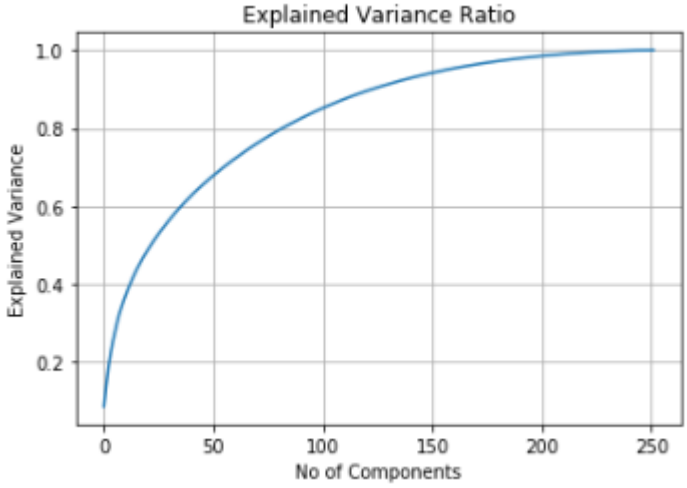

Fig. 13 Explained Variance Ration

According to the graph, we may conclude that 125 features can cover over 90% of the variance in the dataset. Thus, now, we can reduce the number of features to 125.

Elbow Method

The next step is clustering. However, before we may divide the population into different segments, we need to determine the ideal number of clusters (k-value). For solving this task was chosen the Elbow Method. “The elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use [11].”

“The elbow method runs k-means clustering on the dataset for a range of values for k (say from 1-10) and then for each value of k computes an average score for all clusters. By default, the distortion score is computed, the sum of square distances from each point to its assigned center [12].”

We can then collect and plot all the scores for each model and from visualization determine the best k-value.

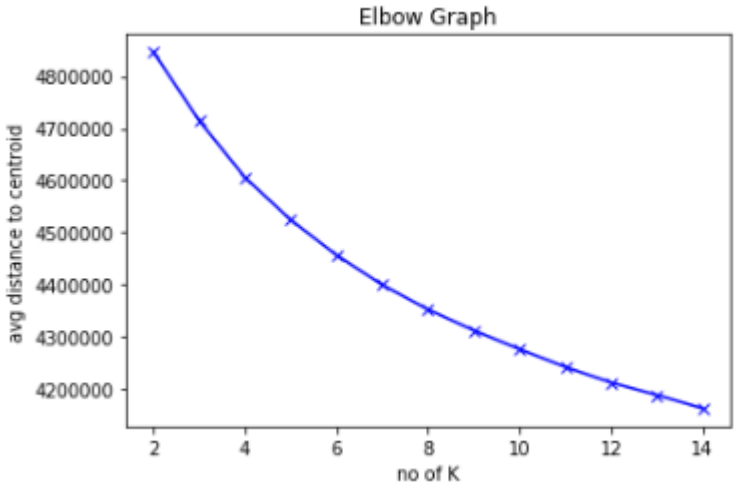

Fig. 14 Elbow graph for 150 components

The number of clusters for our case is 9, as the metric stops to rapidly decrease after this point.

K-Means

“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 (see below). 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 [13].”

Based on the Elbow Method earlier we have chosen to implement KMeans with 9 clusters. After fitting transformed data by PCA into the kmean model we obtained the following results:

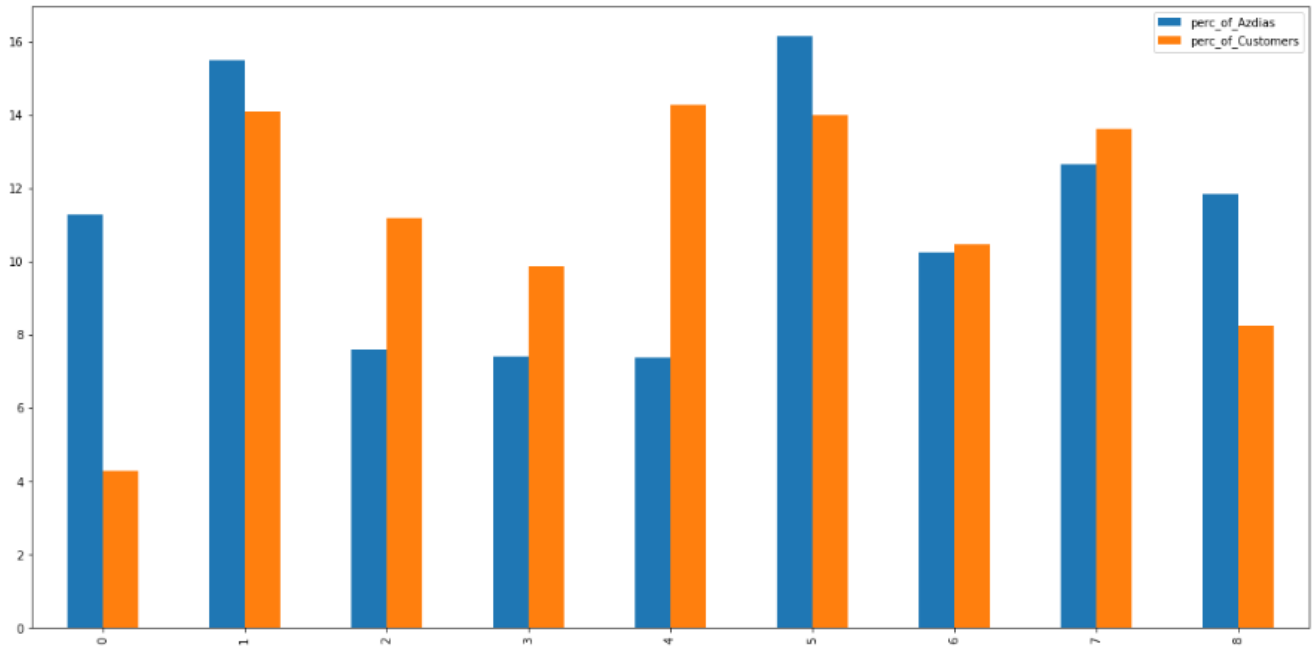

Fig. 15 9 clusters for customers segmentation

According to the graph clusters, 1 and 4 might represent features for the targeted customers, while cluster 0 represents those features that exclude core customers.

In conclusion, the fact of a person being a potential customer is positively affected by the actuality of the last transaction, gender, whether the person from GDR or FRG, with fine social status, the person is dominant minded and dreamily, and fine family type. While financial typology: money saver, number of 6-10 family houses in the PLZ8, and density of inhabitants per square kilometer have a negative impact.

Refinement

Hyperparameters tunning

For the AdaBoost classifier, the following parameters were chosen. As an optimization option was implemented the GridSearchCv. “The grid search provided by GridSearchCV exhaustively generates candidates from a grid of parameter values specified [14].”

```
AB_param_dict = {
 'n_estimators': [25, 50, 100, 150, 200],
 'learning_rate' : [0.01,0.05,0.1,0.3,0.5,1] }

agrid = GridSearchCV(estimator = AdaBoostClassifier(random_state=42),
                     param_grid = AB_param_dict,
                     scoring = "roc_auc",
                     cv = 5,
                     n_jobs=-1,
                     verbose=2)
```

Fig. 16 Hyperparameters grid for the AdaBoost Algorithm

The result was as follows:

```
agrid.best_score_, agrid.best_params_
```

```
(0.7940878574541309, {'learning_rate': 0.05, 'n_estimators': 150})
```

Fig. 17 Best score and parameters for the AdaBoost Algorithm

```
XGB_param_dict = {
    'n_estimators': [300, 400, 500],
    'max_depth': [5, 7, 9],
    'eta': [0.05, 0.2, 0.5],
    'min_child_weight': [1, 5, 10]
}
xgbgrid = GridSearchCV(estimator = xgb.XGBClassifier(random_state=42),
                       param_grid = XGB_param_dict,
                       scoring = "roc_auc",
                       cv = 5,
                       n_jobs=-1,
                       verbose=1)
```

Fig. 18 Hyperparameters grid for the XGBoost Algorithm

Unfortunately, with XGBoost the outcome was not so good. The final best score was lower than the one from AdaBoost.

```
xgbgrid.best_score_, xgbgrid.best_params_
```

```
(0.7635882615059254,
 {'eta': 0.05, 'max_depth': 5, 'min_child_weight': 10, 'n_estimators': 300})
```

Fig. 19 Best score and parameters for the XGBoost Algorithm

IV. Results

The final predictions were made on the Udacity_MAILOUT_052018_TEST.csv, which was pre-processed as the training set.

Fig. 20 Kaggle's score

While the private score of the final performance of my model was 0.79, on Kaggle competition it managed to hold 0.79 of accuracy as well according to the AUC-ROC evaluation metric, which is a good result. However, there is plenty of space for improvement.

Additionally, according to my model D19_SOZIALES feature had the biggest impact. Unfortunately, in the DIAS Information Levels - Attributes 2017.xlsx there is no description for it. Because of the prefix D19, I can assume that it is somehow associated with the Household Information level.

	feature	value
0	D19_SOZIALES	0.66
1	D19_KONSUMTYP_MAX	0.16
2	LP_FAMILIE_FEIN	0.06
3	EINGEZOGENAM_HH_JAHR	0.04
4	CJT_GESAMTTYP	0.04
5	D19_BANKEN_LOKAL	0.02
6	PRAEGENDE_JUGENDJAHRE_decade	0.02
7	WOHNDAUER_2008	0.00
8	KBA13_HALTER_55	0.00
9	KBA13_KRSHERST_AUDI_VW	0.00

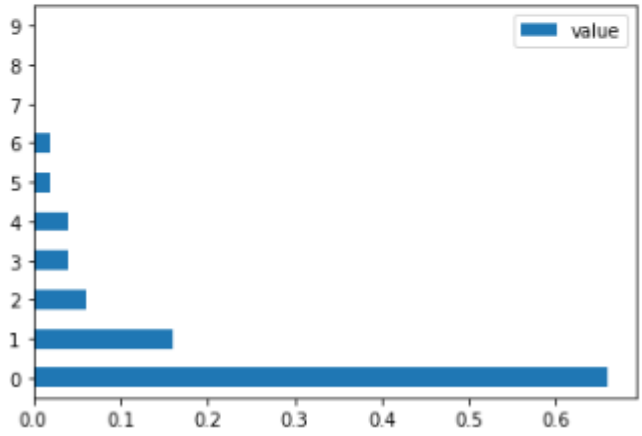

Fig. 21 Highest importance of features from the best model

Justification

If to compare the final results both private and from Kaggle with Linear Regression benchmark model, there was definitely better. Although, the improvement is not high. Generally speaking, the ideal accuracy should be more than 90%. Still, the provided solution adequately handle the problem.

V. Conclusion

Customer-centric marketing is a rapidly developing field, which can benefit from machine learning techniques and capabilities.

Customer Segmentation Report for Arvato Financial Services project has brought me lots of new experience and helped to apply the knowledge that I have obtained while studying for Nanodegree. This project is a representation of a real-life problem in the domain of Data Analysis and Data Science.

Concerning the result, there is still scope for the modifications and improvement. For example, it is possible to implement neural networks for the classification problem, such as a simple fully-connected Neural Network or Convolutional Neural Network. Also, it would be a good idea to rethink the process of data preparation and cleaning, especially verify whether we can derive new features from the existing ones. There is as well a possibility to keep on tuning the hyperparameters of the AdaBoost and XGBoost classifiers.

References

[1] Arvato-Bertelsmann, “Arvato”, Bertelsmann. [Online].

Available: https://www.bertelsmann.com/divisions/arvato/#st-1 [Accessed Sept. 28, 2020]

[2] HelpScout, “How to Build a Winning Customer-Centric Marketing Strategy”, Sarah Chambers. [Online]. Available: https://www.helpscout.com/blog/customer-centric-marketing/ [Accessed Sept. 28, 2020]

[3] Towards Data Science, “Fundamental Techniques of Feature Engineering for Machine Learning”, Emre Rençberoğlu. [Online]. Available: https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114#ad97/ [Accessed Oct. 13, 2020]

[4] “K-means Clustering via Principal Component Analysis”, Chris Ding. 2004

[5] Scikit-learn, “Nearest Neighbors Classification”. [Online].
Available: https://scikit-learn.org/stable/modules/neighbors.html#classification/ [Accessed Oct. 14, 2020]

[6] Scikit-learn, “Decision Trees”. [Online].
Available: https://scikit-learn.org/stable/modules/tree.html#tree/ [Accessed Oct. 14, 2020]

[7] Scikit-learn, “Forests of randomized trees”. [Online].
Available: https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees [Accessed Oct. 14, 2020]

[8] Scikit-learn, “AdaBoost”. [Online].
Available: https://scikit-learn.org/stable/modules/ensemble.html#adaboost [Accessed Oct. 14, 2020]

[9] Scikit-learn, “Gradient Tree Boosting”. [Online].
Available: https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting [Accessed Oct. 14, 2020]

[10] Towards Data Science, “Fundamental Techniques of Feature Engineering for Machine Learning”, Emre Rençberoğlu. [Online]. Available: https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114#ad97/ [Accessed Oct. 13, 2020]

[11] Wikipedia. “Elbow Method”. [Online].
Available: https://en.wikipedia.org/wiki/Elbow_method_(clustering) [Accessed Oct. 13, 2020]

[12] Scikit-Yellowbrick, “Elbow Method”. [Online].
Available: https://www.scikit-yb.org/en/latest/api/cluster/elbow.html#/ [Accessed Oct. 13, 2020]

[13] Scikit-learn, “Clustering”. [Online].
Available: https://scikit-learn.org/stable/modules/clustering.html#k-means/ [Accessed Oct. 13, 2020]

[14] Scikit-learn, “Tuning the hyper-parameters of an estimator”. [Online]. Available: https://scikit-learn.org/stable/modules/grid_search.html#grid-search/ [Accessed Oct. 13, 2020]