Capstone Project

Customer Segmentation Report for Arvato Financial Solutions

Machine Learning Engineering Nanodegree

Background:

This is a blog I have written as a final project submission for my nano degree program with Udacity in Machine Learning Engineering.

Problem Statement:

The main goal of this project is to understand how to acquire more clients or customers efficiently. This was the job of Subject Matter Experts who used to decide on the basis of the experience and business knowledge only. Objective of this project is to help them in making these decisions on the basis of data.

There are majorly 3 components to this project:

- Customer segmentation report
- Supervised learning model
- Kaggle Competition

Customer Segmentation Report

This part will contain the analysis of customer attributes and compare it with the general population. Objective is to identify similar users from the generic population to the company's customer base and optimize the sales cost. Unsupervised learning methods will be used in order to create customer segments.

Supervised Learning Model

This part will be the prediction step where we will try to understand the propensity of a user to become our customer. We will build a machine learning model that predicts whether or not a particular individual will respond to the campaign.

Kaggle Competition

This is the validation step for the model we build in step 2. We will upload our prediction on kaggle and check our scores.

Dataset and Inputs

We were provided with the following 4 data files for 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).

First two files i.e. **AZDIAS** and **CUSTOMERS** will be used in segmentation while **MAILOUT TRAIN** and **TEST** will be used to build our supervised learning model. Each row of the demographic files represents a single person, but also includes information outside of the individual, including information about their household, building, and neighborhood. We used the information from the first two files to figure out how customers (CUSTOMERS) are similar to or differ from the general population at large (AZDIAS), then use this analysis to make predictions on the other two files (MAILOUT), predicting which recipients are most likely to become a customer for the mail-order company.

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. The TRAIN subset of the data contains RESPONSE variables while the TEST subset don’t. We will do the final prediction corresponding to each test row and evaluate the performance in the Kaggle competition.

We were also provided 2 data dictionaries i.e. One of them(***DIAS Information Levels - Attributes 2017.xlsx***) is a top-level list of attributes and descriptions, organized by informational category. The other(***DIAS Attributes - Values 2017.xlsx***) is a detailed mapping of data values for each feature in alphabetical order.

Solution Statements

We have attributes and the demographic details of the existing customers. Idea is to analyze and understand this behavior and see if we can find similar behaving customers from the general population of Germany. This helps the company to identify new potential customers for the company. It helps in bringing efficiency in the customers acquisition process. Instead of reaching out to everyone in the country and blowing the sales budget.

The solution will be comprised of following 3 deliverables:

- **Customer Segmentation Report:** This will be a Jupyter Notebook with markdown notes and visualizations.This notebook will contain all the codes for pre-processing and feature engineering of the data.
- **Supervised Learning Model:** This will be a supervised model object that uses demographics attributes to segment people into potential customers or non-potential customers.
- **Blog post:** This document is the one!

Data Processing

This is the first section of the project where we tried to understand the data and prepare it for segmentation and modeling. Following are the major steps performed:

Convert unknowns to nulls and cleaning

- There are columns like **CAMEO_DEUG_2015** which have 'X' as unknown. Replace **'X'** with **NAN**. Also convert that into float to avoid having 2 different values for the same value. Same is the case with value **-1**.
- Column **CAMEO_DEU_2015** has **'XX'** which was also replaced by **NAN**.
- In columns starting with **'D19_'**, For the value **'no transaction known'**, it is filled with 0 or 10. To bring sanity, we will replace all the 10 with 0s for all the columns.

Encoding categorical columns

- Few columns like **D19_LETZTER_KAUF_BRANCHE**, **EINGEFUEGT_AM** are not available in the data dictionary and have a lot of categories to perform one-hot encoding, so we will drop them.
- **Encoding Columns**: **CAMEO_INTL_2015**, **CAMEO_DEU_2015**, **OST_WEST_KZ**
- **CAMEO_INTL_2015** has 2 components i.e. **wealth** and **life_stage**. We extracted them and created 2 columns corresponding to **CAMEO_INTL_2015**. Similar processing was done for **CAMEO_DEU_2015**.
- **OST_WEST_KZ** is a flag indicating the former GDR/FRG. Since it has only 2 unique values, we convert that into boolean

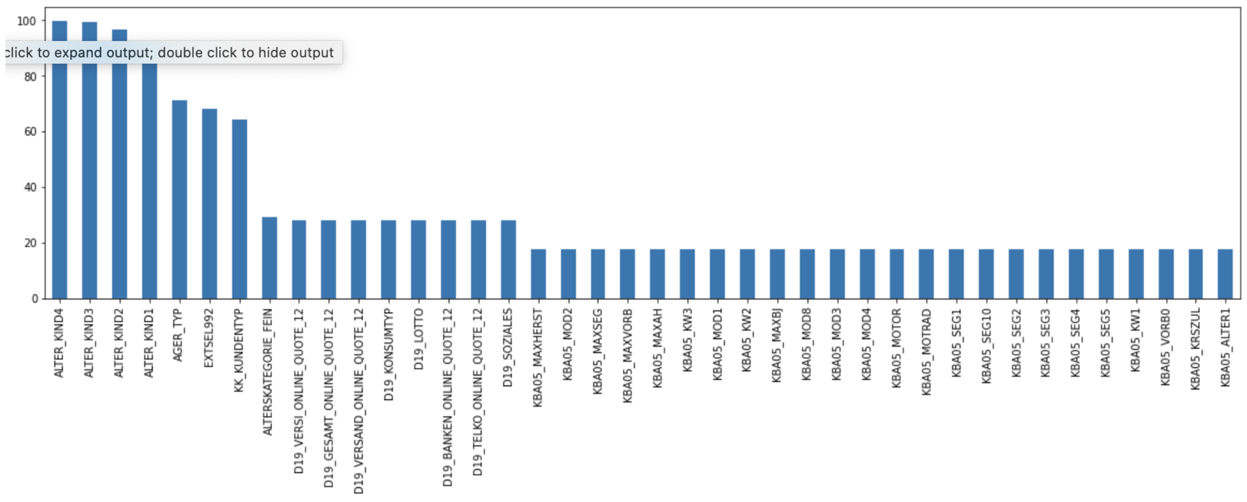
- Drop the year column (**MIN_GEBAEUDEJAHR**) since we should not directly use this as a feature. It can be utilized by creating a feature on top of it.

Encoding numerical column

- There is one more column called **PRAEGENDE_JUGENDJAHRE** which means dominating movement in the person's youth (avantgarde or mainstream). This feature contains 2 pieces of information i.e. decade and movement. We extracted them with the below mapping
- **Decade mapping**: {1:40, 2:40, 3:50, 4:50, 5:60, 6:60, 7:60, 8:70, 9:70, 10:80, 11:80, 12:80, 13:80, 14:90, 15:90}
- **Movement mapping**: {1:0, 2:1, 3:0, 4:1, 5:0, 6:1, 7:1, 8:0, 9:1, 10:0, 11:1, 12:0, 13:1, 14:0, 15:1}

Remove Null majority rows and columns

We removed all the columns which have more than **25%** missing values. Same action was performed on the rows which have more than **50%** missing values.



There were 16 columns which were dropped.

Customer Segmentation

As discussed previously, This part will contain the analysis of customer attributes and compare it with the general populations. We concatenate the dataset of customers with the general populations and perform clustering on the whole dataset. Here are the steps:

1. Join both the base i.e customer and general population.

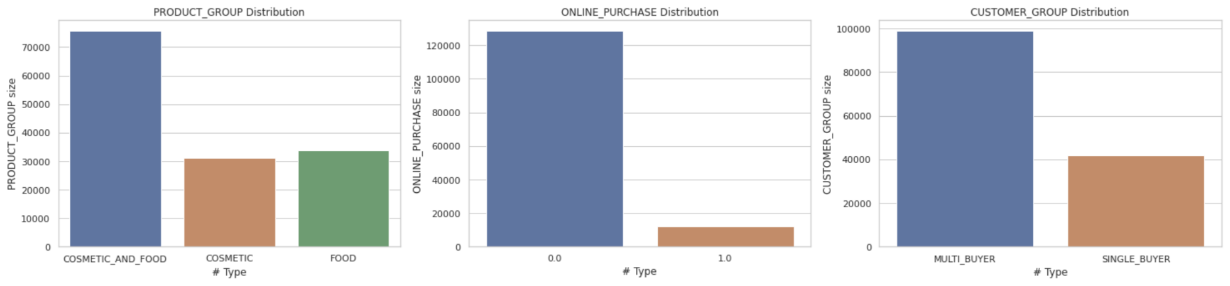
2. Pre-process the data which we discussed in the previous section.
3. Remove the extra columns present in the customer dataset.
4. Find the optimum K using elbow method and silhouette analysis.
5. Apply k-means clustering on the whole combined base
6. Cluster analysis.

Idea is to find the clusters where maximum existing customers exist. Our main objective is to identify customers from the general population who have similar behavior and demographics to the existing customers of the company.

Since we have around 300 features,there are 2 options.

1. Perform PCA on the data and then cluster the principal components. The issue with this approach is we will lose some information and interpretability would become a concern. Moreover, even after reducing the dimensions(features), we will still have a lot of features.
2. To overcome the issues that exist in the previous step, we will go directly with the clustering since 300 features are not really a very high dimension.

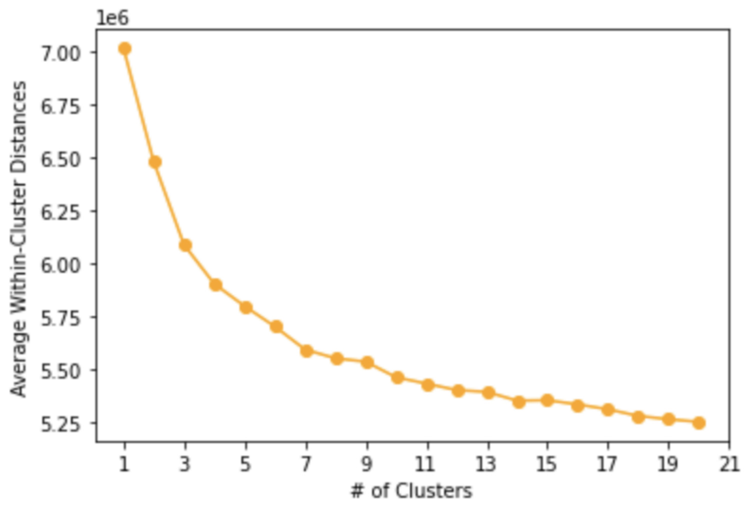
Since we already know that customer data has few extra columns like CUSTOMER_GROUP, ONLINE_PURCHASE, PRODUCT_GROUP. We removed these columns before performing clustering. Here is their Distribution:



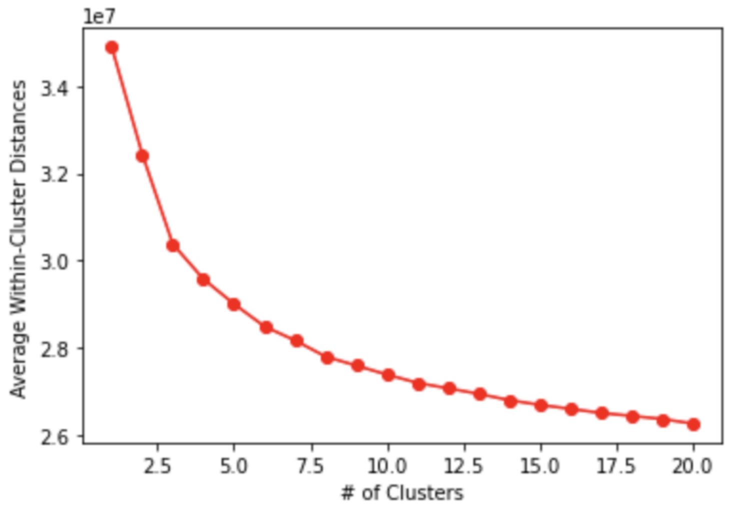
To use the K-Means algorithm, we can not have nulls in the data and also, all the features should be standardized.

Imputing NAN: We have used **SimpleImputer**() to fill the null values and used **most_frequent(mode)** type to fill it.
Normalizing data: We have used **StandardScaler**() of sklearn to normalize the data

Next step was to find the optimum K i.e. number of clusters. For that we have used the **ELBOW Method**. We took the initial range of 2-10. And run the K-Means only on the random smaller subset of the data of size 10k. This subsetting was done to fasten the process. Here is the K vs dispersion in the cluster.



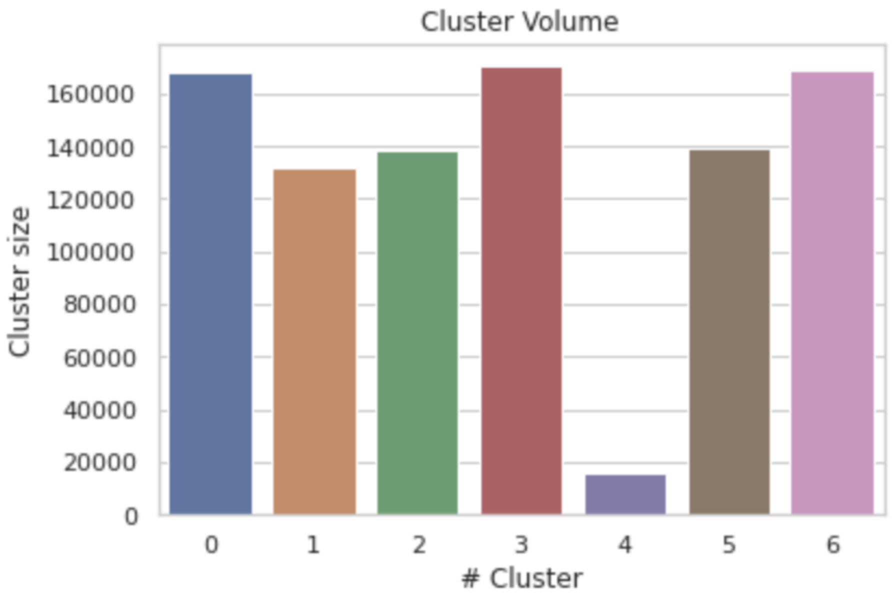
We re-ran the elbow method on the same range but with a bigger sample(100k) to check if k is still the same. Here is the graph:



Above was done to build confidence in terms of the number of clusters(or different groups of customers) present in the data. From the above 2 analysis, it is evident that the number of clusters should be in the range of (6-8). We also calculated silhouette scores for them. Since, there is no significant difference between the dispersion score, we will go ahead with the 7 as the final number of clusters.

Now, let's run the K-Means clustering algorithm on the whole dataset to get the final 7 clusters.

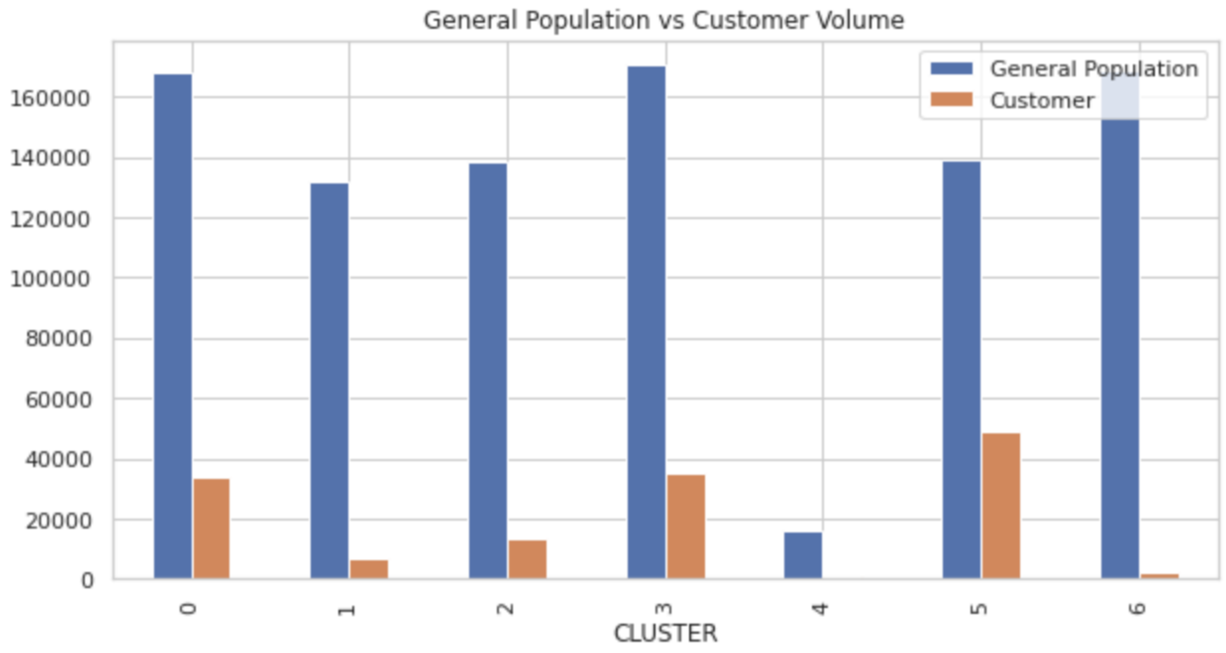
Here is the cluster Volume distribution:



Cluster Analysis:

Once we segmented the users using clustering, we went ahead and did some analysis. As we know that customer dataset have 3 extra columns i.e. 'PRODUCT_GROUP', 'CUSTOMER_GROUP', 'ONLINE_PURCHASE' which are not available in the general population of Germany. We tried to understand the 7 clusters from these 3 features perspectives.

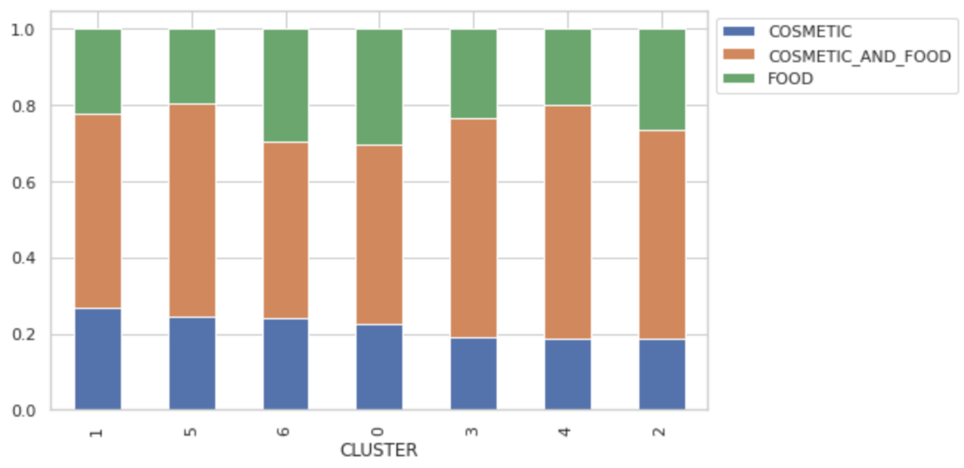
Cluster Analysis: Customer vs General Population Distribution



Above stacked plot represents the volume of existing customers in every cluster. Blue represents the general population and orange represents existing customers. It is evident that cluster 0, 3, and 5 are the most representative clusters of customer demographics.

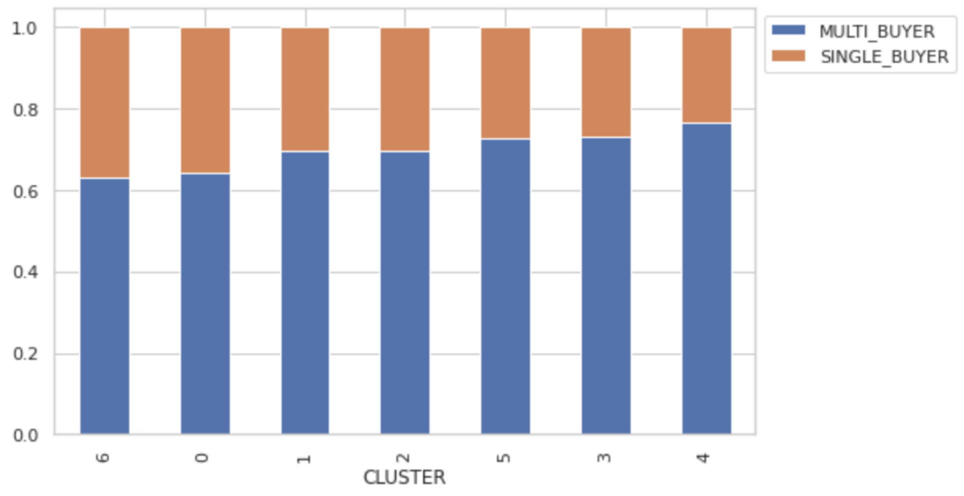
While targeting the population for various products, Arvato should target these 3 segments. Since other clusters are very different from the company's customer base, the company can save marketing costs here.

Cluster Analysis: PRODUCT_GROUP Distribution



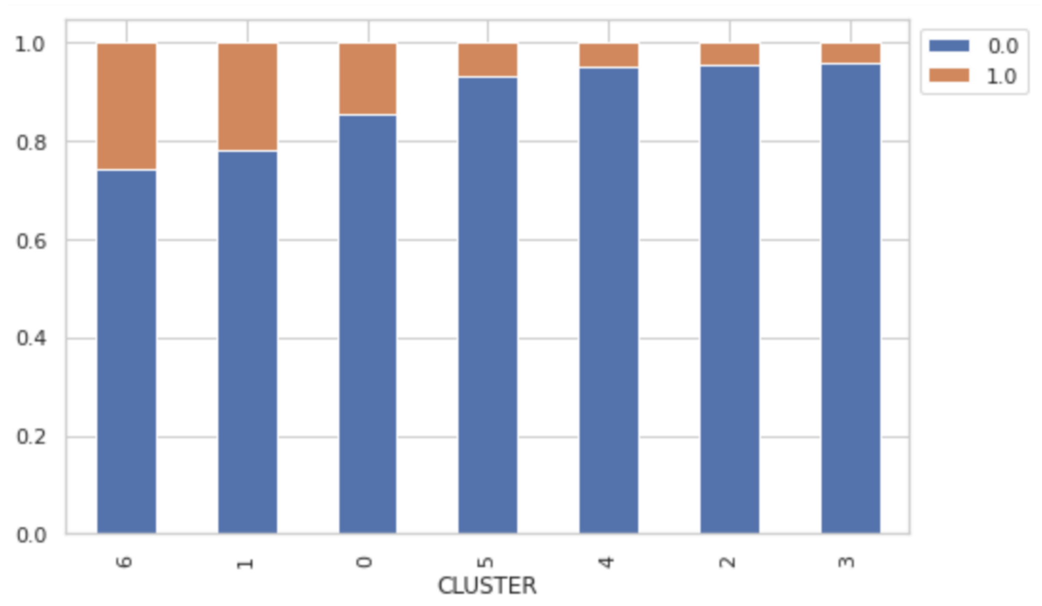
It is evident from the above plot that out of major customer clusters, Cluster 3 users have more inclination towards cosmetics and food both while cluster 0 have significant food only population as well. This can be utilized by the marketing team while sending personalized campaigns.

Cluster Analysis: CUSTOMER_GROUP Distribution



Out of the major 3 customer like clusters, Cluster 0 represents more proportion of single buyer than cluster 3 and 5. This can be utilized by the marketing team while sending personalized campaigns.

Cluster Analysis: ONLINE_PURCHASE Distribution



Cluster 0 has more proportion of online purchase preference. This insight can also be utilized to market accordingly.

This ends the Customer Segmentation Report. We will talk about the Supervised Learning Model in the next part.

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, we will build our 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, we'll predict on the "TEST" partition, where the "RESPONSE" column has been withheld.

Modeling Data set Preparation

In this step, we divided the dataset into 3 parts i.e training, validation and test. Training and validation was used to fit the model and test dataset was used to evaluate the model.

A RESPONSE variable is used to create the target variable and the remaining features were calculated using the same preprocessing steps from the previous section. We performed the feature encoding, cleaning, standardization and imputation on the data. While doing this, we made sure that there is no data leakage in the training data.

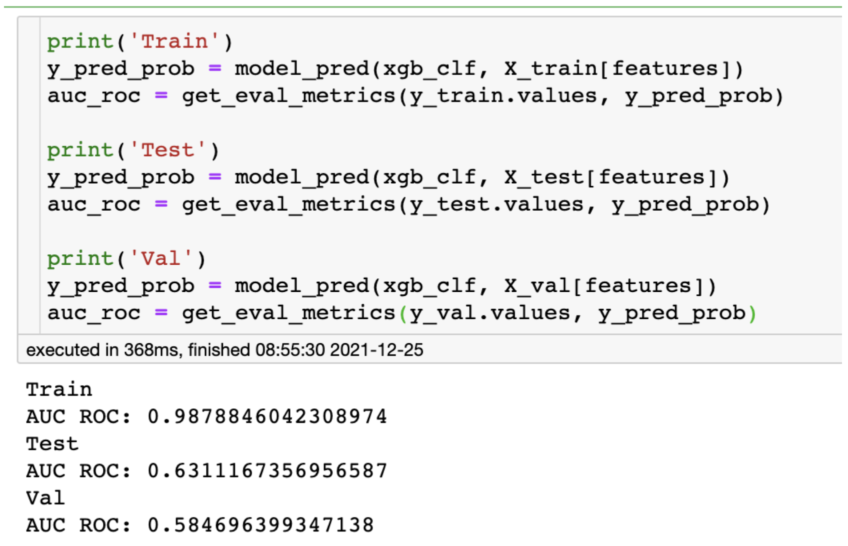
Evaluation Metric

Since it is a classification problem and data is not really very balanced, accuracy may not be the perfect metric to evaluate the performance of the model. We will use AUC(Area Under Curve) ROC CURVE to evaluate the model. It was done with cross validation as well to make sure that there is no high variance in the models.

Baseline Model and Performance

We have used a boosting algorithm specifically xgboost to build our baseline model. All the default parameters were used. Since we know there is a huge difference in the volume of positive-negative classes, we need to find a way to handle this situation. One way could be to use SMOTE or similar techniques to do over-sampling or undersampling. Undersampling would lead to loss of a lot of information and oversampling will lead to a lot of synthetic data.

For this exercise we choose to use the inbuilt feature of xgboost which handles imbalance issues with sample_weight where all the instances will be provided a weight in such a way that the minority class will get enough representation in the training process.



```
print('Train')
y_pred_prob = model_pred(xgb_clf, X_train[features])
auc_roc = get_eval_metrics(y_train.values, y_pred_prob)

print('Test')
y_pred_prob = model_pred(xgb_clf, X_test[features])
auc_roc = get_eval_metrics(y_test.values, y_pred_prob)

print('Val')
y_pred_prob = model_pred(xgb_clf, X_val[features])
auc_roc = get_eval_metrics(y_val.values, y_pred_prob)
```

executed in 368ms, finished 08:55:30 2021-12-25

```
Train
AUC ROC: 0.9878846042308974
Test
AUC ROC: 0.6311167356956587
Val
AUC ROC: 0.584696399347138
```

Once we build the model and evaluate it, it is evident from the data it is highly overfit and performance on validation is not really good. So, we are going to perform hyper-parameter tuning on the model.

Hyper-Parameter Tuning

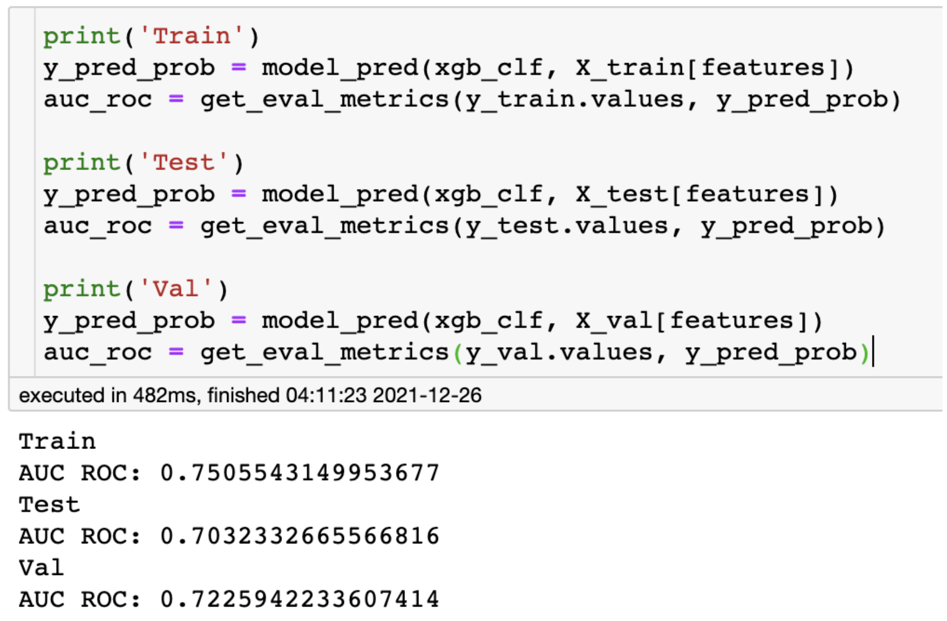
To perform Hyper-parameter tuning, GRID Search or RANDOMIZED search are widely used options but there are a lot of limitations of these methods. They could be compute heavy as every combination of the features needs to be tested which could be very large if we put large space. These methods are also dependent on the range you provide to test.

To avoid this challenge, we will use Bayesian Optimisation technique to do the same. To do the same, we will use the framework of OPTUNA which is an open source hyperparameter optimization framework to automate hyperparameter search. It automates the trial-and-error process of optimizing the hyperparameters. It automatically finds optimal hyperparameter values based on an optimization target.

We ran the optuna optimization for an initial random trial of 25 and main trials of 100. Once we have the final parameters, we will build the new model on these parameters.

Final Model Evaluation

Once the Hyper-parameter tuning is done, we trained our xgboost model with best parameters achieved from bayesian optimization of optuna. All the parameters details can be found in the attached jupyter notebook. Here is the result we achieved after training the data.



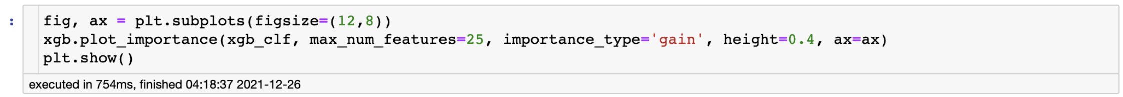
```
print('Train')
y_pred_prob = model_pred(xgb_clf, X_train[features])
auc_roc = get_eval_metrics(y_train.values, y_pred_prob)

print('Test')
y_pred_prob = model_pred(xgb_clf, X_test[features])
auc_roc = get_eval_metrics(y_test.values, y_pred_prob)

print('Val')
y_pred_prob = model_pred(xgb_clf, X_val[features])
auc_roc = get_eval_metrics(y_val.values, y_pred_prob)
```

executed in 482ms, finished 04:11:23 2021-12-26

```
Train
AUC ROC: 0.7505543149953677
Test
AUC ROC: 0.7032332665566816
Val
AUC ROC: 0.7225942233607414
```

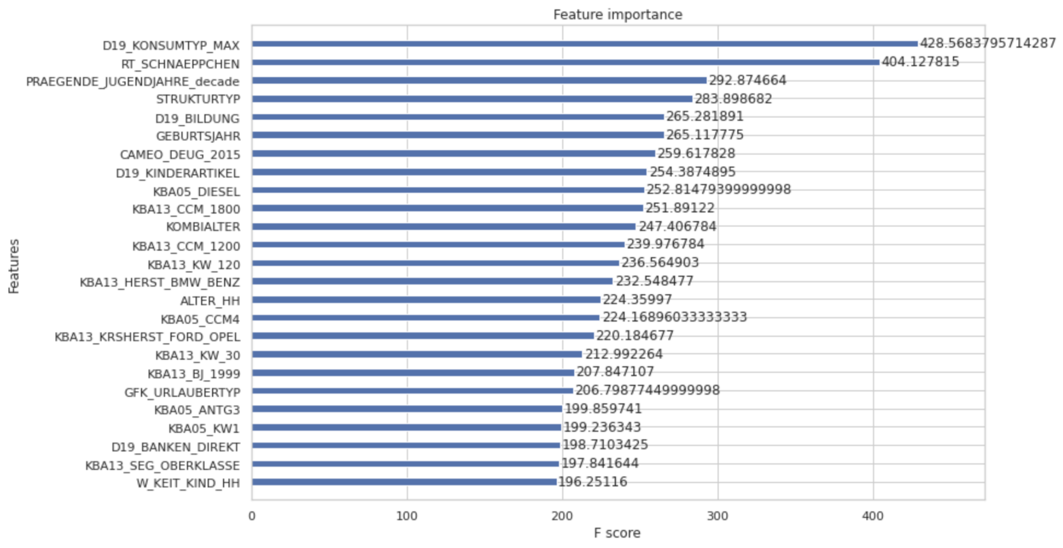
From the above result, we can observe that performance has been improved a lot. There is no overfitting in the data and AUC on the validation and test set is also improved than the baseline model.

We have also done feature importance analysis using the 'gain' method. Here is top 25 features which we used to build the model:



```
: fig, ax = plt.subplots(figsize=(12,8))
  xgb.plot_importance(xgb_clf, max_num_features=25, importance_type='gain', height=0.4, ax=ax)
  plt.show()
```

executed in 754ms, finished 04:18:37 2021-12-26



We will use this model to perform the prediction on the test dataset provided. This file will be used in the next section while participating in the kaggle competition.

Kaggle Competition

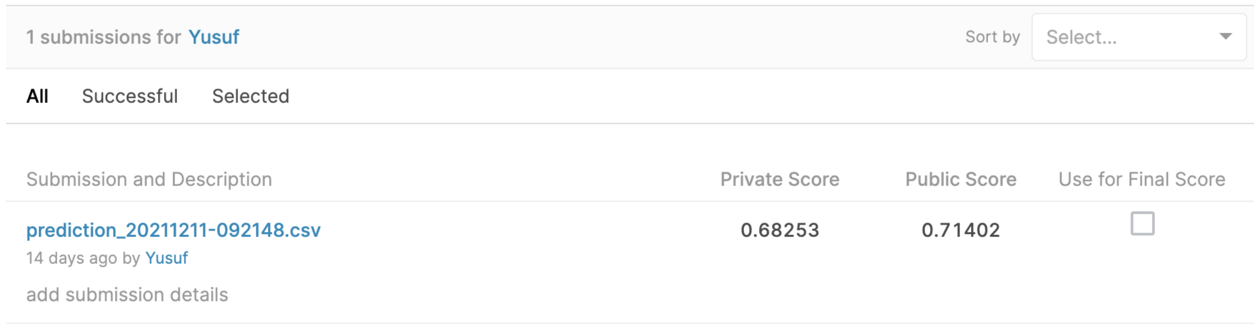
Here is the link to find the competition on Kaggle:

https://www.kaggle.com/c/udacity-arvato-identify-customers/leaderboard#score

We have a created the submission file in csv format with the following 2 columns:

- **LNR**: ID of a user
- **RESPONSE**: Probability of a user responding to the campaign

Here is the performance on Kaggle:



1 submissions for Yusuf

Submission and Description	Private Score	Public Score	Use for Final Score
prediction_20211211-092148.csv 14 days ago by Yusuf add submission details	0.68253	0.71402	☐