Project Stir Bank

NISHANT KUMAR
DEPARTMENT OF CSE(AI)
Noida Institute of Engg. & Tech.
Noida, India
2201331520118
0221csai029@niet.co.in

SURAJ SINGH RANA
DEPARTMENT OF CSE(AI)
Noida Institute of Engg. & Tech.
Noida, India
2201331520187
0221csai030@niet.co.in

VIVEK P. SINGH
DEPARTMENT OF CSE(AI)
Noida Institute of Engg. & Tech.
Noida, India
2201331520230
0221csai113@niet.co.in

Abstract- **This presents a comprehensive analysis of bank marketing campaign data to predict customer subscription to term deposits using machine learning techniques. The study utilizes a dataset from StirBank, comprising 8,000 customers and 21 attributes, including demographic, financial, and campaign-related variables. The primary objective is to identify key factors influencing deposit subscriptions and develop predictive models to optimize future marketing efforts. Three classification models—Logistic Regression, Decision Tree, and Multilayer Perceptron (MLP)—are implemented and evaluated. Data preparation steps, including cleaning, feature scaling, and variable selection, are detailed to ensure model accuracy. Results indicate that the MLP model outperforms the others with an accuracy of 0.811 and an AUC of 0.882. The findings suggest that variables such as contact duration, age, and current balance significantly impact deposit subscriptions. This study provides actionable insights for banks to enhance marketing strategies and improve campaign efficiency.**

Keywords: bank marketing, deposit prediction, machine learning, classification, customer segmentation, logistic regression, decision tree, neural network, feature engineering, predictive modeling.

I. INTRODUCTION

A good market campaign plays a significant role in financial selling and attracting customers to deposit is crucial for the business. The report aims to suggest an optimum set of customers by predicting which attributes mostly impact the subscription to the term deposit by a successful bank marketing calls campaign. It would be advantageous to narrow the range of prospective customers, raising the ratio of prosperity along with restricting the expenses of the telemarketing procedure effectively.

The dataset I have received includes data from StirBank describing 8,000 of its customers and former attempts of marketing calls made to them. This is a historical customer dataset where each row represents one customer. The data is relatively easy to understand, some of the variables are selfexplanatory whereas others demand to think more widely about the general business problem and operations of a bank, as factors which aid to discover the meaning of some hidden-meaning variables. In general, the dataset gives details such as specific attributes of the customers, just to name a few of them; age, customers' education, the town and country they live, their marital status, their current balance on account/s they have with the bank or eventually if they have a loan or not. Moreover, additional valuable information we get from the dataset are details regarding the marketing campaign, this includes the last contact day and month, days since the last contact and possibly the outcome of a former telephone campaign. Lastly regarding the dataset, we are given information on whether each customer responded positively (or negatively) to the marketing program by setting up a regular savings deposit – this corresponds to the variable *made_deposit*. This is a particularly significant variable-attribute because gives the information we need to predict regarding future marketing campaign that focuses on the most suitable set of customers in terms of a positive response to this campaign.

As far as my methodology is concerned, applying Machine Learning (ML) techniques to poke into huge quantities of data can facilitate the process of discovering patterns which aren't straightaway evident. That is what is called data mining. As a formal definition of Machine Learning, ML is the subfield of computer science that gives "computers the ability to learn without being explicitly programmed." Using ML permits to build a model that considers all the feature sets – or the selected attributes used to train the model, alongside the corresponding answer of a positive or negative response of making a deposit, and it learns the pattern of a set of customers and corresponding responses. Therefore, ML algorithms, influenced by the human learning process, iteratively acquire knowledge from data and let computers obtain hidden insights. The type of problem to solve is directing me on which ML technique to use This is a classification case study since this method is utilised in forecasting the class of an instance, such as whether or not a customer will make a deposit and looking directly at the value of the data, we can have two kinds. Classification is the task of predicting a distinguished category or class label. Specifically, this is a supervised learning problem, that means the training data feeding the algorithm include the desired solutions, that are called labels. In this case, study the labels correspond to a positive answer - 'yes' - that a customer made a deposit in the bank or a negative one – 'no' – that means the customer did not respond positively in the marketing campaign, therefore didn't make a deposit.

I will make use of different classification models, particularly this includes decision tree modelling, multi-layer perceptron (MLP) and logistic regression. Finally, these data mining models will also be compared which will enable me to validate the effectiveness of the prediction. All of the processes and experiments for Exploratory Data Analysis (EDA), data manipulation and visualisation, implementation of the models and their evaluation are implemented using Python programming language.

II. DATA SUMMARY

The purpose of this paper is to detect the relationship between the success of bank marketing calls and the rest of attributes, including bank client data, demographic attributes, etc., as well as to predict the success of bank marketing calls given that customers make a deposit. The data has been provided by StirBank, a Scottish institution, and are related to the success of bank marketing campaigns. In this data set, the same customer has been contacted more than once by the bank (the variable *campaign* releases this), in order to subscribe to the term deposit.

There are 8,000 instances in the dataset with each observation to represent one customer, and 21 variables which represent attributes of each customer. At a high-level overview, those attributes are summarised in three categories; bank client data, contacting attributes and other attributes which provide more details about the marketing campaign and previous attempts. Except for the output target *y* (*made_deposit*), there are 20 attributes, containing age, job, marital status, education, default status, etc. The detailed descriptions about the attributes are presented in Table 1. It is also worth noting that StirBank has not provided the exact description of the attributes so the description of the data has primarily come from my research. Some of the variables have self-explanatory names, however for the more cryptic ones in terms of their definition I have found similar datasets from bank marketing campaigns and related academic articles which led me to assume that their meaning is most probably within the bounds of my interpretation.

In closer inspection to the table, twelve out of twenty one variables are nominal. These variables represent discrete units or labels with their corresponding values to belong to two or more categories, which don't have an intrinsic order. The remaining nine variables are numerical from which seven are discrete (accountID, age, cc_tr, last contact day, campaign, days since last contact, previous) and two are continuous (current balance, last contact duration_s). Moreover, I have tested the dataset for null values and it seems that none of the attributes shows instances of missing values. However, by looking closer into the data, I have noted some data entry errors. Those are related with five features (has_loan, last_contact_month, last_contact_duration_s, days_since_last_contact and poutcome). The data preparation section will expand more into this as well as fixing those errors where needed.

Multiple information of the features has been gathered by finding the descriptive statistics (*Table 2*) of the numerical attributes and applying the python function of *value_counts()* for the nominal variables that identify the unique categories. The labelled data of the *'made_deposit'* feature gives the indication of how many of each class are in our dataset that responded positively in the marketing calls. As a result of the latest campaign, 48% of the customers have deposited funds in the bank. As far as the town and country of the bank's customers, I have found that most of the customers live in different cities within the UK, specifically, there is a large number of customers - equal to 1836 people - who live in London, Birmingham, Glasgow, Bristol, Liverpool and Manchester, in priority order from higher to lower as having been written here. The rest of the customers are spread in additional 95 unique towns. However, by checking the unique values of the *country* variable, I note that in the dataset there are two instances where the country is France and one instance per each country for Portugal, Germany and USA, the 99.9% of instances live in the UK.

A way to summarise and understand the data is by using graphics and visualization tools, for example, by plotting a histogram for each numerical feature. The age feature for instance with minimum and maximum equal to 18 and 93 respectively; the customers' age distribution is displayed in Figure 1. It is obvious that this is roughly acknowledged as skewness distribution, which is skewed to the right. Besides, the consulted customers aged between 27 and 41 are in the vast majority. It is concluded that the main target of counselling is middle-aged people. Finally, many histograms are tail heavy, they extend much farther to the right of the median than to the left.

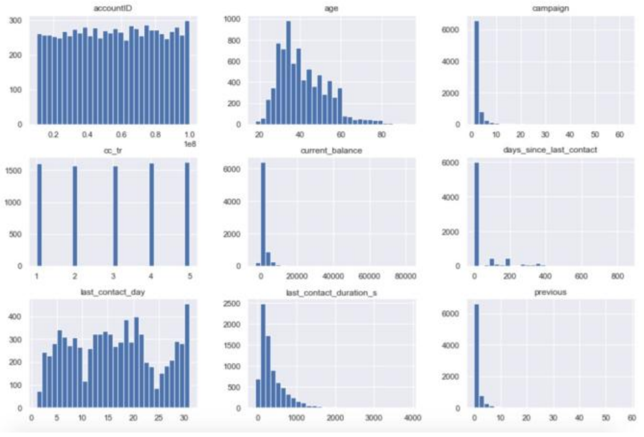


Figure 1. *A histogram for each numerical attribute.*

The distribution of the customers' jobs is shown in Figure 2. As can be seen from this count plot, the bank consults mainly with management, blue-collar workers, technicians and administrators in order to succeed to subscribe to the term deposit. Among them, management account for 23% of total customers. The second is blue-collar workers, who accounted for 17% of the total, only 1 percent more than the number of technicians. The same diagram releases useful information if we split into those who replied positively/negatively to the campaign by subscribing the term deposit. Management workers and administration staff appear a small difference between the groups. However, the bluecollar workers who came second in the customers' job rank appear a big negative difference meaning that many of those did not make a deposit.

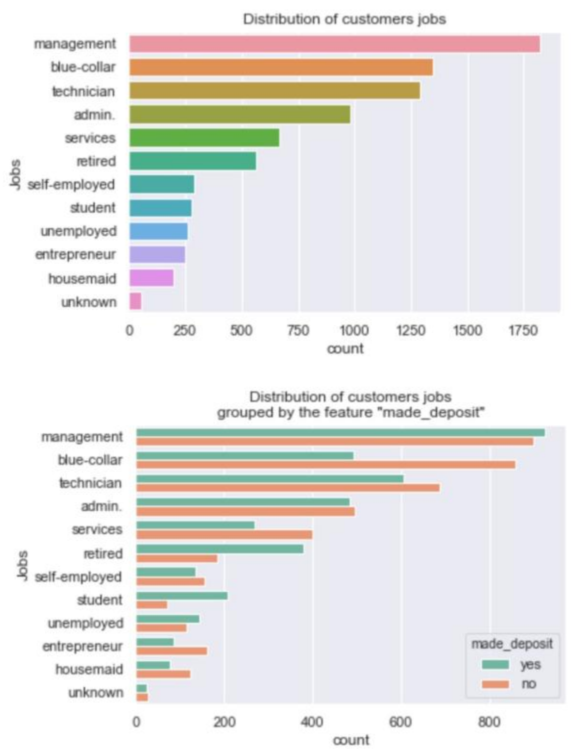


Figure 2. *The proportion of customers' jobs (left), the proportion of customers' jobs grouped by the feature 'made_deposit'.*

The proportion of customers education is illustrated in Figure 3. It is obvious there are four categories with the most frequent to be secondary and tertiary education which represent 49% and 33% respectively.

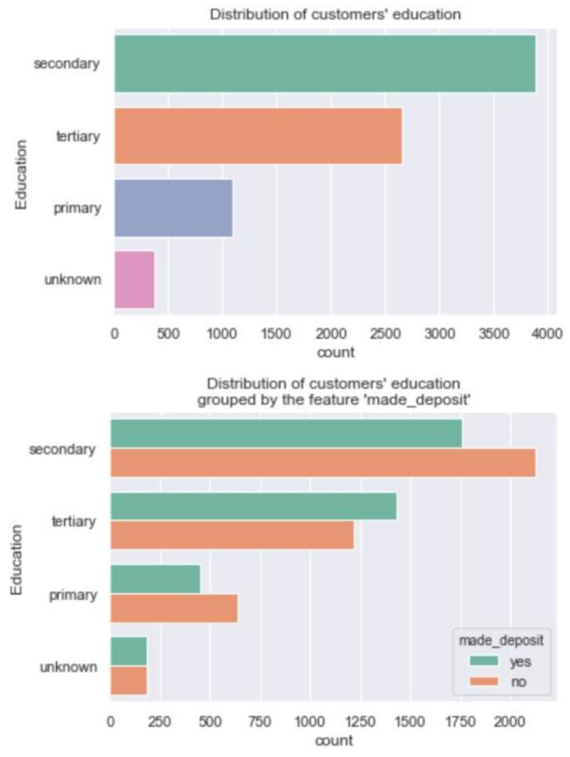


Figure 3. *The proportion of customers' education (left), the proportion of customers' education grouped by the feature 'made_deposit'.*

Regarding the variables that will be used for the next stage, the histogram of the accountID indicates a wide and low distribution, consequently, this variable cannot provide any useful information and should be removed. Moreover, 75% of the observations of the feature '*poutcome*' is unknown; with only 10% to show success or failure of the previous outcome's campaign. Undoubtedly, it wouldn't be useful to drop all this information. However, we cannot base our results on the assumption of 75% of the '*poutcome*' unknown values. Data Preparation section will give insights for the rest of the variables in terms of feature and variable selection.

III. DATA PREPARATION

At this stage, the raw data which are often inconsistent are processed and optimised in order to enhance the predictive effect of modelling. It is of paramount importance to prepare and transform the data before feeding them to the ML algorithms. The following steps have been followed to achieve optimization. Data Cleaning

Most Machine Learning algorithms cannot work with missing features. Exploring the dataset has provided the insight that there are no missing values, however, some variables include data entry errors and unknown values. The solution to this is by making assumptions while considering the overall picture; meaning, for example, the assumption to delete some observations due to data entry errors or even get rid of a whole attribute, would mean loss of information and a small sample. Still, there are other solutions for not losing information such as replacing data with the mean value of a numeric attribute or replacing with the most frequent value if the variable is categorical. This leads to the next point of data quality and quantity. Regarding the latter, the dispersion of values in each of the variables plays an important role in determining the quantity of data needed. Unbalanced variables need more samples to be sure of collecting sufficient examples of rare values. Therefore the points discussed indicate a dependency between data quantity and quality in achieving optimisation and give ML algorithms the best chance to succeed.

In a closer inspection of the dataset quality, visualizing the data with histograms, bar charts and box plots can reveal data entry errors and other anomalies such as outliers and rare values. Figure 4 depicts cases of outliers through box plots of some numerical variables.

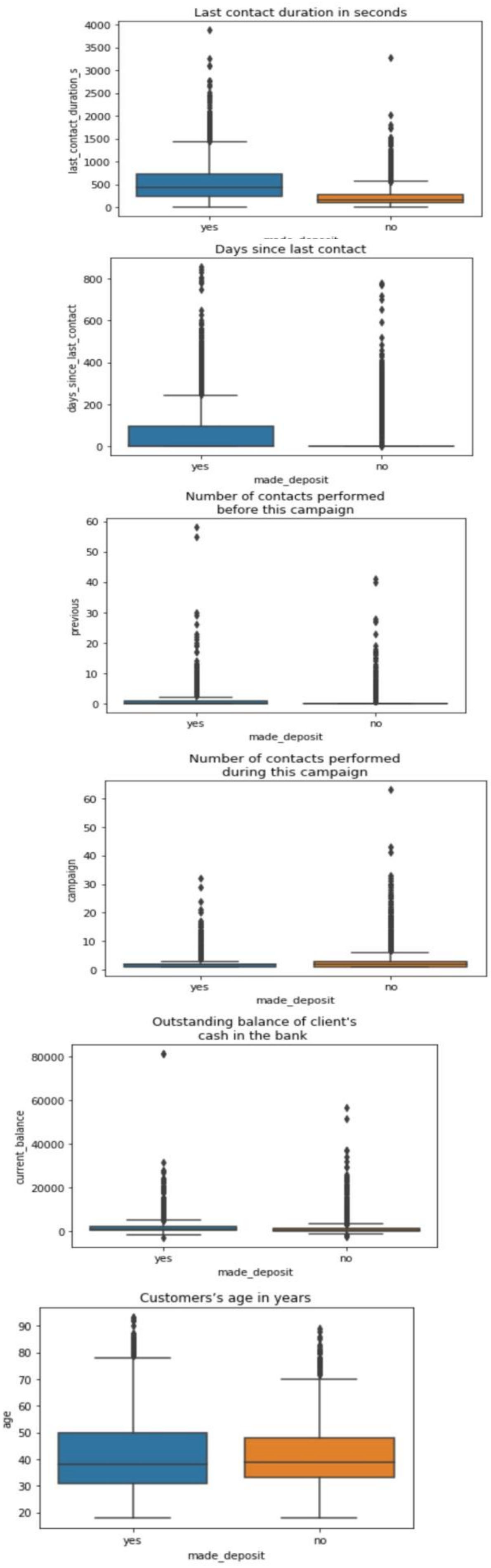


Figure 4. A set of box plots of some numerical variables, which reveal the 25%, 50% (median) and 75% percentiles along with the presence of outliers.

Outliers might indicate data entry and measurement errors or sampling problems. However, they also exhibit natural variation of the population studied. In the last case, they shouldn't be removed. Extreme values can occur, especially when the sample size is large enough, but they accompanied by lower probabilities. In response to that, values that appeared errors have been transformed, this includes, for example, the *last_contact_duration_s* feature which initially had a min value equal to 60 *(Table 2),* after the transformation of replacing the value with 60, the new min value is two. Since this variable represents seconds, it seems logical no to find negative values. The variable *days_since_last_contact* included multiple negative values too which doesn't make any sense; specifically, the 74% was equal to (-1), these have been replaced with positive one. The pairs plot in Figure 5 visualises these two variables after the transformation. The minimum current balance of a customer was found negative but for this, I have made the assumption that might represent an overdraft, consequently, no changes made.

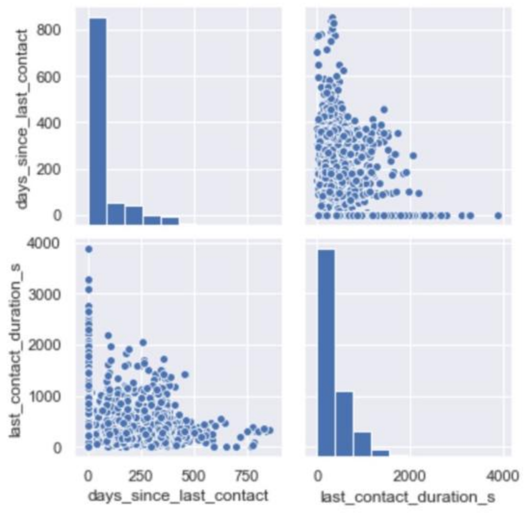


Figure 5. A pairs plot of 'last_contact_duration_s' and 'days_since_last_contact' after the transformation of data entry errors.

Additional variable processing includes the replacement of the error 'j' value in the *last_contact_month,* this has been transformed to 'jul' indicating July was the most frequent month in the dataset starting with the letter 'j'. The variable *has_loan* appeared 5 instances with value 'n', this has been replaced with the value 'no'. Figure 6 shows the distribution of this variable. Finally, the *last_contact* attribute includes two values named 'cell'. The majority, however, has the value 'cellular', consequently, this has been replaced as well.

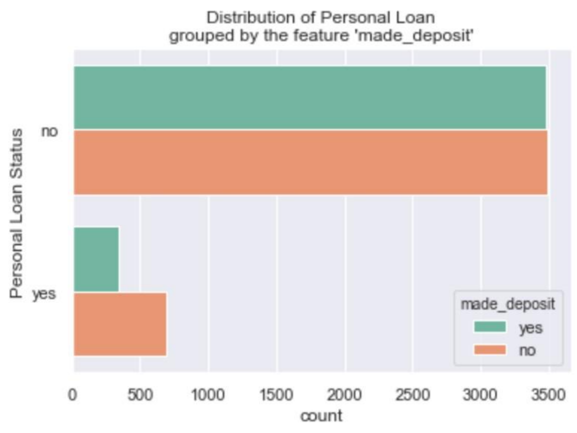


Figure 6. *The proportion of customers' personal loan status grouped by the feature 'made_deposit'.*

Handling Categorical Attributes
Next step in the data preparation process is to transform all categorical variables into numerical ones. For the binary features I have replaced, for example, the most common set of values [*no or yes] with [0 or 1]* respectively, this includes the target variable *made_deposit* as well. Moreover, multi-class categorical variables have also been transformed by utilising one hot encoding process indicating the presence of a class. The whole process aims to prepare the data in order to feed the ML algorithms and make better predictions.

Feature Scaling
One of the most important transformation to apply in the dataset is feature scaling because of the different qualities of the indicators. Standardization is the process I followed in order to transform raw data and get all attributes to have the same scale, an advantage of standardization is that is much less affected by outliers compared with min-max scaling for instance. The standardised values have a zero mean and unit variance. **Optimal Variable sets**

In addition to all of the technique related decisions, there are choices to be made about the data. Variable selection refers to the problem of which variables should be included or selected, whereas feature selection is how the variables are combined or manipulated to help the algorithm learn the correct task. As A starting point, I exclude from my variable set the attributes that will not be used for modelling. Practically, I have dropped the *accountID, and poutcome* as mentioned previously. What is more, after the one-hot-encoding the correlation of the target variable *(made_deposit)* with each of the towns and countries in the dataset found to be really insignificant, as a result, to exclude both *town* and *country*. In this case, maybe the combination of customers' salary combined with the town they live could better predict the subscription to the term deposit. However, customers' salary is not known.

Similarly, the correlation of the target with the independent variables *last_contact, cc_tr,* and *last_contact_day* is insignificant, so they've been excluded. Moreover, the days since the last contact seems more promising than the last contact day for predicting a deposit subscription.

The rest of the attributes have been carried to the process of embedded feature selection. This is a method (a tree-based model which places important features near the root of the tree and less important near the leaves) that assigns importance to each feature and ultimately provides useful insights for variable selection *(Figure 7)*. Additionally, binning the variable *age* into groups as transformation with the purpose of finding hidden insights hasn't proved any efficiency (this has been tested with the embedded method for variable selection), consequently, age remained in the given form.

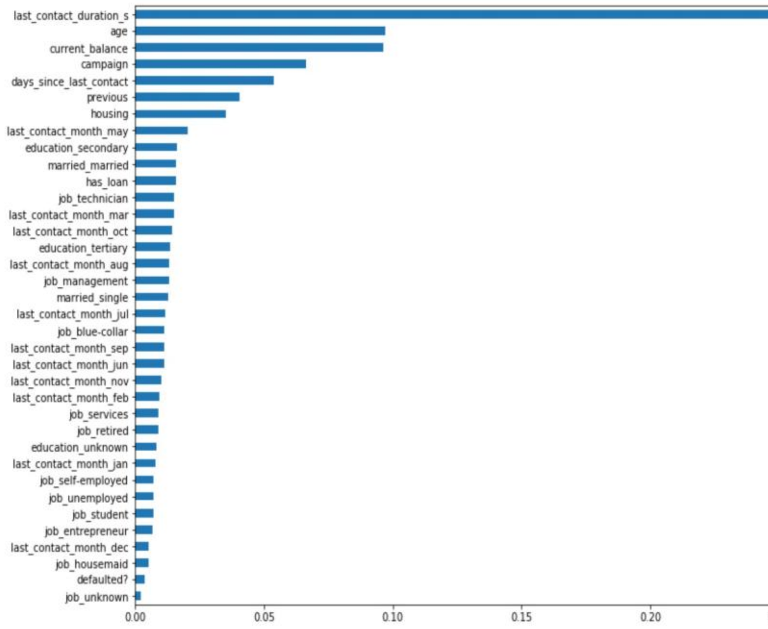


Figure 7. *Embedded feature selection; the plot depicts the importance along with the name of each feature.*

The most important features which will be taken in the modelling section for predicting deposit subscription, based on embedded feature selection, are the following;

- `last_contact_duration_s`
- `age`
- `current_balance`
- `campaign`
- `days_since_last_contact`
- `previous`
- `housing age`

IV. BACKGROUND

Logistic Regression

Logistic regression is a statistical and machine learning method for classification of records of a dataset based on the values of the input fields. Logistic regression is analogous to linear regression but tries to predict a categorical or discrete target field instead of a numeric one. In other words, is a variation of Linear Regression, useful when the observed dependent variable, y, is categorical. It produces a formula that predicts the probability of the class label as a function of the independent variables. In logistic regression, we predict a variable which is binary such as yes/no, true/false, and so on, all of which can be coded as zero or one. Additionally, the dependent variables should be numerical. If categorical, they should be dummy or indicator coded. This data mining technique can also be used for multi-class classification. The decision boundary of logistic regression is a line or a hyperplane. A classifier will classify all the points on one side of the decision boundary as belonging to one class and all those on the other side as belonging to the other class. It is possible also to achieve a complex decision boundary using polynomial processing. Logistic regression fits a special s-shaped curve by taking the linear regression and modifying the numerical estimation to a likelihood with the ensuing function, which is called sigmoid function σ:

$$h_\theta(x) = \sigma(\theta^T X) = \frac{e^{(\theta_0+\theta_1 x_1+\theta_2 x_2+\ldots)}}{1+e^{(\theta_0+\theta_1 x_1+\theta_2 x_2+\cdots)}}$$

$$ProbabilityOfaClass_1 = P(Y=1|X) = \sigma(\theta^T X) = \frac{e^{\theta^T X}}{1+e^{\theta^T X}}$$

In this equation, $\theta^T X$ is the regression result (the sum of the variables weighted by the coefficients), exp is the exponential function and $\sigma(\theta^T X)$ is the sigmoid or logistic function, also called a logistic curve. It is a common "S" shape (sigmoid curve). So, briefly, logistic regression passes the input through the logistic/sigmoid but then treats the result as a probability *(Figure 8):*

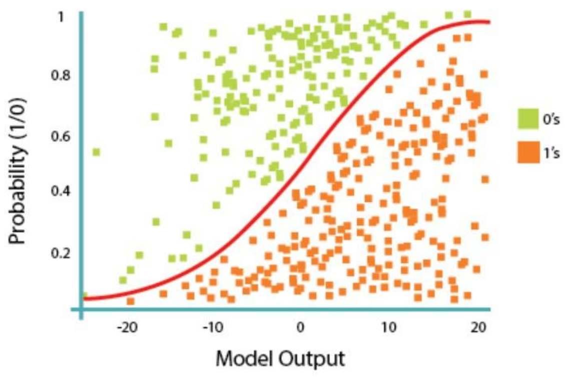


Figure 8. *Logistic Regression; Model Output. (Source: Coursera).*

The objective of Logistic Regression algorithm is to find the best parameters θ, for $h_\theta(x) = \sigma(\theta^T X)$, in such a way that the model best predicts the class of each case. In other words, training a logistic regression model aims to change the parameters of the model, so as to be the best estimation of the labels of the instances in the dataset. This leads to the next point of the cost function, using the derivative of the cost function we can find how to change the parameters to reduce the cost or rather the error. The cost function is the difference between the actual values of y and the model output predictions y hat (equal to $\sigma(\theta^T X)$). This is a general rule for most cost functions in machine learning.

The cost function:

$$Cost(\hat{y}, y) = \sigma(\theta^T X) - y$$

The cost function for all the samples in the training set:

$$J(\theta) = \frac{1}{m}\sum_{i-1}^{m} Cost(\hat{y}, y)$$

The way to find or set the best weights or parameters that minimize this cost function is by calculating the minimum point of this. Although we can find the minimum point using the derivative of a function, there's not an easy way to find the global minimum point for such an equation. The solution is to find another cost function instead, one which has the same behaviour but is easier to find its minimum point. The minus log function provides such a cost function *(Figure 9).*

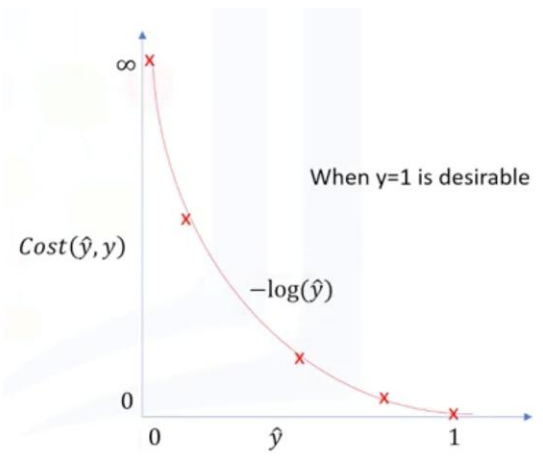


Figure 9. *The minus log function. The function returns zero if the outcome of the model is one. The cost is increasing as the outcome of the model gets further from one, and it is very large if the outcome of the model is close to zero. (Source: Coursera)*

The cost can be calculated as:

$$Cost(\hat{y}, y) = \begin{cases} -\log(\hat{y}) & if\ y = 1 \\ -\log(1-\hat{y}) & if\ y = 0 \end{cases}$$

After plugging this to the total cost function $J(\theta)$, we see the logistic regression cost function:

$$J(\theta) = -\frac{1}{m}\sum_{i-1}^{m} y^i \log(\hat{y}^i) + (1-y^i)\log(1-\hat{y}^i)$$

There are different optimization approaches to minimise the cost function, gradient descent is one of the most effective ones. Gradient descent, in this case, is a technique to use the derivative of a cost function to change the parameter values to minimize the cost or error. Gradient descent is like taking steps in the current direction of the slope of an error surface, and the learning rate is the length of the step or an additional control on how fast is the movement on this surface. To conclude, the training algorithm step-by-step is shown in *Table 3.*

	Logistic Regression Training
1	Initialise the parameters with random values.
2	Feed the cost function with the training set, and calculate the cost.
3	Calculate the gradient of the cost function, a partial derivative should be used.
4	Update the weights with new parameter values.
5	Back to step two and feed the cost function again, which has new parameters (less error is expected as we are going down the error surface. This loop continues until a short value of cost is reached.
6	The parameter should be roughly found after some iterations. This means the model is ready and can be used to predict the probability of a customer depositing or not.

***Table 3.** The training algorithm step-by-step.*

1) *Decision Tree*

Decision trees are constructed by dividing the train set to distinctive knots, where one-knot comprises all or most of one class of the data. Decision trees are about testing an attribute and branching the cases based on the result of the test, using recursive partitioning to classify the data. Every inner node reciprocates to a test, every branch reciprocates to an answer of the test, and every leaf node delegates the customer to a class.

A decision tree can be built by taking into account the feature-variables one by one. Firstly, choosing an attribute from the dataset and calculating the significance of the attribute in the splitting of the data, this will reveal if it's an effective attribute or not. Next, splitting the data based on the value of the best attribute, and going to each branch and repeating the process for the rest of the attributes. After building the tree, it can be utilised to predict the class of unknown cases.

The algorithm chooses the most predictive feature to split the data on. It is important to determine which attribute is the best or more predictive to split data based on the feature. In fact, predictiveness is based on a decrease in the impurity of nodes. The best feature will decrease the impurity of customers in the leaves, after splitting them up based on that feature. The choice of an attribute to split data is very important and it is all about the purity of the leaves after the split. A node in the tree is considered pure if, in 100 percent of the cases, the nodes fall into a specific category of the target field. In fact, the method uses recursive partitioning to split the training records into segments by minimizing the impurity at each step. Impurity of nodes is calculated by the entropy of data in the node. Entropy is the amount of information disorder or the amount of randomness in the data. The entropy in the node depends on how much random data is in that node and is calculated for each node. In Decision Tree models, the smallest entropy in the nodes is desirable. The entropy is utilised to compute the instances' homogeneity in that node, the lower the entropy, the less uniform the distribution, the purer the node. If the samples are entirely identical, the entropy will be zero, if the samples are evenly separated, the entropy equals to one. Entropy can be calculated by:

$$\text{Entropy} = -\,p(A)\log(p(A)) - p(B)\log(p(B))$$

Information gain is the intelligence which may enhance the confidence levelness after splitting. It is the entropy of a tree before the split minus the weighted entropy after the split by an attribute. We can think of information gain and entropy as opposites. As entropy or the amount of randomness decreases, the information gain or amount of certainty increases and vice versa. So, building a decision tree is about discovering attributes which return the upmost information gain. As a procedure, we take into consideration the entropy over the distribution of samples occurring under every leaf node and get the weighted average of this entropy weighted by the proportion of samples falling under that leave. Concluding, the root of the decision tree starts with the attribute that provides the higher information gain after splitting. In order to identify the next attribute after branching, the process should be repeated for each branch and test each of the other attributes to continue to reach the purest leaves.

2) *Multilayer Perceptron*

Multilayer perceptron (MLP) model is a Machine Learning method capable of performing prediction, classification and novelty detection, all by means of regression. An MLP is a technique trained with gradient descent (the learning algorithm) on a squared error cost function with given settings for learning rate, early stopping, hidden units and activation functions, the hyper parameters.

The composition of the model includes single input and output layers, and a hidden layer of variable size. Each layer incorporates an unlike number of perceptron, each determined as an activation function. The implementation of the prediction of the class is carried out by feeding the neural network with the input variables-parameters and passing it through every layer and perceptron until the output. The nature of the function utilised in the output layer relies on the sort of the problem the neural network attempts to resolve. The configuration of the influences of the MLP is trained with the gradient descent method. The procedure of learning begins from the output layer and reaches the input, employing the error backpropagation technique.

V. MODELLING

The different approaches to data mining, which have been discussed in the previous section, are now used to build the models. The classification models proposed, logistic regression, decision tree and multi-layer perceptron, focus on predicting the target through marketing calls in order to subscribe to the term deposit.

It is important that a model has high, out-of-sample accuracy, because the purpose of any model, of course, is to make correct predictions on unknown data. So a way to improve out-of-sample accuracy is by using an evaluation approach which called train and test split. This involves splitting the dataset into training and testing sets respectively, which are mutually exclusive. This will provide a more accurate evaluation of out-of-sample accuracy because the testing dataset is not part of the dataset that have been used to train the data. It is more realistic for real-world problems.

Bank marketing dataset, meaning the predictors selected by embedded feature selection, is splitted in two components, 80 percent will be employed for training and 20 percent will be for the validation and testing. The same variables and amount of data used for train and test will be used across the three models. The training data is utilised to model a well-fitted and rational model, which the main operation entails to detect eventual predictors. Regarding the test dataset, is used to compute the extent of the precision from the predictions, the accuracy of the model indicates the competence and effectiveness of the model.

The models' experimental process is shown in the following paragraphs.

Logistic Regression

The model is built using the function *LogisticRegression* from Scikit-learn package. The function implements the logistic regression and can use different numerical optimizers to find parameters, here I used the 'liblinear' solver and selected the parameter C equal to 0.01. This parameter indicates inverse of regularization strength which must be a positive float. Smaller values specify stronger regularization. The next step is to fit the model with the training set and then make predictions using the test set, the result is an array with predictions of the target variable. What is more, I utilised the *predict_proba* method which returns the probability estimates for all classes which are ordered by the label of classes. So, the first column is the probability of class 1, P(Y=1|X), and the second column is the probability of class 0, P(Y=0|X), the outcome is shown below;

```
array([[0.52, 0.48],
       [0.31, 0.69],
       [0.8 , 0.2 ],
       ...,
       [0.51, 0.49],
       [0.56, 0.44],
       [0.18, 0.82]])
```

1) Decision Tree

The model is built by creating an instance of the DecisionTreeClassifier called *depositTree*. Inside of the classifier, the Criterion is specified as 'entropy' so we can see the information gain of each node and Max Depth equals to four. Next, fitting the data with the training feature matrix X_train and training response vector y_train that have been created previously with *train_test_split* method (mentioned previously). The next step is to make predictions on the testing dataset, i.e. using the trained decision tree to predict the class of an unknown customer, or to find out - given customer's characteristics if subscribes the term deposit. To visually compare the predictions made to the actual values I print the predictions mentioned and the y_test set. For example, a screenshot of this;

```
[1 1 0 0 1]
[1 1 0 1 1]
```

Lastly, I have further explored the decision tree (*Figure 9) by its visualization*.

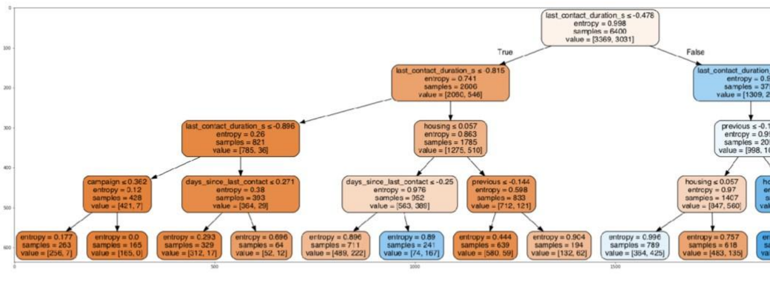


Figure 9. *Visualization of the Decision Tree, depicting the nodes and information gain on each leaf.*

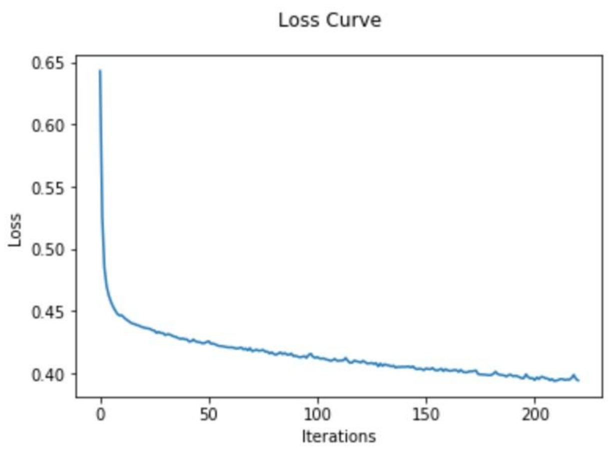
Multilayer Perceptron
The model I trained last is an MLP Classifier. I have used the same attributes as in the previous models which have been noted in the data preparation section. Again, 20 percent of the dataset has been assigned to the test set that will help us validate the accuracy of the model.

The hyperparameters that selected include;

Hidden Layer Sizes to (32, 100) The number of neurons for each hidden layer and the number of hidden layers

Max Iterations to 500	The number of iterations of the solver
Alpha equal to 0.0001	Regularization parameter
Activation to Relu	The activation function for the hidden layers
Solver to Adam	The solver for weight optimization
Verbose to True	Whether to print progress messages to stdout
Shuffle to True	Boolean that defines if to shuffle the data at each iteration
Random State to 21	the seed used by the random number generator
Tol to 0.000000001	Tolerance for the optimization

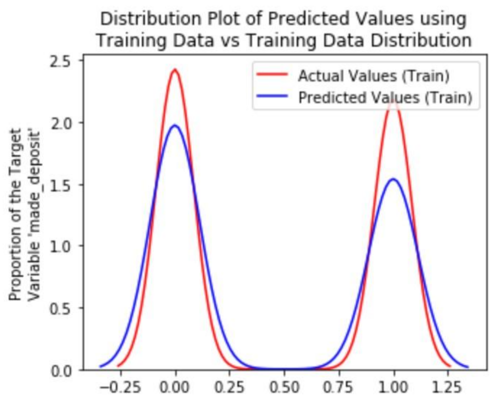
Next, I fitted the model with the training set and made predictions on the testing dataset which revealed the loss according to its iteration time. The models' loss curve is depicted in *Figure 10.*



VI. RESULTS AND ERRORS

The models discussed in the previous sections can help us explore the connection between selected features and predict the target. This section lists the results and compares the accuracies of the predictions made by the models. Furthermore, the representation of the ROC curve of each model will be depicted here. **Logistic Regression**

The distribution plot *(Figure 11)* shows how the model is doing in learning from the training data set. When the model generates new data from the testing dataset we see *(Figure 12)* the distribution of the predicted values is slightly different from the actual target values, however, it seems very close to them. Comparing the figures, the distribution of the data in figure 12 is much better at fitting the data, however, there are small ranges that the opposite is observed.



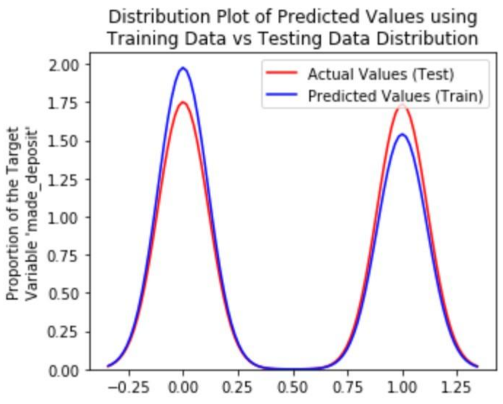


Figure 11. *Logistic Regression; Plot of predicted values* ***Figure 12.*** *Logistic Regression; Plot of predicted values using the training data compared to the training data. using the training data compared to the testing data.*

It turns out that the test data sometimes referred to as the out of sample data is a much better measure of how well your model performs in the real world. One reason for this is overfitting. Overfitting occurs when the model fits the noise, not the underlying process. Therefore when testing the model using the test set, the model does not perform as well as it is modelling noise, not the underlying process that generated the relationship.

Regarding the accuracy of the model, I have utilized the Jaccard index and Log loss metrics, the Jaccard similarity coefficient found equal to 0.771 and the log loss equal to 0.492. Ideal classifiers have progressively smaller values of log loss, meaning lower log loss has better accuracy.

Furthermore, the level of accuracy and errors made by the model is shown in the logistic regression confusion matrix *(Figure 13).*

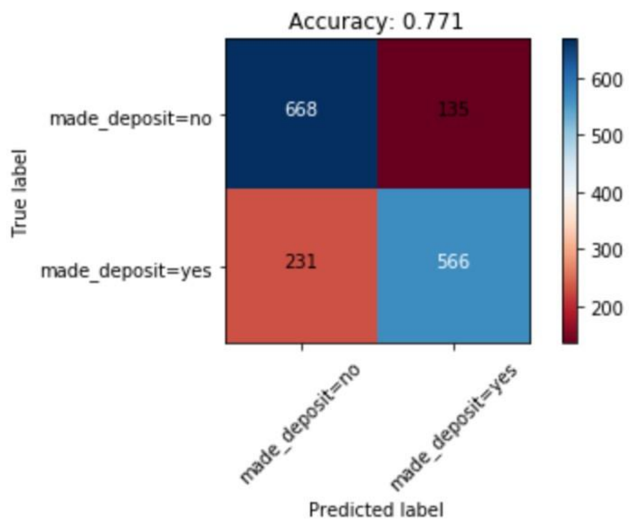
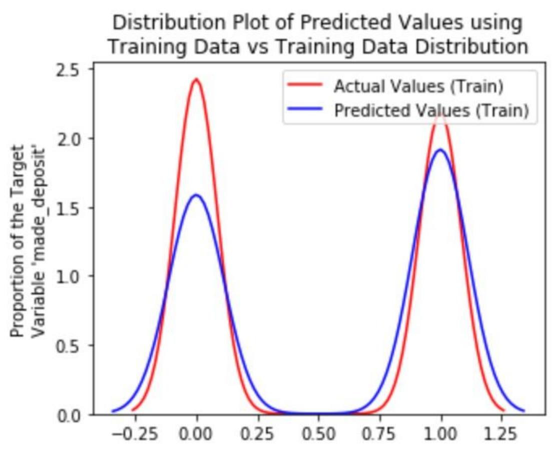


Figure 13. *Logistic Regression Confusion Matrix.*

Decision Tree
The distribution plots, firstly the *Figure 14* depicts how the model is doing in learning from the training data set, and secondly*, Figure 15* shows how the model performs when encounters new data from the testing dataset.



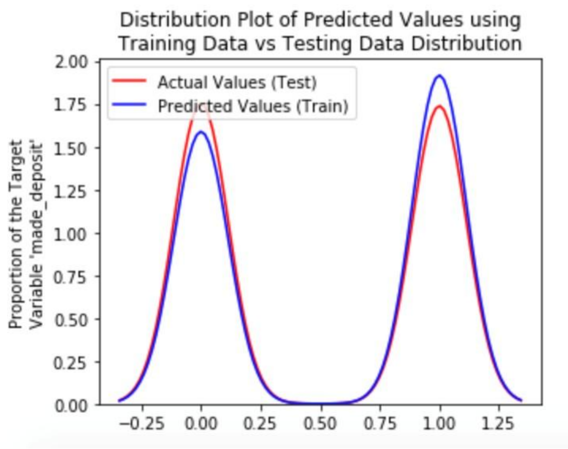


Figure 14. *Decision Tree; Plot of predicted values using the*
Figure 15. *Decision Tree; Plot of predicted values using*
the training data compared to the training data.
training data compared to the test data.

Regarding the accuracy of the model, I have utilized the Jaccard index, F1-score metrics as well as the accuracy score method for comparison purposes. All of them returned the same value of *0.784* approximately.

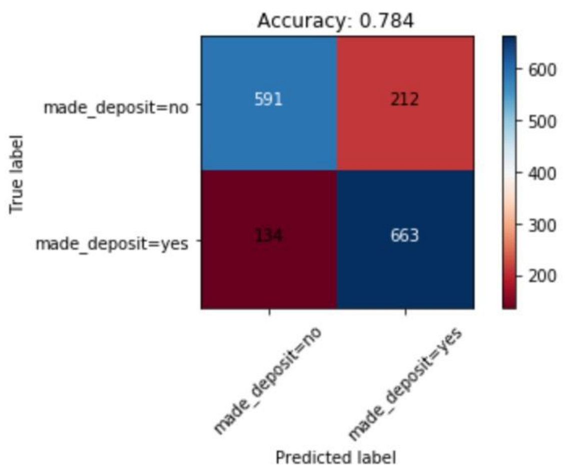
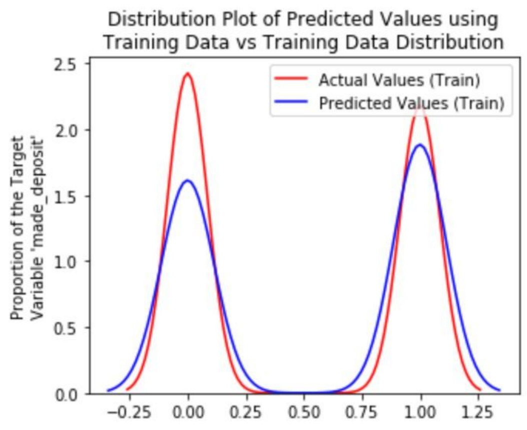


Figure 16. *Decision Tree Confusion Matrix.*

Multilayer Perceptron

Similarly, The distribution plots, firstly the figure 17 depicts how the model is doing in learning from the training data set, and secondly, Figure 18 shows how the model performs when encounters new data from the testing dataset.



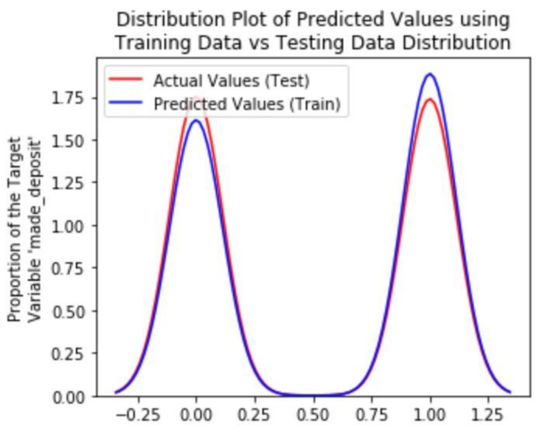


Figure 17. *MLP; Plot of predicted values using the training* ***Figure 18.*** *MLP; Plot of predicted value using the training data compared to the training data. data compared to the test data.*

As far as the accuracy of the model, I have utilized the accuracy score method. The accuracy of the MLP model equals to 0.810. A visualization of the confusion matrix of the model is shown below *(Figure 19).*

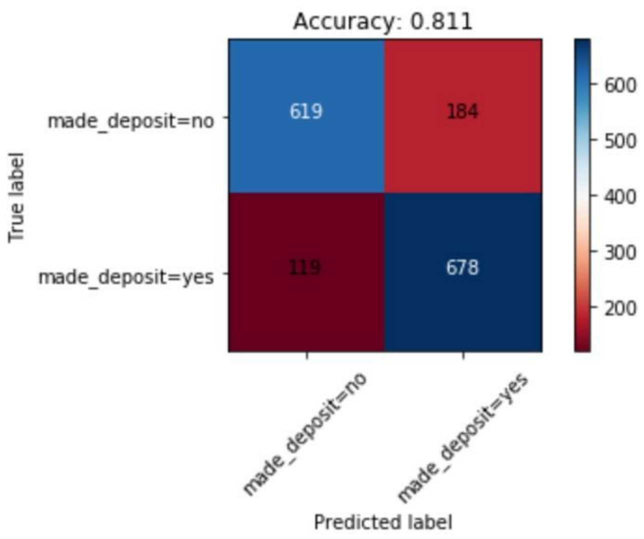


Figure 19. *Multilayer Perceptron Confusion Matrix.*

Receiver Operating Characteristic (ROC) curve is a tool which supports the explanation of probabilistic prediction for binary classification modelling, and here its use is rational since we have roughly equal numbers of instances for each class. The curves show the true positive and false positive rate for every probability threshold and reveal that the most skillful model is the MLP, bowing up to the top left of the plot *(Figure 20), with AUC performance equal to 0.882*.

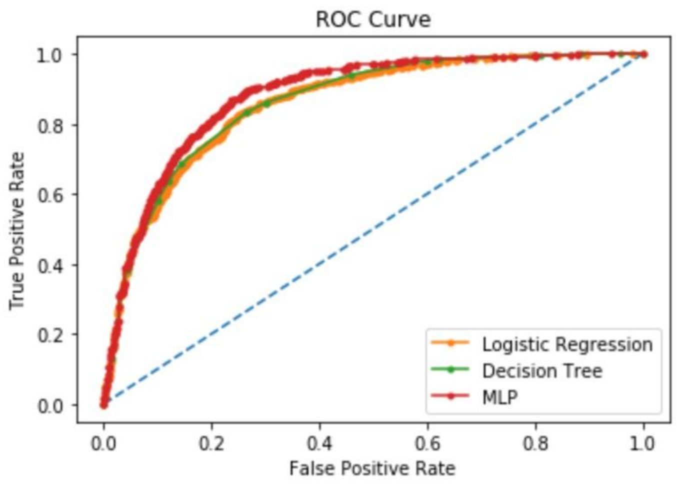


Figure 20. *ROC Curves of the Logistic Regression , Decision Tree and Multilayer Perceptron models. Logistic Regression: AUC=0.860, Decision Tree: AUC=0.861, MLP: ROC AUC=0.882.*

It is important to mention that utilizing a different set of variables or even different classification ML models would give different results and accuracy levels. Moreover, implementing the models with different hyperparameters would probably help in obtaining different results towards the improvement of the predictions. For example, regularization parameters have been used, which is a technique able to solve the overfitting problem in machine learning models. Numerous of alteration in the process I followed would have a different positive or negative effect.

Concluding, overall and all three models taken into consideration, the results suggest that the MLP has better performance than the other classification models since its accuracy is higher. Consequently, with the current data-processing, variables selection, ML models and hyperparameters, the proposed model in order to predict whether a customer will subscribe to the term deposit is the Multilayer Perceptron.

References

[1] R. J. Robles, R. Alcaria, D. M. de Andrés, M. N. de la Cruz, R. Calero, S. Iglesias, and M. Lopez, "An IoT based reference architecture for smart water management processes," *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 6, no. 1, pp. 4–23, 2015.

[2] S. W. Lee, S. Sarp, D. J. Jeon, and J. H. Kim, "Smart water grid: The future water management platform," *Desalination Water Treat.*, vol. 55, no. 2, pp. 339–346, 2015.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[5] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY: Springer, 2009.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[11] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[12] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. New York, NY: Springer, 2013.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[14] S. Raschka, *Python Machine Learning*. Birmingham, UK: Packt Publishing, 2015.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[16] T. Mitchell, *Machine Learning*. New York, NY: McGraw-Hill, 1997.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] A. Ng, "Machine learning yearning," *DeepLearning.ai*, 2018. [Online]. Available: https://www.deeplearning.ai/machine-learning-yearning/

[19] J. D. Kelleher, B. Mac Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics*. Cambridge, MA: MIT Press, 2015.

[20] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. New York, NY: Springer, 2013.