Machine Learning Engineer Nanodegree

Using Supervised Classification Algorithms to Predict Bank Term Deposit Subscription

Fabiano Shoji Yoschitaki
August 25th, 2018

I. Definition

Project Overview

In order to promote products and services, financial institutions like banks generally run marketing campaigns using two approaches [1]: 1) mass campaigns, which targets general indiscriminate public, broadcasting the same message to different customers and 2) directed marketing, which targets specific contacts, creating a directing relationship to customers.

Banks which run marketing campaigns following the first approach have had their campaigns' performance reduced over time, having less than 1% of positive responses [2]. On the other hand, marketing campaigns which follow the second approach have shown better results compared to the first [3]. For this reason, banks are more likely to spend their budget on directed marketing campaigns than on inefficient mass campaigns.

The personal reason to work on this domain background comes from the fact that I've worked on a project related to a bank company with the goal to offer the most coherent products to its customers based on their characteristics. I believe that having applied machine learning techniques could have helped us to get better response rates.

Problem Statement

Given the Bank Marketing dataset [4], which is related to direct marketing campaigns of a Portuguese bank institution, a supervised binary classification model has to be created and trained with the objective of predicting whether or not a client will subscribe to a term deposit.

Metrics

The evaluation metrics that will be considered in this project are: Accuracy and F1-Score. Acuracy is an appropriate metric for supervised classification problems, considering both correct and incorrect classifications in its formula (Figure 1):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 1.1 - Accuracy formula

F1-Score metric, also known as balanced F-score or F-measure, is a metric which can be interpreted as the harmonic average of both precision and recall metrics (Figure 2). This metric was considered due to the class imbalance that our dataset presents (approximately 88% 'no' vs 12% 'yes).

$$\text{F1-score} = \frac{2 \text{ x Precision x Recall}}{\text{Precision} + \text{Recall}}$$

Figure 1.2 - F1-score formula

Precision is the number of TP divided by the number of TP plus the number of FP and recall is the number of TP divided by the number of TP plus the number of FN (Figure 3).

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

Figure 1.3 - Precision and Recall

Where:

- **TP**: True Positive, in our case a person who subscribed a term deposit and is correctly classified.
- **TN**: True Negative, in our case a person who didn't subscribe a term deposit and is correctly classified.
- **FP**: False Positive, in our case a person who didn't subscribe a term deposit and is incorrectly classified as having subscribed a term deposit.
- **FN**: False Negative, in our case a person who subscribed a term deposit and is incorrectly classified as not having subscribed a term deposit.

II. Analysis

Data Exploration

The dataset chosen for this project is related to direct marketing campaigns based on phone calls of a Portuguese banking institution. It was obtained by exploring the University of California Irvine's Machine Learning Repository [5]. The dataset file which will be used is the bank-full.csv and it contains 45211 instances with 17 columns each. The last column is the target label: whether or not the person subscribed a term deposit. The probability for

the label 'yes' (did a term deposit) is approximately 12% and for 'no' (didn't do a term deposit) is approximately 88%. The description of the columns follow:

- Bank client features:
  - **age**: the age of the client (numeric).
  - **job**: the type of job of the client (categorical). Possible values: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown'.
  - **marital**: the marital status of the client (categorical). Possible values: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed.
  - **education**: the education level of the client (categorical). Possible values: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown'.
  - **default**: whether or not the client has credit in default (categorical). Possible values: 'no','yes','unknown'.
  - **balance**: average yearly balance in Euros (numeric).
  - **housing**: whether or not the client has housing loan (categorical). Possible values: 'no','yes','unknown'.
  - **loan**: whether or not the client has personal loan (categorical). Possible values: 'no','yes','unknown'.

- Features related with the last contact of the current campaign:
  - **contact**: contact communication type (categorical). Possible values: 'cellular','telephone'.
  - **day**: last contact day of the month (numeric).
  - **month**: last contact month of year (categorical). Possible values: 'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov', 'dec'.
  - **duration**: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

- Other features:

- **campaign**: number of contacts performed during this campaign and for this client (numeric, includes last contact).
- **pdays**: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted).
- **previous**: number of contacts performed before this campaign and for this client (numeric).
- **poutcome**: outcome of the previous marketing campaign (categorical). Possible values: 'failure','nonexistent','success'.

- Target label:
  - **y**: whether or not the client subscribed to a term deposit (categorical). Possible values: 'yes', 'no'.

Displaying the first ten rows of the dataset below (Figure 4):

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
35	management	married	tertiary	no	231	yes	no	unknown	5	may	139	1	-1	0	unknown	no
28	management	single	tertiary	no	447	yes	yes	unknown	5	may	217	1	-1	0	unknown	no
42	entrepreneur	divorced	tertiary	yes	2	yes	no	unknown	5	may	380	1	-1	0	unknown	no
58	retired	married	primary	no	121	yes	no	unknown	5	may	50	1	-1	0	unknown	no
43	technician	single	secondary	no	593	yes	no	unknown	5	may	55	1	-1	0	unknown	no

Figure 2 - Ten first rows of the dataset

Exploratory Visualization

The Figure 1 shows the imbalance of the target classes 'yes - subscribed' and 'no - didn't subscribed' in our dataset. As we can see, the dataset happens to be very imbalanced: clients who subscribed to a term deposit represent only 11.70% of the size.

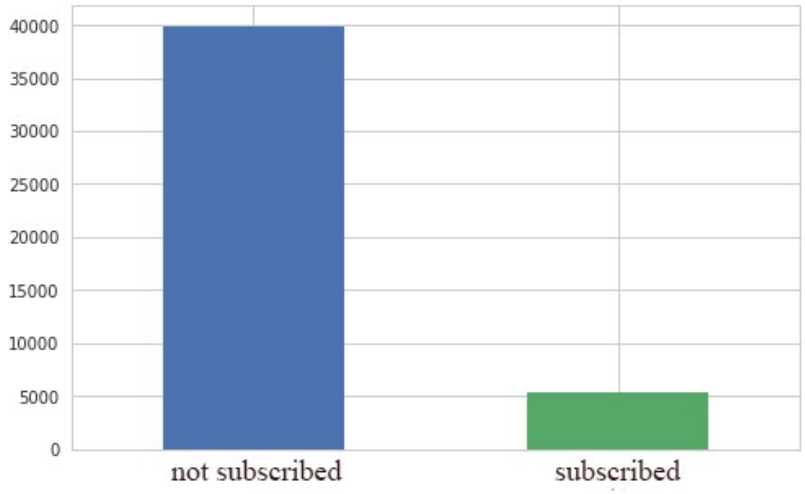


Figure 3 - Clients who subscribed to a term deposit vs clients who didn't subscribe

This age histogram shows that most of the clients of the dataset are between the ages of 30 and 40.

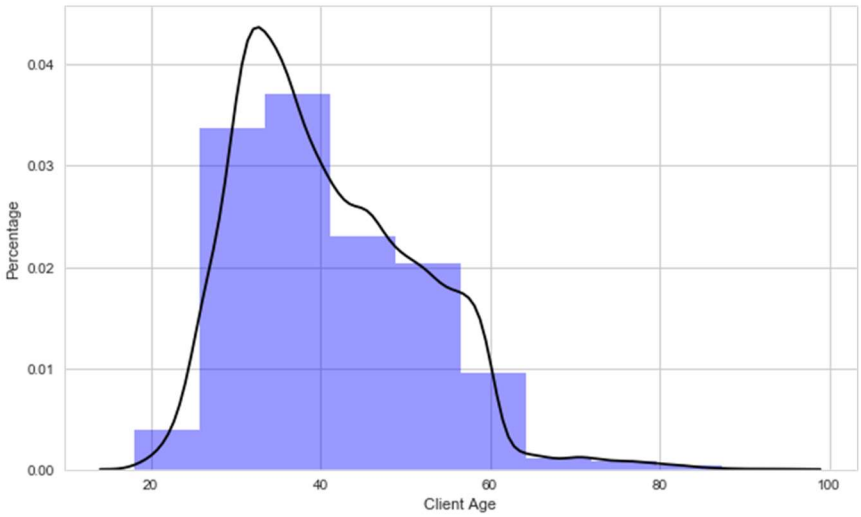


Figure 4.1 - Clients age distribution

This comparison image shows that clients between the ages of 60 and 80 tend to subscribe to a term deposit. Few clients at this age interval didn't subscribe to a term deposit.

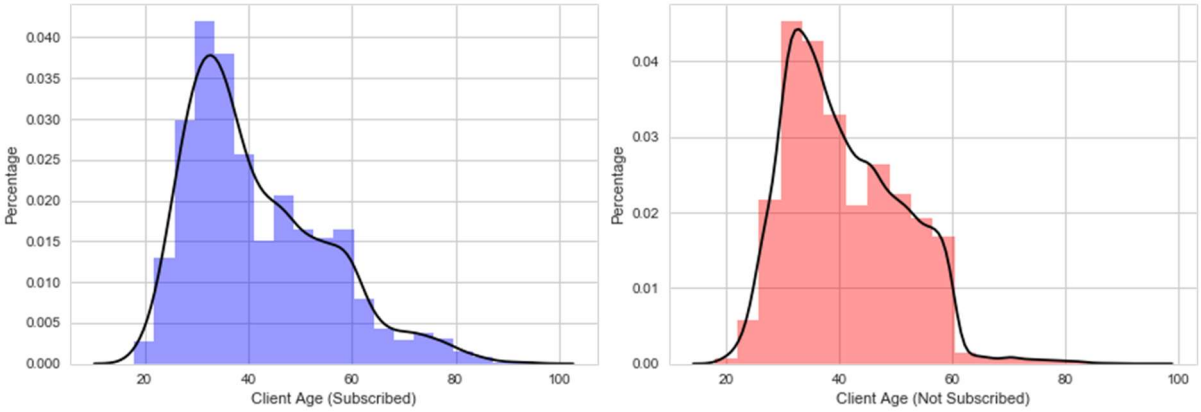


Figure 4.2 - Clients age distribution

This image shows that blue-collar, management and technician are the most common clients related jobs.

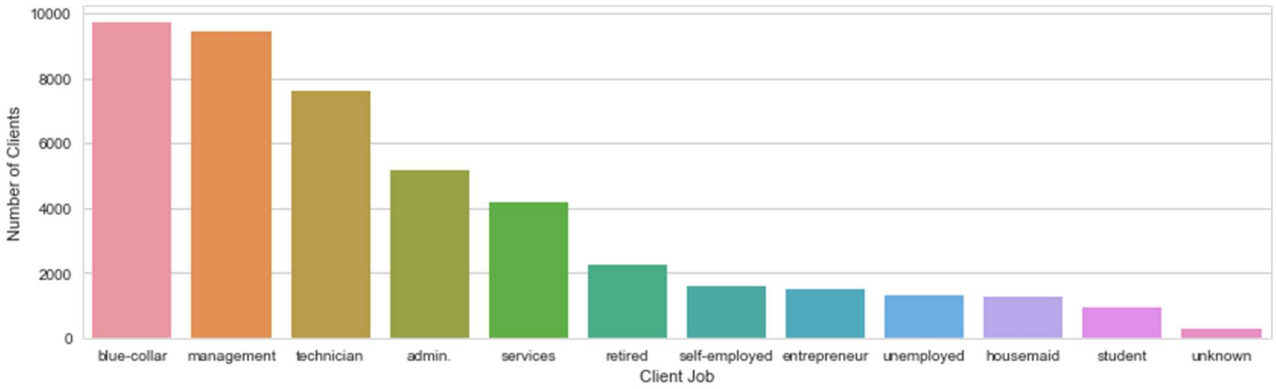


Figure 5.1 - Clients job distribution

This comparison image shows that clients related to blue-collar jobs tend not to subscribe to a term deposit, while retired and students are more propitious to subscribe.

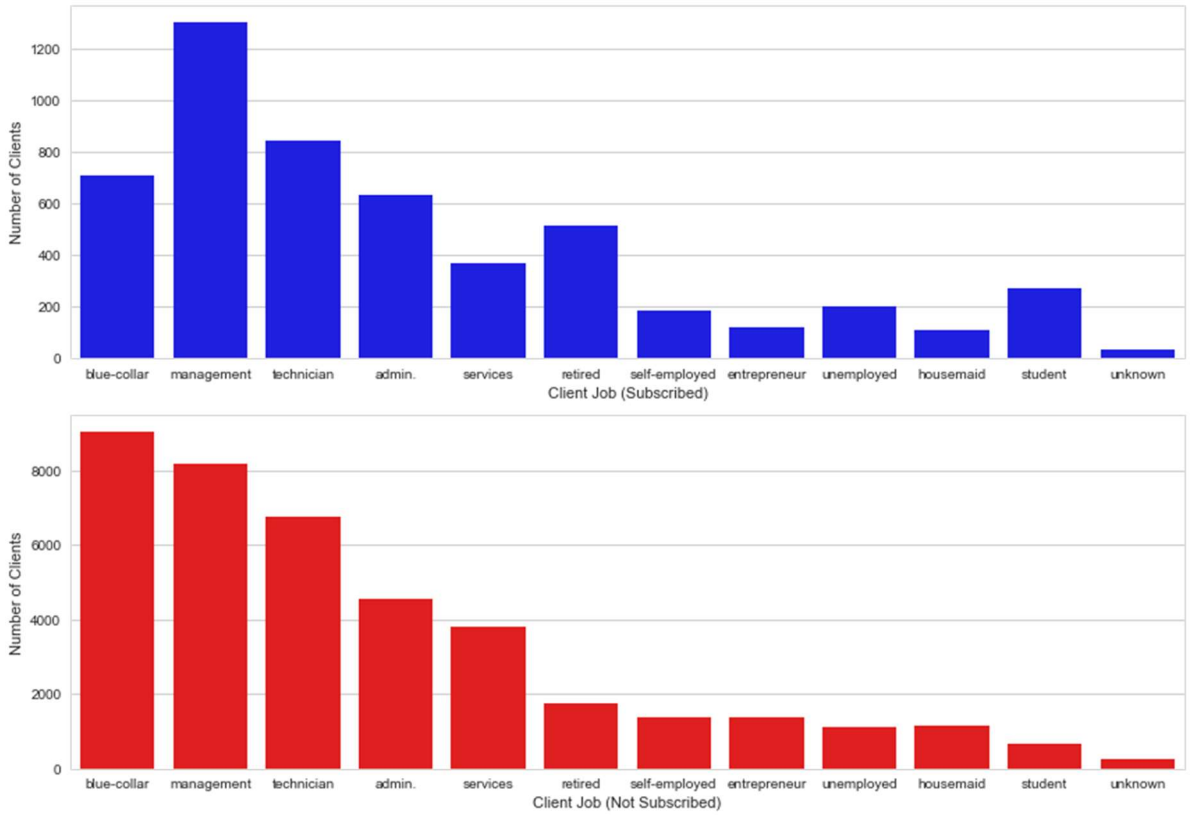


Figure 5.2 - Clients job distribution

This image shows that married clients are more common in the dataset.

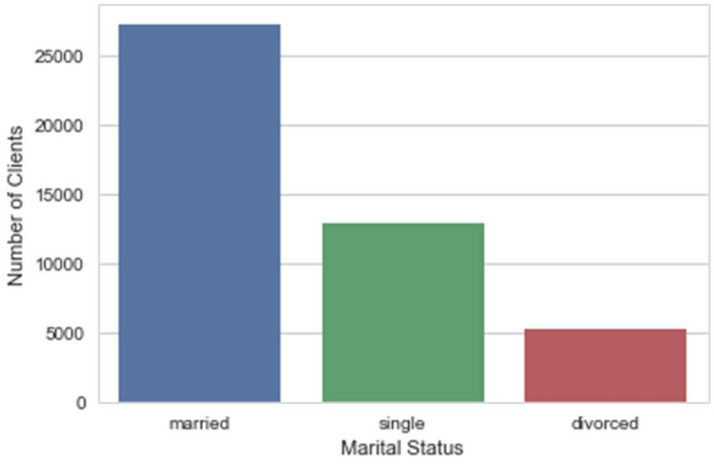


Figure 6.1 - Clients marital status distribution

This comparison image shows that single clients are more propitious to subscribe to a term deposit.

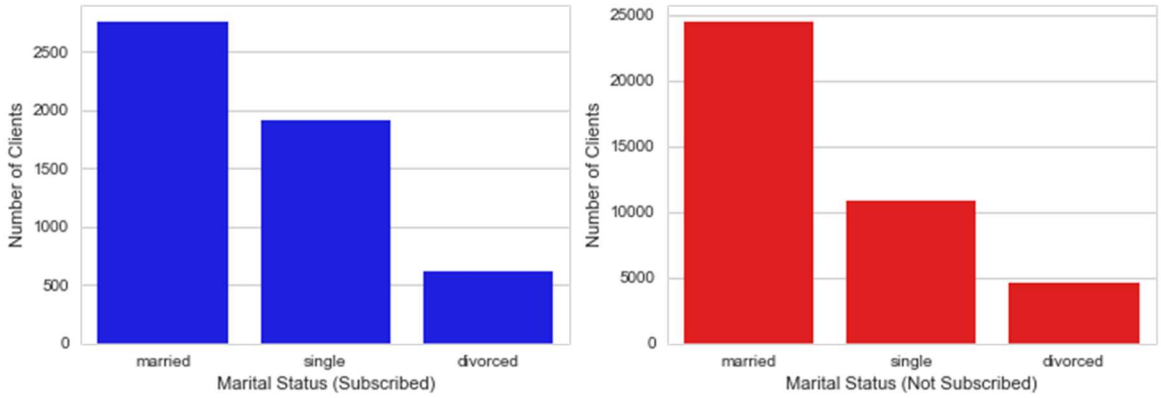


Figure 6.2 - Clients marital status distribution

This image shows that secondary education level clients are more common in the dataset.

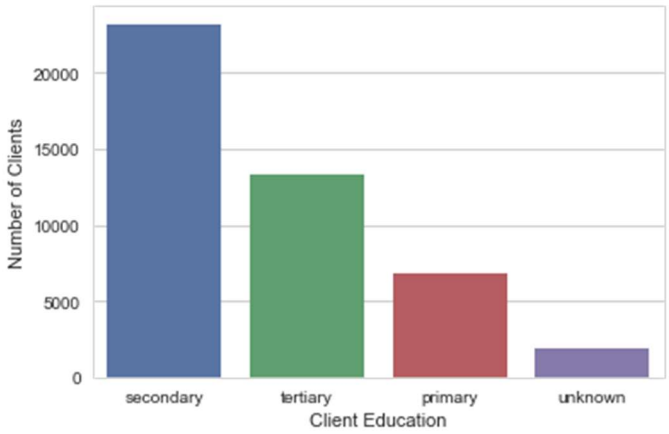


Figure 7.1 - Clients education distribution

This comparison image shows that tertiary education level clients are more propitious to subscribe to a term deposit.

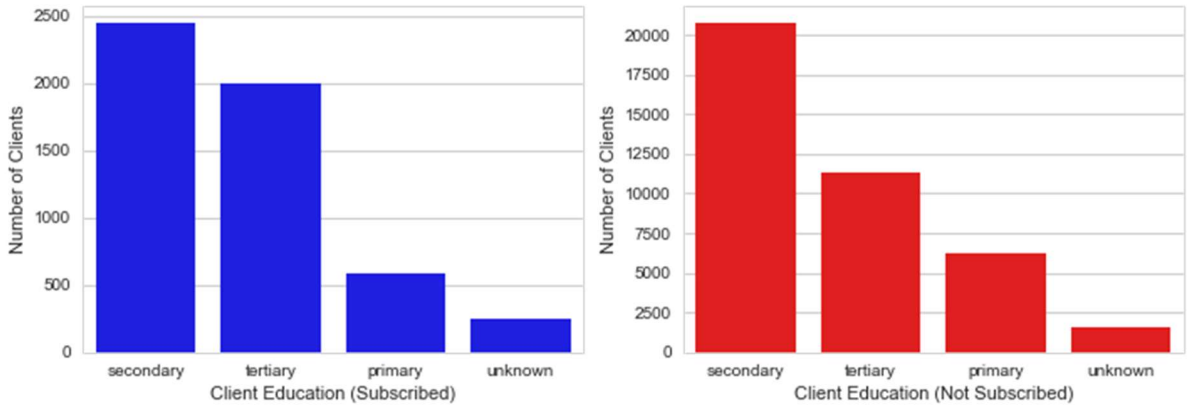


Figure 7.2 - Clients education distribution

This image shows that almost all clients don't have credit in default.

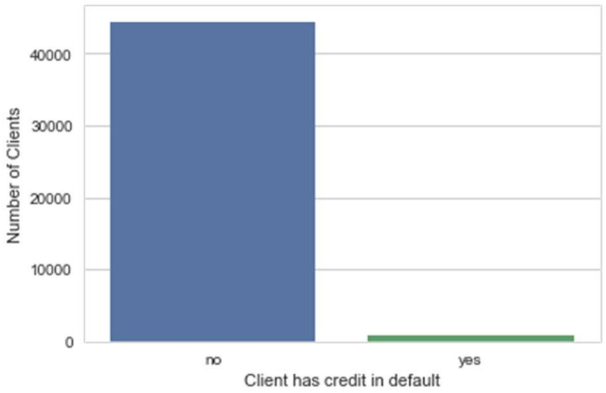


Figure 8.1 - Clients have credit in default distribution

This comparison image shows that there's no relevant difference between clients who subscribed vs clients who didn't regarding to whether the client has credit in default.

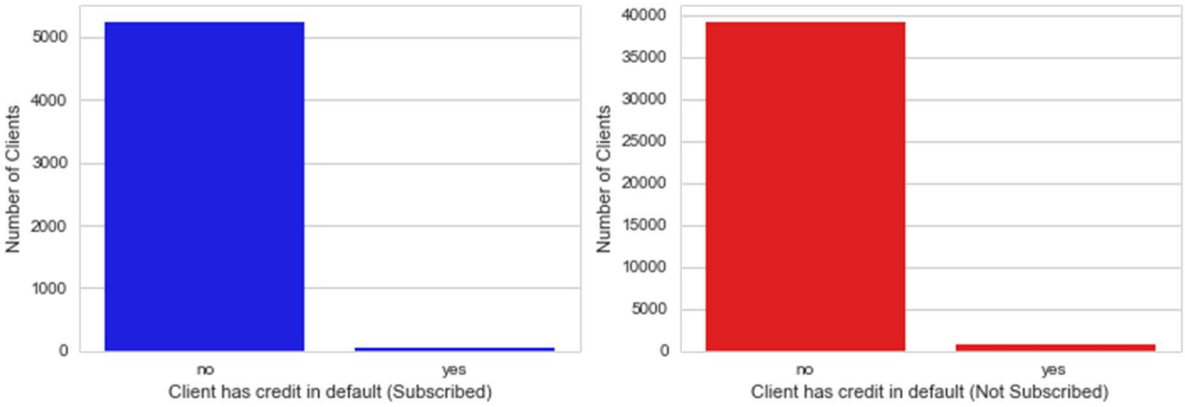


Figure 8.2 - Clients have credit in default distribution

This image shows that the majority of the clients have a housing loan.

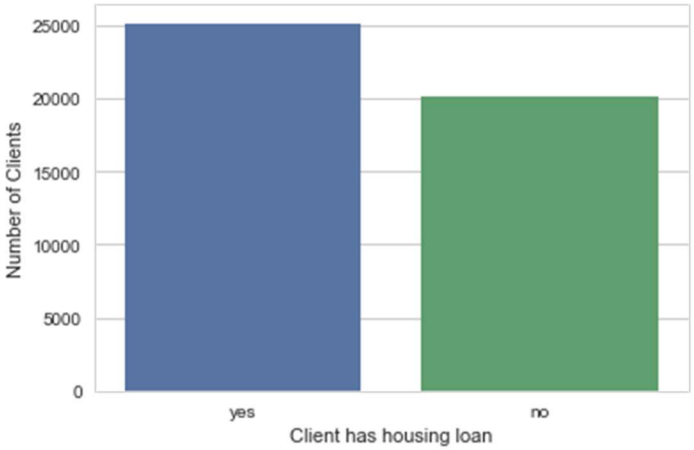


Figure 9.1 - Clients have housing loan distribution

This comparison image shows clients who don't have a housing loan are more propitious to subscribe to a term deposit.

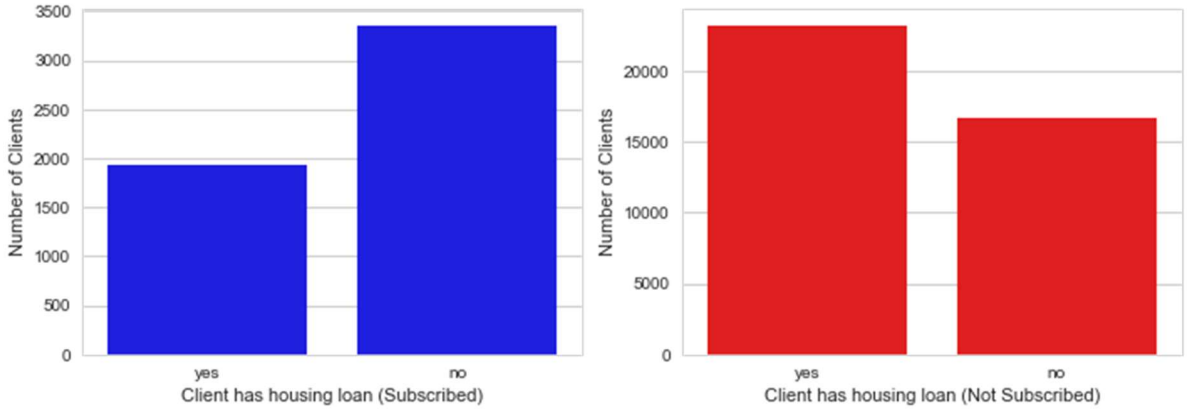


Figure 9.2 - Clients have housing loan distribution

This image shows that the majority of the clients have a personal loan.

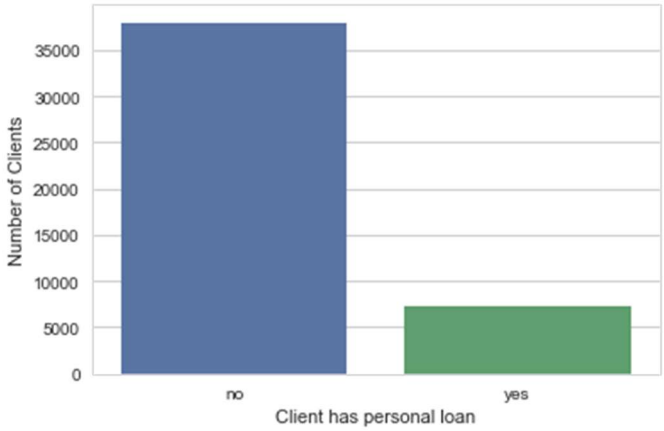


Figure 10.1 - Clients have personal loan distribution

This comparison image shows that clients who don't have personal loan are slightly more propitious to subscribe to a term deposit.

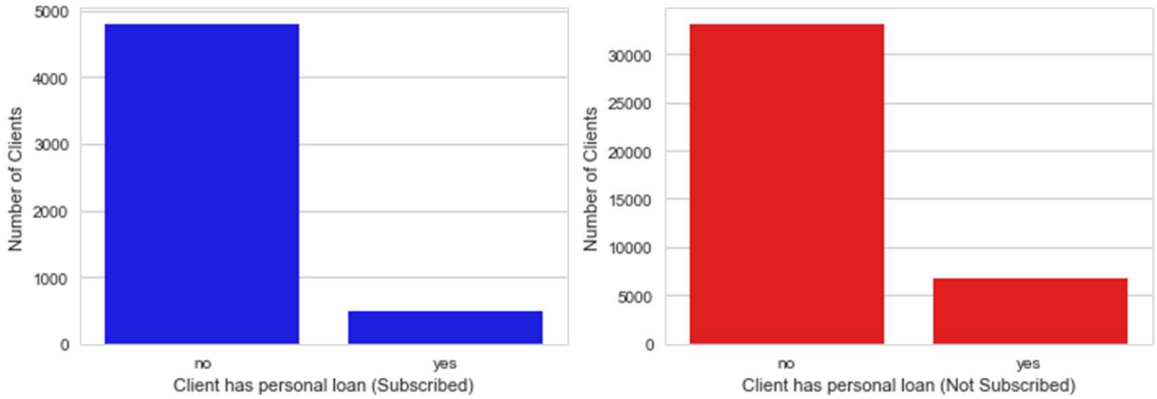


Figure 10.2 - Clients have personal loan distribution

This image shows that may is the month that the majority of the last contacts with the clients are made.

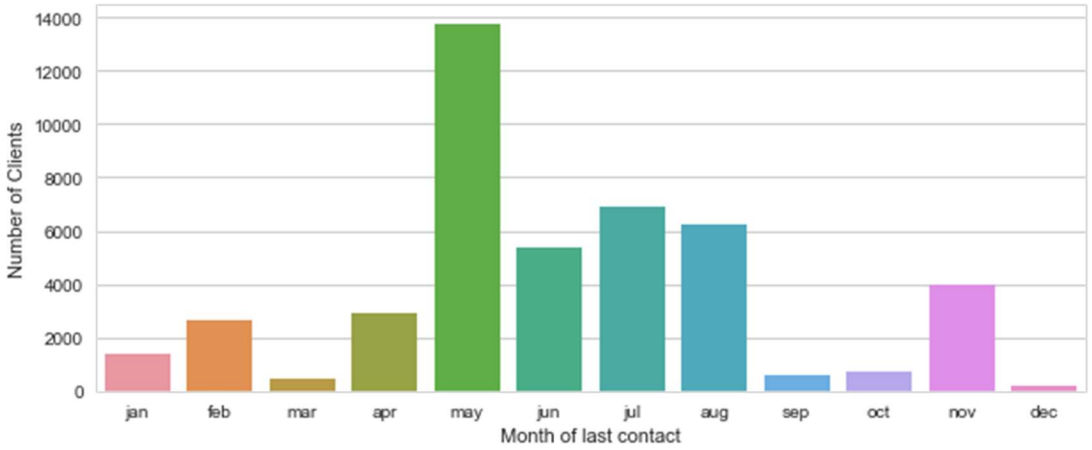


Figure 11.1 - Month of last contact distribution

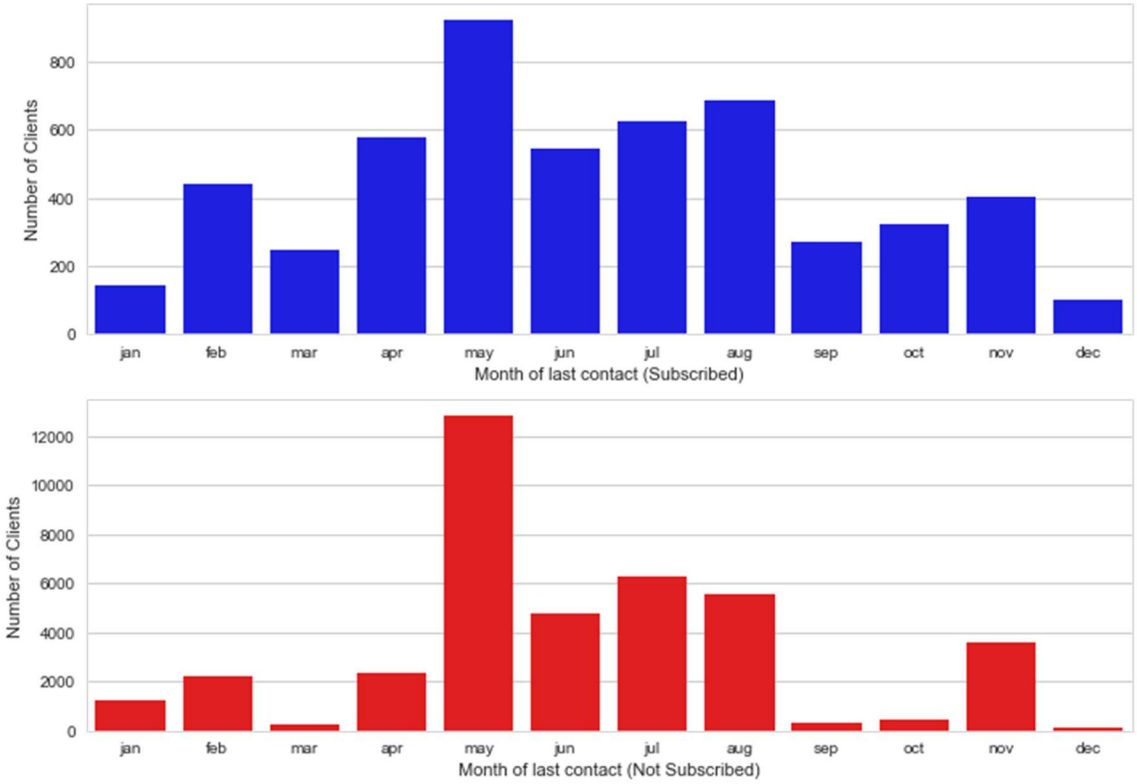


Figure 11.2 - Month of last contact distribution

Visualization of the feature correlation matrix of the dataset.

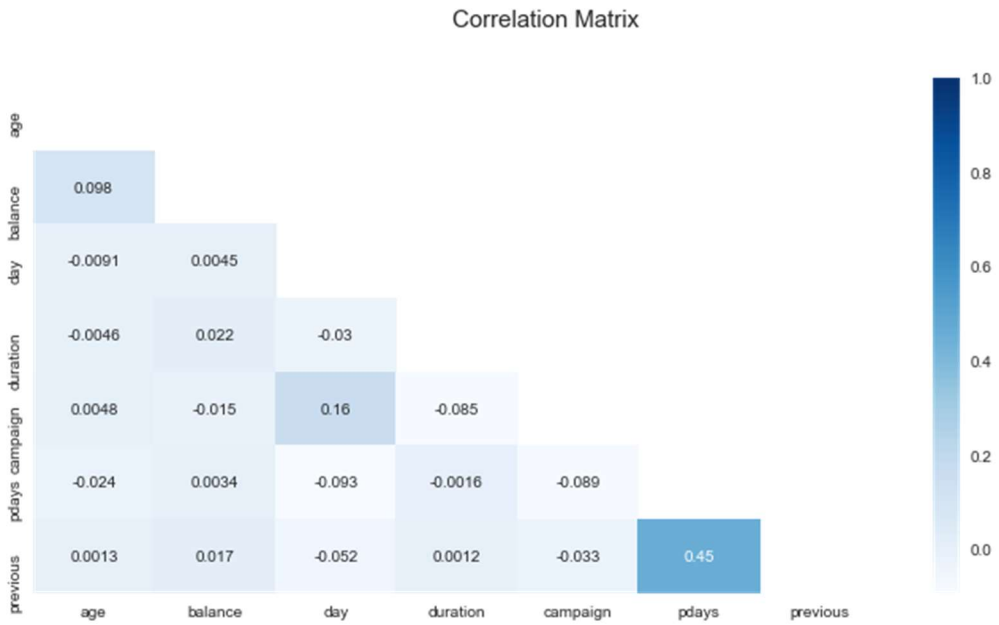


Figure 12 - Correlation Matrix

Algorithms and Techniques

The algorithms that will be used to tackle this binary classification problem are all available in the **scikit-learn** library. As we don't know which algorithm would best fit the data, they were initially used with their default hyper-parameters and trained with the pre-processed training data in order to compare their results and choose the best one (taking into account the accuracy score) for further model tuning.

- **Gaussian Naive Bayes (GaussianNB):** Naive Bayes methods are supervised learning algorithms based on Bayes' theorem. Their are called 'naive' because they assume independence between all features. In this project, GaussianNB wasn't initialized with any parameter.
- **Decision Trees:** Decision Trees are supervised learning algorithms which can be used both for classification or regression. They predict the target variable by learning from the data simple decisions (e.g. is feature x greater than value y?) in the training process. In this project, DecisionTreeClassifier was initialized with random_state=1.
- **Bagging (Ensemble Methods):** Bagging (Bootstrap Aggregating) is an ensemble meta-algorithm that fits base classifiers on random subsets of the original one and eventually aggregate all individual predictions (which can be done by averaging or by voting) yielding a final prediction. In this project, BaggingClassifier was initialized with random_state=1.
- **AdaBoost (Ensemble Methods):** Adaboost (Adaptive Boosting) is also an ensemble meta-algorithm that starts fitting a classifier from the original dataset and later fits copies of the same classifier on the same dataset but this time focusing on

the incorrect classifications, adjusting the weights on these cases. In this project, AdaBoostClassifier was initialized with random_state=1.

- **Random Forest (Ensemble Methods):** Random Forests (or Random Decision Forests) are also an ensemble meta-algorithm that works by fitting various decision trees classifiers on different subsets of the original dataset, averaging the prediction of all decision trees for the final prediction output. In this project, RandomForestClassifier was initialized with random_state=1.
- **Linear Discriminant Analysis (LDA):** Linear Discriminant Analysis is a generalization of Fisher's linear discriminant method, used to find a linear combination of features from the dataset that separate two (or more) classes. The fitted model can also be used for dimensionality reduction. In this project, LinearDiscriminantAnalysis wasn't initialized with any parameter.
- **K-Nearest Neighbors (KNeighbors):** Neighbors-based classification is a kind of instance-based learning, storing instances of the training data and classifying each point by computing from majority vote of the nearest neighbors of each data point. For instance, if 8 of the 10 nearest data points from data point X is classified as class 'A', then data point X is also going to be classified as class 'A'. In this project, KNeighborsClassifier wasn't initialized with any parameter.
- **Stochastic Gradient Descent (SGDC):** Stochastic Gradient Descent is a very efficient approach to discriminative learning of linear classifiers under convex loss functions (such as linear SVM and Logistic Regression). Recently this algorithmn has gained attention in the context of large scale learning. In this project, SGDClassifier wasn't initialized with any parameter.
- **Support Vector Machines:** Support Vector Machines are a set of supervosed machine learning algorithms which can be used both for classification or regression purposes. They are commonly used in classification problems and they work by trying to find a hyperplane that best divides a dataset into classes. Support vectors are the nearest data points to the hyperplane, they are critical elements of the data set because removing them would change the position of the hyperplanes. In this project, SVC was initialized with random_state=1.
- **Logistic Regression:** Logistic Regression a supervised learning technique borrowed from statistics for binary classification problems. Its name comes from the function used at the core of the algorithm: the logistic function (or sigmoid function). Logistic Regression works by modeling the probability of the 'default' class. In this project, LogisticRegression was initialized with random_state=1.
- **eXtreme Gradient Boosting (XGBoost):** XGBoost is an implementation of gradient boosted tree algorithms. It was engineered focusing on efficiency of computing time and memory resources. Boosting is an ensemble method which

adds new models in order to correct the errors made by existing models. In this project, XGBClassifier was initialized with random_state=1.

Benchmark

The main benchmark considered for this project is a naive predictor that predicts the majority class. In this case, the naive predictor has accuracy of **88.51%** and F1-score of **0.0%**. Also, the best untuned model chosen from the Algorithms and Techniques section will be compared to its tuned version. It is expected that the newly tuned model may overcome the benchmark model accuracy and F1-score.

III. Methodology

Data Preprocessing

Before training and testing our classification models, the dataset should be prepared and preprocessed. One of the reasons is that it may contain non-numeric features or null values and most of machine learning algorithms expect numbers to perform computations with.

In our case, the dataset didn't present null values, so the first task was to apply pandas.get_dummies function in order to convert categorical features into binary variables (e.g. feature **marital** has 3 possible string values: single, married and divorced. After the conversion, feature **marital** was replaced by 3 features: **marital_divorced**, **marital_married** and **marital_single** where possible values are 0 or 1), generating what we call dummy variables.

The second preprocessing task was to replace every feature which had either 'yes' or 'no' as possible values to binary values (respectively 1 and 0). After that, the entire dataset was split into features (X - all features but the target) and target (Y - the last column) datasets.

With the features and target datasets in hand, these data were shuffled and split into 4:

- **X_train:** dataset used for model training with 70% of the original size containing all features but the target.
- **y_train:** dataset used for model training with 70% of the original size containing the target variable.
- **X_test:** dataset used for model testing with 30% of the original size containing all features but the target.
- **y_test:** dataset used for model testing with 30% of the original size containing the target variable.

As the last step, X_train and X_test were transformed (normalized) using StandardScaler in order to have mean 0 and a standard deviation 1 before applying the machine learning algorithms.

Implementation

The process steps implemented by this project are:

- **Data Exploration:** in this step some methods and techniques for Exploratory Data Analysis were applied to the dataset. Visualizations of the features were generated, descriptive statistics (mean, standard deviation, min, max, percentiles) were shown and the ratio of clients who subscribed vs clients who didn't subscribe was done. The visualization of the features presented some useful insights (e.g. clients who don't have housing loan are more likely to subscribe to a term deposit). Feature correlation was also presented.
- **Data Preparation:** to prepare the data before training the models, null values were checked and the dataset didn't present null values. Then, all the dataset was processed by applying pandas.get_dummies function with the objective to convert all categorical features into binary variables, called dummy variables. After that, all feature values which were either 'yes' or 'no' were replaced by its binary representation (1/0). After all these changes were done, the target feature (the last column of the dataset 'y') was separated from the features, generating 2 datasets: X_all (with the features) and y_all (with the target class). The last step before the model training step was to call train_test_split function to separate 30% of the data in order to keep it as the test dataset.
- **Model Selection:** here the selected supervised classification algorithms we chose were initialized (untuned), trained with the training data (70% of the data), tested with the testing data (remaining 30% of the data) and their performance were checked with accuracy and F-1 score metrics. The algorithms used were all from the scikit-learn library: GaussianNB, DecisionTreeClassifier, BaggingClassifier, AdaBoostClassifier, RandomForestClassifier, LinearDiscriminantAnalysis, KNeighborsClassifier, SGDClassifier, SVC, LogisticRegression, XGBClassifier.
- **Model Tuning:** in the Model Selection step, XGBClassifier was chosen as the best untuned model because its accuracy was 90.82% in the test set. This step is focused on tuning the XGBClassifier. First, we create cross validation sets using shuffled splits with 20% of testing data and 80% training data (10 splits in total). After that, we started to tune the model with grid search technique.
- **Final Evaluation:** in the final step, the tuned model is created with its hyperparameters found in the previous tuning step and it is compared to the naive predictor as well as its untuned version for benchmarking.

Refinement

To refine our untuned XGBoost model, we used the GridSearchCV to search over specified parameter values the best ones. To create the GridSearchCV instance, first, all training data was splitted into 10 subsets, each containing training size 80% and testing size 20%. Also, the scoring parameter was set as a dict mapping the scorer names to the scorer callables (Accuracy and F1-Score).

The tuning parameters, range of values tested and the best values found by grid search are shown below:

- **scale_pos_weight**: experimented values from 1 to 10. Best value found: 1.
- **objective**: experimented values reg:linear, reg:logistic, binary:logistic, binary:logitraw, binary:hinge and count:poisson. Best value found: reg:logistic.
- **max_depth** and **min_child_weight'**: experimented values from 3 to 10 for max_depth and 1 to 6 for min_child_weight. Best value found for max_depth: 6 and for min_child_weight: 3.
- **subsample** and **colsample_bytree'**: experimented values from 0.5 to 1.0 (loop: 0.1) for both subsample and colsample_bytree. Best value found for both subsample and for colsample_bytree: 1.0.
- **reg_alpha** and **reg_lambda'**: experimented values from 0 to 5 for reg_alpha and 1 to 6 for reg_lambda. Best value found for reg_alpha: 1 and for reg_lambda: 1.
- **gamma'**: experimented values from 0.0 to 1.0 (loop: 0.1). Best value: 0.7.

IV. Results

Model Evaluation and Validation

Based on the model selection section, among all algorithms, XGBClassifier (without tuning) presented the highest accuracy: 90.82% in the test set.

After the model hyperparameters were tuned, XGBClassifier could reach accuracy of 91.26%, so I suggest the final tuned model results do align with our expectation because we could get a better accuracy than the naive predictor, which has accuracy of 88.51% in the test set. The final tuned model could generalize well to unseen data because of the train_test_split (80% train/20% test) we did: the accuracy on the training set was 92.82% and on the testing set (unseen data), as told before, was 91.26%, making the model reliable and this way we can trust the results.

Justification

The results we could get by tuning the XGBClassifier are slightly better than the untuned model and a little better than the naive predictor. One issue I faced in this project was lack of computational processing power, every search for best values for the hyperparameters in the GridSearchCV took a long time or crashed the application. There are probably better values to make the results better, but I had to limit the range for them because of computer resources. Despite the results weren't much higher than the benchmark, I consider the final solution to have solved the problem.

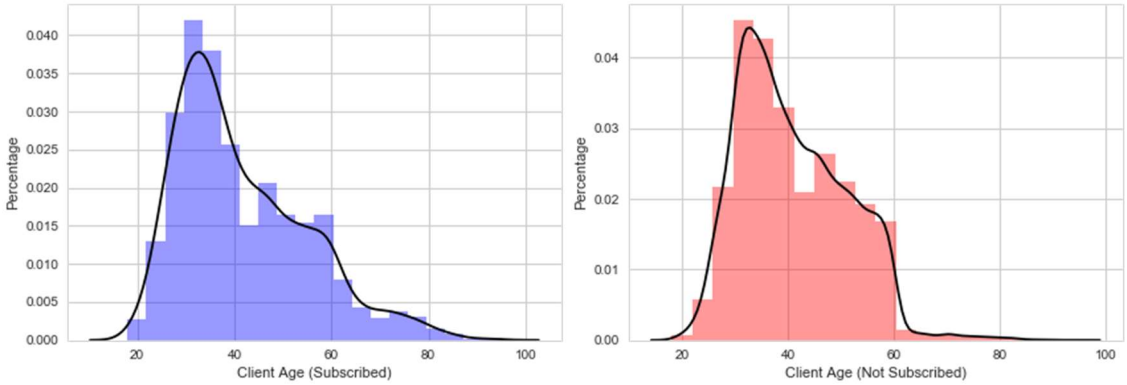
	Naive Predictor	Default XGBClassifier	Tuned XGBClassifier
Accuracy	88.51%	90.82%	91.26%
F1-Score	0%	49.16%	56.19%

Figure 13 - Benchmark Comparison

V. Conclusion

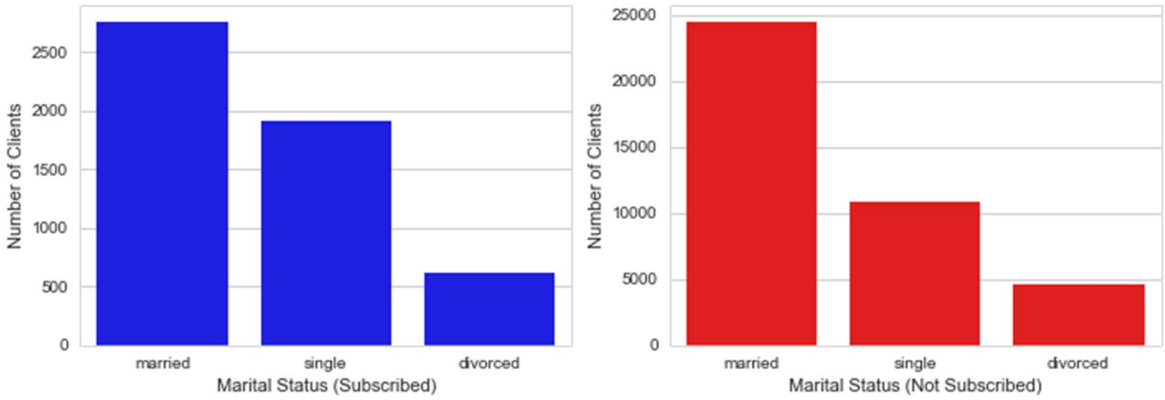
Free-Form Visualization

For this project, I consider interesting the histograms comparisons of some features I did. For example, comparing the age feature (clients who subscribed to a term deposit vs clients who didn't):

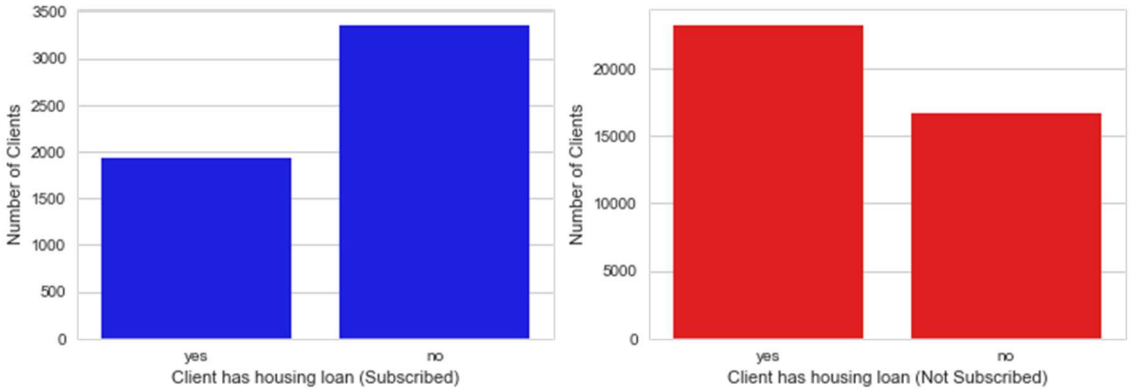


This image shows that clients between the ages of 60 and 80 tend to subscribe to a term deposit. Few clients at this age interval didn't subscribe to a term deposit.

By comparing the marital status of the clients, we can see that clients who are single also have a greater chance to subscribe to a term deposit.



Another interesting comparison was visualizing clients who have housing loan vs who didn't. The image shows that if the client didn't have a housing loan, chances of subscribing to a term deposit are higher.



Reflection

In this project, the biggest challenge for me was to tune the model. After creating a list of classifiers and training, testing and comparing all of them to choose the best one (XGBoost), I spent a lot of time trying to understand its hyperparameters and every search for best values using GridSearchCV took a long time. First, I tried to create a dictionary containing all hyperparameters and specified value ranges, but the processing would not finish. So I started reading how to fine tune the XGBoost in a more concise way.

I think the final tuned model fit my expectations, despite knowing that if I had more computational processing power, I'd probably find even better hyperparameter values.

I found the project interesting because it has imbalanced data (approximately 88% of 'no' and 12% of 'yes' to a term deposit subscription). This characteristic is very common for almost any product a company wants to sell or provide. Understanding which features are the most relevant and connected to a success of a product is extremely useful for any kind of business.

Improvement

There is plenty of room for model improvement. I would suggest as further improvements to try longer intervals values for hyperparameters. Also, there are some hyperparameters that I haven't considered for tuning that could show better results. Another improvent I would suggest was to not use accuracy as the main metric. I understood that late by reading about the 'Accuracy Paradox' that may happen in imbalanced data. I would suggest recall rate, AUC/ROC metric. If this final model was used as the new benchmark, it's very likely that a better solution does exist by spending more time on hyperparameter tuning process.

References

[1] S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimaraes, Portugal, October, 2011. EUROSIS.

[2] Ling, X. and Li, C., 1998. Data Mining for Direct Marketing: Problems and Solutions. In Proceedings of the 4th KDD conference, AAAI Press, 73–79.

[3] Ou, C., Liu, C., Huang, J. and Zhong, N. 2003. On Data Mining for Direct Marketing. In Proceedings of the 9th RSFDGrC conference, 2639, 491–498.

[4] https://archive.ics.uci.edu/ml/datasets/bank+marketing

[5] Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.