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.

Be apply supervised and unsupervised learning to reach this aim.

Unsupervised learning helps us find the similarities within the clusters. I will apply **H2O-Kmeans** to find clusters. After predicting & analyzing the **clusters** I will be able to predict Potential Customers from **Azdias** dataset.

While unsupervised learning helps us to find similar records, **supervised learning** will let us predict directly each record class. Our problem is a classification problem as target values are [0,1] represents not potential and potential customers respectively. I will use **Lightgbm, XGBoost, CatBoost, Random Forest, Logistic Regression Classifier** algorithms with **sklearn** library to predict whether each person is a potential customer or not.

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)

Summarize 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 additional 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

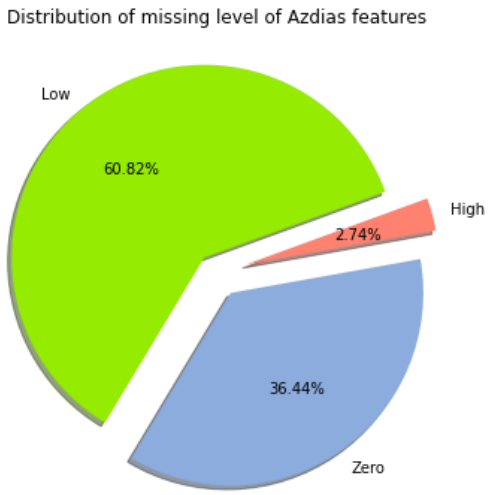
- The files were imported to do some exploratory data analysis. In the beginning, descriptive statistics and calculated distribution of missing values in the dataset were analyzing.
- The features of both data sets were synchronized.

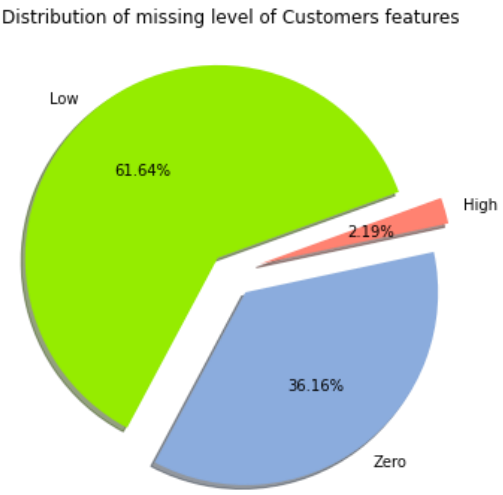
```
# Check the excess columns between customers and azdias dataframe
excess_col_customer = list(set(customers.columns.tolist()) - set(azdias.columns.tolist()))
display(excess_col_customer)
display(customers[excess_col_customer].head(3))
```

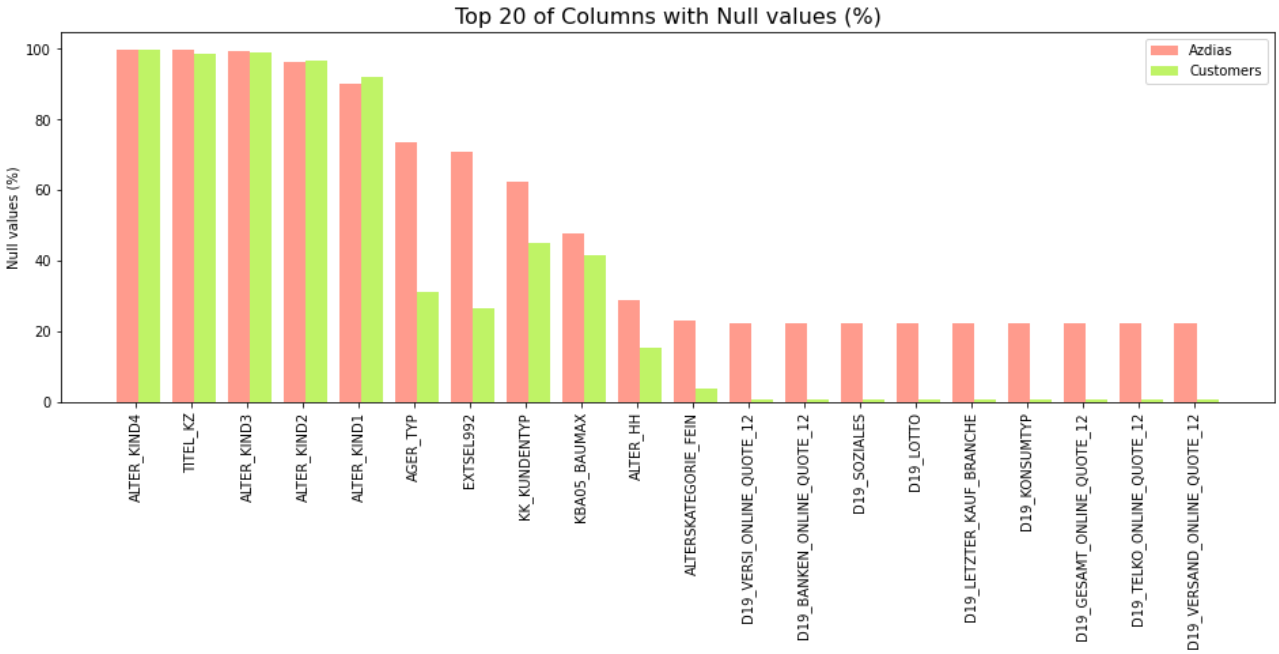
```
['PRODUCT_GROUP', 'CUSTOMER_GROUP', 'ONLINE_PURCHASE']
```

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:

Unsupervised learning techniques use only work on numerically features. Because of this make a few encoding changes or additional assumptions to be able to make continue.

Also, 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.

Encoding of 'RAEGENDE_JUGENDJAHRE' feuture

RAEGENDE_JUGENDJAHRE, In this column give us two different information. One is time serisi, other is two type (Mainstream - Avant Garde)
Must convert mixed information code into two different columns.

```
1 : 40ies - war years (Mainstream,O+W)
2 : 40ies - reconstruction years (Avantgarde,O+W)
3 : 50ies - economic miracle (Mainstream,O+W)
4 : 50ies - milk bar / Individualisation (Avantgarde,O+W)
5 : 60ies - economic miracle (Mainstream,O+W)
6 : 60ies - generation 68 / student protestors (Avantgarde,W)
7 : 60ies - opponents to the building of the Wall (Avantgarde,O)
8 : 70ies - family orientation (Mainstream,O+W)
9 : 70ies - peace movement (Avantgarde,O+W)
10 : 80ies - Generation Golf (Mainstream,W)
11 : 80ies - ecological awareness (Avantgarde,W)
12 : 80ies - FDJ/communist party youth organisation (Mainstream,O)
13 : 80ies - Swords into ploughshares (Avantgarde,O)
14 : 90ies - digital media kids (Mainstream,O+W)
15 : 90ies - ecological awareness (Avantgarde,O+W)
```

```
def encode_df_PJ(df):

    '''
    First Column for convert of time series
        {1:0,2:0,3:1,4:1,5:2,6:2,7:2,8:3,9:3,10:4,11:4,12:4,13:4,14:5,15:5}

    Other Column FOR two type (Mainstream - Avant Garde)
        {1:0, 2:1, 3:0,  5:0,8:0, 10:0, 12:0, 14:0,
        4:1, 6:1, 7:1, 9:1, 11:1, 13:1, 15:1}
    '''

    # PRAEGENDE_JUGENDJAHRE, convert mixed information code into seperate code
    PRAEGENDE_time = {1:0,2:0,3:1,4:1,5:2,6:2,7:2,8:3,9:3,10:4,11:4,12:4,13:4,14:5,15:5}
    PRAEGENDE_avant = {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,

    df['PRAEGENDE_time'] = df['PRAEGENDE_JUGENDJAHRE'].map(PRAEGENDE_time)
    df['PRAEGENDE_avant'] = df['PRAEGENDE_JUGENDJAHRE'].map(PRAEGENDE_avant)
    del df['PRAEGENDE_JUGENDJAHRE']
```

encode_df_PJ hosted with ❤ by GitHub view raw

```
# 06 : Encoding of 'PRAEGENDE_JUGENDJAHRE' feature
#      into 'PRAEGENDE_time' and 'PRAEGENDE_avant'
encode_df_PJ(azdias)
encode_df_PJ(customers)
```

Encode and Investigation of 'GEBURTSJAHR' feuture

GEBURTSJAHR, In this column give us year of birth. This column type is numeric. **Result:** We use this column **GEBURTSJAHR** after below process:

- In this column it has `0` (zero). That means **wrong** or **no entry**. We should replace **zero** with **NaN**

- Divided the values into categorical with {190:193, 191:193, 192:193, 200:199, 201:np.NaN}

```
def encode_map_age(df):
    '''
    Encode of GEBURTSJAHR
    In this column it has 0 (zero). That means wrong or no entry. We should replace zero with
    Divided the values into categorical with {190:193, 191:193, 192:193, 200:199, 201:np.NaN}
    '''
    df['GEBURTSJAHR'] = df['GEBURTSJAHR'].apply(lambda x: int(x/10))
    map_age = {190:193, 191:193, 192:193, 200:199, 201:np.NaN, 0: np.NaN}
    df['GEBURTSJAHR'].replace(map_age, inplace=True)
```

encode_map_age hosted with ❤ by GitHub view raw

```
# Implamenting map_age function for GEBURTSJAHR feature
encode_map_age(customers)
encode_map_age(azdias)
```

Investigation of 'EINGEFUEGT_AM' feature

*'**EINGEFUEGT_AM**, In this column gives us year-mount-day information. We transform year to index from **EINGEFUEGT_AM** feature into **EINGEFUEGT_ind.***

```
def encode_df_EIN(df):
    '''
        Transform year to index for 'EINGEFUEGT_AM' feature
    '''
    # EINGEFUEGT_AM: transform year to index
    df['EINGEFUEGT_AM'] = pd.to_datetime(df['EINGEFUEGT_AM'])

    max_year = df['EINGEFUEGT_AM'].max().year
    df['EINGEFUEGT_ind'] = (max_year - df['EINGEFUEGT_AM'].dt.year)
```

encode_df_EIN hosted with ❤ by GitHub view raw

```
# 02: Encoding of 'EINGEFUEGT_AM' feuture into EINGEFUEGT_ind
encode_df_EIN(azdias)
encode_df_EIN(customers)
```

```
a = azdias['EINGEFUEGT_ind'].value_counts()
c = customers['EINGEFUEGT_ind'].value_counts()
a.name ='Azdias_EINGEFUEGT_ind'
c.name= 'Customers_EINGEFUEGT_ind'

pd.concat([a,c], axis=1, ).sort_index(ascending=False).head()
```

	Azdias_EINGEFUEGT_ind	Customers_EINGEFUEGT_ind
22	3	NaN
21	573887	108495.0
20	23981	3871.0
19	28647	2810.0
18	42225	5635.0

Most probably **first records are taken like 22 years ago** and that year contains **almost %60 of data. Also last 16 to 21 years contains ~%90 of data**.

Result: We drop this column due to huge skewness. Drop columns of **EINGEFUEGT_idn** that created a new, and **EINGEFUEGT_AM**.

```
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()))
```

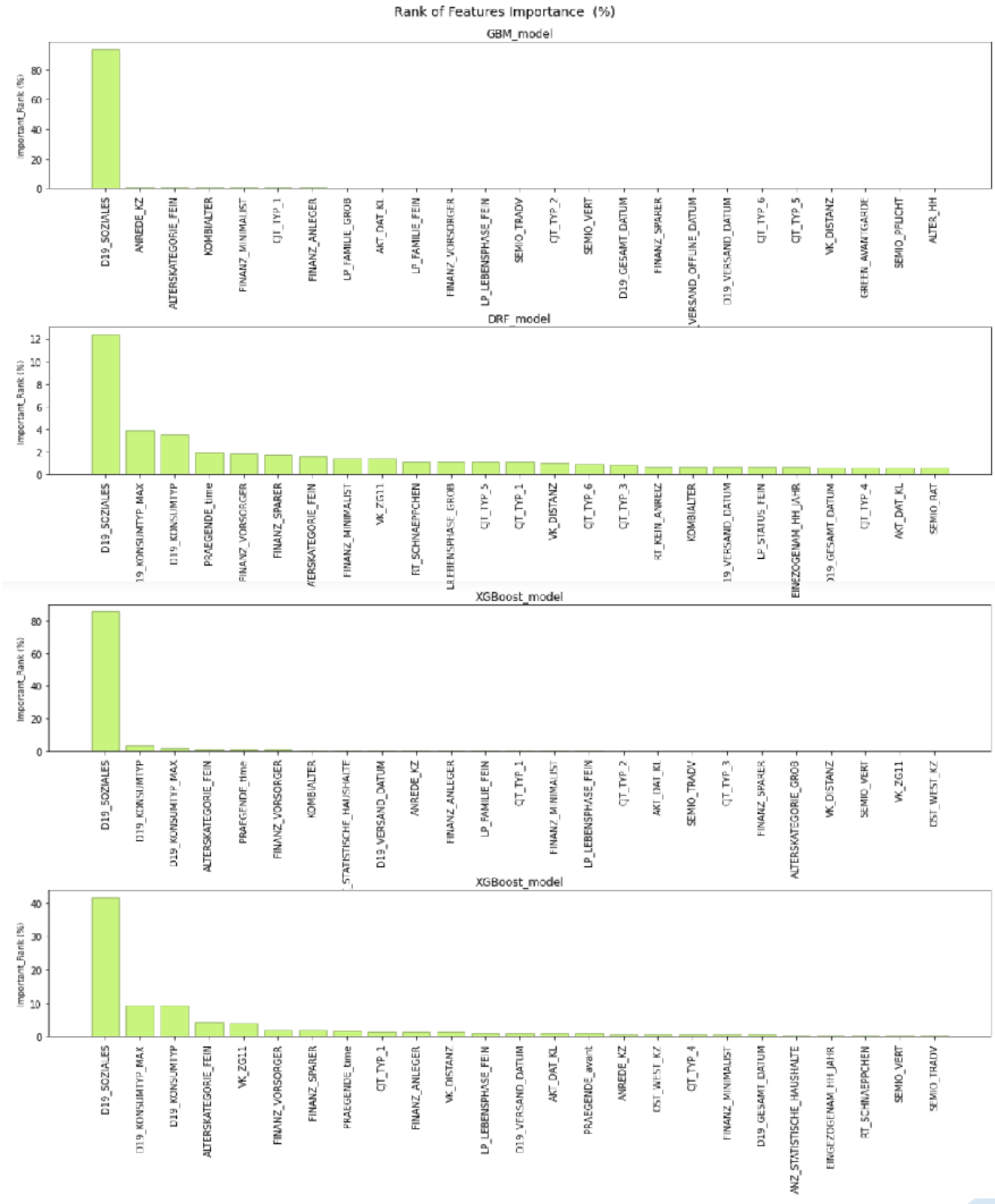
```
Number of NaN before fillna : 6310113
Number of NaN after fillna  : 0
```

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.

```
# 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
Important_Rank (%)
D19_SOZIALES
D19_KONSUMTYP_MAX
D19_KONSUMTYP
PRAEGENDE_time
FINANZ_VORSORGER
FINANZ_SPARER
ALTERSKATEGORIE_FEIN
FINANZ_MINIMALIST
VK_ZG11
RT_SCHNAEPPCHEN
LP_LEBENSPHASE_GROB
CJT_TYP_5
CJT_TYP_1
VK_DISTANZ
CJT_TYP_6
CJT_TYP_3
RT_KEIN_ANREIZ
KOMBIALTER
D19_VERSAND_DATUM
LP_STATUS_FEIN
EINGEZOGENAM_HH_JAHR
D19_GESAMT_DATUM
CJT_TYP_4
AKT_DAT_KL
SEMIO_RAT
XGBoost_model
Important_Rank (%)
D19_SOZIALES
D19_KONSUMTYP
D19_KONSUMTYP_MAX
ALTERSKATEGORIE_FEIN
PRAEGENDE_time
FINANZ_VORSORGER
KOMBIALTER
ANZ_STATISTISCHE_HAUSHALTE
D19_VERSAND_DATUM
ANREDE_KZ
FINANZ_ANLEGER
LP_FAMILIE_FEIN
CJT_TYP_1
FINANZ_MINIMALIST
LP_LEBENSPHASE_FEIN
CJT_TYP_2
AKT_DAT_KL
SEMIO_TRADV
CJT_TYP_3
FINANZ_SPARER
ALTERSKATEGORIE_GROB
VK_DISTANZ
SEMIO_VERT
VK_ZG11
OST_WEST_KZ
XGBoost_model
Important_Rank (%)
D19_SOZIALES
D19_KONSUMTYP_MAX
D19_KONSUMTYP
ALTERSKATEGORIE_FEIN
VK_ZG11
FINANZ_VORSORGER
FINANZ_SPARER
PRAEGENDE_time
CJT_TYP_1
FINANZ_ANLEGER
VK_DISTANZ
LP_LEBENSPHASE_FEIN
D19_VERSAND_DATUM
AKT_DAT_KL
PRAEGENDE_avant
ANREDE_KZ
OST_WEST_KZ
CJT_TYP_4
FINANZ_MINIMALIST
D19_GESAMT_DATUM
ANZ_STATISTISCHE_HAUSHALTE
EINGEZOGENAM_HH_JAHR
RT_SCHNAEPPCHEN
SEMIO_VERT
SEMIO_TRADV

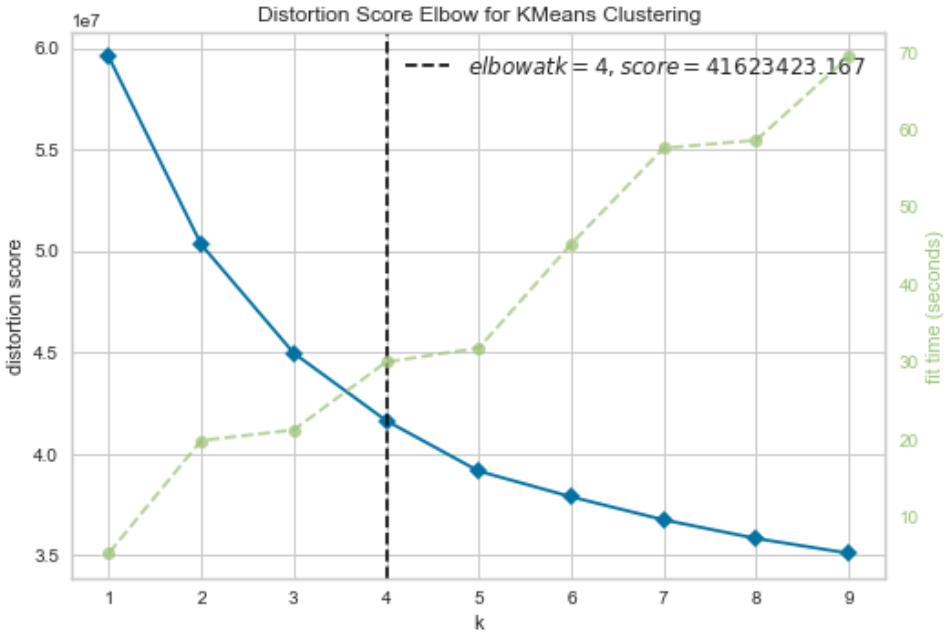
Customer Segmentation using Unsupervised Learning

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. After applying **PCA**, I found that **the number of components** that would represent 90–95% of data is **more than 200 components**. Therefore, I decided to use the **important features** I obtained with feature engineering in the previous section.

The scores I achieved scores with feature important from XGBoost were better than the scores of after applying PCA. That's why I chose to use **important features** only.

After decide that, the next step is to determine the optimal number of clusters. The elbow method was used for this with **KElbowVisualizer** by **Yellowbrick** library.



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** algorithm inside of h2o.estimators by **H2o** and plotted with **Pyplot** by **Matplotlib** library. 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

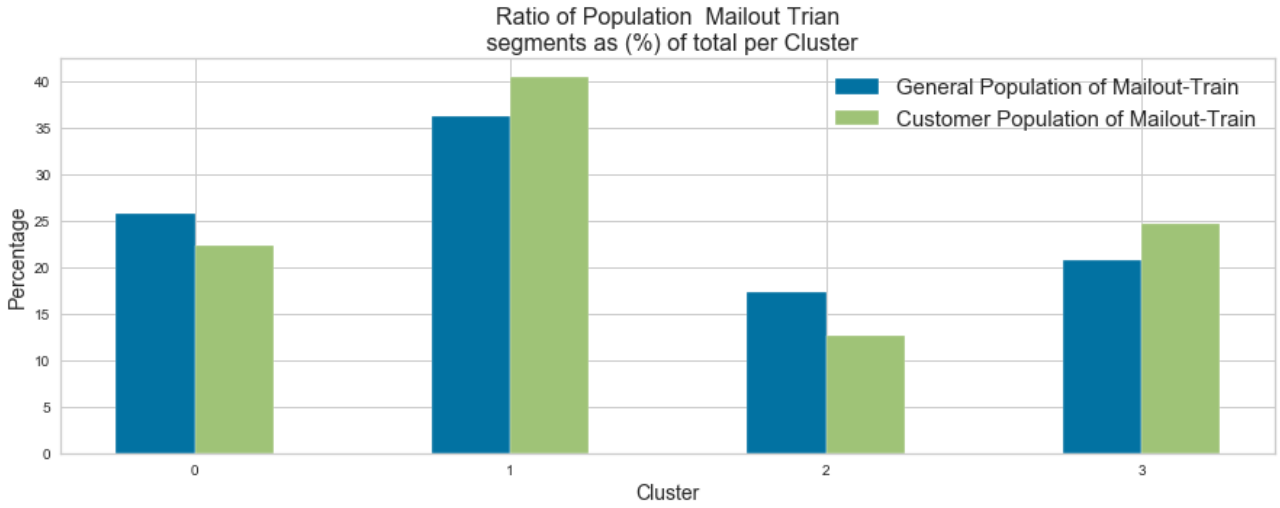


Figure 2Clusterin for Mailout Data Sets

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.

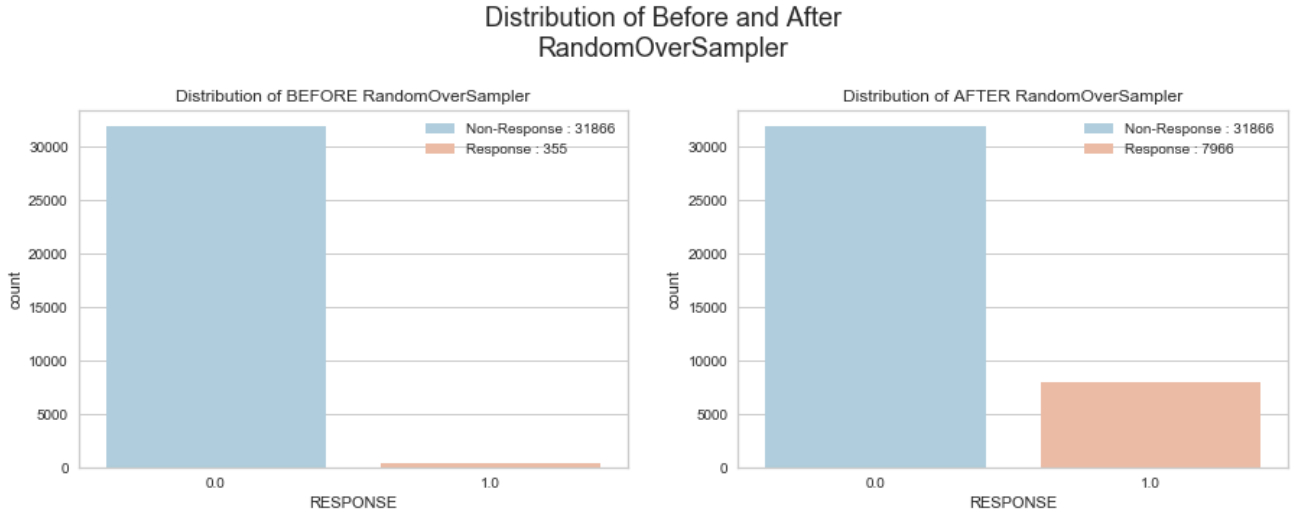
Model Evaluation and Validation

- In terms of evaluation metric to use,

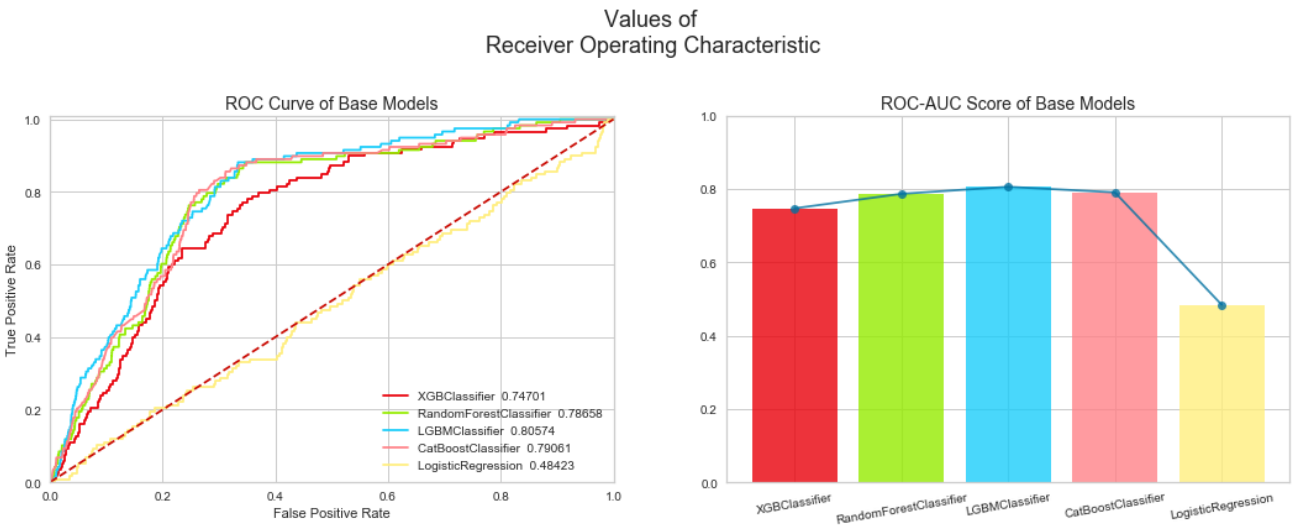
I have tried accuracy, precision, recall and f-score but due to very high imbalance (i.e. In MAILOUT_TRAIN dataset, we can find among 43k individuals, only 532 people responded to the mail-out campaign which means the training data is highly imbalanced.), none of these were a good way to measure and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

Supervised Learning to probably predict customers

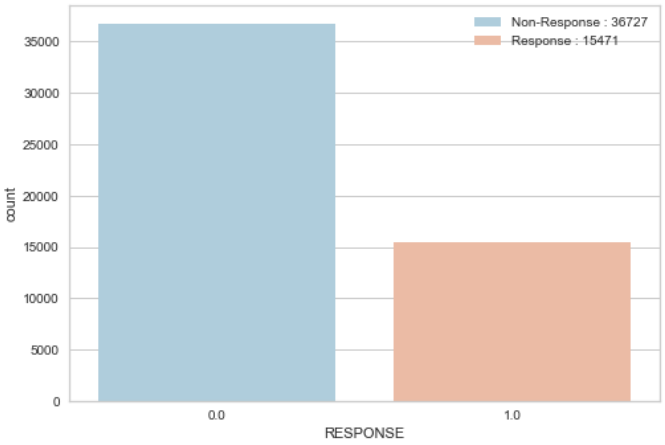
Data Cleaning phase was turned into **Pipeline** and applied for Mailout-Train and Mailout-Test. Data were then arranged according to the **important features**. Before modeling **RandomOverSampling** (sampling_strategy = **0.25**) has been done to solve the imbalance problem.



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

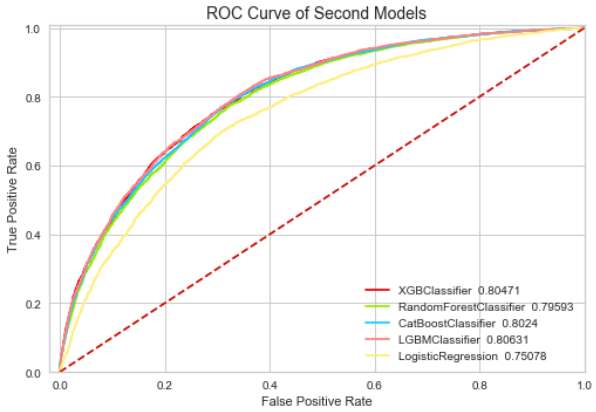


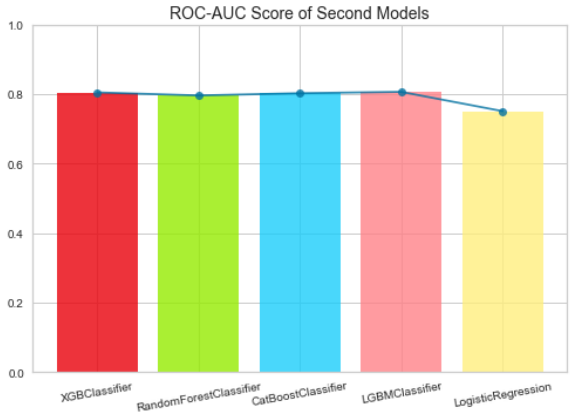
Before applying the VotingClassifier algorithm in Ensemble modeling, I made the adjustment for the imbalance problem. I added some records from the Customers dataset to the Mailout-Train dataset instead of OverSampling artificially. This second model yielded better results, albeit less than the first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

Values of
Receiver Operating Characteristic





When I comparing **Base Model**, **Improved Model**, and **Improved Control**, I saw VotingClassifier and LGBMClassifier will give us the best result. Because of that, the fine tuned VotingClassifier **will be used with the Mailout-Test** set.
The **Improved Model best model score**d around **0.815448 from LGBM** and **0,871796 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.

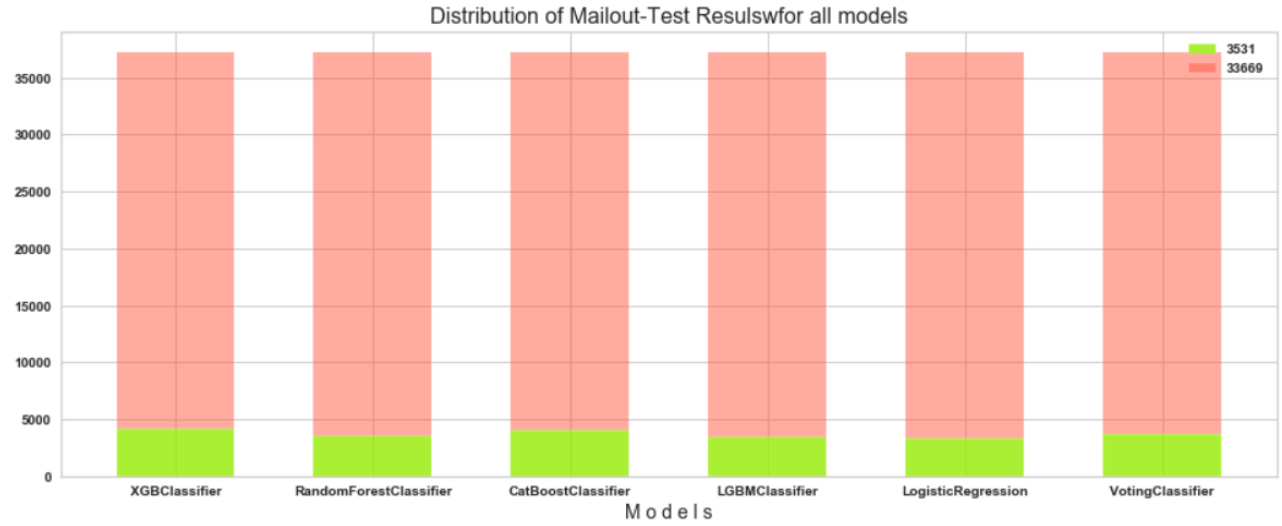
```
# Generate CSV file for the Kaggle competition
df_kaggle.to_csv('../Last/data/kaggle_submission_file.csv', index=False)
```

```
# 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
11786	30444	0.232406	0
29500	1666	0.058297	0
9166	67228	0.318163	0
8994	51228	0.213161	0
34208	566	0.296799	0
21501	54403	0.117465	0

	XGBClassifier	RandomForestClassifier	CatBoostClassifier	LGBMClassifier	LogisticRegression	VotingClassifier
0	32987	33579	33762	33132	33851	33542
1	4213	3621	3438	4068	3349	3658



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.

RESULT

- 4 clusters were obtained with **Kmean**. If **Cluster 3** and **Cluster 1** are chosen in the clusters, it can get positive results quickly. Cluster 3 and Cluster 1 are the **very best segment for potential customers**.

- This process is classifier. Because of this, I choose and tried **Lightgbm, XGBoost, CatBoost, Random Forest, Logistic Regression** classifier models by the library of Sklearn. *"Classifier models based on tree algorithms are robust to outliers. Tree algorithms split the data points on the basis of the same value and so the value of outlier won't affect that much to the split."(6)*

  A Voting classifier model **combines multiple different models into a single model**, which is **more powerful than any of the individual models** alone. *"The idea behind the VotingClassifier is to combine conceptually different machine learning classifiers and use a **majority** vote, **weighting**, or the **average** predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses." (3)* **Within this context, I combined the cluster models that I use with VotingClassifier according to their weight b**y defining high weight to models that give good scores.

```
eclf = VotingClassifier(
    estimators = [('lgb_stack', lgb_BP_my),
                  ('xgb_stack', XGB_BP),
                  ('cat_stack', Cat_BP_my),
                  ('randf_stack', RandF_BP),
                 ],
    voting='soft', weights=[3,3,2,1], flatten_transform=True)
```

VotingClassifier hosted with ❤ by GitHub view raw

- My best result for **ROC-AUC Score** from these models is **0.79613 from Lightgbm**. After new oversampling to data set **improved best ROC-AUC scored** around

0.8063 from Lightgbm and **other one is 0,8588 from VotingClassifier**. Use predicting of the test label using this VotingClassifier model.

- Also Mailout-Test results are {1: 3531, 0 : 33669} with **VotingClassifier** model.

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.**
- Two challenges of this project is the large data size and the data imbalance. Cleaning of this big data and applying GridSearcheCV to the models also requires serious time and machine performance. For the solution, it should spend some more time to get to know the columns and seek high performance without disabling the important columns.
- Implementing GridSearchCV with **10–128 variations** for each model was **a mistake for me**. Was be wiser to focus only on **LGBM** and **XGBoost** models.

- Public code for this analysis may be found in its Github repository.

Some resources used for this project:
(1) Visualization
(2) H2O.aiDocumentation
(3) Ensemble-Voting Classifier
(4) For GridSearch
Some Articles: (6), (7), (8), (9), (10), (11)