arvato
BERTELSMANN

Finding New Customers using Machine Learning

UDACity | NanoDegree Data Science | Capstone Project

July 2020

Table of contents

INTRODUCTION

This analysis of the Bertelsmann-Arvato data set will be a customer segmentation where a data set compiled with customers from a mail-order company is compared with a data set sampled from the general population of Germany.

Data in this project provided as Bertelsmann Arvato Analytics:

Demographics data of customers of a mail-order sales company in Germany,

Was analyzed and compared against demographics information for the general population.

Problem Statement

The main question of interest is to identify segments of the population that are most likely to be customers of their products for a mail-out campaign.

PROJECT STRUCTURE

This project was planned in **4 main** part process:

1. **Data cleaning** and **Data processing**
2. Implementing of **Unsupervised Learning after** investigation of **important features** for customer segmentation
3. Implementing of **Supervised Learning**
4. Kaggle Competition

PART 0 : Data cleaning and Data processing

There are several things which have to be done to a raw data set before you can use it for an Analysis:

Exploratory data analysis to according to statistics of features

Processing of missing values

Transform features and refactoring code

PART 1: Customer Segmentation Report

The purpose in this part is to find features describing the typical customer for the mail order company and to find the part of the population which is not in the focus of the company yet.

Because of that, have done the project with analyze features of established customers and the general population in order to create customer segments by using unsupervised learning methods.

PART 2: Supervised Learning

The purpose of this part is to train an algorithm to be able to find customers who will respond positive to a mailout campaign.

Did to apply Supervised Learning to investigate **MAILOUT-TRAIN** and **MAILOUT-TEST** datasets to predict a person became a customer or not of the mail-order company following the campaign.

The model of the algorithm is chosen with the help of learning curves and the best parameters.

PART 3: Kaggle Competition

The **model** obtained from trained and parameterized **Part-3** is applied to implementation, **a test dataset in Kaggle**, and loaded and rated.

Summarize of Data

Udacity_AZDIAS_052018

Demographics data for the **general population of Germany:**

891 211 persons (rows) x 366 features (columns)

Udacity_AZDIAS_052018

Demographics data for **customers of a mail-order company:**

191 652 persons (rows) x 369 features (columns)

Udacity_MAILOUT_052018_TRAIN

Demographics data for individuals who were targets of a marketing campaign:

42 982 persons (rows) x 367 (columns)

Udacity_MAILOUT_052018_TEST

Demographics data for individuals who were targets of a marketing campaign:

42 833 persons (rows) x 366 (columns)

Summarıze of Features

There is information about what 314 columns are in the data attribute files. Among this information, had information of only 233 columns from 368 columns in datasets. From Germany and interested in the project Someone wrote addıtıonal information for 24 of them in the forum.

Observed that there are 3 columns of differences between **Azdias** and Customers data sets.

PROJECT COMPONENTS

Data cleaning and Data processing

The files were imported to do some exploratory data analysis. In the beginning, descriptive statistics and calculated distribution of missing values in the dataset were analyzing.

The features of both data sets were synchronized.

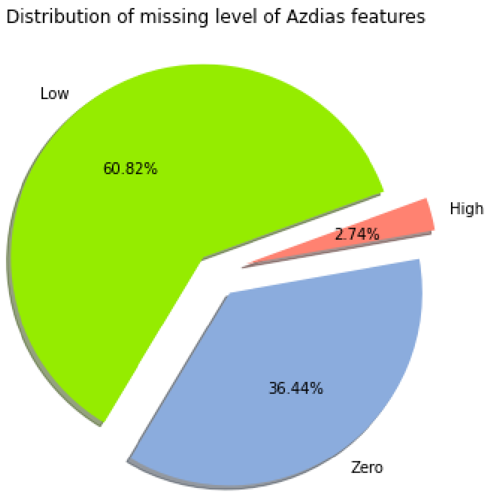
```python
# Check the excess columns between customers and azdias dataframe
excess_col_customer = list(set(customers.columns.tolist()) - set(azdias.columns.tolist()))
display(excess_col_customer)
display(customers[excess_col_customer].head(3))
```

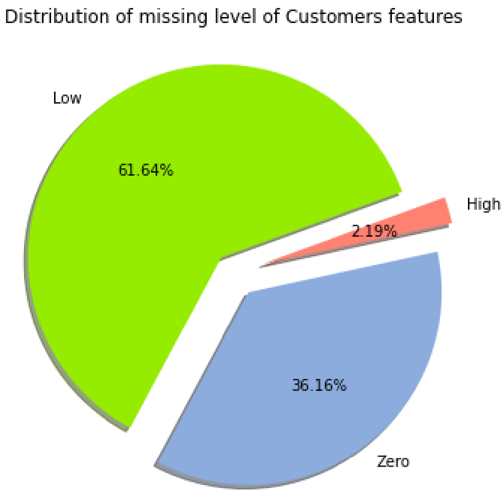
```
['PRODUCT_GROUP', 'CUSTOMER_GROUP', 'ONLINE_PURCHASE']
```

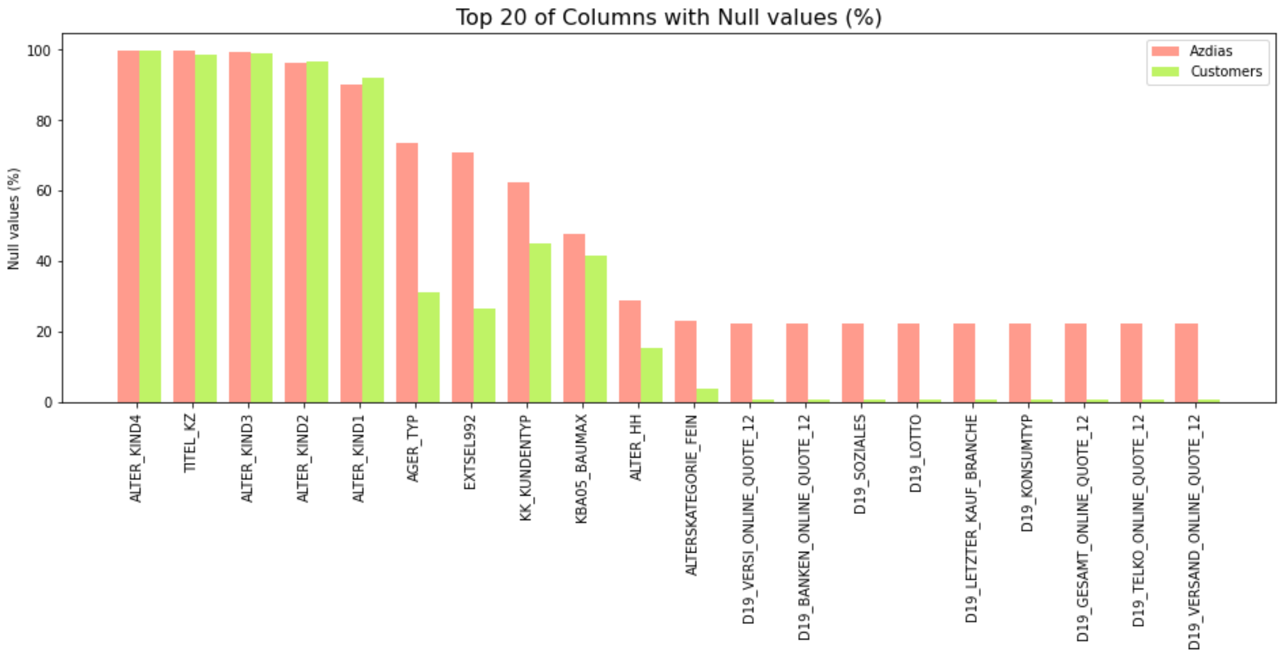
LNR	PRODUCT_GROUP	CUSTOMER_GROUP	ONLINE_PURCHASE
9626	COSMETIC_AND_FOOD	MULTI_BUYER	0
9628	FOOD	SINGLE_BUYER	0
143872	COSMETIC_AND_FOOD	MULTI_BUYER	0

Rows that are 80% null have been dropped.

The missing values of the **columns** were examined. Columns with missing values at the ‘**high**’ level were observed to have the same columns in datasets of Azdias and Customers. The number of the same columns are 8.







Encoding some Columns: CUSTOMERS dataset contains is target dataset that as our customer's source data. Therefore, we will act on the column investigation on this dataset. As we begin this work, which is a preliminary preparation for Encoding, which we will use before implementing the model, we will start with the columns with high '**nunique**' value.

value.

Here is the purpose is to identify important columns and significantly reduce these feature's unique values.

```
customers.nunique().sort_values(ascending=False)[:15]
```

```
EINGEFUEGT_AM                 2760
KBA13_ANZAHL_PKW              1250
ANZ_STATISTISCHE_HAUSHALTE     208
ANZ_HAUSHALTE_AKTIV            208
GEBURTSJAHR                    113
EXTSEL992                       56
VERDICHTUNGSRAUM                46
CAMEO_DEU_2015                  44
LP_LEBENSPHASE_FEIN             41
D19_LETZTER_KAUF_BRANCHE        35
EINGEZOGENAM_HH_JAHR            33
MIN_GEBAEUDEJAHR                29
ALTERSKATEGORIE_FEIN            25
CAMEO_INTL_2015                 21
ALTER_HH                        20
dtype: int64
```

Each of the above features was examined separately. Unnecessary features deleted. Label encoding was performed for each of the others.

If *Missing values are filling with mean will introduce decimals into dataset and this case can effect efficiency*. Because of this, missing values replaced with MODE of each column. Because of this, **missing values replaced with each column's MODE** after removing **duplicates rows**.

```
# Checking count of Null value
print('Number of NaN before fillna : {}'.format(df_total.isnull().sum().sum()))
df_total = df_total.fillna(df_total.mode().iloc[0])
print('Number of NaN after fillna  : {}'.format(df_total.isnull().sum().sum()))
```

```
Number of NaN before fillna : 6310113
Number of NaN after fillna  : 0
```

Feature Engineering

Using 4 different models, we tried to identify the important features in 366 features. Used **GBM, RF, XGBoost**, and **StackedEnsemble** from **H2o library**. After examining the scores of the columns I obtained from each model, I found 57 important features. In the next stages, these 57 features will be used.

```python
# Train a stacked ensemble using the GBM, RF and XGB above
ensemble = H2OStackedEnsembleEstimator(model_id="ensemble",
                                       seed=seed,
                                       keep_levelone_frame=True,
                                       training_frame=train,
                                       base_models=[my_gbm.model_id,
                                                    my_rf.model_id,
                                                    my_xgb.model_id,
                                                    my_xgb_BP.model_id])
ensemble.train(x = features_train, y = label_train,training_frame = train)
```

	gbm	drf	xgboost	stackedensemble
Accuracy	0.926888	0.920274	0.929043	0.986150
AUC Score	0.929944	0.932689	0.945892	0.998671
AUC-pr Score	0.757669	0.775269	0.801327	0.992693
Recall	0.027004	0.000006	0.151395	0.184207

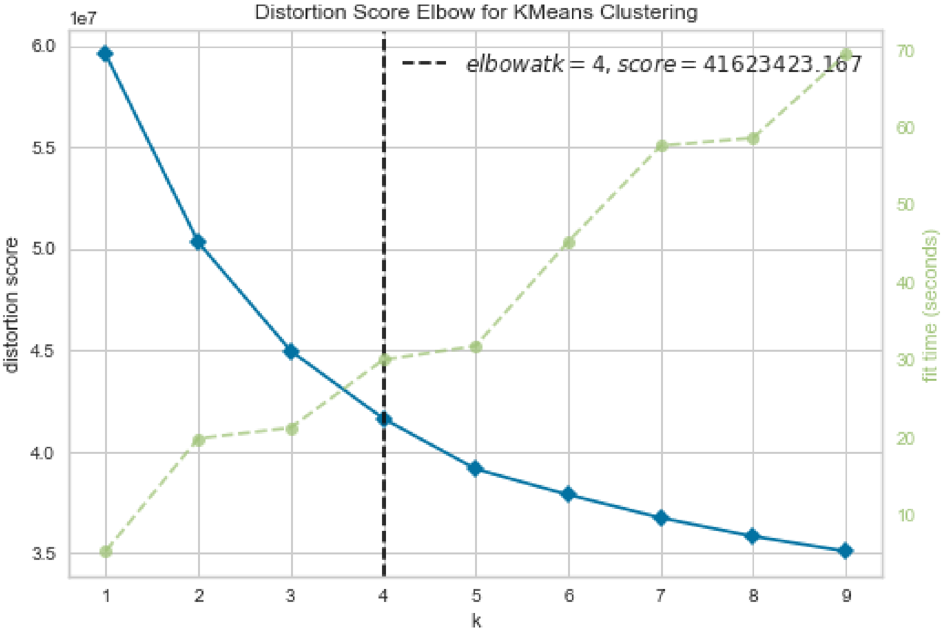

Rank of Features Importance (%)
GBM_model
Important_Rank (%)
D19_SOZIALES
ANREDE_KZ
ALTERSKATEGORIE_FEIN
KOMBIALTER
FINANZ_MINIMALIST
CJT_TYP_1
FINANZ_ANLEGER
LP_FAMILIE_GROB
AKT_DAT_KL
LP_FAMILIE_FEIN
FINANZ_VORSORGER
LP_LEBENSPHASE_FEIN
SEMIO_TRADV
CJT_TYP_2
SEMIO_VERT
D19_GESAMT_DATUM
FINANZ_SPARER
VERSAND_OFFLINE_DATUM
D19_VERSAND_DATUM
CJT_TYP_6
CJT_TYP_5
VK_DISTANZ
GREEN_AVANTGARDE
SEMIO_PFLICHT
ALTER_HH
DRF_model
XGBoost_model
XGBoost_model

Customer Segmentation using Unsupervised Learning

In this section, I will talk about how I deal with unsupervised learning to analyze the characteristics of customer segments.

First of all, let's not forget that there is an IMBALANCE problem in our data set. (Azdias-Customers ~ 17%, Mailout Train- 1.1%) I observed this in exploratory data analysis, and when I applied **PCA**, I found that t**he number of components** that would represent 90–95% of data is **not low**. Therefore, I continued on my way using the **important features** I obtained with feature engineering in the previous section.

After decide that, the next step is to determine the optimal number of clusters. The elbow method was used for this. The average within-cluster distance across variance K clusters were calculated and plotted. It was decided that **4 clusters** would be a good result.



Now it's time for implementing and clustering. Here, was used H2OKMeansEstimator from H2o. Let me tell you without forgetting, I applied clustering the Mailout-Train and Mailout-Test data sets immediately after applying the Azdias-Customers data set.

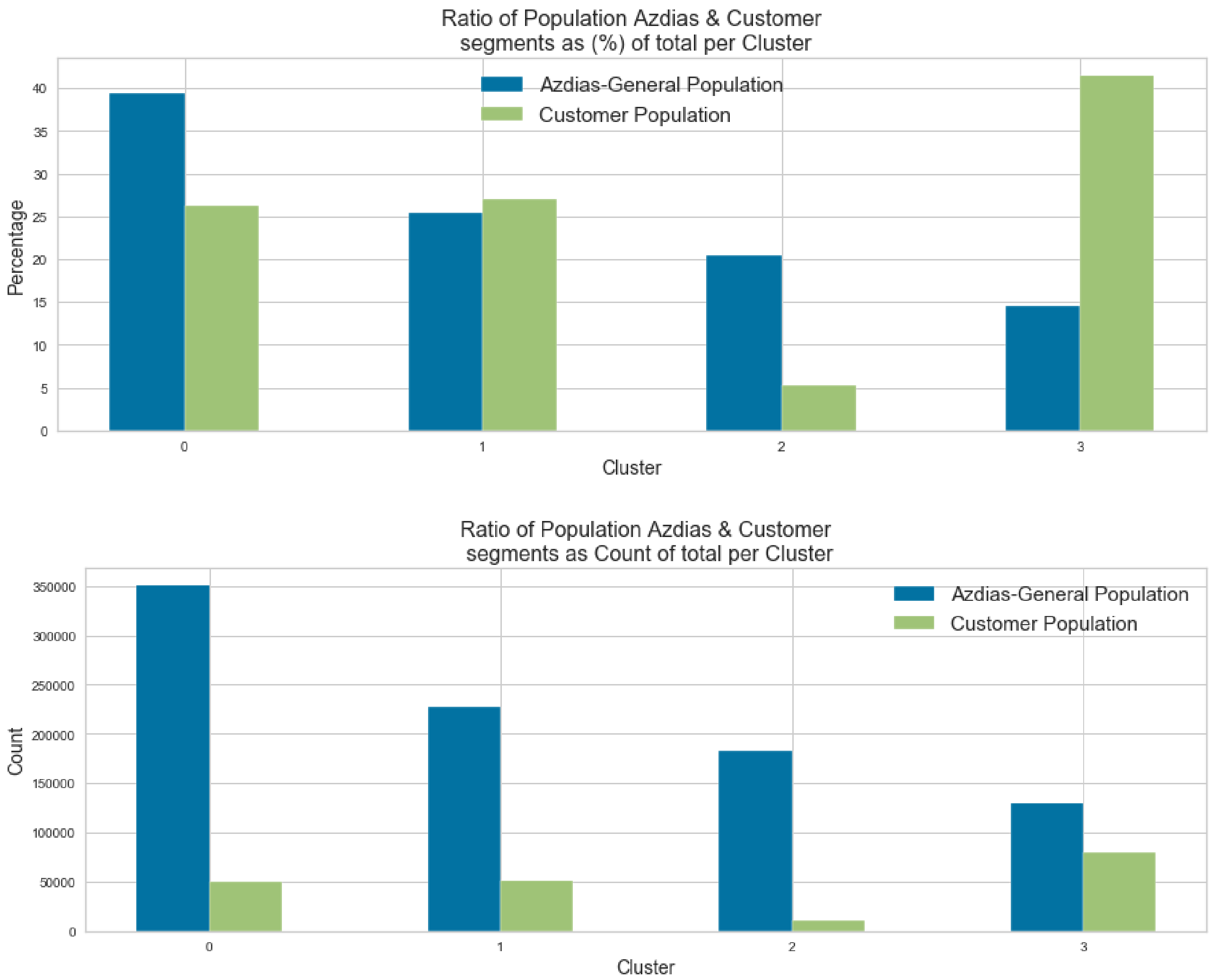
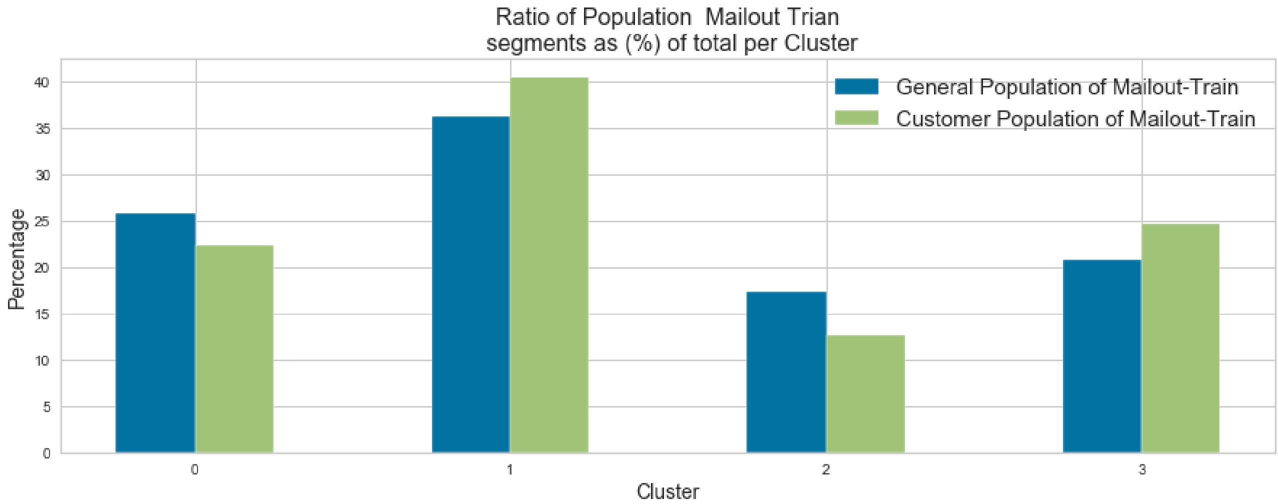


Figure 1Clustering for Azdias-Customers Data Sets



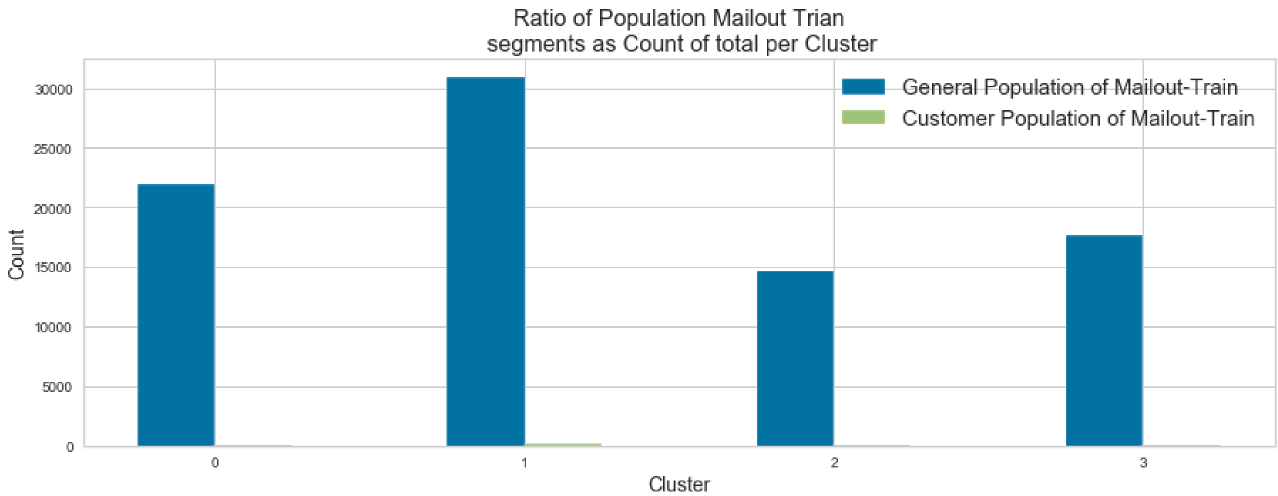


Figure 2Clusterin for Mailout Data Sets

In order to get customers, if the following order is **followed in the cluster selection**, it can **get positive results quickly**.

- Cluster 3 and Cluster 1 are the very best segment for customers
- Cluster 0 may good also
- Cluster 2 is very bad.

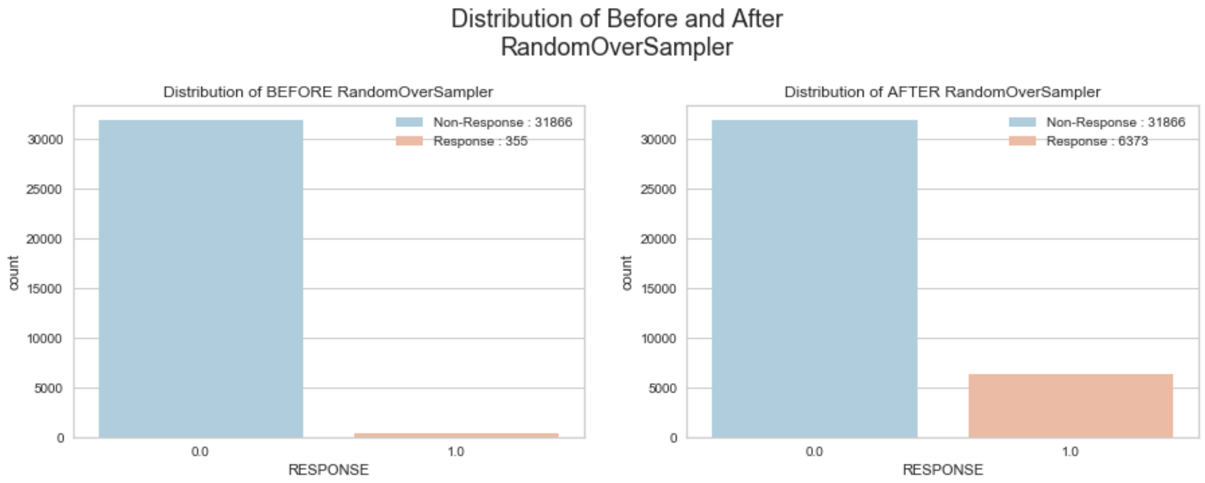
Model Evaluation and Validation

- In terms of evaluation metric to use,

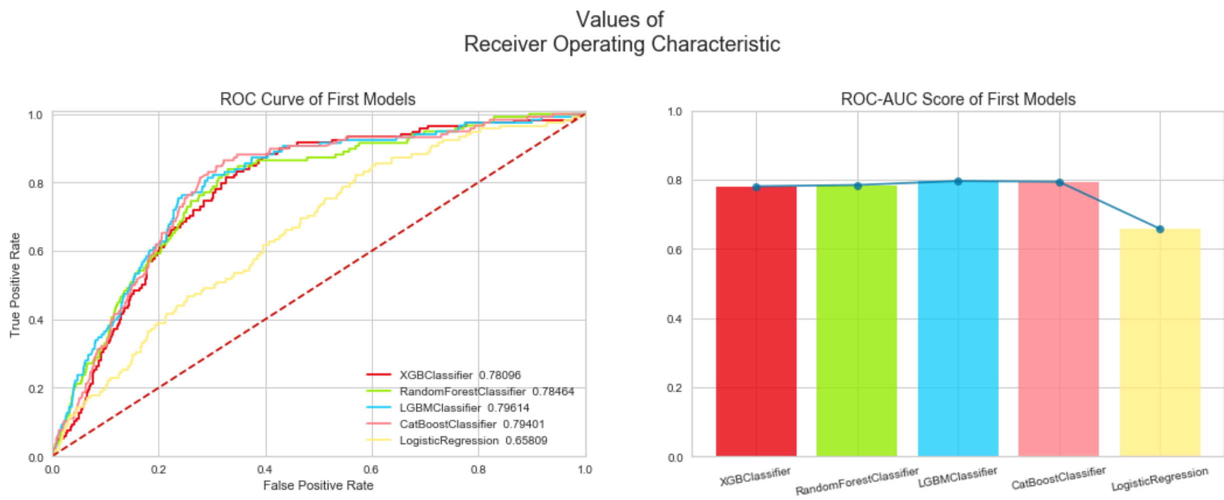
I have tried accuracy, precision, recall and f-score but due to very high imbalance (i.e. In MAILOUT_TRAIN dataset, we can find among 43k individuals, only 532 people responded to the mail-out campaign which means the training data is highly imbalanced.), none of these were a good way to measure and then finalized on AUC and ROC as the evaluation metric to proceed.

Supervised Learning to probably predict customers

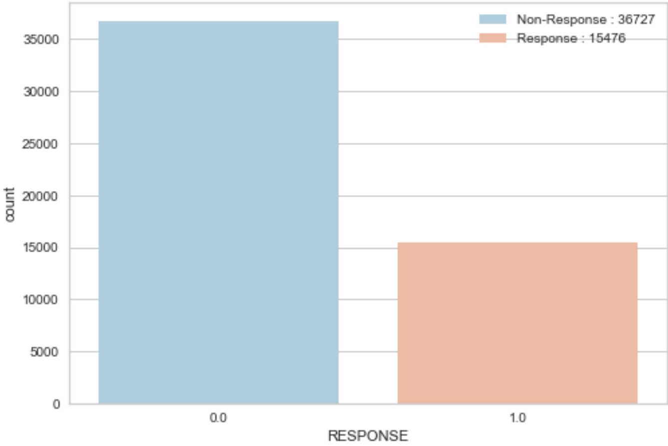
Data Cleaning phase was turned into **Pipeline** and applied for Mailout-Train and Mailout-Test. Data were then arranged according to the **important features**. Before modeling **RandomOverSampling** (sampling_strategy = **0.15**) has been done to solve the imbalance problem.

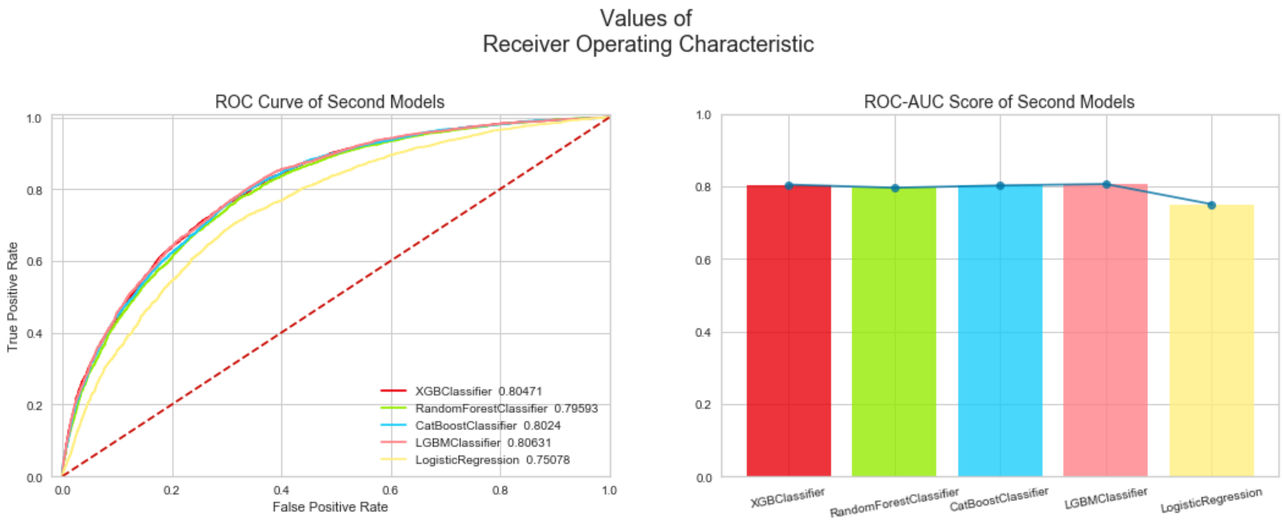


This process is classifier Because of this, I choose and tried Lightgbm, XGBoost, CatBoost, Random Forest, Logistic Regression **classifier models**.



Before applying the VotingClassifier algorithm in Ensemble modeling, I made the adjustment for the imbalance problem. I added some records from the Customers dataset to the Mailout-Train dataset instead of OverSampling artificially. This second model yielded better results, albeit less than the first model.





Name of ROC-AUC Score	First Model	Second Model
VotingClassifier	NaN	0.858809
LGBMClassifier	0.796136	0.806310
XGBClassifier	0.780960	0.804712
CatBoostClassifier	0.794013	0.802397
RandomForestClassifier	0.784644	0.795933
LogisticRegression	0.658086	0.750782

For the **Second data** set **best model score**d around **0.8063 from LGBM** and **0,8588 from VotingClassifier**. Used predicting of the test label using this **VotingClassifier** model**.**

Kaggle Submission

Now, will done of classification of Mailout-Test data with best model trained on train data. In here, all models will be estimated. But will submitted just prediction of the fine-tuned VotingClassifier model.

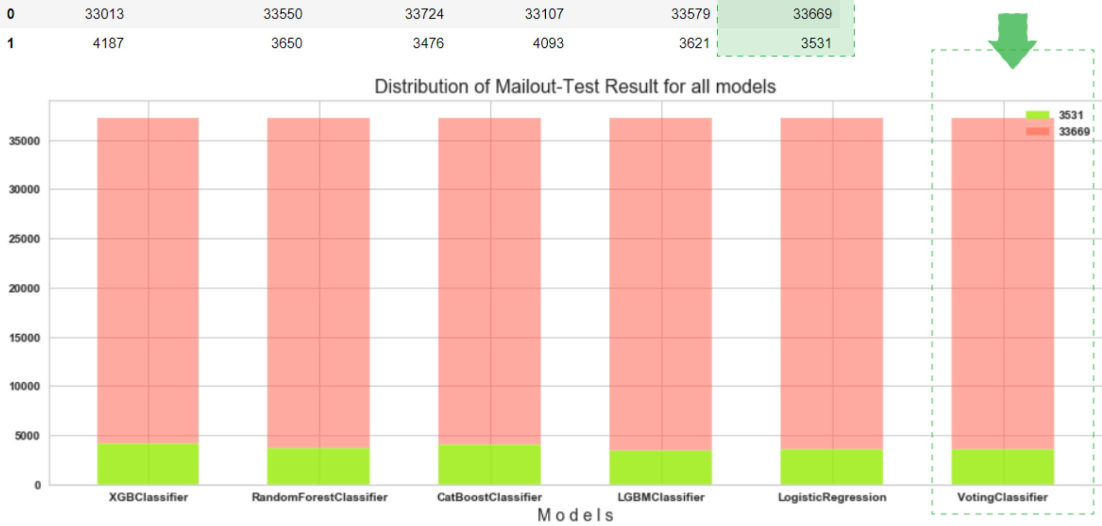
```
# Generate CSV file for the Kaggle competition
df_kaggle.to_csv('../Last/data/kaggle_submission_file.csv', index=False)
```

```
# Label  Encoding for RESPONSE column
df_kaggle['RESPONSE_2'] = df_kaggle['RESPONSE'].apply(lambda x : 1 if x>=0.5 else 0)

df_kaggle.sample(10)
```

	LNR	RESPONSE	RESPONSE_2
4017	36464	0.103992	0
12478	51668	0.093551	0
25	11434	0.125693	0
3059	83792	0.038795	0
665	35062	0.246663	0
510	10566	0.095973	0

	XGBClassifier	RandomForestClassifier	CatBoostClassifier	LGBMClassifier	LogisticRegression	VotingClassifier
0	33013	33550	33724	33107	33579	33669
1	4187	3650	3476	4093	3621	3531



IMPROVEMENT

There are a lot of different approaches and solutions that can be applied from data cleaning to model training. As it is a real-life problem, If the features knowledge will known, the result will be better since the column selections will be different. To sum up, increase the performance of the supervised learning model, the following parts can be performed:

- Learn more about the feature to improve feature engineering,
- Handle unbalanced data: under-sampling or oversampling,
- Increase features or PCA components.
- The LightGBM model could be improved by running more extensive hyper-parameter tuning.

CONCLUSION

Trained a **K-means model** on the general-customers population data sets. Used the model to cluster the customer data for the customer segmentation and then was **compared distributions of clusters**.

- **Stacking** and **Voting** were **useful than a single model result**.
- It would be nice if we present our findings to **the customer and receive feedback.**
- Implementing GridSearchCV with **10–128 variations** for each model was **a mistake for me**.
- Public code for this analysis may be found in its Github repository.

Some resources used for this project:

- - Visualization
- - H2O.ai Documentation
- - For GridSearch
- - Some Articles: 1, 2, 3,4