Customer Segmentation Report for Arvato Financial Services

Project Overview

In this project, we will analyze demographics data for customers of a mail-order sales company in Germany, comparing it against demographics information for the general population. We will use unsupervised learning techniques to perform customer segmentation, identifying the parts of the population that best describe the core customer base of the company. Then, apply what we have learned 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. The data use has been provided by our partners at Bertelsmann Arvato Analytics, and represents a real-life data science task.

Problem Statement

The goal of this project is to characterize customers segment of population, and to build a model that will be able to predict customers for Arvato Financial Solutions.By conducting this analysis, they can increase the efficiency in the customer acquisition process. Instead of reaching out to all people in Germany and targeting them with a marketing campaign, they would just reach out to the people they have identified as becoming likely new customers and then do them targeted advertising.

Metrics

Assessing the quality of your model is one of the most important considerations when deploying any machine learning algorithm. In this solution the performance of model is evaluated via Accuracy, Precision, Recall and Confusion Metrics. A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.

The Confusion Metrics looks like:

	Predicted Class			
Actual Class		Yes	No	
	Yes	True Positive	False Negative	
	No	False Positive	True Negative	

Accuracy = TP+TN/TP+FP+FN+TN

Precision = TP/TP+FP

Recall = TP/TP+FN

Data Exploration

There are four data files provided by Arvato Financial Services:

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

Each row of the demographics files represents a single person, but also includes information outside of individuals, including information about their household, building, and neighborhood. The "CUSTOMERS" file contains three extra columns ('CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP'), which provide broad information about the customers depicted in the file. The original "MAILOUT" file included one additional column, "RESPONSE", which indicated whether or not each recipient became a customer of the company. For the "TRAIN" subset, this column has been retained, but in the "TEST" subset it has been removed; it is against that withheld column that the final predictions will be assessed in the Kaggle competition.
The information from the first two files will be used to figure out how customers are similar to or differ from the general population at large, then the analysis will be used to make predictions on the other two files, predicting which recipients are most likely to become a customer for the mail-order company.
Also two Excel spreadsheets provided with data for more information about columns in the files, DIAS Information Levels - Attributes 2017.xlsx is a top-level list of attributes and descriptions, organized by informational category and DIAS Attributes - Values 2017.xlsx is a detailed mapping of data values for each feature in alphabetical order.

Exploratory Visualization

The general demographic data provided has total of 366 features per person and the initial five rows of dataframe are as follows.

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

5 rows × 366 columns

As we can see many of values are missing (NaN) which we going to encode in pre-processing section.

Following is the dataframe of features values which will be used for encoding the data.

	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

Next is the dataframe of features information describing each feature.

	Information level	Attribute	Description	Additional notes
0	NaN	AGER_TYP	best-ager typology	in cooperation with Kantar TNS; the informatio...
1	Person	ALTERSKATEGORIE_GROB	age through prename analysis	modelled on millions of first name-age-referen...
2	NaN	ANREDE_KZ	gender	NaN
3	NaN	CJT_GESAMTTYP	Customer-Journey-Typology relating to the pref...	relating to the preferred information, marketi...
4	NaN	FINANZ_MINIMALIST	financial typology: low financial interest	Gfk-Typology based on a representative househo...

The customer dataframe looks like below table and it has extra three features than that of general population dataframe.

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

5 rows × 369 columns

Following is the train dataset having extra column of RESPONSE or LABELS.

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

5 rows × 367 columns

The test data set is not having any RESPONSE column which we going to predict in further sections.

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

5 rows × 366 columns

Algorithm and Techniques

The problem solution has two major parts: Customer Segmentation and Prediction Model.

1. **Customer Segmentation:**

- This is to analyze demographics data for customers of the mail-order sales company in Germany (Udacity_CUSTOMERS_052018.csv), comparing it against demographics information for the general population (Udacity_AZDIAS_052018.csv). This will use unsupervised learning techniques to perform customer segmentation, identifying the parts of the population that best describe the core customer base of the company. The solution uses PCA for feature reduction and KMeans algorithm to form clusters.
- **Principal component analysis** (**PCA**) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components.
- **K-means** is a simple unsupervised machine learning algorithm that groups a dataset into a user-specified number (*k*) of clusters. The algorithm is somewhat naive--it clusters the data into *k* clusters, even if *k* is not the right number of clusters to use. Therefore, when using k-means clustering, users need some way to determine whether they are using the right number of clusters.

2. **Prediction Model:**
    - This will use the previous analysis of step 1 to build a supervised machine learning model on Udacity_MAILOUT_052018_TRAIN.csv and Udacity_MAILOUT_052018_TEST.csv that predicts whether or not each individual will respond to the campaign. The solution uses AWS SageMaker inbuilt Linear Learner algorithm.
    - **The Amazon SageMaker linear learner** algorithm provides solution for both classification and regression problems. With this algorithm we can simultaneously explore different training objectives and choose the best solution from a validation set.

Benchmark Model

The benchmark model would be applying the Unsupervised Machine Learning Model to find possible segments of the general population and compare it with segments of core customers. KMeans model will be applied to make segmentation of general population and customers. Further Amazon SageMaker's default LinearLearner algorithm would be used for classification.

Data Pre-processing

Files "DIAS Information Levels - Attributes 2017.xlsx" and "DIAS Attributes – Values 2017.xlsx" are used to understand the features and data values given in all four data sets. Accordingly pre-processing steps are decided and initially applied to General Population dataset.

Step1: Data Cleaning

1.1: Read and Observe Dataset

Details of General Population Dataset is as follows:

```
azdias.head()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891221 entries, 0 to 891220
Columns: 366 entries, LNR to ALTERSKATEGORIE_GROB
dtypes: float64(267), int64(93), object(6)
memory usage: 2.4+ GB
```

1.2: Process Missing data

- Features that are not present in dataset of general population are found out and are removed from feature summary dataframe.
- Features that are present in dataset of general population but missing in feature summary file are separated and a new dataframe of missing features is created to append it to original feature summary dataframe.

1.3: Convert Missing Value Codes to NaNs

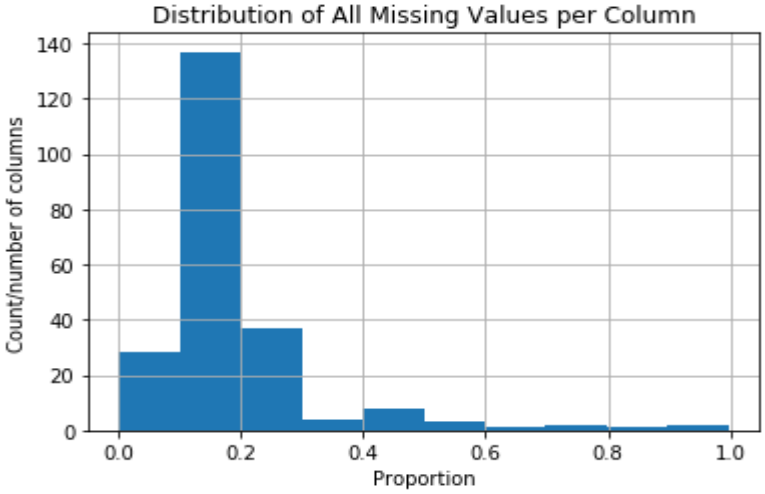
- From data value ("DIAS Attributes – Values 2017.xlsx") file, it has found out that for some of the features unknown values are coded either as 0 and/or -1 and/or 9. A new dataframe is created to depict these unknown values and used it to convert unknown values to NaN.
- Also features which are not listed in data value file are dropped from original general population dataframe as it will be difficult to interpret the codes used for these features.

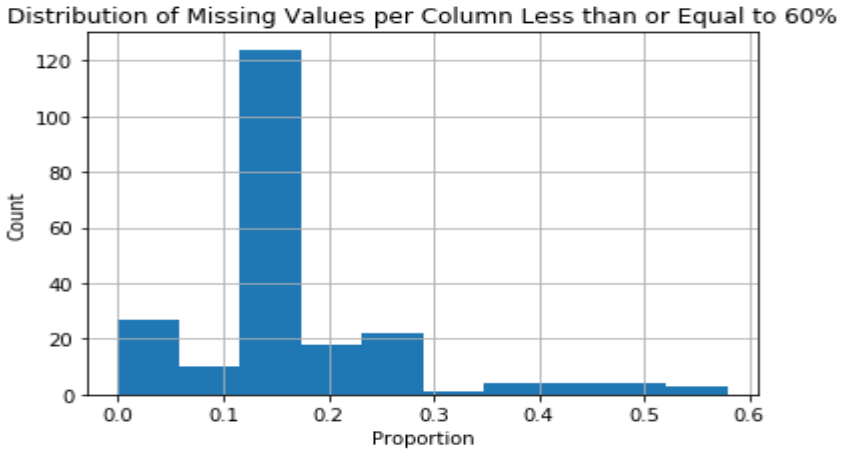
Discussion

There are total of 143 columns which are not listed in data value file and are removed from the feature list of datasets.

1.4: Find Outliers

After performing basic code conversion it is necessary to perform assessment of how much missing data there in each column of the dataset. The histogram of missing data is plotted.





As it is clearly seen that there are very few number of columns which has fraction of missing data above 60% and those columns can be removed for the simplicity without losing much data.

Discussion

These outliers are as follows:

a) 'AGER_TYP': best-ager typology
b) 'KBA05_ANTG3': number of 6-10 family houses in the cell
c) 'KBA05_ANTG4': number of >10 family houses in the cell
d) 'KBA05_SEG6': share of upper class cars (BMW 7er etc.) in the microcell
e) 'KBA05_SEG8': share of roadster and convertibles in the microcell
f) 'TITEL_KZ': flag whether this person holds an academic title

1.5: Categorical and mixed-type features

An investigation of the categorical and mixed-type features is done and decided whether they will be kept, dropped, or re-encoded. Finally, in the last part, a new data frame will be created with only the selected and engineered columns.

For categorical data, levels are encoded as dummy variables. Depending on the number of categories, one of the following is performed

- For binary (two-level) categorical that take numeric values, they are kept without needing to do anything.
- For multi-level categorical (three or more values), the values are encoded using multiple dummy variables.

Discussion

There are 3 binary columns 'PLZ8_ANTG4', 'VERS_TYP', 'ANREDE_KZ' and 213 multilevel categorical columns. 'PRAEGENDE_JUGENDJAHRE' has combine information on three dimensions: generation by decade, movement (mainstream vs. avantgarde), and nation (east vs. west). While there aren't enough levels to disentangle east from west, two new variables are created to capture the other two dimensions: an interval-type variable for decade, and a binary variable for movement.

1.6: Cleaning Function

Pre-processing done on general population dataframe is supposed to be used on all other provided datasets and to simplify this process, all steps defined above are framed as compact function for reuse.

Discussion

Post cleaning and pre-processing the data set size is 89,1221 rows and 1090 columns.

Step2: Feature Transformation

2.1: Feature Scaling

Before dimensionality reduction techniques are applied to the data, feature scaling must be performed so that the principal component vectors are not influenced by the natural differences in scale for features.

- sklearn requires that data not have missing values in order for its estimators to work properly. So, before applying the scaler to the data, the dataframe must be cleaned of the remaining missing values before applying the scaler. This is done by applying an Imputer to replace all missing values.
- For the actual scaling function, a StandardScaler instance is done, scaling each feature to mean 0 and standard deviation 1.
- For these classes, the .fit_transform() method is used to both fit a procedure to the data as well as applied the transformation to the data at the same time.

Discussion

To avoid memory errors and long execution time of processing, only 60% of general population dataset is used for feature transformation.

2.2: Dimensionality Reduction

On the scaled data, dimensionality reduction techniques can now be applied.

- sklearn's Principal Component Analysis class is used to apply principal component analysis on the data, thus finding the vectors of maximal variance in the data. To start, at least half the number of features are set (so there's enough features to see the general trend in variability).
- The ratio of variance explained by each principal component as well as the cumulative variance explained is checked by plotting the cumulative or sequential values using matplotlib's plot() function. Based on the findings, a value for the number of transformed features is retained for the clustering part of the project.
- Once a choice for the number of components to keep has been made, the PCA instance is re-fit to perform the decided-on transformation.

2.3: Interpret Principal Components

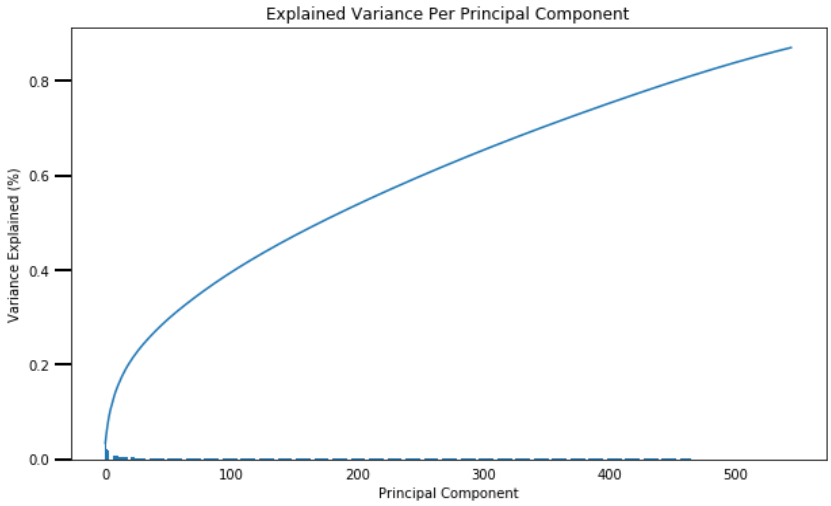
Now that the principal components have been transformed, the weight of each variable on the first few components should be checked to see if they can be interpreted in some fashion.

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

To investigate the features, each weight should be mapped to their corresponding feature name, then the features should be sorted according to weight. The most interesting features for each principal component, then, will be those at the beginning and end of the sorted list.

Discussion

The plot for variance accounted for by each principal component:



For each principal component or dimension, the top 3 and bottom 3 weights with their corresponding feature names are investigated for any associations.

Dimension1:

- SEMIO_VERT_1(0.1133): affinity indicating in what way the person is dreamily(highest affinity)
- FINANZTYP_4(0.1037): best describing financial type for the person(be prepared)
- SEMIO_DOM_6(0.1015): affinity indicating in what way the person is dominant minded(very low affinity)
- KBA13_KMH_110_1.0(-0.0886): share of cars with max speed 110 km/h within the PLZ8(very low)
- KBA13_KMH_251_1.0(-0.0998): share of cars with a greater max speed than 250 km/h within the PLZ8(very low)
- KBA13_KRSSEG_KLEIN_2.0(-0.1213): share of small cars (referred to the county average) - PLZ8(average)

The first principal component is strongly correlated with what way the person is dreamily, financial type of the person and what way the person is dominant minded. KBA13 is not described in the attributes file but it can be related to car ownership. Share of cars with max speed 110 km/h within the PLZ8, a greater max speed than 250 km/h within the PLZ8 and small cars tend to negatively affect this principal component.

Dimension2:

- PLZ8_ANTG1_1.0 (0.1314): number of 1-2 family houses in the PLZ8(low share)
- PLZ8_ANTG3_3.0(0.1279): number of 6-10 family houses in the PLZ8(high share)
- HH_EINKOMMEN_SCORE_6.0(0.1088): estimated household net income (very low income)
- PLZ8_GBZ_5.0(-0.0718): number of buildings within the PLZ8(more than 449 buildings)
- PLZ8_ANTG3_1.0(-0.0721): number of 6-10 family houses in the PLZ8(low share)
- OST_WEST_KZ(-0.1048): flag indicating the former GDR/FRG

Number of 1-2 family houses and 6-10 family houses in the PLZ8, estimated household net income are the factors which are positively correlated to second principal component whereas number of buildings within the PLZ8, low share of 6-10 family houses in the PLZ8 and former GDR/FRG are negatively correlated with this component.

Dimension3:

- EWDICHTE_6.0 (0.1078): density of inhabitants per square kilometer(more than 999 HH/²)
- KBA13_HERST_BMW_BENZ_5.0(0.1077): share of BMW & Mercedes Benz within the PLZ8(very high)
- KBA13_SEG_SPORTWAGEN_5.0(0.1058): share of sports cars within the PLZ8(very high)
- KBA13_SEG_SPORTWAGEN_1.0(-0.0758): share of sports cars within the PLZ8(very low)
- EWDICHTE_1.0(-0.0821): density of inhabitants per square kilometre(less than 34 HH/km²)
- WOHNLAGE_7.0(-0.0869): residential-area (rural neighbourhood)

The Third principal component increases with density of inhabitants per square kilometre and share of BMW & Mercedes Benz & sports cars within the PLZ8 whereas decreases with density of inhabitants per square kilometre and residential-area.

Implementation

Step1: Clustering

1.1: Apply Clustering to General Population Data

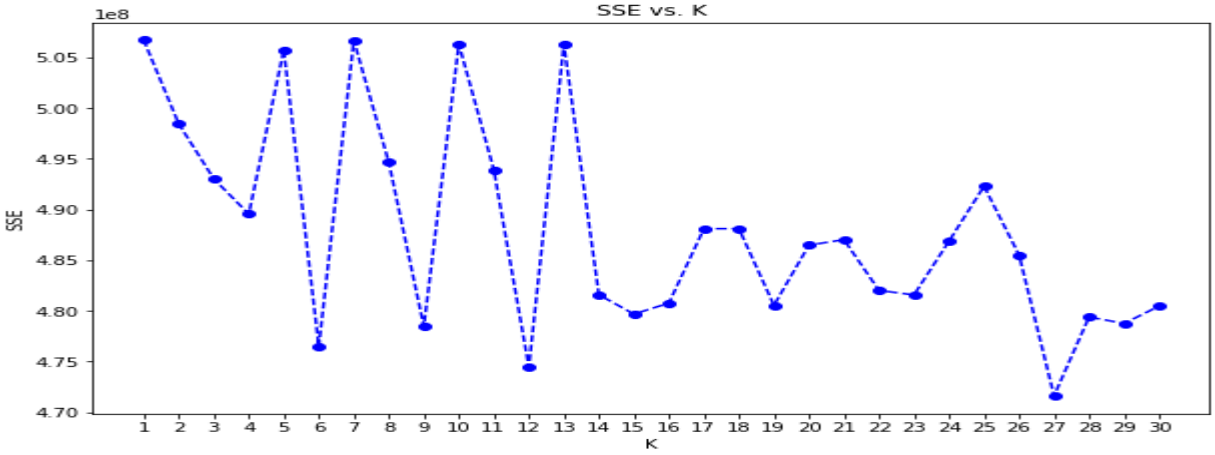
K-means is a simple unsupervised machine learning algorithm that groups a dataset into a user-specified number (*k*) of clusters. The algorithm is somewhat naive--it clusters the data into *k* clusters, even if *k* is not the right number of clusters to use. Therefore, when using k-means clustering, users need some way to determine whether they are using the right number of clusters.

One method to validate the number of clusters is the elbow method. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of *k* and for each value of *k* calculate the sum of squared errors (SSE).

Then, plot a line chart of the SSE for each value of *k*. If the line chart looks like an arm, then the "elbow" on the arm is the value of *k* that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase *k* (the SSE is 0 when *k* is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the centre of its cluster). So our goal is to choose a small value of *k* that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing *k*.

Discussion

Instead of using KMeans class to perform k-means clustering on the PCA-transformed data for a number of different cluster counts, an alternative called MiniBatchKMeans is used instead. The user guide says that it uses mini-batches to reduce the computation time, while still attempting to optimise the same objective function. MiniBatchKMeans converges faster than KMeans, but the quality of the results is reduced. In practice this difference in quality can be quite small.



The SSE vs K plot for the given dataset shows that the score or the sum of the squared errors (SSE) generally decreased as the number of clusters increased. The maximum clusters used is 30. The 'elbow method' is not applicable in the plot because there is no visible levelling observed.

1.2: Apply Clustering to Customer Data

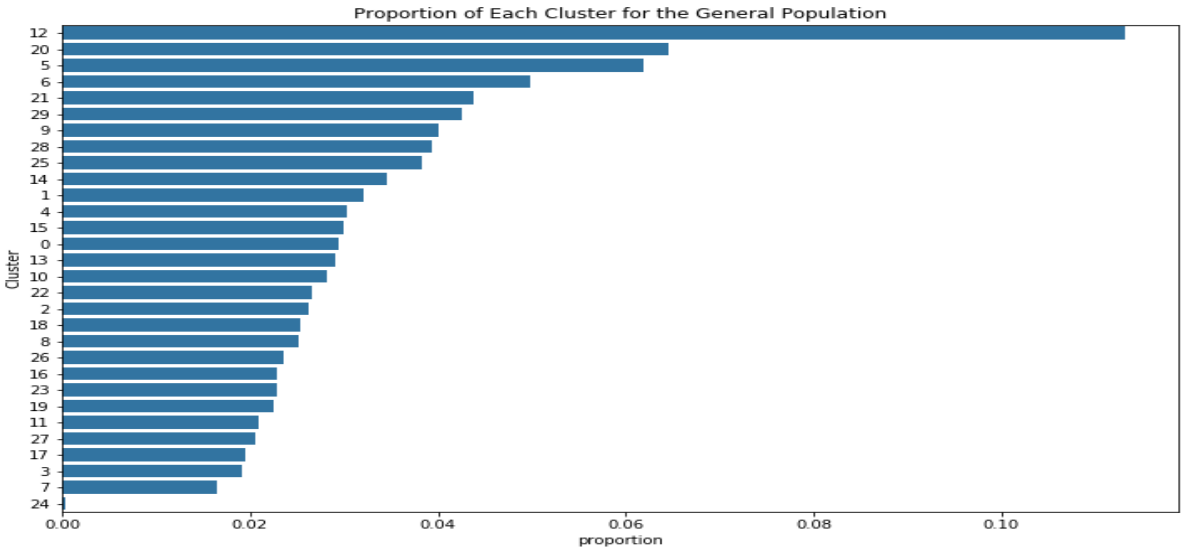
Now that the clusters and cluster centres for the general population have obtained, it's time to see how the customer data maps on to those clusters. The fits from the general population is used to clean, transform, and cluster the customer data. In the next step, there is an interpretation on how the general population fits apply to the customer data.

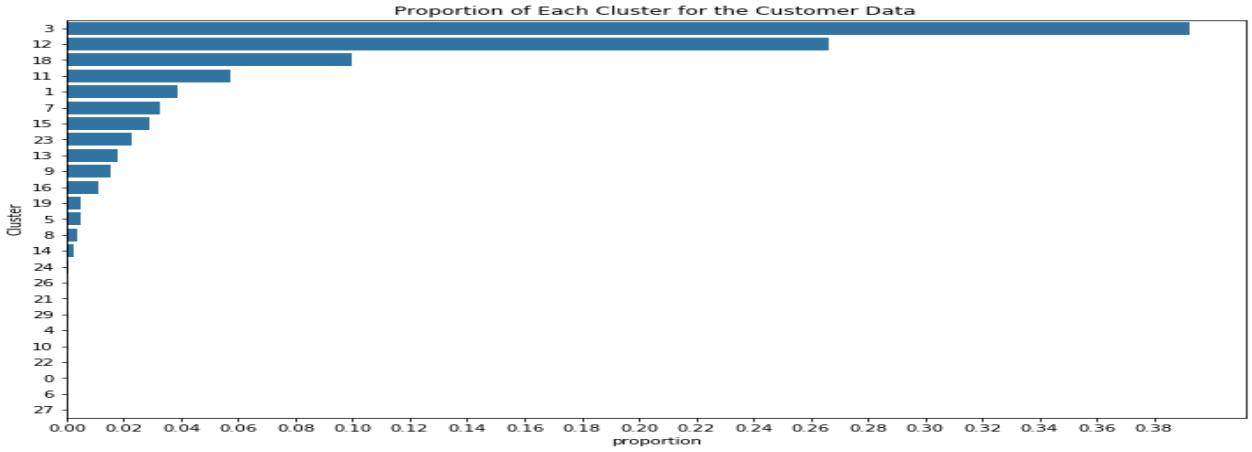
1.3: Compare Customer Data to Demographics Data

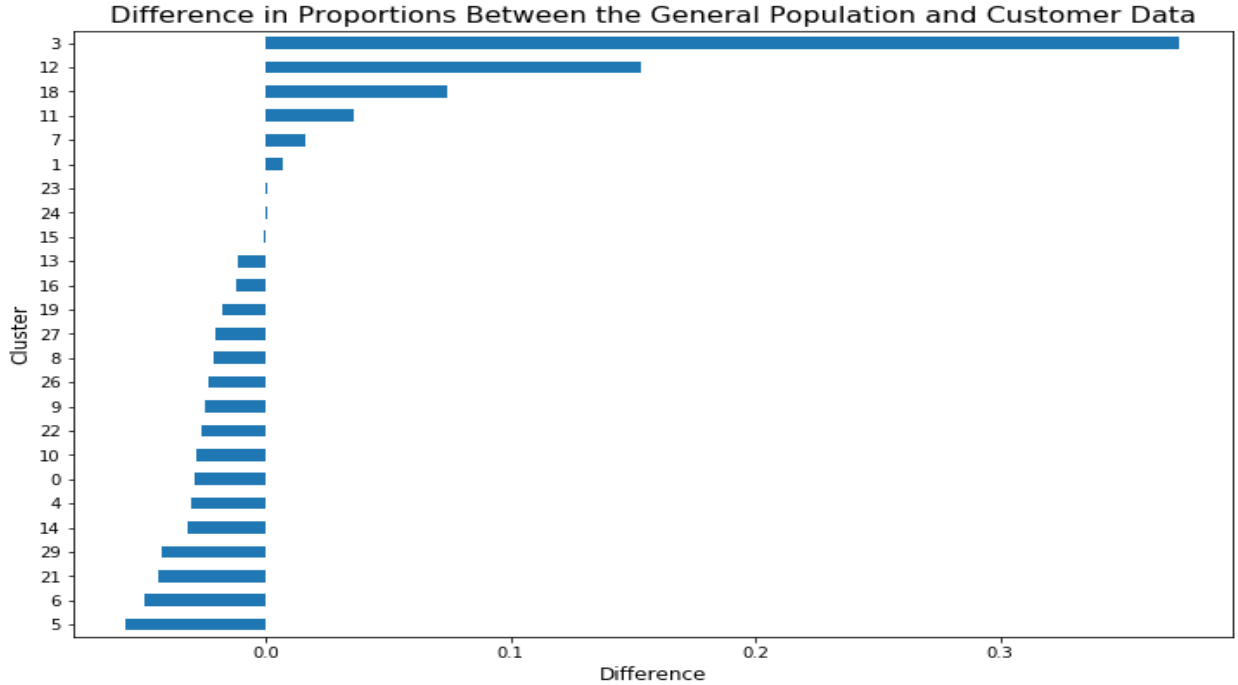
At this point, there are clustered data based on demographics of the general population of Germany, and the customer data for a mail-order sales company has been mapped onto those demographic clusters. In this final sub step, the two cluster distributions are compared to see where the strongest customer base for the company is.

Discussion

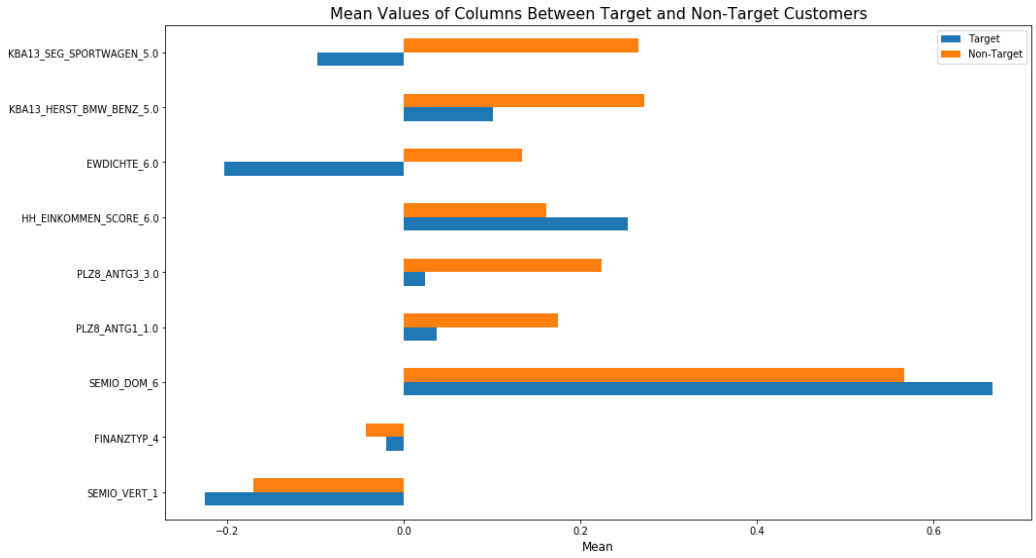
Following figures shows the proportion of data in each cluster for the customer data to the proportion of data in each cluster for the general population.







The cluster which is the most overrepresented is cluster 3 with a difference of 0.3732 and the clu ster which is the most underrepresented is cluster 5 with a difference of -0.0572.



- KBA13_SEG_SPORTWAGEN_5.0 0.363876
- EWDICHTE_6.0 0.337546
- PLZ8_ANTG3_3.0 0.199672
- KBA13_HERST_BMW_BENZ_5.0 0.170769
- PLZ8_ANTG1_1.0 0.137220
- SEMIO_DOM_6 0.099572
- HH_EINKOMMEN_SCORE_6.0 0.092905
- SEMIO_VERT_1 0.055339
- FINANZTYP_4 0.022611

Based on the principal component interpretations, the columns to be interpreted on the original data of the chosen clusters can be identified. Only three out of the nine features above are clearly different.

These PLZ8_ANTG3_3.0, KBA13_SEG_SPORTWAGEN_5.0 and EWDICHTE_6.0. The target customers are having high share of 6-10 family houses in the PLZ8, very high share of sports cars within the PLZ8 and more than 999 HH/²density of inhabitants per square kilometer

Step2: Classification

We are provided with two data files as follows:
Udacity_MAILOUT_052018_TRAIN.csv which is demographics data for individuals who were targets of a marketing campaign. This dataset is composed of 42,982 persons (rows) x 367 (columns: 366 features and 1 label).
Udacity_MAILOUT_052018_TEST.csv which is demographics data for individuals who were targets of a marketing campaign. This dataset is composed of 42,833 persons (rows) x 366 (features columns).

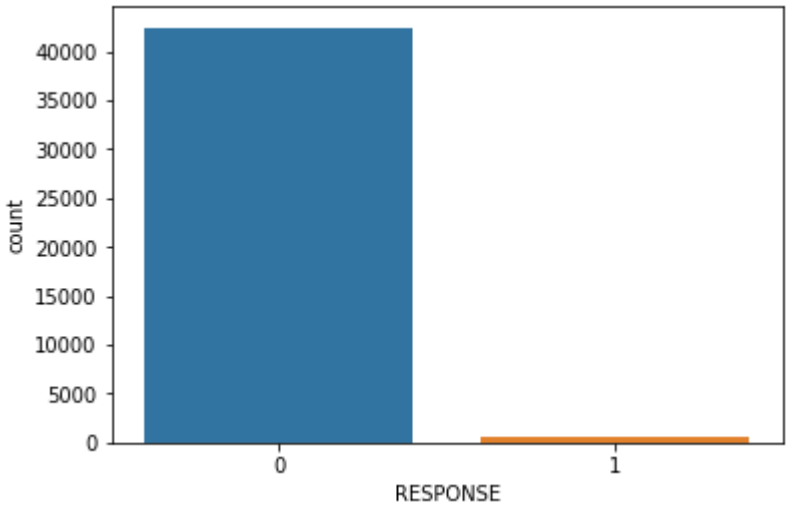
2.1: Data Observation

The Mailout train data is used for traing the classification model.The last column RESPONSE is the target label (whether or not the customer becamse a customer or not). All other applicable columns are features about each individual in the mailout campaign.

The structure of this dataset is as follows:

```
Total number of records: 42962
Individuals who became customers: 532
Individuals who did not become customers: 42430
Percentage of individuals who became customers: 1.2383036171500394%
```

Out of all the 42,962 individuals in the mailout campaign, only 1.24% of the individuals became customers. The dataset is highly imbalanced because of the disproportionate amount of customers and non-customers as shown in figure below.



2.2: Data Pre-processing

Cleaning function defined above is used to pre-process Mailout train and test datasets. Also Mailout train data is then split into training and testing datasets which is used to train and deploy the classification model.
Post pre-processing training dataframe of features consists of 30,073 rows and 1087 columns which is supposed to be used for model training.

2.3 Model Training

AWS SageMaker's inbuilt algorithm LinearLearner is used for classification purpose. Linear models are supervised learning algorithms used for solving either classification or regression problems. For input, you give the model labelled examples (*x*, *y*). *x* is a high-dimensional vector and *y* is a numeric label. For binary classification problems, the label must be either 0 or 1. The algorithm learns a linear function, or, for classification problems, a linear threshold function, and maps a vector *x* to an approximation of the label *y*.

Discussion

The Hyperparameters used to train the model are as follows:

- `predictor_type='binary_classifier'`

Specifies the type of target variable as a binary classification, multiclass classification, or regression

- `epochs=5`
  The maximum number of passes over the training data
- `binary_classifier_model_selection_criteria='precision_at_target_recall'`
  When predictor_type is set to binary_classifier, the model evaluation criteria for the validation dataset (or for the training dataset if you don't provide a validation dataset). Criteria include:
  i. `accuracy`— The model with the highest accuracy.
  ii. `f_beta`— The model with the highest F1 score. The default is F1.
  iii. `precision_at_target_recall`— The model with the highest precision at a given recall target.
  iv. `recall_at_target_precision`— The model with the highest recall at a given precision target.
  v. `loss_function`— The model with the lowest value of the loss function used in training.
- `target_recall=0.9`
  The target recall.
  If binary_classifier_model_selection_criteria is precision_at_target_recall, then recall is held at this value while precision is maximized.
- `positive_example_weight_mult='balanced'`
  The weight assigned to positive examples when training a binary classifier. The weight of negative examples is fixed at 1. If you want the algorithm to choose a weight so that errors in classifying negative *vs.* positive examples have equal impact on training loss, specify balanced. If you want the algorithm to choose the weight that optimizes performance, specify auto.

Refinement

LinearLearner model with hyperparameter “epochs=15” on test dataset generates **Kaggle** public score as 0.50718 whereas when number of epochs decreased to 5 generates Kaggle public score of 0.52613 with rank 50 out of 53.

Model Evaluation and Validation

Initially Amazon SageMaker ‘s LinearLearner with default hyperparameters was used as predictio n model which seems to be over fitted model because of huge class imbalance of data. After fine tuning of hyperparameters as follows:

- **`predictor_type='binary_classifier'`**
- **`epochs=15`**
- **`binary_classifier_model_selection_criteria='precision_at_target_recall'`**
- **`target_recall=0.9`**
- **`positive_example_weight_mult='balanced'`**

The Metrics for LinearLearner:

```
prediction (col)    0.0    1.0
actual (row)
0.0                9867   2861
1.0                 125     36

Recall:     0.224
Precision:  0.012
Accuracy:   0.768
```

Further fine tuning of hyperparameters as follows:

- **predictor_type='binary_classifier'**
- **epochs=5**
- **binary_classifier_model_selection_criteria='precision_at_target_recall'**
- **target_recall=0.9**
- **positive_example_weight_mult='balanced'**

The Metrics for LinearLearner:

```
prediction (col)    0.0    1.0
actual (row)
0.0                9167   3561
1.0                 113     48

Recall:     0.298
Precision:  0.013
Accuracy:   0.715
```

Justification

As per the benchmark model, the first part of solution clustering has to be deployed using KMea ns algorithm. While using direct KMeans algorithm it was observed that it is very difficult to depl oy model directly as it is very time consuming and inaccurate.
So to reduce the number of features Principal Component Analysis is used followed by MiniBatch KMeans to improve computation time.
The second part of solution was supposed to be simple LinearLearner which tends to yield over fi tted model due to dataset imbalance. By customising hyperparameters the accuracy in Kaggle co mpetition has improved from 50.7% to 52.6%.