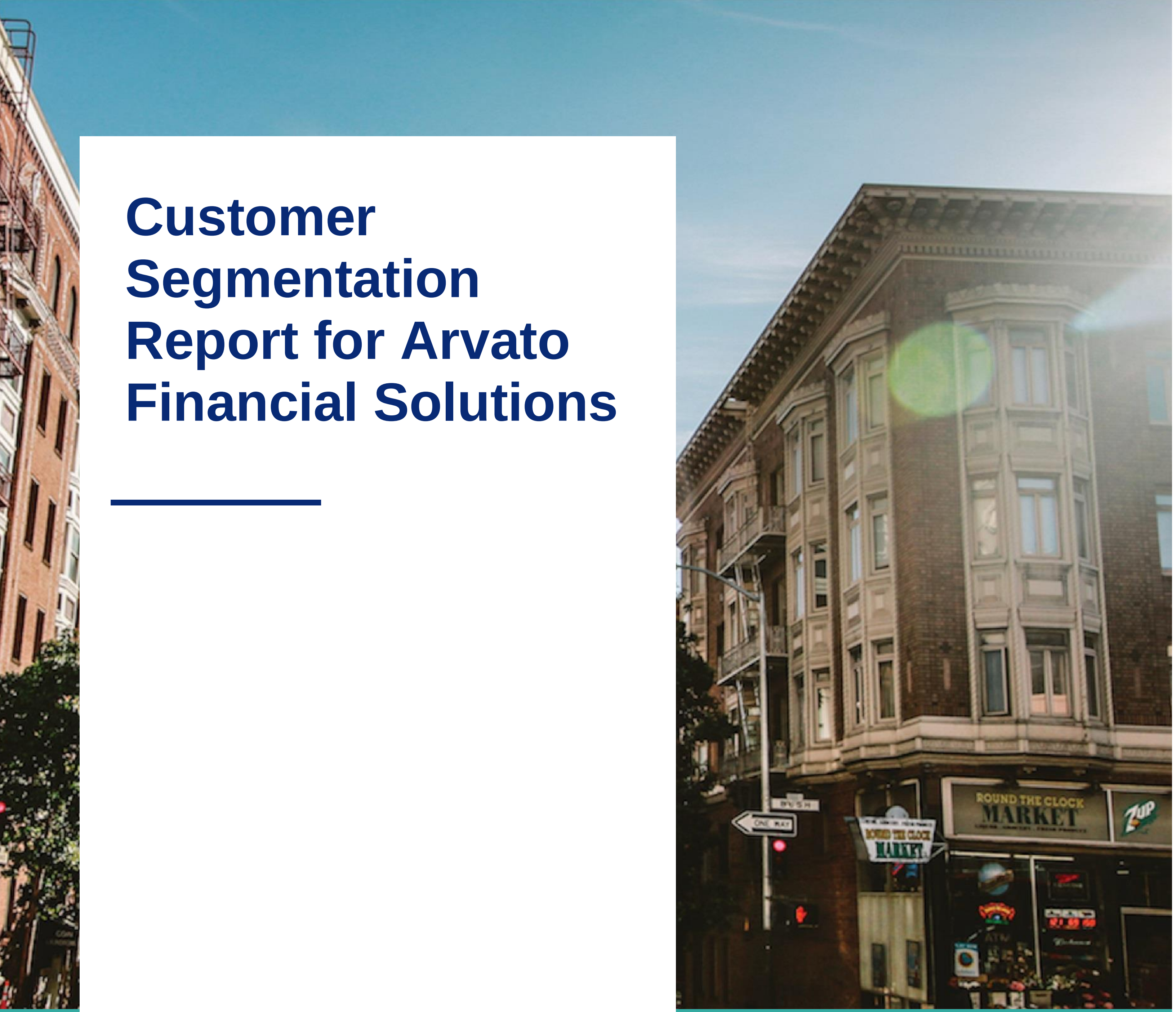


Customer Segmentation Report for Arvato Financial Solutions

JUNE 22

Udacity ML Engineer Nanodegree Course
Authored by: Ruoyu Zhao

Table of Contents

Definition

Project Overview:

Arvato Financial Solutions is a German Company that offers big data solutions, IT, analytics and financial services to give their customers the best possible platform for growth. In this business case, Arvato is looking to help their customer, A mail-order company identify potential customers out of the general population (Germany). This will allow the mail-order company with marketing for their mail-order sales to grow.
In this project, I will analyze demographics data for the company's existing customers and compare the features to the general population in Germany. This will help to understand what demographic attributes are mostly seen in their customer pool.
Additionally, Arvato would like to have a predictive model that can help their customer to target which individuals out of the general population that are most likely to response to their campaigns.
Those following datasets are given by the mail-order company . All tables required for this project are provided by the Udacity Machine Learning Engineer course. Due to the sensitivity nature of the demographics data, the data cannot be found in the Kaggle Competition page.

Udacity_CUSTOMERS_052018.csv

- Size: 191,652 rows and 369 columns
- Content: This table contains demographics data for all prior customers of mail-order company.

Udacity_AZDIAS_052018.csv

- Size: 891,211 rows and 366 columns
- Content: This table contains demographics data for general population of Germany Only.

Udacity_Mailout_052018_train.csv

- Size: 42,982 rows and 367 columns
- Content: This table contains demographics data for people who previously were targeted for the marketing campaign.

Udacity_Mailout_052018_test.csv

- Size: 42,833 rows and 366 columns
- Content: This table contains demographics data for people who previously were on the marketing campaign just like the training data to be predicted on.

DIAS Information Levels - Attributes 2017.xlsx

- Content: a top-level list of attributes and descriptions, organized by informational category

DIAS Attributes - Values 2017.xlsx

- Content: a detailed mapping of data values for each feature in alphabetical order

Project Problem:

The project intends to answer a business question: "If we know the demographic attributes of the whole German population, how can the mail-order company acquire new customer more efficiently?"

There are two parts of answering this business questions:

1. We need to use unsupervised learning techniques to identify demographics segments in both Germany general population and mail-order company's customer population. This will help to understand what demographics attributes that are most significant in the customer population.

2. Build a supervised learning model to predict a given individual is likely to become a customer based on the individual's demographic attributes.

Evaluation Metrics:

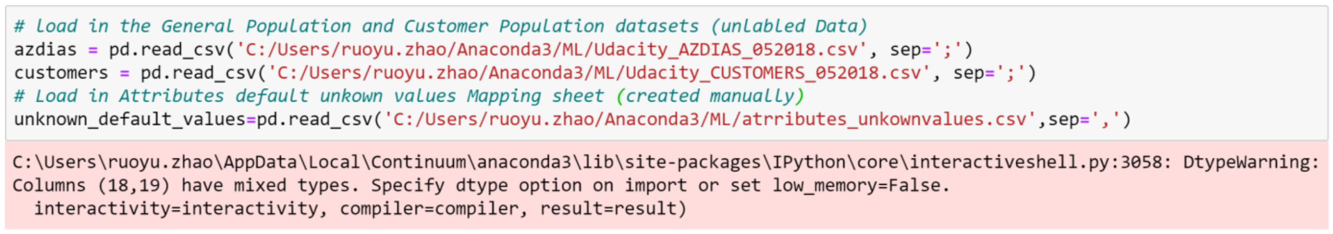
The evaluation metrics for the supervised learning model will be AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve. In this project we are facing a multi-classification problem and AUC-ROC is one of the most popular and important metrics for checking a classification model's performance. This also the evaluation metric required by the Kaggle competition.
ROC is a probability curve which is plotted with True Positive Rate on the y-axis and False Positive Rate on the x-axis. AUC is the area under the ROC probability curve, it shows the degrees of separability of a model. In another word, it tells us the capability of a model to distinguish the data between different classes. The higher the AUC the better (max = 1 and min = 0).
Particularly in this project, the training data has an extreme class imbalance issues, machine learning model will be more inclined to predict toward the majority class. Since AUC ROC will take account of both true positive and false positive, a good AUC ROC score means the model can accurately predict both classes.

Analysis

Data Exploration and Preprocessing:

Dealing with Mixed Data Type Columns

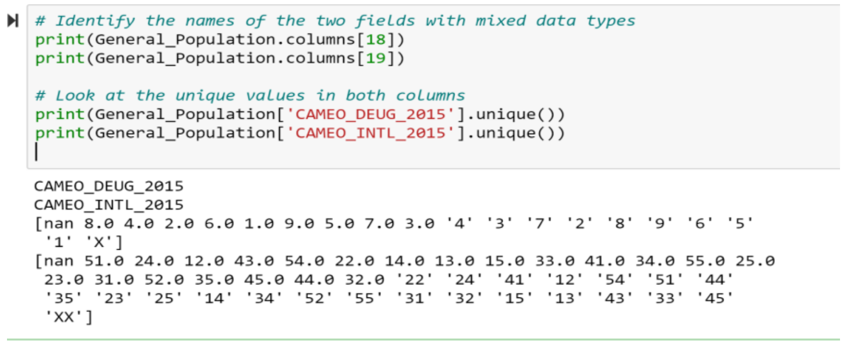
I start with loading the unlabelled data sets (AZDIAS_052018.csv and Customer_052018.csv), and there is an error warning shown, as per below:



```
# Load in the General Population and Customer Population datasets (unlabled Data)
azdias = pd.read_csv('C:/Users/ruoyu.zhao/Anaconda3/ML/Udacity_AZDIAS_052018.csv', sep=';')
customers = pd.read_csv('C:/Users/ruoyu.zhao/Anaconda3/ML/Udacity_CUSTOMERS_052018.csv', sep=';')
# Load in Attributes default unkown values Mapping sheet (created manually)
unknown_default_values=pd.read_csv('C:/Users/ruoyu.zhao/Anaconda3/ML/atrributes_unkownvalues.csv',sep=',')
```

```
C:\Users\ruoyu.zhao\AppData\Local\Continuum\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3058: DtypeWarning:
Columns (18,19) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

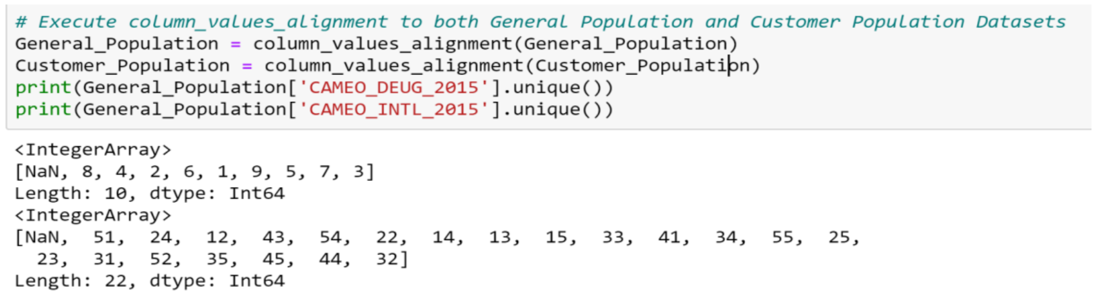
This is due to two columns having mixed types ('CAMEO_DEUG_2015' and 'CAMEO_DEUG_2015'), as can be seen below, the two columns contain mixed of numeric and string values.



```
# Identify the names of the two fields with mixed data types
print(General_Population.columns[18])
print(General_Population.columns[19])

# Look at the unique values in both columns
print(General_Population['CAMEO_DEUG_2015'].unique())
print(General_Population['CAMEO_INTL_2015'].unique())
```

```
CAMEO_DEUG_2015
CAMEO_INTL_2015
[nan 8.0 4.0 2.0 6.0 1.0 9.0 5.0 7.0 3.0 '4' '3' '7' '2' '8' '9' '6' '5'
 '1' 'X']
[nan 51.0 24.0 12.0 43.0 54.0 22.0 14.0 13.0 15.0 33.0 41.0 34.0 55.0 25.0
 23.0 31.0 52.0 35.0 45.0 44.0 32.0 '22' '24' '41' '12' '54' '51' '44'
 '35' '23' '25' '14' '34' '52' '55' '31' '32' '15' '13' '43' '33' '45'
 'XX']
```

I have also identified values such as 5.0 and '5' should be the same value, therefore we need to align them. Therefore, those two columns would need an additional data cleansing. As can be seen, the values in those two fields are much cleaner and converted into integer after.



```
# Execute column_values_alignment to both General Population and Customer Population Datasets
General_Population = column_values_alignment(General_Population)
Customer_Population = column_values_alignment(Customer_Population)
print(General_Population['CAMEO_DEUG_2015'].unique())
print(General_Population['CAMEO_INTL_2015'].unique())
```

```
<IntegerArray>
[NaN, 8, 4, 2, 6, 1, 9, 5, 7, 3]
Length: 10, dtype: Int64
<IntegerArray>
[NaN,  51,  24,  12,  43,  54,  22,  14,  13,  15,  33,  41,  34,  55,  25,
  23,  31,  52,  35,  45,  44,  32]
Length: 22, dtype: Int64
```

Dealing inconsistent Column Names

By observing the column names, it appears that many of the columns in the "***DIAS Attributes - Values 2017.xlsx***" has a prefix "_RZ" whereas the columns in AZDIAS_052018.csv and Customer_052018.csv data sets don't have the prefix. I will need to rename them so they are consistent, this will increase the number of "in common" columns between the "XX_052018" data sets and the DIAS Attributes - Values 2017.xlsx.

Column Commonality between Population Data and Attributes List

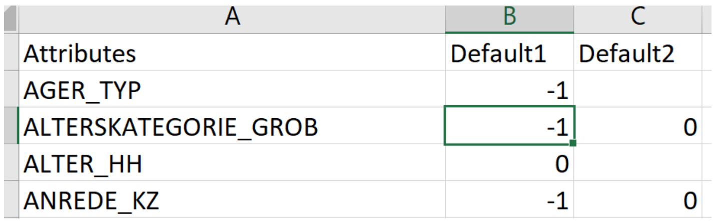
Based on the analysis:

1. There are 272 features in common between general/customer population and attribute information lists.

2. There are 3 columns only in the customer population and they are not relevant to the studies (need to be dropped).
That means the remaining fields in the AZDIAS_052018.csv and Customer_052018.csv (excluding “LNR” which is the identifier) don’t have description in the DIAS Attribute information lists. I will need to apply common sense and guess the data type of those columns.

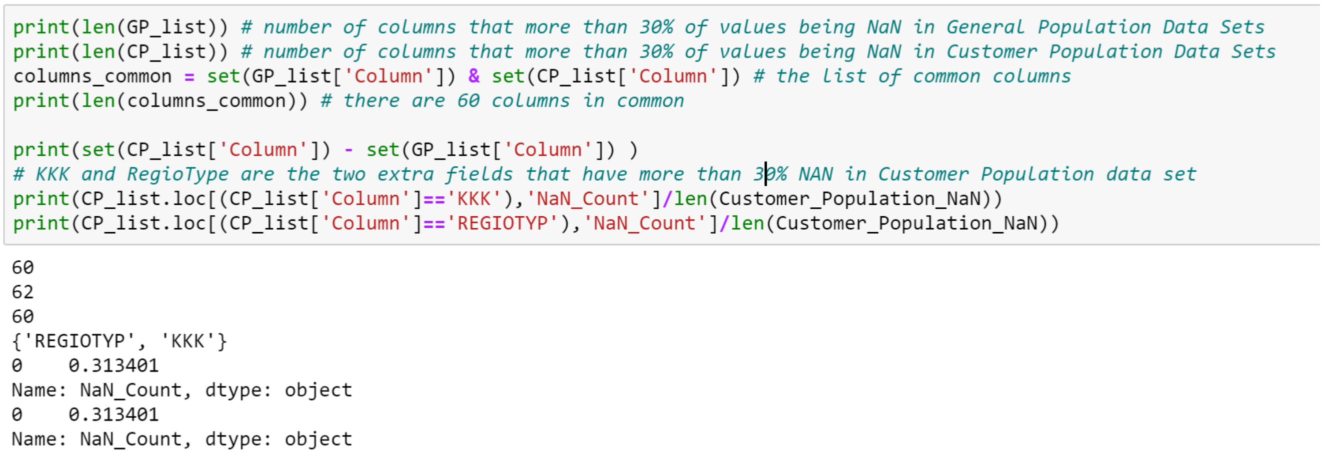
Dealing with Default NaN (unknown) values in Columns

By observing the data and looking at the attribute information sheets. Unfortunately, not all unknown values are being put as NaN in the General and Customer population datasets. Many of them are put as (-1,0,9) as default unknown values. Therefore, if we don't convert those default values into NaN, we are underestimating the number of NaN values in each column and will miss out droping columns where there are too many NaN. Therefore, I have created an excel sheet based on the ***DIAS Attributes - Values 2017.xlsx*** to map the default unknown value for each column. (a selection of the data below) (the data is imported and named as “unknown_default_values”)



A	B	C
Attributes	Default1	Default2
AGER_TYP	-1	
ALTERSKATEGORIE_GROB	-1	0
ALTER_HH	0	
ANREDE_KZ	-1	0

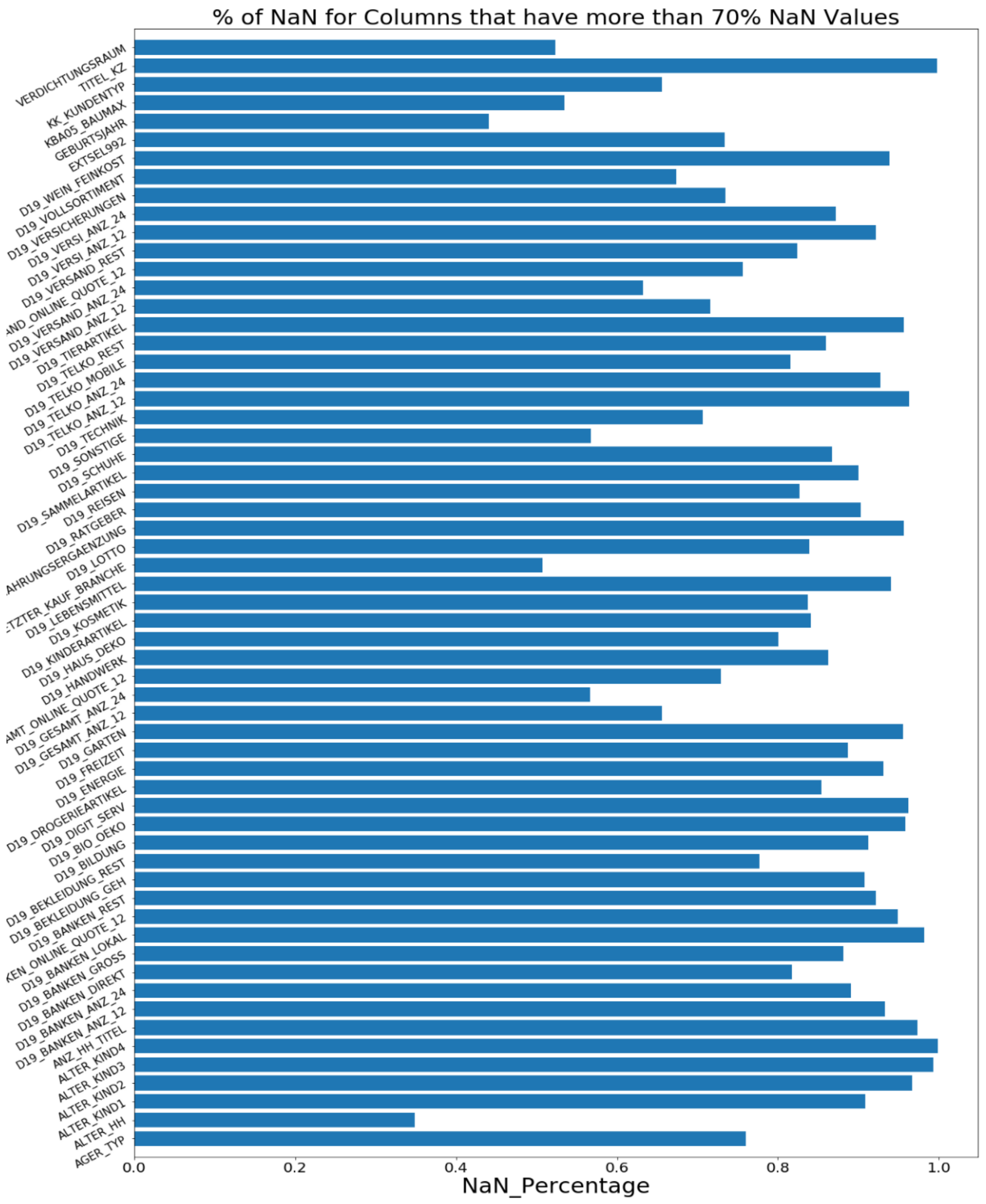
After converting unknown values into NaN values, we need to understand how many NaN values in each column and what are the columns in the General and Population Data Sets that have more 30% of the values being NaN. As a general practice, I have picked a threshold of 30% to drop columns. We also need to check if these dropped columns are the same among the Customer and General population data sets.
As you can see below, there are 60 columns in General Population dataset and 62 Columns in Customer Population dataset that are above this threshold. The two extra columns ‘KKK’ and “RegioType” in Customer Data set which only have 31.3% of data, which is slight higher than 30% thresholds, which I believe it is reasonable to keep since in General Population the dataset has less than 30% of the values being NaN.



```
print(len(GP_list)) # number of columns that more than 30% of values being NaN in General Population Data Sets
print(len(CP_list)) # number of columns that more than 30% of values being NaN in Customer Population Data Sets
columns_common = set(GP_list['Column']) & set(CP_list['Column']) # the list of common columns
print(len(columns_common)) # there are 60 columns in common

print(set(CP_list['Column']) - set(GP_list['Column']) )
# KKK and RegioType are the two extra fields that have more than 30% NAN in Customer Population data set
print(CP_list.loc[(CP_list['Column']=='KKK'),'NaN_Count']/len(Customer_Population_NaN))
print(CP_list.loc[(CP_list['Column']=='REGIOTYP'),'NaN_Count']/len(Customer_Population_NaN))
```

```
60
62
60
{'REGIOTYP', 'KKK'}
0    0.313401
Name: NaN_Count, dtype: object
0    0.313401
Name: NaN_Count, dtype: object
```

The below graph shows the percentage of values that are NaN in each of the 60 columns:

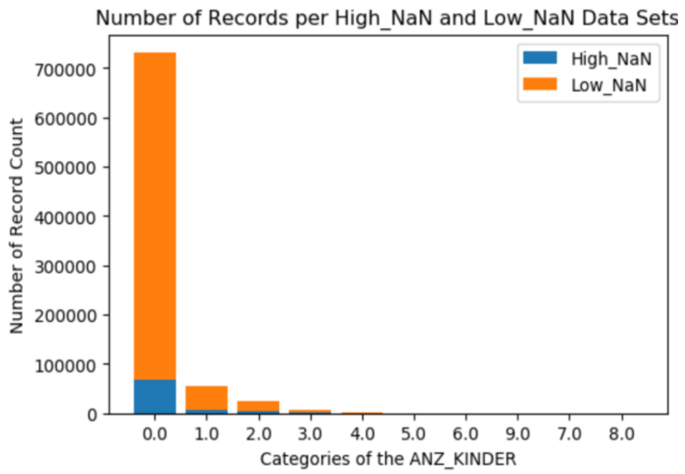


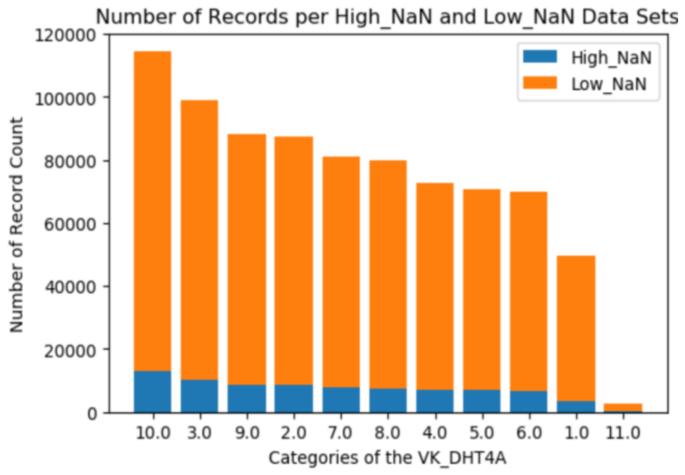
Dealing with rows with lots of NaN

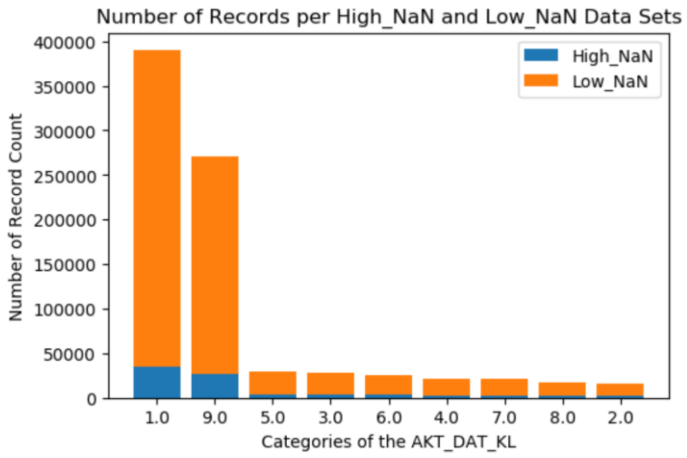
About 17% of the rows in General Population data set have more than 10% of the column being NaN, this is quite good data quality on a row-based level. However, the percentage is slightly higher for Customer Population Data set (~ 30%), this could be due to demographic nature of the customers of the mail-order company.

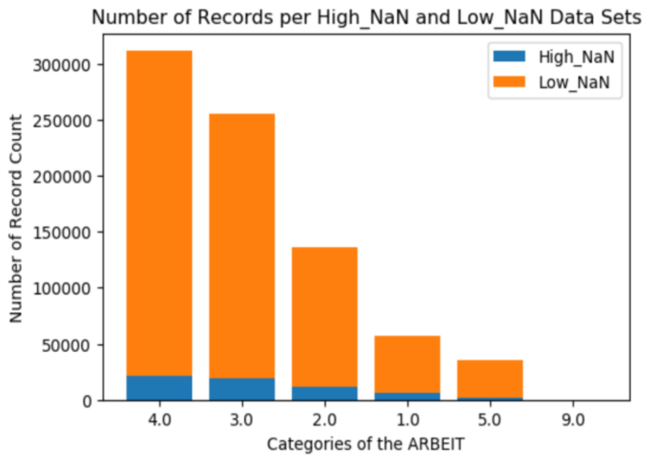
I have decided to split the General Population Dataset based the NaN Count Level. The data set where the rows have less than 10% of the columns being NaN and another data set where the rows having more than 10% of the columns being NaN.

I have graphed the two datasets distribution of values for some randomly selected columns. And from the graph, it appears there is no significant difference in distribution for the columns between two datasets. Therefore, I have decided not to remove those rows with more than 10% of the columns having NaN values as dropping rows from a data set can results in decreasing the accuracy of the model. Later, I will conduct data imputation where NaN values will be converted into certain value.







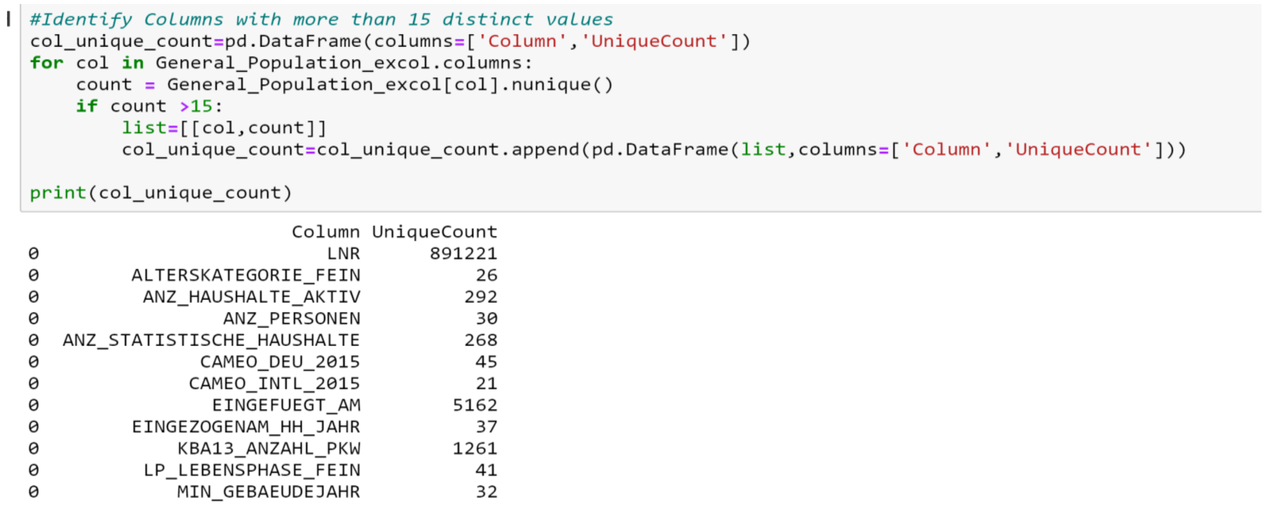


Dropping Additional Columns

The next step is to take a further look at the columns where there are more than 15 distinct values. We need to understand:

- If the values in those columns (as per below) are numerical or categorical. If the columns are categorical then we may want to put them into fewer buckets instead.
- Understand the definition of the column and see if the data in the columns is relevant to the study.

AS can be seen below, those are the columns with more than 15 different values:



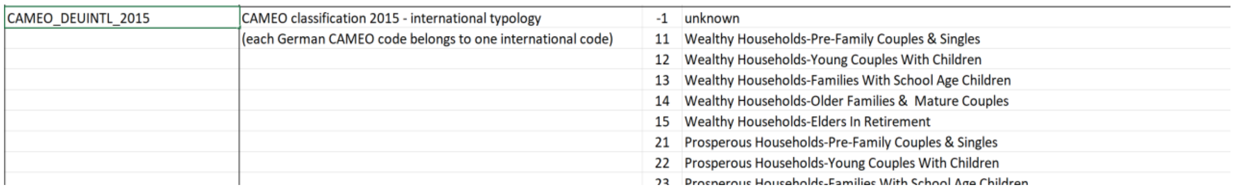
```
#Identify Columns with more than 15 distinct values
col_unique_count=pd.DataFrame(columns=['Column','UniqueCount'])
for col in General_Population_excol.columns:
    count = General_Population_excol[col].nunique()
    if count >15:
        list=[[col,count]]
        col_unique_count=col_unique_count.append(pd.DataFrame(list,columns=['Column','UniqueCount']))

print(col_unique_count)
```

```
                      Column UniqueCount
0                        LNR      891221
0         ALTERSKATEGORIE_FEIN          26
0          ANZ_HAUSHALTE_AKTIV         292
0                ANZ_PERSONEN          30
0  ANZ_STATISTISCHE_HAUSHALTE         268
0               CAMEO_DEU_2015          45
0              CAMEO_INTL_2015          21
0                EINGEFUEGT_AM        5162
0         EINGEZOGENAM_HH_JAHR          37
0             KBA13_ANZAHL_PKW        1261
0          LP_LEBENSPHASE_FEIN          41
0             MIN_GEBAEUDEJAHR          32
```

My Analysis and actions on the fields:

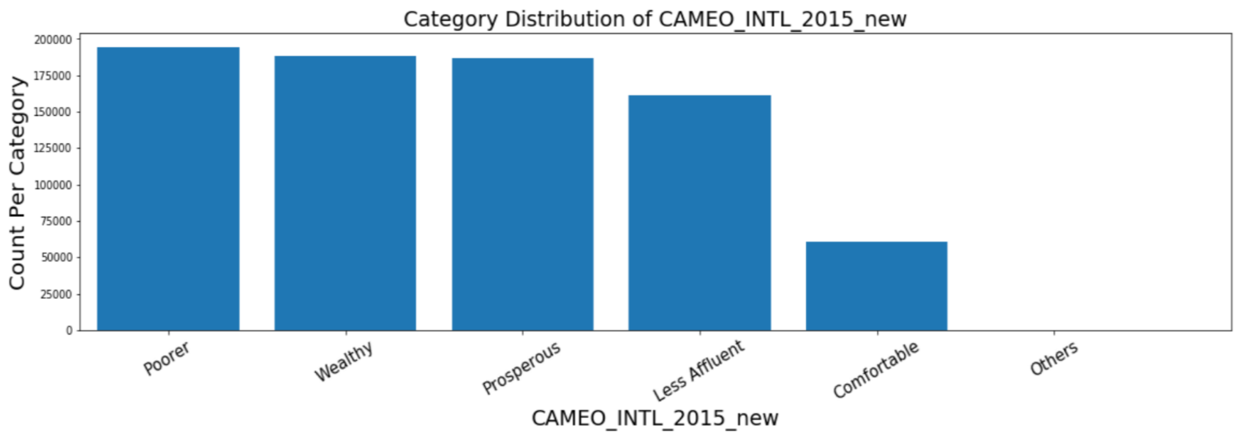
- **LNR**:
  Unique identifier (needs to be drop for training, not an attribute that is useful for training)
- **ALTERSKATEGORIE_FEIN**:
  Age Category (translated from German based on google translator) This field doesn't have a description in the attribute information list; however we have ALTERSKATEGORIE_GROB field which also represents Age Category that is available in the attribute information list (but with fewer distinct values). The two fields appear to show the same information; therefore I have decided to drop this field and keep only ALTERSKATEGORIE_GROB.
- **EINGEZOGENAM_HH_JAHR**:
  Cannot find information of this field in attribute information list, based on google translator the field seems to be the move in year of the individual. This field seems to be irrelevant to a problem we are to solve. Therefore, this field will be dropped.
- **MIN_GEBAEUDEJAHR**
  Based on attribute information list, this field means "year the building was first mentioned in our database". This field seems to be irrelevant to a problem we are to solve. Therefore, this field will be dropped.
- **ANZ_TITEL**
  Although numeric, about 91% of the values in the column is 0, dropped to lack of information that is not 0.
- **LP_LEBENSPHASE_FEIN**
  Like ALTERSKATEGORIE _FEIN, I will keep **LP_LEBENSPHASE_GROB** field only, since there is duplication of data between the two fields.

- **EINGEFUEGT_AM**
  This appears to be timestamp of when the data was inserted, therefore it should be dropped as it is not an attribute of the individual.
- **CAMEO_DEU_2015**
  Like ALTERSKATEGORIE _FEIN, I will keep **CAMEO_DEUG_2015** field only, since there is duplication of data between the two fields.
- **CAMEO_INTL_2015**
  There are 21 categories in this field, however I cannot find information of this field in attribute information list. But there is a field called “**CAMEO_DEUINTL_2015”** with similar name and the same unique values therefore I can group 21 categories into a few buckets based on the individual’s wealth level:
  { 'Wealthy', 'Prosperous', 'Comfortable', 'Less Affluent', 'Poorer', 'Others' }



CAMEO_DEUINTL_2015	CAMEO classification 2015 - international typology	-1	unknown
	(each German CAMEO code belongs to one international code)	11	Wealthy Households-Pre-Family Couples & Singles
		12	Wealthy Households-Young Couples With Children
		13	Wealthy Households-Families With School Age Children
		14	Wealthy Households-Older Families & Mature Couples
		15	Wealthy Households-Elders In Retirement
		21	Prosperous Households-Pre-Family Couples & Singles
		22	Prosperous Households-Young Couples With Children
		23	Prosperous Households-Families With School Age Children

The old **CAMEO_INTL_2015** field will be dropped and the new field **“CAMEO_INTL_2015_new**” is created
After this transformation, we can see the distribution of values in the 5 different buckets as per below in the graph:



Data Imputation and Standardization:

Before applying dimensionality reduction techniques (in this case Principle component analysis), we need to do those three steps below:

1. **Data Imputations:**
   We need to replace all NaN into default values, this is because principle component analysis normally performs poorly with missing values. In this study the logic for data imputation is as per below:
   1. For binary data column: I will replace NaN with most frequent value (mode)
   2. For numerical data: I will replace NaN with median value in the column
   3. For categorical data column: NaN values will be NULL as default.
2. **Data Standardization:**

Convert Categorical string values such as “Wealthy” into an integer 0. This will help the next step – feature scaling.

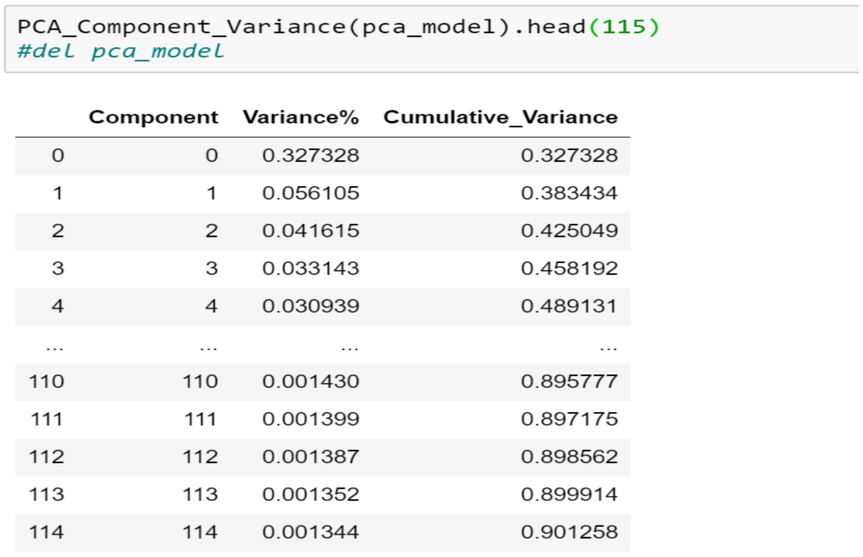
3. **Feature Scaling:**
   Apply standard scaling to all columns in both the general and customer population datasets. Standardizing features is critical when measurements units in each columns of the data sets are different. If we don’t apply scale standardization, then variables that are measured at different scales will not contribute equally to the model and can create a biased result.

Algorithms and Techniques

Dimensionality Reduction (Principle Component Analysis):

Principle Component Analysis (PCA) is an important dimensionality reduction methodology used to reduce the number of dimensions of a large data set. Particularly for data sets with large number of variables, PCA helps by transforming a large set of variables into a smaller set but at the same time ensuring the smaller set will still contain most of the information. By decreasing the number of variables of a large data set, we can make training machine learning algorithms faster and simpler at a little expense of accuracy.

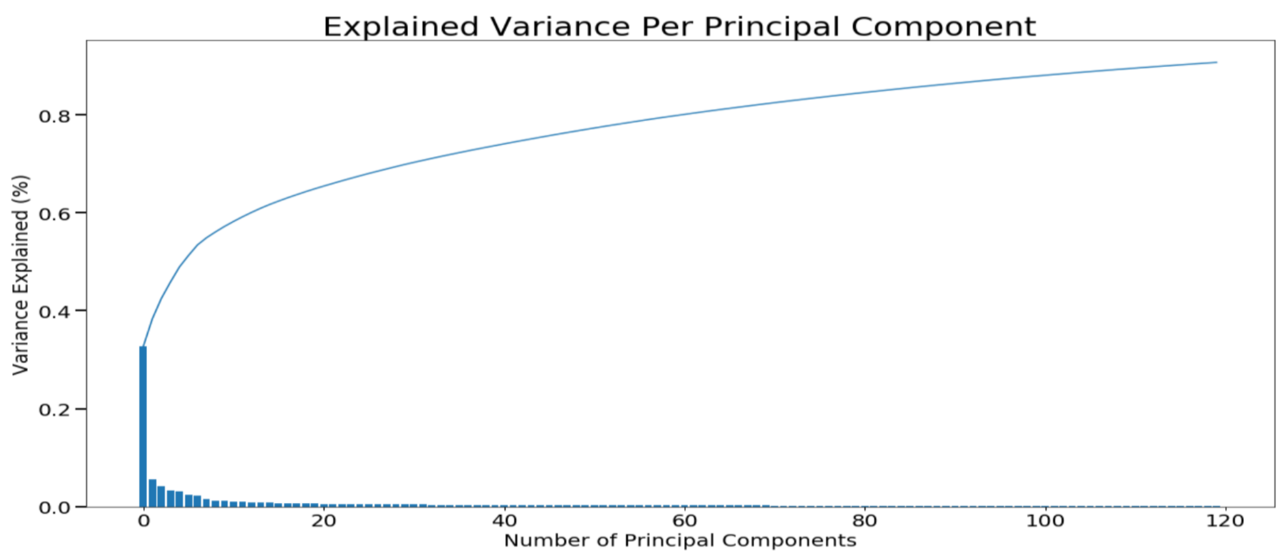
I start with creating a PCA model with undefined number of components, my goal is to find out the minimum number of components that I should keep, in order to have at least 90% of the variance explained.



```
PCA_Component_Variance(pca_model).head(115)
#del pca_model
```

	Component	Variance%	Cumulative_Variance
0	0	0.327328	0.327328
1	1	0.056105	0.383434
2	2	0.041615	0.425049
3	3	0.033143	0.458192
4	4	0.030939	0.489131
...	...	...	...
110	110	0.001430	0.895777
111	111	0.001399	0.897175
112	112	0.001387	0.898562
113	113	0.001352	0.899914
114	114	0.001344	0.901258

As can be seen in the below graph, the top 115 components will explain 90% of the variance in the data and the first component (0) can explain about 32.7% of the variance in the data. With the help of PCA, I have managed to reduce more than half of the number of features, from 298 to 115.
Therefore, I will set components = 115 and retraining the PCA model again and below you can see the scree plot of PCA model with 115 components.



If we look at the top 1 and 2 components in the PCA model and see what are the top 5 and last 5 features that are most significant in those two components. As you can below:

Component 1:

	Feature	Weight
278	SEMIO_VERT	-0.033774
276	SEMIO_SOZ	-0.031499
54	HH_EINKOMMEN_SCORE	-0.031035
20	D19_GESAMT_DATUM	-0.022137
266	SEMIO_ERL	-0.022015
...	...	...
64	KBA05_ANTG4	0.078364
196	KBA13_KW_30	0.079634
176	KBA13_KMH_110	0.080474
183	KBA13_KMH_251	0.080904
188	KBA13_KRSSEG_KLEIN	0.090656

For Component 1:

KBA13_KRSSEG_KLEIN, KBA13_KMH_251, KBA13_KMH_110, KBA13_KW_30 are all car related features.
KBA05_ANTG4: number of >10 family houses in the cell.
SEMIO_SOZ, SEMIO_ERL and SEMIO_VERT: shows the affinity of different personalities of an individual
(whether they are social minded, dreamily, eventful orientated)
HH_EINKOMMEN_SCORE: estimated household net income
D19_GESAMT_DATUM: Actuality of the last transaction with the complete file TOTAL

Component 2:

	Feature	Weight
244	LP_STATUS_GROB	-0.169311
41	FINANZ_MINIMALIST	-0.168849
246	MOBI_REGIO	-0.134795
61	KBA05_ANTG1	-0.132427
72	KBA05_GBZ	-0.130950
...	...	...
38	EWDICHTE	0.132513
131	KBA13_BAUMAX	0.133099
54	HH_EINKOMMEN_SCORE	0.133405
126	KBA13_ANTG3	0.143144
7	CAMEO_DEUG_2015	0.144653

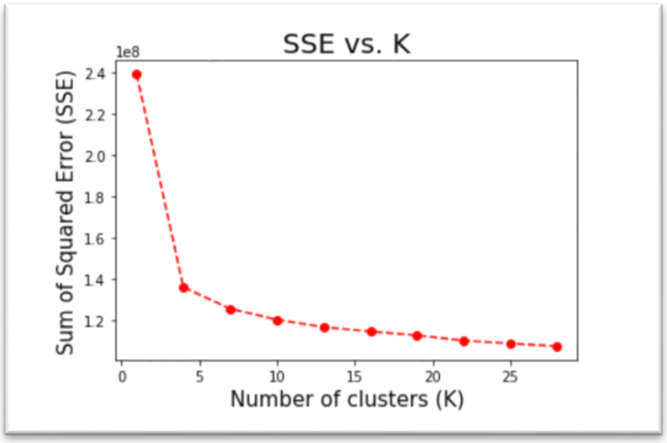
For Component 2:

LP_STATUS_GROB: social status rough
FINANZ_MINIMALIST: financial typology: low financial interest
MOBI_REGIO: moving patterns
KBA05_ANTG1: number of 1-2 family houses in the cell
KBA05_GBZ: number of buildings in the microcell
KBA13_ANTG3: can't find description in the attribute list, but based on most of the KBA13 features are car related. This should be a car related attributes.
CAMEO_DEUG_2015:CAMEO classification 2015 – Uppergroup (this is the financial class of this individual)
This component seems to focus on the financial status and household characteristics of an individual.

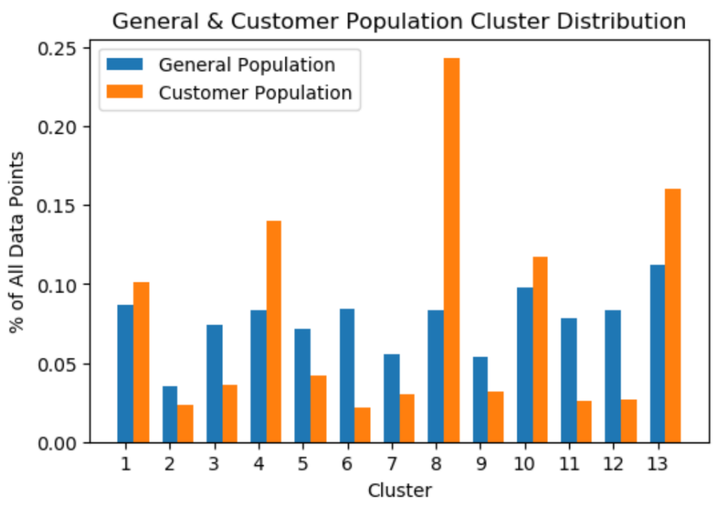
Customer Segmentation Analysis – K-means Clustering

The idea behind data clustering is to group a collection of of data points together due to their similarities in certain attributes. K-means algorithm starts with a group of randomly selected centroids where a centroid is the centre of a cluster. Every data point is allocated to a certain cluster. The model will intend to optimise the position of the centroids by reducing the in-cluster sum of squares of each data point to the cluster it is being allocated. Our goal is to allocates every data point in the population dataset to the closest cluster as possible, but at the same minimizing the number of centroids as possible. I have fitted the K-means clustering model using General population data that has been processed by the PCA (dimensionality reduced). The number of centroids is a hyperparameter and must be defined, therefore I will loop through a range of number (1-30) and use the elbow method to pick the appropriate K number of centroids. The elbow method gives insights into the sum of squared distance (SSE) between each data point to their allocated cluster's centroids for each given number of clusters (K). We generally pick "K" where SSE beginning to flatten out and an elbow is formed.

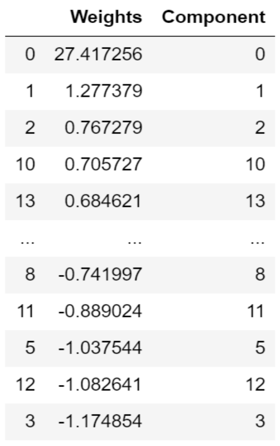
I have plotted an elbow graph, so we can visualise by which K the elbow shape will be formed. As you see below in the elbow graph, there is a repaid decreased in SSE for the first 4-5 clusters and moderate decreases in SSE from between 5-13 clusters. After cluster 13, the decrease in SSE is comparatively minimal. Therefore, the number of clusters I picked is K = 13 for this project.



The next step of clustering analysis is to compare the general and customer population data sets for those 13 clusters. (the graph below)
And you can see, Customer population is significantly more presented in cluster 8, cluster 3 and cluster 13 comparing to the general population. On the other hand, Customer population is significantly less presented in the cluster 6, cluster 11 and cluster 12.



If we look at what PCA components make up cluster 8:

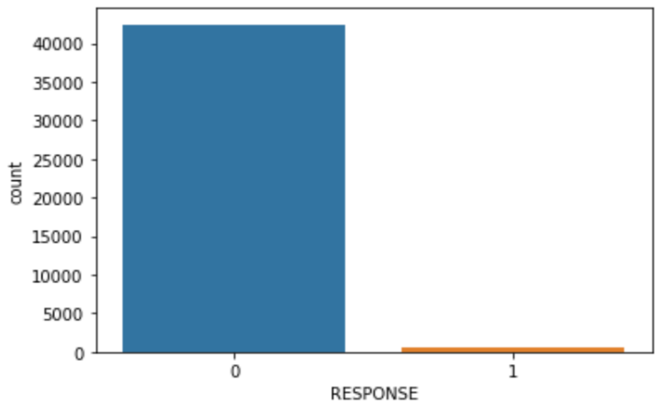


	Weights	Component
0	27.417256	0
1	1.277379	1
2	0.767279	2
10	0.705727	10
13	0.684621	13
...	...	...
8	-0.741997	8
11	-0.889024	11
5	-1.037544	5
12	-1.082641	12
3	-1.174854	3

We can see PCA component 1 and 2 have the most weight. This means that features that are most significant in those two PCA components are also most significant in the cluster 8 as well.

Supervised Learning Models

Now moving on to build and train a supervised learning model to help the mail-order company to identify potential customers out of the general population. I started off loading in the training data set "Udacity_Mailout_052018_train.csv" and immediately I identified there is a massive class imbalance between the Response Class. As can be seen below, this means we need to deal with class imbalance issue before we can train our model, otherwise our machine learning model will be very inclined to predict 0.

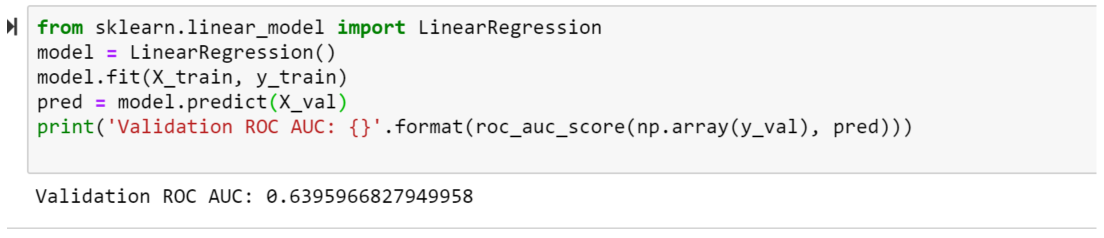


The method I used to deal with class imbalance is called "Resampling". Resampling is a up-sampling method, which it will randomly duplicate observations from the minority class with replacement. This will reinforce the minority class importance in the data set. The aim is to end up having the same number of records in both classes after resampling.

Next I will clean the train data set using the same process as for the unsupervised learning study. Then splitting the data set into train and validation two data sets. As a general practice, 70% of the data will be allocated to the train data set and 30% of the data will be allocated to the validation data set.

Benchmark Model:

The benchmark model for this study is “Logistic Regression model” and without any hyperparameter tunning and using only the default parameters of the model, I am getting a ROC AUC of 0.639.



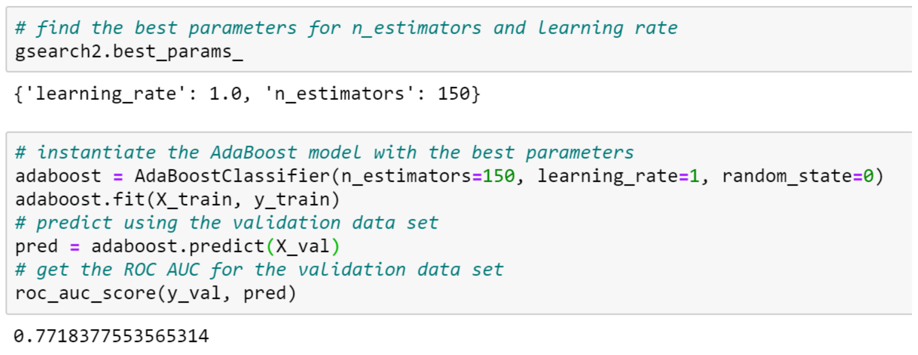
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
pred = model.predict(X_val)
print('Validation ROC AUC: {}'.format(roc_auc_score(np.array(y_val), pred)))
```

```
Validation ROC AUC: 0.6395966827949958
```

Those following models has been trained for this project and hyperparameter tunning has been applied to find the best models per each machine learning model:

AdaBoost Classifier Model:

Hyperparameters tuning shows the best parameters for AdaBoost Classifier Model is:



```
# find the best parameters for n_estimators and learning rate
gsearch2.best_params_
```

```
{'learning_rate': 1.0, 'n_estimators': 150}
```

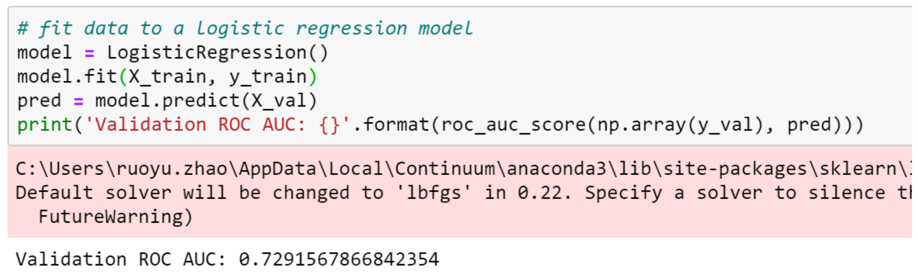
```
# instantiate the AdaBoost model with the best parameters
adaboost = AdaBoostClassifier(n_estimators=150, learning_rate=1, random_state=0)
adaboost.fit(X_train, y_train)
# predict using the validation data set
pred = adaboost.predict(X_val)
# get the ROC AUC for the validation data set
roc_auc_score(y_val, pred)
```

```
0.7718377553565314
```

ROC AUC on the validation data set is 0.77.

Logistic Regression Model:

A basic logistic regression model has a ROC AUC of 0.729:

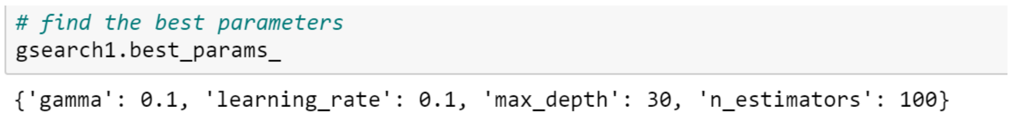


```
# fit data to a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
pred = model.predict(X_val)
print('Validation ROC AUC: {}'.format(roc_auc_score(np.array(y_val), pred)))
```

```
C:\Users\ruoyu.zhao\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence t
  FutureWarning)

Validation ROC AUC: 0.7291567866842354
```

XGBoost Model:

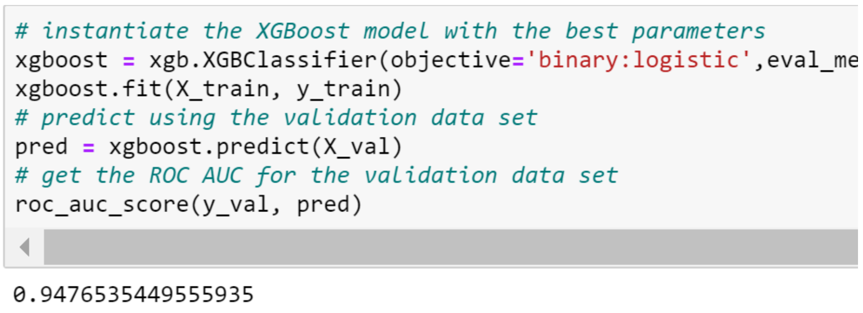
Hyperparameters tuning shows the best parameters for XGBoost Model is:



```
# find the best parameters
gsearch1.best_params_
```

```
{'gamma': 0.1, 'learning_rate': 0.1, 'max_depth': 30, 'n_estimators': 100}
```

The ROC AUC score is high on the validation data set (achieving 0.947) using this best trained model.



```
# instantiate the XGBoost model with the best parameters
xgboost = xgb.XGBClassifier(objective='binary:logistic',eval_me
xgboost.fit(X_train, y_train)
# predict using the validation data set
pred = xgboost.predict(X_val)
# get the ROC AUC for the validation data set
roc_auc_score(y_val, pred)
```

```
0.9476535449555935
```

Results

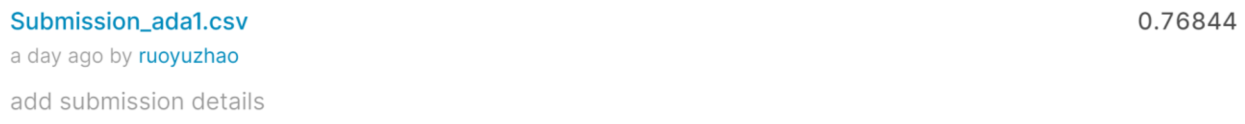
Validation ROC AUC:

Based on ROC AUC score on the validation data set, XGBoost model appears to be the best model out of all the other machine learning models trained by achieving a 0.947 ROC AUC.

Kaggle Competition:

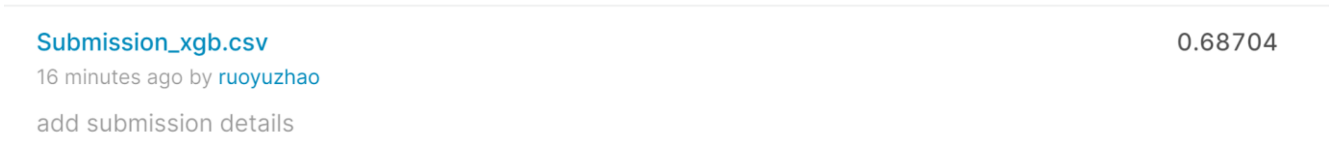
However, based on ROC AUC score calculated by Kaggle Competition, AdaBoost Classification model appears to be the best model out of all the other machine learning models trained by achieving a 0.768 ROC AUC. But the XGBoost model only achieved a ROC AUC score of 0.687. This may be due to the XGBoost model being overfitted.

ADABoost model:



Submission_ada1.csv 0.76844
a day ago by ruoyuzhao
add submission details

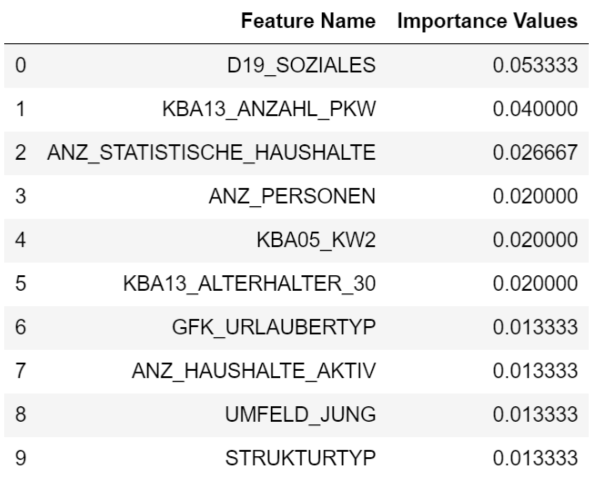
XGBoost model:



Submission_xgb.csv 0.68704
16 minutes ago by ruoyuzhao
add submission details

Most Important Features:

It is also important to see what the most important attributes to a machine learning model are. As can be seen below, for the best performing model AdaBoost Classification model, those are the top 10 features that are most important to the model based on the important values assigned to them.



	Feature Name	Importance Values
0	D19_SOZIALES	0.053333
1	KBA13_ANZAHL_PKW	0.040000
2	ANZ_STATISTISCHE_HAUSHALTE	0.026667
3	ANZ_PERSONEN	0.020000
4	KBA05_KW2	0.020000
5	KBA13_ALTERHALTER_30	0.020000
6	GFK_URLAUBERTYP	0.013333
7	ANZ_HAUSHALTE_AKTIV	0.013333
8	UMFELD_JUNG	0.013333
9	STRUKTURTYP	0.013333

Top 5 features

D19_Soziales: can't find description in the attribute lists.

KBA13_ANZAHL_PKW: Number of cars an individual has.

ANZ_STATISTISCHE_HAUSHALTE: can't find description in the attribute lists. But should be related to household.

ANZ_PERSONEN: Number of adults in a household

KBA05_KW2: share of cars with an engine power between 60 and 119 KW

Further Improvements

There are spaces for further improvements to improve the accuracy of the model, such as:

- Understand the meaning of all the features and make wiser decision on how we should process and clean the data at a column wise level.
- More feature engineering can be conducted.
- Train the data on other machine learning models, such as support vector machine, Naïve bayes classification and Gradient Boosting Classifier.
- Enhance the scope of hyperparameters training on the machine learning model (grid search on more hyperparameters and setting the search range of each parameter bigger)
- Do more investigations into the cluster features in the customer population.