Capstone Project
Machine Learning Engineer
Nanodegree

Kadichari Coelho
April 26, 2020

Create a Customer Segmentation Report for Arvato Financial Solutions

Project Overview

This project aims at a real-life data science task that was provided by partners like Bertelsmann Arvato Analytics.
Analyzes of the demographic data of customers of a sales company, where they are located in Germany and comparing with demographic information of the general population, will be carried out.
Unsupervised learning techniques will be used to segment customers, identifying groups of the population that best describe the company's main customer base. In a third set of data, supervised learning techniques will be applied to predict which people are most likely to order by mail who may become your customers.

Problem Statement

This project aims to predict which people are most likely to become a customer of a mail order company in Germany.
For that it is necessary:

1. Pre-processing of data: Analyzing the dataset, it is necessary to convert fields that have no value to NaN. Remove empty rows and columns.

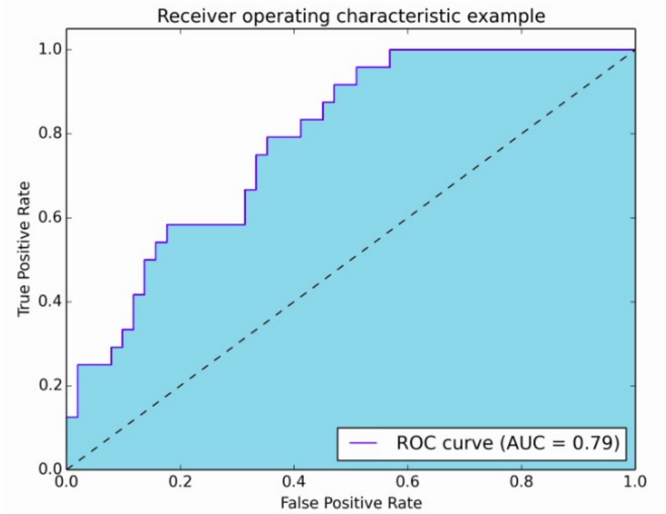
2. Customer Segmentation Report: use unsupervised learning methods to analyze attributes of established customers and the general population in order to create customer segments.

3. Supervised Learning Model: You will have access to a third set of data with attributes of the destinations of a direct mail campaign. Where you will use the previous analysis to create a machine learning model that predicts whether each individual will respond to the campaign or not.

4. Kaggle competition: After choosing a model, it will be used to make predictions in the campaign data as part of a Kaggle competition. Sorting individuals by the likelihood of becoming customers and you will see how their modeling skills compare to others.

Metrics

Because it is a dataset in which we have a great imbalance of classes, 33760 for class 0 and 424 for class 1, and for this reason, model training will be affected and affect the choice of metrics.
The ROC AUC metric is used in this project because it indicates how much the model is capable of differentiating classes. This metric is based on the Confusion Matrix in which the highest area under the curve (AUC) is best on a graph, with 1 being the maximum value. The curve is defined by plotting the false positive rate in relation to the true positive rate.
In the image below we have an example.



Analysis

1. **Data exploration and Visuzalization**

- **Azdias Data**

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_/
0	910215	-1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	910220	-1	9.0	0.0	NaN	NaN	NaN	NaN	21.0	
2	910225	-1	9.0	17.0	NaN	NaN	NaN	NaN	17.0	
3	910226	2	1.0	13.0	NaN	NaN	NaN	NaN	13.0	
4	910241	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	

5 rows × 366 columns

```
azdias.describe()
```

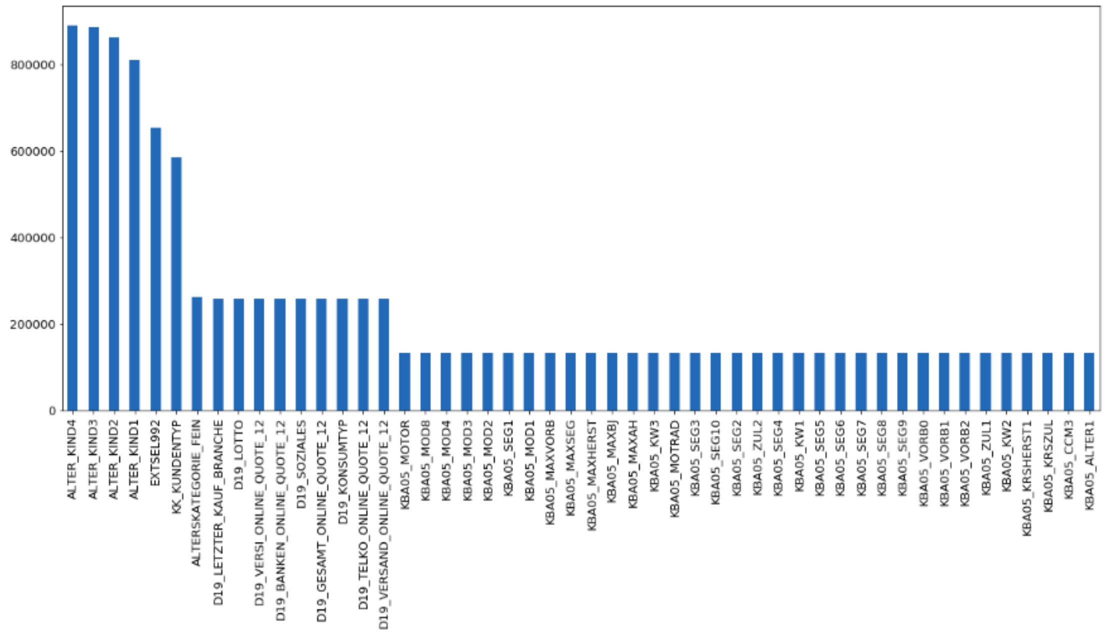
	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	AI
count	8.912210e+05	891221.000000	817722.000000	817722.000000	81058.000000	29499.000000	6170.000000	1205.000000	628274.000000	
mean	6.372630e+05	-0.358435	4.421928	10.864126	11.745392	13.402658	14.476013	15.089627	13.700717	
std	2.572735e+05	1.198724	3.638805	7.639683	4.097660	3.243300	2.712427	2.452932	5.079849	
min	1.916530e+05	-1.000000	1.000000	0.000000	2.000000	2.000000	4.000000	7.000000	0.000000	
25%	4.144580e+05	-1.000000	1.000000	0.000000	8.000000	11.000000	13.000000	14.000000	11.000000	
50%	6.372630e+05	-1.000000	3.000000	13.000000	12.000000	14.000000	15.000000	15.000000	14.000000	
75%	8.600680e+05	-1.000000	9.000000	17.000000	15.000000	16.000000	17.000000	17.000000	17.000000	
max	1.082873e+06	3.000000	9.000000	21.000000	18.000000	18.000000	18.000000	18.000000	25.000000	

8 rows × 360 columns

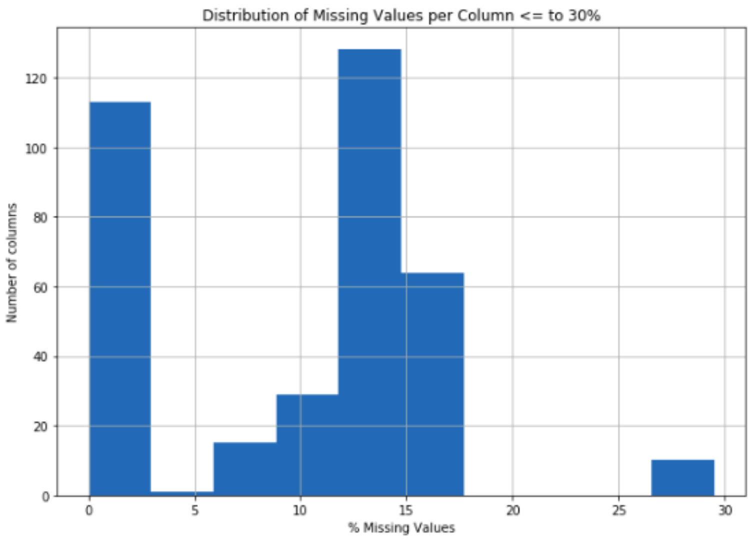
```
azdias.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891221 entries, 0 to 891220
Columns: 366 entries, LNR to ALTERSKATEGORIE_GROB
dtypes: float64(267), int64(93), object(6)
memory usage: 2.4+ GB
```

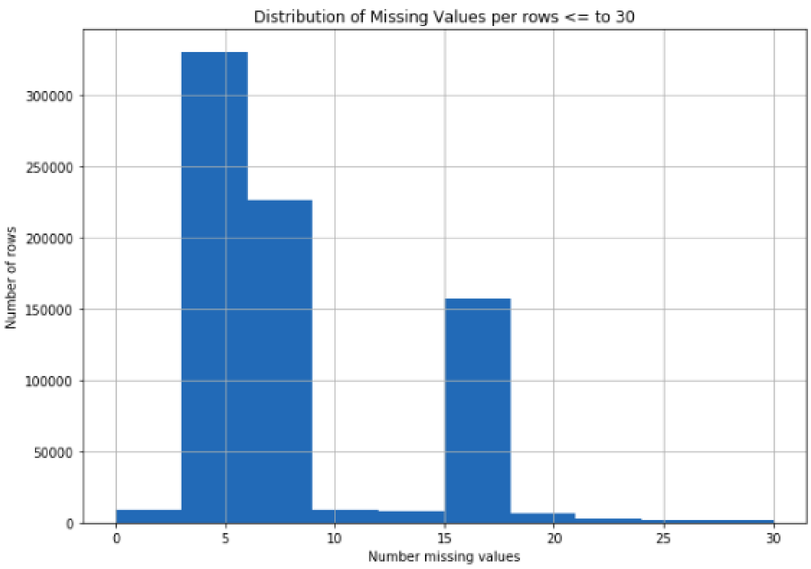
In the image below we can see the top 50 resources with data naturally missing from the Azdias data set.



We can see that columns with more than 30% of missing values are outliers in the histogram below. Here, we investigate these columns and decide whether it is possible to remove them from the data frame.



We can analyze that the majority of 84% of the rows in the data frame have between 0 and 30 missing values, therefore, we remove the lines that have more than 30 missing values.



- **Customers Data**

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_A
0	9626	2	1.0	10.0	NaN	NaN	NaN	NaN	10.0	
1	9628	-1	9.0	11.0	NaN	NaN	NaN	NaN	NaN	
2	143872	-1	1.0	6.0	NaN	NaN	NaN	NaN	0.0	
3	143873	1	1.0	8.0	NaN	NaN	NaN	NaN	8.0	
4	143874	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	

5 rows × 369 columns

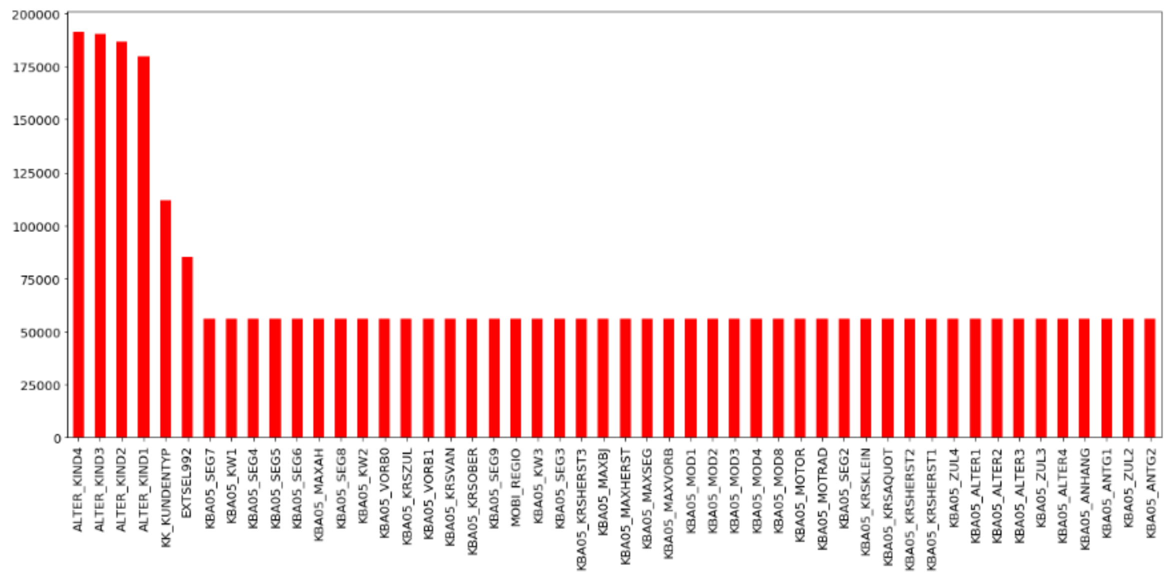
	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN
count	191652.000000	191652.000000	145056.000000	145056.000000	11766.000000	5100.000000	1275.000000	236.000000	139810.000000
mean	95826.500000	0.344359	1.747525	11.352009	12.337243	13.672353	14.647059	15.377119	10.331579
std	55325.311233	1.391672	1.966334	6.275026	4.006050	3.243335	2.753787	2.307653	4.134828
min	1.000000	-1.000000	1.000000	0.000000	2.000000	2.000000	5.000000	8.000000	0.000000
25%	47913.750000	-1.000000	1.000000	8.000000	9.000000	11.000000	13.000000	14.000000	9.000000
50%	95826.500000	0.000000	1.000000	11.000000	13.000000	14.000000	15.000000	16.000000	10.000000
75%	143739.250000	2.000000	1.000000	16.000000	16.000000	16.000000	17.000000	17.000000	13.000000
max	191652.000000	3.000000	9.000000	21.000000	18.000000	18.000000	18.000000	18.000000	25.000000

8 rows × 361 columns

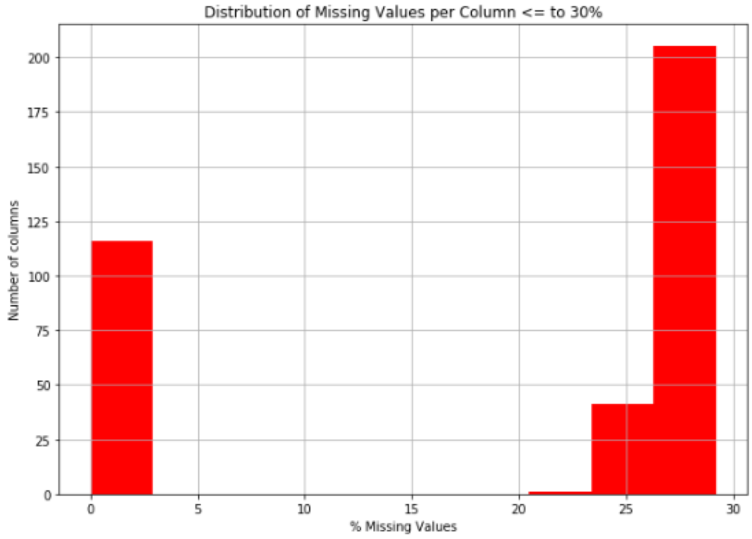
```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 191652 entries, 0 to 191651
Columns: 369 entries, LNR to ALTERSKATEGORIE_GROB
dtypes: float64(267), int64(94), object(8)
memory usage: 539.5+ MB
```

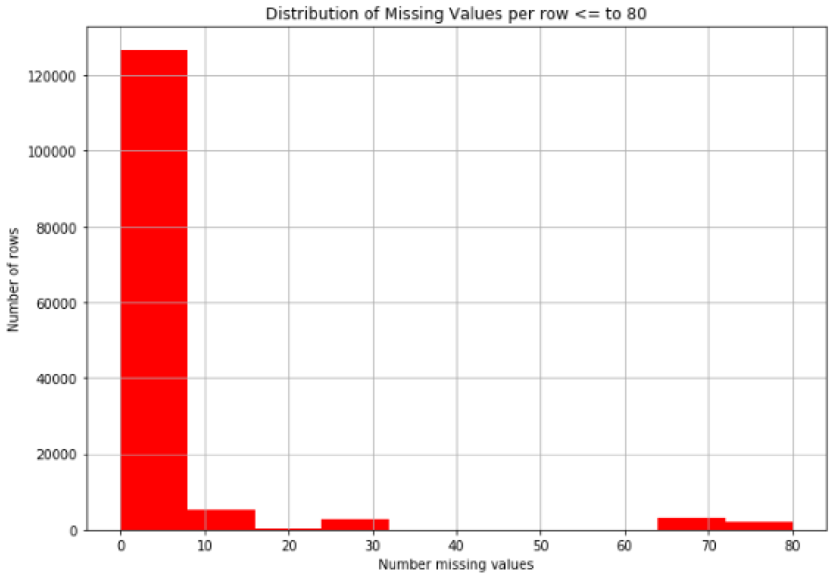
In the image below we can see the top 50 resources with data naturally missing from the Customer data set.



We can see that columns with more than 30% of missing values are outliers in the histogram below. Here, we investigate these columns and decide whether it is possible to remove them from the data frame.



We can see that about 73% of the lines have between 0 and 80 missing values.



2. **Algorithms and Techniques**

We used the Python open source machine learning Scikit-Learn library, which offered us several data pre-processing algorithms, static estimation and metric scoring functions. We use this library as the basis for all stages of the project.

To perform the prediction of the data set, three different algorithms were tested, such as the LogisticRegression, RandomForestClassifier and KneighborsRegressor. Thus, the model that obtains the best result will be implemented in the customized terminal to make prediction by HTTP request.

We will use LogisticRegression as a benchmark and RandomForestClassifier and KneighborsRegressor for comparison.

3. **Benchmark model**

As mentioned above, the reference model will be given by LogisticRegression and then its performance will be compared with RandomForestClassifier and KneighborsRegressor. Then, whichever results the best, we will improve the hyperparameters.

Methodology

1. **Data Pre-processing**

A pre-processing function called clean_data was created to perform data cleaning, removing features, rows and columns that have more than 30% of missing values, removing columns that are not present in the DIAS Attributes - Values 2017 spreadsheet, which case, there were 48 columns. Binary resources, multi-level resources, mixed resources were also re-coded and the missing values have been replaced by -1. The three images below show the function.

Creating the dataset preprocessing function

```
In [3]: def clean_data(df_input):
            """
            Perform feature trimming, re-encoding, and engineering for demographics
            data

            INPUT: Demographics DataFrame
            OUTPUT: Trimmed and cleaned demographics DataFrame
            """
            #Now let's remove the columns that have more than 30% of missing data, which represents only 6
            columns_removed = df_input.columns[df_input.isnull().mean()>0.30]

            #Removing lines that have up to 30 missing values
            rows_removed = df_input.loc[df_input.isnull().sum(axis=1) > 30]

            # remove selected rows
            rows_removed = df_input.drop(rows_removed.index)

            # remove selected columns
            df_input_new = rows_removed.drop(columns_removed, axis=1)

            data_missing_48_col = ['LNR', 'AKT_DAT_KL', 'ALTERSKATEGORIE_FEIN', 'ANZ_KINDER',
               'ANZ_STATISTISCHE_HAUSHALTE', 'CJT_KATALOGNUTZER', 'CJT_TYP_1',
               'CJT_TYP_2', 'CJT_TYP_3', 'CJT_TYP_4', 'CJT_TYP_5', 'CJT_TYP_6',
               'D19_KONSUMTYP_MAX', 'D19_LETZTER_KAUF_BRANCHE', 'D19_SOZIALES',
               'D19_TELKO_ONLINE_QUOTE_12', 'D19_VERSI_ONLINE_QUOTE_12',
               'DSL_FLAG', 'EINGEFUEGT_AM', 'EINGEZOGENAM_HH_JAHR',
               'FIRMENDICHTE', 'GEMEINDETYP', 'HH_DELTA_FLAG', 'KBA13_ANTG1',
               'KBA13_ANTG2', 'KBA13_ANTG3', 'KBA13_ANTG4', 'KBA13_BAUMAX',
               'KBA13_CCM_1401_2500', 'KBA13_GBZ', 'KBA13_HHZ', 'KBA13_KMH_210',
               'KOMBIALTER', 'KONSUMZELLE', 'MOBI_RASTER', 'RT_KEIN_ANREIZ',
               'RT_SCHNAEPPCHEN', 'RT_UEBERGROESSE', 'STRUKTURTYP', 'UMFELD_ALT',
               'UMFELD_JUNG', 'UNGLEICHENN_FLAG', 'VERDICHTUNGSRAUM', 'VHA',
               'VHN', 'VK_DHT4A', 'VK_DISTANZ', 'VK_ZG11']

            #There are 48 resources that are present in the azdias_new dataset,
            #but are not present in data_info, so we will remove
            df_input_new.drop(data_missing_48_col, axis=1, inplace=True)

            #Recoding binary features
            df_input_new['OST_WEST_KZ'] = df_input_new['OST_WEST_KZ'].replace({'O':0.0, 'W':1.0})
```

```python
#Recoding multilevel features
categ_mult = ['AGER_TYP',
              'CJT_GESAMTTYP',
              'FINANZTYP',
              'GFK_URLAUBERTYP',
              'LP_FAMILIE_FEIN',
              'LP_FAMILIE_GROB',
              'LP_STATUS_FEIN',
              'LP_STATUS_GROB',
              'NATIONALITAET_KZ',
              'SHOPPER_TYP',
              'TITEL_KZ',
              'VERS_TYP',
              'ZABEOTYP',
              'D19_KONSUMTYP',
              'D19_GESAMT_ANZ_12',
              'D19_GESAMT_ANZ_24',
              'D19_BANKEN_ANZ_12',
              'D19_BANKEN_ANZ_24',
              'D19_GESAMT_OFFLINE_DATUM',
              'D19_GESAMT_ONLINE_DATUM',
              'D19_GESAMT_DATUM',
              'D19_BANKEN_OFFLINE_DATUM',
              'D19_BANKEN_ONLINE_DATUM',
              'D19_BANKEN_DATUM',
              'D19_TELKO_OFFLINE_DATUM',
              'D19_TELKO_ONLINE_DATUM',
              'D19_TELKO_DATUM',
              'D19_VERSAND_OFFLINE_DATUM',
              'D19_VERSAND_ONLINE_DATUM',
              'D19_VERSAND_DATUM',
              'D19_VERSI_OFFLINE_DATUM',
              'D19_VERSI_ONLINE_DATUM',
              'D19_VERSI_DATUM',
              'D19_GESAMT_ONLINE_QUOTE_12',
              'D19_BANKEN_ONLINE_QUOTE_12',
              'D19_VERSAND_ONLINE_QUOTE_12',
              'GEBAEUDETYP',
              'CAMEO_DEUG_2015',
              'CAMEO_DEU_2015']

df_input_enc = pd.get_dummies(df_input_new, columns=categ_mult)

#Recoding mixed features
decade1 = [1, 3, 5, 7, 9, 11, 13, 15]
decade2 = [2, 4, 6, 8, 10, 12, 14]

df_input_enc['DECADE'] = df_input_enc['PRAEGENDE_JUGENDJAHRE']
```

```python
main = df_input_enc['PRAEGENDE_JUGENDJAHRE'].isin(decade1)
df_input_enc.loc[main, 'MOVEMENT'] = 1.0
avant = df_input_enc['PRAEGENDE_JUGENDJAHRE'].isin(decade2)
df_input_enc.loc[avant, 'MOVEMENT'] = 2.0

df_input_enc['CAMEO_INTL_2015'] = df_input_enc['CAMEO_INTL_2015'].replace(np.nan,-1)
df_input_enc['CAMEO_INTL_2015'] = df_input_enc['CAMEO_INTL_2015'].replace('XX',-1)
df_input_enc['CAMEO_INTL_2015'] = df_input_enc['CAMEO_INTL_2015'].astype(int)

df_input_enc['WEALTH'] = df_input_enc['CAMEO_INTL_2015'] // 10
df_input_enc['LIFE_STAGE'] = df_input_enc['CAMEO_INTL_2015'] % 10

df_input_enc = pd.get_dummies(df_input_enc, columns=['LP_LEBENSPHASE_GROB', 'WOHNLAGE'])

df_input = df_input_enc.drop(['CAMEO_INTL_2015', 'PRAEGENDE_JUGENDJAHRE'], axis=1)

df_input = df_input.replace([np.inf, -np.inf], np.nan)
df_input = df_input.fillna(-1)

# Return the cleaned dataframe.
return df_input
```

This function was applied to all datasets, such as:

- Udacity_AZDIAS_052018.csv
- Udacity_CUSTOMERS_052018.csv
- Udacity_MAILOUT_052018_TRAIN.csv
- Udacity_MAILOUT_052018_TEST.csv

Below are the tables for each dataset before and after going through the pre-processing function:

- Udacity_AZDIAS_052018.csv
  - Before:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_A
0	910215	-1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	910220	-1	9.0	0.0	NaN	NaN	NaN	NaN	21.0	
2	910225	-1	9.0	17.0	NaN	NaN	NaN	NaN	17.0	
3	910226	2	1.0	13.0	NaN	NaN	NaN	NaN	13.0	
4	910241	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	

5 rows × 366 columns

  - After:

	ALTER_HH	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_PERSONEN	ANZ_TITEL	ARBEIT	BALLRAUM	D19_BANKEN_DIREKT	D19_BANKEN_GROSS	D
1	0.0	11.0	0.0	2.0	0.0	3.0	6.0	0	0	
2	17.0	10.0	0.0	1.0	0.0	3.0	2.0	0	0	
3	13.0	1.0	0.0	0.0	0.0	2.0	4.0	0	0	
4	20.0	3.0	0.0	4.0	0.0	4.0	2.0	1	2	
5	10.0	5.0	0.0	1.0	0.0	2.0	6.0	0	0	

5 rows × 669 columns

- Udacity_CUSTOMERS_052018.csv
  - Before:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_A
0	9626	2	1.0	10.0	NaN	NaN	NaN	NaN	10.0	
1	9628	-1	9.0	11.0	NaN	NaN	NaN	NaN	NaN	
2	143872	-1	1.0	6.0	NaN	NaN	NaN	NaN	0.0	
3	143873	1	1.0	8.0	NaN	NaN	NaN	NaN	8.0	
4	143874	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	

5 rows × 369 columns

  - After:

	ALTER_HH	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_PERSONEN	ANZ_TITEL	ARBEIT	BALLRAUM	D19_BANKEN_DIREKT	D19_BANKEN_GROSS	D
0	10.0	1.0	0.0	2.0	0.0	1.0	3.0	0	0	
2	6.0	1.0	0.0	1.0	0.0	3.0	7.0	0	0	
3	8.0	0.0	-1.0	0.0	0.0	1.0	7.0	0	0	
4	20.0	7.0	0.0	4.0	0.0	3.0	3.0	5	0	
5	11.0	1.0	0.0	2.0	0.0	3.0	7.0	0	0	

5 rows × 669 columns

- Udacity_MAILOUT_052018_TRAIN.csv
  - Before:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AK
0	1763	2	1.0	8.0	NaN	NaN	NaN	NaN	8.0	1
1	1771	1	4.0	13.0	NaN	NaN	NaN	NaN	13.0	
2	1776	1	1.0	9.0	NaN	NaN	NaN	NaN	7.0	
3	1460	2	1.0	6.0	NaN	NaN	NaN	NaN	6.0	
4	1783	2	1.0	9.0	NaN	NaN	NaN	NaN	9.0	5

5 rows × 367 columns

- After:

	ALTER_HH	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_PERSONEN	ANZ_TITEL	ARBEIT	BALLRAUM	D19_BANKEN_DIREKT	D19_BANKEN_GROSS	D
0	8.0	15.0	0.0	1.0	0.0	3.0	5.0	0	0	
1	13.0	1.0	0.0	2.0	0.0	2.0	5.0	0	0	
2	9.0	0.0	-1.0	0.0	0.0	4.0	1.0	0	0	
3	6.0	4.0	0.0	2.0	0.0	4.0	2.0	0	0	
4	9.0	53.0	0.0	1.0	0.0	3.0	4.0	0	0	

5 rows × 669 columns

- Udacity_MAILOUT_052018_TEST.csv
  - Before:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AK
0	1754	2	1.0	7.0	NaN	NaN	NaN	NaN	6.0	
1	1770	-1	1.0	0.0	NaN	NaN	NaN	NaN	0.0	2
2	1465	2	9.0	16.0	NaN	NaN	NaN	NaN	11.0	
3	1470	-1	7.0	0.0	NaN	NaN	NaN	NaN	0.0	
4	1478	1	1.0	21.0	NaN	NaN	NaN	NaN	13.0	

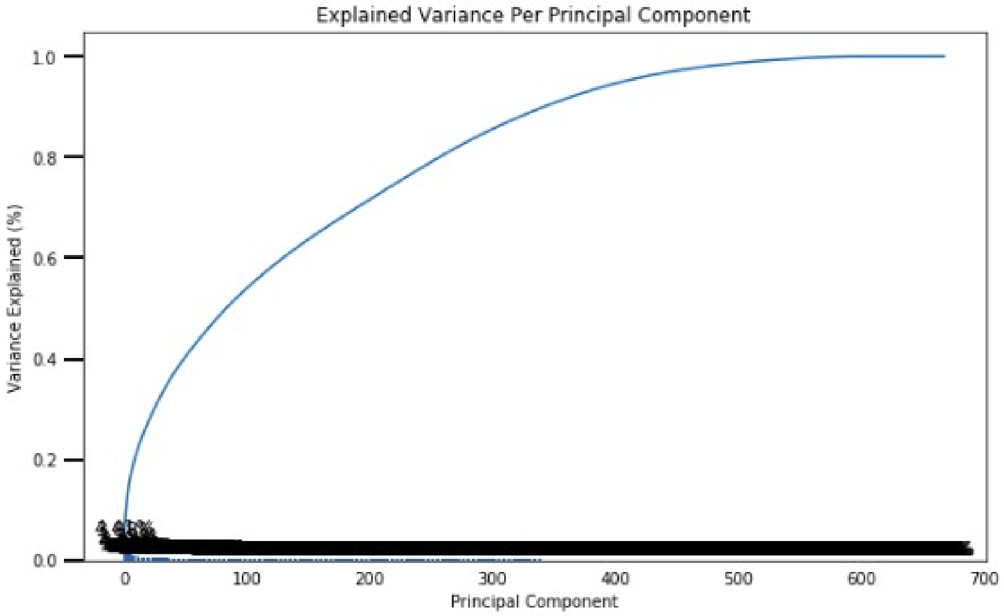
5 rows × 366 columns

  - After:

	ALTER_HH	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_PERSONEN	ANZ_TITEL	ARBEIT	BALLRAUM	D19_BANKEN_DIREKT	D19_BANKEN_GROSS	D
0	7.0	2.0	0.0	2.0	0.0	3.0	6.0	0	0	
1	0.0	20.0	0.0	1.0	0.0	4.0	7.0	0	0	
2	16.0	2.0	0.0	4.0	0.0	4.0	1.0	0	0	
3	0.0	1.0	0.0	0.0	0.0	4.0	1.0	0	0	
4	21.0	1.0	0.0	4.0	0.0	3.0	6.0	2	2	

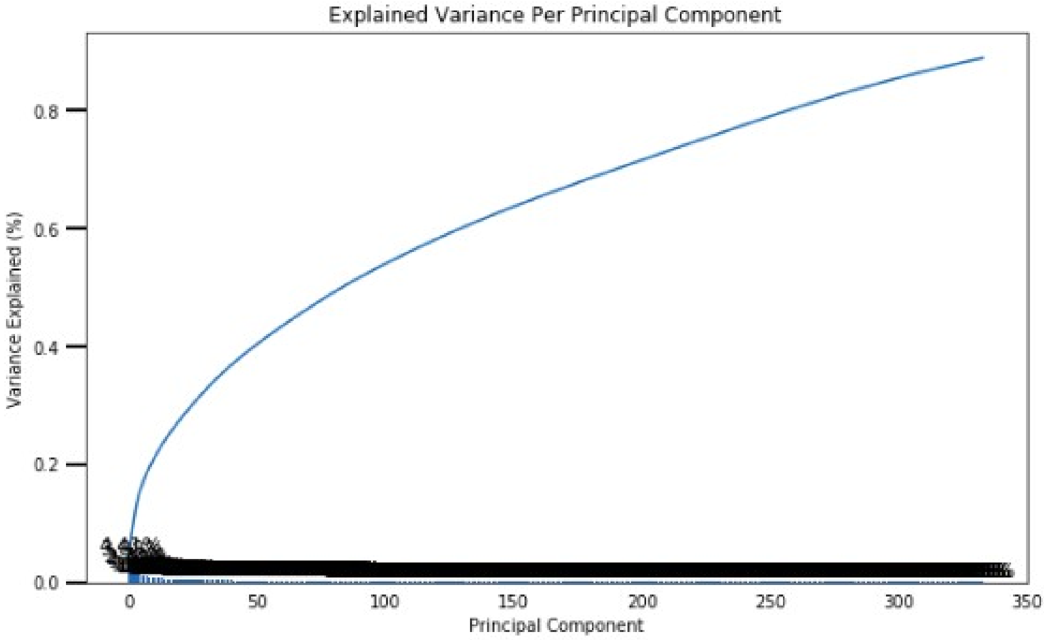
5 rows × 669 columns

After the pre-processing of the data, the resource scale was applied so that the main component vectors were not influenced by the differences in the resource scale, as we will then apply the dimensionality reduction technique that is the PCA. For this we use StandardScaler.
The next step was to perform the dimensionality reduction, as previously mentioned, we used sklearn's PCA class to apply principal component analysis to the data, so the maximum variation vectors were located in the data.
The first strategy used was not to adopt any parameters, so we obtained an accumulated rate of variation of vectors of 99%, with this in order to reduce the dimensionality, we do not have a clear way to choose the number of components, so we follow the recommendation to use half the number of resources for the analysis of the main components of the data, with 334 components, obtaining an accumulated rate of variation of 88%. Below are two graphs showing the variation ratio explained by each major component, as well as the cumulative explained variation of 669 and 334, respectively.




```
sum(pca.explained_variance_ratio_)
```

```
0.9999999999999996
```




```
sum(pca_1.explained_variance_ratio_)
```

```
0.8884260550972088
```

We interpret the main components:

- The first component increases with the high number of total transaction activities in the last 24 months and decreases with online affinity.

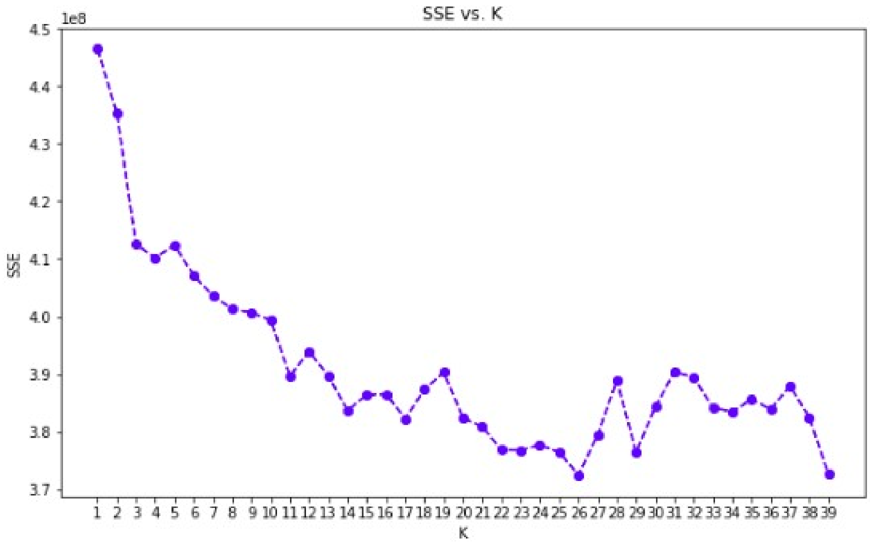
- The second component has a high number of car owners, such as people who own high-class cars. And the component of households' estimated net income decreases.

- The third component is related to financing, describing a person who saves money and transacts on the Internet, and the number decreases with little financial interest.

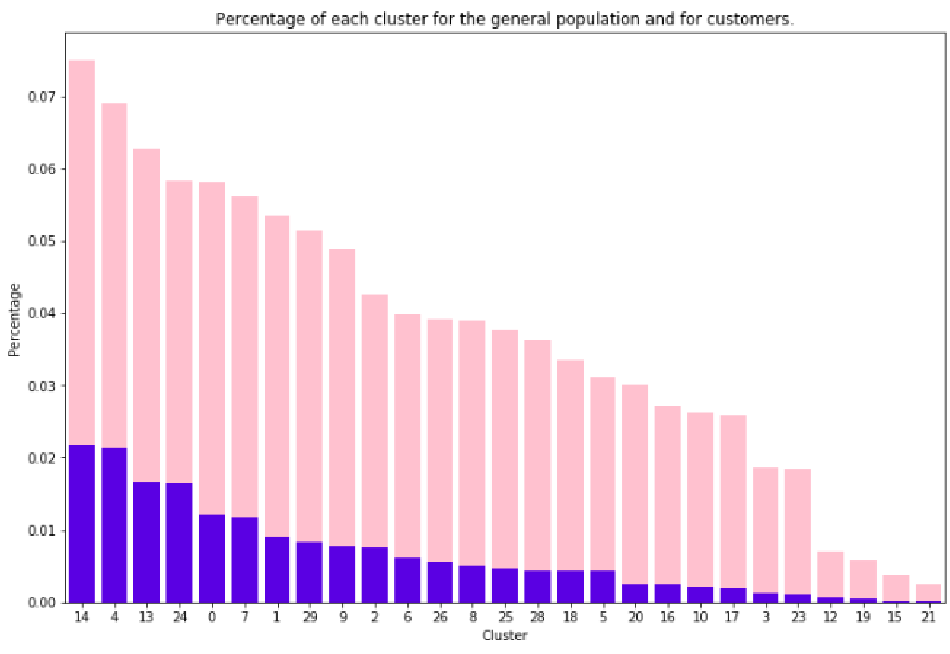
To perform the segmentation, we use the skeans class Kmeans to execute the k-means cluster and group data in similar clusters in the data transformed by the PCA. The k-means clustering algorithm can be divided into a few steps, which are:

- Find the centroid where the point is closest
- Assign this point to this cluster
- For each centroid in the cluster, it is necessary to move this point so that it is in the center of all points assigned to that cluster.
- Repeat the previous steps until convergence is achieved and the points no longer change the cluster membership or until the specified number of iterations is reached.

Our data set has a large size so we tested the count of 30 clusters to get a complete picture.



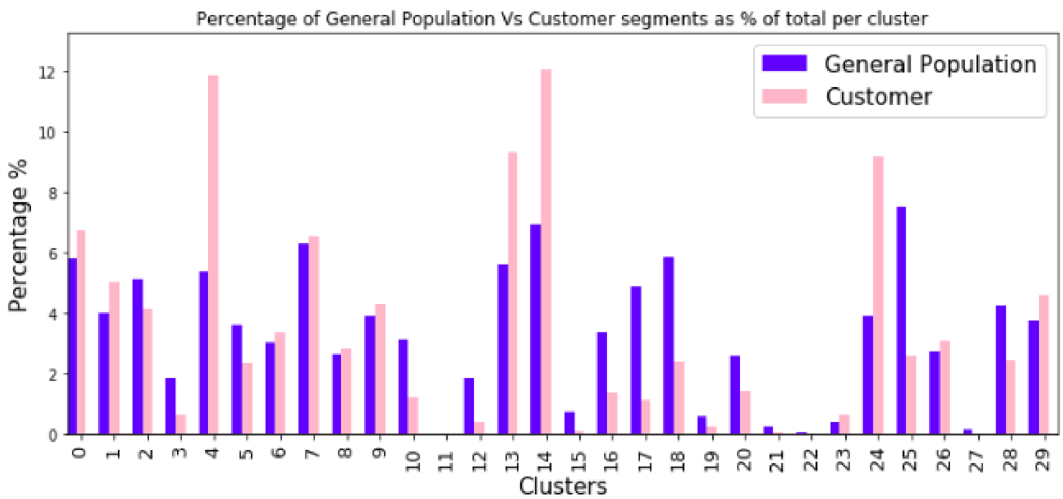
We then compare the customer data with the general population data. Recalling that, as previously mentioned, pre-processing of data from the customer dataset was performed.
In the graph below, the one in pink represents the clusters of the general population, while the one in blue represents the clusters of the client.



We have the difference in proportion for each cluster, in which we can see that the underrepresented cluster is 25 with a difference of 0.049278 and the overrepresented cluster is cluster 4 with a difference of 0.064803.

	Cluster	Difference	Customers	General
0	0	0.008965	9055.0	43599
1	1	0.010399	6787.0	29917
2	2	-0.009755	5630.0	38614
3	3	-0.012049	895.0	14018
4	4	0.064803	15978.0	40167
5	5	-0.012562	3184.0	27126
6	6	0.003516	4545.0	22622
7	7	0.002792	8848.0	47082
8	8	0.002210	3840.0	19684
9	9	0.004074	5804.0	29201
10	10	-0.018810	1666.0	23378
11	11	NaN	NaN	95
12	12	-0.014414	540.0	13820
13	13	0.036743	12554.0	42197
14	14	0.051346	16269.0	51885
15	15	-0.005995	144.0	5300
16	16	-0.019977	1831.0	25171
17	17	-0.037662	1511.0	36667
18	18	-0.034386	3242.0	43829
19	19	-0.003347	327.0	4330
20	20	-0.011569	1931.0	19416
21	21	-0.001801	92.0	1863
22	22	NaN	NaN	327
23	23	0.002457	852.0	2891
24	24	0.052452	12374.0	29406
25	25	-0.049278	3486.0	56363
26	26	0.003543	4135.0	20323
27	27	NaN	NaN	1047
28	28	-0.018172	3298.0	31970
29	29	0.008431	6226.0	28276

Each individual can be divided into 30 groupings. The graph below shows the distribution comparing the general population and the population of customers. Looking at the data, we can conclude that the strongest basis for becoming a customer is in groups where the customer population represents the majority of the general population graphs, which in our case are groups 4, 13, 14 and 24. In addition that it is possible to analyze the resources that have more weight until less weight for the cluster under representation in case 25 and for the over representation that is cluster 4.



In part 2 of this project we work with the Supervised learning model, for that we carry out the study of the data from the file MAILOUT_TRAIN.csv and all the pre-processing done previously, in the clena_data () function.
This dataset includes the RESPONSE column, which indicates whether a person is a customer of the company or not after the campaign. Evaluating the RESPONSE column in the data set, we have the following division of classes in which 33760 represents class 0, that is, individuals who have become customers and 424 belong to class 1 who are individuals who have not become customers.

```
#Number of people who did not become a customer
customers = mailout_train_clean[mailout_train_clean['RESPONSE']==0].shape
#Number of people who became customers
not_customers = mailout_train_clean[mailout_train_clean['RESPONSE']==1].shape
```

```
print("Number of people who became customers: {}".format(customers[0]))
print("Number of people who did not become a customer: {}".format(not_customers[0]))
```

```
Number of people who became customers: 33760
Number of people who did not become a customer: 424
```

With the data ready, let's move on to the implementation part.

2. Implementation

- Benchmark Implementation

  Como mencionado anteriomente o modelo benckmark é o classificador LogisticRegression e utilizamos o ROC-AUC para valiar o nosso modelo. Assim o score foi de 59,8%

```
LogisticRegression
Time taken : 89.37 secs
Best score : 0.598
```

- Others Implementation¶

Observing the table below, we can conclude that, of the three models, the one that obtained the best score was the Logistic Regression, however it is necessary to improve this value. We will do this by changing some hyperparameters in the model.

	best_score	time_taken	best_est
LogisticRegression	0.597958	89.37	LogisticRegression(C=1.0, class_weight=None, d...
RandomForestClassifier	0.499896	4.31	(DecisionTreeClassifier(class_weight=None, cri...
KNeighborsRegressor	0.506314	4290.72	KNeighborsRegressor(algorithm='auto', leaf_siz...

3. Refinement

The following hyperparameters can be changed in order to improve the score:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

Sklearn provides the GridSearchCV class which consists of creating a grid with all possible past hyperparameters and obtaining the model with each possibility. The best model will be saved and used to deploy to the Flask Endpoint.

Improving the LogisticRegression model

```
hiperparametros = {"penalty":["l1","l2"],
                   "max_iter": [100, 300, 500]}

LogisticReg = LogisticRegression(random_state=42)
LogisticReg_best_score, LogisticReg_best_est, _ = fit_classifier(LogisticReg, hiperparametros)
LogisticReg_best_est
```

After training all possible models and seeing which are the best parameters, it can be seen that we did not obtain a very good score, which was 60.28%.

```
LogisticRegression
Time taken : 2661.72 secs
Best score : 0.6028
****************************************

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

Results

Now that we have a model to predict which individuals are most likely to respond to a campaign, let's test this model for competition in Kaggle.
Remembering that for this, we explored and prepared the test data that were saved in the file MAILOUT_052018_TEST.csv, using the clean_data () function, then we performed the transformation of test data resources using the StandardScaler.
To submit the data in the Kaggle competition we prepared the file kaggle_submission_file.csv which has two columns, the first being a copy of the LNR, which has the identification number of each person and the second column RESPONSE which is the probability of each individual becoming one client.

Testing the best model

```
test_response = LogisticReg_best_est.predict(test_scaled)
```

```
test_response
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
test_response_proba = LogisticReg_best_est.predict_proba(test_scaled)
```

```
test_response_proba.shape
```

```
(34152, 2)
```

```
LNR.shape
```

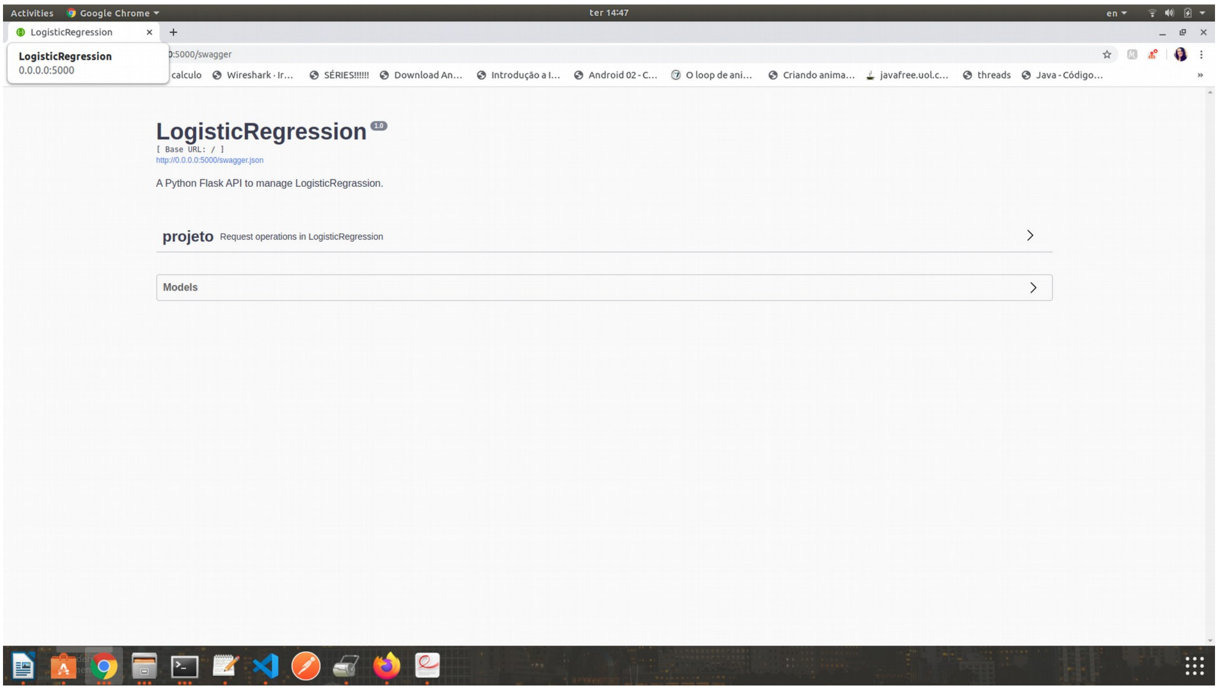
```
(42833,)
```

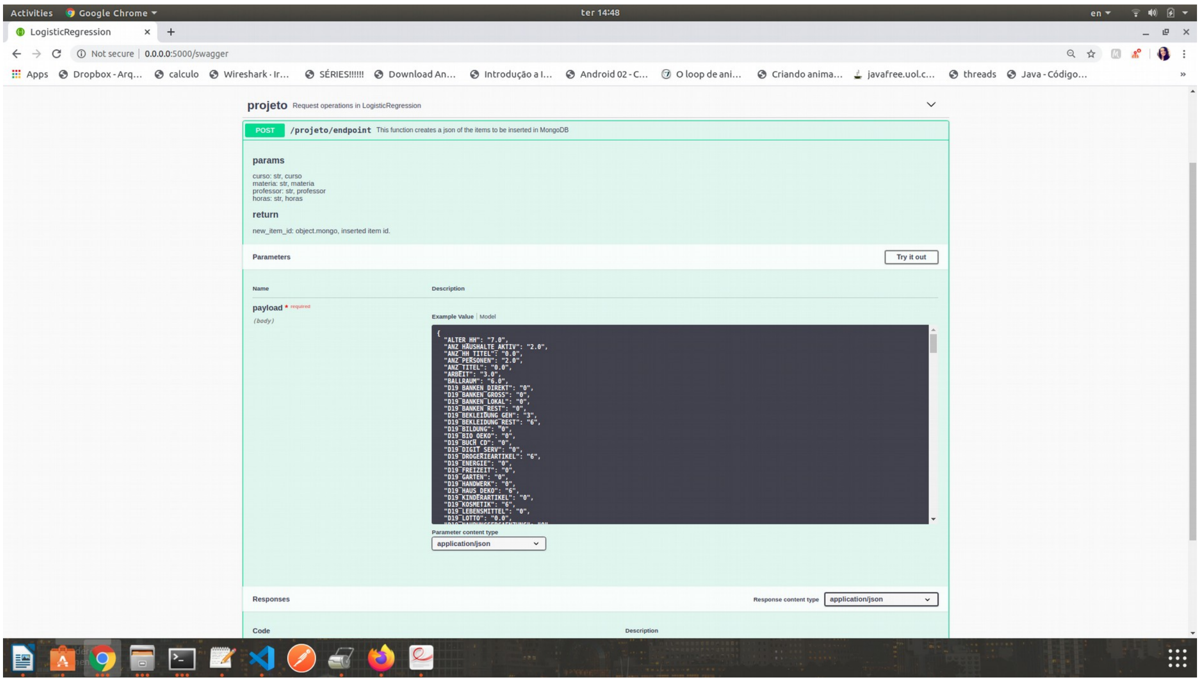
```
kaggle = pd.DataFrame({'LNR': LNR, 'RESPONSE': test_response_proba[:, 1]})
kaggle.head()
```

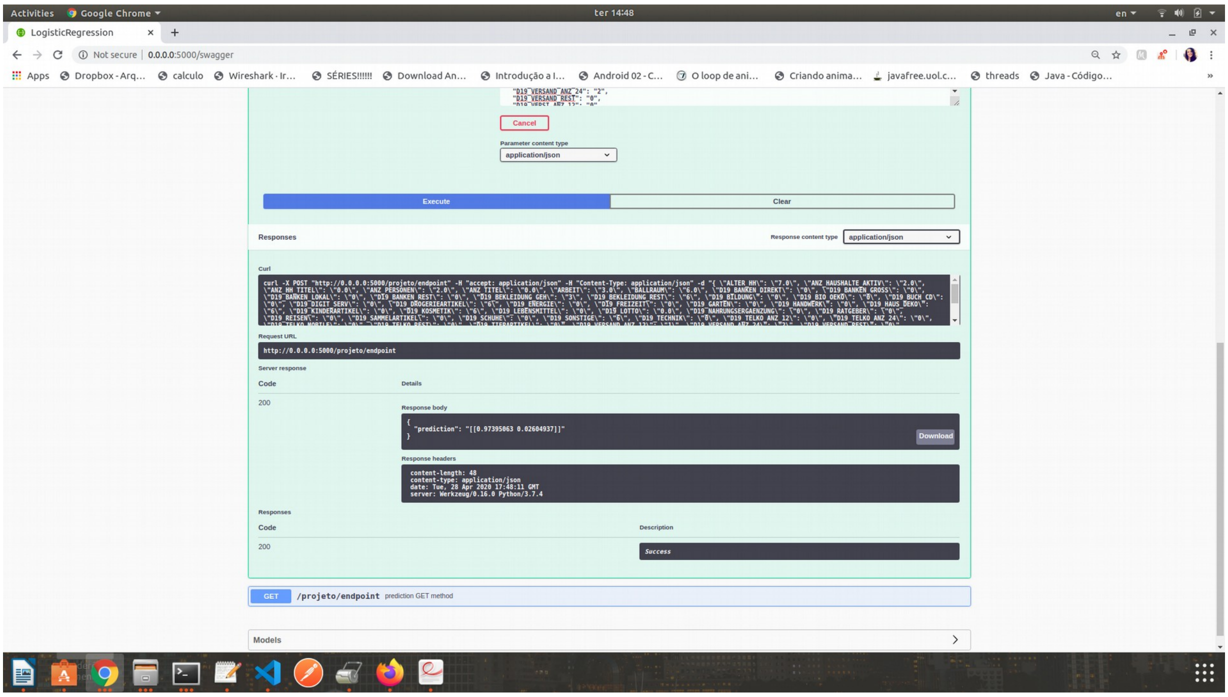
	LNR	RESPONSE
0	1754	0.110566
1	1770	0.014949
2	1465	0.000740
3	1470	0.000491
4	1478	0.012286

Deployment

During the Nanodregree program, we used the AWS Deploy Endpoint tool, so we developed an endpoint using Python Flask and Python Flask-Restful, it is a web application that contains only one endpoint and one HTTP (POST) method.
To access the endpoint, it is necessary to run the app.py code and then access the swagger at http://0.0.0.0:5000/swagger, there is an example of data, which will be made the prediction.







Improvements

The roc_auc score can be improved by trying other algorithms that perform better with unbalanced data, after researching, I realized that a possible algorithm would be XGBOOST. Techniques could be used to deal with unbalanced classes, in addition to further reducing the dimensions with the PCA. And better adjusting the hyperparameters of the best model to classify this data.

References

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
https://www.kaggle.com/c/udacity-arvato-identify-customers/leaderboard
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
https://scikit-learn.org/stable/modules/grid_search.html
https://flask-restplus.readthedocs.io/en/stable/parsing.html