Machine Learning Engineer Nanodegree

Udacity

Capstone Project Proposal:

Customer Segmentation (Analysis)

Miguel Diaz

March 24th, 2020

Table of contents

Project Overview

This project, besides being one of the proposed for the Udacity Machine Learning Nanodegree, is part of how Bertelsmann Arvato's clients can acquire new customers based on their mail-order marketing campaigns. In other words, we need to predict which individual's types of the general population can be used to target good or bad responders to the campaign.

In order to solve the problem, we were given 2 main datasets about the general population demographics, which is about customers and clients response of prior campaigns. There are also some specifications and conditions which include privacy and sensitive data.

The project is divided in 3 parts:

1. Unsupervised Learning: Create a customer segmentation mode and then identify segments of the German population.
2. Supervised Learning: Create supervised models to calculate the probability of customer's conversion from the general population data.
3. Kaggle submission: Submit the test predictions.

Problem Statement

The problem statement is no other than finding the best way the company can acquires new client base in an efficient way but is important also to have a solid business background to save time and sources.

Solution Statement

The problem stated before proposes to apply Machine Learning techniques to predict good recipients by analyzing data of specific populations and their relationships between them.

We need to use unsupervised and supervised machine learning techniques, but before we apply them, we have to analyze the best possible algorithms for the specific purposes of this problem.

For the first task we need to clean and apply techniques for unsupervised modeling, for example: encoding data, feature scaling and handling missing data in order to not affect the results. We will use PCA for dimensionality reduction and identify variables (principal components) whose behavior as a function of simulation parameters can expose the presence of phase transition; and a K-Means algorithm as a way of splitting up the cluster and as part of the prediction process. Besides being one of the most popular among clustering algorithms, it has a high dependency on the initial conditions so the implementation could change according to the size of the dataset. We could implement recursive and parallel approximation to this algorithm in order to scale well on both the number of instances and dimensionality of the problem.

When we get the cluster segments, we will apply the supervised learning models to test them and analyze their final results to compare them and choose the best approach. We will use:

a. Logistic Regression
b. XGBoostClassifier (Decision Tree Regressor)

As Machine Learning engineers, we sometimes need to adjust our approaches to get valuable insights and the best possible results, so we need to be flexible and try different experiments.

Evaluation Metrics

What we want to solve is the problem of customer segmentation, by applying machine learning techniques on demographics and customer data.

At first place, we implement a K-Means, because we need to reduce high-dimensional data set into fewer dimensions while retaining significant data, allowing to untangle data into independent components. So, it would be possible to identify the segments. For the second step, in the supervised learning modeling, it’s important to use precision, recall, accuracy, F1, AUC as main metrics to evaluate the model performance and customer classifications. The outputs of this analysis are part of a marketing strategy to identify potential clients.

Analysis

a. Data Exploration and Exploratory Visualization

In this section, we explore the datasets in order to get some insights and patterns. Then we decide, according to our business problem, transform, delete or omit changes in the data.

The data is contained in 4 datasets and 2 metadata files:
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 the 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).

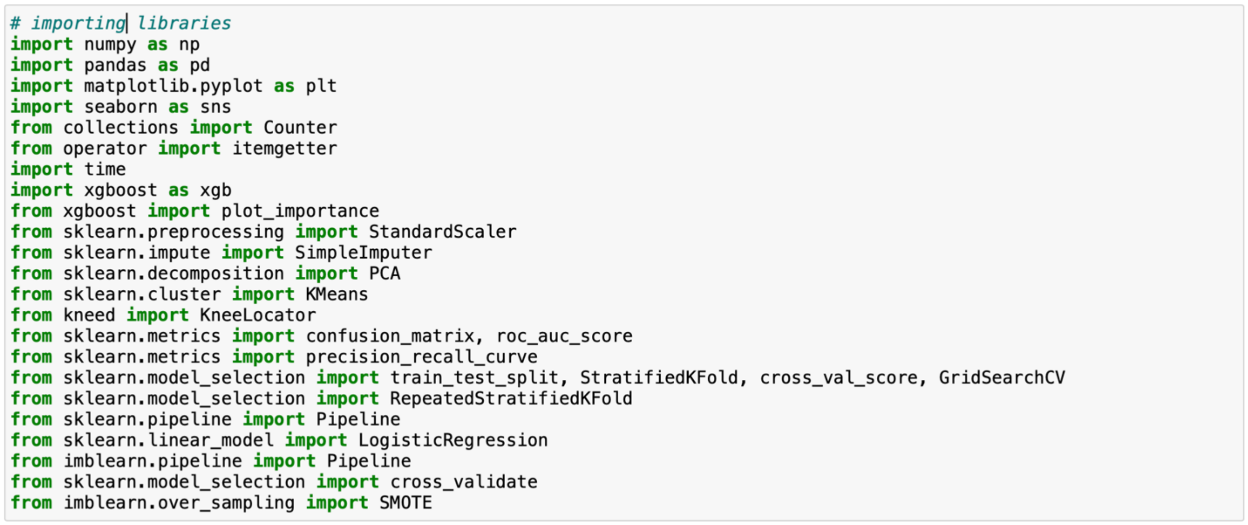
Udacity_MAILOUT_052018_TEST.csv: Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

DIAS Information Levels – Attributes 2017.xlx: Top-level list of attributes and descriptions, organized by informational category.

DIAS Attributes – Values 2017.xlsx: Detailed mapping data values for each feature in alphabetical order.

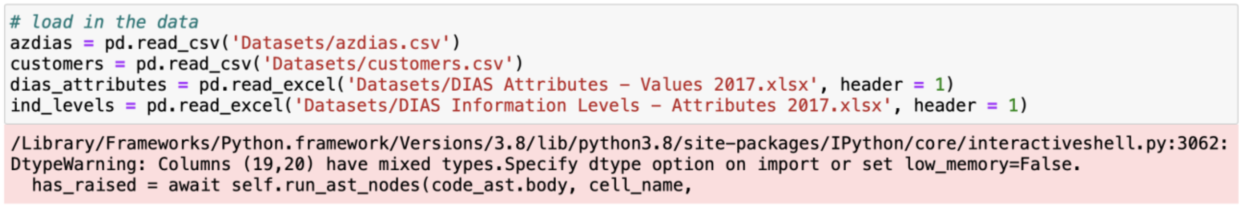
We explored dataset by dataset, in order to find patterns and then we could make some conclusions about data exploration.

First of all, we loaded the datasets in order to work with them. Besides that, we also loaded some useful python libraries and sklearn libraries to work the models.



```
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from operator import itemgetter
import time
import xgboost as xgb
from xgboost import plot_importance
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from kneed import KneeLocator
from sklearn.metrics import confusion_matrix, roc_auc_score
from sklearn.metrics import precision_recall_curve
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score, GridSearchCV
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from imblearn.pipeline import Pipeline
from sklearn.model_selection import cross_validate
from imblearn.over_sampling import SMOTE
```

Figure 1: Importing python libraries.



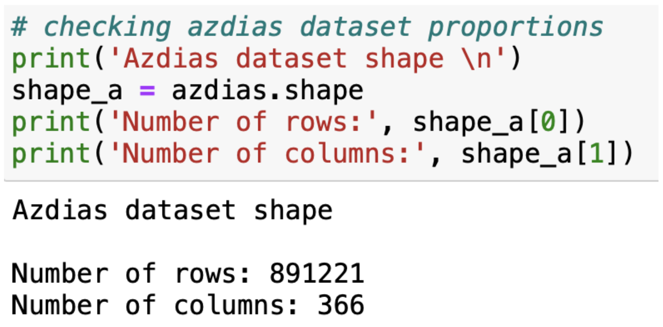
```
# load in the data
azdias = pd.read_csv('Datasets/azdias.csv')
customers = pd.read_csv('Datasets/customers.csv')
dias_attributes = pd.read_excel('Datasets/DIAS Attributes - Values 2017.xlsx', header = 1)
ind_levels = pd.read_excel('Datasets/DIAS Information Levels - Attributes 2017.xlsx', header = 1)
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3062:
DtypeWarning: Columns (19,20) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Figure 2: Loading data.

For the azdias datset we got these proportions:

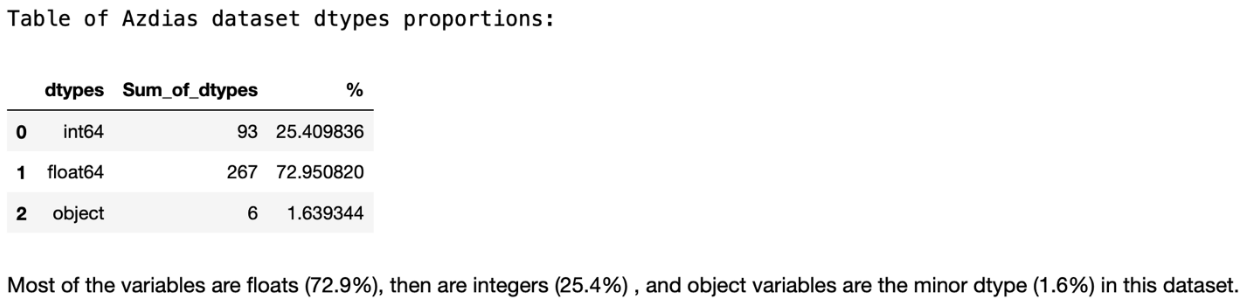
Most of the variables were floats (72.9%), integers (25.4%), and object variables were the minor data types (1.6%) in this dataset.



```python
# checking azdias dataset proportions
print('Azdias dataset shape \n')
shape_a = azdias.shape
print('Number of rows:', shape_a[0])
print('Number of columns:', shape_a[1])
```

```
Azdias dataset shape

Number of rows: 891221
Number of columns: 366
```

Figure 3: Azdias Dataset proportion.



```
Table of Azdias dataset dtypes proportions:
```

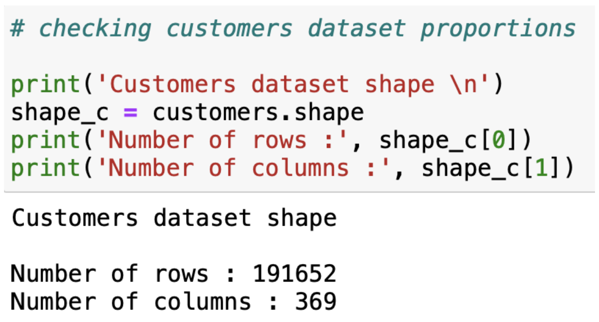
	dtypes	Sum_of_dtypes	%
0	int64	93	25.409836
1	float64	267	72.950820
2	object	6	1.639344

Most of the variables are floats (72.9%), then are integers (25.4%) , and object variables are the minor dtype (1.6%) in this dataset.

Figure 4: data types proportions.

For the customers dataset we got these proportions:

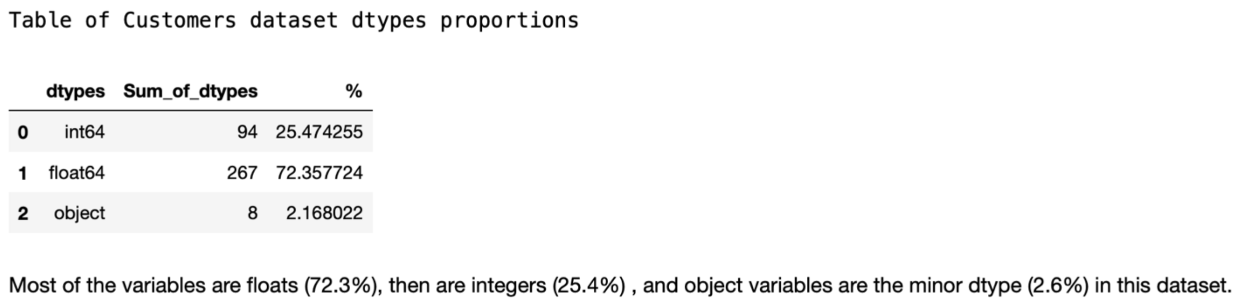
Most of the variables were floats (72.3%), integers (25.4%), and object variables were the minor data types (2.6%) in this dataset.



```python
# checking customers dataset proportions

print('Customers dataset shape \n')
shape_c = customers.shape
print('Number of rows :', shape_c[0])
print('Number of columns :', shape_c[1])
```

```
Customers dataset shape

Number of rows : 191652
Number of columns : 369
```

Figure 5: Customers proportions.



```
Table of Customers dataset dtypes proportions
```

	dtypes	Sum_of_dtypes	%
0	int64	94	25.474255
1	float64	267	72.357724
2	object	8	2.168022

Most of the variables are floats (72.3%), then are integers (25.4%) , and object variables are the minor dtype (2.6%) in this dataset.

Figure 6: Customers proportions.

Some aspects we considered:

- It was helpful to delete variables with certain threshold of missing values.
- 'CAMEO_DEU_2015' is alphanumeric, so we could convert it with integer encoding or one-hot encoding, but it depends on the real meaning of the variable, in order to not affect the results. It seems like a score value.
- There were 3 variables that were not common in azdias and customers datasets were: 'CUSTOMER_GROUP', 'ONLINE_PURCHASE', 'PRODUCT_GROUP', so it would be helpful to delete them.

We transformed some variables to model them, according to our business problem. The number of common columns between both datasets were 366 features, but there were 3 that don't exist in azdias dataset: 'CUSTOMER_GROUP', 'ONLINE_PURCHASE', 'PRODUCT_GROUP'.

The encoded values with missing or unknown values were converted to NANs to work with all the features. This were applied to azdias dataset first and then to the customers dataset.

And working with missing values we got this:

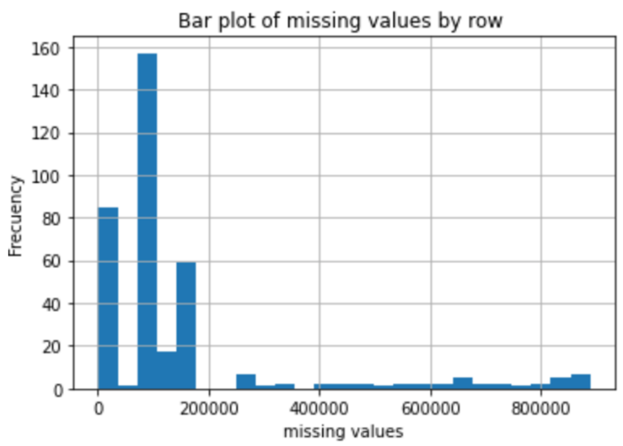


Figure 7: Distribution of missing values by column of the azdias dataset.

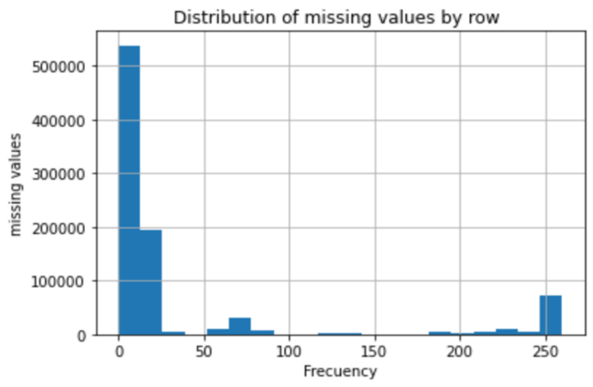


Figure 8: Distribution of missing values per row of the azdias dataset.

We removed columns which had > 600000 missing values, we got 891221 rows and 340 variables. In other words, we deleted 7% of the total variables (340/366). We decided to get most of the variables for better insights. With the row's reduction, we obtained 737288 columns and 340 columns. We reduced 17.3% (737288/891221) of rows for this section.

And for customers dataset we got this:

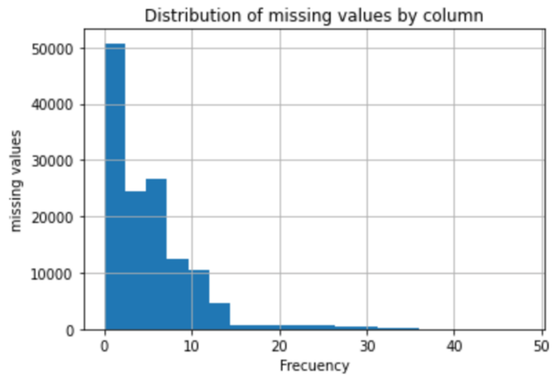


Figure 9: Distribution of missing values of Customers dataset.

We remove also the same threshold as above and we chose to remove rows which contained more than 50 missing values per row. We cleaned the data. Finally, we got 134246(- 30%) rows, and 340 (-8%) columns.

b. Algorithms and Techniques

We chose 4 algorithms. For the unsupervised learning, we selected a PCA to bring out strong patterns in the datasets, and a K-means clustering, because it divides the unlabeled data into k different clusters, grouping by similarities and properties. And for the supervised learning, we chose a Logistic Regression Classifier and a XGBoost

Classifier. Because the first is easy to implement and for binary classification tend to perform well. And the second, because it provides highly scalable and parallelizable techniques, and is quick to execute.

c. Benchmark

We got different scores applying 2 powerful algorithms, applying sampling and normalization in the data:

	Unscaled			Scaled		
Models	Score	Cross-V mean	Time (s)	Score	Cross-V mean	Time (s)
Logistic R. Classifier	0.67	0.668	10343.9	0.686	0.680	862.8
XGboost Classifier	0.776	0.770	41579.5	0.782	0.780	18191.8

Figure 10: Model performance comparison.

XGBoost it the best model according to the previous scores, in this case the Roc-auc score improved with GridSerach.

Methodology

a. Data Processing

We decided to remove for simplicity the following features in both datasets: 'CAMEO_DEU_2015',' CAMEO_DEUG_2015', 'CAMEO_INTL_2015',' D19_LETZTER_KAUF_BRANCHE', 'EINGEFUEGT_AM' (they were alpha-numeric variables or don’t have enough information to transform them). Just transformed one categorical variable ‘OST_WEST_KZ’.

b. Implementation

Dimensionality Reduction

Before scaling the data, we just analyzed the kurtosis and skewness of the datasets in order to know about the data distribution. And the final result was that almost all the data followed a normal distribution, so we applied the Standard Scaler function.

We preferred to impute the missing values with the mean, this method is not the best ideal because out model could fall in bias and add more variance. To maintain 90% of variance, the number of components required were 173. According to the plot:

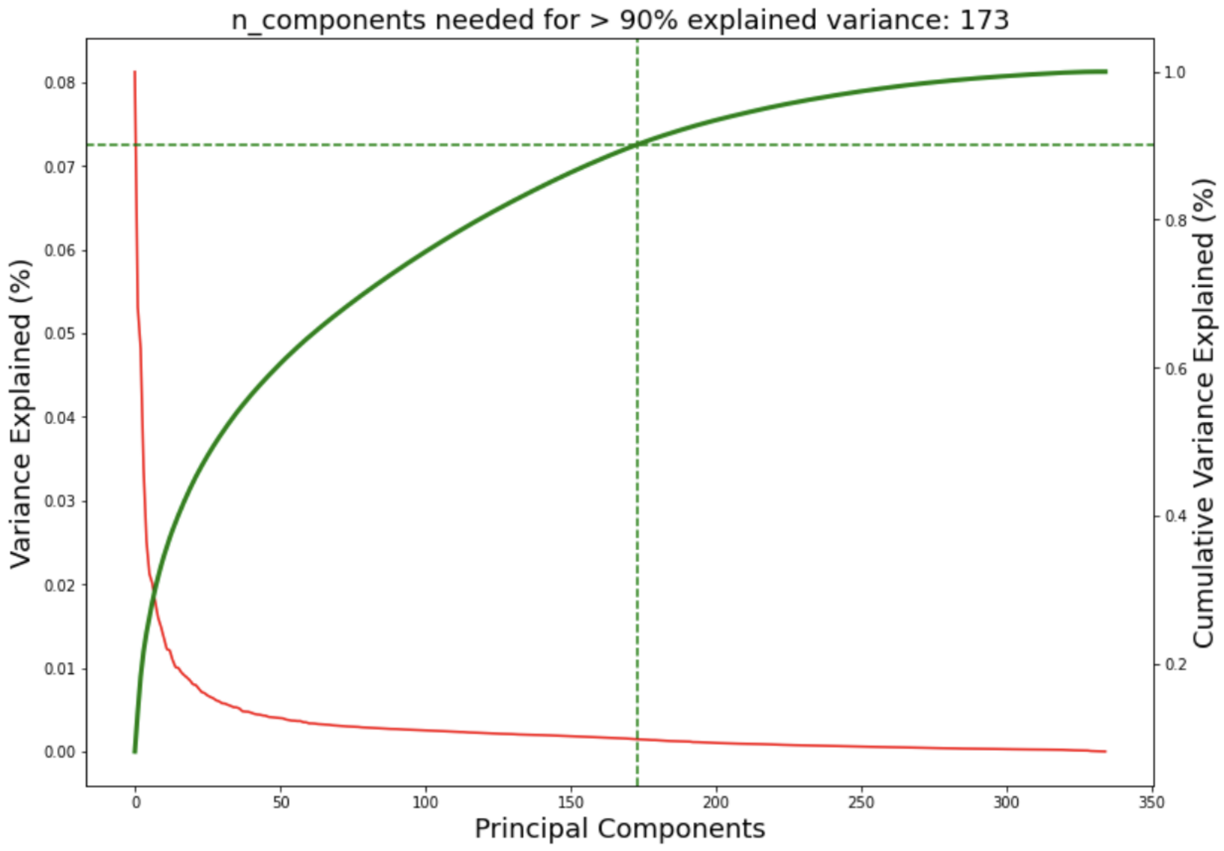


Figure 11: Variance and Cumulative Variance of Principal Components.

The level of cumulative variance is explained by including 173 components. We choose these components which explain 90% of the variance.

We calculated the weights and their direct features in order to get a deeper understanding. Also, we tested “N" principal components, to know the relationships with the features and compared them.

The weights of the variables helped us to determine the position and the relationship between them. For example, if the weight is far from zero, the more the principal component is in the same direction of the feature. If 2 features have large weights in the same side (+/-), then increases in one are associated with increases in the other feature. In the contrary, features with different signs can show negative correlation.

Unsupervised Learning (K-means)

We created the K-means clustering to analyze how data behaves in the principal component’s quadrant. The optimal number for clusters is 7. According to the next graph.

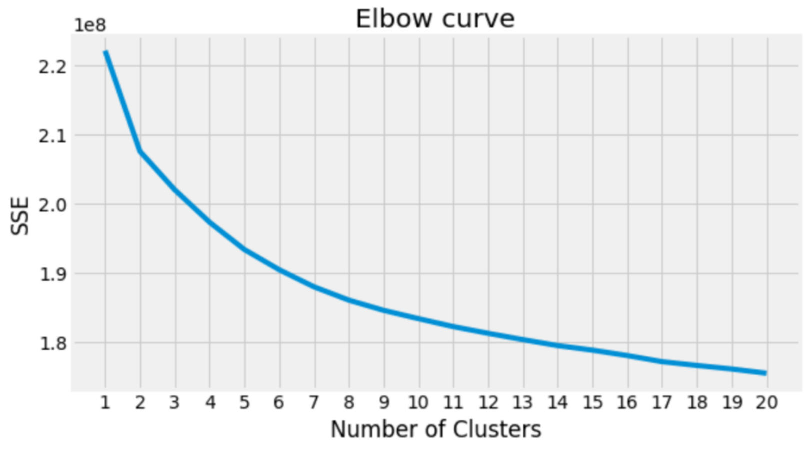


Figure 12: Elbow method.

We created the clusters on the demographics data and observed how customer data behaves on the clusters, also analyzed the cluster distributions because we wanted to identify the most important customers, according to the company's question.
The next plot showed us the distribution of clusters from the azdias and custumers datasets. We observed in cluster 3, that the customers density is greater than the general population density, so this could be a good cluster to interpret it as a good one for being part of the main customer base for the mail-order company's customer base, than the others.

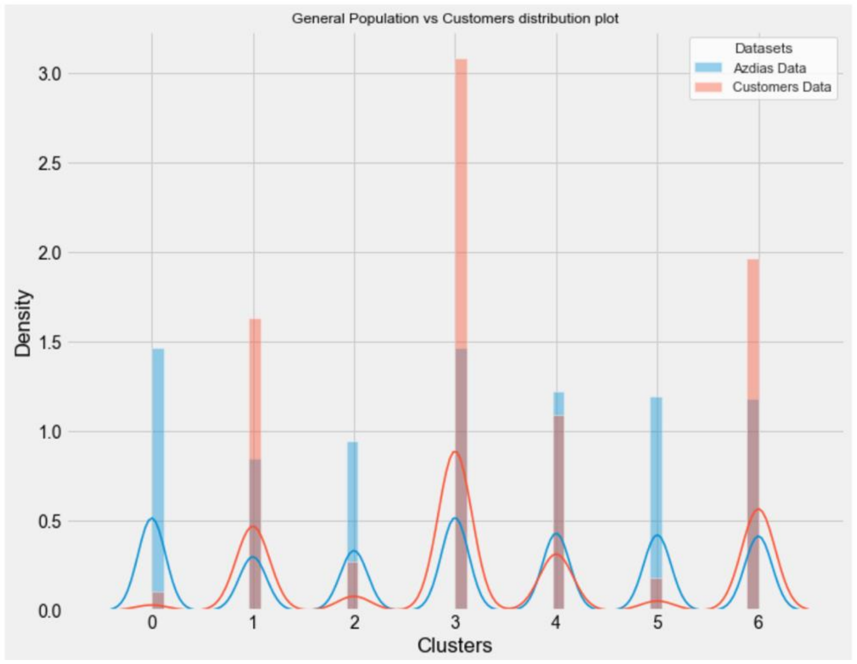


Figure 13: Data distributions in the clusters.

Customers were represented by the cluster number 1,3 and 6 mostly. They could be the target we were looking for. On the other side, the clusters number 0, 2 and 5 were the ones with the most underrepresented among all.

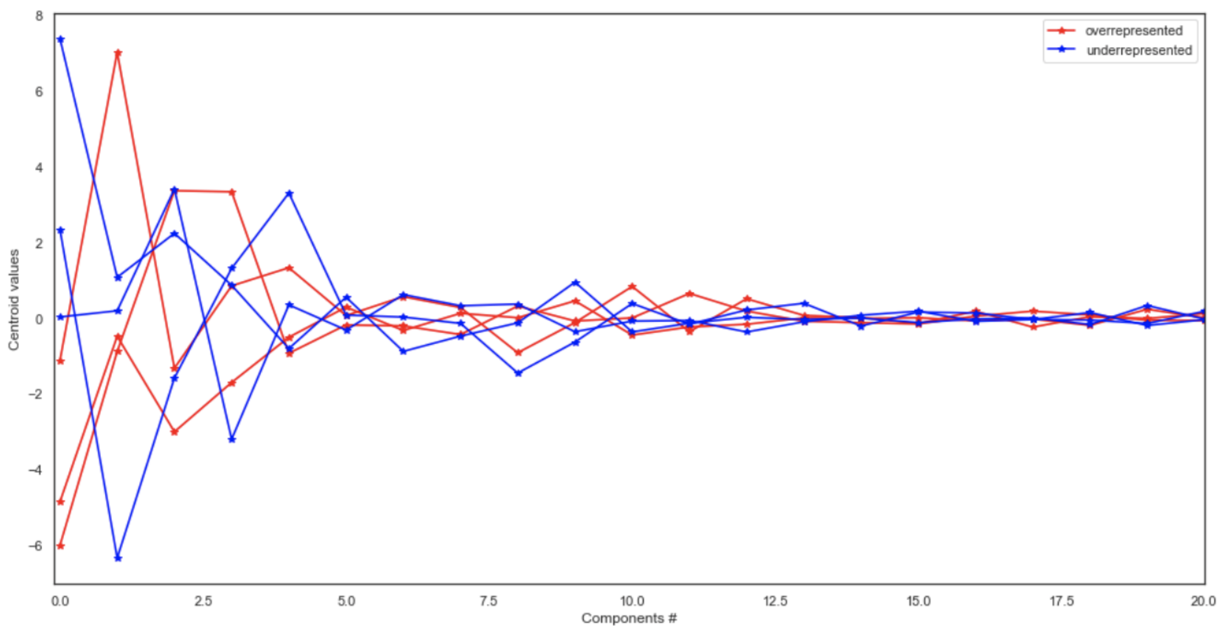


Figure 14: Overrepresented and underrepresented features with centroids and components (clusters centers: 1,3,6 as overrepresented and 0,2,5 as underrepresented).

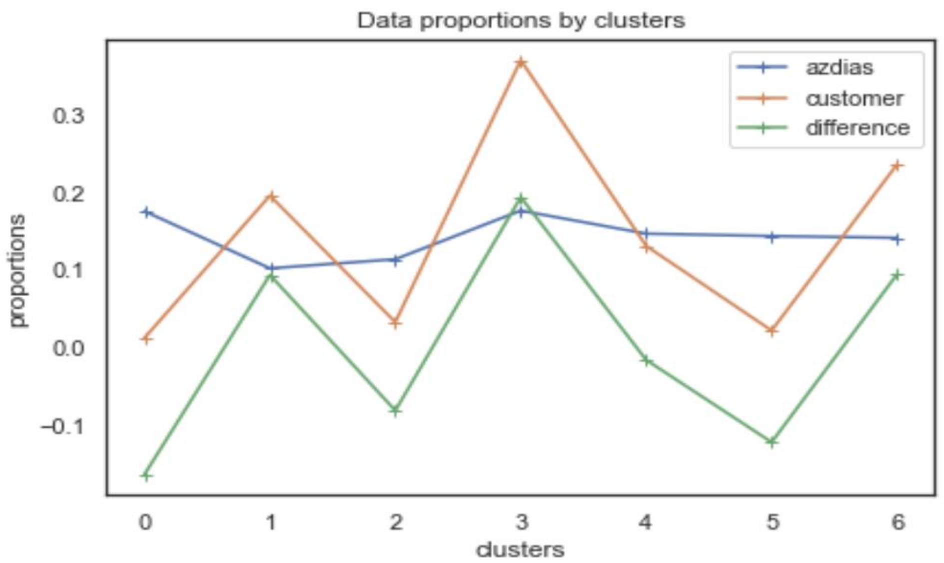


Figure 15: Data proportions by clusters.

Supervised Learning (Logistic Regression Classifier and XGBoost Classifier)

Before we dive into the algorithms for modeling, we visualized how the target variable (positive and negative responses) behaved in the clusters.

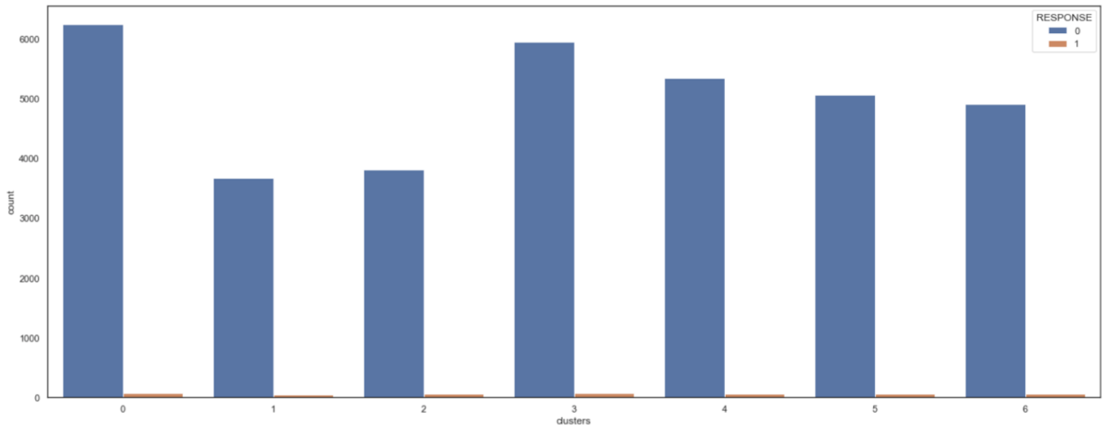


Figure 16: Response target variables distribution in clusters.

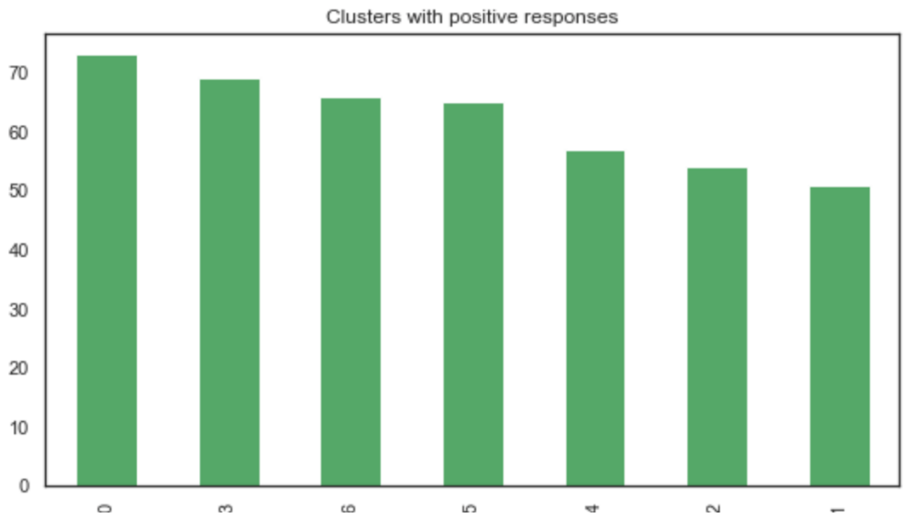


Figure 17: Positive Responses distribution in clusters.

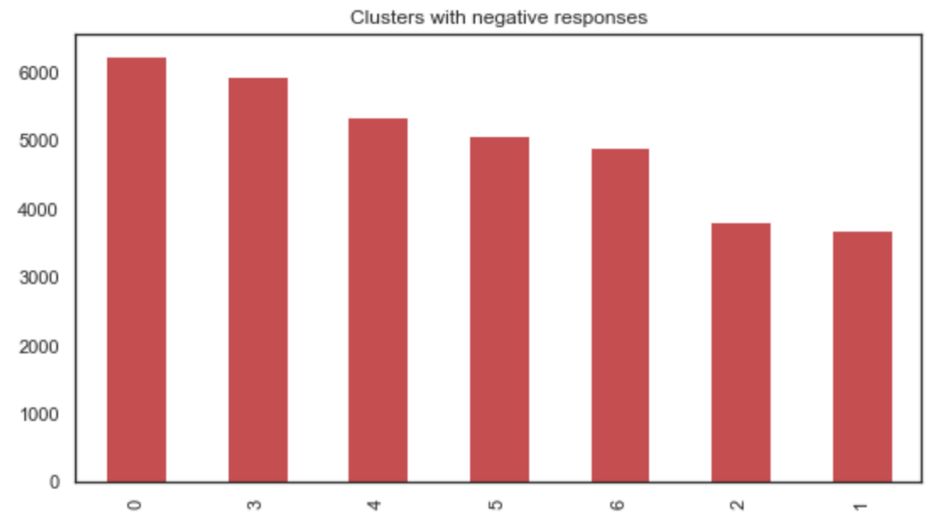
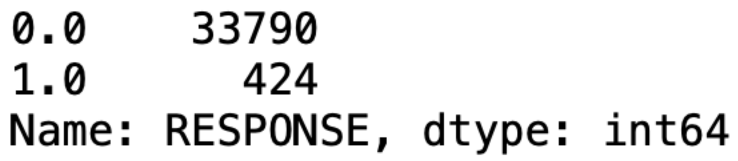


Figure 18: Negative Responses distribution in clusters.

We got that cluster 0 has the biggest difference of responses among all clusters. The green and red plots show us the positive and negative responses, respectively, of the train data. This is an imbalanced classification, because the negative responses were the most repeated classification.



```
0.0    33790
1.0      424
Name: RESPONSE, dtype: int64
```

Figure 19: Proportions of the target variable Response, (0 = no customer email, 1 = customer).

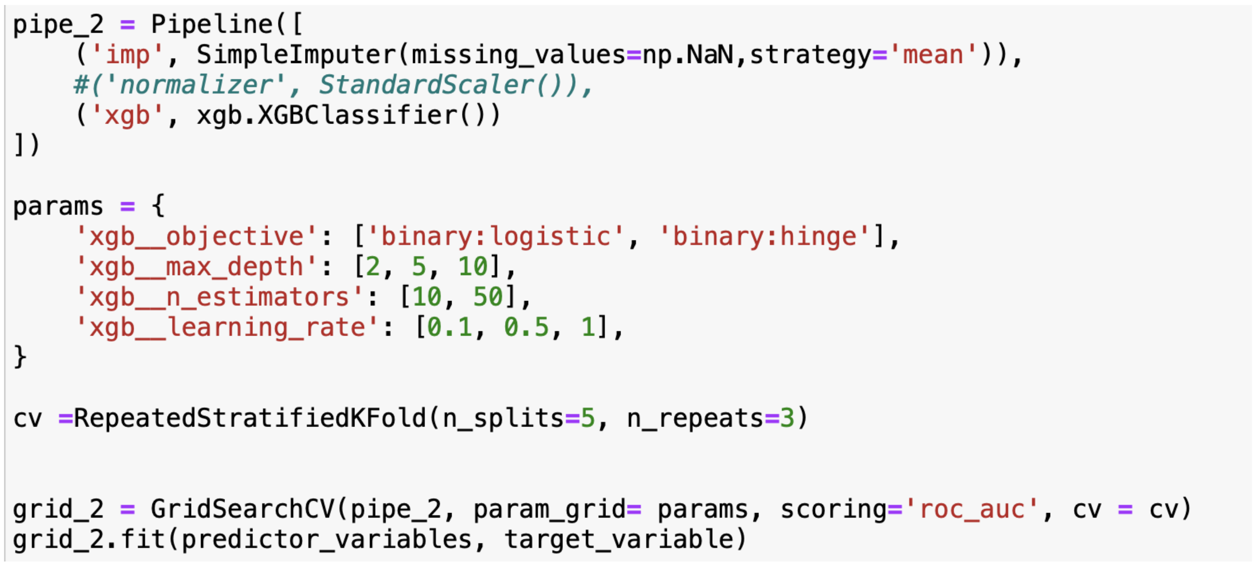
For the imbalanced classification we can use the ROC_AUC and Precision-Recall AUC. The first because it a helpful metric under a severe class imbalance dataset (when the number of examples in the minority class is small). And the second, is a helpful diagnostic tool for evaluating a single classifier but challenging for comparing classifiers. But as we don’t have the target variable in the mailout test, we rely by the moment, just with the first score.

Before we applied the Logistic Regression Classifier algorithm, we imputed the data with the mean, then we normalized it with StandarScaler function, and we sampled with Smote function (balance the minority class), and finally applied Gridserach to test the best parameters for the algorithm.

Also, before we applied the XGBoost Classifier algorithm, we imputed the data with the mean, then we normalized it with StandarScaler function, and finally applied Gridserach to test the best parameters for the algorithm.

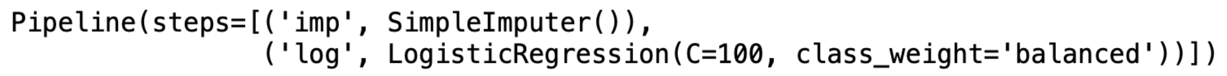
c. Refinement

Before we applied the models, we applied Gridsearch for each model, and we tested for the best possible parameters. Also, we applied RepeatedStratifiedKFold in order to improve the estimated performance of the models, which the mean is expected to be a more accurate estimate of the true unknown underlying mean performance of the model on the dataset.



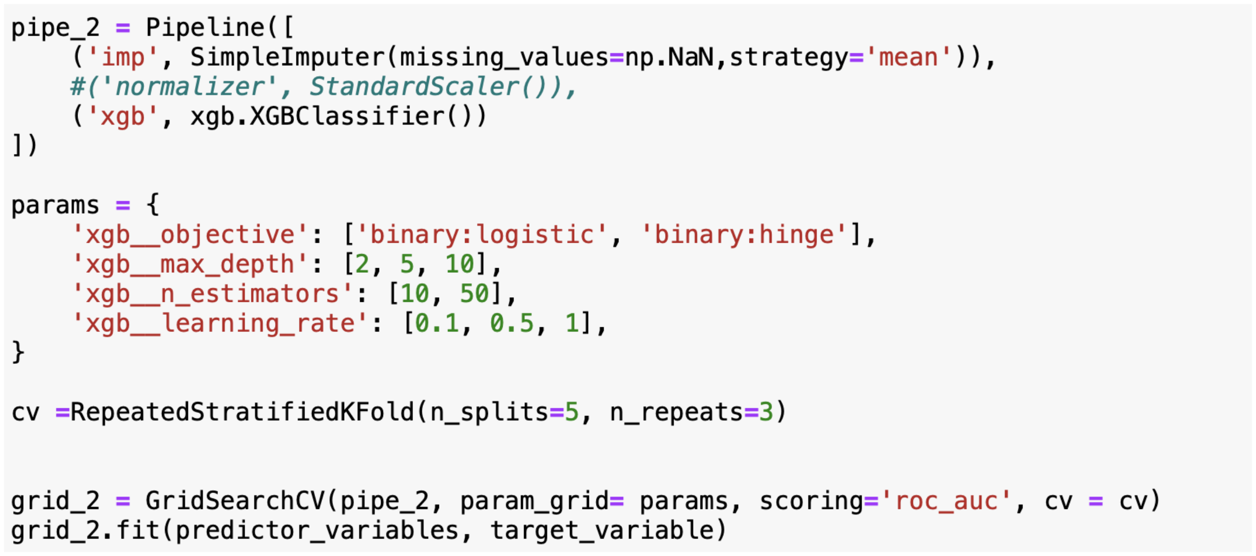
```
pipe_2 = Pipeline([
    ('imp', SimpleImputer(missing_values=np.NaN,strategy='mean')),
    #('normalizer', StandardScaler()),
    ('xgb', xgb.XGBClassifier())
])

params = {
    'xgb__objective': ['binary:logistic', 'binary:hinge'],
    'xgb__max_depth': [2, 5, 10],
    'xgb__n_estimators': [10, 50],
    'xgb__learning_rate': [0.1, 0.5, 1],
}

cv =RepeatedStratifiedKFold(n_splits=5, n_repeats=3)


grid_2 = GridSearchCV(pipe_2, param_grid= params, scoring='roc_auc', cv = cv)
grid_2.fit(predictor_variables, target_variable)
```

Figure :19 Setting GridSearch and Repeated Strarified K Fold for Logistic Regression Classifier algorithm.



```
Pipeline(steps=[('imp', SimpleImputer()),
                ('log', LogisticRegression(C=100, class_weight='balanced'))])
```

Figure 20: Best parameters for Logistic Regression Classifier algorithm.



```
pipe_2 = Pipeline([
    ('imp', SimpleImputer(missing_values=np.NaN,strategy='mean')),
    #('normalizer', StandardScaler()),
    ('xgb', xgb.XGBClassifier())
])

params = {
    'xgb__objective': ['binary:logistic', 'binary:hinge'],
    'xgb__max_depth': [2, 5, 10],
    'xgb__n_estimators': [10, 50],
    'xgb__learning_rate': [0.1, 0.5, 1],
}

cv =RepeatedStratifiedKFold(n_splits=5, n_repeats=3)


grid_2 = GridSearchCV(pipe_2, param_grid= params, scoring='roc_auc', cv = cv)
grid_2.fit(predictor_variables, target_variable)
```

Figure 21: Setting GridSearch and Repeated Strarified K Fold for XGBoost Classifier algorithm.

*

Figure 22: Best parameters for XGboost Classifier algorithm.

Once we applied the parameters for each algorithm, the final score results tend to change a little bit.

Results

a. Model Evaluation and Validation

For the Dimensionality Reduction analysis, we got these results:

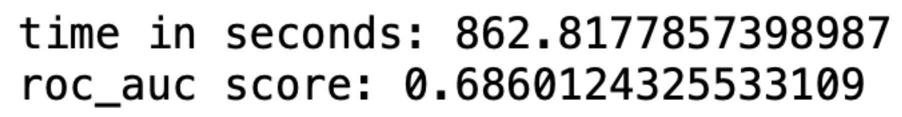
- For component 1, the top 5 positive features weights are related to 6-10 family houses in the PLZ8, number of >10 family houses in the PLZ8; and the top negative are social status fine, social status rough, moving patterns and number of 1-2 family houses in the PLZ8.
- For component 2,the top 5 positive features are related to number of cars with less than 5 seats in the PLZ8, share of BMW within the PLZ8, share of MERCEDES within the PLZ8,share of BMW & Mercedes Benz within the PLZ8; and the top 5 negative weights are share of small and very small cars (Ford Fiesta, Ford Ka etc.) in the PLZ8, share of cars with max speed between 140 and 210 km/h within the PLZ8, share of car owners between 21 and 25 within the PLZ8, share of cars with max speed between 110 km/h and 180km/h within the PLZ8.
- For component 3, the top 5 positive features weights are related to main age within the household, financial typology: money saver, dominating movement in the person's youth (avantgarde or mainstream); and the top 5 negatives weights are age classification through prename analysis, financial typology: be prepared.
- For component 6, the top 5 positive weights features are related to affinity indicating in what way the person is familiar minded, affinity indicating in what way the person is cultural minded, affinity indicating in what way the person is social minded, affinity indicating in what way the person is dreamily; and the top 5 negative weights are gender, affinity indicating in what way the person is dominant minded, affinity indicating in what way the person is eventful orientated, affinity indicating in what way the person is of a frightful attitude.

For the Unsupervised Learning analysis, we got the next results:

- For cluster number 3 as it was the most overrepresented cluster for the customer data, the components with the highest and negative weights were number 10 and 1 respectively.
- For cluster number 1, as it was the middle overrepresented cluster for the customer data, the components with the highest and negative weights were number 3 and 1 respectively.
- For cluster number 5, as it was the most underrepresented cluster for the customer data, the components with the highest and negative weights were number 1 and 7 respectively.

And for the supervised learning part, we got the following results (We select the scaled data because we saw improvements in the final results):

For the Logistic Regression Classifier, the time and Roc-auc score were:

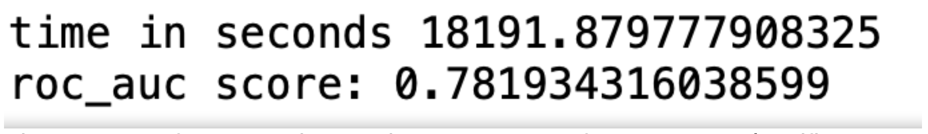


```
time in seconds: 862.8177857398987
roc_auc score: 0.6860124325533109
```

Figure 23: Performance time and Roc-auc score for Logistic R. Classifier.

When we applied cross validation score, we observed the function helped us to evaluate the roc-auc score by cross-validation, so in this case we got a mean of 0.680. It's almost the same as the previous calculated (0.684).

And for the XGBoost Classifier, the time and Roc-auc score were:



```
time in seconds 18191.879777908325
roc_auc score: 0.781934316038599
```

Figure 24: Performance time and Roc-auc score for XGBoost Classifier.

When we applied cross validation score, we observed the function helped us to evaluate the Roc-auc score by cross-validation, so in this case we got a mean of 0.780. It's almost the same as the previous calculated (0.782).

Feature importance for the XGboost model:

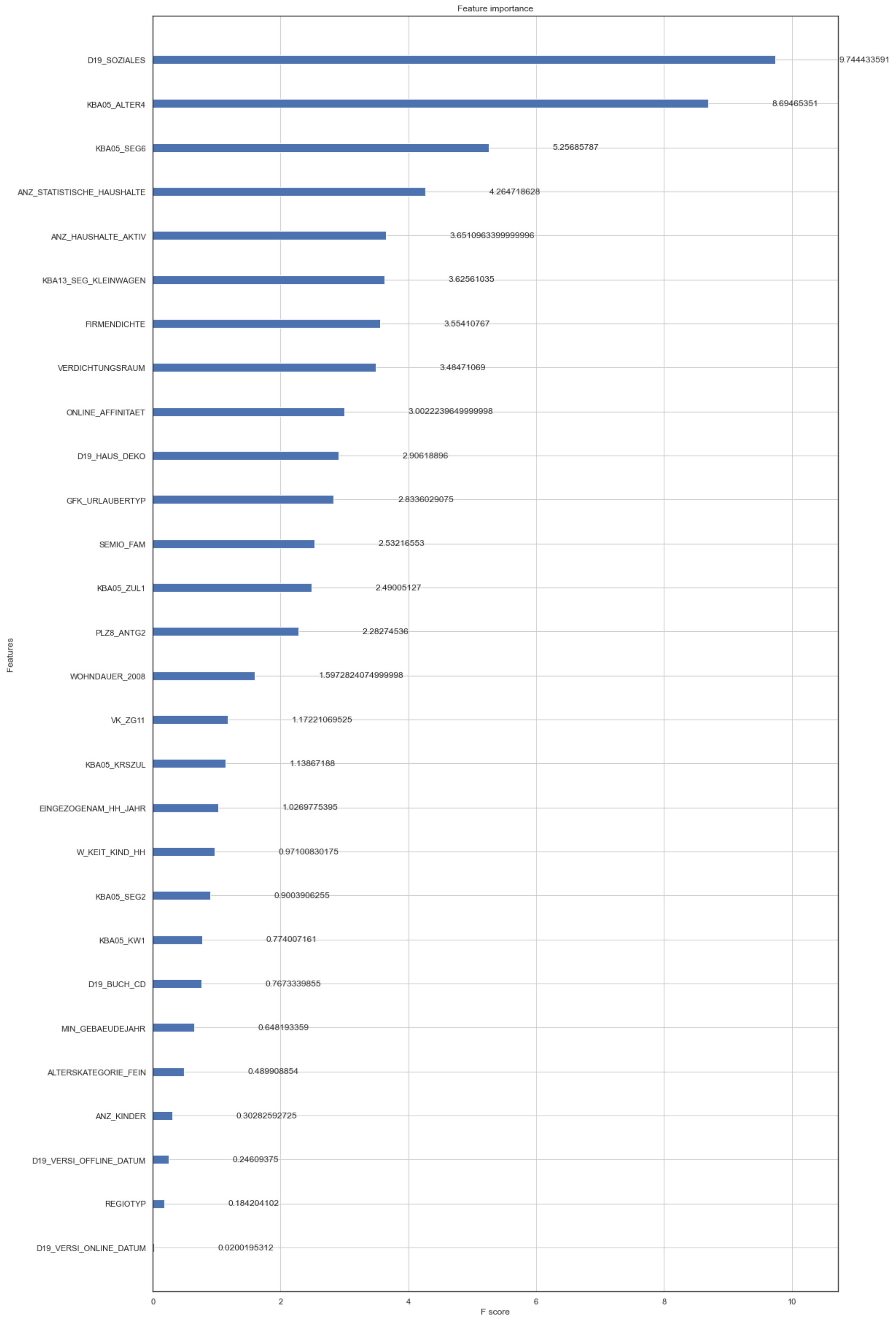


Figure 25: XGBoost Feature importance.

b. Justification

In this analysis, two distinct algorithms were trained to identify patterns in the general population and predict the category of customers or no customers.

The benchmark algorithm is a Logistic Regression Classifier that’s a statistical method for predicting score for observations, the dependent variables follow Bernoulli Distribution, and the estimation is done though maximum likelihood.

The proposed algorithm is a XGBoost Classifier that’s implements decision trees with boosted gradient, enhanced performance and speed. Also provides a parallelization in tree building and cache optimization.

The final results demonstrate that both have high scores. Hence, the time performance seems to be huge between them.

Conclusion

Once we got the results, we need to present the results to the marketing team and the directors in order to build a strategic plan in the short and middle term, this to tackle the most appropriate offers to send to each customer.

We must consider every detail in a dynamic environment. We should implement constant and straight analysis by customer to avoid waste of resources and time; and tire out the customer with unnecessary messages.

Past events help to decide and build new strategies and value the current facts. So, we should consider a detailed analysis to implement certain algorithms according to our business problem and be careful with the time we want to study.

We tried a good approach, but we need to discuss it as a whole team and try another test and verify new methodologies or verify the data sources.

Table of Figures

References

[1] R. M. Woloshyn, "Learning phase transitions: comparing PCA and SVM", ArXiv, 2019.

[2] M. Capo, A. Perez, and J. A. Lozano, "An efficient K-means clustering algorithms for massive data", ArXiv, vol. 14, no. 8, 2015.

[3] C.P. Ezenkwu, S. Ozuomba, C. Kalu, "Application of K-means algorithm for efficient customer segmentation: A strategy for targeted customer services", IJARAI, vol. 4, No. 10, 2015.

[4] P.C. Chaitra, K. Saravana, "A review of multi-class classification algorithms", IJPAM, vol. 118, No. 14, 2018

[5] C. Bentejac, A. Csorgo, G. Martinez-Munoz, "A comparative analysis of XGBoost", ArXiv, 2019.