Predicting the success of Marketing Campaigns using Machine Learning.

Sebastiano Denegri

April 15th, 2021

Business has only two functions - marketing and innovation.
Milan Kundera

Supervised Learning: Classification – IBM / Coursera.

Table of contents.

1. Introduction.

1.1. Scope of the project.

When it comes to marketing, historical data should always drive strategy and planning. **Predictive analytics is the next level of using that data for marketing success.**

Predictive analytics is the use of data, statistical algorithms and AI techniques to identify possible future outcomes. This can help companies stay ahead of the curve and assess the future of their marketing efforts.

Scope of this project was building a predictive Model, using historical data and AI algorithms, able to forecast whether a customer will respond positively, or negatively, to a given Marketing Campaign. Predictive information such as this is, of course, very valuable for organizations, since it can **help companies make more sound decisions when it comes to strategic planning, marketing spending, customer and target market segmentation, ROI prediction, and understand the main drivers/reasons that lead to the success (or unsuccess) of a marketing campaign.**

To develop such a predictive model, I used an **analytic, predictive, machine-learning driven approach**, to build a Classifier algorithm able to use the available historical data to predict what the customer's response to a given campaign would be. Main focus of the Classification Model was on **Prediction**.

Target market of the project are markerting-savy companies, willing to leverage the power of Artificial Intelligence to better anticipate marketing campaign's return, manage budget in a more efficient way, fine-tune their marketing efforts, as well as understand main customer drivers.

Source of data: Kaggle.com.

2.2. Data Understanding.

The dataset contained information about tele-marketing campaigns from a Portuguese banking institution. Purpose of these campaigns was to prompt their clients to subscribe for a specific financial product (Term Deposit). After each call, clients were asked their intentions of either subscribing to the product (indicating a successful campaign) or not (unsuccessful campaign).

The dataset had **41,188 instances of calls to clients** (rows) and **21 variables** (columns): **11 object-types** (including the Target Variable) and **10 numeric-types**, 5 floats and 5 integers. In some cases, same client was contacted multiple times, although each call was considered independent from another even if the client was the same.

Variables can be divided into following groups:

1. **Customer-related variables:**
    a. age: client's age (range: 17 - 98)
    b. job: client's type of job (12 categories)
    c. marital: marital status (4 categories: 'married', 'single', 'divorced', 'unknown')
    d. education: level of education (8 categories: 'illiterate', 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'professional.course', 'university.degree', 'unknown')
    e. default: if the client has credit in default (3 categories: 'no', 'unknown', 'yes')
    f. housing: if the client has a mortgage on his/her house (3 categories: 'no', 'unknown', 'yes')
    g. loan: if the client has a personal loan (3 categories: 'no', 'unknown', 'yes')
2. **Variables related to last contact (current campaign):**
    a. contact: type of communication (2 categories: 'telephone', 'cellular')
    b. month: month of last contact (10 categories)
    c. day_of_week: day of last contact (5 categories)
    d. duration: call duration, in seconds (range: 0 - 4,918 (more than 1 hour and 20 minutes))
    e. campaign: number of contacts performed during this campaign and for this client (range: 1 - 56)
3. **Other marketing-related variables:**
    a. pdays: number of days from last contact (range: 0 - 999)
    b. previous: number of contacts performed before this campaign and for this client (range: 0 - 7)
    c. poutcome: outcome of previous marketing campaign (3 categories: 'nonexistent', 'failure', 'success')
4. **Socioeconomic variables:**
    a. emp.var.rate: employement variation rate - quarterly indicator (range: (-3.4) - 1.4)
    b. cons.price.idx: consumer price index - monthly indicator (range: 92.201 - 94.767)
    c. cons.conf.idx: consumer confidence index - monthly indicator (range: (-50.8) - (-26.9))
    d. euribor3m: euribor 3 month rate - daily indicator (range: 0.634 - 5.045)
    e. nr.employed: number of employees - quarterly indicator (range: 4963.6 - 5228.1)
5. **Target variable:**
    a. subscribed: whether the customer signed up to the product (2 categories: 'yes', 'no').

Some first considerations about the dataset, after conducting an initial descriptive statistical analysis:

- Several categorical variables had quite an imbalanced distribution: job, marital, education, default, loan, month, poutcome.

- Some numeric-type variable had quite a skewed (either right or left) distribution: duration, campaign, pdays, previous….
- **Target Variable is greatly imbalanced with almost 89% of customers not subscribing (negative class)** to the advertised product.

2. Methodology.

In order to build an efficient Classification model, with focus on **prediction,** I followed the below methodology:

- Defined the scope of the project: to develop a Classification Model to predict whether instances of call of a telemarketing campaign will be successful or not, with regard to the subscription of the advertised product; model's main focus was **prediction**.
- Selected an analytic, predictive, machine-learning driven approach.
- Data understanding: content and format analysis.
- Data Exploration:
    - Data Cleaning
    - Exploratory Data Analysis for Feature Selection.
    - Data Preparation for modeling
- Model Development:
    - Classifier 1: K-Nearest Neighbors (KNN)
    - Classifier 2: Random Forest
    - Classifier 3: Support Vector Machines
    - Classifier 4: Ensemble Models - Stacking Classifier
- Model Evaluation: compare performance of different models, and select the best for the project purpose.

2.1. Data Cleaning.

Duplicates.

Dataset had only 12 duplicates, that were dropped.

```
Number of duplicates in the dataset: 12.
Number of rows, after removing duplicates: 41,176.
```

Missing values.

No missing values were found in the dataset.

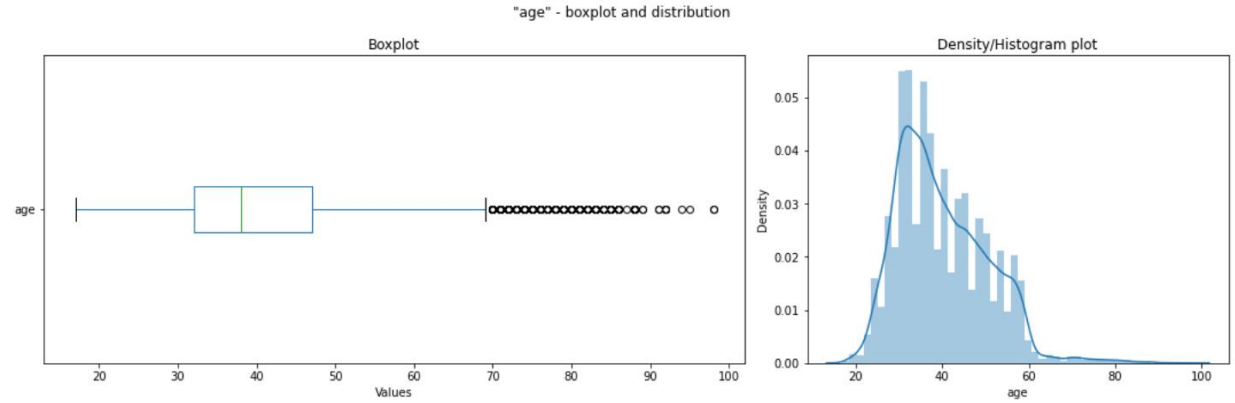
Outlier and Distribution Analysis.

I created a function to visualize both Boxplots and Density/Histogram plots of numeric-type variables, to check for possible outliers and variable distributions; I, then, created a function to

find out the “No-outlier region”, the number of outliers, and percentage of outliers, using the Interquartile Range method.

age

```
age - Data is not normally distributed.
```

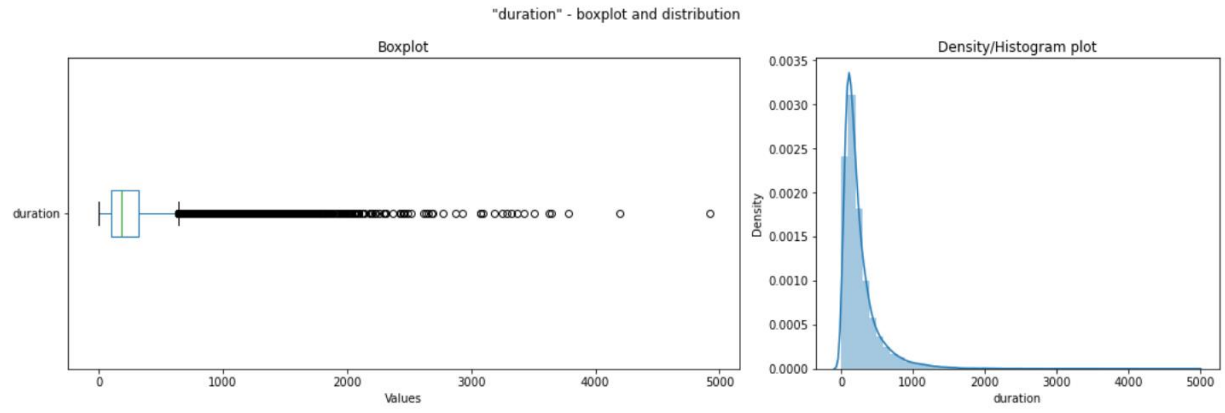



```
age attribute - no outlier region: 17 - 69.5.
Number of outliers: 468
Percentage of outliers: 1.14%.
First ten outliers: [70, 76, 73, 88, 88, 88, 88, 88, 88, 88]
```

age attribute had few outliers (1% of data) which seemed to be correct observations: no customers was older than 100 years. The variable was not normally distributed.

duration

```
duration - Data is not normally distributed.
```



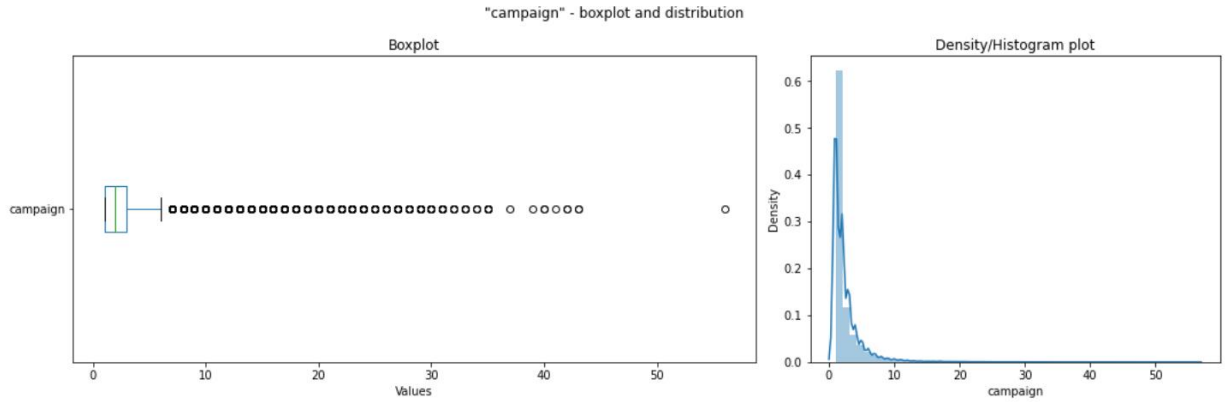

```
duration attribute - no outlier region: 0 - 644.5.
Number of outliers: 2963
```

```
Percentage of outliers: 7.2%.
First ten outliers:[1666, 787, 812, 1575, 1042, 1467, 849, 677, 2033, 843]
```

Outliers accounted for more than 7% of data, all beyond the max limit: 645 secs (about 10 minutes of call). 2 outliers are beyond 4,000 seconds (more than 1-hour call). Although this seemed to be quite a long duration for a marketing call, these observations may have been part of the natural variability of the data: I kept those records in the dataset.

campaign

```
campaign - Data is not normally distributed.
```

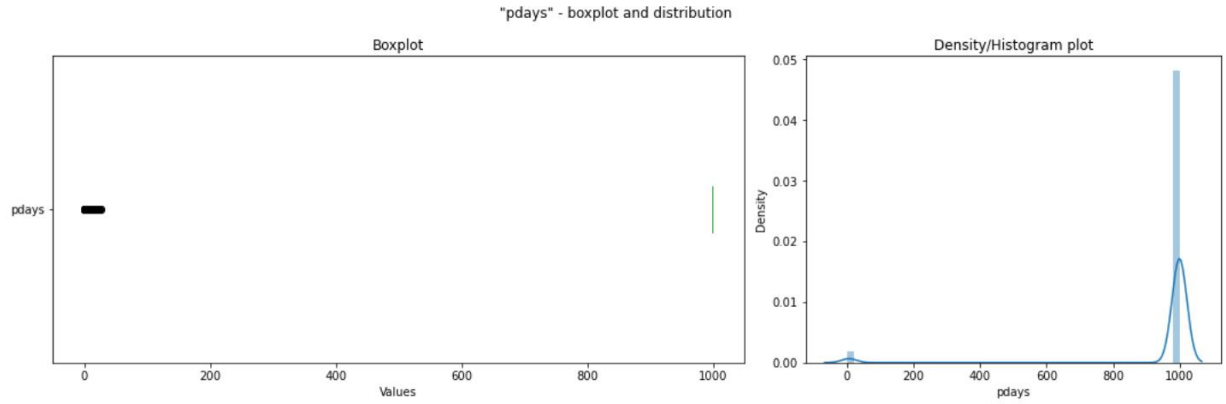



```
campaign attribute - no outlier region: 1 - 6.0.
Number of outliers: 2406
Percentage of outliers: 5.84%.
First ten outliers: [7, 8, 8, 7, 7, 7, 7, 9, 9, 7]
```

The variable was not normally distributed. **campaign** outliers accounted for almost 6% of data, and, again, they were all beyond the max limit (6 contacts); almost 95% of customers have been contacted from 1 to 6 times. 1 customer have been contacted 56 times in the current campaign. Again, this seemed to be quite a big number, but the observation may have been part of the natural variability of the data. I kept those outliers in the dataset.

pdays

```
pdays - Data is not normally distributed.
```

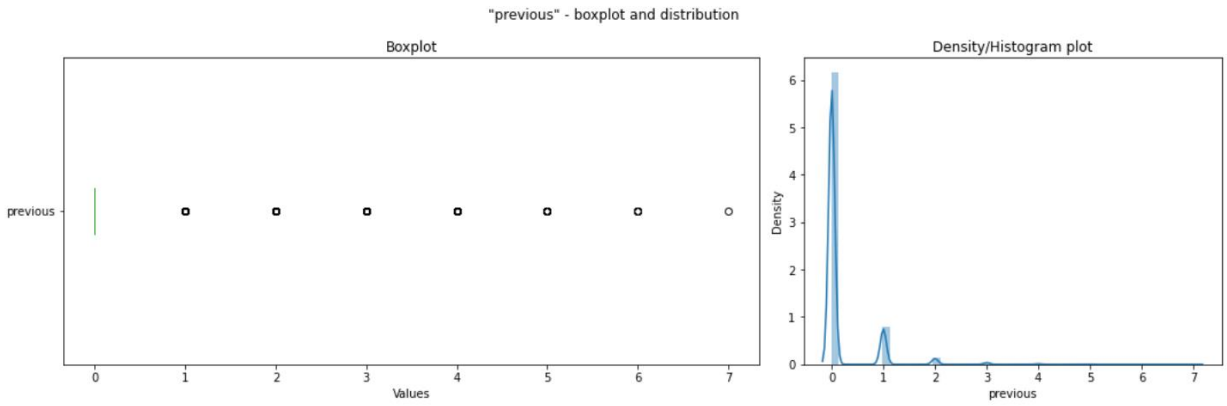



```
pdays attribute - no outlier region: 999.0 - 999.0.
Number of outliers: 1515
Percentage of outliers: 3.68%.
First ten outliers: [6, 4, 4, 3, 4, 5, 5, 1, 6, 4]
```

pdays distribution looked odd; 96% of data had the same value: 999, that is to say the last contact had happened 999 days before the new campaign; 1,515 observations (3.7% of data) had been contacted from 0 to 27 days before the current campaign. The attribute seemed to be wrongly recorded: therefore, I decided to drop the column.

previous

```
previous - Data is not normally distributed.
```




```
previous attribute - no outlier region: 0.0 - 0.0.
Number of outliers: 5625
Percentage of outliers: 13.66%.
First ten outliers: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

All values that were not 0 were considered outliers: 5,625 observations (13.66% of data). I had a deeper look at the value frequencies (%):

previous values and frequencies (% values):

```
0    0.863391%
1    0.110768%
2    0.018312%
3    0.005246%
4    0.001700%
5    0.000437%
6    0.000121%
7    0.000024%
Name: previous, dtype: float64
```

previous values and frequencies (absolute values):

```
0    35551
1     4561
2      754
3      216
4       70
5       18
6        5
7        1
Name: previous, dtype: int64
```

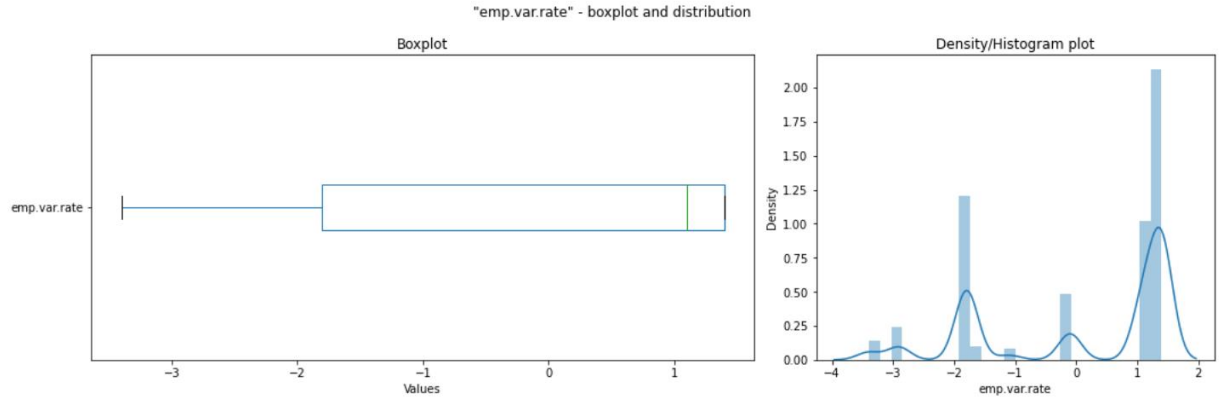
86% of customers had never been contacted prior to this campaign (value = 0), 11% had been contacted once, 1.8% twice, the remaining 1.2% from 3 to 7 times.
To verify the correctness of the attribute, I took a look at the categories of **poutcome** (the outcome of previous campaigns), and compared the categories' frequencies. Outcomes are as follows:

poutcome values and frequencies (absolute values):

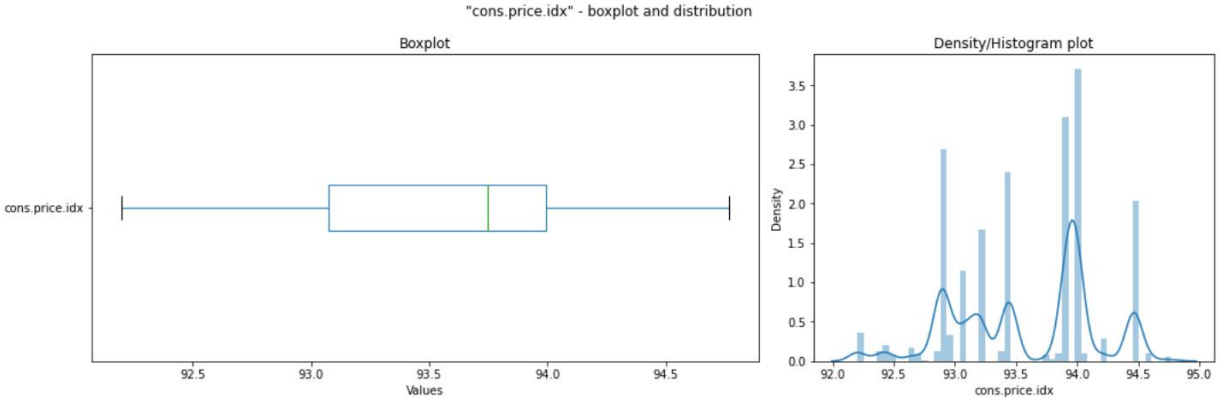
```
nonexistent    35551
failure         4252
success         1373
Name: poutcome, dtype: int64

All customers who had never been contacted prior to this campaign have
"nonexistent" as poutcome value.
All customers with a "poutcome" different that "nonexistent" had been
contacted prior to this campaign.
Attributes seem to be correctly recorded.
```

Socioeconomic Variables

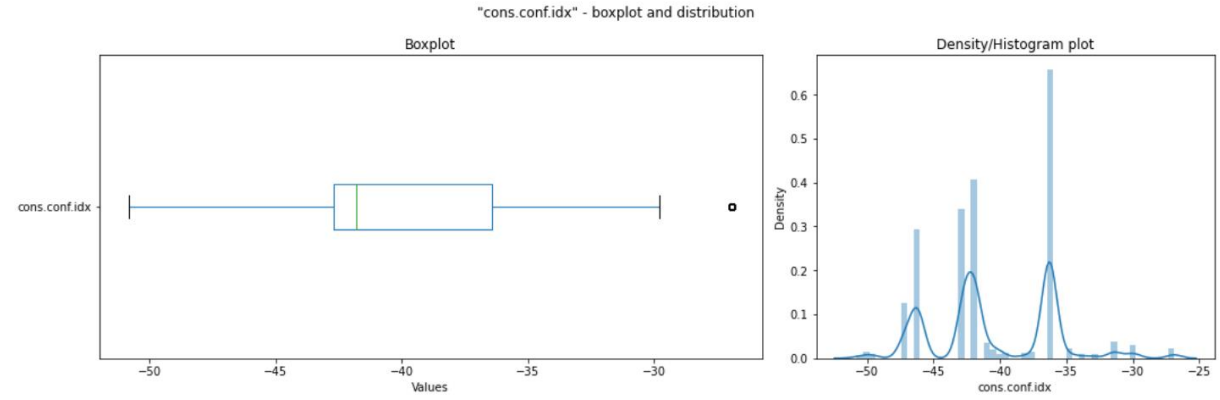
emp.var.rate - Data is not normally distributed.



cons.price.idx - Data is not normally distributed.

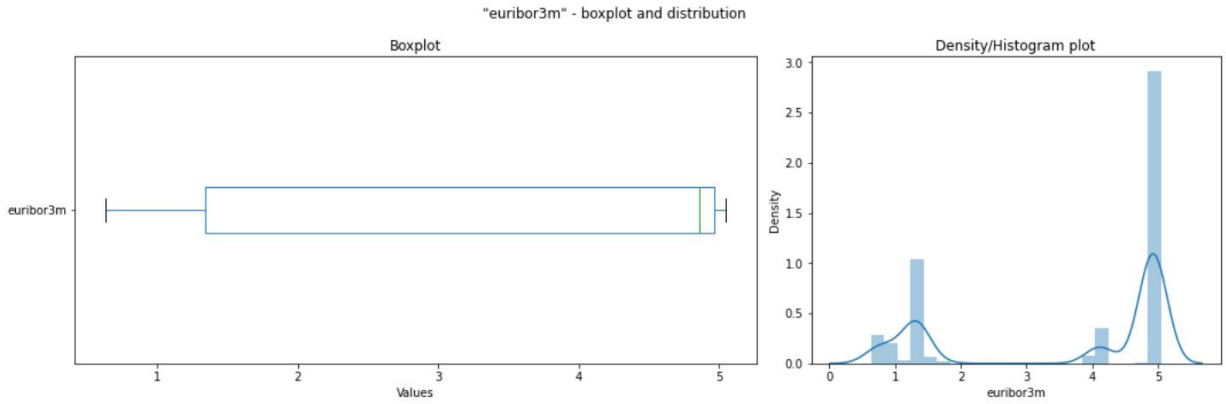


cons.conf.idx - Data is not normally distributed.

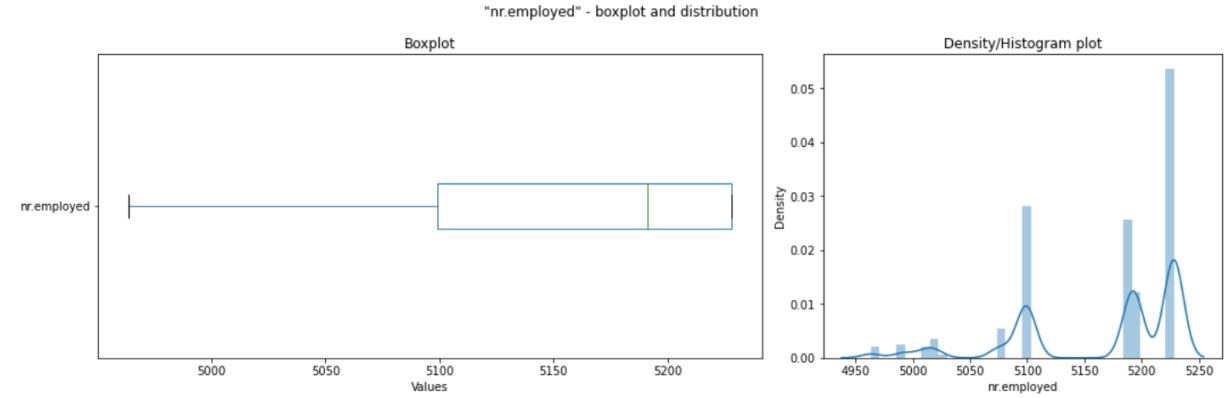


cons.conf.idx attribute - no outlier region: -50.8 - -26.9.
Number of outliers:0
Percentage of outliers: 0.0%.

```
euribor3m - Data is not normally distributed.
```



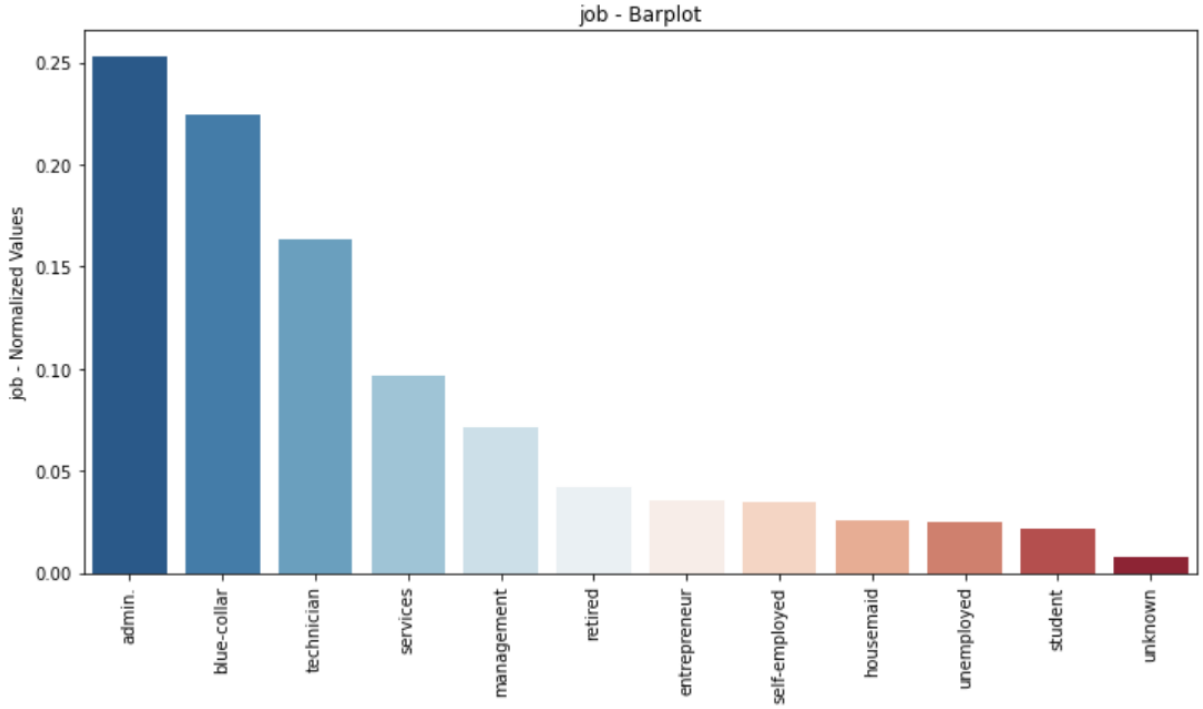

```
nr.employed - Data is not normally distributed.
```



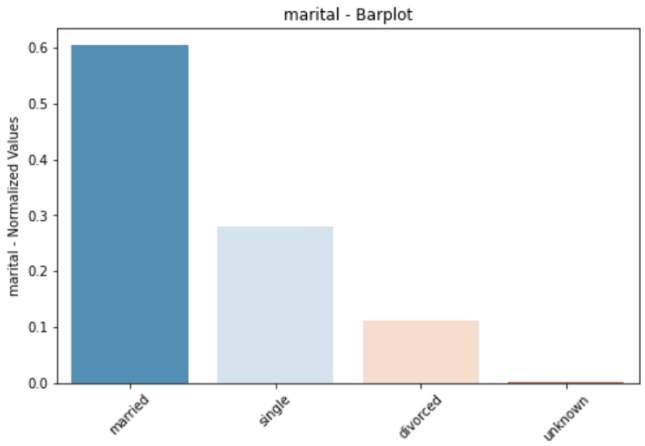
All socioeconomic variables are not normally distributed, and have no outliers.

Frequency Distribution Analysis for Categorical Variables.

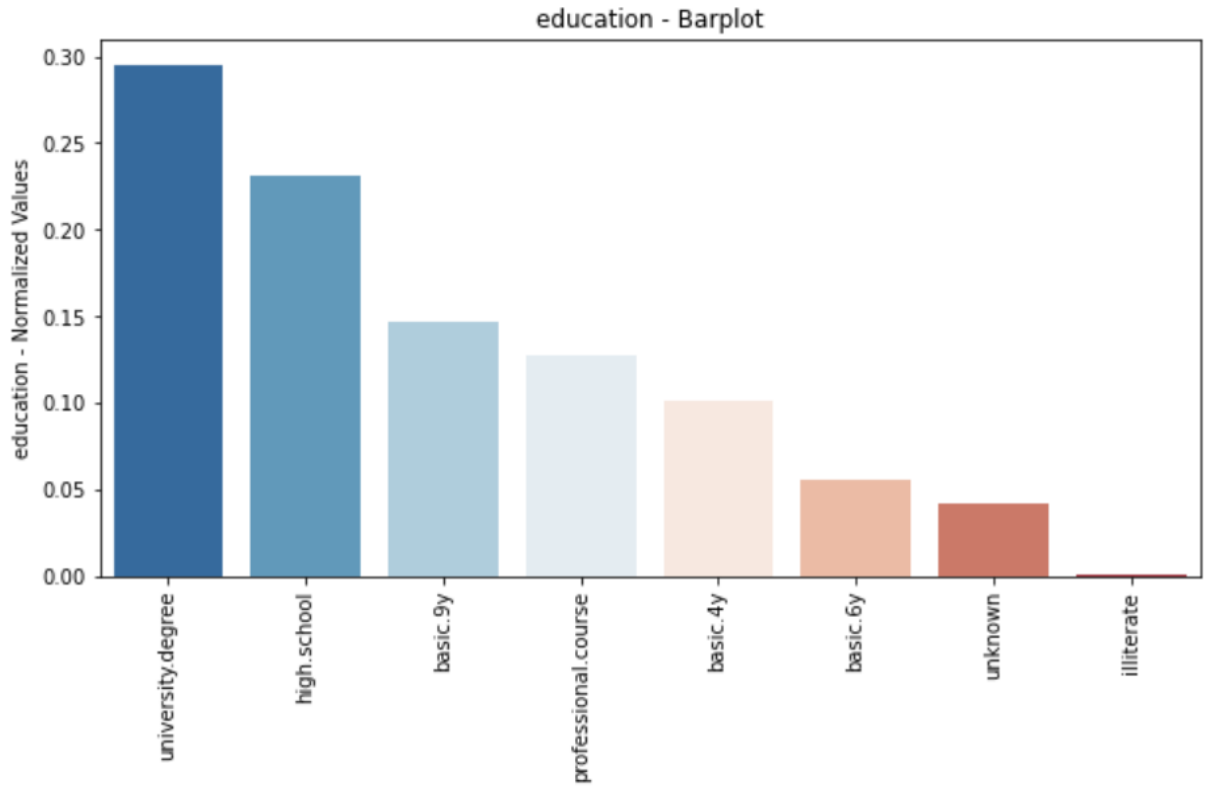
I created a function to visualize Histogram plots and outputs the categories and their frequency.



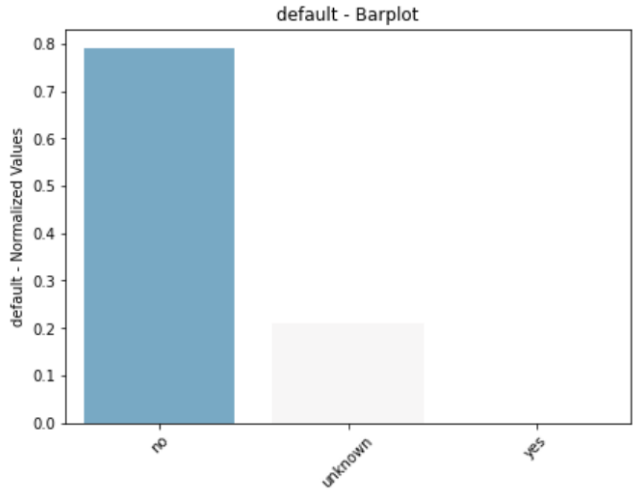

```
job - Value Counts
admin.           10419
blue-collar       9253
technician        6739
services          3967
management        2924
retired           1718
entrepreneur      1456
self-employed     1421
housemaid         1060
unemployed        1014
student            875
unknown            330
Name: job, dtype: int64
```



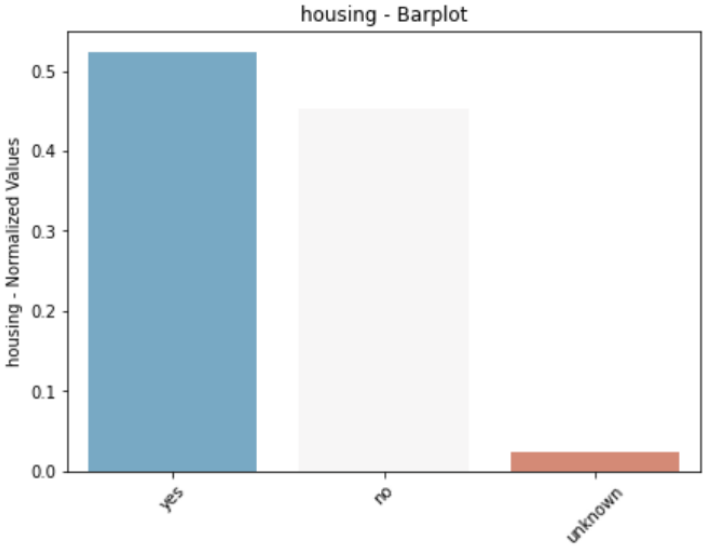

```
marital - Value Counts
married     24921
single      11564
divorced     4611
unknown        80
Name: marital, dtype: int64
```



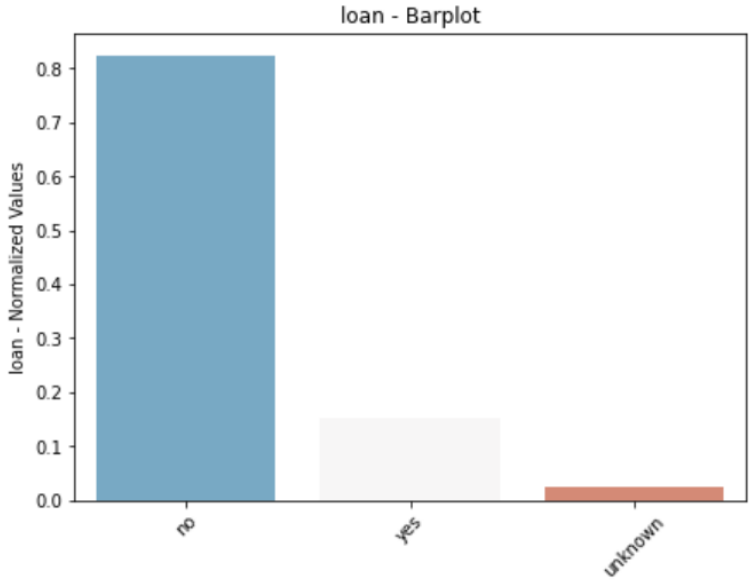

```
education - Value Counts
university.degree      12164
high.school             9512
basic.9y                6045
professional.course     5240
basic.4y                4176
basic.6y                2291
unknown                 1730
illiterate                18
Name: education, dtype: int64
```




```
default - Value Counts
no          32577
unknown      8596
yes             3
Name: default, dtype: int64
```

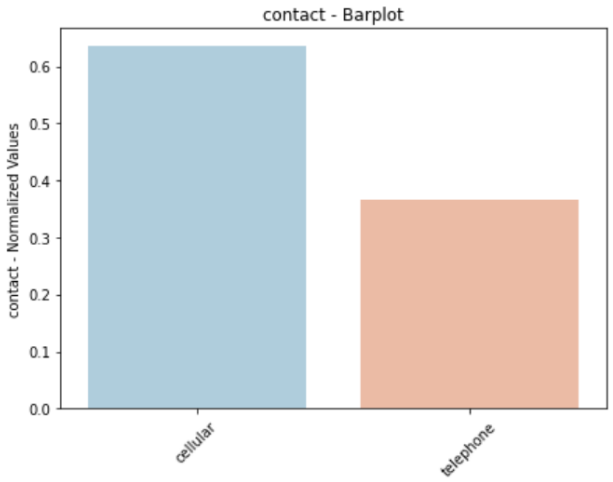


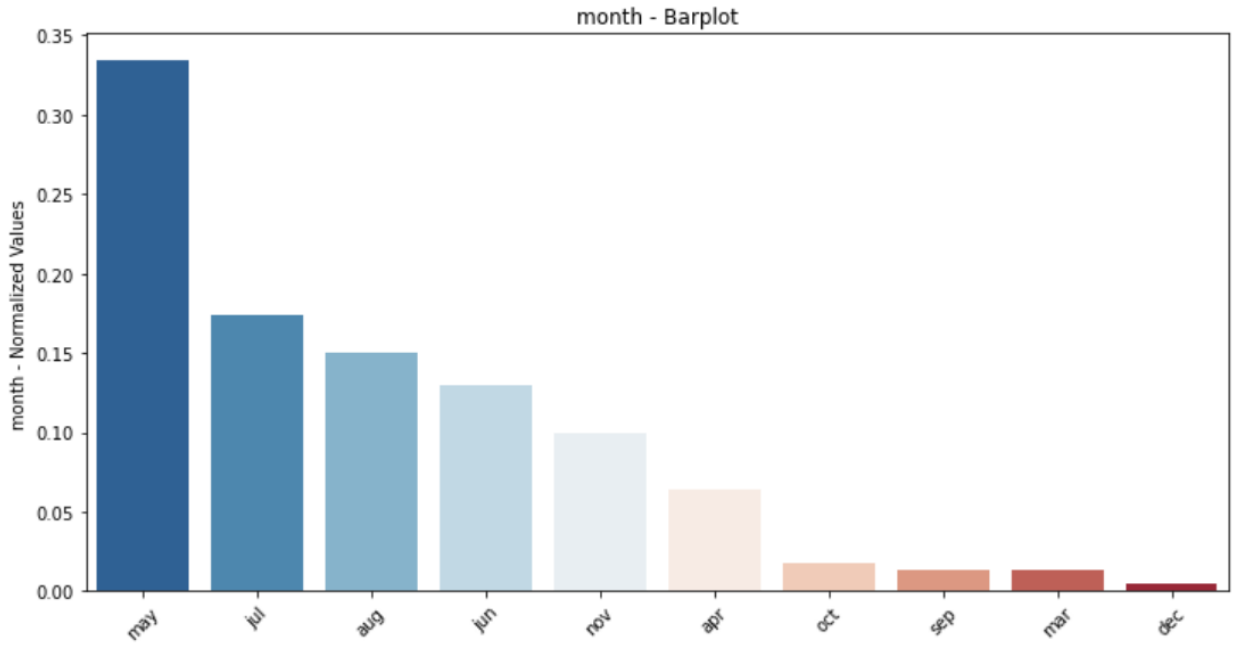

```
housing - Value Counts
yes         21571
no          18615
unknown       990
Name: housing, dtype: int64
```



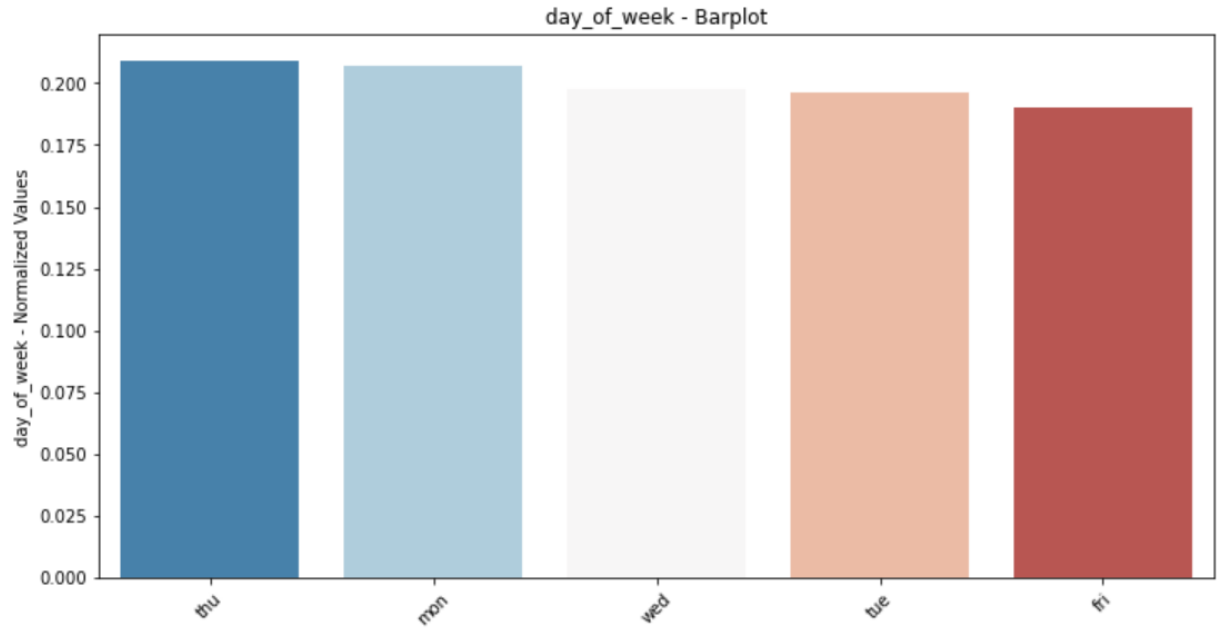

```
loan - Value Counts
no          33938
yes          6248
unknown       990
Name: loan, dtype: int64
```

```
contact - Value Counts
cellular     26135
telephone    15041
Name: contact, dtype: int64
```

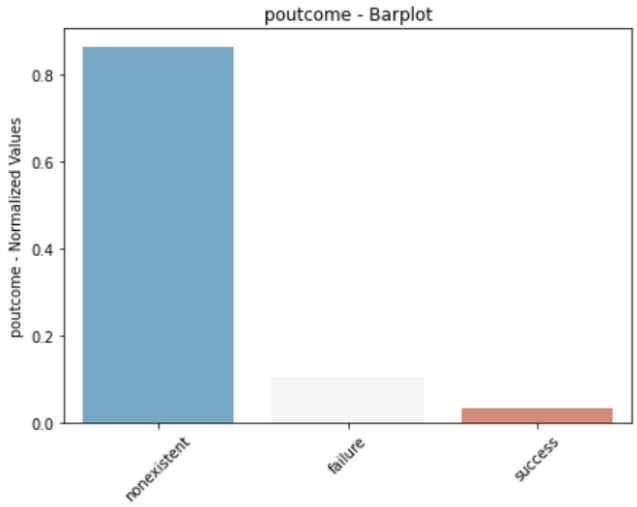





```
month - Value Counts
may    13767
jul     7169
aug     6176
jun     5318
nov     4100
apr     2631
oct      717
sep      570
mar      546
dec      182
Name: month, dtype: int64
```

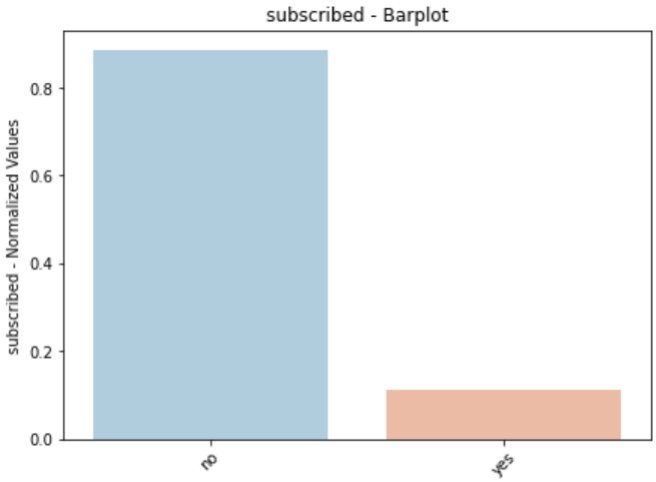



```
day_of_week - Value Counts
thu    8618
mon    8512
wed    8134
tue    8086
fri    7826
Name: day_of_week, dtype: int64
```




```
poutcome - Value Counts
   nonexistent    35551
   failure         4252
   success         1373
Name: poutcome, dtype: int64
```

```
subscribed - Value Counts
no      36537
yes      4639
Name: subscribed, dtype: int64
```



Several categorical attributes had quite an imbalanced distribution (job, marital, education, default, loan, month, poutcome), including the Target Variable **subscribed**. No mistakes or wrongly recorded data had been found, although some variables had a very low number of values in few categories. This, although the data was not wrong per se, could create some problems during the statistical analysis. Therefore, I performed the transformations below:

- Attribute **job**: grouped *illiterate* and *unknown* categories.
- Attribute **default**: grouped *yes* and *unknown* categories.

After the cleaning, the dataset was made of **41,176 observations and 20 attributes.**

2.2. EDA - Exploratory Data Analysis.

2.2.1. Examine Relationships between Categorical Variables and the Target

To check for possible correlations between the target and categorical predictors, I used the following visual and statistical techniques:

- Stacked Bar Charts
- Cross Tables
- Cramer’s V metric

The assumptions for Cramer's V are the followings, which were met:

- Variables of interest are categorical
- 2 or more unique values per category

Cramer's V is a measure of the association between variables, based on the chi-square test. Chi-square test assumptions are below:

- Categories of the variables are mutually exclusive.
- Observations must be independent.

- Variables are categorical, either nominal or ordinal.
- In contingency tables, values in each cell should be 5 or more in at least 80% of the cells, and no cell should have a value less than one.

  The first 3 conditions for the chi-square test were met. I built contingency tables to check for the 4th assumption.

job subscribed	admin.	blue-collar	entrepreneur	housemaid	management	retired	self-employed	services	student	technician	unemployed	unknown
no	9068	8615	1332	954	2596	1284	1272	3644	600	6009	870	293
yes	1351	638	124	106	328	434	149	323	275	730	144	37

marital subscribed	divorced	married	single	unknown
no	4135	22390	9944	68
yes	476	2531	1620	12

education subscribed	basic.4y	basic.6y	basic.9y	high.school	professional.course	university.degree	unknown/illiterate
no	3748	2103	5572	8481	4645	10495	1493
yes	428	188	473	1031	595	1669	255

default subscribed	no	unknown/yes
no	28381	8156
yes	4196	443

housing subscribed	no	unknown	yes
no	16590	883	19064
yes	2025	107	2507

loan subscribed	no	unknown	yes
no	30089	883	5565
yes	3849	107	683

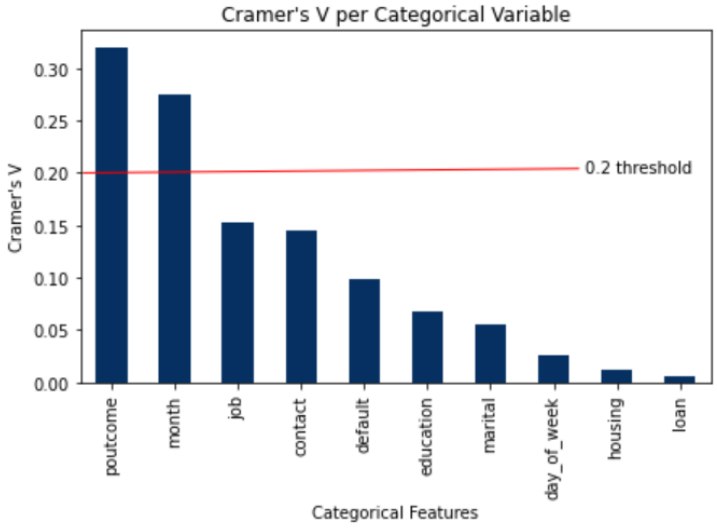
contact subscribed	cellular	telephone
no	22283	14254
yes	3852	787

month subscribed	apr	aug	dec	jul	jun	mar	may	nov	oct	sep
no	2092	5521	93	6521	4759	270	12881	3684	402	314
yes	539	655	89	648	559	276	886	416	315	256

day_of_week	fri	mon	thu	tue	wed
subscribed					
no	6980	7665	7574	7133	7185
yes	846	847	1044	953	949

poutcome	failure	nonexistent	success
subscribed			
no	3647	32411	479
yes	605	3140	894

In all contingency tables, there were not cells with a value less than 5: all assumptions were met, so I moved forward with the analysis, using Cramer's V as the metric to measure the associations between the categorical predictors and the target Variable.



For feature selection, I kept only the variables with a Cramer's V greater than 0.2 (moderate association):

- poutcome (outcome of previous marketing campaign: 'nonexistent', 'failure', 'success')
- month (month of last contact (current campaign))

2.2.2. Examine Relationships between Numeric-Type Variables and the Target

None of the numeric-type variables were normally distributed, therefore a metric like Point-Biserial Correlation (mathematically equivalent to the Pearson correlation) is not valid (normality of the continuous variables is one of the key assumptions for Point-biserial Correlation).

Therefore, to estimate potential associations between the numeric-type and the categorical variable, I used the Kruskal-Wallis H test, that is the non-parametric version of one-way ANOVA. This approach consists in measuring the variance in each group (category of the target variable) and comparing it to the overall population variance. If there's a big difference between the variance within the groups and the variation between the group means, it means that the variables are strongly correlated (the numeric-type predictors can explain the target variable). If the variables have no correlation, then the variance in the groups is expected to be similar to the population variance. Since Kruskal-Wallis H test is a non-parametric method, it does not assume a normal distribution of the variables, nor it makes any other key assumptions on the data distribution.

Kruskal-Wallis H test significance level = 5%.

Kruskal-Wallis H test outputs whether there is a significant difference between groups, that is if the numeric-type variable has some impact in explaining the categorical target variable.

```
Kruskal-Wallis H-test results for "age" and "subscribed" variables:
Test-Statistic: 5.843156420996497
P-Value: 0.015637749222475184
Reject NULL hypothesis - Significant differences exist between groups.
The continuous variable has some impact over the target variable.

Kruskal-Wallis H-test results for "duration" and "subscribed" variables:
Test-Statistic: 5008.952641567334
P-Value: 0.0
Reject NULL hypothesis - Significant differences exist between groups.
The continuous variable has some impact over the target variable.

Kruskal-Wallis H-test results for "campaign" and "subscribed" variables:
Test-Statistic: 166.83794105975488
P-Value: 3.629350163987945e-38
Reject NULL hypothesis - Significant differences exist between groups.
The continuous variable has some impact over the target variable.

Kruskal-Wallis H-test results for "previous" and "subscribed" variables:
Test-Statistic: 1662.3956401363578
P-Value: 0.0
Reject NULL hypothesis - Significant differences exist between groups.
The continuous variable has some impact over the target variable.

Kruskal-Wallis H-test results for "emp.var.rate" and "subscribed"
variables:
Test-Statistic: 2520.7461098911913
P-Value: 0.0
Reject NULL hypothesis - Significant differences exist between groups.
The continuous variable has some impact over the target variable.

Kruskal-Wallis H-test results for "cons.price.idx" and "subscribed"
variables:
Test-Statistic: 614.0899232769726
P-Value: 1.4427370342045907e-135
```

```
Reject NULL hypothesis - Significant differences exist between groups.
The continuous variable has some impact over the target variable.

Kruskal-Wallis H-test results for "cons.conf.idx" and "subscribed"
variables:
Test-Statistic: 69.64459433834713
P-Value: 7.101309014386372e-17
Reject NULL hypothesis - Significant differences exist between groups.
The continuous variable has some impact over the target variable.

Kruskal-Wallis H-test results for "euribor3m" and "subscribed" variables:
Test-Statistic: 2930.314757187436
P-Value: 0.0
Reject NULL hypothesis - Significant differences exist between groups.
The continuous variable has some impact over the target variable.

Kruskal-Wallis H-test results for "nr.employed" and "subscribed"
variables:
Test-Statistic: 3318.9556210068745
P-Value: 0.0
Reject NULL hypothesis - Significant differences exist between groups.
The continuous variable has some impact over the target variable.
```

All p-values were less than the Significance level (5%): I, therefore, rejected the Null Hypothesis (the mean is equal across all groups = no association between variables) for all numeric-type variables. In other words, all numeric-type variables had some type of correlation, and explanatory value, over the target.

duration was the variable with the greatest Kruskal-Wallis test statistic value: so it was the most associated variable with the target. Unfortunately, the **duration** of a call is not a known variable at the time of attempting a prediction, so it cannot be used as one of the predictors. Therefore, I dropped **duration** and selected all other numeric-type variables as model features.

Number of selected features for model development: 10 (2 categorical and 8 numerical features).

2.2.3. Feature Engineering.

To prepare the feature set for modeling, I performed following transformations:

- One-Hot Encoding transformation for nominal-categorical variables, dropping one column per each feature to avoid multi-collinearity issues.
- Divide feature set into shuffled train (80%) and test (20%) sets, with stratified technique.

2.3. Model Development

Models have been built, evaluated, and compared following the below methodology:

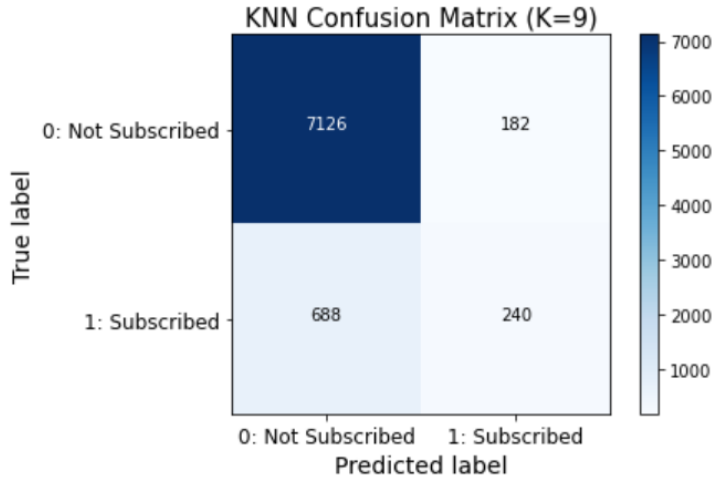
- Created a Pipeline with MinMaxScaler and the classification model.
- Fitted the model on the training set, using GridSearchCV method to find the best hyperparameters with stratified Cross Validation technique (5 stratified shuffled splits).
- Set GridSearchCV scoring argument to F-1 score, **to select the best model with the greatest harmonic mean of precision and recall of the positive class**: considering the greatly imbalanced distribution of the target variable, using accuracy score, as GridSearchCV scoring argument, might have led to select as best model a classifier not really able to predict examples belonging to the minority class (the positive one).
- Make predictions using the test set, and compute the following out-of-sample metrics: Confusion Matrix, Precision, Recall, F1 Score, Receiver Operating Characteristic and Precision/Recall curves, and Area Under the Curve scores.
- Whereas the classifier had probabilistic outcomes, I moved the probability threshold, and calculated the best trade-off between the Sensitivity (True Positive Rate) of positive class and the False Positive Rate, and Precision and Recall, using the following metrics:
    - Youden's J statistic: True Positive Rate – False Positive Rate;
    - F1 Score: harmonic mean between Precision and Recall;
    - Selected the decision threshold with the **greatest macro-averaged F1-score** (the macro average doesn't take into account the class weights, so the model with highest macro-averaged F1-score is the classifier with the best trade-off between negative and positive class predictions).
- If classifiers didn't have the parameter "class_weight", to deal with imbalanced classes, I applied some resampling approaches and compared the model performance, again moving the decision thresholds to find out the best predictive model. The tested resampling approaches were:
    - Undersampling the majority class with **Edited Nearest Neighbors**
    - Oversampling the minority class with **SMOTE (Synthetic Minority Oversampling Technique) algorithm**
    - Resampling the whole dataset combining **Edited Nearest Neighbors and SMOTE.**
- If the models had the parameter "class_weight" to deal with imbalanced classes, I included that parameter as hyperparameter to be tested using GridSearchCV; tested options for "class_weight" were:
    - original weights
    - balanced weights: automatically adjust weights inversely proportional to class frequencies)
    - balanced subsample: the balanced weights are computed based on the bootstrap sample for every tree
    - customized weights: 80% vs 20%, and 70% vs 30% (negative vs positive class).

2.3.1. Classifier 1: K-Nearest Neighbors (KNN).

KNN best hyperparameters, found with GridSearchCV (5 stratified shuffled splits) fitted on the train set:

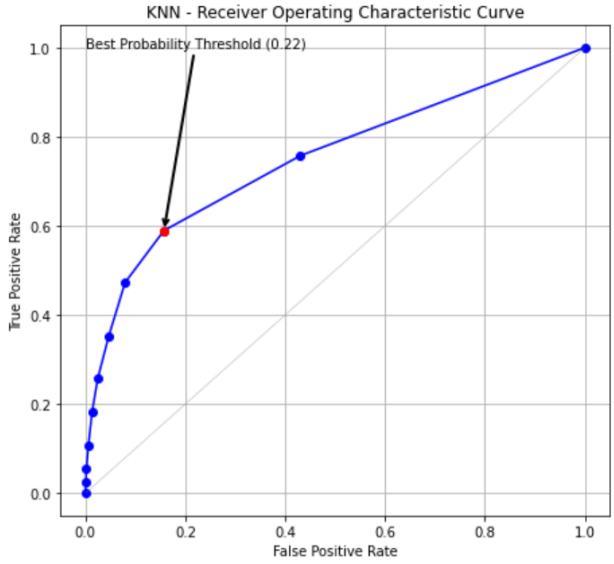
- Number of Neighbors (K): 9.
- Type of Distance: Manhattan Distance (L1).
- Weights of Neighbors: uniform (all points in each neighborhood are weighted equally).

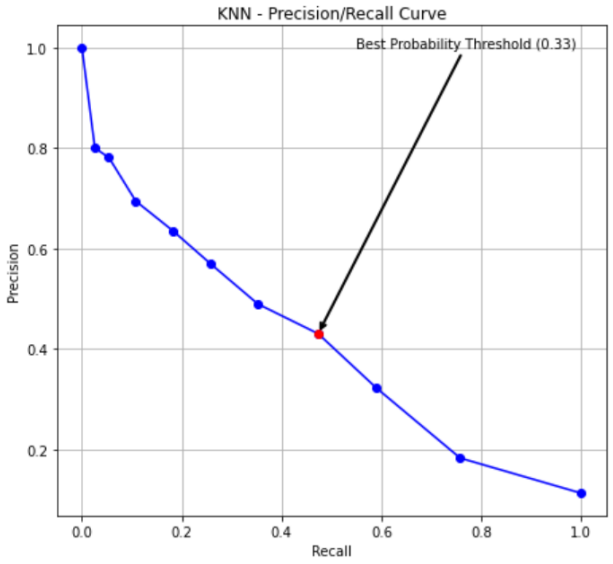
Confusion Matrix and Classification Report:



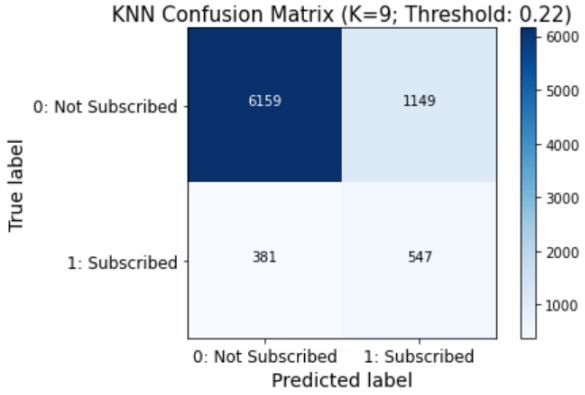

```
KNN: Classification Report
                precision    recall  f1-score   support

Not Subscribed       0.91      0.98      0.94      7308
    Subscribed       0.57      0.26      0.36       928

      accuracy                           0.89      8236
     macro avg       0.74      0.62      0.65      8236
  weighted avg       0.87      0.89      0.88      8236

ROC - Area Under the Curve:  0.75
```

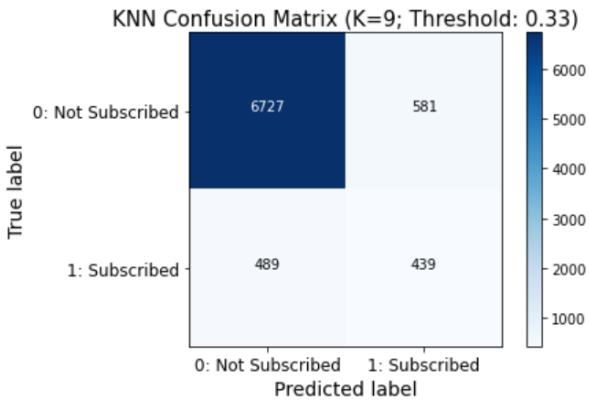
ROC and Precision/Recall curves, with the best decision threshold:





Confusion Matrix and Classification Report - best decision thresholds:






```
KNN: Classification Report (K=9; Threshold: 0.22)
                precision    recall  f1-score   support

Not Subscribed       0.94      0.84      0.89      7308
    Subscribed       0.32      0.59      0.42       928

      accuracy                           0.81      8236
     macro avg       0.63      0.72      0.65      8236
  weighted avg       0.87      0.81      0.84      8236
```

```
KNN: Classification Report (K=9; Threshold: 0.33)
                precision    recall  f1-score   support

Not Subscribed       0.93      0.92      0.93      7308
    Subscribed       0.43      0.47      0.45       928

      accuracy                           0.87      8236
     macro avg       0.68      0.70      0.69      8236
  weighted avg       0.88      0.87      0.87      8236
```

	Precision - Positive Class	Recall - Positive Class	F1_Score - Positive Class	Accuracy	F1_Score - macro avg
Threshold (TH): 0.5 (default)	0.57	0.26	0.36	0.89	0.65
Threshold (TH): 0.22	0.32	0.59	0.42	0.81	0.65
Threshold(TH): 0.33	0.43	0.47	0.45	0.87	0.69

Probability threshold at 0.33 was the threshold with the best trade-off between Precision and Recall: both positive class and macro-averaged F-1 scores were the greatest, whilst the accuracy was only slightly less than the default threshold (0.5).

KNN algorithm doesn't have the parameter "class_weight", therefore I tested the model applying the following resampling approaches:

- Undersampling: Edited Nearest Neighbors
- Oversampling: SMOTE (Synthetic Minority Oversampling Technique)
- Resampling: Edited Nearest Neighbors + SMOTE.

Undersampling approach: Edited Nearest Neighbors

Edited Nearest Neighbors locates the majority-class points that are missclassified (using K-Nearest Neighbors algorithm), and drops them. Edited Nearest Neighbors works as follows:

- Pick an example and compute its K nearest neighbors.
- If the example is a majority class instance and is misclassified by its K nearest neighbors, then it is removed from the dataset.
- If the example is a minority class instance and is misclassified by its K nearest neighbors, then the majority class instances among the neighbors are removed.

Edited Nearest Neighbors removes majority-class points that are ambiguous and noisy (that is majority-class points too close to minority-class points), thus creating more distinct classes.

Edited Nearest Neighbors algorithm has some hyperparameters to be tuned: the number of neighbors (k), and the strategy to use in order to exclude samples (kind_sel: "all" (all neighbours will have to agree with the samples of interest to not be misclassified), or "mode" (majority

vote of the neighbours of the samples of interest to not be misclassified). Therefore, I followed the below methodology:

- Created a Pipeline object with MinMax Scaler, Edited Nearest Neighbors (ENN), and KNN Classifier.
- Fitted the pipeline on the training set, using GridSearchCV method to find the best hyperparameters for ENN algorithm, and KNN model, and the same cross-validation approach (5 stratified shuffled splits).
- Make predictions using the test set, and compute the relevant out-of-sample metrics.

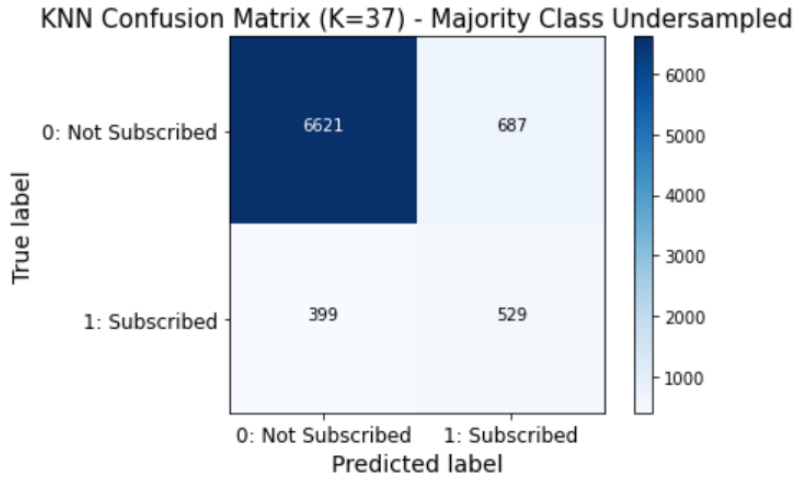
The optimum hyperparameters were:

- Edited Nearest Neighbours:
    - kind_sel: "all" -> All neighbors have to agree with the samples of interest to not be misclassified.
    - Number of neighbors: 6.
- KNN Classifier:
    - Number of Neighbors (K): 37.
    - Type of Distance: Manhattan Distance (L1).
    - Weights of Neighbors: uniform (all neighbors have same weight).

ENN algorithm reduced the number of observations as follows:

```
Number of observations before undersampling: 32,940
Number of observations after undersampling: 23,637
Number of removed observations (majority class): 9,303
```

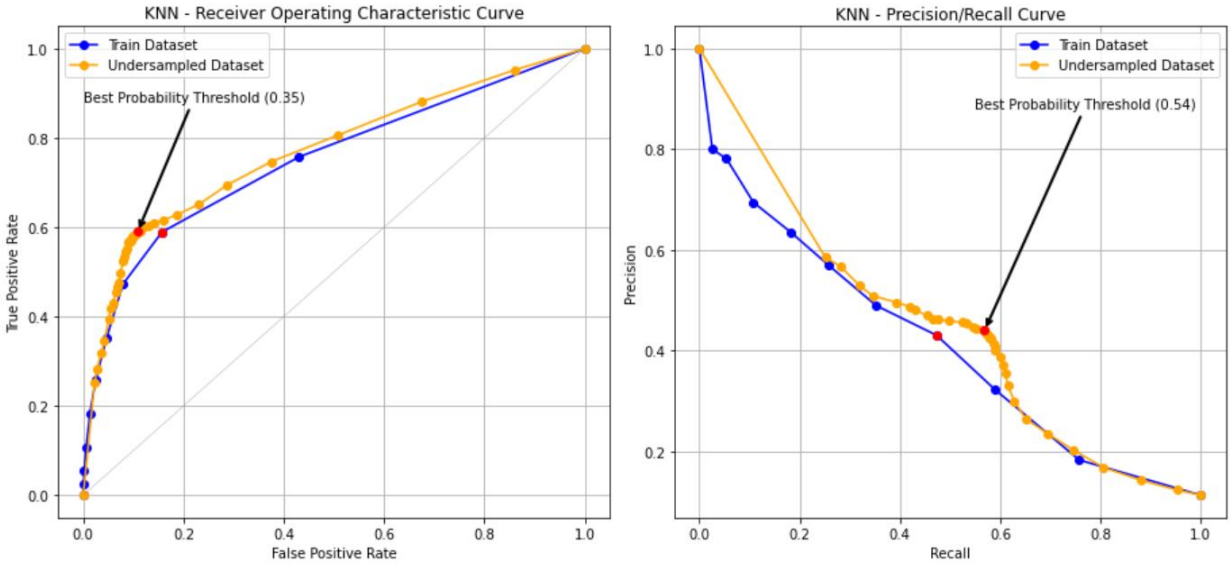
Confusion Matrix and Classification Report:



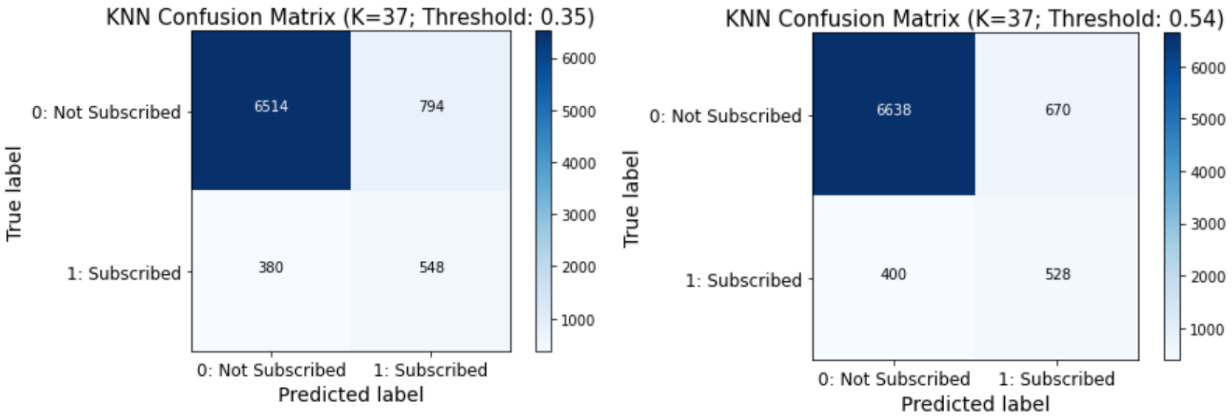
```
KNN: Classification Report (K=37) - Majority Class Undersampled
                precision    recall  f1-score   support

Not Subscribed       0.94      0.91      0.92      7308
    Subscribed       0.44      0.57      0.49       928

      accuracy                           0.87      8236
     macro avg       0.69      0.74      0.71      8236
  weighted avg       0.89      0.87      0.88      8236


ROC Area Under the Curve - Original DS:  0.75
ROC Area Under the Curve - Undersampled DS:  0.77

Precision/Recall Area Under the Curve - Original DS:  0.41
Precision/Recall Area Under the Curve - Undersampled DS:  0.45
```

Undersampling the majority class increased the model F1-Score from 0.69 to 0.71 (macro average), the Area Under the Curves, and all metrics related to the Positive Class.

ROC and Precision/Recall curves with best decision threshold:



Confusion Matrix and Classification Report - best decision thresholds:




```
KNN: Classification Report (K=37; Threshold: 0.35)
                precision    recall  f1-score   support

Not Subscribed       0.94      0.89      0.92      7308
    Subscribed       0.41      0.59      0.48       928

      accuracy                           0.86      8236
     macro avg       0.68      0.74      0.70      8236
  weighted avg       0.88      0.86      0.87      8236
```

```
KNN: Classification Report (K=37; Threshold: 0.54)
                precision    recall  f1-score   support

Not Subscribed       0.94      0.91      0.93      7308
    Subscribed       0.44      0.57      0.50       928

      accuracy                           0.87      8236
     macro avg       0.69      0.74      0.71      8236
  weighted avg       0.89      0.87      0.88      8236
```

	Precision - Positive Class	Recall - Positive Class	F1_Score - Positive Class	Accuracy	F1_Score - macro avg
Threshold (TH): 0.5 (default)	0.57	0.26	0.36	0.89	0.65
Threshold (TH): 0.22	0.32	0.59	0.42	0.81	0.65
Threshold(TH): 0.33	0.43	0.47	0.45	0.87	0.69
DS Undersampled (default TH (0.5))	0.44	0.57	0.49	0.87	0.71
DS Undersampled (TH: 0.35)	0.41	0.59	0.48	0.86	0.70
DS Undersampled (TH: 0.54)	0.44	0.57	0.50	0.87	0.71

Undersampling the Dataset, and moving the probability threshold to 0.54, made the best classifier, with the greatest positive class and macro-averaged F1-Scores, without significantly losing accuracy.

Oversampling: SMOTE (Synthetic Minority Oversampling Technique)

SMOTE (Synthetic Minority Oversampling Technique) is a technique that generates new samples by interpolation. Considering the mixed types of data in the dataset (continuous, discrete and categorical), I used SMOTENC, that is an extension of SMOTE algorithm able to deal with categorical data (SMOTE algorithm automatically takes care of discrete features).

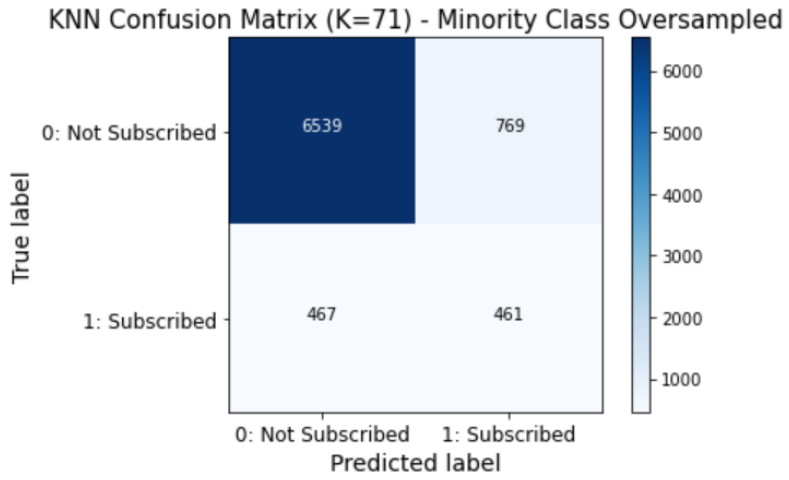
SMOTENC creates synthetic samples of the minority class, by creating new samples between the minority points and their k neighbors for continuous/discrete features (regular SMOTE), and by resampling the categorical features.

Best hyperparameters:

- Number of Neighbors (K): 71.
- Type of Distance: Manhattan Distance (L1).
- Weights of Neighbors: Distance (samples are weighted by the inverse of their distance: closer neighbors have a greater influence).

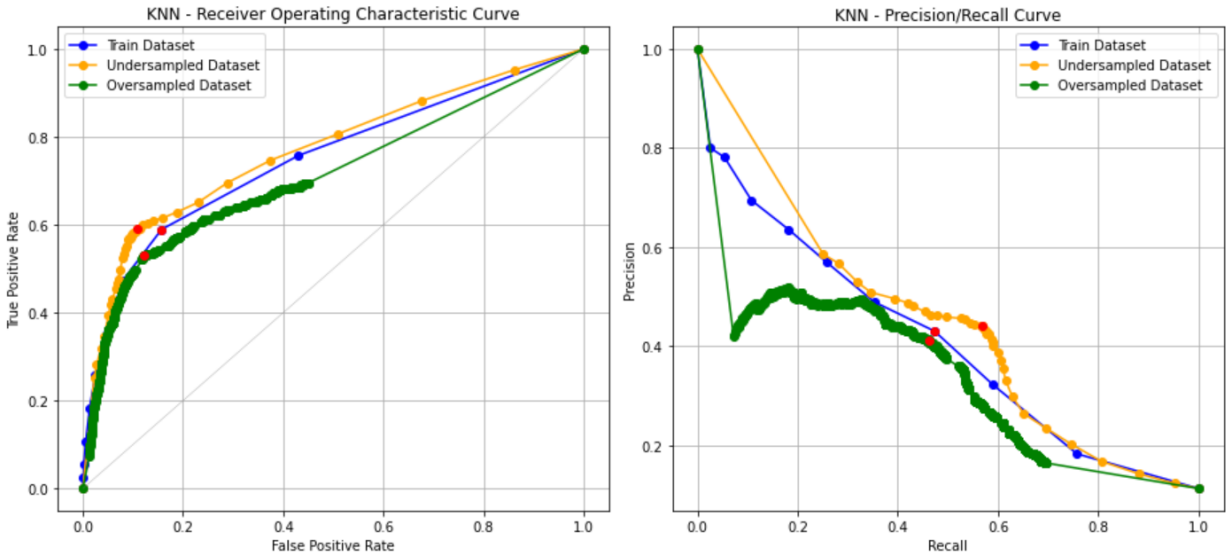
SMOTENC algorithm balanced the classes, and increased the number of observations as follows:

```
Number of observations before oversampling: 32,940
Number of observations after oversampling: 58,458
Number of synthetic samples (minority class): 25,518
```



```
KNN: Classification Report (K=71) - Minority Class oversampled
                precision    recall  f1-score   support

Not Subscribed       0.93      0.89      0.91      7308
    Subscribed       0.37      0.50      0.43       928

      accuracy                           0.85      8236
     macro avg       0.65      0.70      0.67      8236
  weighted avg       0.87      0.85      0.86      8236


ROC Area Under the Curve - Original DS:  0.75
ROC Area Under the Curve - Undersampled DS:  0.77
ROC Area Under the Curve - Oversampled DS:  0.71

Precision/Recall Area Under the Curve - Original DS:  0.41
Precision/Recall Area Under the Curve - Undersampled DS:  0.45
Precision/Recall Area Under the Curve - Oversampled DS:  0.34
```

ROC and Precision/Recall curves, with the best decision thresholds:



Oversampling the minority class didn't make a better classifier: all relevant metrics were less than the up-to-now-best-classifier (undersampled DS with threshold at 0.54).

Resampling : ENN + SMOTE

I wanted to try a last sampling technique: Resampling, that is combining undersampling (Edited Nearest Neighbours) with oversampling (SMOTE). Oversampling with SMOTE didn't produce better results compared to undersampling, so I downsampled with ENN, and added on top of

that SMOTE oversampling , using a different minority-to-majority ratio instead of the default one (0.2 instead of 0.5).

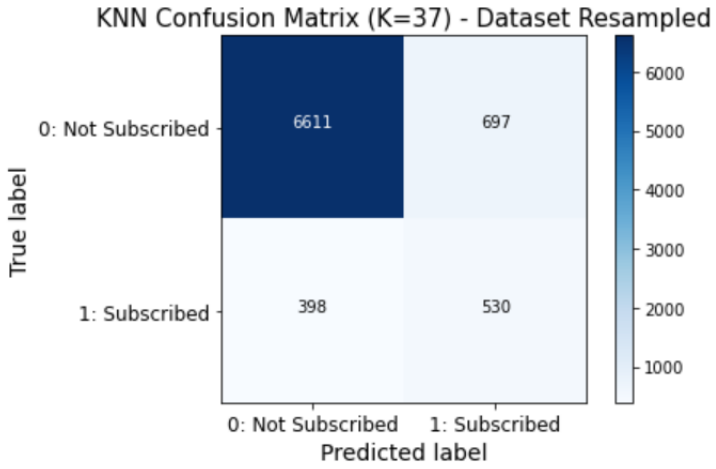
Best hyperparameters:

- Number of Neighbors (K): 37.
- Type of Distance: Manhatthan Distance (L1).
- Weights of Neighbors: Uniform

Resampling changed the number of observations as follows:

```
Number of observations before resampling: 32940
Number of removed observations: 9303
Number of synthetic samples (minority class): 274
Number of observations after resampling: 23911
Original dataset and resampled dataset delta:  9029
```

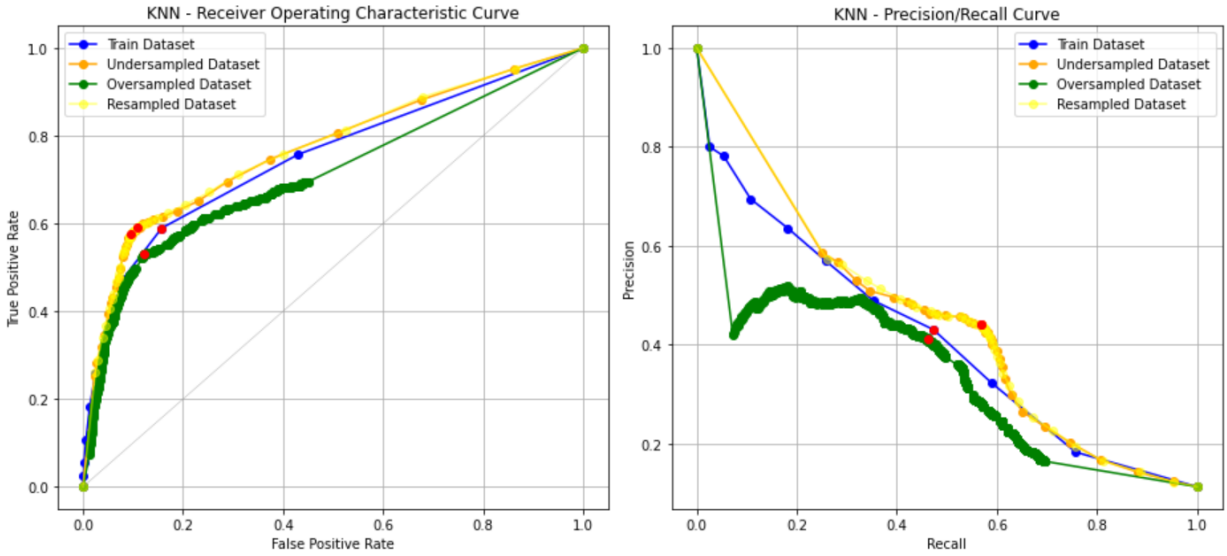
Confusion Matrix and Classification Report:




```
KNN: Classification Report (K=37) - Dataset resampled
                precision    recall  f1-score   support

Not Subscribed       0.94      0.90      0.92      7308
    Subscribed       0.43      0.57      0.49       928

      accuracy                           0.87      8236
     macro avg       0.69      0.74      0.71      8236
  weighted avg       0.89      0.87      0.87      8236


ROC Area Under the Curve - Original DS:  0.75
ROC Area Under the Curve - Undersampled DS:  0.77
ROC Area Under the Curve - Oversampled DS:  0.71
ROC Area Under the Curve - Resampled DS:  0.77
```

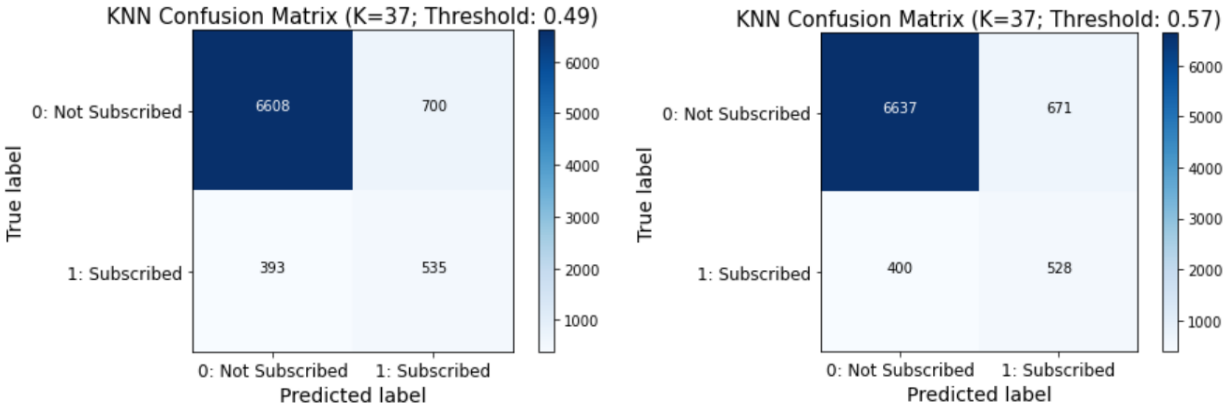
```
Precision/Recall Area Under the Curve - Original DS:  0.41
Precision/Recall Area Under the Curve - Undersampled DS:  0.45
Precision/Recall Area Under the Curve - Oversampled DS:  0.34
Precision/Recall Area Under the Curve - Resampled DS:  0.45
```

ROC and Precision/Recall curves with best decision threshold:



Resampling the dataset made approximately the same performance as Undersampling, with the best probability thresholds of the 2 models very close to each other.

Confusion Matrix and Classification Report - best decision thresholds, Resampled dataset:



```
KNN: Classification Report (K=37; Threshold: 0.49)
                precision    recall  f1-score   support

Not Subscribed       0.94      0.90      0.92      7308
    Subscribed       0.43      0.58      0.49       928

      accuracy                           0.87      8236
     macro avg       0.69      0.74      0.71      8236
  weighted avg       0.89      0.87      0.88      8236


KNN: Classification Report (K=37; Threshold: 0.57)
                precision    recall  f1-score   support

Not Subscribed       0.94      0.91      0.93      7308
    Subscribed       0.44      0.57      0.50       928

      accuracy                           0.87      8236
     macro avg       0.69      0.74      0.71      8236
  weighted avg       0.89      0.87      0.88      8236
```

Report table – models with greatest macro-averaged F1-score:

	Precision - Positive Class	Recall - Positive Class	F1_Score - Positive Class	Accuracy	F1_Score - macro avg
DS Undersampled (default TH (0.5))	0.44	0.57	0.49	0.87	0.71
DS Undersampled (TH: 0.54)	0.44	0.57	0.50	0.87	0.71
DS Resampled (default TH (0.5))	0.43	0.57	0.49	0.87	0.71
DS Resampled (TH: 0.49)	0.43	0.58	0.49	0.87	0.71
DS Resampled (TH: 0.57)	0.44	0.57	0.50	0.87	0.71

KNN Best Classifier:

- Undersampling method - Edited Nearest Neighbors:
  - kind_sel: "all" (all neighbors will have to agree with the samples of interest to not be misscassified).
  - n_neighbors: 6.
- KNN Classifier:
  - Number of Neighbors (K): 37.
  - Type of Distance: Manhatthan Distance (L1).
  - Weights of Neighbors: uniform (all neighbors have same weight).
  - Probability Threshold: 0.54

2.3.2. Classifier 2: Random Forest.

Random Forest is a non-parametric model, therefore there's no need to make any assumptions on the data distribution, and data don't need any pre-processing transformations, like scaling.

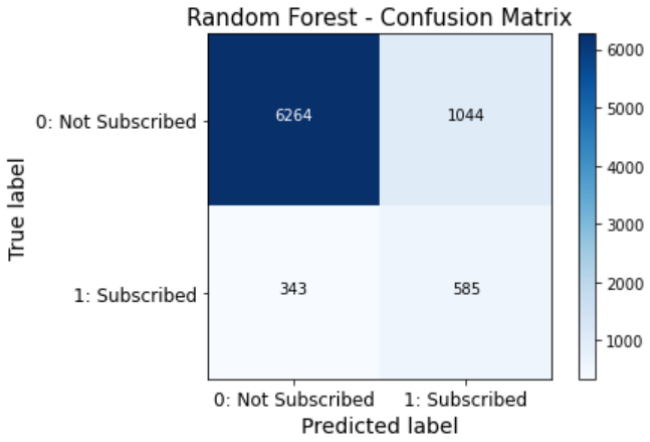
I followed the same methodology as per the KNN model:

- Fitted the model on the training set, using GridSearchCV to find the best hyperparameters with stratified Cross Validation technique (5 stratified shuffled splits).
- Calculated out-of-sample metrics using the test set to make predictions (confusion matrix, precision, recall, F1, Receiver Operating Characteristic Curve, precision/recall curve, and Area Under the Curves).

Best Random Forest hyperparameters:

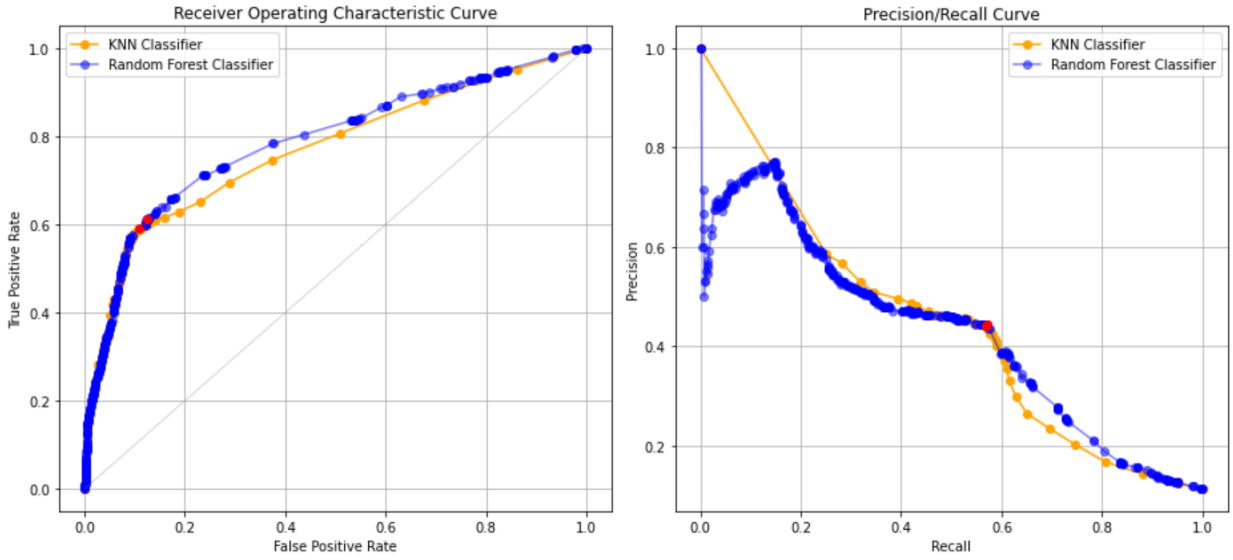
- Class weight: balanced (automatically adjust weights inversely proportional to class frequencies).
- Criterion to compute the information gain: Gini.
- Max depth of decision trees: 5.
- Number of trees: 10.

Confusion Matrix and Classification Report:



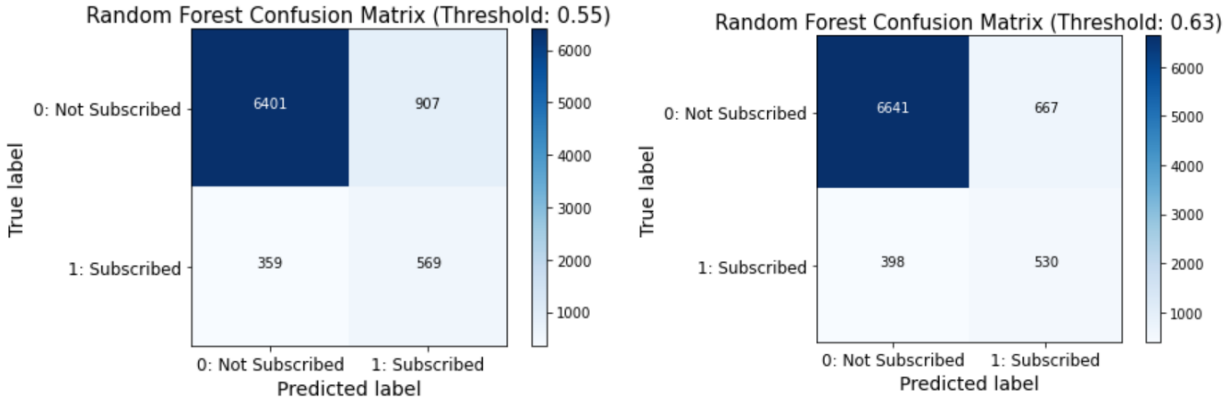

```
Random Forest: Classification Report
                precision    recall  f1-score   support

Not Subscribed       0.95      0.86      0.90      7308
    Subscribed       0.36      0.63      0.46       928

      accuracy                           0.83      8236
     macro avg       0.65      0.74      0.68      8236
  weighted avg       0.88      0.83      0.85      8236


ROC Area Under the Curve - KNN Best Classifier:  0.77
ROC Area Under the Curve - Random Forest:  0.79
```

```
Precision/Recall Area Under the Curve - KNN Best Classifier:  0.41
Precision/Recall Area Under the Curve - Random Forest:  0.42
```



KNN and Random Forest best probability thresholds seemed to be very close to each other.

Confusion Matrix and Classification Report – Random Forest best decision thresholds.




```
Random Forest: Classification Report (Threshold: 0.55)
              precision    recall  f1-score   support

Not Subscribed       0.95      0.88      0.91      7308
    Subscribed       0.39      0.61      0.47       928

      accuracy                           0.85      8236
     macro avg       0.67      0.74      0.69      8236
  weighted avg       0.88      0.85      0.86      8236
```

```
Random Forest: Classification Report (Threshold: 0.63)
                precision    recall  f1-score   support

Not Subscribed       0.94      0.91      0.93      7308
    Subscribed       0.44      0.57      0.50       928

      accuracy                           0.87      8236
     macro avg       0.69      0.74      0.71      8236
  weighted avg       0.89      0.87      0.88      8236
```

	Precision - Positive Class	Recall - Positive Class	F1_Score - Positive Class	Accuracy	F1_Score - macro avg
KNN - Best Classifier	0.44	0.57	0.50	0.87	0.71
Random Forest - TH: 0.5	0.36	0.63	0.46	0.83	0.68
Random Forest - TH: 0.55	0.39	0.61	0.47	0.85	0.69
Random Forest - TH: 0.63	0.44	0.57	0.50	0.87	0.71

Random Forest best classifier:

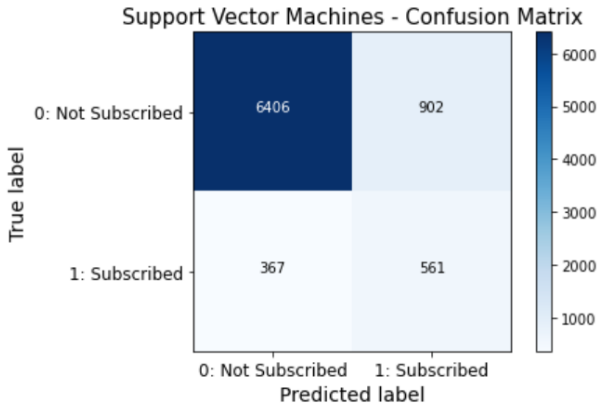
- Number of estimators/trees: 10
- Criterion to measure the Information Gain: Gini
- Maximum depth of Decision Trees: 5
- Class Weight: balanced weights
- Probability Threshold: 0.63

Random Forest and KNN Best Classifiers performed very similarly.

2.3.3. Classifier 3: Support Vector Machines.

Best Support Vector Machines hyperparameters:

- Kernel used to map data to higher dimensions: Radial Basis Function.
- Regularization Parameter: 1.
- Class weight: Balanced.



```
Support Vector Machines: Classification Report
                precision    recall  f1-score   support

Not Subscribed       0.95      0.88      0.91      7308
    Subscribed       0.38      0.60      0.47       928

      accuracy                           0.85      8236
     macro avg       0.66      0.74      0.69      8236
  weighted avg       0.88      0.85      0.86      8236
```

Report Table:

	Precision - Positive Class	Recall - Positive Class	F1_Score - Positive Class	Accuracy	F1_Score - macro avg
KNN - Best Classifier	0.44	0.57	0.50	0.87	0.71
RF - Best Classifier	0.44	0.57	0.50	0.87	0.71
Support Vector Machines	0.38	0.60	0.47	0.85	0.69

Support Vector Machine was able to catch more examples belonging to the minority class (Recall on the positive class is the greatest), but at cost of Precision: overall K-Nearest Neighbors and Random Forest, optimized with best hyperparameters and probability-thresholds, performed better, with greater F-1 scores and accuracy.

Support Vector Machine doesn't output probabilities, therefore it wasn't possible moving the decision threshold to optimize the classifier.

2.3.4. Classifier 4: Ensemble Model - Stacking Classifier.

As last Classifier I wanted to stack the 3 trained model (base estimators), and train a meta-classifier (final estimator) using the outputs of the base estimators as inputs for the final estimator: this is an Ensemble Model called "Stacking".

The base estimators (KNN, Random Forest, and SVM) were trained with the (already found) best hyperparameters, and I fitted the Stacking Classifier on the training set, using GridSearchCV to test what meta-classifier and parameters worked best.

Hyperparameter grid was as follows:

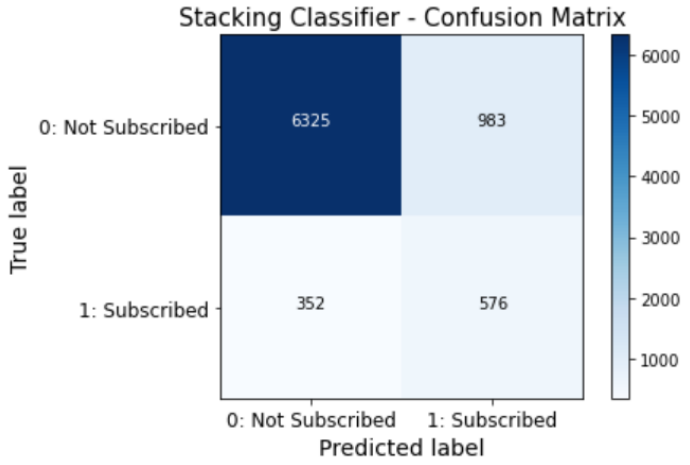
- Meta-classifiers: Logistic Regression, Decision Tree
    - Hyperparameters for Logistic Regression:
        - Type of penalty for regularization: L2 penalty, no penalty
        - Regularization Parameter - C: 1.0, 0.5, 0.1

        - Class weight: original weights, balanced weights
    - Hyperparameters for Decision Tree:
        - Criterion for information gain: Gini, Entropy
        - Max depth for decision Trees: None, 2
        - Class weight: original weights, balanced weights

The final-estimator best hyperparameters are:

- Logistic Regression.
- Penalty: L2 penalty.
- C - Regularization Parameter: 0.1.
- Class weight: balanced weights.

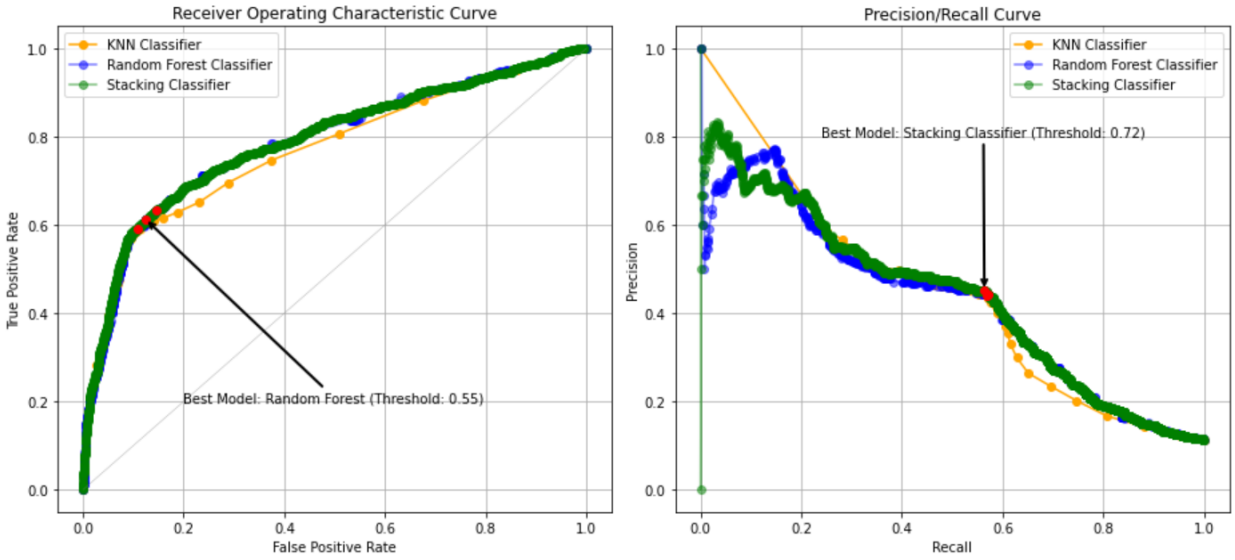
Confusion Matrix and Classification Report:




```
Stacking Classifier: Classification Report
                precision    recall  f1-score   support

Not Subscribed       0.95      0.87      0.90      7308
    Subscribed       0.37      0.62      0.46       928

      accuracy                           0.84      8236
     macro avg       0.66      0.74      0.68      8236
  weighted avg       0.88      0.84      0.85      8236


ROC Area Under the Curve - KNN Best Classifier:  0.77
ROC Area Under the Curve - Random Forest Best Classifier:  0.79
ROC Area Under the Curve - Stacking Classifier: 0.79
```

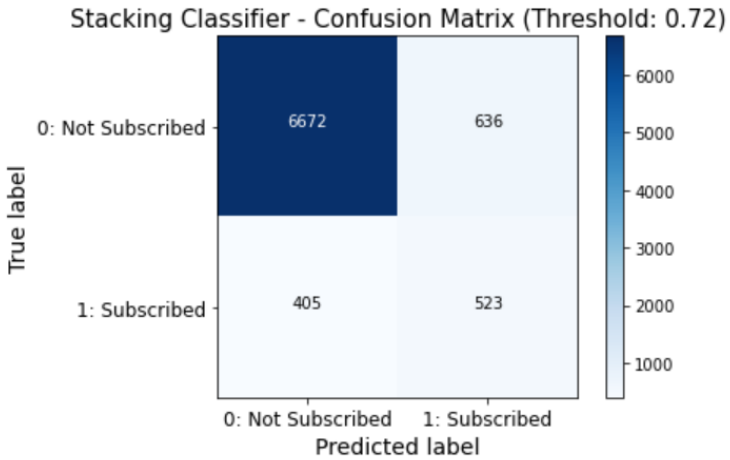
```
Precision/Recall Area Under the Curve - KNN Best Classifier:  0.41
Precision/Recall Area Under the Curve - Random Forest Best Classifier:0.42
Precision/Recall Area Under the Curve - Stacking Classifier: 0.43
```

ROC and Precision/Recall curves with best decision threshold:




```
Random Forest is the model with the greatest Youden's J statistic.
Best probability threshold:  0.5472990213846068

Stacking Classifier is the model with the greatest F1-Score.
Best probability threshold:  0.7153119838862827
```

Confusion Matrix and Classification Report for Stacking Classifier - best probability threshold:



```
Stacking Classifier: Classification Report (Threshold: 0.72)
                precision    recall  f1-score   support

Not Subscribed       0.94      0.91      0.93      7308
    Subscribed       0.45      0.56      0.50       928

      accuracy                           0.87      8236
     macro avg       0.70      0.74      0.71      8236
  weighted avg       0.89      0.87      0.88      8236
```

Report Table:

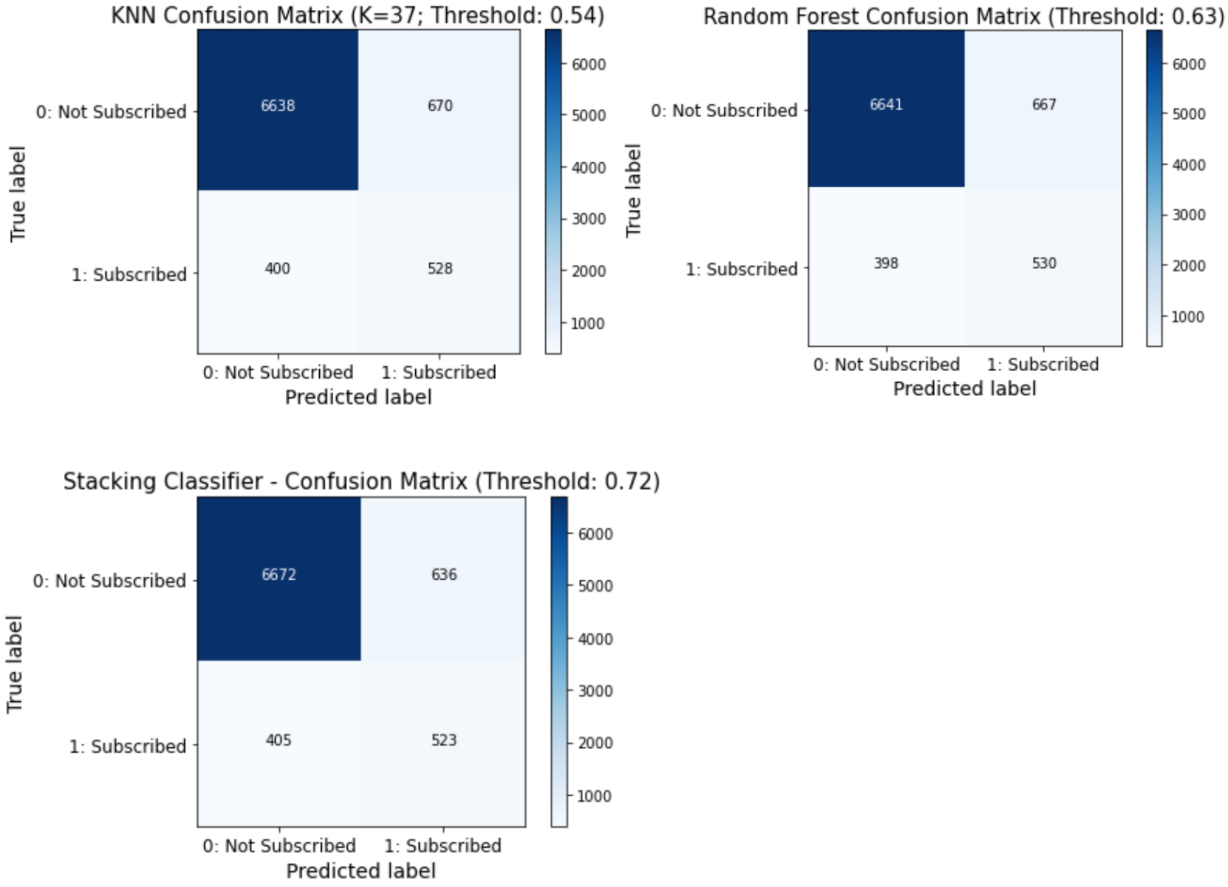
	Precision - Positive Class	Recall - Positive Class	F1_Score - Positive Class	Accuracy	F1_Score - macro avg
KNN - Best Classifier	0.44	0.57	0.50	0.87	0.71
RF - Best Classifier	0.44	0.57	0.50	0.87	0.71
Support Vector Machines	0.38	0.60	0.47	0.85	0.69
SC - Best Classifier	0.45	0.56	0.50	0.87	0.71

K-Nearest Neighbors, Random Forest, and Stacking Classifier, optimized with best hyperparameters, class sampling, and best decision threshold, have very similar performances. Support Vector Machine is the classifier which performs the worst.

Best Stacking Classifier:

- Meta-Classifier: Logistic Regression.
  - Penalty: L2 penalty.
  - C - Regularization Parameter: 0.1.
  - Class weight: balanced weights.

3. Results.



K-Nearest Neighbors, Random Forest, and Stacking Classifier were the best classifiers, with performance metrics very close to each other. Stacking Classifier, amongst the 3 models, had the greatest precision, but the lowest recall. From a marketing perspective, a marketing performance predictive model should predict as many correct successful examples (observations belonging to the positive class) as possible; therefore, considering the business priorities, KNN and Random Forest fitted the business purpose better.
KNN and Random Forest had almost identical out-of-sample performance metrics; although Random Forest slightly exceeds KNN performance by correctly predicting 2 more positive examples.

I selected Random Forest as the classifier that best suited the project purpose, optimized with the following hyperparameters:

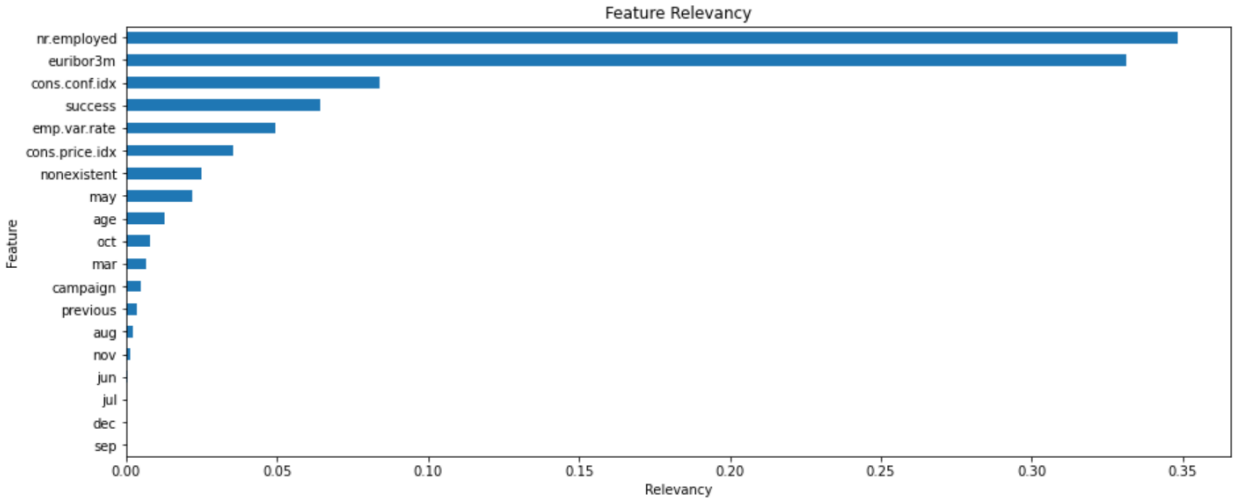
- Number of estimators/trees: 10
- Criterion to measure the information gain: Gini
- Maximum depth of Decision Trees: 5
- Class Weight: balanced weights
- Probability Threshold: 0.63

4. Discussion.

Following the mentioned methodology (class sampling, decision threshold moving, performance benchmarking…), I managed to increase model F1-score (macro-average) from 0.65 (first KNN trained model) to 0.71 (Random Forest), measured on an out-of-sample set.

Random Forest was the classifier that best suited the project purpose (K-Nearest Neighbors, with Edited Nearest Neighbors undersampling technique, produced comparable results).

Regarding the relevance of features, I looked into each feature importance with regard to target variable prediction:



The 5 most relevant features include 4 socio-economic variables, nr.employed (number of employees), euribor3m (euribor 3 month rate), cons.conf.idx (consumer confidence index), and emp.var.rate (employement variation rate), and 1 marketing-related variable (whether if the previous marketing campaign was successful or not).

nr.employed and euribor3m are the features with, by far, the greatest importance for predicting the success of the marketing campaign under consideration.

All other features have a relevancy score less than 0.05.

5. Conclusion.

In this project I built a Classification model to predict whether an instance of a telemarketing campaign is successful or not. The main focus was on **prediction**, therefore my main purpose was to build a predictive model able to correctly predict as many positive example as possible, without losing too much accuracy.

A major problem was that the classes were quite imbalanced, so I had to consider following performance metrics (besides the Accuracy score):

- Precision, Recall, and F1-score for Positive Class
- Model F1-score (macro average)

I dealt with the imbalanced classes in 2 ways:

- Whereas available, I used the "class_weight" hyperparameter.
- If "class_weight" wasn't available as model parameter, I applied below sampling approaches:
    - Undersampling the majority class with **Edited Nearest Neighbors**
    - Oversampling the minority class with **SMOTE (Synthetic Minority Oversampling Technique) algorithm**
    - Resampling the whole dataset combining **Edited Nearest Neighbors and SMOTE.**

To further improve classifier performance, I plotted ROC and Precision/Recall curves, check for the best decision threshold, and compared different model performances (which were indeed quite similar).

Quality of data was good: I found very few duplicates and wrongly recorded data, no missing values, and no outliers.

The selected model (Random Forest with probability threshold at 0.63) is able to correctly predict more than half of observations belonging to the positive class, with an overall accuracy of 87%. Therefore, the model's performance can still improve, specially with regard to minority class predictions. For this purpose, **additional data is needed to train the model on a larger dataset, and capture the still-not-seen underlying relationships between variables.**

6. Appendix.

Reference Notebook Link - GitHub:

https://github.com/SebastianoDenegri/classifier_marketing_campaign_success/blob/main/Marketing%20Campaigns-updated_version.ipynb

Decision Tree (bootstrap sample) from Random Forest - GitHub:

https://raw.githubusercontent.com/SebastianoDenegri/classifier_marketing_campaign_success/main/decision_tree_from_rf.png