# Project Overview

The idea behind this project is to analyse what are the features that best identify potential customers for a given company. In order to do so, two datasets are provided, one that contains information of the whole population, and a second dataset with information about real customers. By studying these two datasets using unsupervised learning techniques it will be observed what are the particularities of customers when compare to the whole population. These findings, in turn, will enable better assess whether a new individual may be a potential customer. By using supervised machine learning algorithms, a recommendation on whether to target a new individual or not will be given.

In short, the purpose of this study is twofold. First, to help a given company achieve a better understanding of what their current customers are, what is their profile and what are the features that set them apart. Second, to help them be more efficient in their targeting campaign by using ML techniques that will assign certain probability to a new individual become a future customer.

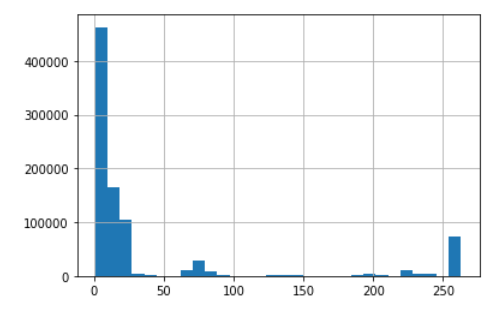
# Data Exploration

The datasets provided were very dirty and there were a few differences between the customer and the population dataset in terms of number of features. The customer dataset contained three additional features that are specific to customers and help understand what type of customer a particular individual is. For example, it may help the company understand differences among customers to select different marketing campaigns depending on type of customer. However, these three features are not relevant as far as unsupervised learning is concerned. Also, LNR column, analogous to individual id, does not have any predicting power as it’s a random number assigned to each individual. Hence, the first step to take is to align both datasets so they contain the same number of features.

The next thing to note, is that datasets contained duplicate entries that had to be removed. Duplicate entries do not have a positive impact on algorithm prediction and so it is rather an indication that data is dirty.

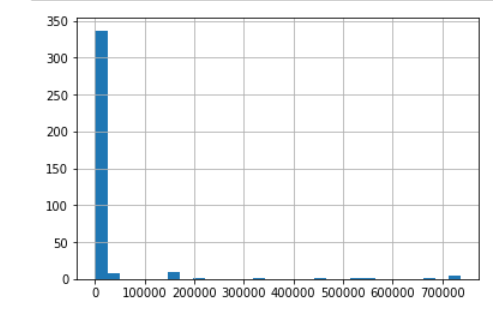
Another important task when doing data exploration is to look at missing values. Missing values in these dataframes take two forms. One is represented by NaN values that clearly indicate a value is missing. The other representation for missing values is through an actual number that indicates the value is missing. The approached followed was to look for these numbers that represent missing values and replace those with NaNs. This, leads to a more consistent approach later one when deciding what to do with missing values. In order to enable this task, a csv file was manually generated. In this file a mapping between feature and invalid value was generated. This csv file was read as a dictionary and values were correctly replaced for NaNs.

Once the previous task was completed, data was more thoroughly inspected. Some histogram plots like the one below massively helped understand the distribution of missing values



In the plot above it can be observed that most individuals for the population dataset are missing not more than 50 features, and it’s only a few individuals that can be regarded as outliers. The other dataset showed a similar behaviour. In order to ease the analysis and considering that those individuals missing more than 50 features do not offer a significant amount of information to enhance the predicting power of the algorithms it was decided to drop these individuals from both datasets.

Interestingly, when dropping these individuals the number of columns where lots of individuals are missing is heavily reduced. After the previous step of dropping individuals it could be observed that most features were either complete or almost complete as shown in the graph below



The graph above shows that most features contain very few or no missing individuals. Those features that are missing more than 100k individuals are really an exception or outlier. For consistency, it was decided to drop all those features that were missing more 100k individuals.

After all these steps there were still 354 features taken for analysis, so the amount of information dropped was kept to a minimum. It has to be said that columns to be dropped were different between customer and population datasets and so the union of both was taken when deciding which features to drop from the analysis.

Thus, both dataframes have now no duplicates, outliers have been removed and missing values all have the same shape. Missing values imputation could be the next step, but the problem encountered was that column types were different and so it was very hard to follow a programmatic approach to imputing missing values. Instead, columns that were not floats were analysed to find out what type of column was and whether it makes sense to change the contents to numerical values. It was found out that some columns with numerical values were truly categorical and it was better to use a one hot encoding for these types of columns. The problem with one hot encoding is that datasets already had a big number of columns and using one hot encoding for every single categorical value would have made matters much worse. Therefore, one hot encoding was only used for those features that did not have any numerical meaning. Those that expressed ranking in some sort of form were kept as numerical. It was also observed that some other features encoded several bits of information in one number. It was decided to split these features into as many as pieces of information they were encoding. At the end of the process all columns could be converted to type float and imputing missing values could start.

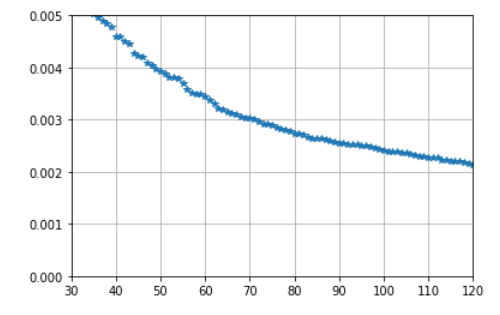
There were several approaches considered when imputing values. Dropping that individual or feature is the simplest alternative. The problem is that overall there were many missing values and this would have caused a great deal of information being lost. The second alternative would have been using a single value (either mean or median or any other alternative) to impute all missing values for a given feature. This approached was considered too simplistic and introduces, in my opinion, a big bias into that feature. With these two approaches discarded there were two more that were taken into consideration. Performing a bootstrap sample of as many as missing values for a particular feature. In this case the distribution of values for the whole feature would not change. The other approach was to use machine learning to predict the missing value for a given individual. The algorithm could use all features that contain valid information for that individual and create a prediction model using a random forest classifier to come up with an informed prediction of a potential value for that missing cell. This approach was considered the best approach as it’s able to leverage the power of machine learning prediction to impute missing values.

In the Jupyter notebook more detailed information can be found with regards to each step described in this section.

# Group segmentation

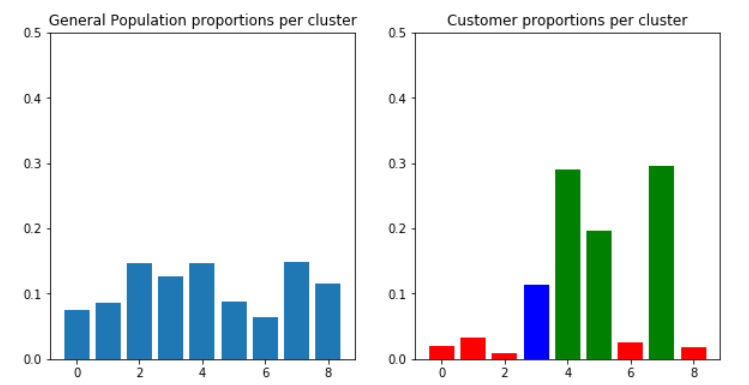
The idea of this section is to describe the procedure used to identify the features that differentiate customers from other groups of a wider population. Unsupervised learning techniques come in handy for this exercise. Using clustering techniques the population can be split into different groups or clusters each of them with their own particularities. It can be seen in the Jupyter notebook for example that one group of customers can be described as middle age people, with above average income leaving in densely populated areas. There are more traits to each cluster this is only a very simplified example. Different groups have their own characteristics and so the idea is to split the whole population in clusters. This sheds light into roughly how many different groups there are in the population. Using the same clustering process, the customer population can be segmented too and the weight that each group plays into each dataframe can be analyse. Thus, it can be claimed that those clusters that are overrepresented in the customer dataset compared to the population dataset are the main customer groups. The cluster where each individual belongs will therefore play a role in predicting how likely a new person will be a customer for the company. Cluster number needs therefore be included as an engineered feature for the following section.

The problem is that clustering is a computationally intensive exercise. The fact that the datasets contains so many features will only prove detrimental for splitting the population into groups. In order to speed up the whole clustering process without sacrificing much information, principal component analysis (PCA) comes to the rescue. PCA linearly combines features to form a single feature, thus reducing the dimension of the dataset and in turn improving time taken to compute the clusters. The statement that no information will be lost when using PCA, is not entirely correct. If it’s decided to keep only a certain number of principal components, then all those that are discarded will contribute to some degree of information loss. However, the amount of information loss can be limited. For example, in this particular case and as shown in the graph below, keeping 70 components means that each of the components discarded will contribute with less than 0.3% of information loss



Performing a PCA analysis on the datasets also adds a bit of extra work when it comes to explaining what features contribute to explain what company’s customers are like. For example, feature selection translates into principal component selection which in turn is a linear combination of features. For each component there needs to be an analysis that explains what are the features that contribute the most towards that component. Therefore, the analysis can be more tedious, and also the main characteristics of groups are harder to decode.

The Jupyter notebook does a good job to explain what the characteristics of the customer groups are and delves into analysing the main characteristics. As a brief finding, the following plot illustrates that the whole population can be roughly split into 9 different clusters. When analysing the customer dataset and performing a clustering similar to the population group, it can be observed that three clusters seem to be overrepresented. These clusters will be the customer base of the organization and these are the people that they should be targeting when for example they decide to design a marketing campaign.



In order to keep things simple, if those three customer groups were to be combined, the prototype of customer would be a middle age person with a comfortable financial situation. This person is somewhat present online but far from younger generations. They live in middle-class areas and have a stable work location. This means that targeting a 25-year-old who has just started to work may not be the best strategy the company can make.

# Classification algorithm

The last step of this study is to use what has been learned before to build a machine learning model able to predict whether a new individual will become a customer or not. As previously mentioned, we can add another piece of information to input data to this algorithm and that is the cluster they belong to. Cluster number can be a good predictor and so it makes sense to include in this classification algorithm.

In order to create the model a new dataset was shared. This new dataset contained as many features as datasets in the previous section but in addition to that it contained another column that indicated whether that individual became customer or not. This dataset, however, needs to be wrangled in the same way the previous two datasets were cleaned. Therefore, all the process of identifying missing values, dropping outliers and imputing values was followed. Then, the transformers and predictors in the previous section were reused to identify the cluster that individuals belonged to.

Even though the Jupyter notebook only shows a gradient boosting model, this only means that this is the model that was finally selected as it outperformed the other options. Grid search was also used to make sure the hyperparameters, that make the model worked best, were selected.

With regards to grid search, it’s important to talk about the metric that was used to fine-tune the model. The problem with the dataset is that it was highly unbalanced. This is generally not a very good thing for classification algorithms as they will overfit the predominant class. Accuracy is obviously not a very good metric as predicting always the predominant class will give the model a pretty good accuracy, but the model is not very good at generalization. It was therefore decided to optimize for recall as the idea is to get as many 1s correct as possible.

Worth mentioning that when training the dataset with the full dataset given for training, then recall score was 0. This is because there is a high bias in the model to predict the negative class as they are an overwhelming majority. In order to tweak the model parameters the dataset was engineered so the ratio of 0s to 1s was only 3:1.

With this in mind, probabilities for the test dataset were obtained and fed into Kaggle and no difference was observed in the final score when compared to the case where the full dataset was used for training.

The score obtained of around 0.7 means that the algorithm is much better at predicting whether an individual will become a customer than pure chance alone.