Name: Venkata Krishnan Satiyam

CAPSTONE REPORT

Customer segmentation and optimizing the customer acquisition process with arvato financial solutions

Venkata Krishnan Satiyam

Udacity Machine Learning Nanodegree Programme

Name: Venkata Krishnan Satiyam

I. Project Overview

The project requires the user to analyse demographics data for customers of a mail-order sales company in Germany, comparing it against demographics information for the general population. In the first part, unsupervised learning techniques would be applied to perform customer segmentation, identifying the parts of the population that best describe the core customer base of the company.

In the second phase, the techniques applied developed in an unsupervised will be packaged into a single pipeline to pre-process a third dataset with demographics information for targets of a marketing campaign for the company (Training and Test Data).

In the final supervised learning phase, supervised learning techniques would be explored in greater depth whilst using accuracy as a benchmark to evaluate classifier models. The model with the highest accuracy would then be selected to be trained on the processed training dataset, which will then be deployed using Amazon Sagemaker prior to making predictions on a third test dataset.

II. Problem Statement

Untargeted marketing expends resources and leads to a low return on investment. How can we use market research data such as customer user profile and German population data to create a targeted marketing model that can accurately classify if he is a likely customer of the mail order sales company, given that he resides in Germany?

III. Evaluation Metrics

The metrics chosen in this case would be accuracy. In the initial phase for boosting volume of sales, our marketing campaigns must be correctly targeted at portions of German population with higher affinity for our marketing campaigns. Whilst it is debatable that an opportunity cost could be lost due to not considering the false negative population, the project's end goal is a people – centric model, that takes greater consideration for ensuring possibility of rogue calls is reduced or eliminated through accurate classification of False Positive Cases.

According to (Agarwal, 2019), for a classification machine learning problem such the example cited above with a high susceptibility to class imbalance (Chances of a random German resident being a non – customer is significantly higher than being a customer), metrics such as accuracy can fail. However, as we will see in our supervised learning process, oversampling of the minority classes through SMOTE (Synthetic Minority Oversampling Technique) will ensure training dataset is balanced, even prior to model training.

IV. Benchmark

In (Uyar, 2021), Uyar claims achieving a 77% accuracy with dimensionality reduction to 215 principal components at a 95% variance using PCA and k=7 clusters using K-means clustering elbow method. He used SKlearn XGBoost algorithm to create a supervised model to predict outcome of model.

Name: Venkata Krishnan Satiyam

V. Data Analysis

The fact datasets utilized for the data analysis portion of this project include the following:

- Udacity_AZDIAS_052018.csv: Demographics data for the general population of Germany; 891 211-person (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). This dataset additionally contains three additional columns: ('CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP'). While these features are effective for a multinomial prediction project, our end goal would be classification project and these columns can be excluded for the sake of dimensionality reduction.

For the informational datasets, two dimensional/reference data files have been provided for understanding the description of the column features and the type of values present in the feature columns. These dimensional datasets are:

- DIAS Attributes - Values 2017.xlsx: Top level list of attributes and descriptions, organized by international category.

- DIAS Information Levels - Attributes 2017.xlsx: Detailed mapping of data values for each feature in alphabetical order.

IV.I Data Exploration

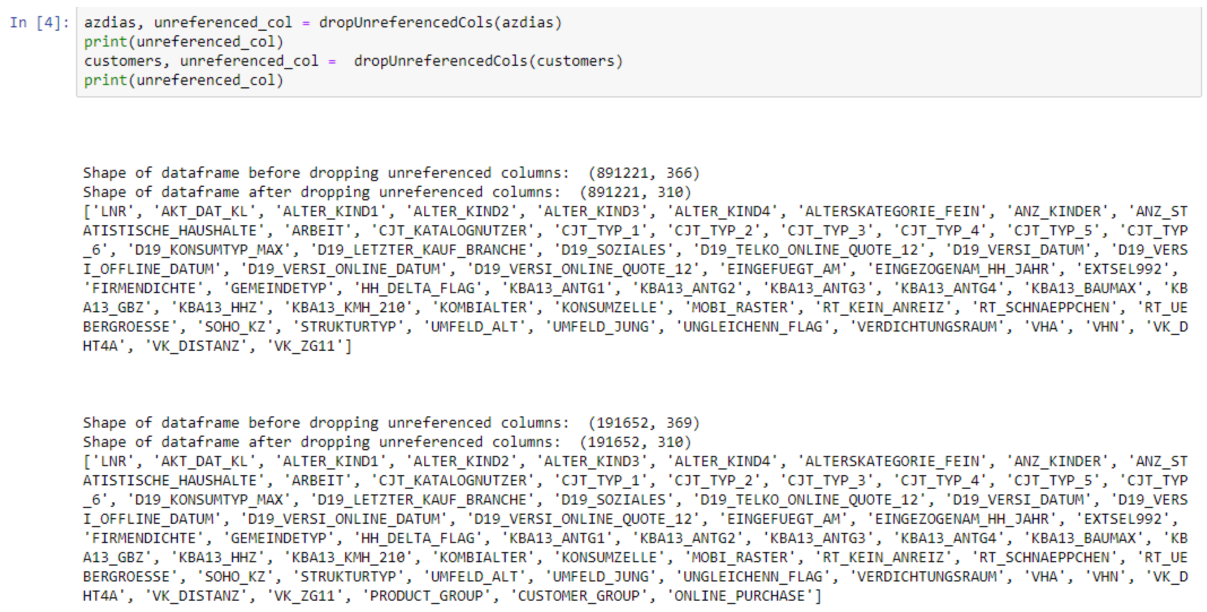
In the data exploration process, five key observations were drawn. They include:

- Inconsistencies in column feature naming between the fact datasets (*azdias/customers*) and the dimensional datasets.
- Unreferenced columns in the *azdias/customers* dataset and the dimensional datasets.
- Inconsistencies in the labelling of unknown values in the dataset within the columns.
- Presence of outliers in the columns with numeric (continuous) data.
- Columns with high percentage of missing data and strong correlations in nullity.
- Columns with nearly perfect correlations presenting with high cramer's associations

IV.I.I Inconsistencies column feature naming and unreferenced columns

Whilst exploring the fact datasets (*azdias/customers*) and the dimensional datasets, there is a high frequency of unreferenced columns in the fact datasets. We can note this through the shape of azdias dataset which has a shape of (891221,366) and customers dataset which has a shape of (191652, 369) whilst for the dimensional dataset, the number of columns is 313. Through the function, ***dropUnreferencedCols*****,** the feature columns in the fact datasets have been synchronized with the dimensional datasets through fixing inconsistencies in the naming of the columns between the fact table and the dimension table as well as dropping any unreferenced columns, preceding the renaming of the columns in the fact table. The pseudocode will be discussed more in the methodology section of the report. The output (Figure 1) shows the output of the code with after performing the function on the azdias and customers datasets. The list of columns printed below are preceding columns that are unreferenced in the dimension dataset.

Output



```
In [4]: azdias, unreferenced_col = dropUnreferencedCols(azdias)
        print(unreferenced_col)
        customers, unreferenced_col =  dropUnreferencedCols(customers)
        print(unreferenced_col)
```

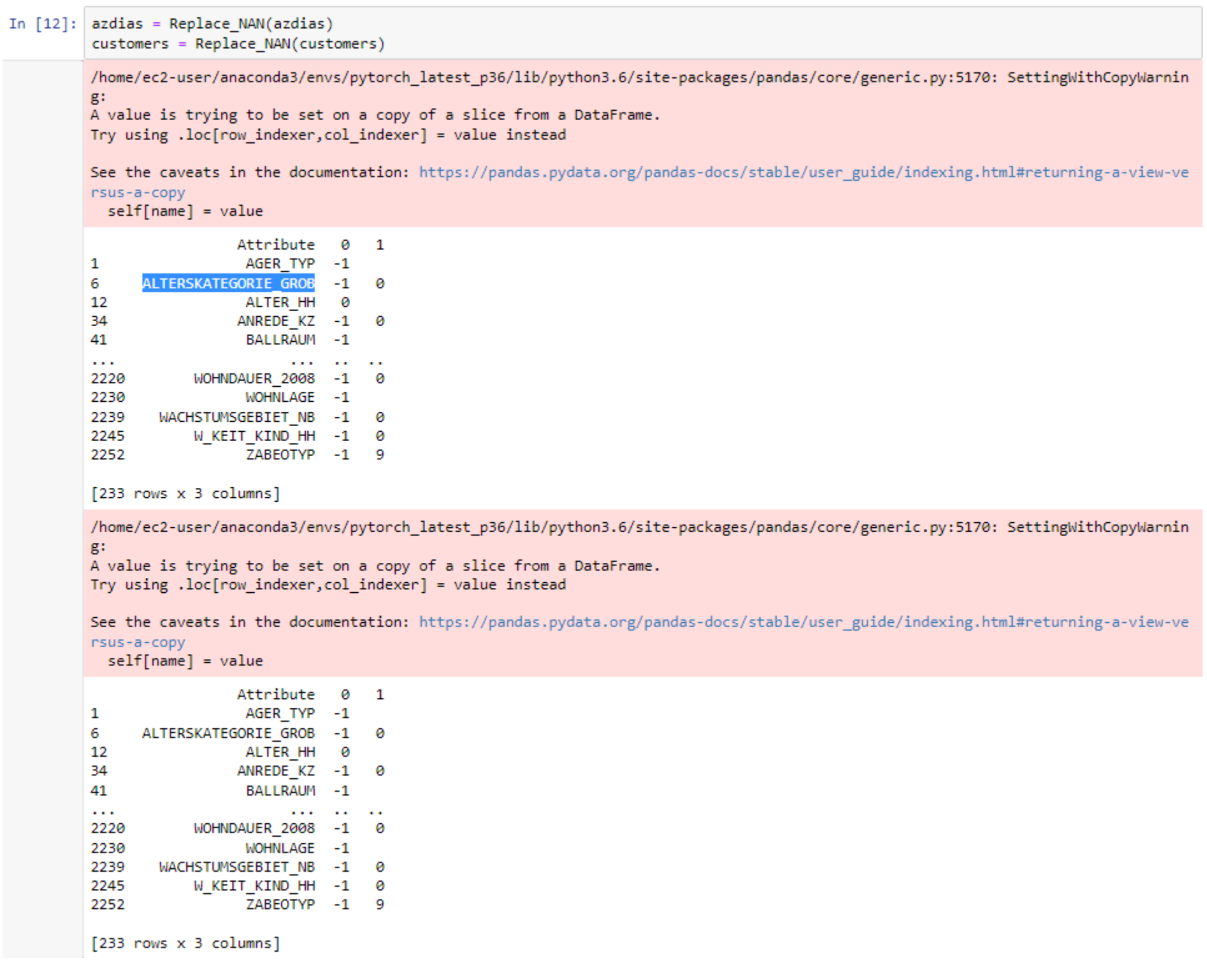
```
Shape of dataframe before dropping unreferenced columns:  (891221, 366)
Shape of dataframe after dropping unreferenced columns:  (891221, 310)
['LNR', 'AKT_DAT_KL', 'ALTER_KIND1', 'ALTER_KIND2', 'ALTER_KIND3', 'ALTER_KIND4', 'ALTERSKATEGORIE_FEIN', 'ANZ_KINDER', 'ANZ_ST
ATISTISCHE_HAUSHALTE', 'ARBEIT', 'CJT_KATALOGNUTZER', 'CJT_TYP_1', 'CJT_TYP_2', 'CJT_TYP_3', 'CJT_TYP_4', 'CJT_TYP_5', 'CJT_TYP
_6', 'D19_KONSUMTYP_MAX', 'D19_LETZTER_KAUF_BRANCHE', 'D19_SOZIALES', 'D19_TELKO_ONLINE_QUOTE_12', 'D19_VERSI_DATUM', 'D19_VERS
I_OFFLINE_DATUM', 'D19_VERSI_ONLINE_DATUM', 'D19_VERSI_ONLINE_QUOTE_12', 'EINGEFUEGT_AM', 'EINGEZOGENAM_HH_JAHR', 'EXTSEL992',
'FIRMENDICHTE', 'GEMEINDETYP', 'HH_DELTA_FLAG', 'KBA13_ANTG1', 'KBA13_ANTG2', 'KBA13_ANTG3', 'KBA13_ANTG4', 'KBA13_BAUMAX', 'KB
A13_GBZ', 'KBA13_HHZ', 'KBA13_KMH_210', 'KOMBIALTER', 'KONSUMZELLE', 'MOBI_RASTER', 'RT_KEIN_ANREIZ', 'RT_SCHNAEPPCHEN', 'RT_UE
BERGROESSE', 'SOHO_KZ', 'STRUKTURTYP', 'UMFELD_ALT', 'UMFELD_JUNG', 'UNGLEICHENN_FLAG', 'VERDICHTUNGSRAUM', 'VHA', 'VHN', 'VK_D
HT4A', 'VK_DISTANZ', 'VK_ZG11']


Shape of dataframe before dropping unreferenced columns:  (191652, 369)
Shape of dataframe after dropping unreferenced columns:  (191652, 310)
['LNR', 'AKT_DAT_KL', 'ALTER_KIND1', 'ALTER_KIND2', 'ALTER_KIND3', 'ALTER_KIND4', 'ALTERSKATEGORIE_FEIN', 'ANZ_KINDER', 'ANZ_ST
ATISTISCHE_HAUSHALTE', 'ARBEIT', 'CJT_KATALOGNUTZER', 'CJT_TYP_1', 'CJT_TYP_2', 'CJT_TYP_3', 'CJT_TYP_4', 'CJT_TYP_5', 'CJT_TYP
_6', 'D19_KONSUMTYP_MAX', 'D19_LETZTER_KAUF_BRANCHE', 'D19_SOZIALES', 'D19_TELKO_ONLINE_QUOTE_12', 'D19_VERSI_DATUM', 'D19_VERS
I_OFFLINE_DATUM', 'D19_VERSI_ONLINE_DATUM', 'D19_VERSI_ONLINE_QUOTE_12', 'EINGEFUEGT_AM', 'EINGEZOGENAM_HH_JAHR', 'EXTSEL992',
'FIRMENDICHTE', 'GEMEINDETYP', 'HH_DELTA_FLAG', 'KBA13_ANTG1', 'KBA13_ANTG2', 'KBA13_ANTG3', 'KBA13_ANTG4', 'KBA13_BAUMAX', 'KB
A13_GBZ', 'KBA13_HHZ', 'KBA13_KMH_210', 'KOMBIALTER', 'KONSUMZELLE', 'MOBI_RASTER', 'RT_KEIN_ANREIZ', 'RT_SCHNAEPPCHEN', 'RT_UE
BERGROESSE', 'SOHO_KZ', 'STRUKTURTYP', 'UMFELD_ALT', 'UMFELD_JUNG', 'UNGLEICHENN_FLAG', 'VERDICHTUNGSRAUM', 'VHA', 'VHN', 'VK_D
HT4A', 'VK_DISTANZ', 'VK_ZG11', 'PRODUCT_GROUP', 'CUSTOMER_GROUP', 'ONLINE_PURCHASE']
```

Figure 1: Output after dropping and renaming unreferenced columns

IV.I.II Inconsistencies in the labelling of unknown values within the dataset columns

An examination of the dimensional datasets reveals a key insight in how unknown values within the feature columns have non – standardized encodings. A key observation is that '-1' is most frequently encoded as being unknown within most columns. Whilst a suggested alternative would be to replace the '-1' values with *nan* within the fact datasets, it may lead to loss of key information in other columns where this categorical encoding has a meaning to its value. Some columns even have multiple values for unknown encodings such as column "ALTERSKATEGORIE_GROB" that contains '-1' and '-9' as encodings for unknown variables. To deal with this, a dataframe of feature columns padded with 2 extra columns containing values encoding unknown variables within these features is used as a reference table for iterating across these columns and replacing with *nan* value on a column – level operation. Figure 2 shows the reference tables that have been created for the *azdias* and *customers* datasets respectively. The pseudocode of will be discussed more in the methodology section.

Name: Venkata Krishnan Satiyam



```
In [12]: azdias = Replace_NAN(azdias)
         customers = Replace_NAN(customers)

/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/pandas/core/generic.py:5170: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  self[name] = value

                Attribute   0   1
1                AGER_TYP  -1
6     ALTERSKATEGORIE_GROB  -1   0
12                ALTER_HH   0
34               ANREDE_KZ  -1   0
41                BALLRAUM  -1
...                   ...  ..  ..
2220        WOHNDAUER_2008  -1   0
2230              WOHNLAGE  -1
2239      WACHSTUMSGEBIET_NB  -1   0
2245         W_KEIT_KIND_HH  -1   0
2252              ZABEOTYP  -1   9

[233 rows x 3 columns]

/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/pandas/core/generic.py:5170: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  self[name] = value

                Attribute   0   1
1                AGER_TYP  -1
6     ALTERSKATEGORIE_GROB  -1   0
12                ALTER_HH   0
34               ANREDE_KZ  -1   0
41                BALLRAUM  -1
...                   ...  ..  ..
2220        WOHNDAUER_2008  -1   0
2230              WOHNLAGE  -1
2239      WACHSTUMSGEBIET_NB  -1   0
2245         W_KEIT_KIND_HH  -1   0
2252              ZABEOTYP  -1   9

[233 rows x 3 columns]
```

Figure 2: Output after standardizing unknown values to np.nan

IV.I.III Presence of outliers

Prior to any data imputation process, it is necessary to undergo outlier removal as the outliers skew the feature values to extremes, resulting in inaccuracies during the imputation process, when filling with the mode or mean of the dataset. Through visualizing the outliers in a box and whiskers plot developed for the azdias and customers datasets, a threshold condition was formulated: The outlier will be removed, given that 95% of the data is within the interquartile range. Additionally, the box – and – whiskers plot also gave further insights within the feature columns allowing for further data cleaning as shown in Figure 3. The steps undertaken will be discussed in the methodology.

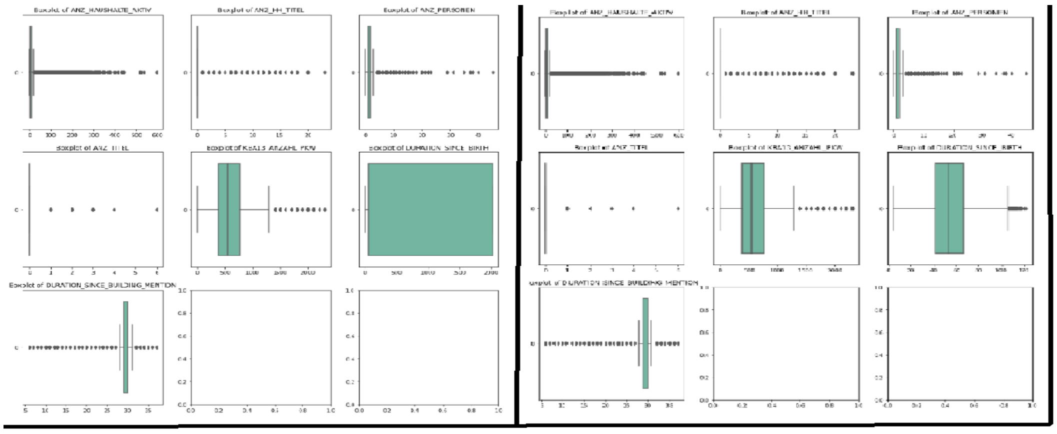


Figure 3: Inconsistencies in column "GEBURTSJAHR" with unknown values padded as 0 causing distortions in box and whiskers plot (Left picture: Before replacing 0 to np.nan; Right: After replacing 0 to np.nan)

Name: Venkata Krishnan Satiyam

IV.I.IV Columns with high percentage of missing data

Another notable observation was that a lot of columns had high percentage of missing data. 5 columns were missing demonstrated such high percentage of missing data in both the azdias and the customers datasets. They are:

- TITEL_KZ column: Flag that shows whether a person holds any academic title.
- D19_KK_KUNDENTYP: Consumption movement in the last 12 months.
- KBA05_BAUMAX: Most common building type within the cell
- GEBURTSJAHR: Year of birth
- AGER_TYP: Best ager-typology (passive elderly/cultural elderly/experience-driven elderly)

To deal with such high percentage of missing data, a heatmap of the nullity coefficients and dendrograms were generated to examine in detail how the nullity of these columns was impacted by nullity of other columns before proceeding to drop these columns. Through the analysis, it was observed that feature "TITEL_KZ", which presented more than 98% missing data, had lower correlations of nullity with the other features and presented itself as Missing Not At Random (MNAR) variable. As for the other columns, "D19_KK_KUNDENTYP" had a higher degree of mismatch, followed by "KBA05_BAUMAX", "AGER_TYP" and lastly by "GEBURTSJAHR".

IV.I.V Reducing multicollinearity through removing columns with perfect correlations.

Multicollinearity undermines the statistical significance of an independent variable as it will influence the coefficients of a regression classifier. Through analysis, it was found that certain columns present perfect correlations with one another. An interesting find was that certain elements has greater correlations with other features. These elements include gender, the lifestage the respective individuals are at, family type, CAMEO detailed classification, social status, wealth of household, mail order transaction activity done online within last six months. While these are generic elements, they are strong predictors that will influence the classifier weights/coefficients in the unsupervised learning if not dealt with accordingly.

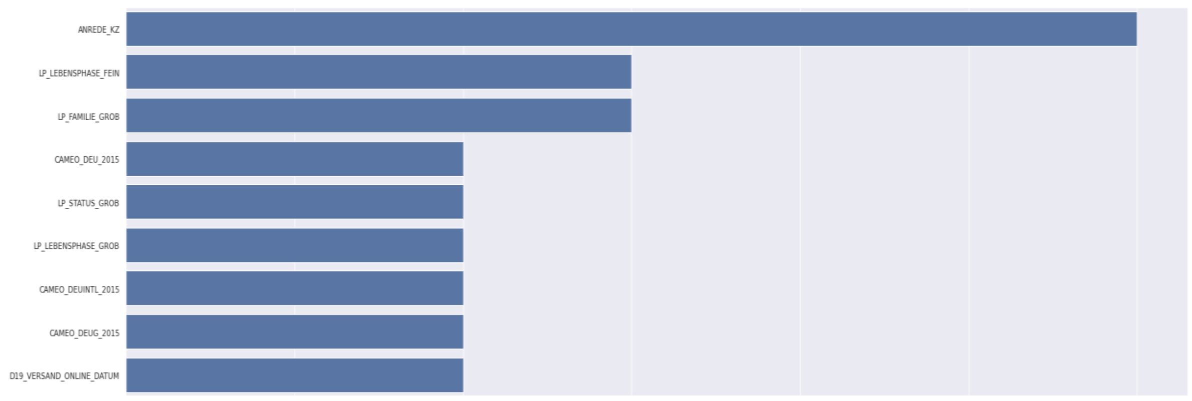


Figure 4:Top 9 features with highest frequency of nearly perfect classifications

VI. Algorithms and Techniques

V.I Unsupervised Learning Algorithms

V.I.I Tukey Outlier Removal

The Tukey outlier removal is a data removal algorithm that uses a box and whisker plot to highlight feature outliers that exists beyond the Interquartile Range[1], as illustrated in Figure 5.

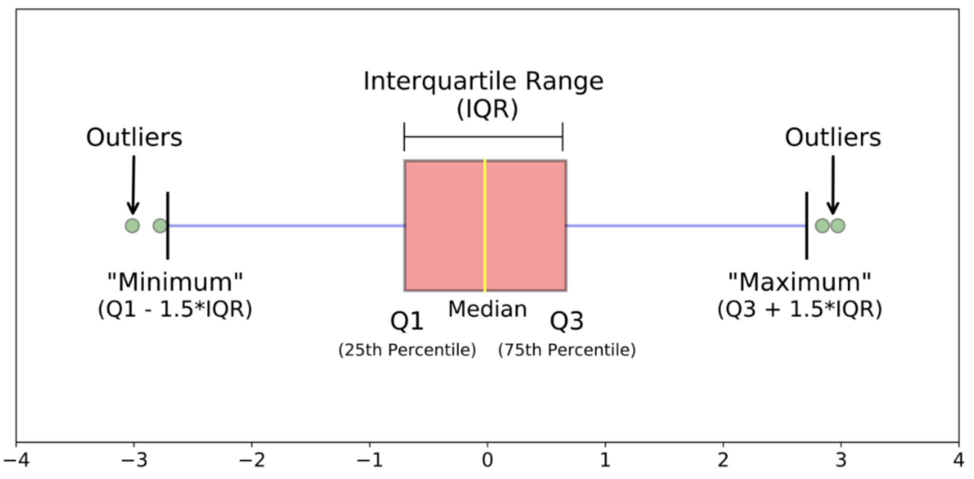


Figure 5: Interquartile Range (Saka, 2019)

V.I.I Chi squared test of independence

The Chi – squared test of independence determines if relationship exists amongst categorical variables for a population (STATISTICS SOLUTION, n.d.). The null hypothesis of a Chi – square distribution is that no relationship exists. A threshold called the P-value is used to accept or reject the hypothesis.

$$\chi^2 = \sum \frac{(f_o - f_e)^2}{f_e}$$

Figure 6: Chi-square value formula (STATISTICS SOLUTION, n.d.)

V.I.II Cramer's rule

The Cramer's value measures the strength of association between two categorical variables. (Hariharan, 2020)

Cramér's V is computed by taking the square root of the chi-squared statistic divided by the sample size and the minimum dimension minus 1:

$$V = \sqrt{\frac{\varphi^2}{\min(k-1, r-1)}} = \sqrt{\frac{\chi^2/n}{\min(k-1, r-1)}}$$

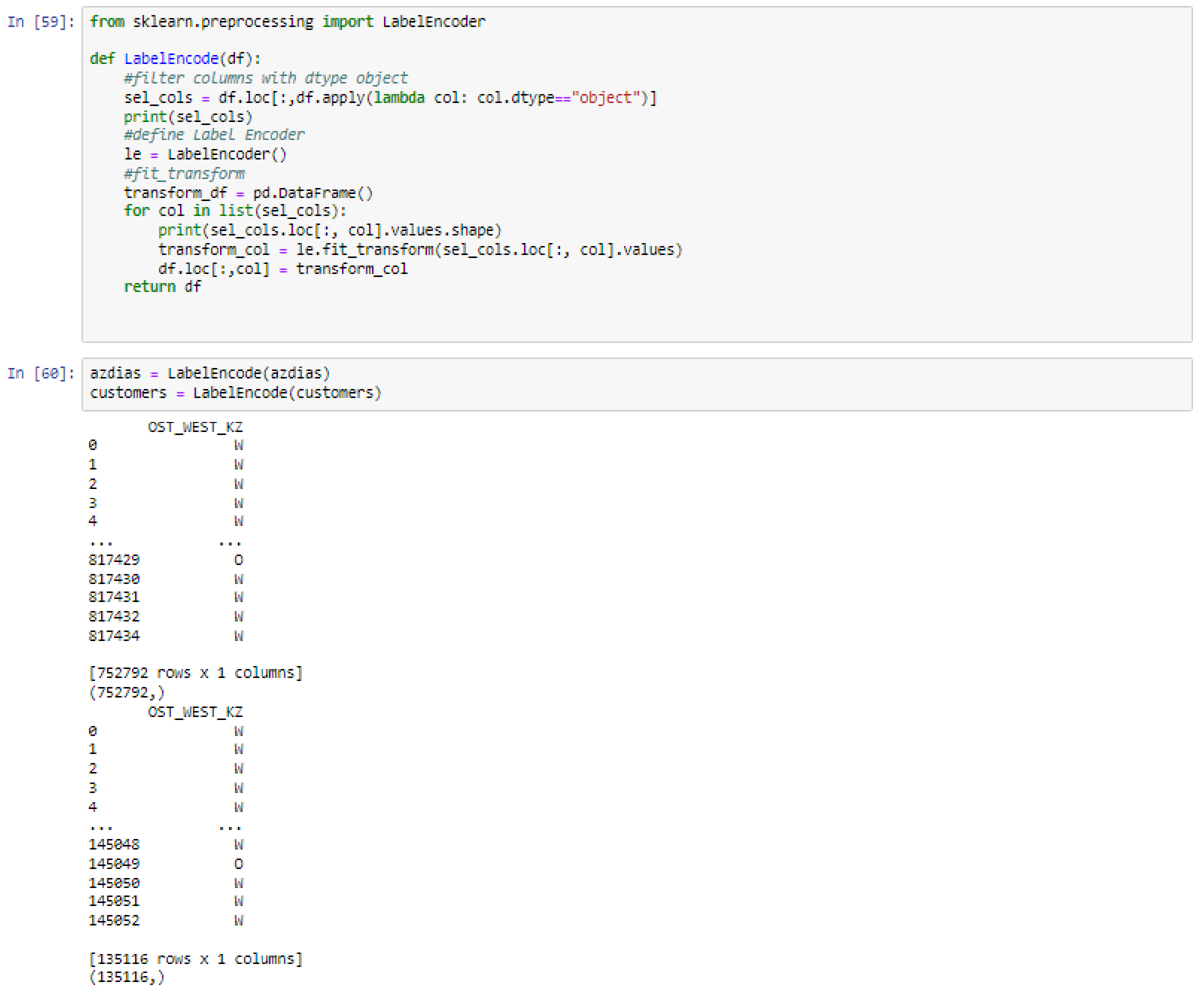
Figure 7: Cramer's Value Computation (Hariharan, 2020)

[1] Defined as the difference between the 25th percentile and the 75th percentile (Saka, 2019)

Name: Venkata Krishnan Satiyam

V.I.III Label Encoder

The label encoder encodes target labels with value between 0 and n_classes 1. In the case of this project, the Label Encoder identified the column that OST_WEST_KZ (Determines whether individual from East Germany or West Germany) and encoded it to 0 or 1 in a binary form.



```
In [59]: from sklearn.preprocessing import LabelEncoder

         def LabelEncode(df):
             #filter columns with dtype object
             sel_cols = df.loc[:,df.apply(lambda col: col.dtype=="object")]
             print(sel_cols)
             #define Label Encoder
             le = LabelEncoder()
             #fit_transform
             transform_df = pd.DataFrame()
             for col in list(sel_cols):
                 print(sel_cols.loc[:, col].values.shape)
                 transform_col = le.fit_transform(sel_cols.loc[:, col].values)
                 df.loc[:,col] = transform_col
             return df
```

```
In [60]: azdias = LabelEncode(azdias)
         customers = LabelEncode(customers)

        OST_WEST_KZ
0                W
1                W
2                W
3                W
4                W
...            ...
817429           O
817430           W
817431           W
817432           W
817434           W

[752792 rows x 1 columns]
(752792,)
        OST_WEST_KZ
0                W
1                W
2                W
3                W
4                W
...            ...
145048           W
145049           O
145050           W
145051           W
145052           W

[135116 rows x 1 columns]
(135116,)
```

Figure 8: Label Encoder Used in the Code

V.I.IV PRINCIPAL COMPONENT ANALYSIS (PCA)

The Principal Component Analysis is a dimensionality reduction method to convert a dataframe with large dimensions to a smaller set but still possessing all the information within it. As shown in Figure 9, the black rotating axis, called the PCA variable maximizes variances of the datapoints from the axis through minimizing the distances of the mapped datapoints to the orthogonal axis. (Jaadi, 2021)

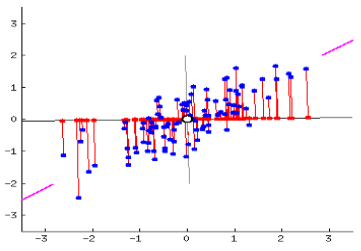


Figure 9: Principal Component Analysis (PCA) maximizes variances of datapoints (Jaadi, 2021)

Name: Venkata Krishnan Satiyam

V.I.V K – MEANS CLUSTERING

The K-means algorithm clusters data by separating samples in k groups of equal variances through minimizing distances from the data points to the cluster centroids. It minimizes a loss function called inertia, which is a sum of squares of these distances. (scikit-learn, n.d.).

VII. METHODOLOGY

The data analytics phase is split into data cleaning and data pre-processing phases. The objective of data cleaning phase would be to fixing inconsistencies in data, outlier removal and uncovering insights into presence of missing data within the feature columns prior to the data pre-processing phase.

In the data pre – processing phase, when the data is considered clean, data undergoes pre – processing steps such as reduction of multicollinearity through dropping highly correlated columns, label encoding for categorical columns, data imputation and feature scaling prior to application of unsupervised learning techniques such as Principal Component Analysis and K-means clustering, which will be covered

VI.I Data Cleaning Phase

In the data cleaning phase, the following steps were conducted:

- Dropping unreferenced columns after synchronizing feature naming consistency between the dimensional and the fact (azdias/customers) data tables.
- Standardization of unknown variables by replacing with *np.nan*
- Tukey Outlier Removal
- Dealing with feature columns having high percentage of missing data
- Removing one of the columns which have nearly perfect correlations.

VI.I.I Synchronizing feature naming consistency and dropping unreferenced features

The pseudocode for carrying out the procedure is as follows:

- Match similarity of column name features in the dimension table to the fact table and vice versa.
- If column exists in fact table but not in the dimension table, it will be a recommended modification for the dimension table and stored as a tuple of format (recommended modification, header) in the list colname_modifications_1.
- Similarly, if column is not in the fact table but in the dimension table, raise it as a recommended modification for fact table. However, it is stored as a tuple of format (header, recommended modification).
- We will try to then produce a list of modifications to make to the fact table through intersecting colname_modifications_1 and colname_modifications_2 and these intersections will be stored in the list approved_colname_mod.

- Typecast approved_colname_mod as a dictionary and raise it as an argument under the columns argument of the function df. rename (…), where df is an arbitrary dataframe like customers or azdias.
- Check again if the columns in the fact table are present in the dimension table. If it is not there, drop the column from fact table as these are unreferenced columns.

V.I.II Standardization of unknown variables by replacing with np.nan

The pseudocode for carrying the procedure is as follows:

- For every feature, find the corresponding value that corresponds to unknown.
- For that specific column, replace those values with np.nan
- Iterate through the remaining columns and return a new dataframe.
- Additionally, in column CAMEO_INTL_2015 and CAMEO_DEUG_2015, replace ‘X’ and ‘XX’ with np.nan

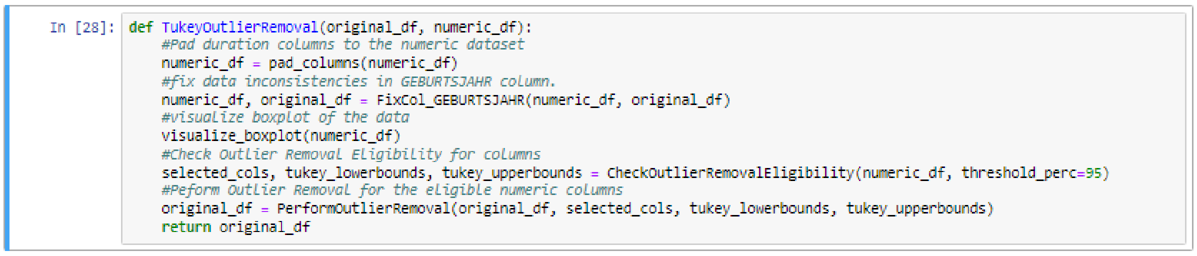
One limitation of this type of replacement is that it utilizes the *DIAS-Attributes – Values 2017.xlsx* file to take reference for the values that correspond to unknown in every feature column. For numeric variables on the other hand, the ‘unknown’ value is not explicitly mentioned in the ‘Meaning’ column (column E) for numeric continuous variables. Therefore, in the next step, the plotting of box and whiskers allows for the data inconsistencies of the numeric columns to be easily fixed.

V.I.III Tukey Outlier Removal

The Tukey Outlier Removal is conducted for only the numeric columns. Thus, below is the pseudocode for the outlier removal process:

- For the numeric columns with datetime format, convert the columns to duration format by subtracting the original column datetime from the datetime now.Pad these 2 extra columns.
- After plotting the box and whiskers plot as shown in the left of Figure 3, it can be observed that column GEBURTSJAHR, corresponding to the year of birth of individual is 0. Replace these values of 0 to np.nan.
- Visualize the boxplot again and check for outlier removal eligibility, which dictates a column’s outliers can only be removed if less than 5% of its data presents themselves as outliers, lying outside the interquartile range. This condition is introduced so as to ensure no major loss of critical information of individuals that presents themselves as a significant outlier pattern is lost.
- For these columns which have eligibility for outlier removal, proceed to remove the outliers from the columns through filtering for values from the main fact data table for the condition of lying outside the IQR range (Figure 5).

Name: Venkata Krishnan Satiyam



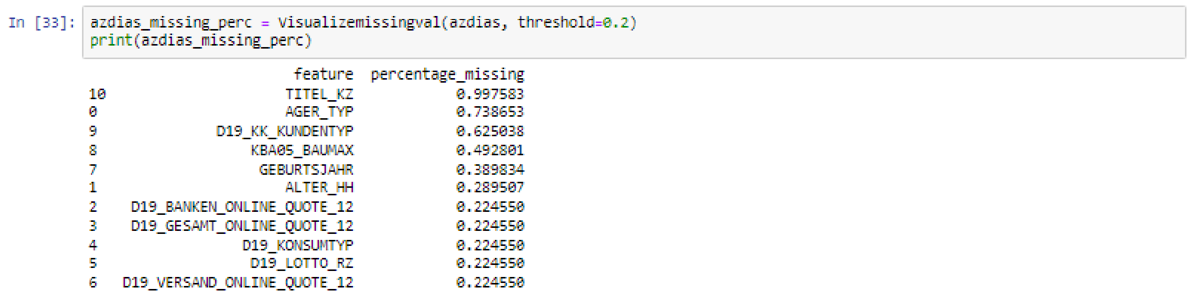
```
In [28]: def TukeyOutlierRemoval(original_df, numeric_df):
             #Pad duration columns to the numeric dataset
             numeric_df = pad_columns(numeric_df)
             #fix data inconsistencies in GEBURTSJAHR column.
             numeric_df, original_df = FixCol_GEBURTSJAHR(numeric_df, original_df)
             #visualize boxplot of the data
             visualize_boxplot(numeric_df)
             #Check Outlier Removal Eligibility for columns
             selected_cols, tukey_lowerbounds, tukey_upperbounds = CheckOutlierRemovalEligibility(numeric_df, threshold_perc=95)
             #Peform Outlier Removal for the eligible numeric columns
             original_df = PerformOutlierRemoval(original_df, selected_cols, tukey_lowerbounds, tukey_upperbounds)
             return original_df
```

Figure 10: Tukey Outlier Removal Code

V.I.IV Dealing with columns with high percentage missing data

Below is the pseudocode for dealing with columns consisting of high percentage of missing data:

1. Use the **Visualizemissingval** to visualize the columns with more than 20% missing data.



```
In [33]: azdias_missing_perc = Visualizemissingval(azdias, threshold=0.2)
         print(azdias_missing_perc)
```

```
                    feature  percentage_missing
10                 TITEL_KZ            0.997583
0                  AGER_TYP            0.738653
9           D19_KK_KUNDENTYP           0.625038
8              KBA05_BAUMAX            0.492801
7               GEBURTSJAHR            0.389834
1                  ALTER_HH            0.289507
2    D19_BANKEN_ONLINE_QUOTE_12        0.224550
3    D19_GESAMT_ONLINE_QUOTE_12        0.224550
4             D19_KONSUMTYP            0.224550
5               D19_LOTTO_RZ           0.224550
6   D19_VERSAND_ONLINE_QUOTE_12        0.224550
```

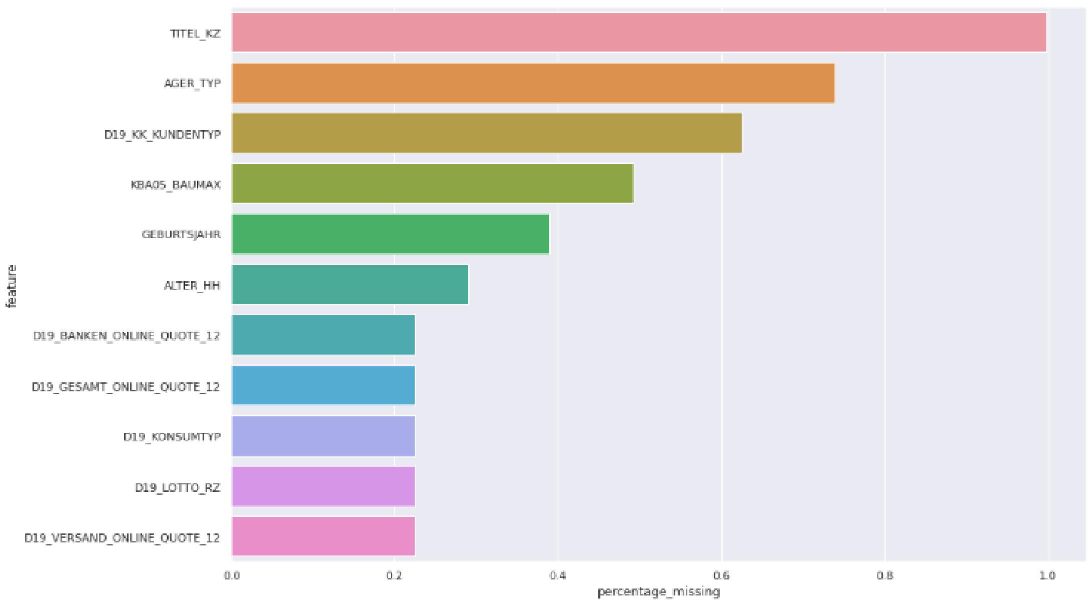


Figure 11:Features with most percentage of missing data (azdias)

Name: Venkata Krishnan Satiyam

```
In [34]: customers_missing_perc = Visualizemissingval(customers, threshold=0.2)
         print(customers_missing_perc)
```

```
            feature  percentage_missing
4          TITEL_KZ            0.984061
3  D19_KK_KUNDENTYP            0.450449
2       KBA05_BAUMAX            0.433883
1        GEBURTSJAHR            0.320055
0          AGER_TYP            0.313747
```

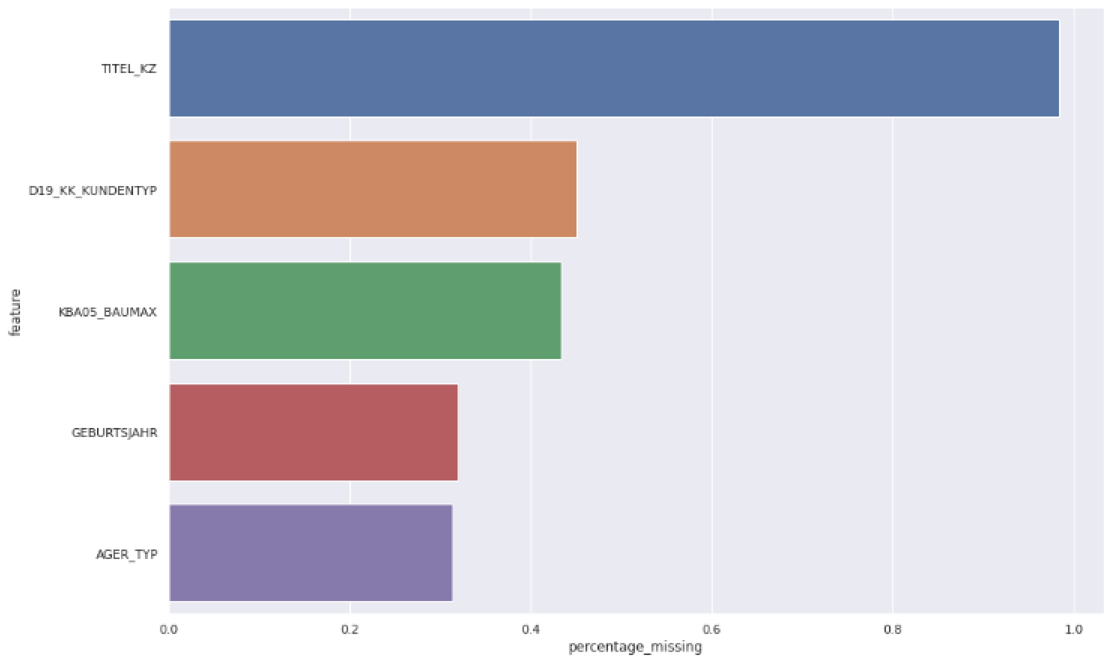


Figure 12: Columns with most amount of missing data (customers)

Figure 11 and Figure 12 summarize the observation in IV.I.IV Columns with high percentage of missing data, which are:

- TITEL_KZ column: Flag that shows whether a person holds any academic title.
- D19_KK_KUNDENTYP: Consumption movement in the last 12 months.
- KBA05_BAUMAX: Most common building type within the cell
- GEBURTSJAHR: Year of birth
- AGER_TYP: Best ager-typology (passive elderly/cultural elderly/experience-driven elderly)

2. We will then proceed to make a nullity heatmaps for both azdias and customers datasets. After making the heatmap, we will study the nullity correlation coefficients for 5 major columns through a bar plot as shown in Figure 5. As seen from the plots, 'TITEL_KZ' presents lower correlations of nullity with the other features. This could be an indicator that data could be completely missing at random, or data could be not missing at random (Purposely omitted).

Additionally, whilst comparing the nullity correlation plots between customers and azdias for every feature, a few observations can be made:

- Customers data is a subset of the azdias dataset due to similar patterns in nullability for the features in general.

Name: Venkata Krishnan Satiyam

- The year of birth presents significantly higher correlations of nullability with the other features in the azdias dataset (moderate to stronger) as compared to the customers dataset, with weaker to moderate correlations of nullity. For the azdias dataset, the year of birth is not null, wherever information on transactional activity (e.g., D19_LOTTO_RZ, D19_VERSAND_ONLINE_QUOTE_12, etc) is present. This is not true for the customers dataset. Another unique observation is that the AGER_TYPE null value count is lower in the customers dataset than the azdias dataset, which is reflective of an aging population pool within the customers dataset. One possibility could be that younger population would prefer to divulge their year of birth more openly in their transactional activity as compared to the older population.

- In both the datasets, we observe that the feature KBA05_BAUMAX has a higher degree of nullity correlation (moderate to stronger) across with all features. This is a clear indication of values within this column missing at random. Generally, except for TITEL_KZ, the other features are clearly missing at random.

- Transactional patterns in the last 12 months that describes whether a person is a regular buyer/active buyer, etc presents higher nullity correlations with the information on transactional activity like year of birth. One possibility could be probably because majority of transactions conducted by the general population in the azdias dataset could be a younger audience conducting online transactions containing higher degree of traceability.

Name: Venkata Krishnan Satiyam

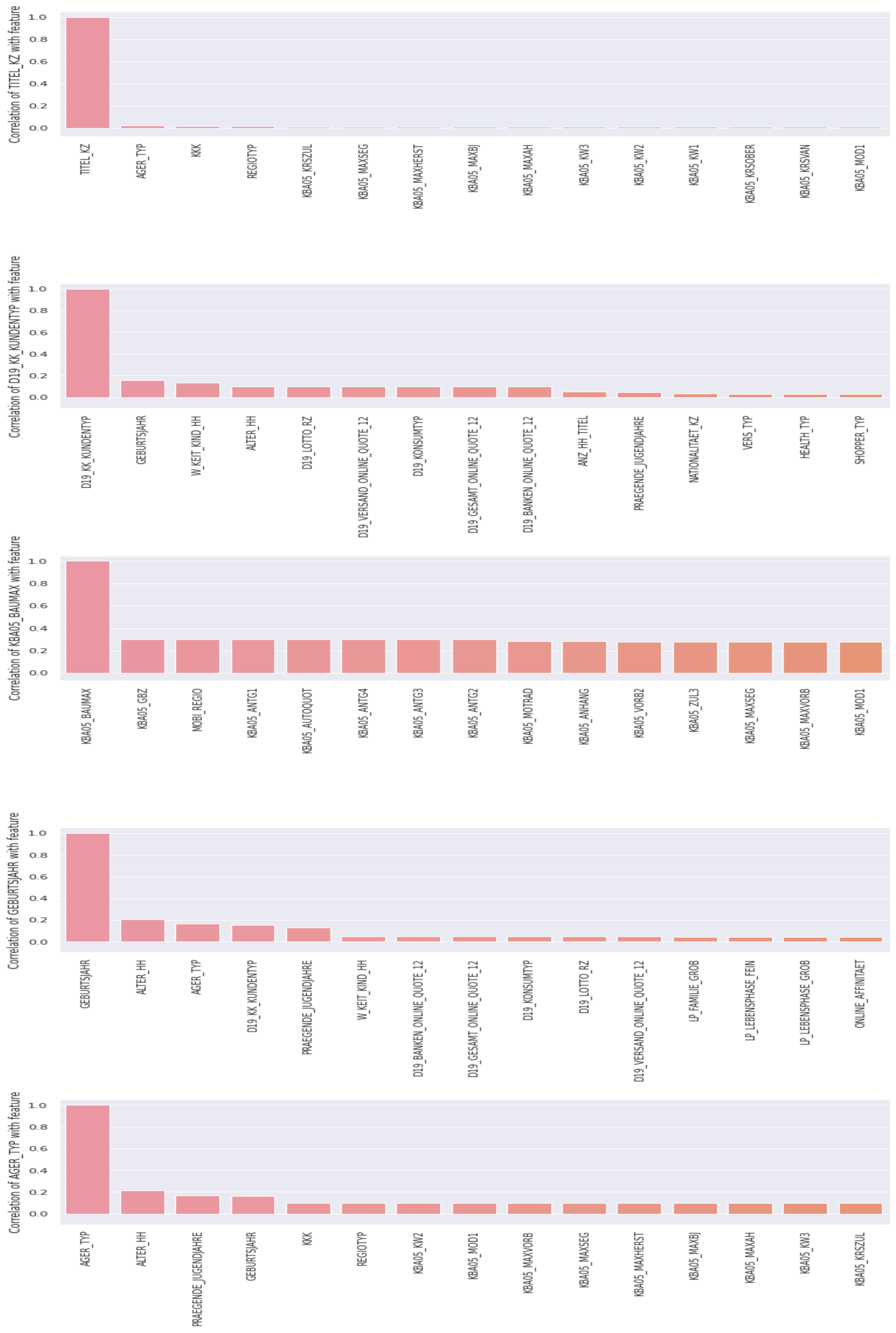


Figure 13:Sorted Barplot of correlation coefficients with other features for the columns with highest missing data

3. We will proceed to create hierarchical dendrograms to gain more insights from the data. To do this, we will import the ***missingno*** library on python and proceed to run msno. dendrogram(azdias) and msno. dendrogram(customers) respectively. These dendrograms are saved in the folder "Miscellaneous/dendrograms."

The dendrograms illustrate that there is a mismatch of 800 listwise variables for azdias dataset that do not match each other in nullity between the each of the feature's TITEL_KZ, AGER_TYP and D19_KK_KUNDENTYP and the rest of the features. One more observation is that the columns, "D19_KK_KUNDENTYP" and "TITEL_KZ" have a higher degree of mismatch, followed by "KBA05_BAUMAX", "AGER_TYP" and lastly by "GEBURTSJAHR". To simplify, whilst dropping these columns might improve the correlations of nullity, it might lead to loss of essential information.

Given the information from the dendrogram and the nullity correlation heatmap, the following will be the actions taken:

- The feature "TITEL_KZ" which could have values missing completely at random or not missing at random should be dropped due to its lack of influence on the nullity of other features.
- There could be a listwise drop of the rows across the entire datasets with the more than 5% missing values with null value present in at least one of the columns but not all columns: "D19_KK_KUNDENTYP", "AGER_TYP", "KBA05_BAUMAX" and "GEBURTSJAHR". By doing this, we will eliminate rows with dissimilarity instead of losing critical information through dropping the columns.

The code snippet below shows the actions undertaken:

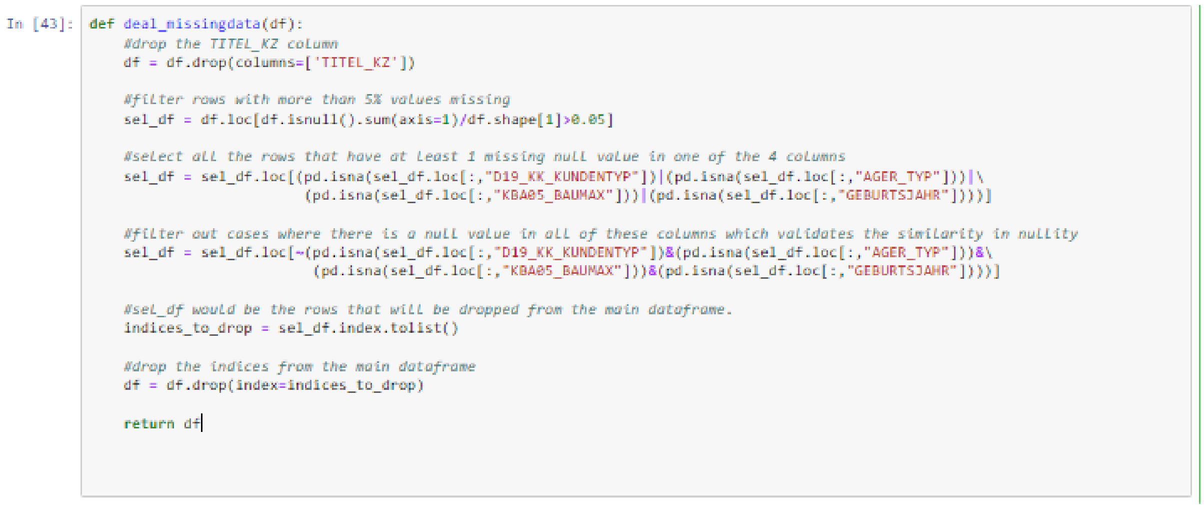

```
In [43]: def deal_missingdata(df):
             #drop the TITEL_KZ column
             df = df.drop(columns=['TITEL_KZ'])

             #filter rows with more than 5% values missing
             sel_df = df.loc[df.isnull().sum(axis=1)/df.shape[1]>0.05]

             #select all the rows that have at least 1 missing null value in one of the 4 columns
             sel_df = sel_df.loc[(pd.isna(sel_df.loc[:,"D19_KK_KUNDENTYP"]))|(pd.isna(sel_df.loc[:,"AGER_TYP"]))|\
                                 (pd.isna(sel_df.loc[:,"KBA05_BAUMAX"]))|(pd.isna(sel_df.loc[:,"GEBURTSJAHR"]))]

             #filter out cases where there is a null value in all of these columns which validates the similarity in nullity
             sel_df = sel_df.loc[~(pd.isna(sel_df.loc[:,"D19_KK_KUNDENTYP"])&(pd.isna(sel_df.loc[:,"AGER_TYP"]))&\
                                  (pd.isna(sel_df.loc[:,"KBA05_BAUMAX"]))&(pd.isna(sel_df.loc[:,"GEBURTSJAHR"])))]

             #sel_df would be the rows that will be dropped from the main dataframe.
             indices_to_drop = sel_df.index.tolist()

             #drop the indices from the main dataframe
             df = df.drop(index=indices_to_drop)

             return df
```

Figure 14: The code deal_missing data removes rows with at least 5% missing data and containing a value in one of the four columns "D19_KK_KUNDENTYP", "AGER_TYP", "KBA05_BAUMAX" and "GEBURTSJAHR" but not all the columns.

Name: Venkata Krishnan Satiyam

V.I.IV Dealing with highly correlated columns

For dealing with highly correlated columns, below is the pseudocode:

1. Apply Chi – squared testing through calling the **Test_Var_Independence** function. The function will create a contingency table between 2 feature columns and calculate the cramer's value through a function **cramerscorrectedstat.** The values from the analysis have been cached in a CSV file within a local directory as "./azdias_cramer_analysis.csv'.

2. Using the null hypothesis and the p-value, create a column "Independent". This column value will be False if p-value is less than critical value of 0.05 (reject null hypothesis of no association) and True otherwise.

3. Using a Strength of Association, create a column "SOA" that takes on categorical values of "Nearly Perfect," "Very Strong"," Strong"," Moderate", "Weak" and "Negligible" respectively.
    a. Cramer val (>0.8): Nearly Perfect
    b. Cramer val (0.6-0.8): Very Strong
    c. Cramer val (0.5-0.6): Strong
    d. Cramer val (0.4-0.5): Moderate
    e. Cramer val (0.3-0.4): Weak
    f. Cramer val (<0.3): Negligible

4. Upon calling the function **VisualizeAssociations,** we will see a bar plot as shown in Figure 4. To see the entire plot, please kindly check the file at the folder location "./Miscellaneous/Nearly_Perfect_Correlations.png."

5. Filter the original table via the column "SOA" to only the "Nearly Perfect" correlations.

6. Drop duplicates for the column, "cramer value."

7. For the remaining non – duplicate correlations existing, drop one of the columns in these correlations by extracting the features to a list variable and re – slicing the list by a step of 2, removing alternate values.

8. Store the list variable **Nearlyperfect_features** and sent this list as an input to **DropNearlyPerfect** function to drop one of the columns giving rise to the "Nearly Perfect" correlations.

VI.II Data Processing Phase

The data processing phase consists of label encoding, data imputation and feature scaling.

VI.II.I Label Encoding of Categorical Columns

The label encoder process consists of the following steps:

- Identify columns with type object
- Use a label encoder on those columns and process.
- Overwrite columns in the original dataset.

The code for the label encoder is in Figure 8. As seen in the output, only one column still had non – numeric alphabets as their values. For ease of imputation, we have converted it to a 0 or 1.

VI.II.II Data Imputation

Data Imputation made use of the Simple Imputer Class from *sklearn* library with the strategy as being "most frequent" due to majority of the columns being categorical in nature. Below are the steps for imputation:

- SimpleImputer() object was created and fit_transform function was called on the azdias dataset.
- After training the imputer model, a transform function was performed on the customers dataset.
- Imputed data frames were cached and stored in the directory, ". /Data_Imputed."

VI.II.III Feature Scaling

For feature scaling, the standard scaler was imported. Like the imputer class, the StandardScaler() object created was fit on the azdias dataset. Then the trained scaler was used to transform azdias dataframe and the customers dataframe.

VIII. IMPLEMENTATION

The implementation consists of unsupervised learning to learn information about over – represented and under – represented clusters prior to the supervised learning of the project.

VII.I DIMENSIONALITY REDUCTION (PCA)

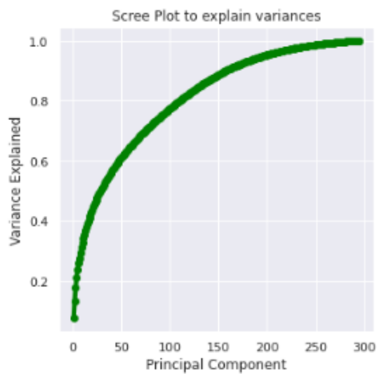
As mentioned under section V.I.IV PRINCIPAL COMPONENT ANALYSIS (PCA), principal component analysis involves reducing dimension of a dataset along its' columns through conversion to lower dimensions that can capture variances of the original features accurately, as good as the original dataset. Prior to PCA, a Scree Plot will be plotted to select only features that explain up to certain percentage (e.g., 95% variance) are selected as they best explain the relationships within the model.

As shown in Figure 15 , 193 principal components can explain up to 95% of the variance within the dataset. Through this method, there is a reduction of 100 columns from the dimensions of the original dataset consisting of 295 columns.

.

```
In [66]: ## UNCOMMENT IF RUNNING FOR FIRST TIME ##
pca_fit, pca, azdias = PCA_fit_transform(azdias, pca_fit = None, pca=None, method="fit_transform", dim_cols=None)
## UNCOMMENT IF RERUNNING FROM DATA IMPUTATION TO PREVENT REFITTING. DIRECTLY PERFORM TRANSFORM ##
#pca_fit, pca, azdias_pca = PCA_fit_transform(azdias, pca_fit = pca_fit, pca=pca, method="transform", dim_cols=None)
_,_,customers_pca = PCA_fit_transform(customers, pca_fit = pca_fit, pca=pca, method="transform", dim_cols=None)
```

```
(752792, 295)
```



```
(135116, 295)
```

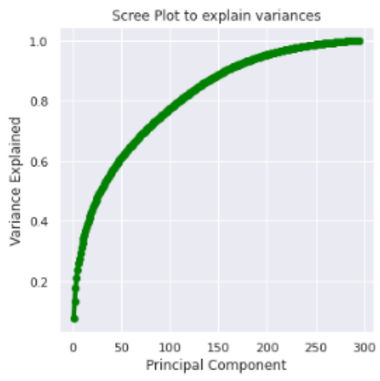


Figure 15: Scree Plots of Customers and Azdias dataframe 193 principal components can explain up to 95% of the variance.

VII.II K-means clustering for forming clusters

The objective of the K-means clustering is to form meaningful clusters within the dataset that can explain critical clusters within the azdias dataset and the customers dataset, which might be under – represented or over – represented. Like the PCA plot, a K-means elbow test had to be conducted for selecting the optimal value of k at which the most unique clusters result. As shown in Figure 12, at k=12, there is an elbow in both the azdias dataset and the customers dataset.

Name: Venkata Krishnan Satiyam

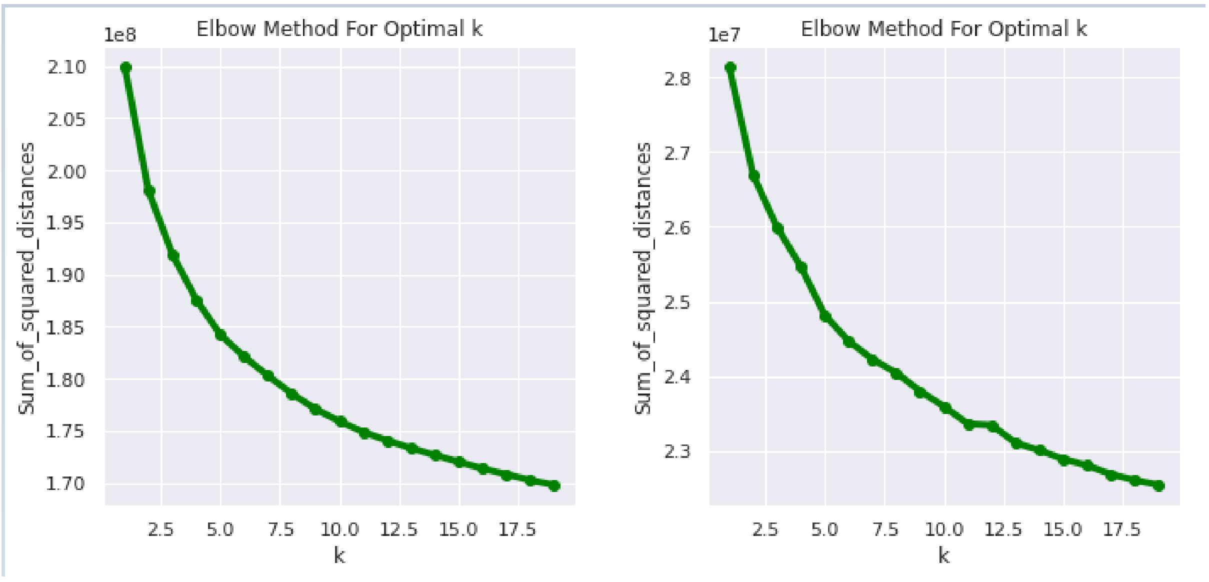


Figure 16: k = 12 selected after the elbow test on azdias (left) and customers (right)

Using the value of k=12, a k – means model was fit and predictions were made on the azdias dataset. The same model was used to make another set of predictions on the customers dataset. After deriving the clusters, a barplot was plotted to find out the percentage of features in the original azdias and customers dataset contribute to the clusters.

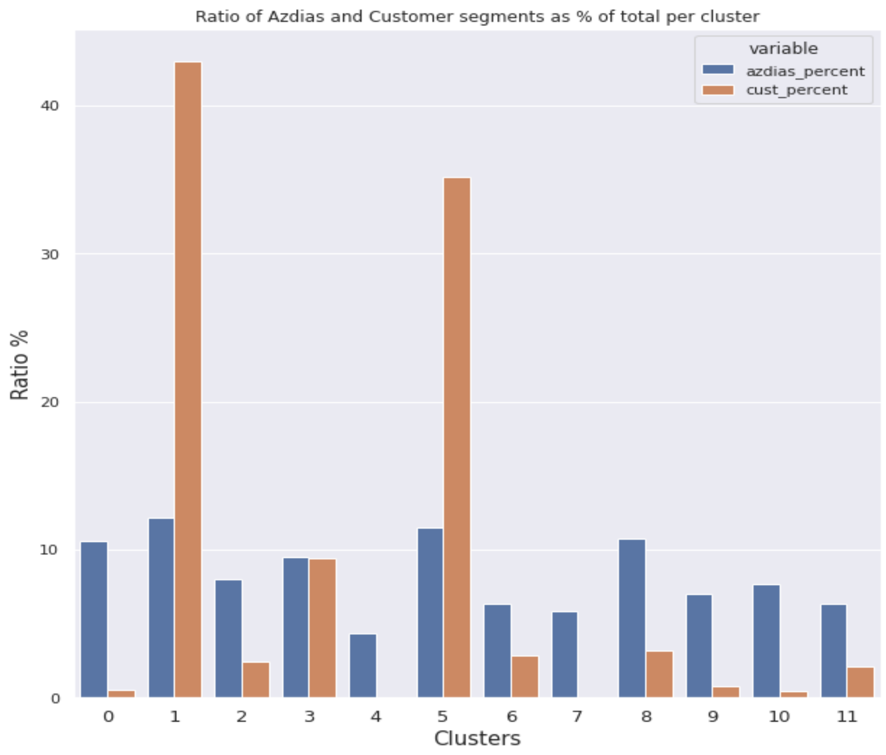


Figure 17:Ratio of Azdias and Customer segments as a percentage of the total per cluster

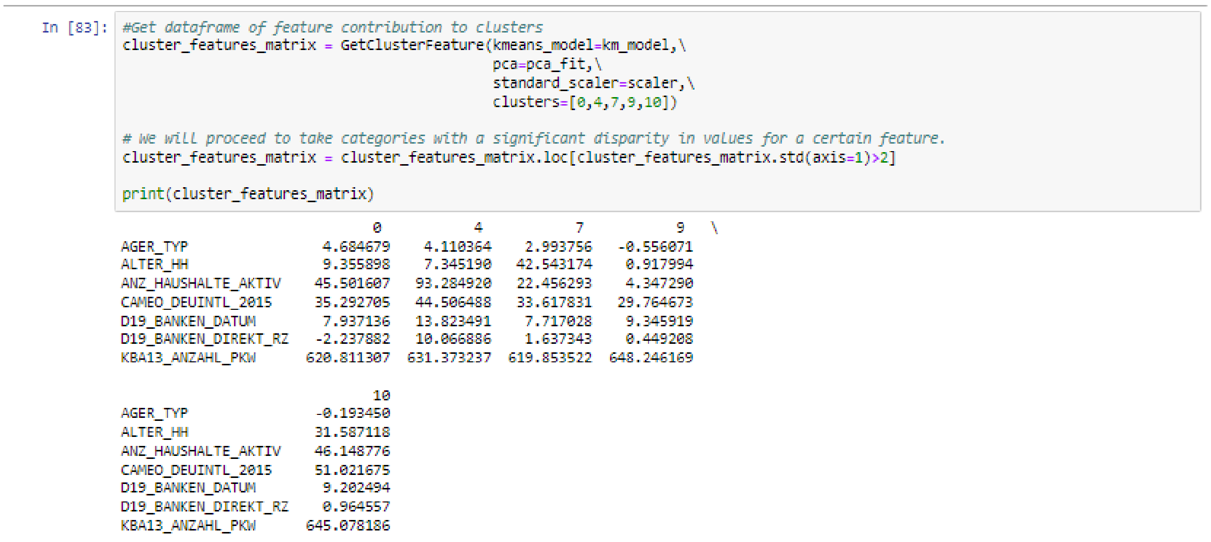
As seen from Figure 17:

- Clusters 1 and 5 are significantly underrepresentative of the customers dataset.
- Clusters 0,4,7,9 and 10 are significantly overrepresentative of the customers dataset.

Name: Venkata Krishnan Satiyam

In the next step, we will do an inverse PCA transform on these k-mean cluster centroids to get back the original features that contribute to the respective clusters. This will give us insights into the differences between the customer group we are dealing with and the general population of Germany, defined by the azdias dataset.

The function **GetClusterFeature** will extract a feature dataframe with its original attribute values. We then proceed to find out specific features whose standard deviation is more than 2 to find major differences in features across the 5 clusters



```
In [83]: #Get dataframe of feature contribution to clusters
         cluster_features_matrix = GetClusterFeature(kmeans_model=km_model,\
                                                     pca=pca_fit,\
                                                     standard_scaler=scaler,\
                                                     clusters=[0,4,7,9,10])

         # We will proceed to take categories with a significant disparity in values for a certain feature.
         cluster_features_matrix = cluster_features_matrix.loc[cluster_features_matrix.std(axis=1)>2]

         print(cluster_features_matrix)
```

```
                             0           4           7           9  \
AGER_TYP              4.684679    4.110364    2.993756   -0.556071
ALTER_HH              9.355898    7.345190   42.543174    0.917994
ANZ_HAUSHALTE_AKTIV  45.501607   93.284920   22.456293    4.347290
CAMEO_DEUINTL_2015   35.292705   44.506488   33.617831   29.764673
D19_BANKEN_DATUM      7.937136   13.823491    7.717028    9.345919
D19_BANKEN_DIREKT_RZ -2.237882   10.066886    1.637343    0.449208
KBA13_ANZAHL_PKW    620.811307  631.373237  619.853522  648.246169

                            10
AGER_TYP             -0.193450
ALTER_HH             31.587118
ANZ_HAUSHALTE_AKTIV  46.148776
CAMEO_DEUINTL_2015   51.021675
D19_BANKEN_DATUM      9.202494
D19_BANKEN_DIREKT_RZ  0.964557
KBA13_ANZAHL_PKW    645.078186
```

For example, for the clusters 0,4,7,9 and 10, as we have been seeing in the data cleaning and data preprocessing steps, we observe they are of 5 major categories:

Cluster 0:

- Comfortable Households-Elders In Retirement,
- Actuality of the last transaction for the segment banks TOTAL shown an activity elder than 1,5 years
- Owns car on a lesser occasion
- Lives in a flat
- Born between 1935 and 1939

Cluster 4:

- Born between 1925 and 1929
- Less Affluent Households-Older Families & Mature Couples
- Don’t perform any transactions in bank

Cluster 7:

- Comfortable Households-Families With School Age Children
- Multi-buyer of the product group DIRECT BANKS

Cluster 9:

- Owns cars more frequently

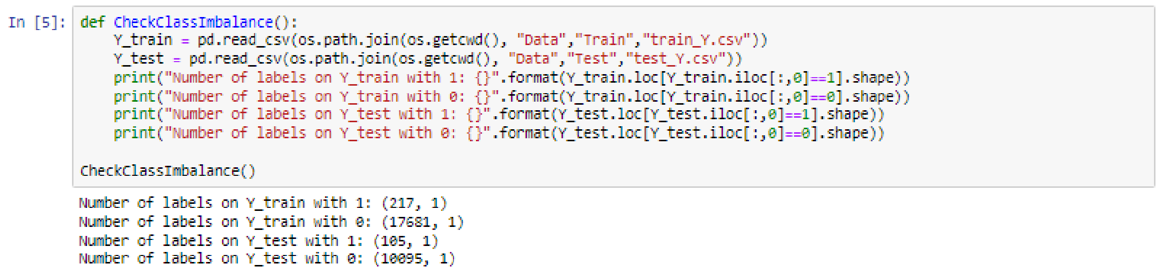
- A mix of Prosperous Households-Elders In Retirement and Comfortable Households-Pre-Family Couples & Singles

Cluster 10:

- Poorer Households-Pre-Family Couples & Singles
- Generally live in non – crowded households.

VII.III DEALING WITH CLASS IMBALANCES

To study for class imbalances, we will have to first look at the training dataset labels and look at their value counts. This is done by the **CheckClassImbalance** function.

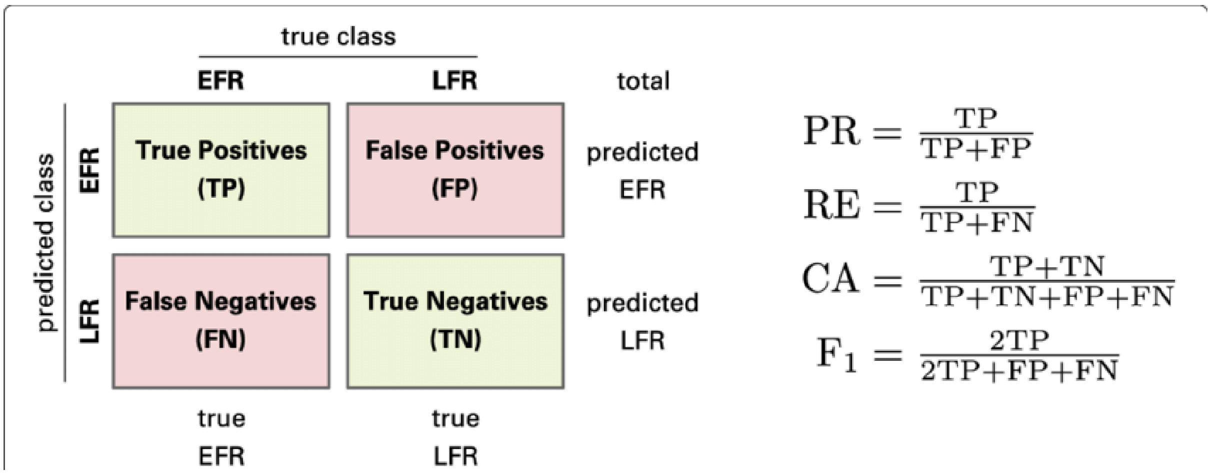


```
In [5]: def CheckClassImbalance():
            Y_train = pd.read_csv(os.path.join(os.getcwd(), "Data","Train","train_Y.csv"))
            Y_test = pd.read_csv(os.path.join(os.getcwd(), "Data","Test","test_Y.csv"))
            print("Number of labels on Y_train with 1: {}".format(Y_train.loc[Y_train.iloc[:,0]==1].shape))
            print("Number of labels on Y_train with 0: {}".format(Y_train.loc[Y_train.iloc[:,0]==0].shape))
            print("Number of labels on Y_test with 1: {}".format(Y_test.loc[Y_test.iloc[:,0]==1].shape))
            print("Number of labels on Y_test with 0: {}".format(Y_test.loc[Y_test.iloc[:,0]==0].shape))

        CheckClassImbalance()
```

```
Number of labels on Y_train with 1: (217, 1)
Number of labels on Y_train with 0: (17681, 1)
Number of labels on Y_test with 1: (105, 1)
Number of labels on Y_test with 0: (10095, 1)
```

Figure 18: Checking for class imbalance prior to model training

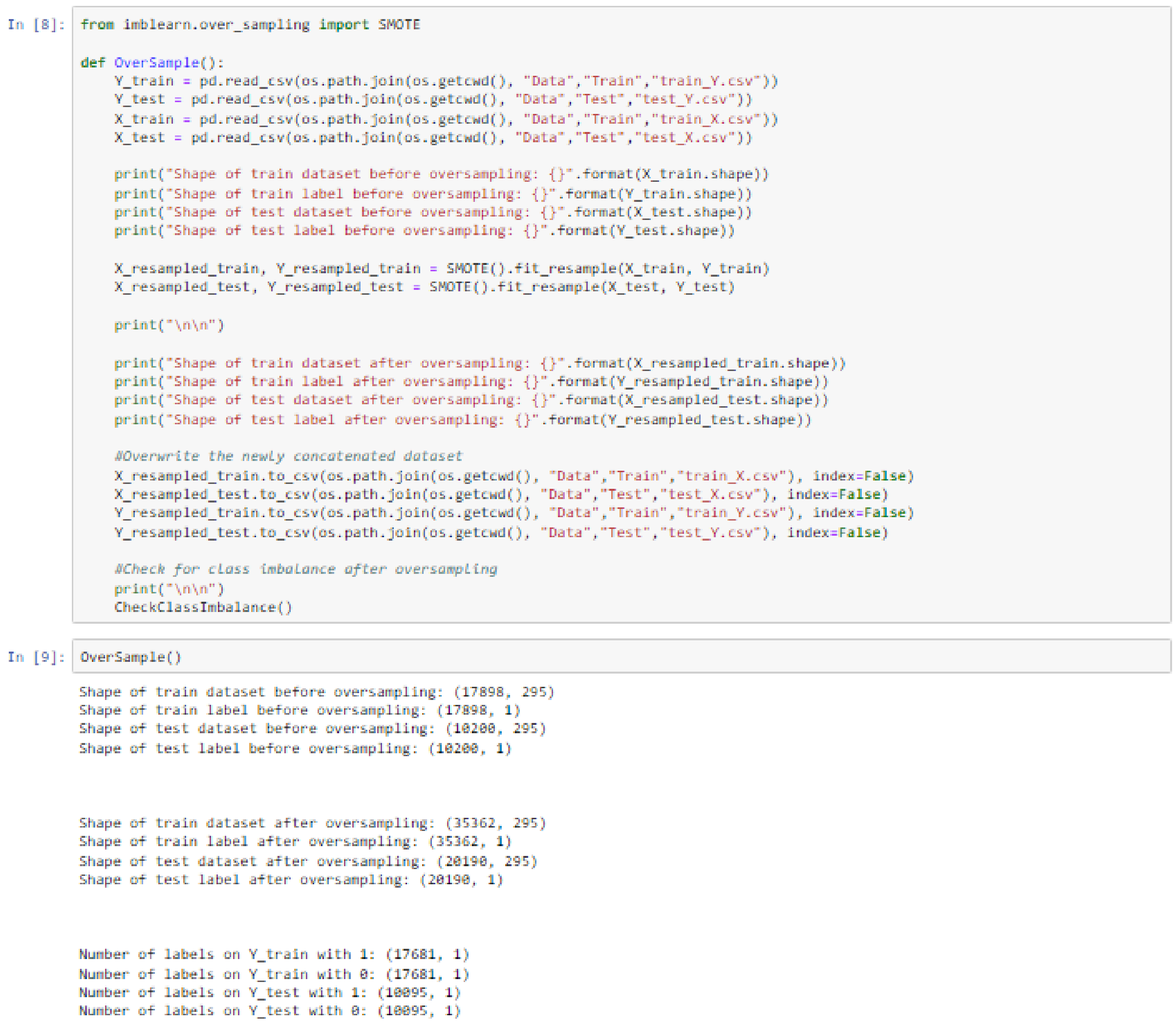
In this scenario, given that the number of actual positive cases is low (chances of picking a customer is extremely low) and a classifier is developed to predict everything as being negative, the accuracy would be falsely presented as a high value.



$$\mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$
$$\mathrm{RE} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$
$$\mathrm{CA} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$$
$$\mathrm{F}_1 = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$$

Figure 19: Contingency Table (Bittrich, n.d.)

To deal with this class imbalance, a technique called "OverSampling" is used, whereby the new examples of data can be synthesized from the existing data with a actual positive case. The algorithm used for oversampling the data is SMOTE (Synthetic Minority Oversampling Technique) and can be imported through the imblearn library. SMOTE uses K-nearest neighbour algorithm in its backend to synthesize the data (Wijaya, 2020). The function **OverSample()** as shown in utilizes SMOTE in the project to oversample the minority classes.



```
In [8]: from imblearn.over_sampling import SMOTE

        def OverSample():
            Y_train = pd.read_csv(os.path.join(os.getcwd(), "Data","Train","train_Y.csv"))
            Y_test = pd.read_csv(os.path.join(os.getcwd(), "Data","Test","test_Y.csv"))
            X_train = pd.read_csv(os.path.join(os.getcwd(), "Data","Train","train_X.csv"))
            X_test = pd.read_csv(os.path.join(os.getcwd(), "Data","Test","test_X.csv"))

            print("Shape of train dataset before oversampling: {}".format(X_train.shape))
            print("Shape of train label before oversampling: {}".format(Y_train.shape))
            print("Shape of test dataset before oversampling: {}".format(X_test.shape))
            print("Shape of test label before oversampling: {}".format(Y_test.shape))

            X_resampled_train, Y_resampled_train = SMOTE().fit_resample(X_train, Y_train)
            X_resampled_test, Y_resampled_test = SMOTE().fit_resample(X_test, Y_test)

            print("\n\n")

            print("Shape of train dataset after oversampling: {}".format(X_resampled_train.shape))
            print("Shape of train label after oversampling: {}".format(Y_resampled_train.shape))
            print("Shape of test dataset after oversampling: {}".format(X_resampled_test.shape))
            print("Shape of test label after oversampling: {}".format(Y_resampled_test.shape))

            #Overwrite the newly concatenated dataset
            X_resampled_train.to_csv(os.path.join(os.getcwd(), "Data","Train","train_X.csv"), index=False)
            X_resampled_test.to_csv(os.path.join(os.getcwd(), "Data","Test","test_X.csv"), index=False)
            Y_resampled_train.to_csv(os.path.join(os.getcwd(), "Data","Train","train_Y.csv"), index=False)
            Y_resampled_test.to_csv(os.path.join(os.getcwd(), "Data","Test","test_Y.csv"), index=False)

            #Check for class imbalance after oversampling
            print("\n\n")
            CheckClassImbalance()

In [9]: OverSample()

        Shape of train dataset before oversampling: (17898, 295)
        Shape of train label before oversampling: (17898, 1)
        Shape of test dataset before oversampling: (10200, 295)
        Shape of test label before oversampling: (10200, 1)



        Shape of train dataset after oversampling: (35362, 295)
        Shape of train label after oversampling: (35362, 1)
        Shape of test dataset after oversampling: (20190, 295)
        Shape of test label after oversampling: (20190, 1)



        Number of labels on Y_train with 1: (17681, 1)
        Number of labels on Y_train with 0: (17681, 1)
        Number of labels on Y_test with 1: (10095, 1)
        Number of labels on Y_test with 0: (10095, 1)
```

Figure 20: OverSampling Data through SMOTE algorithm. Output shows how class imbalance has been solved.

VI.IV SUPERVISED LEARNING ALGORITHM

The supervised learning algorithm makes use Amazon Sagemaker to run the training, deployment, and prediction of the model. To train a custom *sklearn* estimator via the SageMaker interface, a train.py script is created. This train.py script:

- Loads training data from a specified directory
- Parses any training & model hyperparameters
- Instantiate a model with any specified hyperparameters
- Trains that model
- Finally, saves the model so that it can be hosted/deployed, later

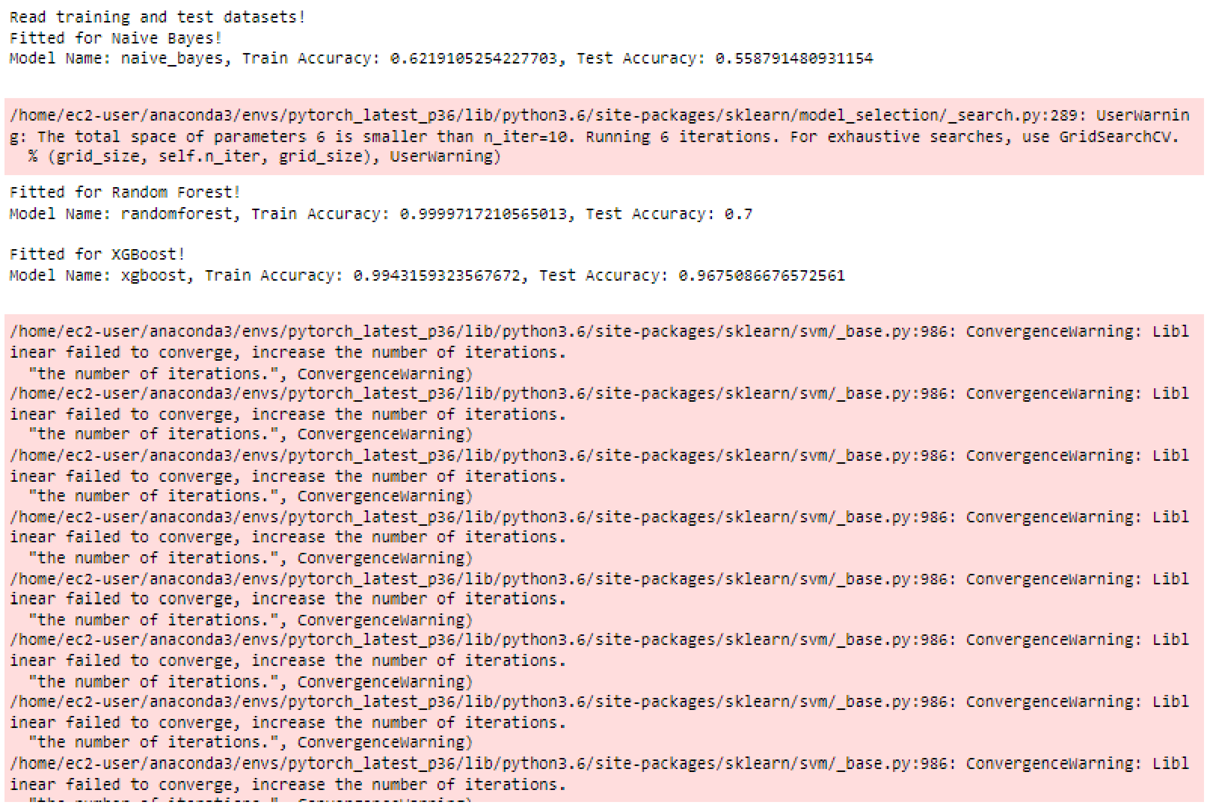
Whilst there are a range of classifiers to choose from for devising an appropriate estimator model, the type of classifier and metrics formulated will be based upon the objective considered. The objective considered for this scenario would be to predict how likely a member of the public in Germany would be a customer to Arvato Financial Services. As this is a binary classification project, we will investigate four main algorithms with high accuracies for handling Binary Classification which are namely Linear

Name: Venkata Krishnan Satiyam

Support Vector Machines, Naïve Bayes Multinomial Classifier, Random Forests (Decision Tree) and Gradient Boosted Trees.

We will proceed to write a function that takes in the sklearn models of the four classifiers, implements RandomnizedSearchCV and optimizes the model to its best hyperparameters. Amongst the models, we will select the one with the highest score on the test and training dataset. To aid in the model selection, a function **RandomnizedSearch** is written.

This function takes in a model name, scoring metric, a train dataset and test dataset to tune the model hyperparameters over a range of hyperparameters. After training the model on the train dataset and having tuned the hyperparameters, the best estimator model selected for each sklearn model is scored against a test dataset and a training dataset. Based on the scores, an analyst can have an intuition as to which model will rank highly for the task.



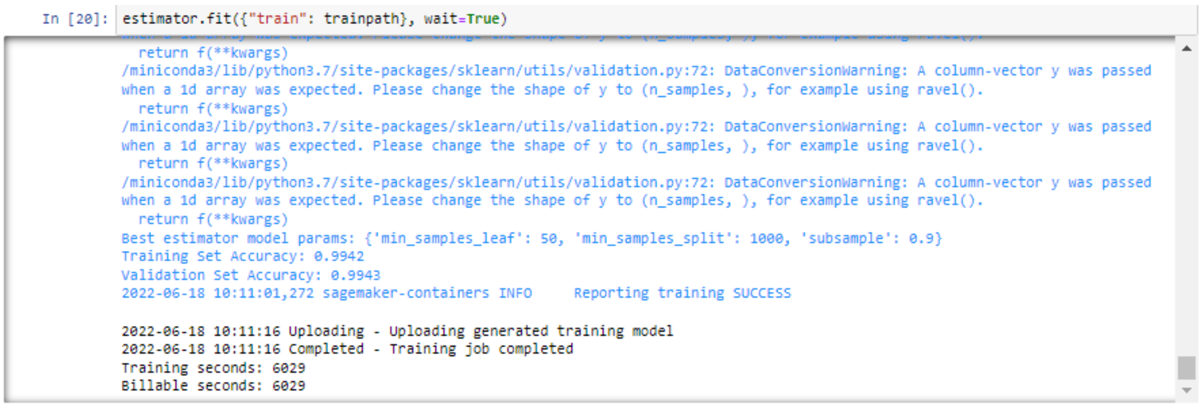
```
Read training and test datasets!
Fitted for Naive Bayes!
Model Name: naive_bayes, Train Accuracy: 0.6219105254227703, Test Accuracy: 0.558791480931154

/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/sklearn/model_selection/_search.py:289: UserWarnin
g: The total space of parameters 6 is smaller than n_iter=10. Running 6 iterations. For exhaustive searches, use GridSearchCV.
  % (grid_size, self.n_iter, grid_size), UserWarning)

Fitted for Random Forest!
Model Name: randomforest, Train Accuracy: 0.9999717210565013, Test Accuracy: 0.7

Fitted for XGBoost!
Model Name: xgboost, Train Accuracy: 0.9943159323567672, Test Accuracy: 0.9675086676572561

/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/sklearn/svm/_base.py:986: ConvergenceWarning: Libl
inear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/sklearn/svm/_base.py:986: ConvergenceWarning: Libl
inear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/sklearn/svm/_base.py:986: ConvergenceWarning: Libl
inear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/sklearn/svm/_base.py:986: ConvergenceWarning: Libl
inear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/sklearn/svm/_base.py:986: ConvergenceWarning: Libl
inear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/sklearn/svm/_base.py:986: ConvergenceWarning: Libl
inear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/sklearn/svm/_base.py:986: ConvergenceWarning: Libl
inear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/home/ec2-user/anaconda3/envs/pytorch_latest_p36/lib/python3.6/site-packages/sklearn/svm/_base.py:986: ConvergenceWarning: Libl
inear failed to converge, increase the number of iterations.
```

Figure 21: RandomnizedSearchCV scores on RandomForest, Naive Bayes and gradient boosted tree algorithm

As seen from the training and test scores, naïve bayes algorithm that assumes features are independent does not perform well in contrast to the Random Forest Algorithm and gradient boosted tree algorithm. For RandomForest algorithm scores, we observe overfitting (High Bias Error and Low Variance). This could be because slight changes in data changes the outcome of the decision trees entirely, thus causing it to be prone to overfitting.

For the Gradient Boosted Trees, whilst training took long to complete, modifying the learning rate to a higher value, optimizing the *min_samples_leaf* and *min_samples_split* gives rise to a model that can score highly on both the training and test datasets.

Once, the model was chosen, the gradient boosted tree model was declared on the train.py script, whereby, it underwent further rounds of hyperparameter tuning through a GridSearchCV. In the train.py script, the hyperparameters tuned were *min_samples_split, subsample* and *min_samples_leaf.*

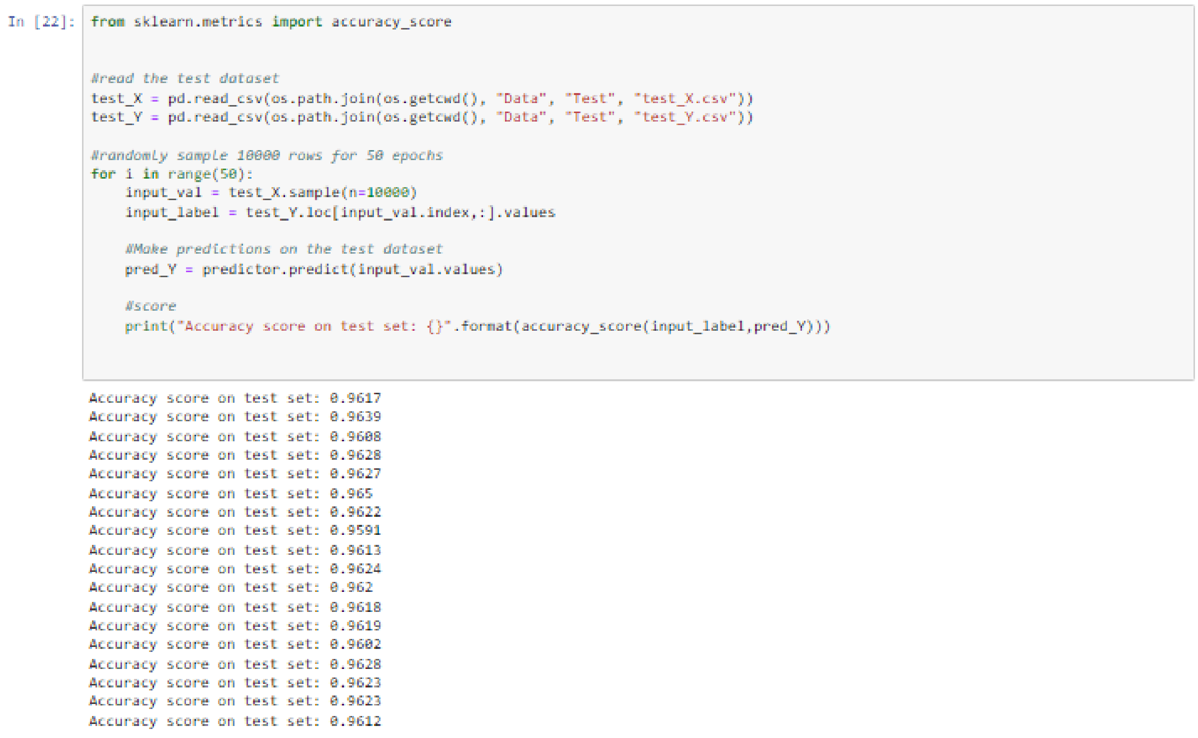


```
In [20]: estimator.fit({"train": trainpath}, wait=True)
```

```
  return f(**kwargs)
/miniconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:72: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
/miniconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:72: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
/miniconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:72: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
Best estimator model params: {'min_samples_leaf': 50, 'min_samples_split': 1000, 'subsample': 0.9}
Training Set Accuracy: 0.9942
Validation Set Accuracy: 0.9943
2022-06-18 10:11:01,272 sagemaker-containers INFO     Reporting training SUCCESS

2022-06-18 10:11:16 Uploading - Uploading generated training model
2022-06-18 10:11:16 Completed - Training job completed
Training seconds: 6029
Billable seconds: 6029
```

Figure 22: Gradient Boosting Tree model accuracy after tuning through GridSearchCV algorithm

The best fit model chosen was deployed on sagemaker and its accuracy was scored whilst making predictions on the test set.



```
In [22]: from sklearn.metrics import accuracy_score


         #read the test dataset
         test_X = pd.read_csv(os.path.join(os.getcwd(), "Data", "Test", "test_X.csv"))
         test_Y = pd.read_csv(os.path.join(os.getcwd(), "Data", "Test", "test_Y.csv"))

         #randomly sample 10000 rows for 50 epochs
         for i in range(50):
             input_val = test_X.sample(n=10000)
             input_label = test_Y.loc[input_val.index,:].values

             #Make predictions on the test dataset
             pred_Y = predictor.predict(input_val.values)

             #score
             print("Accuracy score on test set: {}".format(accuracy_score(input_label,pred_Y)))
```

```
Accuracy score on test set: 0.9617
Accuracy score on test set: 0.9639
Accuracy score on test set: 0.9608
Accuracy score on test set: 0.9628
Accuracy score on test set: 0.9627
Accuracy score on test set: 0.965
Accuracy score on test set: 0.9622
Accuracy score on test set: 0.9591
Accuracy score on test set: 0.9613
Accuracy score on test set: 0.9624
Accuracy score on test set: 0.962
Accuracy score on test set: 0.9618
Accuracy score on test set: 0.9619
Accuracy score on test set: 0.9602
Accuracy score on test set: 0.9628
Accuracy score on test set: 0.9623
Accuracy score on test set: 0.9623
Accuracy score on test set: 0.9612
```

Figure 23: Few of the fifty iterations of predictions on test set produced. Model developed scored ~96% accuracy on test set.

IX. Improvements

Whilst the model produced an extremely high accuracy that outperformed the benchmark stated at 75%, there is more room for improvement. As this is a model developed in the initial stages to minimize rogue calls to disinterested parties during the marketing campaign, the next phase of development could be to focus on using ROC metric to tackle the opportunity cost of missing potential customers, thinking they are disinterested. Another improvement would be to use an XGBoost Model with an advanced regularization capability, allowing it to achieve higher model generalization capabilities.

X. References

1. Agarwal, R. (18 September, 2019). *towardsdatascience*. Retrieved from The 5 Classification Evaluation metrics every Data Scientist must know: https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

2. Bittrich, S. (n.d.). *Research Gate*. Retrieved from https://www.researchgate.net/figure/Confusion-matrix-Exemplified-CM-with-the-formulas-of-precision-PR-recall-RE_fig1_330174519.

3. Hariharan, S. (24 September, 2020). *Statistical test for MCAR in python…*. Retrieved from Towards Data Science: https://towardsdatascience.com/statistical-test-for-mcar-in-python-9fb617a76eac

4. Jaadi, Z. (1 April, 2021). *A Step-by-Step Explanation of Principal Component Analysis (PCA)*. Retrieved from EXPERT CONTRIBUTOR NETWORK: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

5. Saka, A. T. (12 September, 2019). *Practical Guide to Outlier Detection Methods*. Retrieved from Towards Data Science: https://towardsdatascience.com/practical-guide-to-outlier-detection-methods-6b9f947a161e

6. scikit-learn. (n.d.). *https://scikit-learn.org/stable/modules/clustering.html#k-means*. Retrieved from 2.3 Clustering.

7. STATISTICS SOLUTION. (n.d.). *Using Chi-Square Statistic in Research*. Retrieved from Complete Dissertation: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/using-chi-square-statistic-in-research/

8. Wijaya, C. Y. (14 September, 2020). *5 SMOTE Techniques for Oversampling your Imbalance Data*. Retrieved from Towards Data Science: https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5

XI. ACKNOWLEDGEMENTS

I would like to thank the Udacity Support Team, my session advisor, Muru Raj and Xinyi for their unrelenting support in helping me complete my first proper machine learning project with time extension. I also would like to thank my parents for excusing me from household chores for me to successfully finish this project.