# The 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'll 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, we'll apply what you've 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.

## Problem Statement

In this project, we'll try to solve 2 business questions:

- How to help Arvato‘s customer which is a mail-order company to acquire new potential customers more efficiently?
- How to make customer dataset more consistent and effective for data analysis?

## Part 0: Get to Know the Data

There are four data files associated with this project:

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

# Part 1: Customer Segmentation Report

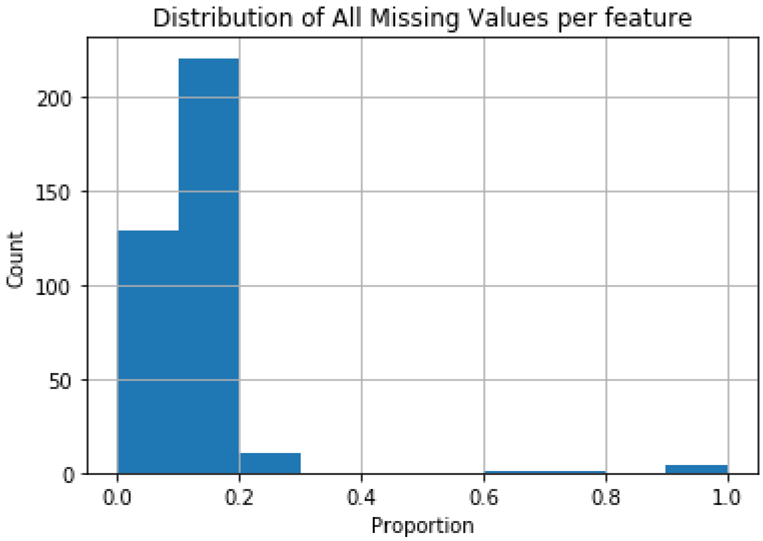
The main bulk of your analysis will come in this part of the project. Here, you should use unsupervised learning techniques to describe the relationship between the demographics of the company's existing customers and the general population of Germany. By the end of this part, you should be able to describe 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.

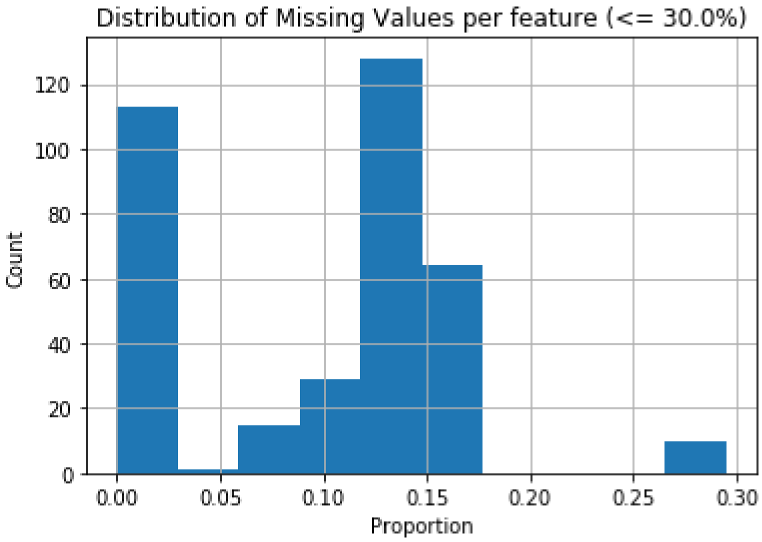
## Section 1: Preprocessing data

1.1: Assessing Missing Data The feature summary file contains all properties per each demographics data column and the properties definitions of additional columns is also provided in this project to be assessed manually. The file "DIAS Attributes - Values 2017.xlsx" will be studied for this assessment and it will help us to make decisions in the project.

1.1.1: Convert Missing Value Codes to NaNs and convert all values to numeric In the current dataset, some features values are incomplete and they can’t be used directly. They should be converted to a new data format which machine can use. The missing features will be added to the original features information dataframe.

1.1.2: Analyze Missing Data per feature Few features are outliers in terms of the proportion of values that are missing. Now Matplotlib hist() function will be used to visualize the distribution of missing value counts to find these columns. For simplicity, these columns will be removed from the dataframe because they can affect data distribution.(Other data engineering tasks such as re-encoding and imputation will be done later.). First we need to delete all empty records and perform an missing data assessment per each feature of the current dataset and analyze missing data patterns of each column by threshold.





a. Based on the histogram of the distribution of All Missing Values per feature, we can see the distribution of the amount of missing data is skewed to the right. Most of the features have about 30% of missing data on the histogram; less of features have greater than 30% of missing data.

b. Based on the histogram of Distribution of Missing Values per feature(<=30%), it's a bimodel distribution and there are two patterns: the first pattern is an almost single bar for no missing values; the second pattern has a slightly normal distribution for missing data < 30% but not equal to 0 and the features which are related have missing values with almost similar proportions.

Then we remove some outlier features from the azdias dataset which can affect our model prediction.

### 1.2: Choose and Encode Features

The most common unsupervised learning model is for cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. Since the unsupervised learning techniques will only work on data that is encoded numerically, there are some cases need to to handled: First, a few encoding changes or additional assumptions will be made. Second, almost all of the values in the dataset are encoded using numbers and but not all of them represent numeric values. some values of feature still use characters. Last, the features dictionary will be checked for a summary of types of measurement.

After checking the azdias dataset, we need to focus on those things:

1) In the azdias dataset, there are types of some features which are unknown or string. They must be identified first and encorde them.

2) The features with numeric and interval data can be kept without changes.

3) Most of the variables in the dataset are ordinal in nature. The ordinal values may be non-linear technically in distribution space, we can make an assumption that the ordinal variables can be treated as being interval in nature.

4) For the variable types: categorical and mixed-type, we need a special handling.

Based on the previous investigation, our implementation steps are like that:

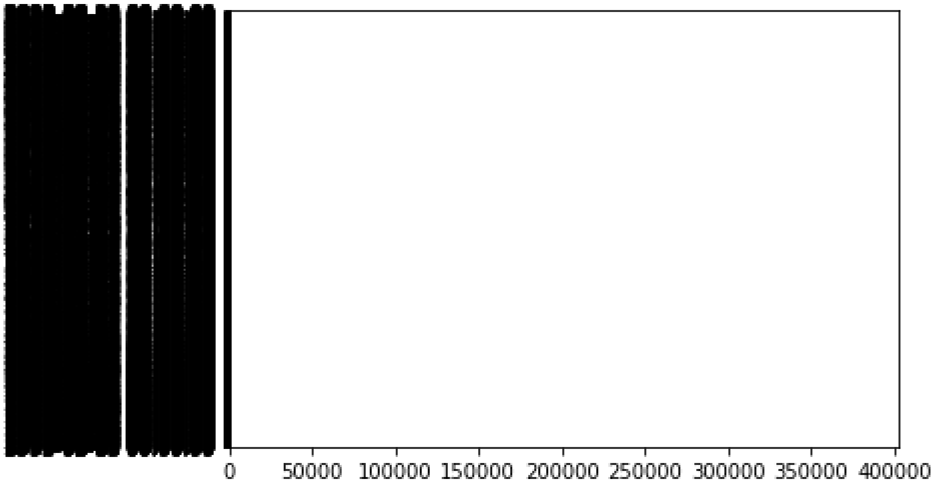
First, the features which have unknown types will be examined and assumptions will be made.

Second, an investigation of the categorical and mixed-type features will be investigated and decide whether they will be kept, dropped, or re-encoded.

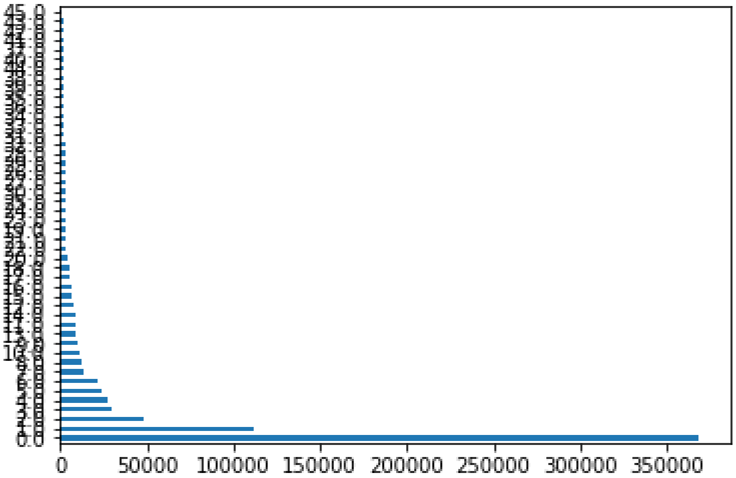
Finally, in the last part, a new data frame will be created with only the selected and engineered columns.

From the previous list, there are 360 of ordinal and categorical and numerical features and 6 of string features. According to our agreement, we need to keep those values.

For example, This 'EINGEFUEGT_AM' category has many levels which may not be captured accurately.
azdias_sub['EINGEFUEGT_AM'].value_counts().plot(kind='barh')



This 'VERDICHTUNGSRAUM' category has many levels too and, from that chart, we can see most of 'VERDICHTUNGSRAUM' values are 0.0 and so it's not good for data analysis.



There are three features which will be dropped:

- LNR - This is unique for each row so it is not useful for machine learning.
- EINGEFUEGT_AM - This can be categorical but there are many levels which may not be captured accurately.
- VERDICHTUNGSRAUM - This can also be categorical but there are many levels too.

In addition, they don't also exist in the DIAS Attributes - Values 2017.xls file so the solution is to drop them.

### *1.2.1 Convert NAN values for numerical/ordinal/categorical features(numeric)*

For categorical data, 'NAN' values exist in lots of features. They need to convert to numeric value and then they can be applied to machine learning model for data analysis.

- Numeric features: using 0 instead of 'nan' values
- Ordinal features: using 'Unknown' numeric values instead of 'nan' values
- Categorical features: using 'Unknown' numeric values instead of 'nan' values

In the feature_dict dictionary, there are some 'Unknow' values which contains two values(e.g. -1,9). we can use one of them. In my project, I'll use first value instead of 'nan'

### *1.2.2 Convert values for categorical features(string)*

There are two string category features:'CAMEO_DEU_2015', 'D19_LETZTER_KAUF_BRANCHE'. Their values can't be analyzed directly. we need to convert them based on their definition in feature_dict dictionary.

- using 'Unknown' numeric values instead of 'nan' values

### *1.2.3: Re-Encode Categorical Features*

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

- For binary (two-level) categoricals that take numeric values, they will be kept without needing to do anything.
- There is one binary variable that takes on non-numeric values. For this one, the values will be re-encoded as numbers.
- For multi-level categoricals (three or more values), the values will be encoded using multiple dummy variables.

Re-Encode Categorical Features

- The binary features have different binary numeric values and so re-encoding has the benefit of not weighting a value improperly.
- The rest of the multi-category features were re-encoded using one-hot-encoding through the use of the pd.get_dummies function from pandas. The number of columns will increase due to the additional columns created by the one-hot-encoding step.

### *1.2.4 Assess Undifined Features*

In the dataset, some features are without its definition. we need to clean up them. CAMEO_INTL_2015 - it contains numeric and strinng values and but its definition doesn’t exists in feature definition file.

#### 1.2.5 Build Clean-up Function to clean up azdias dataset

## Section 2 Feature Transformation

Feature transformation involves mapping a set of values for the feature to a new set of values to make the representation of the data more suitable or easier to process for the downstream analysis. A common feature transformation operation is a scaling.

### 2.1 Apply 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 was done by applying an Imputer to replace all missing values.
- For the actual scaling function, a StandardScaler instance was done, scaling each feature to mean 0 and standard deviation 1.
- For these classes, the .fit_transform() method was used to both fit a procedure to the data as well as apply the transformation to the data at the same time.

Apply Feature Scaling: There are 349223 records with missing values which is about 39% of 891221 records and they should not be dropped. So, there are several ways to handle missing value:

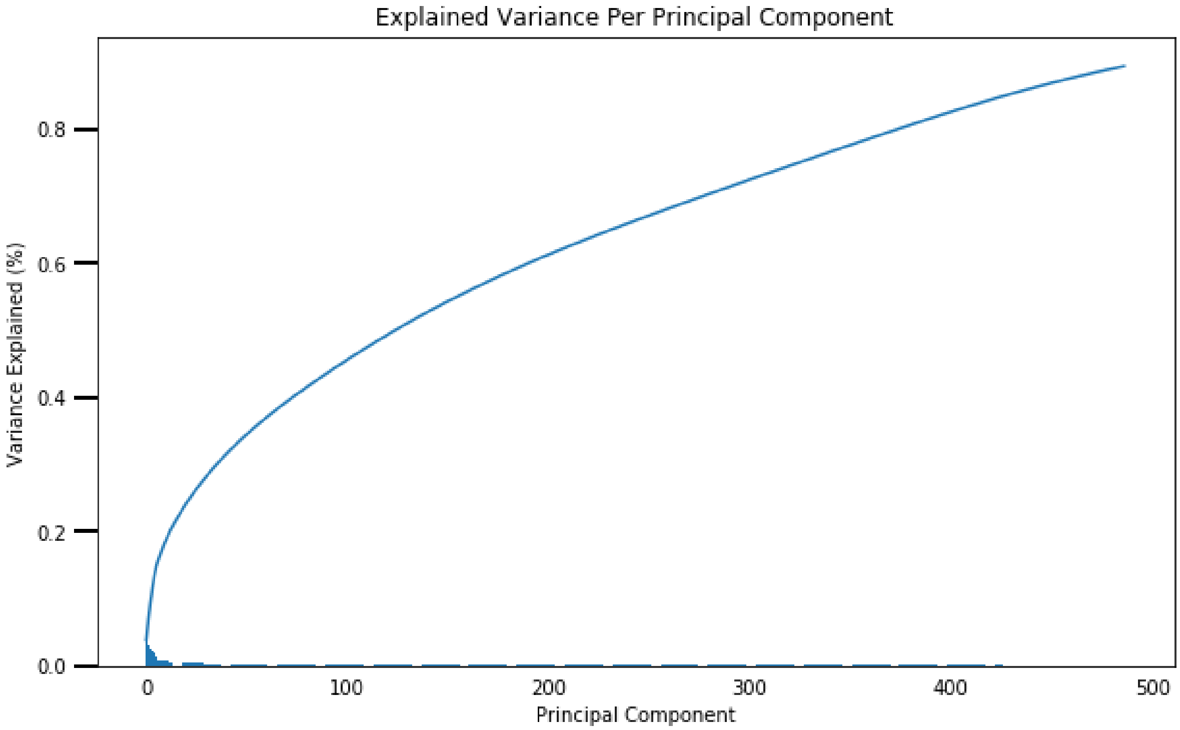
- all missing values were replaced using 'unknown' value per feature if there are missing values.
- all missing values were replaced by the mean value which is provided by Imputer along all features(columns) of dataset if there are missing values.

Then the Feature scaling is applied by StandardScaler instance of sklearn and its '.fit_transform()' method fits to the data and transform it at the same time. The StandardScaler instance returns an array list for the next step Performance Dimentionality Reduction - PCA.

### 2.2 Perform Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

- Sklearn PCA model will handle principal component analysis on the data, thus find the vectors of maximal variance in the data.
- The ratio of variance explained by each principal component as well as the cumulative variance explained will be checked by plotting the cumulative or sequential values using matplotlib's plot() function.
- PCA model will be re-fit to perform the decided-on transformation.



After appling PCA to the dataset and check the variance ratio of components, when using mean instead of missing values, its explained variance is 0.8932687682414621 (when using "unknown" value instead of messing values, its explained variance is 0.8955750104090971)

Perform Dimensionality Reduction

- As analyzed by PCA, 50% of features (487 out of 974) was used as the initial number of components for the principal component analysis of the data. The 487 components can explain about 89.33% of the variability in the original dataset. This is good enough for further analysis and no further re-application is necesssary.

### 2.3 Interpret the key results for Principal Components Analysis

Principal components analysis is based on the correlation matrix of the variables involved, and correlations usually need a large sample size before they stabilize. The principal components have been transformed and the weight of each variable on the first few components should be checked 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),which are correlated. 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.

***Dimension 1:(display top 5 features)***

- D19_KONSUMTYP_MAX (0.1048) - consumption type
- VK_DHT4A (0.1017) - no information
- D19_VERSAND_DATUM_10 (0.0998) - actuality of the last transaction for the segment mail-order TOTAL
- VK_DISTANZ (-0.0976) - no information
- D19_GESAMT_DATUM_10 (-0.0966) - actuality of the last transaction with the complete file TOTAL

Interpretation: The first principal component is strongly correlated with consumption type to ‘none’(VK_DHT4A). D19_VERSAND_DATUM_10 is described in the attributes file but it can be related to consumption type. Higher consumption and no transactions known tend to negatively affect this first principal component.

***Dimension 2:(display top 5 features)***

- KBA05_MAXAH_9.0 0.1641 - most common age of car owners in the microcell(unknown)
- KBA05_SEG6_9.0 0.1641 - share of upper class cars (BMW 7er etc.) in the microcell(unknown)
- KBA05_MAXBJ_9.0 0.1641 - most common age of the cars in the microcell(unknown)
- KBA05_MAXHERST_9.0 0.1641 - most common car manufacturer in the microcell(unknown)
- KBA05_MAXSEG_9.0 0.1641 - most common car segment in the microcell(unknown)

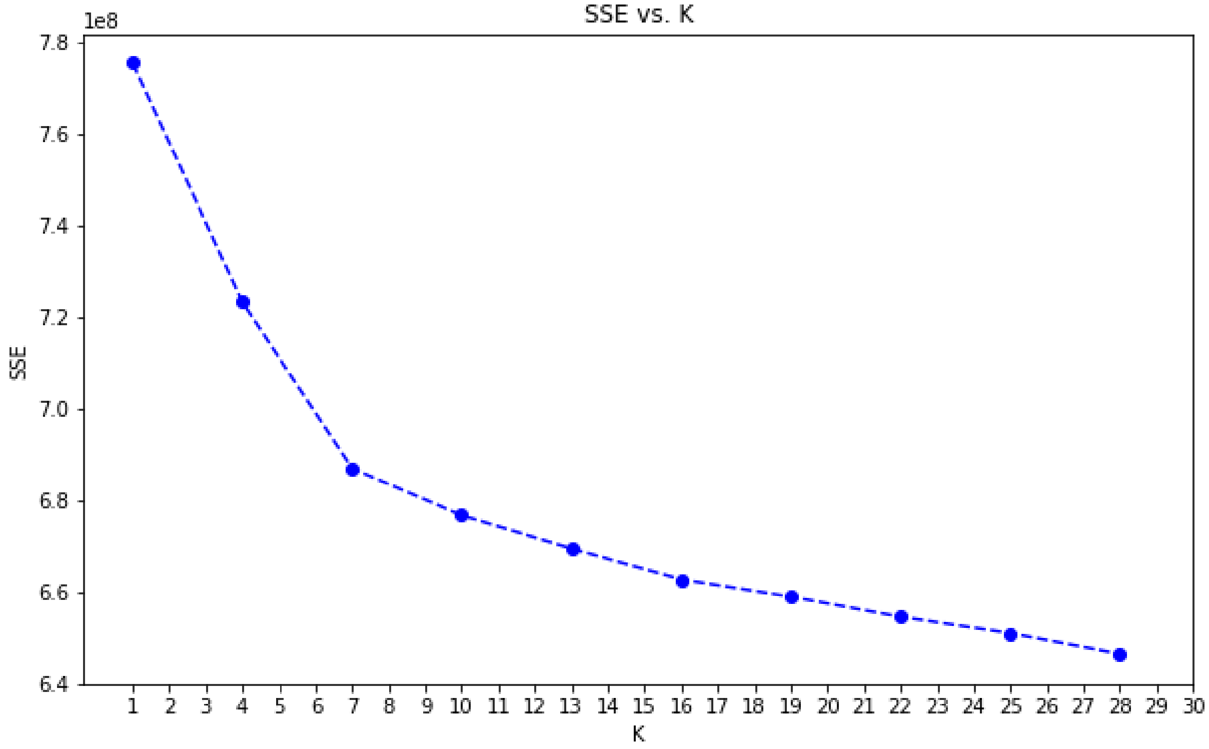
Interpretation: The second principal component is strongly correlated with most common age of car owners in the microcell and share of upper class cars (BMW 7er etc.) in the microcell and most common age of the cars in the microcell and most common car manufacturer in the microcell and most common car segment in the microcell. They tend to negatively affect this second principal component.

## Section 3 Clustering data

### 3.1 Apply Clustering to General Population

Now, it's time to see how the data clusters in the principal components space. In this substep, k-means clustering will be applied to the dataset and the average within-cluster distances from each point to their assigned cluster's centroid will be used to decide on a number of clusters to keep. sklearn's KMeans class will be used to perform k-means clustering on the PCA-transformed data.Then, the average difference from each point to its assigned cluster's center will be computed.The above two steps will be performed for a 30 different cluster counts to see how the average distance decreases with an increasing number of clusters. Once final number of clusters to use is selected, KMeans instance will be re-fit to perform the clustering operation.

The scree plot shows that the score or the sum of the squared errors (SSE) generally decreased as the number of clusters increased. As the instruction suggested, the maximum clusters used was 30. The 'elbow method' is not applicable in the plot because there is no visible leveling observed. Even though 30 clusters did not produce the lowest SSE, it was still used as the number of clusters for the full KMeans clustering operation.

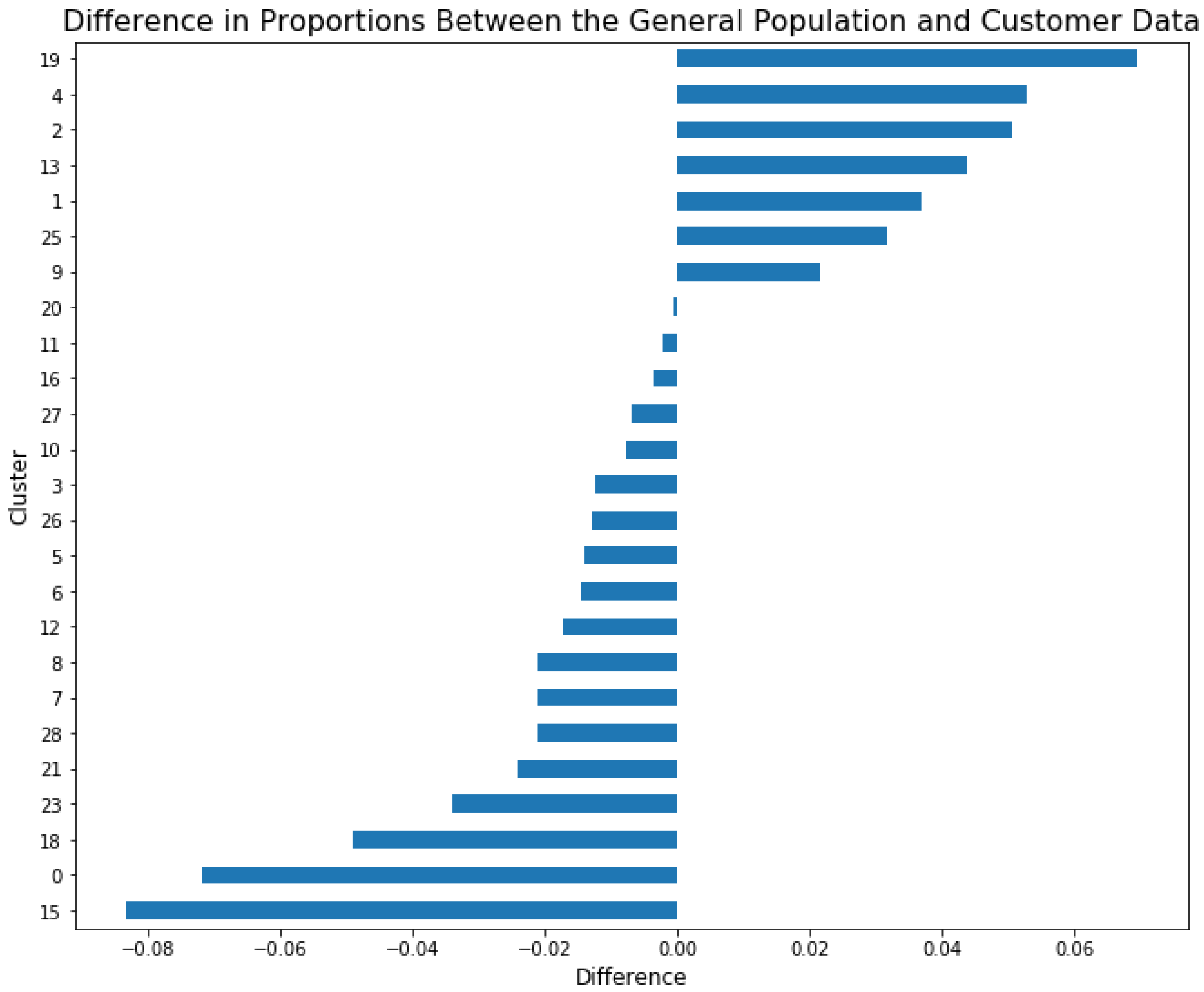


### 3.2 Analyze the Customer Dataset

We'll apply the clusters and cluster centers of the general population to the customer data. Also The general population must be cleaned up, transformed, and clustered. At the end, we'll analyze how the general population fit is appled to the customer data.

### 3.3 Compare Customer Data to Demographics Data

The clustered data is based on demographics of the general population of Germany, and the customer dataset which is from a mail-order sales company has already been mapped to those demographic clusters. At the end, the distributions of these two clusters will be compared to check where the real customer group for the company is. We'll show the difference of proportions between the general population and customer dataset:



***Compare Customer Data to Demographics Data***

Since there are over 900 features, it's impractical to check all features to interpret the two clusters. Based on the previous principal component interpretations, the columns ('D19_KONSUMTYP_MAX','VK_DHT4A','D19_VERSAND_DATUM_10','VK_DISTANZ','D19_GESAMT_DATUM_1 to be interpreted on the original data of the chosen clusters can be identified. The target customers make transactions by other ways('no transactions known').

# Part 2: Supervised Learning Model

Now that you've found which parts of the population are more likely to be customers of the mail-order company, it's time to build a prediction model. Each of the rows in the "MAILOUT" data files represents an individual that was targeted for a mailout campaign. Ideally, we should be able to use the demographic information from each individual to decide whether or not it will be worth it to include that person in the campaign.

The "MAILOUT" data has been split into two approximately equal parts, each with almost 43 000 data rows. In this part, you can verify your model with the "TRAIN" partition, which includes a column, "RESPONSE", that states whether or not a person became a customer of the company following the campaign. In the next part, you'll need to create predictions on the "TEST" partition, where the "RESPONSE" column has been withheld.

There are 42,962 individuals in the mailout dataset, but only 1.24% of the individuals which become customers. so the dataset is highly imbalanced.

## Section 1 Prepare Data

First step, it often must be cleaned, transformatted, and restructured — this is typically known as preprocessing. Then we can use the same cleanup dataset function to clean up the Mailout dataset.

After checking the Mailout dataset, its features are similar to the general population dataset and the customers dataset. So we can use cleanup_dataset function to clean up the Mailout dataset.

## Section 2 Process and Split Dataset(Training dataset and Testing dataset)

The mailout dataset has been preprocessed and it's ready for machine learning, which split the data into features and their labels. Based on the preprocessed mailout dataset, we'll split it into training dataset and testing dataset. For class imbalance, we can implement a variation of KFold Cross-Validation which returns stratified folds. The folds made will preserve the percentage of samples for each class.

## Evaluating Model Performance

If there is a large output class imbalance, for predicting individual classes and using accuracy, it does not seem to be an appropriate performance evaluation method. ROC-AUC (which Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores) will be implemented in the model to evaluate performance.

### Improving Model Results

In this supervise learning model, the model `GradientBoostingRegressor` is applied to this customer dataset and the `GradientBoostingRegressor` model is based on a small margin.

Actually, Extreme Gradient Boosting is an advanced and more efficient implementation of Gradient Boosting so it makes sense that it should have the highest score. Next step, we'll use `GradientBoostingRegressor` to build our supervised learning model. But we have to improve its parameters:

- Tune `max_depth` and `min_child_weight` These will be tuned first because they have the highest impact on the model outcome.
- Tune `min_samples_leaf` and `min_samples_split`
- Tune Regularization Parameters
- Reduce `learning_rate` and increase `n_estimators`
- Check Most Important Feature from training dataset

## Part 3: Kaggle Competition

the supervised learning model which was created in second part is used to predict which individuals are most likely to respond to a mailout campaign and to test that model in the class competition through Kaggle. The finely-tuned model was used to fit the complete training dataset to predict the response variable of the testing dataset.

## Part 4: Future Improvements

In supervised learning model, now I used sklearn pipeline and GradientBoostingRegressor model to train and test customer dataset. But its accuracy of prediction is not good and it can reach about 80%. I think I can use deep learning on customer dataset. We know the knowledges of Deep Learning is better and it’s been applied to some fields(NLP, Image Recognition etc.). For the data transformation, we can use median or gaussian function to replace missing data.

This project is hosted in Github through this [link (https://github.com/JerrySmithLu/Capstone)](https://github.com/JerrySmithLu/Capstone).