MACHINE LEARNING ENGINEER NANODEGREE CAPSTONE REPORT

OUSHNIK DEY

1. DEFINITION

1.1 PROJECT OVERVIEW

Machine learning techniques are widely used these days to target potential customers by companies across all fields. Technique such as clustering has proven to be extremely helpful in this area. Arvato Financial Solutions has paired up with Udacity for their nanodegree program to create a customer segmentation report for them and predict which individuals are most likely to become a customer of their company.

The data that I will use has been provided by Udacity's partners at Bertelsmann Arvato Analytics, and represents a real-life data science task.

There are four data files associated with this project:

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

Each row of the demographics files represents a single person, but also includes information outside of individuals, including information about their household, building, and neighborhood. I will be using the information from the first two files to figure out how customers ("CUSTOMERS") are similar to or differ from the general population at large ("AZDIAS"), then use this analysis to make

predictions on the other two files ("MAILOUT"), predicting which recipients are most likely to become a customer for the mail-order company.

1.2 PROBLEM STATEMENT

I will be analyzing demographics data for customers of a mail-order sales company in Germany, comparing it against demographics information for the general population. I will have to perform customer segmentation, identifying the parts of the population that best describe the core customer base of the company. Then I will have to apply this learning on a third dataset with demographics information for targets of a marketing campaign for the company and use a model to predict which individuals are most likely to convert into becoming customers for the company.

In the first part of the project I will use unsupervised learning technique (k-means clustering) to describe the relationship between the demographics of the company's existing customers and the general population of Germany. This will help us to describe parts of the general population that are more likely to be part of the mail-order company's main customer base, and which parts of the general population are less so.

In the second part of the project I will be using supervised learning techniques to predict which individuals are most likely to become a customer of the company. As a part of this I will try out various classifiers like AdaBoost, Random Forest, Gradient Boosting. I will be using Grid Search to find out the best estimator.

1.3 METRICS

The evaluation metric for this problem that I have chosen is AUC for the ROC curve, relative to the detection of customers from the mail campaign. This is one of the most popular metrics for binary classification problem like the one we have at hand.

2. ANALYSIS

2.1 DATA EXPLORATION AND VISUALIZATION

2.1.1 DIAS Attributes - Values 2017.xlsx

This file provides the information about the features of the demographic datasets provided for this problem. Below is a sample of this file.

In [8]:
```
# Load the feature information
feat_info = pd.read_excel('DIAS Attributes - Values 2017.xlsx')
del feat_info['Unnamed: 0']
feat_info.head(10)
```

Out[8]:

	Attribute	Description	Value	Meaning
0	AGER_TYP	best-ager typology	-1	unknown
1	NaN	NaN	0	no classification possible
2	NaN	NaN	1	passive elderly
3	NaN	NaN	2	cultural elderly
4	NaN	NaN	3	experience-driven elderly
5	ALTERSKATEGORIE_GROB	age classification through prename analysis	-1, 0	unknown
6	NaN	NaN	1	< 30 years
7	NaN	NaN	2	30 - 45 years
8	NaN	NaN	3	46 - 60 years
9	NaN	NaN	4	> 60 years

From the above sample we can see that there are values in the Attribute column are NaN. We will be replacing these null values using pandas forward fill method. We can also see that for a particular attribute there are more than 1 values for which the meaning is unknown or there is no classification possible. We will be creating another dataframe containing the attributes and its corresponding list of values for which the meaning is unknown, or classification is not possible.

2.1.2 Udacity_AZDIAS_052018.csv

This is demographics data for the general population of Germany which has 891211 persons (rows) x 366 features (columns). Below is sample of this dataset.

In [5]:
```
azdias.head()
```

Out[5]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTI
0	910215	-1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	910220	-1	9.0	0.0	NaN	NaN	NaN	NaN	21.0	11.(
2	910225	-1	9.0	17.0	NaN	NaN	NaN	NaN	17.0	10.(
3	910226	2	1.0	13.0	NaN	NaN	NaN	NaN	13.0	1.(
4	910241	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	3.(

5 rows × 366 columns

There are 33492923 values which are naturally missing in this dataset.

In [13]:
```
print('Number of naturally missing values: {}'.format(azdias.isnull().sum().sum()))
```

```
Number of naturally missing values: 33492923
```

Attributes of this dataset will also have values for which classification is not possible or meaning is unknown. I will also be converting these values to NaN before proceeding with our analysis.

There are 94 attributes of this dataset which is not present in the file DIAS Attributes - Values 2017.xlsx. I will be dropping these columns as an explanation regarding the values of these columns cannot be provided when comparing the general population data with the customers data later in the analysis.

In [10]:
```
# Get the column names which are in azdias dataset but not in feat_info
attr_not_in_feat_info = np.setdiff1d(np.array(azdias.columns), np.array(feat_info.Attribute), assume_unique=True)
print(len(attr_not_in_feat_info))
print(attr_not_in_feat_info)
```

```
94
['LNR' 'AKT_DAT_KL' 'ALTER_KIND1' 'ALTER_KIND2' 'ALTER_KIND3' 'ALTER_KIND4'
 'ALTERSKATEGORIE_FEIN' 'ANZ_KINDER' 'ANZ_STATISTISCHE_HAUSHALTE' 'ARBEIT'
 'CAMEO_INTL_2015' 'CJT_KATALOGNUTZER' 'CJT_TYP_1' 'CJT_TYP_2' 'CJT_TYP_3'
 'CJT_TYP_4' 'CJT_TYP_5' 'CJT_TYP_6' 'D19_BANKEN_DIREKT' 'D19_BANKEN_GROSS'
 'D19_BANKEN_LOKAL' 'D19_BANKEN_REST' 'D19_BEKLEIDUNG_GEH'
 'D19_BEKLEIDUNG_REST' 'D19_BILDUNG' 'D19_BIO_OEKO' 'D19_BUCH_CD'
 'D19_DIGIT_SERV' 'D19_DROGERIEARTIKEL' 'D19_ENERGIE' 'D19_FREIZEIT'
 'D19_GARTEN' 'D19_HANDWERK' 'D19_HAUS_DEKO' 'D19_KINDERARTIKEL'
 'D19_KONSUMTYP_MAX' 'D19_KOSMETIK' 'D19_LEBENSMITTEL'
 'D19_LETZTER_KAUF_BRANCHE' 'D19_LOTTO' 'D19_NAHRUNGSERGAENZUNG'
 'D19_RATGEBER' 'D19_REISEN' 'D19_SAMMELARTIKEL' 'D19_SCHUHE'
 'D19_SONSTIGE' 'D19_SOZIALES' 'D19_TECHNIK' 'D19_TELKO_MOBILE'
 'D19_TELKO_ONLINE_QUOTE_12' 'D19_TELKO_REST' 'D19_TIERARTIKEL'
 'D19_VERSAND_REST' 'D19_VERSI_DATUM' 'D19_VERSI_OFFLINE_DATUM'
 'D19_VERSI_ONLINE_DATUM' 'D19_VERSI_ONLINE_QUOTE_12' 'D19_VERSICHERUNGEN'
 'D19_VOLLSORTIMENT' 'D19_WEIN_FEINKOST' 'DSL_FLAG' 'EINGEFUEGT_AM'
 'EINGEZOGENAM_HH_JAHR' 'EXTSEL992' 'FIRMENDICHTE' 'GEMEINDETYP'
 'HH_DELTA_FLAG' 'KBA13_ANTG1' 'KBA13_ANTG2' 'KBA13_ANTG3' 'KBA13_ANTG4'
 'KBA13_BAUMAX' 'KBA13_CCM_1401_2500' 'KBA13_GBZ' 'KBA13_HHZ'
 'KBA13_KMH_210' 'KK_KUNDENTYP' 'KOMBIALTER' 'KONSUMZELLE' 'MOBI_RASTER'
 'RT_KEIN_ANREIZ' 'RT_SCHNAEPPCHEN' 'RT_UEBERGROESSE' 'SOHO_KZ'
 'STRUKTURTYP' 'UMFELD_ALT' 'UMFELD_JUNG' 'UNGLEICHENN_FLAG'
 'VERDICHTUNGSRAUM' 'VHA' 'VHN' 'VK_DHT4A' 'VK_DISTANZ' 'VK_ZG11']
```

I will also be identifying the outlier columns based on the percentage of missing values and dropping them. I will also be dropping rows which have greater than a certain number of missing values.

2.1.3 Udacity_CUSTOMERS_052018.csv

This is demographics data for customers of a mail-order company which has 191652 persons (rows) x 369 features (columns). Below is a sample of the dataset:

```
In [78]: customers.head()
```

Out[78]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTI
0	9626	2	1.0	10.0	NaN	NaN	NaN	NaN	10.0	1.(
1	9628	-1	9.0	11.0	NaN	NaN	NaN	NaN	NaN	Na
2	143872	-1	1.0	6.0	NaN	NaN	NaN	NaN	0.0	1.(
3	143873	1	1.0	8.0	NaN	NaN	NaN	NaN	8.0	0.(
4	143874	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	7.(

5 rows × 369 columns

All the preprocessing steps applied to the Azdias dataset will also be applied to the customers dataset.

2.1.4 Udacity_MAILOUT_052018_TRAIN.csv

This is demographics data for individuals who were targets of a marketing campaign. It has 42962 persons (rows) x 367 (columns). It will be used to train the supervised learning model in the second half of the project.

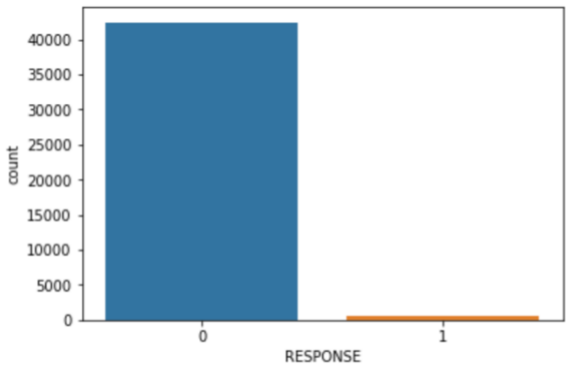
```
In [92]: mailout_train.shape
Out[92]: (42962, 367)

In [93]: mailout_train.head()
```

Out[93]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV
0	1763	2	1.0	8.0	NaN	NaN	NaN	NaN	8.0	15.0
1	1771	1	4.0	13.0	NaN	NaN	NaN	NaN	13.0	1.0
2	1776	1	1.0	9.0	NaN	NaN	NaN	NaN	7.0	0.0
3	1460	2	1.0	6.0	NaN	NaN	NaN	NaN	6.0	4.0
4	1783	2	1.0	9.0	NaN	NaN	NaN	NaN	9.0	53.0

5 rows × 367 columns

```
In [94]: # Get the number of responses for each type of response
         mailout_train['RESPONSE'].value_counts()

Out[94]: 0    42430
         1      532
         Name: RESPONSE, dtype: int64

In [95]: # Distribution for response type
         sns.countplot(mailout_train['RESPONSE'])

Out[95]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc7c79bb780>
```



From the above plot we can see the mailout_train dataset is highly imbalanced. Only 532 people responded to the campaign and 42430 people did not respond.

In this dataset **2217179** values are naturally missing.

```
In [99]: print('Number of naturally missing values: {}'.format(mailout_train.isnull().sum().sum()))
         Number of naturally missing values: 2217179
```

Similar to the general population demographics data and customers data, I will be converting the values for which classification is not possible or meaning is unknow to NaN.

I will also be identifying the outletier columns based on the percentage of missing values and dropping them. I will also be dropping the columns which are not in the attribute information excel file.

2.1.5 Udacity_MAILOUT_052018_TEST.csv

This is demographics data for individuals who were targets of a marketing campaign; 42833 persons (rows) x 366 (columns). It will be used to test the supervised learning model.

```
In [119]: mailout_test.shape
Out[119]: (42833, 366)

In [120]: mailout_test.head()
```

Out[120]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV
0	1754	2	1.0	7.0	NaN	NaN	NaN	NaN	6.0	2.0
1	1770	-1	1.0	0.0	NaN	NaN	NaN	NaN	0.0	20.0
2	1465	2	9.0	16.0	NaN	NaN	NaN	NaN	11.0	2.0
3	1470	-1	7.0	0.0	NaN	NaN	NaN	NaN	0.0	1.0
4	1478	1	1.0	21.0	NaN	NaN	NaN	NaN	13.0	1.0

5 rows × 366 columns

Same preprocessing steps will be applied to this dataset which will be applied to Udacity_MAILOUT_052018_TRAIN.csv

2.2 ALGORITHMS AND TECHNIQUES

I will use unsupervised learning technique (k-means clustering) to describe the relationship between the demographics of the company's existing customers and the general population of Germany. Clustering is the process of partitioning or grouping a given set of data into disjoint clusters. K-means clustering algorithm is one of the partitioning based algorithms where the objective is to obtain a fixed number of clusters which minimizes the sum of squared Euclidean distance between the objects and the centroids of the clusters [1]. The Euclidean distance between points p and q is the length of the line segment connecting them. In Cartesian coordinates, if p = (p1, p2,..., pn) and q = (q1, q2,..., qn) are two points in Euclidean n-space, then the distance (d) from p to q, or from q to p is given by the Pythagorean formula [2]:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

I will start with the preprocessing step where I will be assessing missing data across rows and columns. I will be converting the missing values in the Azdias dataset to NaN. I will be identifying the outlier columns based on the

percentage of missing and dropping those columns. I will be also be identifying rows having missing values greater than a threshold value and dropping these rows from our analysis. This threshold value will be decided after I have converted the missing values to NaN and dropped the outlier columns.

In the second part of the preprocessing step I will be re-encoding categorical and mixed features. The binary categorical features which do not have numerical values will be encoded as 0 and 1. The ones having numerical values will be left untouched. The multi-level categorical columns will be encoded using OneHot Encoding. OneHot Encoding is used to convert it into a non-ordinal numerical representation for use in machine learning algorithms. [3] This is an important step because k-means uses Euclidean distance to determine the centroids of the clusters and Euclidean distance between two points having categorical values does not make sense. [4]

I will then be imputing the dataset with the most frequent value strategy as clustering methods cannot analyze items that have missing values. Next I will be scaling the data using Standard Scaler. The dataset should be scaled because the standard deviation of the features of the dataset are quite different.

```
In [7]: azdias.describe()
```

Out[7]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_
count	8.912210e+05	891221.000000	817722.000000	817722.000000	81058.000000	29499.000000	6170.000000	1205.000000	628274.000000	
mean	6.372630e+05	-0.358435	4.421928	10.864126	11.745392	13.402658	14.476013	15.089627	13.700717	
std	2.572735e+05	1.198724	3.638805	7.639683	4.097660	3.243300	2.712427	2.452932	5.079849	
min	1.916530e+05	-1.000000	1.000000	0.000000	2.000000	2.000000	4.000000	7.000000	0.000000	
25%	4.144580e+05	-1.000000	1.000000	0.000000	8.000000	11.000000	13.000000	14.000000	11.000000	
50%	6.372630e+05	-1.000000	3.000000	13.000000	12.000000	14.000000	15.000000	15.000000	14.000000	
75%	8.600680e+05	-1.000000	9.000000	17.000000	15.000000	16.000000	17.000000	17.000000	17.000000	
max	1.082873e+06	3.000000	9.000000	21.000000	18.000000	18.000000	18.000000	18.000000	25.000000	

K-means treats the data space as isotropic which means that it tends to produce clusters which are nearly round rather than elongated. Now if the features of the dataset have unequal variances, more weight will be assigned to the features with small variances. This will lead to clusters being separated along features with greater variance. [5]

Next I will be performing dimensionality reduction using PCA (Principal Component Analysis). PCA is one of the most popular dimension reduction methods. It finds the best combination of the original features so that the various across the new variable is maximum. [6]

Then I will be applying k-means clustering on the PCA transformed data to find out which parts of the general population are more likely to be part of the mail order company's main customer base.

In the second half of the project I will be building a prediction model to decide whether or not it will be worth to include that person in the campaign. For this

we have been provided with the train and the test dataset. Since, the train dataset is labelled I will be using Supervised Learning to build the prediction model. I will be trying out classifiers like Logistic Regression, Random Forest, AdaBoost and Gradient Boosting. I have chosen these algorithms keeping in mind that the problem in hand is a binary classification problem.

a. **Logistic Regression**: It is one of the most popular algorithms for binary classification problems. The input values are combined linearly using weights or co-efficient values to predict an output value. This output value is a probability which can be converted to binary value (0 or 1) based on a step function.
Logit function is the logarithms of the odds ratio. It can be used to express the linear relationships between feature values and the log odds:

$$\text{logit}(P(y=1|x)) = W_0X_0 + \ldots + W_mW_m = \text{Sum}(W_iX_i) = W^TX$$

P(y=1|x) is the conditional probability that a particular sample belongs to class 1 given its features x.

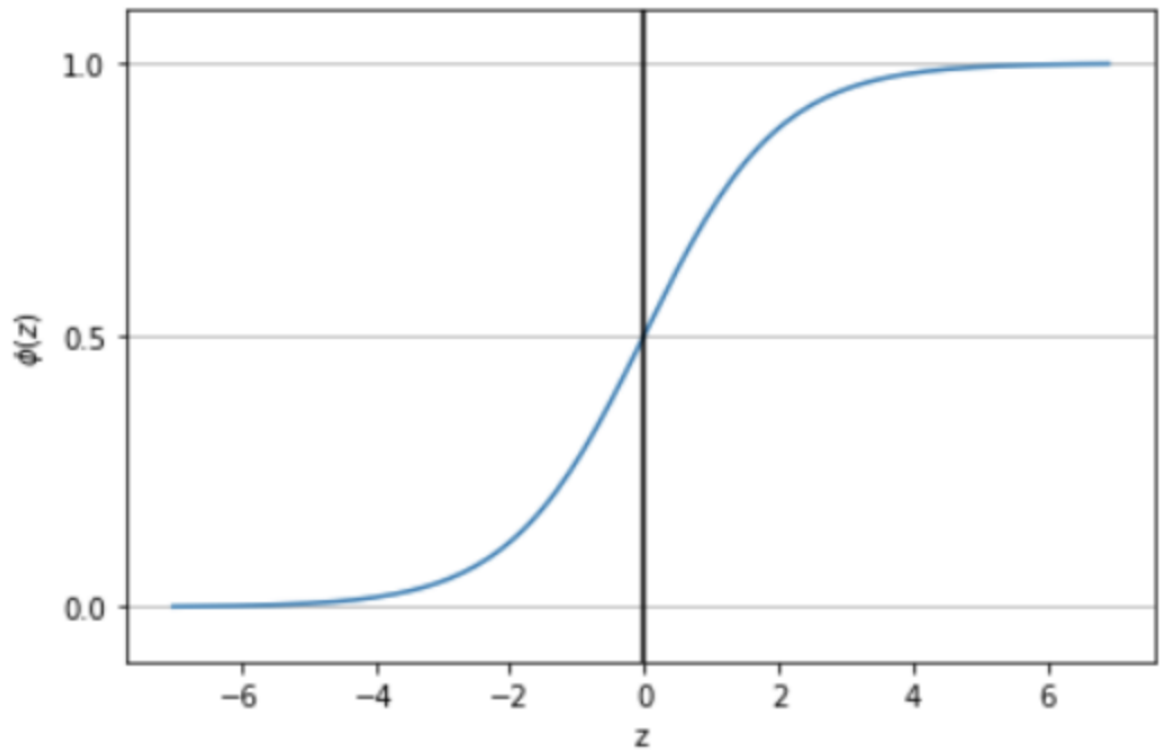
We need to predict the probability that a sample belongs to a certain class. This is called the sigmoid function which is the inverse of logit function. The formula for sigmoid function is as follows:

$$\phi(z) = \frac{1}{1+e^{-z}}$$

where *z* can be calculated as follows:

$$z = w_Tx = w_0 + w_1x_1 + \ldots\ldots + w_mx_m$$

The graphical representation of this is as follows:



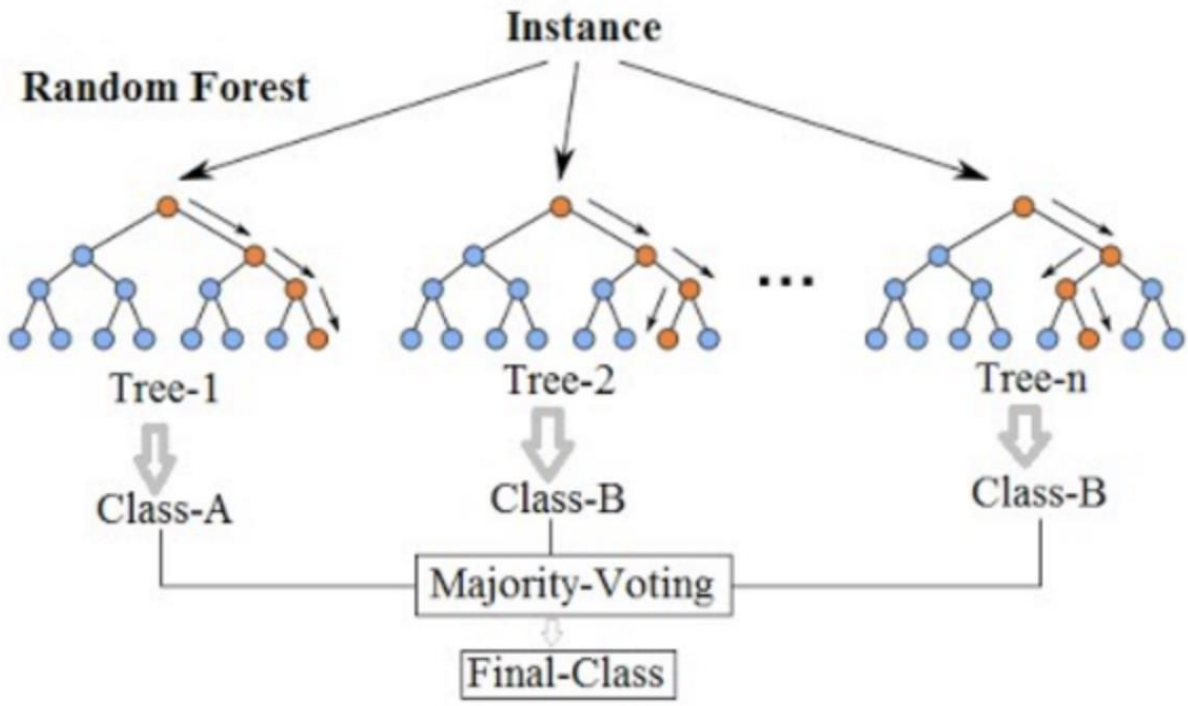
From the graph we can see that the sigmoid function approaches to 1 if z tends to infinity and to 0 if z approaches to minus infinity. So, this function takes the values and converts them to the range [0, 1]. Now this predicted value can be converted to a binary value using a step function as follows: [7]

$$\hat{y} = \begin{cases} 1 & if\ \phi(z) \geq 0.5 \\ 0 & otherwise \end{cases}$$

b. **Random Forest**: Random Forest is a supervised learning algorithm which can be used for classification and regression problems, It is a decision tree algorithm which works by selecting columns randomly and making decision trees for each batch of columns. This solves the problem of overfitting which generally occurs when applying decision trees on datasets having large number of features. The idea here is to combine a number of weak learners to build a stronger and more robust learner which will finally be used to predict classes. This is how Random Forest works:
    - Select random samples from a given dataset.

- Construct a decision tree for each sample and get a prediction from each decision tree(weak learners)
- Perform a vote for each predicted result.
- Select the prediction with the most votes as the final result.

Here is a pictorial representation of Random Forest algorithm: [7]



c. **AdaBoost**: AdaBoost is an ensemble boosting classifier. It is a supervised learning algorithm. It works by combining a number of weak classifiers (weak learners) to build a strong classifier (strong learner) to get better predictions from this strong classifier. Any machine learning algorithm can be used as the weak classifier. These are the steps involved:
   - It initially selects a training subset randomly.
   - Iteratively trains the AdaBoost machine learning model by selecting the training set based on the prediction of the last training.
   - Higher weights are assigned to the wrongly classified points so that in the next iteration these observations will get the higher probability for classification.
   - This continues until there is no error or the maximum number of estimators have been reached.

d. **Gradient Boosting**: Gradient Boosting is a machine learning technique which is used for regression and classification problems. Like AdaBoost or any other boosting algorithm, Gradient Boosting combines a number of weak learners to build a strong learner which is then used to make predictions. It depends on a loss function which should be differentiable.

It has two parts: weak learner and an additive component. It uses decision trees as the weak learners. The additive component of a gradient boosting model comes from the fact that the trees are added to the model over time and when this occurs the existing trees are not manipulated.
A process similar to gradient descent is used to minimize the error between the given parameters. This is done by taking the calculated loss and performing gradient descent to reduce that loss. Then the parameters of the trees are modified to reduce the residual loss. [8]

Then I will be using GridSearch to come up with the best estimator and tune its hyperparameters to improve the performance of that classifier.

2.3 BENCHMARK

Since this is a part of Kaggle competition, the benchmarking will be done based on the highest score (0.80819) for this competition.

3. METHODOLOGY

3.1 DATA PREPROCESSING

3.1.1 DIAS Attributes - Values 2017.xlsx

As discussed earlier we will be replacing null values in the Attribute column using pandas forward fill method.

In [9]:
```
feat_info['Attribute'] = feat_info['Attribute'].fillna(method='ffill')
feat_info.head(10)
```

Out[9]:

	Attribute	Description	Value	Meaning
0	AGER_TYP	best-ager typology	-1	unknown
1	AGER_TYP	NaN	0	no classification possible
2	AGER_TYP	NaN	1	passive elderly
3	AGER_TYP	NaN	2	cultural elderly
4	AGER_TYP	NaN	3	experience-driven elderly
5	ALTERSKATEGORIE_GROB	age classification through prename analysis	-1, 0	unknown
6	ALTERSKATEGORIE_GROB	NaN	1	< 30 years
7	ALTERSKATEGORIE_GROB	NaN	2	30 - 45 years
8	ALTERSKATEGORIE_GROB	NaN	3	46 - 60 years
9	ALTERSKATEGORIE_GROB	NaN	4	> 60 years

Another dataframe containing the attributes and its corresponding list of values for which the meaning is unknown, or classification is not possible was created.

In [11]:
```
feat_info_missing = feat_info[(feat_info["Meaning"].str.contains("unknown") |
                               feat_info["Meaning"].str.contains("no "))]
feat_info_missing.head(20)
```

Out[11]:

	Attribute	Description	Value	Meaning
0	AGER_TYP	best-ager typology	-1	unknown
1	AGER_TYP	NaN	0	no classification possible
5	ALTERSKATEGORIE_GROB	age classification through prename analysis	-1, 0	unknown
11	ALTER_HH	main age within the household	0	unknown / no main age detectable
33	ANREDE_KZ	gender	-1, 0	unknown
40	BALLRAUM	distance to next urban centre	-1	unknown
48	BIP_FLAG	business-flag indicating companies in the buil...	-1	unknown
49	BIP_FLAG	NaN	0	no company in the building
51	CAMEO_DEUG_2015	CAMEO classification 2015 - Uppergroup	-1	unknown
105	CAMEO_DEUINTL_2015	CAMEO classification 2015 - international typo...	-1	unknown
131	CJT_GESAMTTYP	customer journey typology	0	unknown
138	D19_BANKEN_ANZ_12	transaction activity BANKS in the last 12 months	0	no transactions known
145	D19_BANKEN_ANZ_24	transaction activity BANKS in the last 24 months	0	no transactions known

In [12]:
```
feat_info_missing = feat_info_missing['Value'].groupby([feat_info_missing.Attribute]).apply(list).reset_index()
feat_info_missing.rename(columns={'Value': 'missing_or_unknown'}, inplace=True)
feat_info_missing
```

Out[12]:

	Attribute	missing_or_unknown
0	AGER_TYP	[-1, 0]
1	ALTERSKATEGORIE_GROB	[-1, 0]
2	ALTER_HH	[0]
3	ANREDE_KZ	[-1, 0]
4	BALLRAUM	[-1]
5	BIP_FLAG	[-1, 0]
6	CAMEO_DEUG_2015	[-1]
7	CAMEO_DEUINTL_2015	[-1]
8	CJT_GESAMTTYP	[0]
9	D19_BANKEN_ANZ_12	[0]
10	D19_BANKEN_ANZ_24	[0]
11	D19_BANKEN_DATUM	[10]

3.1.2 Udacity_AZDIAS_052018.csv

Firstly, I have converted the values for which the meaning is unknown, or classification is not possible to NaN. After the conversion the number of missing values in the Azdias dataset increased to **54942758.**

In [14]:
```
%%time
# Identify missing or unknown data values and convert them to NaNs.

for idx in feat_info_missing.index:
    column_name = feat_info_missing.iloc[idx]['Attribute']
    print(column_name)
    missing_or_unknown = feat_info_missing.iloc[idx]['missing_or_unknown']
    if column_name in azdias.columns:
        if isinstance(missing_or_unknown[0], str):
            missing_or_unknown = [int(value)
                                  for value in feat_info_missing.iloc[idx]['missing_or_unknown'][0].split(',')]
        azdias[column_name] = azdias[column_name].replace(missing_or_unknown, np.nan)
```

In [15]:
```
print('Number of missing values after conversion: {}'.format(azdias.isnull().sum().sum()))
```

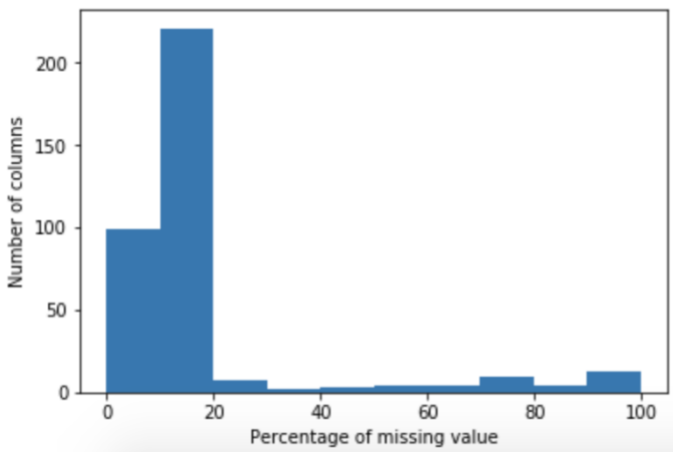
```
Number of missing values after conversion: 54942758
```

Then the outlier columns were detected and dropped based on the percentage of missing values per columns.

In [16]:
```
# Find out number of missing data there is in each column of the dataset.
missing_values_per_column = azdias.isnull().sum()
print(missing_values_per_column)
```

```
LNR                         0
AGER_TYP                685843
AKT_DAT_KL               73499
ALTER_HH                310267
ALTER_KIND1             810163
ALTER_KIND2             861722
ALTER_KIND3             885051
ALTER_KIND4             890016
ALTERSKATEGORIE_FEIN    262947
ANZ_HAUSHALTE_AKTIV      93148
ANZ_HH_TITEL             97008
ANZ_KINDER               73499
```

In [17]:
```
# Investigate pattern in the percentage of missing data in each column
plt.hist(missing_values_per_column/azdias.shape[0] * 100)
plt.xlabel('Percentage of missing value')
plt.ylabel('Number of columns')
plt.show()
```



From the above histogram we can see that columns with more than 20% of missing values are outliers. Hence, I have dropped these columns. There were a total of 46 columns having more than 20% missing values.

```
In [18]: # Finding the outlier columns from the dataset.
outlier_column_names = missing_values_per_column[missing_values_per_column/azdias.shape[0] > 0.2].index.tolist()
print(outlier_column_names)
print(len(outlier_column_names))
```

```
['AGER_TYP', 'ALTER_HH', 'ALTER_KIND1', 'ALTER_KIND2', 'ALTER_KIND3', 'ALTER_KIND4', 'ALTERSKATEGORIE_FEIN', 'D19_BAN
KEN_ANZ_12', 'D19_BANKEN_ANZ_24', 'D19_BANKEN_DATUM', 'D19_BANKEN_OFFLINE_DATUM', 'D19_BANKEN_ONLINE_DATUM', 'D19_BAN
KEN_ONLINE_QUOTE_12', 'D19_GESAMT_ANZ_12', 'D19_GESAMT_ANZ_24', 'D19_GESAMT_DATUM', 'D19_GESAMT_OFFLINE_DATUM', 'D19_
GESAMT_ONLINE_DATUM', 'D19_GESAMT_ONLINE_QUOTE_12', 'D19_KONSUMTYP', 'D19_LETZTER_KAUF_BRANCHE', 'D19_LOTTO', 'D19_SO
ZIALES', 'D19_TELKO_ANZ_12', 'D19_TELKO_ANZ_24', 'D19_TELKO_DATUM', 'D19_TELKO_OFFLINE_DATUM', 'D19_TELKO_ONLINE_DATU
M', 'D19_TELKO_ONLINE_QUOTE_12', 'D19_VERSAND_ANZ_12', 'D19_VERSAND_ANZ_24', 'D19_VERSAND_DATUM', 'D19_VERSAND_OFFLIN
E_DATUM', 'D19_VERSAND_ONLINE_DATUM', 'D19_VERSAND_ONLINE_QUOTE_12', 'D19_VERSI_ANZ_12', 'D19_VERSI_ANZ_24', 'D19_VER
SI_ONLINE_QUOTE_12', 'EXTSEL992', 'KBA05_ANTG1', 'KBA05_ANTG2', 'KBA05_ANTG3', 'KBA05_ANTG4', 'KBA05_BAUMAX', 'KK_KUN
DENTYP', 'TITEL_KZ']
46
```

```
In [19]: # Drop the outlier columns from the azdias dataframe
azdias.drop(outlier_column_names, axis=1, inplace=True)
```

I have also dropped the LNR columns since it is a unique id assigned to each individual.

```
In [20]: # Drop LNR column as it is a unique id assigned to each individual
azdias.drop('LNR', axis=1, inplace=True)
```

After dropping the outlier columns and LNR column, the total number of columns reduced to 319.

```
In [21]: # Shape of azdias dataframe after dropping outlier columns and LNR column
azdias.shape
```

```
Out[21]: (891221, 319)
```

Post this I have found out the column names which are in the Azdias dataset but not in the feature information file. I have decided to drop these columns as it would not be possible to provide an explanation about the effects of these columns on the result without the description of these columns and their range of values. As discussed earlier, there were 94 such columns.

```
In [22]: # Drop the columns from azdias dataset which are not in feat_info
columns_to_drop = set(attr_not_in_feat_info.tolist()) - set(outlier_column_names) - {'LNR'}
azdias.drop(list(columns_to_drop), axis=1, inplace=True)
```

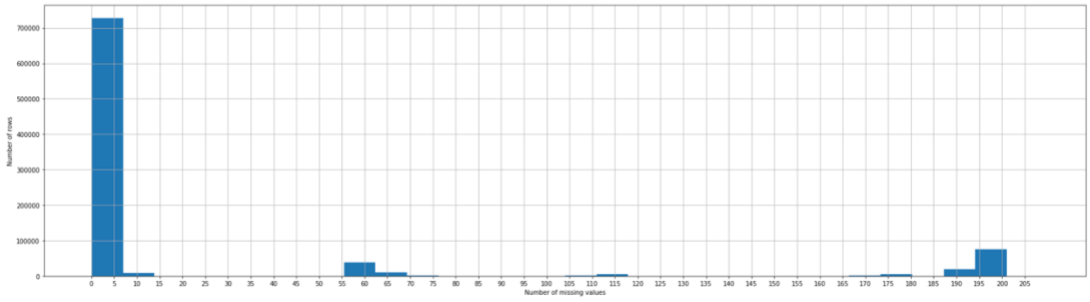
After dropping these, the number of columns reduced to 238.

```
In [23]: azdias.shape
Out[23]: (891221, 238)
```

Next, I have assessed the missing data in each row and tried to find divide the dataset into two groups based on the number of missing values per row.

```
In [24]: # Number of missing values in each row of azdias dataframe
         missing_values_per_row = azdias.isnull().sum(axis=1)

In [25]: missing_values_per_row.value_counts().sort_index()
Out[25]: 0       614103
         1        32760
         2        36149
         3        13715
         4        12117
         5         9871
         6         9520
         7         3412
         8          706
         9          289
         10        3879
         11         111
         12         259
```

```
In [26]: plt.figure(figsize=(30,8))
         missing_values_per_row.hist(bins=30)
         plt.xlabel('Number of missing values')
         plt.ylabel('Number of rows')
         plt.xticks(np.arange(0, 210, step=5))
         plt.show()
```



From the above plot we could see that most of the rows have 7 or less missing values. Hence, going forward I will be dropping the rows having more than 7 missing values.

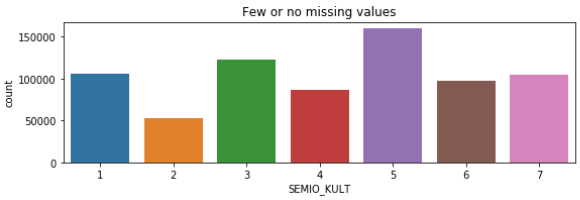
```
In [27]: # Dividing the azdias dataframe into subsets containing <=7 missing values per row
         # and >7 missing values per row respectively
         azdias_missing_values_lte_7 = azdias[missing_values_per_row <= 7]
         azdias_missing_values_gt_7 = azdias[missing_values_per_row > 7]
```

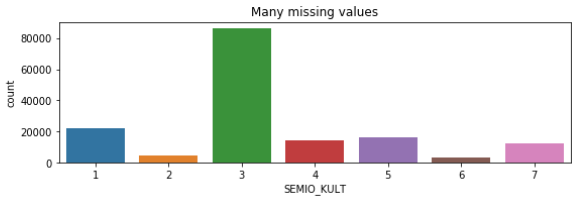
```
In [28]: print("Percentage of rows where missing less than or equal to 7 vales are missing: {}".format(
             len(azdias_missing_values_lte_7)/azdias.shape[0]*100))
         print("Percentage of rows where missing more than 7 vales are missing: {}".format(
             len(azdias_missing_values_gt_7)/azdias.shape[0]*100))
```

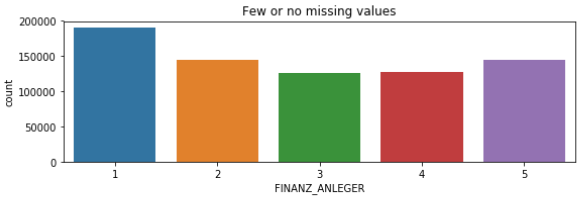
```
Percentage of rows where missing less than or equal to 7 vales are missing: 82.09490126466949
Percentage of rows where missing more than 7 vales are missing: 17.90509873533052
```

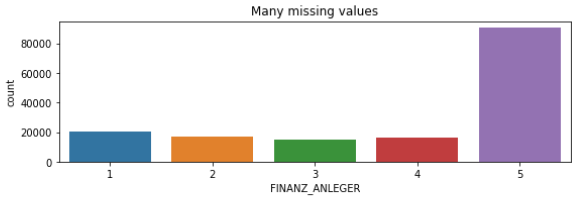
From the above analysis we could see that 82% percentage of the rows had less than or equal to 7 missing values. Remaining 18% of the rows have greater than 7 missing values.

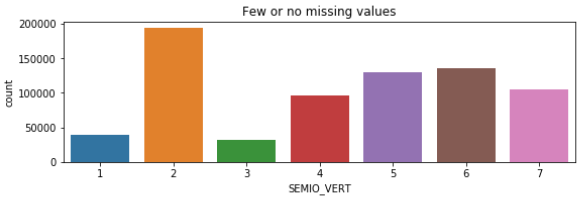
I have also compared the distribution of five randomly selected columns which have <=3 missing values in the two subsets **azdias_missing_values_lte_7** and **azdias_missing_values_lte_7.**

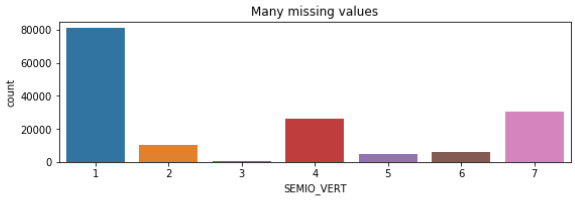


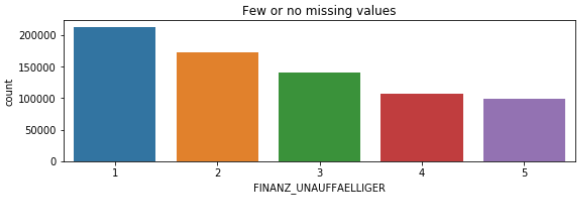


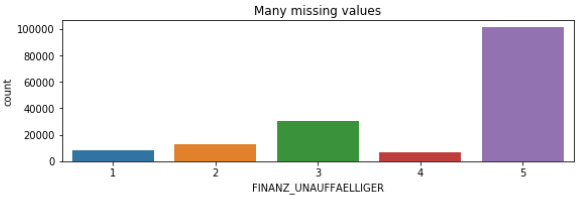


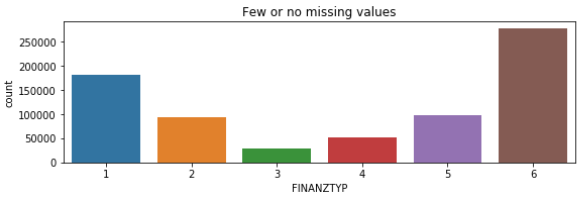


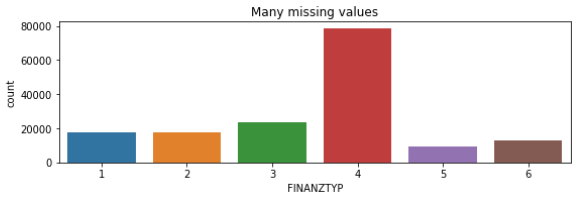












We could see from the above plots that the all the randomly selected columns have different distribution in the two subsets.

Then, I have dropped the rows having >7 missing values. After dropping these rows, the number of rows reduced from **891221** to **731647**.

```
In [32]: # Drop rows with >7 missing values in the azdias dataset.
         print('Total number of rows in azdias dataset: {}'.format(azdias.shape[0]))

         azdias = azdias[azdias.index.isin(azdias_missing_values_lte_7.index)]

         print('Number of rows after dropping rows with more than 7 missing values : {}'.format(azdias.shape[0]))
```

```
Total number of rows in azdias dataset: 891221
Number of rows after dropping rows with more than 7 missing values : 731647
```

```
In [33]: azdias.head()
```

Out[33]:

	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_PERSONEN	ANZ_TITEL	BALLRAUM	CAMEO_DEU_2015	CAMEO_DEUG_2015	CJT_GESAMTTYP	EWDICHTE
1	11.0	0.0	2.0	0.0	6.0	8A	8	5.0	3.0
2	10.0	0.0	1.0	0.0	2.0	4C	4	3.0	4.0
3	1.0	0.0	0.0	0.0	4.0	2A	2	2.0	2.0
4	3.0	0.0	4.0	0.0	2.0	6B	6	5.0	5.0
5	5.0	0.0	1.0	0.0	6.0	8C	8	2.0	2.0

5 rows × 238 columns

To avoid memory issue down the line I have taken a sample of the Azdias dataset to proceed forward. I have taken 20% of the data from each of the group of rows having less than or equal to 7 missing values.

```
In [34]: # Number of missing values in azdias after removing rows with more than 7 missing values
         missing_values_per_row_latest = azdias.isnull().sum(axis=1)
```

```
In [35]: missing_values_per_row_latest.value_counts().sort_index()
```

```
Out[35]: 0    614103
         1     32760
         2     36149
         3     13715
         4     12117
         5      9871
         6      9520
         7      3412
         dtype: int64
```

```
In [36]: # Take a sample of the azdias dataset
         azdias_sample = pd.concat([azdias[missing_values_per_row_latest == missing_count].sample(frac=0.2, random_state=42)
                                    for missing_count in range(0,8)])
```

```
In [37]: azdias_sample.shape
```

```
Out[37]: (146329, 238)
```

The sample dataframe has 146329 rows and 238 columns.

Since, no information was provided about the type of columns, I have manually gone through the file **DIAS Attributes - Values 2017.xlsx** and selected the categorical columns.

In [38]:
```
categorical_columns = ['ALTERSKATEGORIE_GROB', 'ALTER_HH', 'ANREDE_KZ', 'BALLRAUM', 'BIP_FLAG',
                       'CAMEO_DEUG_2015', 'CAMEO_DEU_2015', 'CJT_GESAMTTYP', 'D19_BANKEN_ANZ_12',
                       'D19_BANKEN_ANZ_24', 'D19_BANKEN_DATUM', 'D19_BANKEN_DIREKT_RZ',
                       'D19_BANKEN_GROSS_RZ', 'D19_BANKEN_LOKAL_RZ', 'D19_BANKEN_OFFLINE_DATUM',
                       'D19_BANKEN_ONLINE_DATUM', 'D19_BANKEN_ONLINE_QUOTE_12', 'D19_BANKEN_REST_RZ',
                       'D19_BEKLEIDUNG_GEH_RZ', 'D19_BEKLEIDUNG_REST_RZ', 'D19_BILDUNG_RZ',
                       'D19_BIO_OEKO_RZ', 'D19_BUCH_RZ', 'D19_DIGIT_SERV_RZ', 'D19_DROGERIEARTIKEL_RZ',
                       'D19_ENERGIE_RZ', 'D19_FREIZEIT_RZ', 'D19_GARTEN_RZ', 'D19_GESAMT_ANZ_12',
                       'D19_GESAMT_ANZ_24', 'D19_GESAMT_DATUM', 'D19_GESAMT_OFFLINE_DATUM',
                       'D19_GESAMT_ONLINE_DATUM', 'D19_GESAMT_ONLINE_QUOTE_12', 'D19_HANDWERK_RZ',
                       'D19_HAUS_DEKO_RZ', 'D19_KINDERARTIKEL_RZ', 'D19_KONSUMTYP', 'D19_KK_KUNDENTYP',
                       'D19_KOSMETIK_RZ', 'D19_LEBENSMITTEL_RZ', 'D19_LOTTO_RZ', 'D19_NAHRUNGSERGAENZUNG_RZ',
                       'D19_RATGEBER_RZ', 'D19_REISEN_RZ', 'D19_SAMMELARTIKEL_RZ', 'D19_SCHUHE_RZ',
                       'D19_SONSTIGE_RZ', 'D19_TECHNIK_RZ', 'D19_TELKO_ANZ_12', 'D19_TELKO_ANZ_24',
                       'D19_TELKO_DATUM', 'D19_TELKO_MOBILE_RZ', 'D19_TELKO_OFFLINE_DATUM',
                       'D19_TELKO_ONLINE_DATUM', 'D19_TELKO_REST_RZ', 'D19_TIERARTIKEL_RZ',
                       'D19_VERSAND_ANZ_12', 'D19_VERSAND_ANZ_24', 'D19_VERSAND_DATUM',
                       'D19_VERSAND_OFFLINE_DATUM', 'D19_VERSAND_ONLINE_DATUM', 'D19_VERSAND_ONLINE_QUOTE_12',
                       'D19_VERSAND_REST_RZ', 'D19_VERSICHERUNGEN_RZ', 'D19_VERSI_ANZ_12', 'D19_VERSI_ANZ_24',
                       'D19_VOLLSORTIMENT_RZ', 'D19_WEIN_FEINKOST_RZ',
                       'EWDICHTE', 'FINANZTYP', 'GREEN_AVANTGARDE', 'GEBAEUDETYP',
                       'GEBAEUDETYP_RASTER', 'GFK_URLAUBERTYP', 'HAUSHALTSSTRUKTUR', 'HEALTH_TYP',
                       'INNENSTADT', 'KBA05_HERSTTEMP', 'KBA05_MODTEMP', 'KONSUMNAEHE', 'LP_FAMILIE_FEIN',
                       'LP_STATUS_FEIN', 'LP_STATUS_GROB', 'NATIONALITAET_KZ', 'OST_WEST_KZ', 'SHOPPER_TYP', 'VERS_TYP'
                       'WOHNLAGE', 'ZABEOTYP']
```

I have divided the categorical columns into two subsets: binary level categorical columns (having <=2 unique values) and multi-level categorical columns (having >2 unique values).

In [39]:
```
# Get the binary level and multi-level categorical columns
binary_categoricals_columns = []
multi_level_categorical_columns = []

for column in categorical_columns:
    if column in azdias_sample.columns:
        if len(azdias_sample[column].dropna().unique()) > 2:
            multi_level_categorical_columns.append(column)
        else:
            binary_categoricals_columns.append(column)
print('Binary categorical columns are: {}'.format(binary_categoricals_columns))
print('Multi-level categorical columns are: {}'.format(multi_level_categorical_columns))
```

```
Binary categorical columns are: ['ANREDE_KZ', 'GREEN_AVANTGARDE', 'OST_WEST_KZ', 'VERS_TYP']
Multi-level categorical columns are: ['ALTERSKATEGORIE_GROB', 'BALLRAUM', 'CAMEO_DEUG_2015', 'CAMEO_DEU_2015', 'CJT_G
ESAMTTYP', 'EWDICHTE', 'FINANZTYP', 'GEBAEUDETYP', 'GEBAEUDETYP_RASTER', 'GFK_URLAUBERTYP', 'HEALTH_TYP', 'INNENSTAD
T', 'KBA05_HERSTTEMP', 'KBA05_MODTEMP', 'KONSUMNAEHE', 'LP_FAMILIE_FEIN', 'LP_STATUS_FEIN', 'LP_STATUS_GROB', 'NATION
ALITAET_KZ', 'SHOPPER_TYP', 'WOHNLAGE', 'ZABEOTYP']
```

Out of the four binary categorical columns only OST_WEST_KZ do not have numerical values. Hence, we will only be re-encoding this column with 1 and 0.

In [40]:
```
azdias_sample['OST_WEST_KZ'].replace(['W','O'], [1,0], inplace=True)
```

In [41]:
```
for column in binary_categoricals_columns:
    print(azdias_sample[column].value_counts())
```

```
2    76428
1    69901
Name: ANREDE_KZ, dtype: int64
0    113550
1     32779
Name: GREEN_AVANTGARDE, dtype: int64
1    115569
0     30760
Name: OST_WEST_KZ, dtype: int64
2.0    73813
1.0    65926
Name: VERS_TYP, dtype: int64
```

I have encoded the multi-level categorical columns using OneHot Encoding.

```
In [42]: # OneHot Encoding of multi level categorical columns
         azdias_sample = pd.get_dummies(azdias_sample, columns=multi_level_categorical_columns,
                                        prefix=multi_level_categorical_columns)
```

After OneHot Encoding of the multi-level categorical columns, the number of features of the azdias dataset increased to 409.

```
In [43]: azdias_sample.shape
Out[43]: (146329, 409)
```

In the azdias_sample dataframe there was only one mixed-type feature: **PRAEGENDE_JUGENDJAHRE**. I have created two new columns **PRAEGENDE_JUGENDJAHRE_DECADE** and **PRAEGENDE_JUGENDJAHRE_MOVEMENT** from **PRAEGENDE_JUGENDJAHRE** by mapping the values with the Attribute Values file.

The column **PRAEGENDE_JUGENDJAHRE_DECADE** was created using the below mapping function:

```
In [45]: decade_mapping_dict = {0: [1, 2], 1: [3, 4], 2: [5, 6, 7], 3: [8, 9], 4: [10, 11, 12, 13], 5:[14, 15]}

         def map_decade(x):
             try:
                 for key, value in decade_mapping_dict.items():
                     if x in value:
                         return key
             except ValueError:
                 return np.nan
```

```
In [46]: azdias_sample['PRAEGENDE_JUGENDJAHRE_DECADE'] = azdias_sample['PRAEGENDE_JUGENDJAHRE'].apply(map_decade)
```

The column **PRAEGENDE_JUGENDJAHRE_MOVEMENT** was created using the below mapping function:

```
In [47]: mainstream_mapping = [1, 3, 5, 8, 10, 12, 14]

         def map_movement(x):
             try:
                 if x in mainstream_mapping:
                     return 0
                 else:
                     return 1
             except ValueError:
                 return np.nan
```

```
In [48]: azdias_sample['PRAEGENDE_JUGENDJAHRE_MOVEMENT'] = azdias_sample['PRAEGENDE_JUGENDJAHRE'].apply(map_movement)
```

Then the original column **PRAEGENDE_JUGENDJAHRE** was dropped.

```
In [49]: azdias_sample = azdias_sample.drop(['PRAEGENDE_JUGENDJAHRE'], axis=1)
```

I have then imputed the azdias_sample dataframe to replace the NaN values with the **most_frequent** strategy.

```
In [54]: # Impute NaN with 'most_frequent' strategy
         fill_missing = Imputer(strategy='most_frequent')
         azdias_sample_imputed = pd.DataFrame(fill_missing.fit_transform(azdias_sample))

In [55]: azdias_sample_imputed.columns = azdias_sample.columns
         azdias_sample_imputed.index = azdias_sample.index
```

Then I have scaled the imputed dataframe using StandardScaler.

```
In [57]: # Apply standard scaler to the imputed dataset
         scaler = StandardScaler()
         azdias_sample_scaled = scaler.fit_transform(azdias_sample_imputed)

In [58]: azdias_sample_scaled = pd.DataFrame(azdias_sample_scaled, columns=azdias_sample_imputed.columns)

In [59]: azdias_sample_scaled.head()
```

Out[59]:

	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_PERSONEN	ANZ_TITEL	FINANZ_ANLEGER	FINANZ_HAUSBAUER	FINANZ_MINIMALIST	FINANZ_SPARER	FI
0	-0.476087	-0.128365	1.936369	-0.058917	1.457283	0.597156	-0.752829	1.526478	
1	-0.210364	-0.128365	-0.632809	-0.058917	1.457283	0.597156	-1.476790	0.856819	
2	-0.143933	-0.128365	-0.632809	-0.058917	-0.575659	-0.126702	-0.028869	0.187161	
3	0.121790	-0.128365	-0.632809	-0.058917	-0.575659	-0.850560	-0.028869	0.187161	
4	-0.343225	-0.128365	0.223584	-0.058917	0.101988	-0.850560	-0.028869	-0.482498	

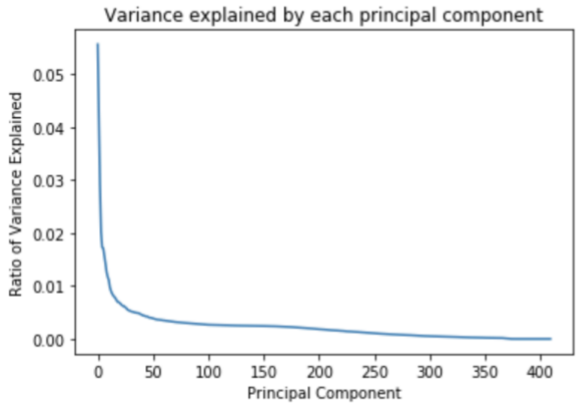
5 rows × 410 columns

Next I have performed dimensionality reduction on the scaled dataframe using PCA.

```
In [60]: pca = PCA()
         pca.fit(azdias_sample_scaled)
```

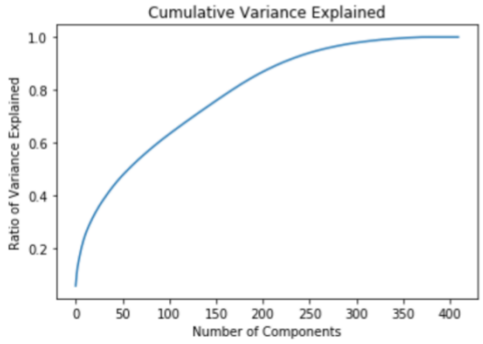
Out[60]:
```
PCA(copy=True, iterated_power='auto', n_components=None, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

```
In [61]: # Investigate the variance accounted for by each principal component.
         plt.plot(range(len(pca.explained_variance_ratio_)), pca.explained_variance_ratio_)
         plt.title("Variance explained by each principal component")
         plt.xlabel("Principal Component")
         plt.ylabel("Ratio of Variance Explained")
         plt.show()
```



The above plot shows the variance explained by each principal component.

In [62]:
```
plt.plot(range(len(pca.explained_variance_ratio_)),np.cumsum(pca.explained_variance_ratio_), '-')
plt.title("Cumulative Variance Explained")
plt.xlabel("Number of Components")
plt.ylabel("Ratio of Variance Explained")
plt.show()
```



This plot depicts the cumulative variance explained by the components. We can see that the first 200 principal components explain more that 80% of the variance. Hence, going forward I will be using the first 200 components.

In [63]:
```
# Re-apply PCA to the data by selecting first 200 components to retain.
pca_200 = PCA(n_components=200)
azdias_sample_pca = pca_200.fit_transform(azdias_sample_scaled)
```

Interpreting the first few principal components:

In [68]:
```
print("Top 10 positives for component 0")
pca_weight_0.head(10)
```

```
Top 10 positives for component 0
```

Out[68]:

HH_EINKOMMEN_SCORE	0.150794
LP_STATUS_GROB_1.0	0.143744
PLZ8_ANTG3	0.141806
PLZ8_BAUMAX	0.138273
PLZ8_ANTG4	0.137580
KBA13_HALTER_30	0.113666
FINANZ_HAUSBAUER	0.111365
EWDICHTE_6.0	0.104439
ORTSGR_KLS9	0.103173
KBA13_ALTERHALTER_30	0.098765

In [69]:
```
print("Top 10 negatives for component 0")
pca_weight_0.tail(10)
```

```
Top 10 negatives for component 0
```

Out[69]:

GREEN_AVANTGARDE	-0.096766
KBA05_ANHANG	-0.100155
KBA13_SEG_GELAENDEWAGEN	-0.105817
PLZ8_GBZ	-0.120426
FINANZ_MINIMALIST	-0.121853
KBA13_AUTOQUOTE	-0.126916
KBA05_AUTOQUOT	-0.133679
KBA05_GBZ	-0.133748
PLZ8_ANTG1	-0.147583
MOBI_REGIO	-0.150565

From component 0 we can see that **HH_EINKOMMEN_SCORE** and **MOBI_REGIO** are negatively co-related. This means that the moving pattern of the families is inversely proportional to the estimated net household income. We can also see that **PLZ8_ANTG3** and **PLZ8_ANTG4** have positive correlation which means that number of 6-10 family houses and number of 10+ family houses in the PLZ8 region increases together.

In [70]:
```
print("Top 10 positives for component 1")
pca_weight_1.head(10)
```

```
Top 10 positives for component 1
```

Out[70]:

KBA13_HERST_BMW_BENZ	0.172023
KBA13_SEG_OBEREMITTELKLASSE	0.151299
KBA13_BMW	0.146759
KBA13_MERCEDES	0.144161
KBA13_SITZE_4	0.140762
ORTSGR_KLS9	0.137747
KBA13_SEG_SPORTWAGEN	0.133133
KBA13_SEG_OBERKLASSE	0.119848
KBA13_KMH_211	0.114497
KBA05_HERST1	0.113420

In [71]:
```
print("Top 10 negatives for component 1")
pca_weight_1.tail(10)
```

```
Top 10 negatives for component 1
```

Out[71]:
```
KBA13_HERST_FORD_OPEL   -0.104972
KBA13_HALTER_25         -0.105721
KBA13_AUTOQUOTE         -0.106657
KBA13_SEG_KLEINWAGEN    -0.116896
KBA13_HALTER_50         -0.121042
KBA13_KMH_140_210       -0.122144
KBA13_HALTER_55         -0.122649
KBA13_HALTER_20         -0.130724
KBA13_SITZE_5           -0.136530
KBA13_ALTERHALTER_60    -0.137811
```

From component 1 we can see that the columns **KBA13_SEG_SPORTWAGEN** and **KBA13_SEG_OBERKLASSE** are positively co-related which indicates that the people who have upper class cars also have sports cars.

In [72]:
```
print("Top 10 positives for component 2")
pca_weight_2.head(10)
```

```
Top 10 positives for component 2
```

Out[72]:
```
FINANZ_VORSORGER          0.205847
ALTERSKATEGORIE_GROB_4    0.184661
ZABEOTYP_3                0.179142
SEMIO_ERL                 0.167410
SEMIO_LUST                0.158406
RETOURTYP_BK_S            0.142434
FINANZ_MINIMALIST         0.120630
CJT_GESAMTTYP_2.0         0.104282
W_KEIT_KIND_HH            0.094203
FINANZTYP_2               0.091893
```

```
In [73]: print("Top 10 negatives for component 2")
         pca_weight_2.tail(10)

         Top 10 negatives for component 2

Out[73]: FINANZTYP_1                      -0.148924
         SEMIO_KULT                       -0.154474
         SEMIO_RAT                        -0.170201
         FINANZ_ANLEGER                   -0.198809
         SEMIO_TRADV                      -0.199459
         FINANZ_UNAUFFAELLIGER            -0.206760
         SEMIO_REL                        -0.210979
         SEMIO_PFLICHT                    -0.212891
         PRAEGENDE_JUGENDJAHRE_DECADE     -0.230107
         FINANZ_SPARER                    -0.231108
```

From component 2 we can see that the columns **ALTERSKATEGORIE_GROB** and **FINANZ_VORSORGER** have positive correlation. This means that as a person becomes older, he/she is more prepared financially. We can also see that the columns **ALTERSKATEGORIE_GROB** and **PRAEGENDE_JUGENDJAHRE_DECADE** have negative correlation. This means that as **PRAEGENDE_JUGENDJAHRE_DECADE** increases the age decreases.

3.1.3 Udacity_CUSTOMERS_052018.csv

The customers dataset was preprocessed using the exact same steps followed in the preprocessing of Azdias dataset. The dataset was imputed, scaled and then dimensionality reduction was performed on it using PCA. Similar to the Azdias dataset, I have considered the first 200 principal components.

3.1.4 Udacity_MAILOUT_052018_TRAIN.csv

Similar to the general population demographics data, I have written a function to cleaning function to perform preprocessing and feature engineering on the mailout train dataset. After the number of columns increased to 413.

```
In [106]: mailout_clean.shape
Out[106]: (42962, 413)
```

I have then dropped the response column and imputed the dataset using ‘median’ strategy. The imputed dataset was then scaled using Standard Scaler.

3.1.5 Udacity_MAILOUT_052018_TEST.csv

The same cleaning function from the previous step was applied to the test dataset. After that the cleaned dataset was imputed and scaled.

3.2 IMPLEMENTATION

3.2.1 Unsupervised Learning

After the Azdias (general population) dataset was preprocessed, k-means clustering was applied to it. I have applied k-means clustering to the general population data using k=1 to k=25, where k=number of clusters. Then I have plotted a graph **k vs SSE** to predict the elbow point.

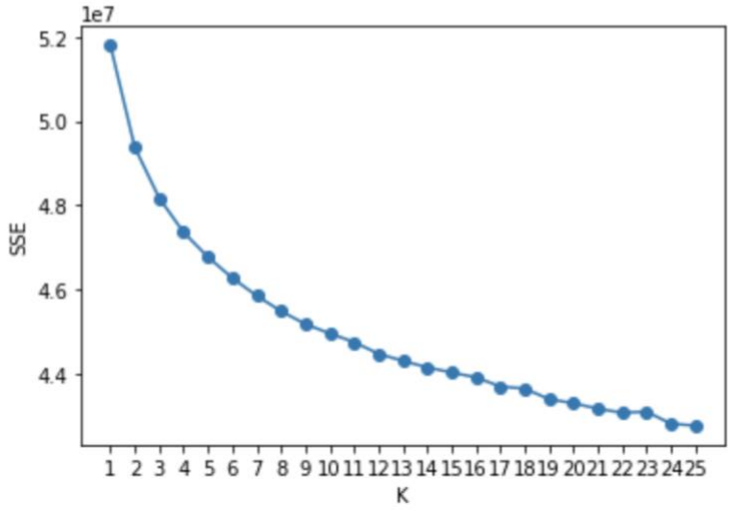
In [74]:
```
def perform_kmeans(data, n_cluster):
    kmeans = KMeans(n_clusters = n_cluster)
    model = kmeans.fit(data)
    score = np.abs(model.score(data))
    return score
```

In [75]:
```
scores = []
number_of_clusters = list(range(1,26))
for k in number_of_clusters:
    print("Performing KMeans for k={}".format(k))
    scores.append(perform_kmeans(azdias_sample_pca, k))
```

```
Performing KMeans for k=1
Performing KMeans for k=2
Performing KMeans for k=3
Performing KMeans for k=4
Performing KMeans for k=5
Performing KMeans for k=6
Performing KMeans for k=7
Performing KMeans for k=8
Performing KMeans for k=9
Performing KMeans for k=10
Performing KMeans for k=11
Performing KMeans for k=12
Performing KMeans for k=13
Performing KMeans for k=14
Performing KMeans for k=15
```

```
In [76]: plt.plot(number_of_clusters, scores, linestyle='-', marker='o')
         plt.xticks(number_of_clusters)
         plt.xlabel('K')
         plt.ylabel('SSE')
```

```
Out[76]: Text(0,0.5,'SSE')
```



From the above plot it looks like 15 is the elbow point. Hence, we will be going forward with **k=15**. I have then re-fit the k-means models with 15 clusters and obtained the cluster predictions for the general population demographics data.

```
In [75]: # Re-fit the k-means model with 15 clusters and obtain cluster predictions for the general population demographics data
         kmeans = KMeans(n_clusters=15)
         model_15 = kmeans.fit(azdias_sample_pca)
         azdias_sample_pred = model_15.predict(azdias_sample_pca)
```

I have used the same k-means model to obtain the clusters for the customers data. Later I have compared the clusters of the general population and customers data which is provided in the next section of this document.

3.2.2 Unsupervised learning

In this section I have used four different classifiers: Logistic Regression, Random Forest, AdaBoost and Gradient Boosting. I have trained the mailout_train dataset using these four different classifiers.

In [115]:
```
classifier_names = []
classifier_scores = []
classifier_best_estimators = []
classifiers_time_taken = []
classifiers = [logistic_regression, random_forest, ada_boost, gradient_boosting]

for classifier in classifiers:
    best_score, best_estimator, time_taken = fit_classifier(classifier, {}, mailout_features_imputed_scaled,
                                                            mailout_response)
    classifier_names.append(classifier.__class__.__name__)
    classifier_scores.append(best_score)
    classifier_best_estimators.append(best_estimator)
    classifiers_time_taken.append(time_taken)
```

```
Classifier: LogisticRegression
Time taken in seconds : 195.1445095539093
Best score : 0.5843367744664455
---------------------------------------------------------------------------------------------------
Classifier: RandomForestClassifier
Time taken in seconds : 12.033421754837036
Best score : 0.48384133347778
---------------------------------------------------------------------------------------------------
Classifier: AdaBoostClassifier
Time taken in seconds : 89.1865701675415
Best score : 0.5788606432962725
---------------------------------------------------------------------------------------------------
Classifier: GradientBoostingClassifier
Time taken in seconds : 317.96957516670227
Best score : 0.5945723442335306
---------------------------------------------------------------------------------------------------
```

We can see here that the Gradient Boosting classfier produced the best score (roc_auc). I will be tuning this classifier using param grid in GridSearchCV to come up with the best estimator.

3.3 REFINEMENT

To improve the performance of the GradientBoosting classifier model, I have tuned the hyperparameters using GridSearch to come up with the best estimator. After tuning, the training score improved from 0.5945 to 0.6151.

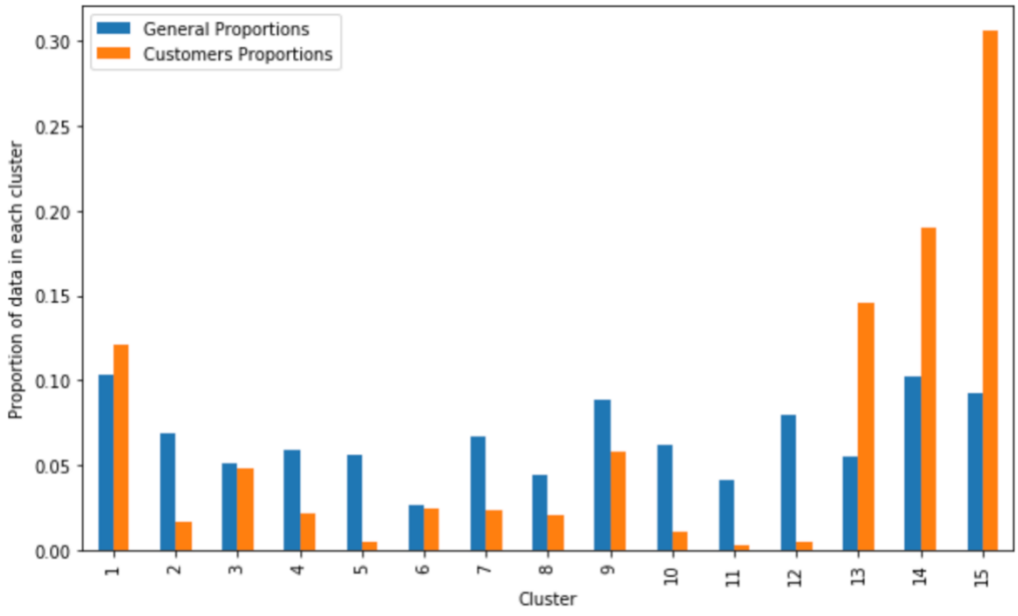
In [116]:
```
# Tune the best classifier(GradientBoostingClassifier) with the help of param grid in GridSearchCV

param_grid = {'loss': ['exponential'],
              'learning_rate': [0.01],
              'n_estimators': [100],
              'max_depth': [6, 8]
             }

gradient_boosting = GradientBoostingClassifier(random_state=42)
gradient_boosting_best_score, gradient_boosting_best_est, time_taken = fit_classifier(gradient_boosting, param_grid,
                                                                                      mailout_features_imputed_scaled,
                                                                                      mailout_response)
```

```
Classifier: GradientBoostingClassifier
Time taken in seconds : 3691.3175506591797
Best score : 0.6151378596718878
```

4. RESULTS

4.1.1 Unsupervised Learning

After applying k-means clustering on the general population demographics data and customers data I have made comparisons of the cluster in which the customers data is overrepresented and the cluster in which the general populations data is overrepresented.



Cluster 15 is overrepresented in the Customer Data compared to Demographics Data. Some of the characteristics of this cluster are as follows:

- We can also see that people in this cluster are not financial minimalist.
- People of this cluster have high interest in making investments and saving money.
- They have high interest in their own house.

Hence, this segment of the population should be relatively popular with the mail order company.

Cluster 12 is overrepresented in the Customer Data compared to Demographics Data. Some of the characteristics of this cluster are as follows:

- We can also see that people in this cluster are financial minimalist.
- People of this cluster have low interest in making investments and saving money.

- They have low interest in their own house.

Hence, this segment of the population should be relatively unpopular with the mail order company.

4.1.2 Supervised Learning

The Gradient Boosting classifier model was used to predict the responses of test dataset Udacity_MAILOUT_052018_TEST.csv. The predictions were then dumped in a csv file uploaded to the Kaggle Competition. I could achieve a score of 0.62992 on Kaggle.

5. IMPROVEMENTS

- There were 94 columns which were present in the demographics data but in the feature information file. I have decided to drop these columns because it would not have been possible to explain the effect of these columns on the results. Dropping these columns surely resulted in a huge amount of information loss. I feel if some of those columns could be included in my analysis, I could have achieved a better score.
- Information regarding the type of columns were provided. I had to choose the categorical columns by manually going through the feature information file. It is possible that there were few misclassifications in doing that.

6. References

[1] https://www.ijert.org/research/analysis-and-study-of-k-means-clustering-algorithm-IJERTV2IS70648.pdf

[2] https://en.wikipedia.org/wiki/Euclidean_distance

[3] https://bambielli.com/til/2018-02-11-one-hot-encoding/

[4] https://www.quora.com/Why-does-K-means-clustering-perform-poorly-on-categorical-data-The-weakness-of-the-K-means-method-is-that-it-is-applicable-only-when-the-mean-is-defined-one-needs-to-specify-K-in-advance-and-it-is-unable-to-handle-noisy-data-and-outliers

[5] https://stats.stackexchange.com/questions/21222/are-mean-normalization-and-feature-scaling-needed-for-k-means-clustering/21226#21226

[6] https://blog.paperspace.com/dimension-reduction-with-principal-component-analysis/

[7] https://towardsdatascience.com/supervised-learning-basics-of-classification-and-main-algorithms-c16b06806cd3

[8] https://en.wikipedia.org/wiki/Gradient_boosting