Capstone Project: Create a Customer Segmentation Report for Arvato Financial Services

By: Jefferson de Souza Santos

Content

1.Introduction

In this project, demographic data from an email database of a German company will be analyzed, comparing them with data from the general population.

We have two main challenges:

1. Understand how the customer base relates to the general population base;
2. Make a prediction of the propensity of customers to respond to emails that the company sends.

For the first challenge, a customer segmentation technique will be used with the creation of population clusters and with that we will replicate to the customer base and understand how the two are similar.

For the prediction challenge, we will test a series of algorithms to understand which one can better predict customers who might respond to emails through AUC and recall metrics.

2.Cluster Segmentation

For this part of the project we started by exploring the available bases. There are two bases to be worked on, which are:

- 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).

And two support bases, in which we have essential information about the descriptions of the bases.

2.1. Data Cleaning

A primeira etapa do nosso data cleaning foi entender se tínhamos informações sobre todas as features da base. Ao realizarmos encontramos uma série de features sem informações que precisaram ser retiradas, uma vez que não podemos inferir nada sobre elas.

In [11]:
```
y = [col for col in azdias.columns if col not in features_info.index]
print("Columns without context ", y)
```

```
Columns without context  ['LNR', 'AKT_DAT_KL', 'ALTER_KIND1', 'ALTER_KIND2', 'ALTER_KIND3', 'ALTER_KIND4', 'ALTERSKATEGORIE_FEIN', 'ANZ_KI
NDER', 'ANZ_STATISTISCHE_HAUSHALTE', 'CAMEO_INTL_2015', 'CJT_KATALOGNUTZER', 'CJT_TYP_1', 'CJT_TYP_2', 'CJT_TYP_3', 'CJT_TYP_4', 'CJT_TYP_
5', 'CJT_TYP_6', 'D19_BANKEN_ANZ_12', 'D19_BANKEN_ANZ_24', 'D19_BANKEN_DIREKT', 'D19_BANKEN_GROSS', 'D19_BANKEN_LOKAL', 'D19_BANKEN_REST',
'D19_BEKLEIDUNG_GEH', 'D19_BEKLEIDUNG_REST', 'D19_BILDUNG', 'D19_BIO_OEKO', 'D19_BUCH_CD', 'D19_DIGIT_SERV', 'D19_DROGERIEARTIKEL', 'D19_E
NERGIE', 'D19_FREIZEIT', 'D19_GARTEN', 'D19_GESAMT_ANZ_12', 'D19_GESAMT_ANZ_24', 'D19_HANDWERK', 'D19_HAUS_DEKO', 'D19_KINDERARTIKEL', 'D1
9_KONSUMTYP_MAX', 'D19_KOSMETIK', 'D19_LEBENSMITTEL', 'D19_LETZTER_KAUF_BRANCHE', 'D19_LOTTO', 'D19_NAHRUNGSERGAENZUNG', 'D19_RATGEBER',
'D19_REISEN', 'D19_SAMMELARTIKEL', 'D19_SCHUHE', 'D19_SONSTIGE', 'D19_SOZIALES', 'D19_TECHNIK', 'D19_TELKO_ANZ_12', 'D19_TELKO_ANZ_24', 'D
19_TELKO_MOBILE', 'D19_TELKO_ONLINE_QUOTE_12', 'D19_TELKO_REST', 'D19_TIERARTIKEL', 'D19_VERSAND_ANZ_12', 'D19_VERSAND_ANZ_24', 'D19_VERSA
ND_REST', 'D19_VERSI_ANZ_12', 'D19_VERSI_ANZ_24', 'D19_VERSI_ONLINE_QUOTE_12', 'D19_VERSICHERUNGEN', 'D19_VOLLSORTIMENT', 'D19_WEIN_FEINKO
ST', 'DSL_FLAG', 'EINGEFUEGT_AM', 'EINGEZOGENAM_HH_JAHR', 'EXTSEL992', 'FIRMENDICHTE', 'GEMEINDETYP', 'HH_DELTA_FLAG', 'KBA13_ANTG1', 'KBA
13_ANTG2', 'KBA13_ANTG3', 'KBA13_ANTG4', 'KBA13_BAUMAX', 'KBA13_CCM_1401_2500', 'KBA13_CCM_3000', 'KBA13_CCM_3001', 'KBA13_GBZ', 'KBA13_HH
Z', 'KBA13_KMH_210', 'KK_KUNDENTYP', 'KOMBIALTER', 'KONSUMZELLE', 'MOBI_RASTER', 'RT_KEIN_ANREIZ', 'RT_SCHNAEPPCHEN', 'RT_UEBERGROESSE',
'SOHO_KZ', 'STRUKTURTYP', 'UMFELD_ALT', 'UMFELD_JUNG', 'UNGLEICHENN_FLAG', 'VERDICHTUNGSRAUM', 'VHA', 'VHN', 'VK_DHT4A', 'VK_DISTANZ', 'VK
_ZG11']
```

As we don't have information about these features we will exclude them from further analysis, as it would be very difficult to extract insights since we don't know what they mean.

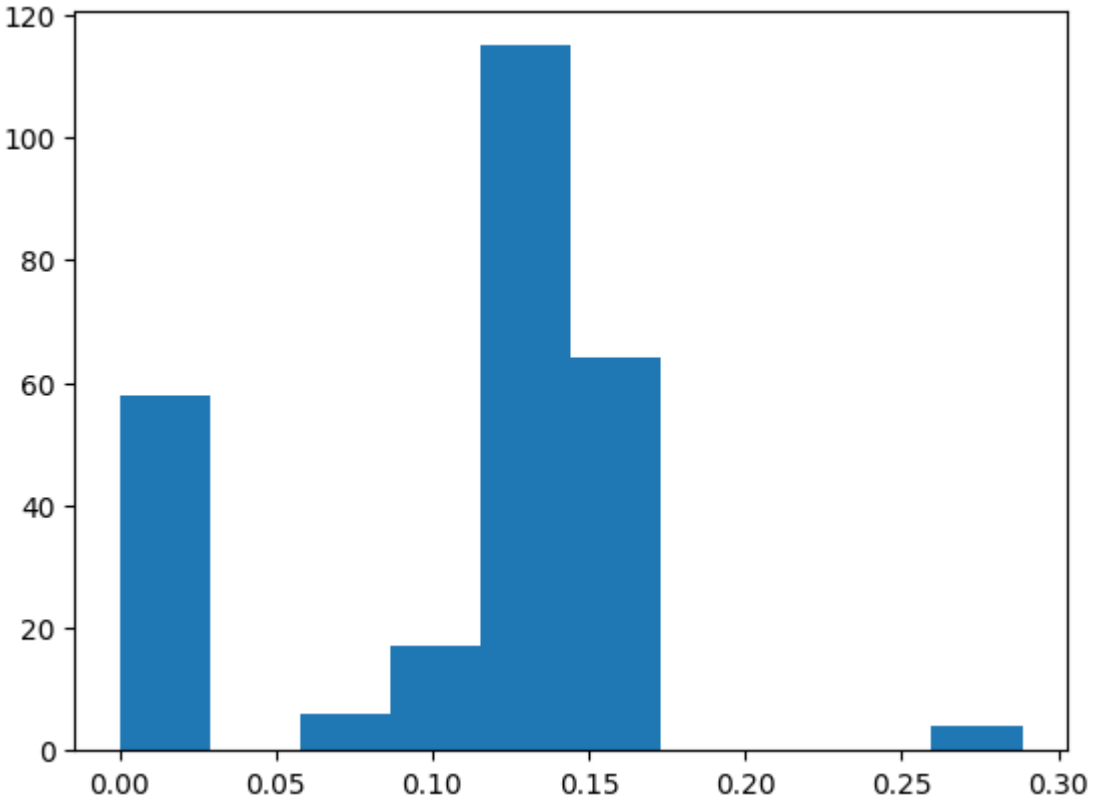
In [12]:
```
azdias = azdias.drop(columns = y)
```

2.1.1. Missing Values

Understanding the distribution of null values and handling them is extremely important in this project. We will start by analyzing the null values of azdia in order to find patterns and understand what can be excluded from the base and what should undergo other types of interventions.

By Collumns

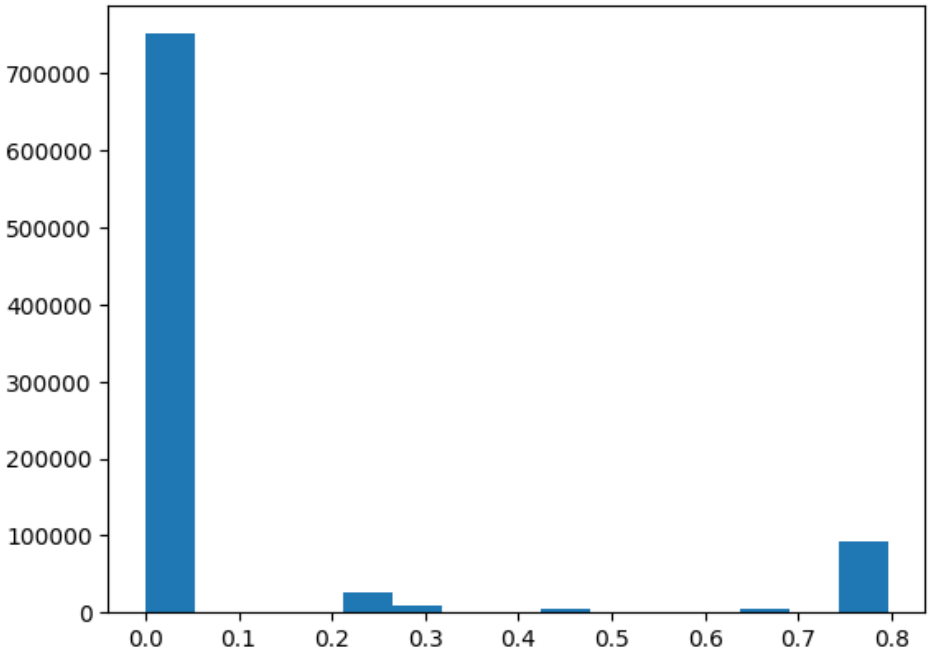
We start by analyzing the percentage of missing values within the columns.



We noticed that most of the distribution was between 0.10 and 0.15, so a threshold of 0.2 was considered to keep the data in the dataset.

By Rows

For lines we apply the same logic, we add the amount of missing values inside the lines and plot a histogram.



We can notice that in this case the big concentration was at 0.0, so we could exclude everything after 0.1. At this time we exclude 15% of the base.

2.1.2. Categorical Columns

For the categorical columns we had three different approaches:

1. Delete the fields for which we do not have information in the auxiliary tables.
2. Adjust fields that have small errors in the data
3. Create dummy variables

```
In [26]: cate = azdias.select_dtypes(include=['object','bool','category']).columns
         azdias[cate].head()
```

Out[26]:

	CAMEO_DEU_2015	CAMEO_DEUG_2015	OST_WEST_KZ
1	8A	8.0	W
2	4C	4.0	W
3	2A	2.0	W
4	6B	6.0	W
5	8C	8.0	W

2.1.3. Nan to -1

Finally, we fill the remaining Nan values with -1 since this is the unknow value provided in the support table.

```
azdias = azdias.fillna(value= -1)
```

Now that we have finished cleaning the azdias dataframe, we can create a function to be used in the other dataframes of this analysis.

2.1.4. Function

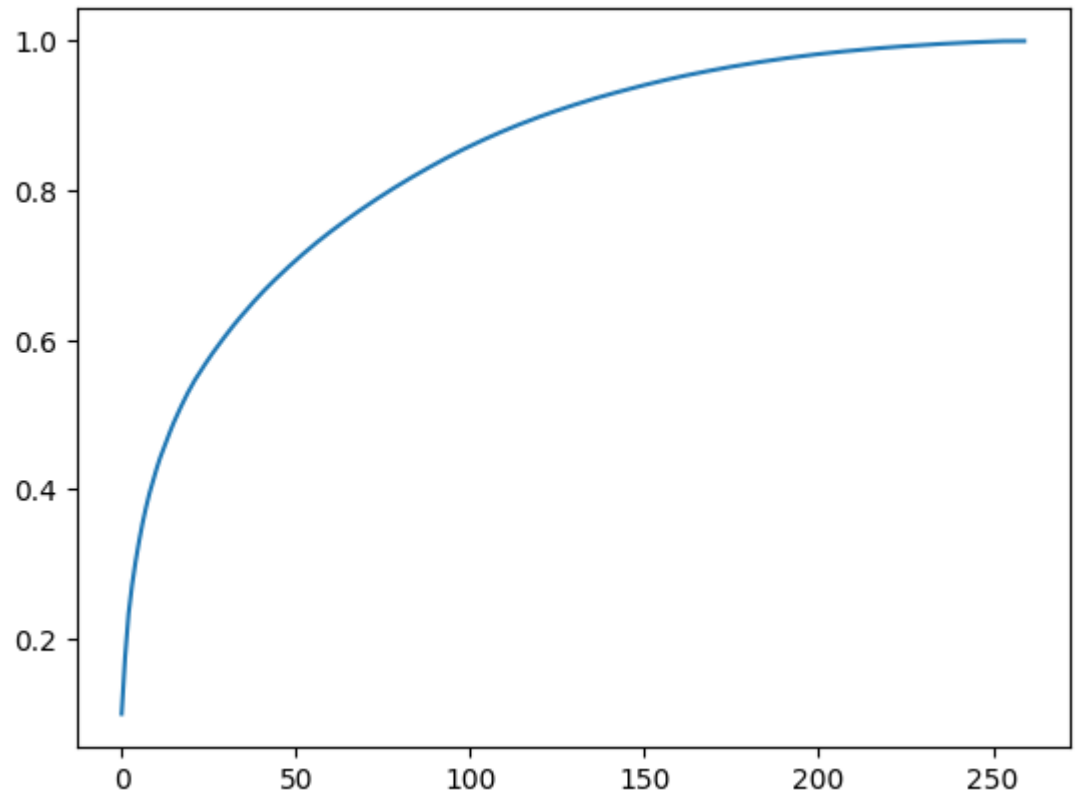
With this, we created a dataframe cleaning function that will be applied to all bases in future analyses. To ensure that all columns are the same in the function, text was used instead of dynamic values.

In [34]:

```
def clean_data(df):
    '''
    This function clears the columns and rows of the dataframes of this analysis.
    input: df
    output: df_clean
    '''
    #Clean columns with no context
    drop_context = ['LNR', 'AKT_DAT_KL', 'ALTER_KIND1', 'ALTER_KIND2', 'ALTER_KIND3', 'ALTER_KIND4', 'ALTERSKATEGORIE_FEIN',
     'ANZ_KINDER', 'ANZ_STATISTISCHE_HAUSHALTE', 'CAMEO_INTL_2015', 'CJT_KATALOGNUTZER', 'CJT_TYP_1', 'CJT_TYP_2', 'CJT_TYP_3',
     'CJT_TYP_4', 'CJT_TYP_5', 'CJT_TYP_6', 'D19_BANKEN_ANZ_12', 'D19_BANKEN_ANZ_24', 'D19_BANKEN_DIREKT', 'D19_BANKEN_GROSS',
     'D19_BANKEN_LOKAL', 'D19_BANKEN_REST', 'D19_BEKLEIDUNG_GEH', 'D19_BEKLEIDUNG_REST', 'D19_BILDUNG', 'D19_BIO_OEKO', 'D19_BUCH_CD',
     'D19_DIGIT_SERV', 'D19_DROGERIEARTIKEL', 'D19_ENERGIE', 'D19_FREIZEIT', 'D19_GARTEN', 'D19_GESAMT_ANZ_12', 'D19_GESAMT_ANZ_24',
     'D19_HANDWERK', 'D19_HAUS_DEKO', 'D19_KINDERARTIKEL', 'D19_KONSUMTYP_MAX', 'D19_KOSMETIK', 'D19_LEBENSMITTEL',
     'D19_LETZTER_KAUF_BRANCHE', 'D19_LOTTO', 'D19_NAHRUNGSERGAENZUNG', 'D19_RATGEBER', 'D19_REISEN', 'D19_SAMMELARTIKEL',
     'D19_SCHUHE', 'D19_SONSTIGE', 'D19_SOZIALES', 'D19_TECHNIK', 'D19_TELKO_ANZ_12', 'D19_TELKO_ANZ_24', 'D19_TELKO_MOBILE',
     'D19_TELKO_ONLINE_QUOTE_12', 'D19_TELKO_REST', 'D19_TIERARTIKEL', 'D19_VERSAND_ANZ_12', 'D19_VERSAND_ANZ_24', 'D19_VERSAND_REST',
     'D19_VERSI_ANZ_12', 'D19_VERSI_ANZ_24', 'D19_VERSI_ONLINE_QUOTE_12', 'D19_VERSICHERUNGEN', 'D19_VOLLSORTIMENT', 'D19_WEIN_FEINKOS
     'DSL_FLAG', 'EINGEFUEGT_AM', 'EINGEZOGENAM_HH_JAHR', 'EXTSEL992', 'FIRMENDICHTE', 'GEMEINDETYP', 'HH_DELTA_FLAG', 'KBA13_ANTG1',
     'KBA13_ANTG2', 'KBA13_ANTG3', 'KBA13_ANTG4', 'KBA13_BAUMAX', 'KBA13_CCM_1401_2500', 'KBA13_CCM_3000', 'KBA13_CCM_3001', 'KBA13_GB
     'KBA13_HHZ', 'KBA13_KMH_210', 'KK_KUNDENTYP', 'KOMBIALTER', 'KONSUMZELLE', 'MOBI_RASTER', 'RT_KEIN_ANREIZ', 'RT_SCHNAEPPCHEN',
     'RT_UEBERGROESSE', 'SOHO_KZ', 'STRUKTURTYP', 'UMFELD_ALT', 'UMFELD_JUNG', 'UNGLEICHENN_FLAG', 'VERDICHTUNGSRAUM', 'VHA', 'VHN',
     'VK_DHT4A', 'VK_DISTANZ', 'VK_ZG11']
    df = df.drop(columns = drop_context)


    #Clean columns
    drop_columns = ['D19_BANKEN_ONLINE_QUOTE_12', 'D19_GESAMT_ONLINE_QUOTE_12', 'D19_KONSUMTYP', 'D19_VERSAND_ONLINE_QUOTE_12']
    df = df.drop(columns = drop_columns)

    #Clean rows
    df_mv_r = df.isnull().sum(axis = 1)/df.shape[1]
    df_row_threshold = 0.1
    df = df[df_mv_r <= df_row_threshold]

    #Categorical collumns
    df = df.drop(columns = ['CAMEO_DEU_2015'])

    df['CAMEO_DEUG_2015'] = df['CAMEO_DEUG_2015'].replace('X', -1)
    df['CAMEO_DEUG_2015'] = df['CAMEO_DEUG_2015'].astype(float)

    to_dummy = 'OST_WEST_KZ'
    dummy = pd.get_dummies(df[to_dummy])
    df.drop(to_dummy, axis=1, inplace=True)
    df = pd.concat([df, dummy], axis=1)

    #Nan to -1
    df = df.fillna(value= -1)

    #Drop KBA columns
    #kba_cols = df.columns[df.columns.str.startswith('KBA')]
    #df = df.drop(columns = kba_cols)

    return df
```

2.2. PCA Processing

With the clean base, the PCA (Main Component Analysis) process was started, this method was used to reduce the number of variables within the base, thus facilitating the application of the k-means grouping method later.
We started the process by understanding what would be the ideal number of components for the base.




```
In [42]: threshold = .90
         i = np.argmax(cumvals > threshold)
         print(i, cumvals[i])

121 0.9005056945986579
```

In this graph, in which we see the accumulated value of the understanding of the base, we observe that about 121 components explain 90% of our base. In this way, we can use the value 0.9 as threshold to create a new PCA with just these values.

2.2.1. PCA Interpreting

As when applying the PCA we lose legibility of the base, it is necessary to analyze the main features of the components. For this, a function was used that prints the three features that most impact the component and the three that have the least impact.

Component 0

Feature Weight: 9.97%

Highest:

KBA05_KRSVAN: share of vans (referred to the county average) 0.169
KBA05_KRSOBER: share of upper class cars (referred to the county average) 0.170
KBA05_SEG6: share of upper class cars (BMW 7er etc.) in the microcell 0.189

Lowest:

KBA05_ANTG3: number of 6-10 family houses in the cell -0.022
PRAEGENDE_JUGENDJAHRE: dominating movement in the person's youth -0.021

FINANZ_ANLEGER: financial typology: investor -0.020

PC0: is an indicator mainly linked to automobiles and inversely proportional to age.

Component 1

Feature Weight: 7.65%

Highest:

PLZ8_BAUMAX : most common building-type within the PLZ8 0.157
PLZ8_ANTG4: number of >10 family houses in the PLZ8 0.158
PLZ8_ANTG3: number of 6-10 family houses in the PLZ8 0.162

Lowest:

MOBI_REGIO: moving patterns -0.169
PLZ8_ANTG1: dnumber of 1-2 family houses in the PLZ8 -0.163
KBA05_ANTG1: number of 1-2 family houses in the cell -0.160

PC1: It is mainly linked to housing.

Component 2

Feature Weight: 5.78%

Highest:

KBA13_KMH_140_210: share of cars with max speed between 140 and 210 km/h within the PLZ8 0.139
KBA13_SEG_KLEINWAGEN: nshare of small and very small cars (Ford Fiesta, Ford Ka etc.) in the PLZ8 0.141
KBA13_SITZE_5: number of cars with 5 seats in the PLZ8 0.164

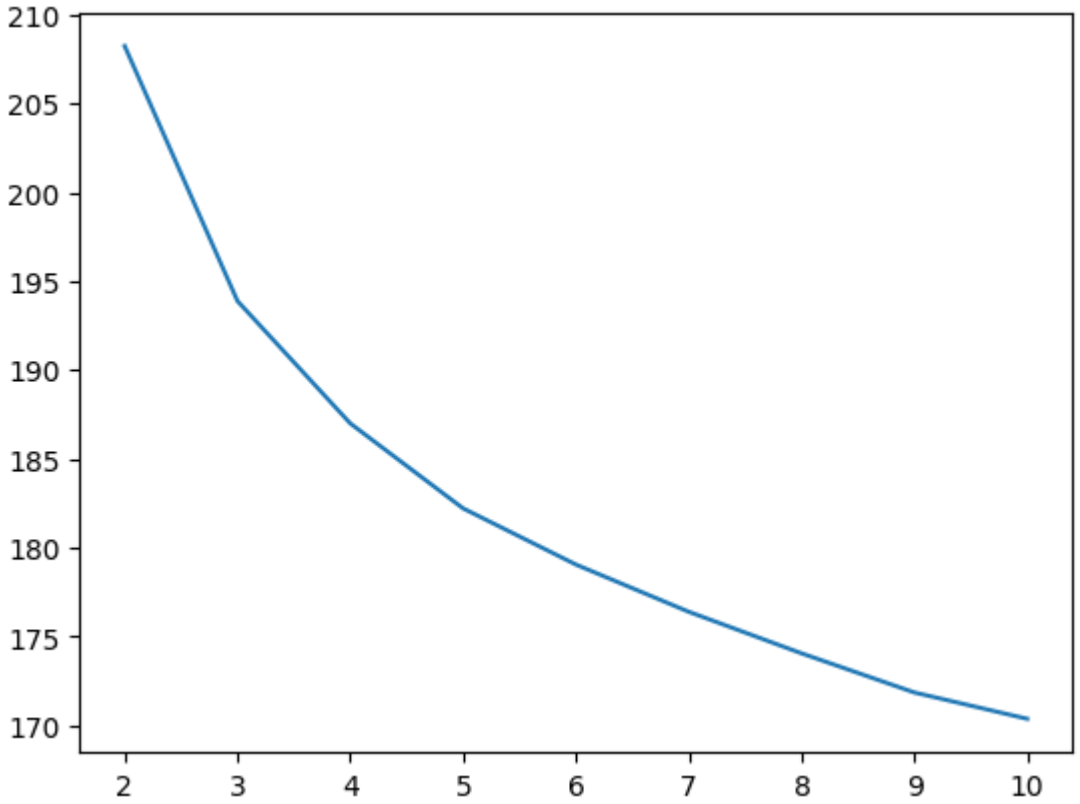
Lowest:

KBA13_HERST_BMW_BENZ: share of BMW & Mercedes Benz within the PLZ8 -0.201
KBA13_SEG_OBEREMITTELKLASSE: share of upper middle class cars and upper class cars (BMW5er, BMW7er etc.) -0.175
KBA13_MERCEDES: share of MERCEDES within the PLZ8 -0.171

PC2: Extremely data-oriented component about cars.

2.3. Clustering

After reducing the dimensions, we can start the grouping process by similar groups. In this work, the k-means algorithm will be used. We started the analysis by plotting a graph with 10 groups and the sum of squared errors.

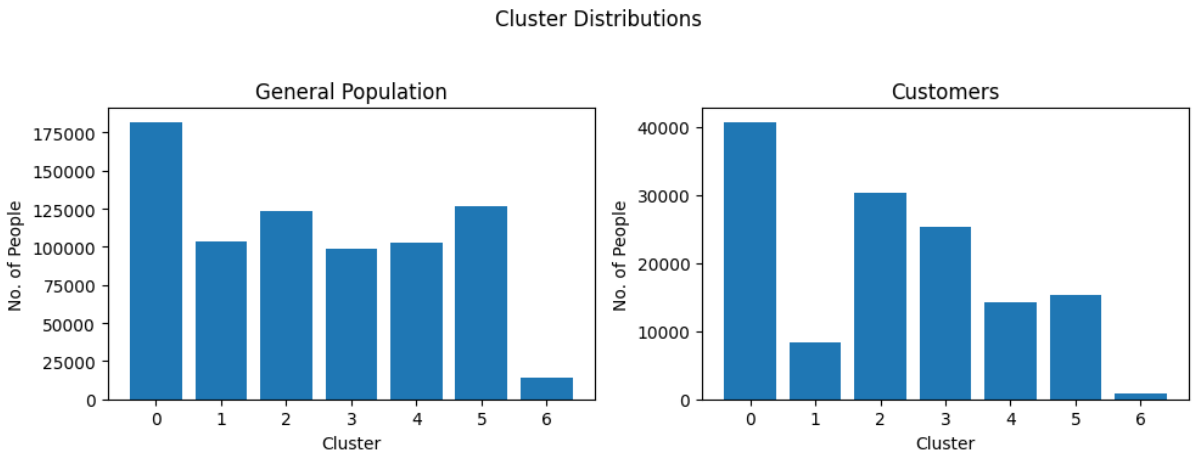


The choice of the number of clusters must be made taking into account the curve of the sum of squared error. For our project we will consider a cluster of 7, since after this value we observe a weaker decay.

After creating the clusters, we need to classify both bases and understand the number of people in each of the clusters.

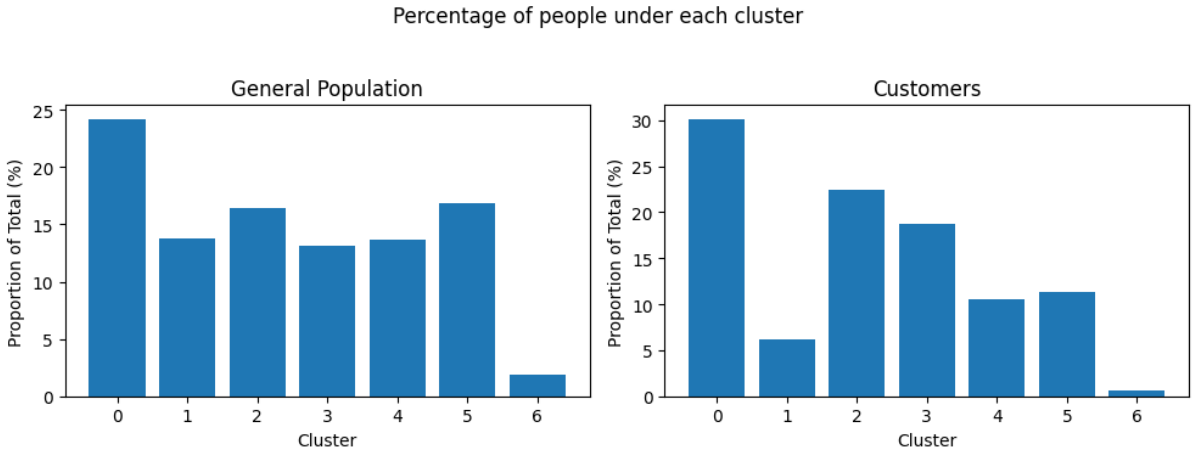
Out[51]:

	Cluster	Population	Customers
0	0	181921	40736
1	1	103717	8335
2	2	123463	30284
3	3	99115	25295
4	4	102695	14306
5	5	126377	15290
6	6	14043	898

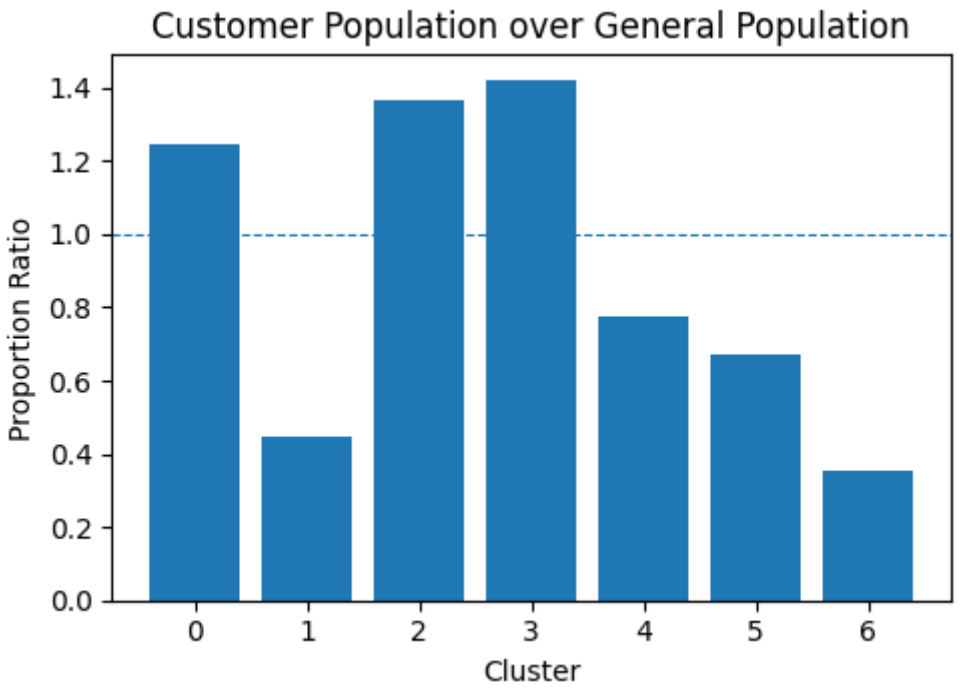


From the graphs, we can see that in both bases, group 4 has the highest participation. At the base of the general population, groups 0, 1, 2, 3 and 5 have a similar distribution. In the customer base, we have a greater share of groups 0 and 5.

Then we analyze the graph in percentage.



Finally, to understand the relationship between the two graphs, we divided the percentage values of consumers by the data of the general population in order to understand in which cases we have an above-average adherence to the brand.



We observe in this way that the best groups for the company are 0, 4 and 5.

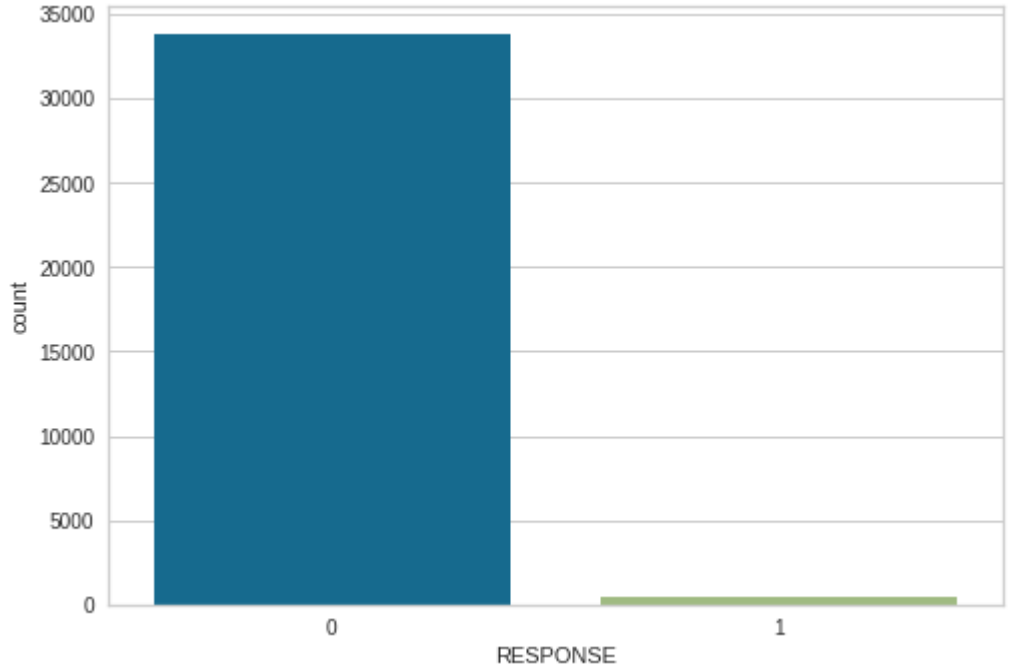
If the company is planning a performance campaign it should focus on these 3 groups. Now if she wants to increase her participation in other groups she could start with an awareness campaign in the other groups.

3.Supervised Learning Model

For the second part of the analysis we need to create a prediction model to calculate the propensity of customers to respond or not to the company's emails. This challenge is extremely strategic, since by reducing the email base we are being more assertive and saving money.

We started by going up the training and test bases and then passing them through the cleaning function created earlier.

We also look at the ratio of customers who respond to those who do not.



At this point we realized that the classes were unbalanced and therefore we will have to take this into account when creating the models.

To build the prediction model we will use the pycaret library. The choice for this library is due to the practicality of performing the setup and the ease of viewing the best models. The library also offers options for tuning the model and visualization tools.
We start by configuring the training base target, the variables that will be used and the method to adjust the class imbalance.

In [28]:
```
target = 'RESPONSE'
variables = list(mail_train.columns.values)
variables.remove(target)
num_vars = variables
adasyn1 = ADASYN(sampling_strategy='minority')
```

In [29]:
```
#Setup do Pycaret
exp_setup = setup(data=mail_train,
                  target=target,
                  numeric_features = num_vars,
                  use_gpu=True,
                  normalize=True,
                  remove_outliers = True,
                  fix_imbalance=True,
                  fix_imbalance_method= adasyn1,
                  remove_multicollinearity = True,
                  multicollinearity_threshold = 0.8,
                  session_id=42)
```

Then we use compare_models to test 16 different models so we can decide which one makes the most sense for our reality.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.9875	0.5782	0.0000	0.0000	0.0000	0.0000	0.0000	4.598
lightgbm	Light Gradient Boosting Machine	0.9875	0.5693	0.0000	0.0000	0.0000	0.0000	0.0000	10.184
catboost	CatBoost Classifier	0.9875	0.5606	0.0000	0.0000	0.0000	0.0000	0.0000	77.110
lda	Linear Discriminant Analysis	0.7111	0.5597	0.3381	0.0148	0.0284	0.0046	0.0133	4.738
lr	Logistic Regression	0.7357	0.5587	0.3274	0.0158	0.0301	0.0064	0.0174	5.444
gbc	Gradient Boosting Classifier	0.9875	0.5521	0.0000	0.0000	0.0000	0.0000	0.0000	107.520
et	Extra Trees Classifier	0.9875	0.5352	0.0000	0.0000	0.0000	0.0000	0.0000	10.600
rf	Random Forest Classifier	0.9875	0.5317	0.0000	0.0000	0.0000	0.0000	0.0000	30.520
nb	Naive Bayes	0.4672	0.5237	0.5670	0.0132	0.0259	0.0015	0.0074	3.108
ada	Ada Boost Classifier	0.9687	0.5185	0.0316	0.0208	0.0246	0.0098	0.0101	23.242
knn	K Neighbors Classifier	0.2261	0.5095	0.7925	0.0127	0.0249	0.0004	0.0030	137.148
qda	Quadratic Discriminant Analysis	0.9875	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	4.136
dummy	Dummy Classifier	0.4025	0.5000	0.6000	0.0075	0.0148	0.0000	0.0000	2.944
dt	Decision Tree Classifier	0.9633	0.4964	0.0176	0.0098	0.0125	-0.0041	-0.0045	12.358
svm	SVM - Linear Kernel	0.7017	0.0000	0.3697	0.0156	0.0300	0.0062	0.0184	6.096
ridge	Ridge Classifier	0.7111	0.0000	0.3381	0.0148	0.0284	0.0046	0.0133	3.158

To decide which models we will choose we need to think about the application. Our main objective is to send emails to users who are likely to respond to the communication. So, in addition to looking at AUC, we also need to consider Recall as an important metric. Among the available options, the Linear Discriminant Analysis algorithm showed the best performance, so it was chosen to proceed with the analysis.

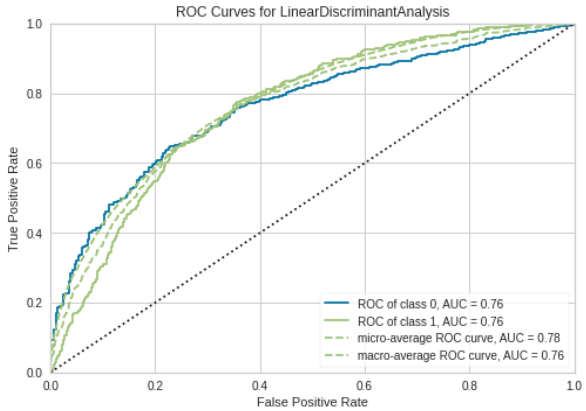
With that we create a model with Linear Discriminant Analysis.After creation we run the model through two rounds of tuning. The first with a fold of 5 and the second with a fold of 10.

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7021	0.6495	0.4483	0.0193	0.0369	0.0128	0.0377
1	0.6976	0.5267	0.3929	0.0161	0.0310	0.0075	0.0227
2	0.6927	0.5852	0.4286	0.0173	0.0332	0.0098	0.0298
3	0.7077	0.5949	0.3571	0.0152	0.0292	0.0057	0.0168
4	0.6993	0.6523	0.5714	0.0233	0.0447	0.0215	0.0654
5	0.7029	0.5067	0.2500	0.0106	0.0203	-0.0034	-0.0101
6	0.7165	0.5296	0.2857	0.0126	0.0242	0.0006	0.0019
7	0.6800	0.5769	0.4483	0.0179	0.0345	0.0102	0.0316
8	0.7125	0.5595	0.3103	0.0140	0.0268	0.0025	0.0070
9	0.7059	0.5118	0.2414	0.0107	0.0205	-0.0040	-0.0116
Mean	0.7017	0.5693	0.3734	0.0157	0.0301	0.0063	0.0191
Std	0.0098	0.0500	0.0992	0.0038	0.0072	0.0074	0.0224

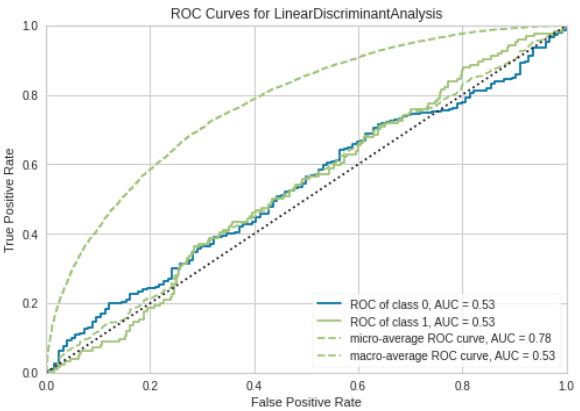
To help evaluate the model, we print some analyses.

Starting with the ROC curve that shows the relationship between True Positive Rate and False Positive. By default, we used a threshold of 0.5 in the model.

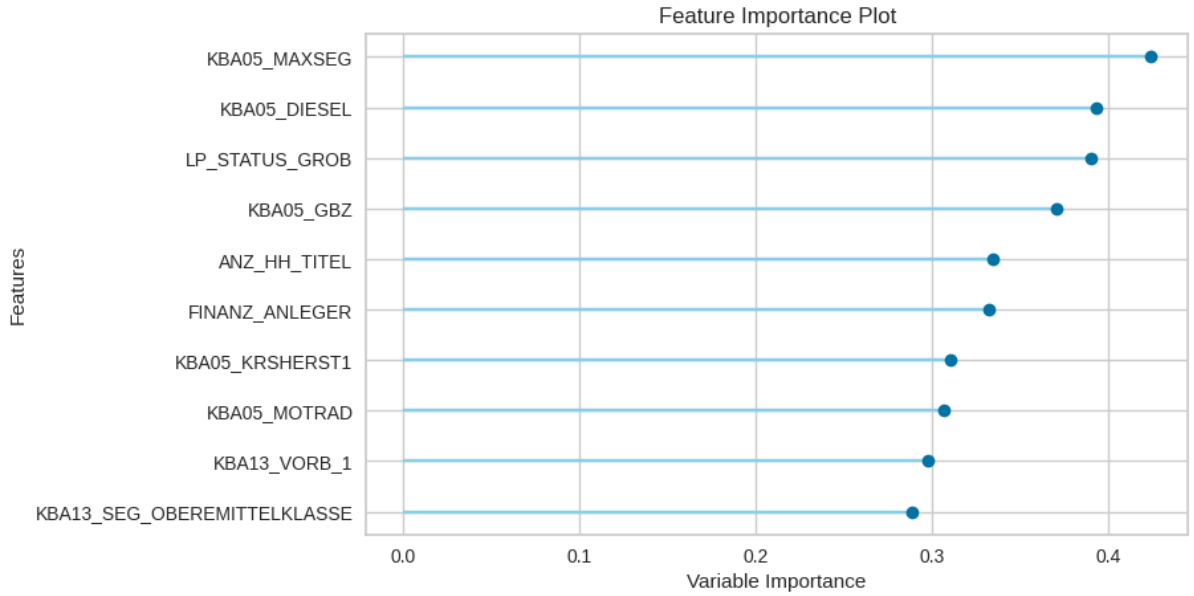
Train Data = True



Train Data = False

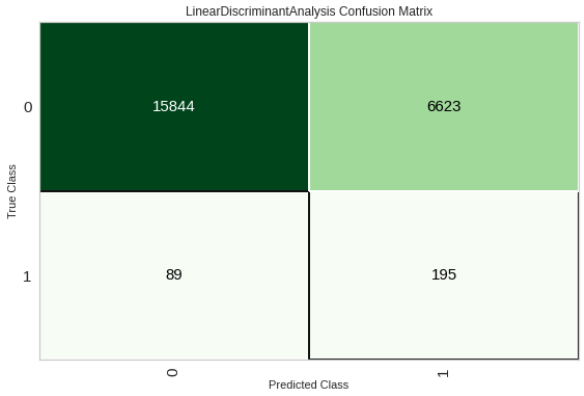


We also analyze the features that most impact the model. We observed a lot of car-related featues (KBA). But we look at LP_STATUS_GROB in the third place which is related to social status

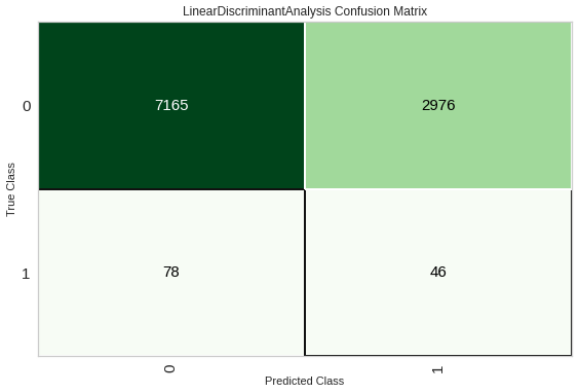


In the confusion matrices, we observed that the results are acceptable, since True Positives are more than twice as many as False Positives and most of them are being classified as True Negatives.

Train Data = True



Train Data = False



Finally, we print the model and save it to be able to use it in the prediction.

In [42]:
```
predict_model(PredModel_tune)
```

```
INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=0.05,
                           solver='lsqr', store_covariance=False, tol=0.0001), probability_threshold=None, encoded_labels=False, drift_rep
ort=False, raw_score=False, round=4, verbose=True, ml_usecase=MLUsecase.CLASSIFICATION, display=None, drift_kwargs=None)
INFO:logs:Checking exceptions
INFO:logs:Preloading libraries
INFO:logs:Preparing display monitor
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Linear Discriminant Analysis	0.7025	0.5315	0.371	0.0152	0.0292	0.0062	0.0186

Out[42]:

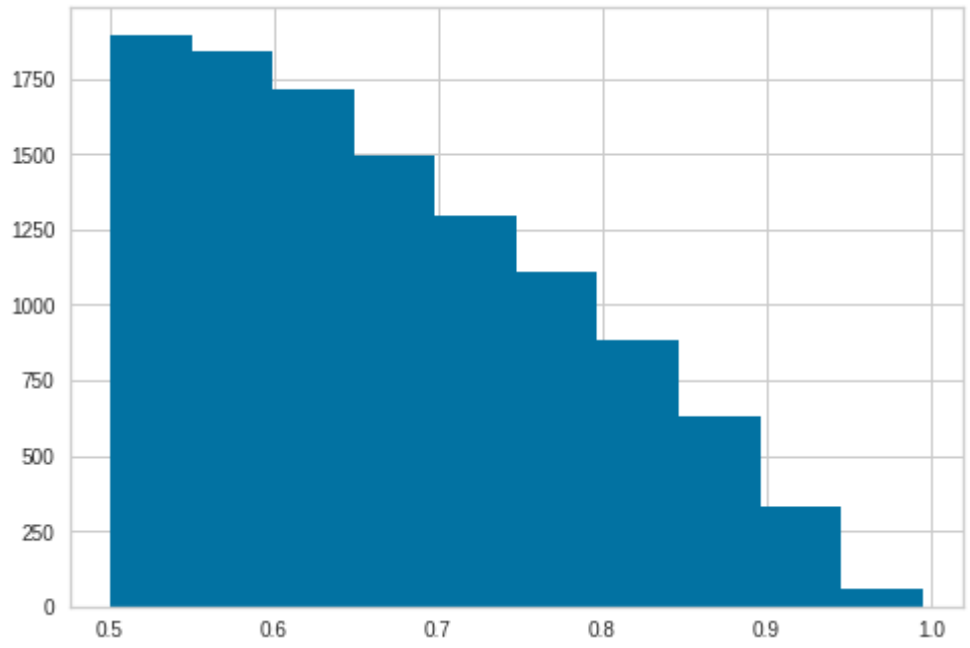
	AGER_TYP	ALTER_HH	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_PERSONEN	ANZ_TITEL	ARBEIT	BALLRAUM	CAMEO_DEUG_2015	CJT_GESAMTTYP
0	0.866400	-0.374200	0.862181	-0.092721	-0.755264	-0.090770	0.908380	-1.470913	1.657533	0.116527
1	-1.362962	-1.688715	-0.304921	-0.092721	-0.022192	-0.090770	-0.026305	1.255803	-0.392618	-0.512012
2	1.609521	-0.538514	-0.304921	-0.092721	-0.755264	-0.090770	-0.026305	0.801350	-0.392618	-0.512012
3	0.123279	-0.209885	-0.240082	-0.092721	-0.022192	-0.090770	-0.026305	-0.107555	-0.392618	-0.512012
4	0.866400	-1.688715	-0.369760	-0.092721	-0.022192	-0.090770	-0.026305	-0.562008	-0.392618	2.002146
...	...	...	...	...	...	...	...	...	...	...
10260	0.123279	0.118744	-0.369760	-0.092721	-0.022192	-0.090770	0.908380	1.255803	-0.802648	-0.512012
10261	0.123279	0.447372	-0.369760	-0.092721	-0.022192	-0.090770	-0.026305	0.801350	-0.392618	-0.512012
10262	-1.362962	0.283058	0.473147	2.532081	-0.022192	20.225666	0.908380	-1.470913	-1.622709	0.745067
10263	-1.362962	1.433259	-0.240082	-0.092721	0.710880	-0.090770	-0.026305	-0.107555	0.427442	0.745067
10264	-1.362962	0.940315	1.121537	-0.092721	0.710880	-0.090770	0.908380	-1.470913	0.017412	0.116527

10265 rows × 227 columns

3.1. Predictions

With the model saved, we perform the test base prediction. By our prediction we arrived at the value of 11,265 people likely to respond, corresponding to 32.9% of the base. In other words, with the algorithm we were able to reduce the size of the base by two thirds, saving and optimizing the campaign budget.

Next, we have the distribution of the propensity score. We observed that most of the predictions have scores close to 0.5 and this value drops when approaching 1.



4.References

Pycaret Official: https://pycaret.gitbook.io/docs/

Arvato Bertelsmann Customer Analysis: https://medium.com/@mei.eisenbach/arvato-bertelsmann-customer-analysis-ae1aac59a1ef

Bertelsmann-Arvato-customer-segmentation: https://github.com/pranaymodukuru/Bertelsmann-Arvato-customer-segmentation