A PROJECT REPORT ON

TERM DEPOSIT SUBSCRIPTION PREDICTION USING DIFFERENT MACHINE LEARNING ALGORITHMS

Submitted towards the fulfilment for the training on:

DATA ANALYTICS, MACHINE LEARNING AND AI USING PYTHON

SUBMITTED BY:

SWAGAT SOURAV

Veer Surendra Sai University of Technology

Burla, Odisha

Guided By:

BIPUL KUMAR SHAHI

CTO, DIGINIQUE TECHLABS

ACKNOWLEDGEMENT

My sincere thanks and gratitude for this project is towards Mr. Bipul Kumar Shahi sir, CTO Diginique Techlabs. Without his support and guidance this report would not have come to its conclusion. His immense patience and constant motivation always prompted me to boost my knowledge in the respective field and its was due to his imparted knowledge that I have completed my project work. He was quite patient in hearing my doubts, however small it may be and always provided us with the conceptual understanding of the relevant things rather than going for the prescribed way. His training helped me to get my concepts really strengthened in field of Machine Learning. His teaching provided me with the concepts and ability to impart the knowledge in the real-world scenario and hence this project was completed. Lastly, I want to thank my parents for their support in whatever I do. I also hereby declare to the best of my knowledge that the project work submitted is unique and is not submitted anywhere else.

DATA SOURCE

This is the classic marketing bank dataset uploaded originally in the UCI Machine Learning Repository. The dataset gives us information about a marketing campaign of a financial institution in which we will have to analyse in order to find ways to look for future strategies in order to improve future marketing campaigns for the bank.

Link:https://www.kaggle.com/gowthamchowdry/bank-classifying-term-deposit-subscriptions/notebook

(The dataset has been modified a little for the uniqueness of this project and to create a unique solution)

IMPACT AREA

A term deposit is a cash investment held at a financial institution. Our money is invested for an agreed rate of interest over a fixed amount of time, or term. The bank has various outreach plans to sell the term deposits to their customers such as email marketing, advertisements, telephonic marketing and digital marketing.

Telephonic marketing still remains one of the most effective way to reach out to the people. However, they require huge investment as large call centres are hired to actually execute these campaigns. Hence, it is crucial to identify the customers most likely to convert beforehand so that they can be specifically targeted via call.

We are provided with the client data such as: age of the client, their job type, their marital status etc. Along with client data we are also provided with the information of the call such as the duration of the call, day and month of the call, etc. Given this

information, our task is to predict if the client will subscribe to the term deposit or not.

Data

We are provided with the following files:

1. train.csv: We use this dataset to train the model. This file contains all the client and the call details as well as the target variable “subscribed”. We have to train the model using this file.

2. test.csv: We use the trained model to predict whether a new set of clients will subscribe to term deposit.

Data Dictionary

Variable	Definition
ID	Unique client ID
age	Age of the client
job	Type of the job
marital	Marital status of the client
education	Education level
default	Credit in default
housing	Housing Loan
loan	Personal Loan
contact	Type of communication
month	Contact month
day_of_week	Day of week of contact
duration	Contact duration
campaign	Number of contacts performed during this campaign to the client

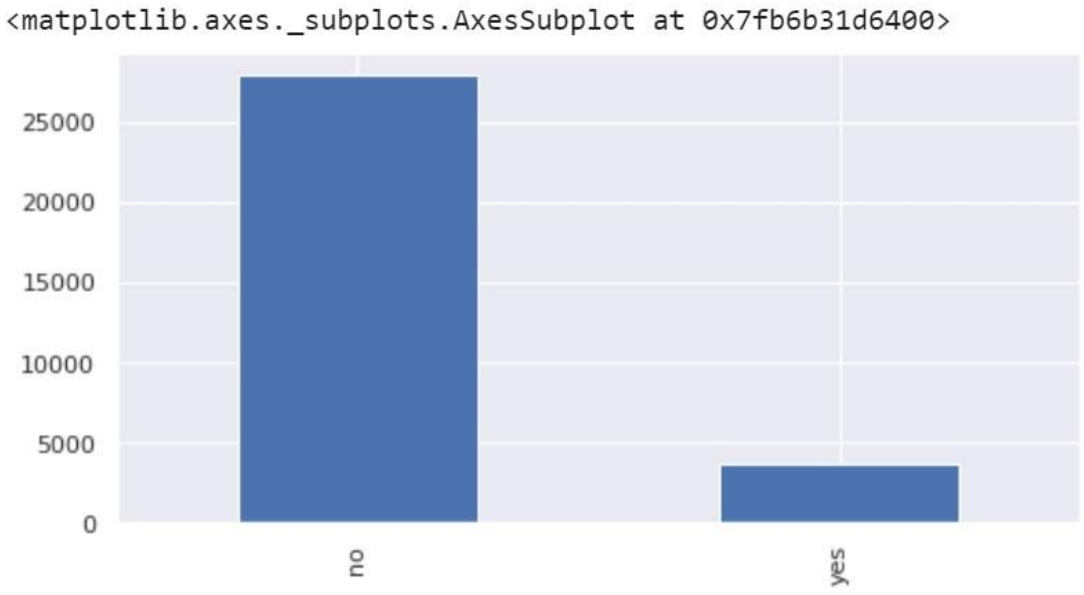
pdays	Number of days that passed by after the client was last contacted
previous	Number of contacts performed before this campaign
poutcome	Outcome of the previous marketing campaign
Subscribed(target)	has the client subscribed to a term deposit

UNIVARIATE ANALYSIS

Univariate analysis refers to the analysis of a single variable and its data distribution present in the dataset. This variable can be the target variable as well as the feature variables. It helps in getting a particular type of an analogy from the data.

Distribution of the Subscribed variable

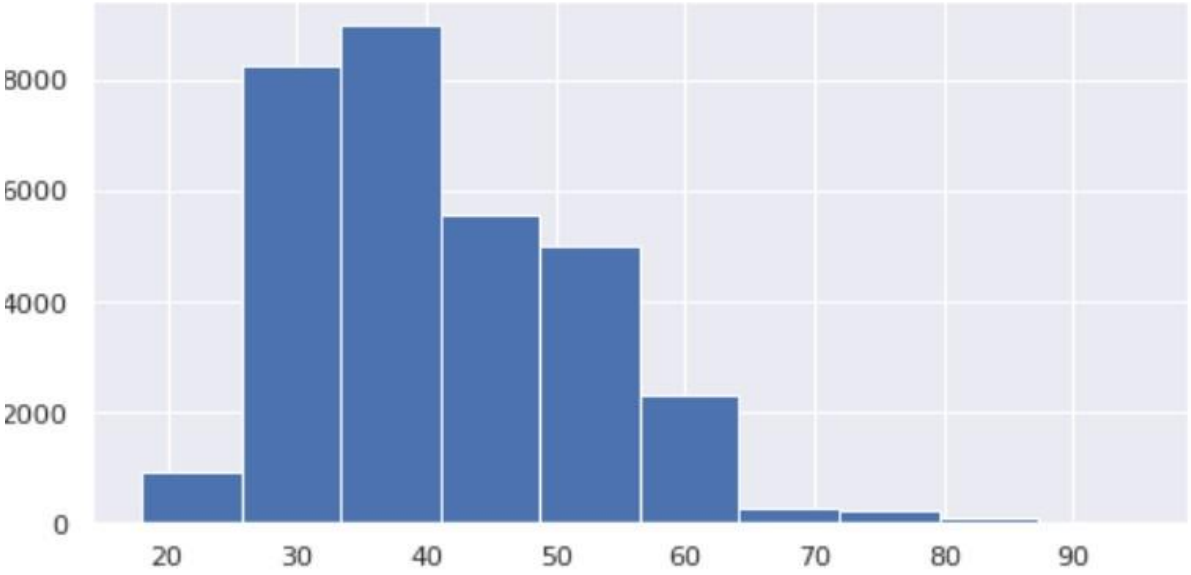
Let’s now see how our target variable “subscribed” is distributed.



So, 3715 users out of total 31647 have subscribed which is around 12%.

Distribution of the Age variable

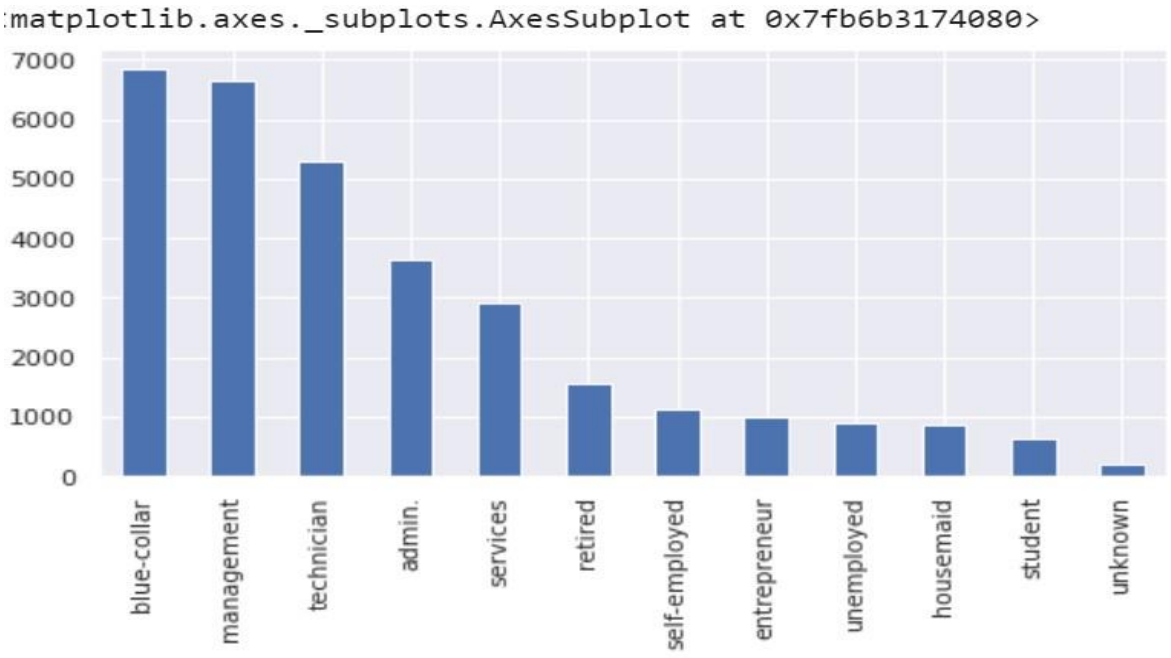
Let's first look at the distribution of age variable to see how many people belongs to a particular age group.



We can infer that most of the clients fall in the age group between 20-60.

Distribution of the different types of jobs

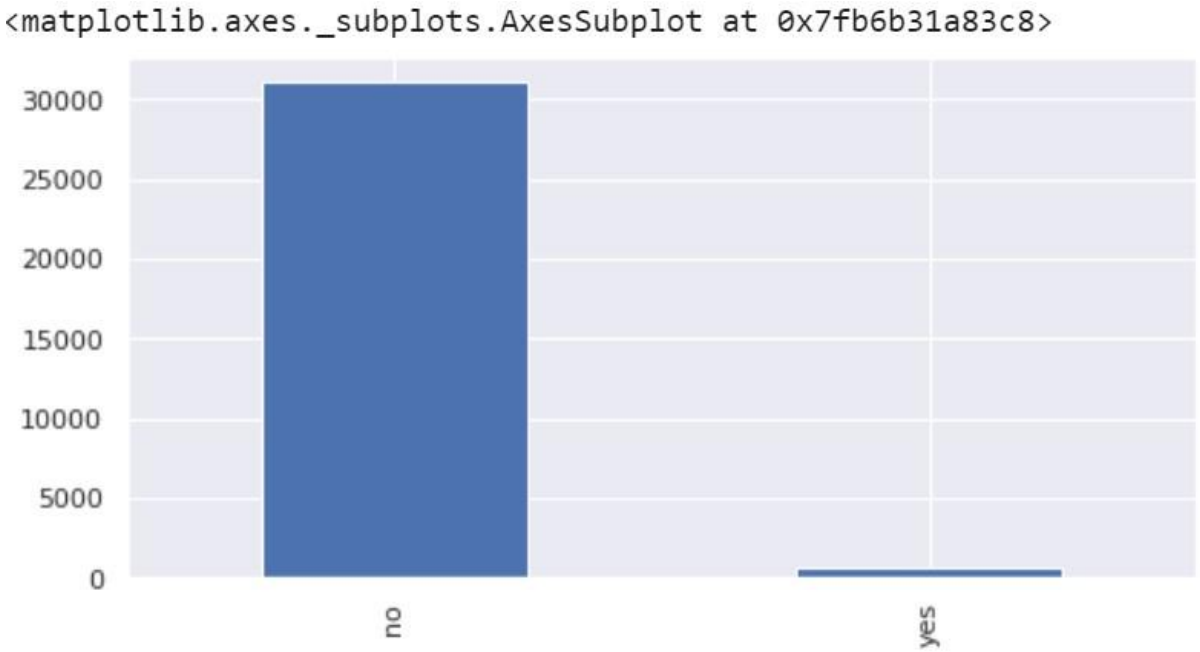
Let’s look at different types of the jobs of the client:



We see that most of the clients belongs to blue-collar job and the students are least in number as students generally do not take a term deposit.

Distribution of the default history

Let's also look at how many clients have default history.



More than 90% of the clients have no default history.

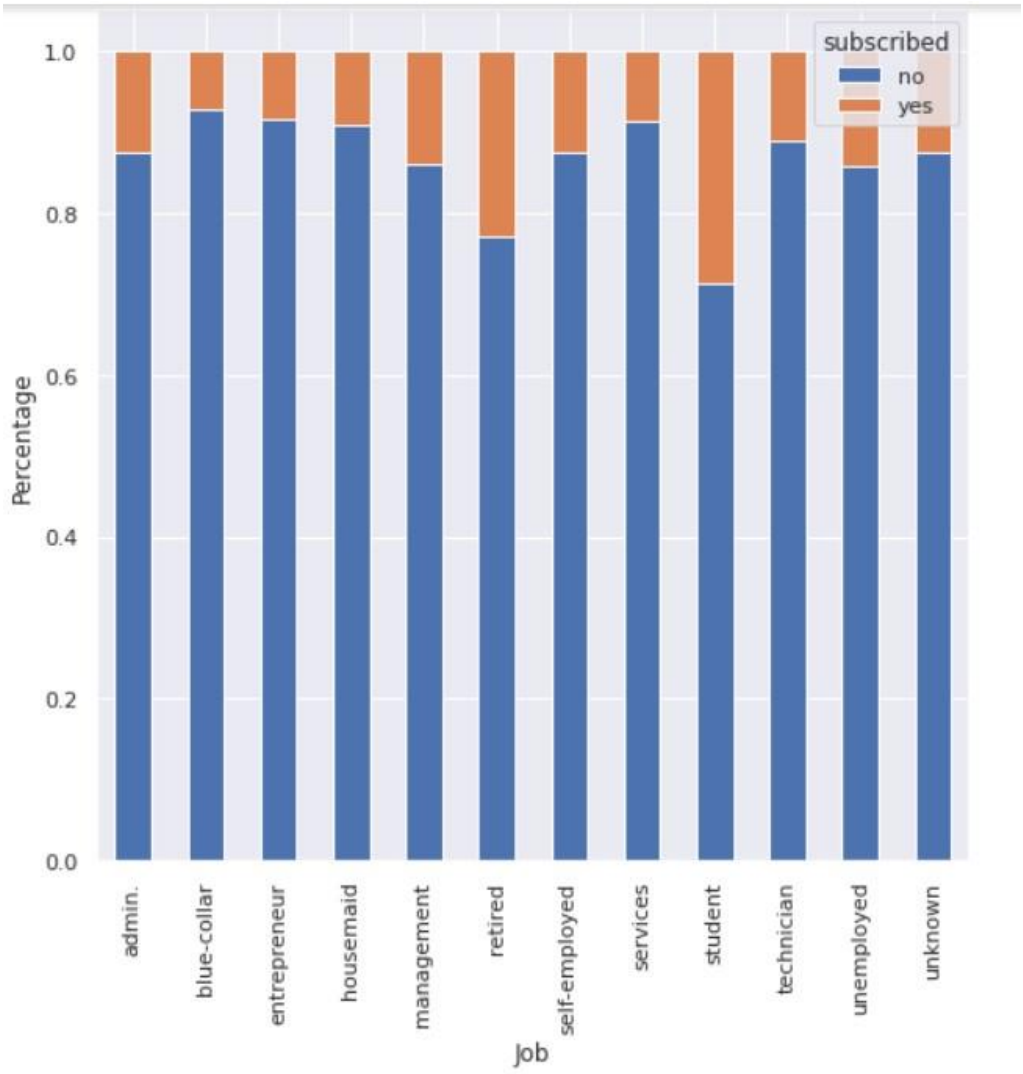
BIVARIATE ANALYSIS

Bivariate analysis refers to the analysis between two variables namely the target variable and one of the feature variables. This helps in decoding how our target variable is dependent on the feature variables.

Bivariate analysis is one of the simplest forms of quantitative analysis. It involves the analysis of two variables, for the purpose of determining the empirical relationship between them. Bivariate analysis can be helpful in testing simple hypotheses of association.

Age vs. Subscription distribution

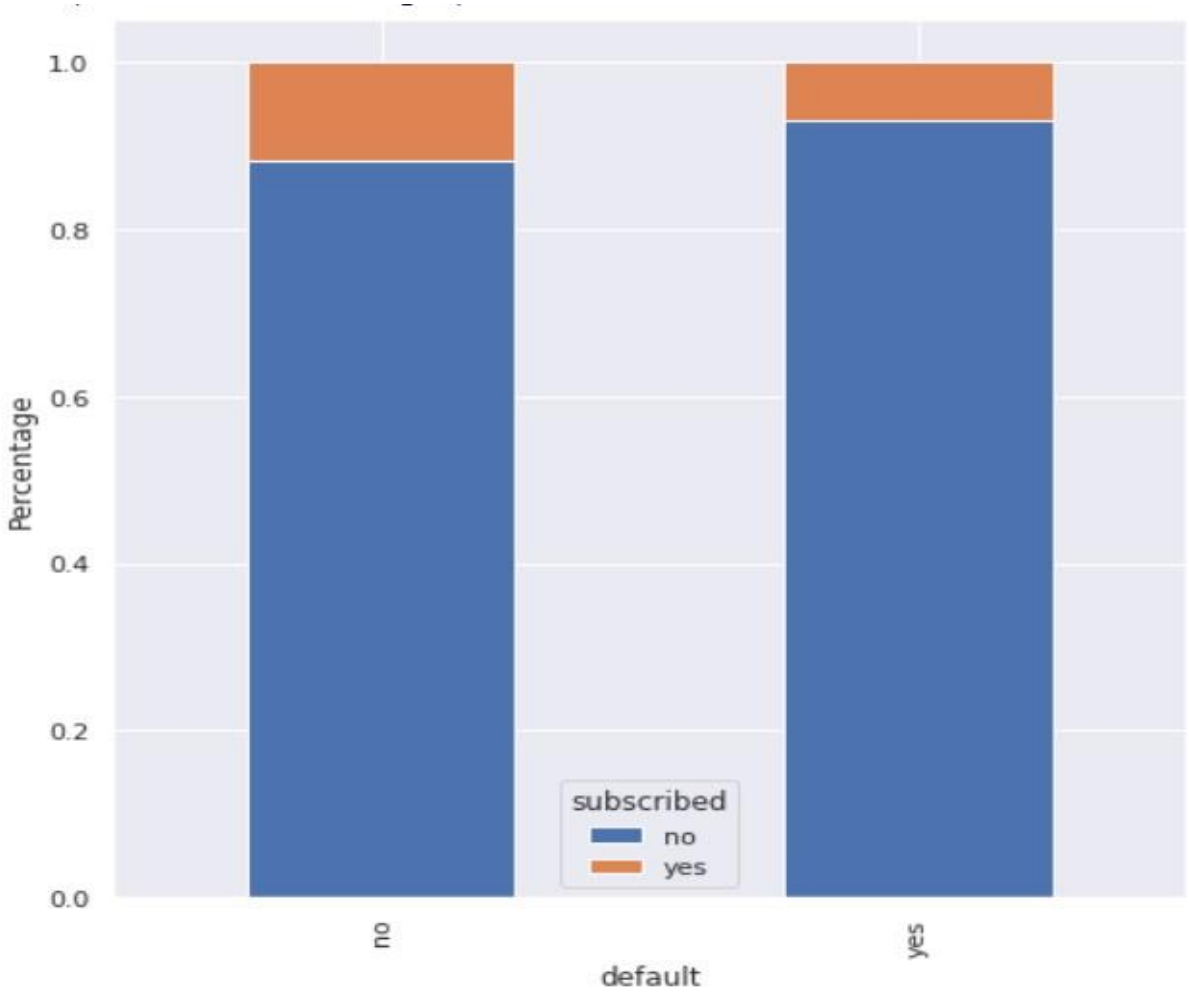
Let’s see how clients belonging to different age groups have subscribed to the term deposit.



From the above graph we can infer that students and retired people have higher chances of subscribing to a term deposit, which is surprising as students generally do not subscribe to a term deposit. The possible reason is that the number of students in the dataset is less and comparatively to other job types, more students have subscribed to a term deposit.

Default vs Subscribed distribution

Now let’s try to explore the default variable against subscribed variable. It will give us the idea of how the defaulters subscribe to the term deposit.



We can infer that clients having no previous default have slightly higher chances of subscribing to a term loan as compared to the clients who have previous default history.

Correlation of different variables

Let's now look at how correlated our numerical variables are. We will see the correlation between each of these variables and the variable which have high negative or positive values are correlated. By this we can get an overview of the variables which might affect our target variable.

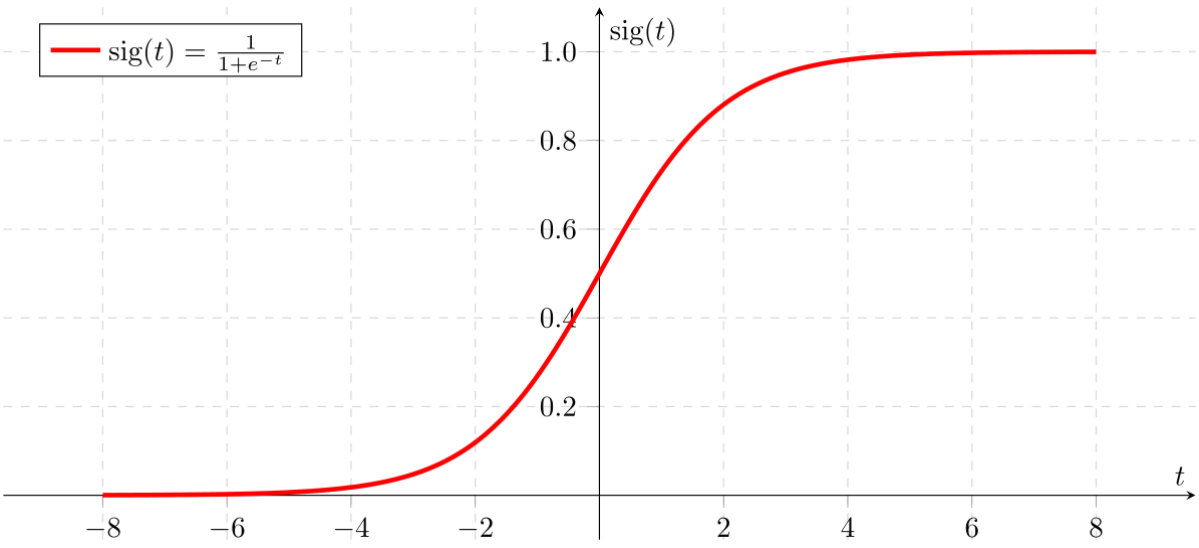
We can infer that duration of the call is highly correlated with the target variable. This can be verified as well. As the duration of the call is more, there are higher chances that the client is showing interest in the term deposit and hence there are higher chances that the client will subscribe to term deposit.

Algorithms Used

Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

Logistic Regression uses a sigmoid function to limit the outputs of the regression model in between 0 and 1. The sigmoid function is given below.



We apply the logistic regression algorithm on the X_train and y_train dataset.

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

On applying this algorithm, we get an accuracy of about 89% on the test datasets (y_val, X_val)

```
# calculating the accuracy score
accuracy_score(y_val, prediction)

0.8913112164296998
```

Decision Tree

Decision Tree algorithm belongs to a family of supervised learning algorithms. The goal of the decision tree is to create a training model that can predict the class or the target variable by learning simple decision rules inferred from the prior data (training dataset).

We fit the datasets X_train and y_train into the decision tree model.

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=4, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

By fitting our datasets into the decision tree model, we get an accuracy of about 90% on the validation datasets.

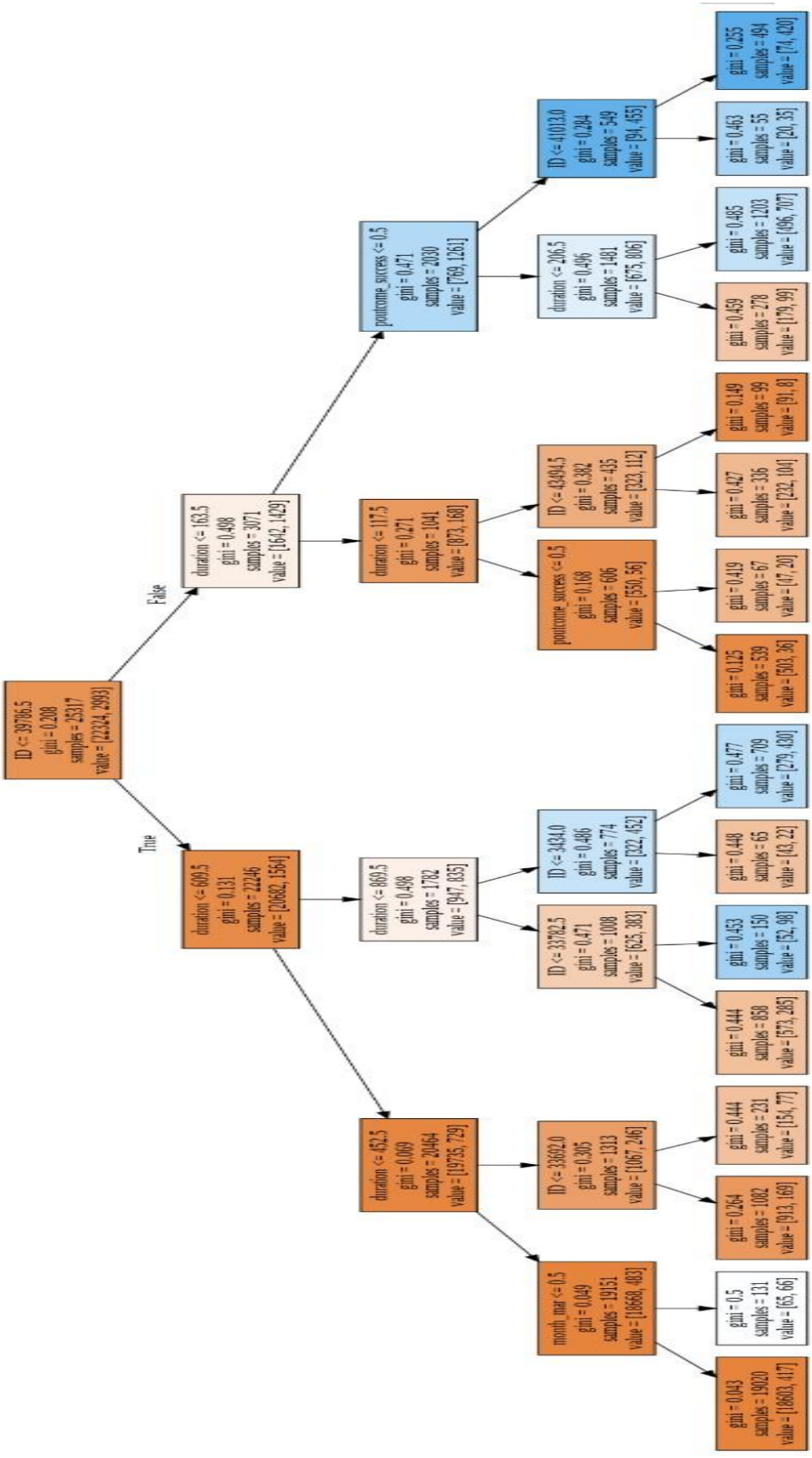
```
[ ]  # making prediction on the validation set
     predict = clf.predict(X_val)
```

```
[ ]  # calculating the accuracy score
     accuracy_score(y_val, predict)
```

```
0.9042654028436019
```

We can try to improve the model's accuracy by tuning in the hyperparameters that include specifying the maximum depth of the tree, minimum number of elements at the leaf node and minimum samples for the split.

Visualising the Decision Tree



Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

We fit the datasets X_train and y_train into the random forest model.

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

We then compute the accuracy of our model on both the testing and training datasets.

```
[ ] # Printing the train and test accuracy
    print(rmodel.score(X_train,y_train))
    print(rmodel.score(X_val,y_val))
```

```
0.9930481494647865
0.9011058451816746
```

We see that we have got an accuracy of around 99% on the training dataset which states that the model would behave as an overfit model in case of the training data. But on the testing data we get the accuracy of about 90% which is similar to the accuracy got in the decision tree classifier. From the above observations we can conclude that decision tree would be a better model in the prediction of the people who will subscribe to the term deposit.

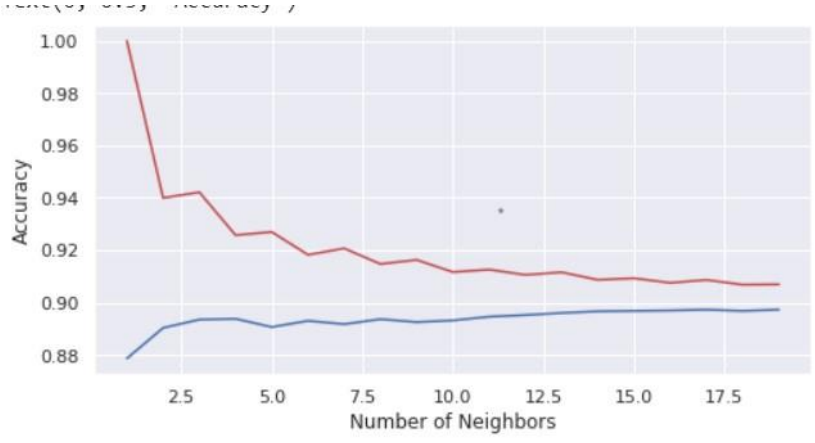
K-Neighbours Classifier

The K-nearest neighbours (KNN) algorithm is a type of supervised machine learning algorithms. KNN is extremely easy to implement in its most basic form, and yet performs quite complex classification tasks. KNN is a non-parametric learning algorithm, which means that it doesn't assume anything about the underlying data. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

We fit the X_train and y_train datasets into KNN model and observe the accuracy.

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                     weights='uniform')
```

The KNN algorithm consists of the hyperparameter of the number of nearest neighbours (n_neighbors). We need to find the optimal number of neighbours so that our model gives us higher amount of accuracy and at the same time we should not also select a greater number of neighbours because that would increase the amount of computational power required to compute the result through the algorithm. We will observe the optimal number of neighbours through a graph of testing accuracy and training accuracy with the number of neighbours.



From the above graph we can interpret that the distance between the test accuracy and train accuracy keeps on decreasing as we move from lower number of neighbours to higher number of neighbours but the test accuracy almost remains constant on the test dataset. Besides that, implementing higher number of neighbours require more computational power. So, from this we can infer that although test accuracy almost remains constant, the train accuracy decreases and optimal no of neighbours can be taken as 6.

Neural Network

Artificial neural networks or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules

In our consideration we have defined 2 layers excluding the input layer. The input layer consists of all the training examples which are more than 31,000. The 2^{nd} layer is the hidden layer in our neural network which consists of 3 hidden units and finally the 3^{rd} layer consists of the output layer which consists of 2 units. The 2 units are for predicting whether the term deposit will be subscribed or not (0/1).

We use the stochastic gradient descent as our optimiser and the cross entropy to calculate our loss. We also use the evaluation metric to see our accuracy.

We fit the X_train and y_train datasets into our neural network and observe the values of the accuracy and the cost.

```
[113] # Train your model
      model.fit(np.array(X_train),np.array(y_train),epochs=10)
```

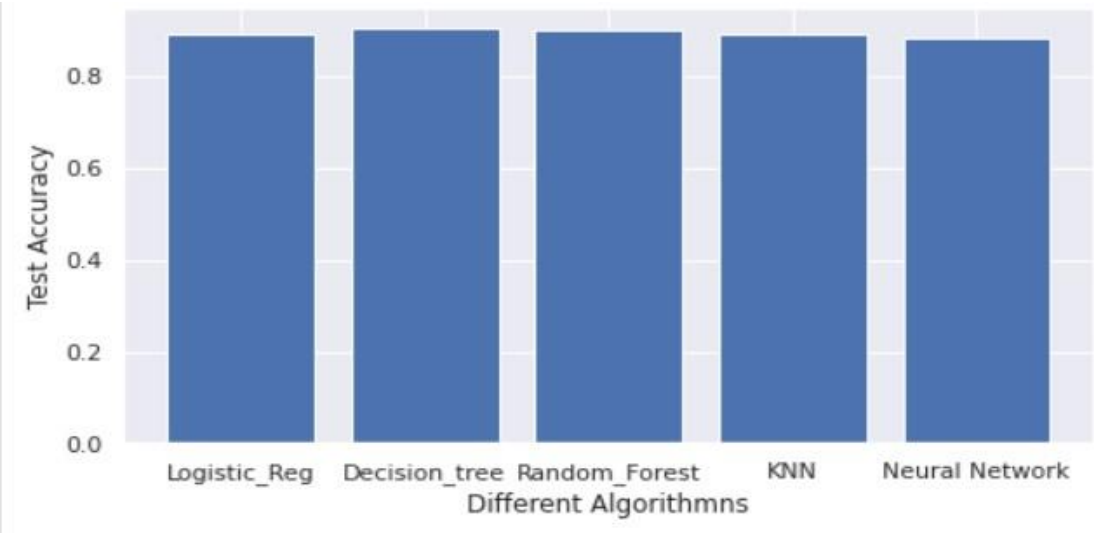
```
Epoch 1/10
792/792 [==============================] - 1s 1ms/step - loss: 0.4273 - accuracy: 0.8380
Epoch 2/10
792/792 [==============================] - 1s 1ms/step - loss: 0.3634 - accuracy: 0.8818
Epoch 3/10
792/792 [==============================] - 1s 1ms/step - loss: 0.3634 - accuracy: 0.8818
Epoch 4/10
792/792 [==============================] - 1s 1ms/step - loss: 0.3634 - accuracy: 0.8818
Epoch 5/10
792/792 [==============================] - 1s 1ms/step - loss: 0.3634 - accuracy: 0.8818
Epoch 6/10
792/792 [==============================] - 1s 1ms/step - loss: 0.3634 - accuracy: 0.8818
Epoch 7/10
792/792 [==============================] - 1s 1ms/step - loss: 0.3635 - accuracy: 0.8818
Epoch 8/10
792/792 [==============================] - 1s 1ms/step - loss: 0.3634 - accuracy: 0.8818
Epoch 9/10
792/792 [==============================] - 1s 1ms/step - loss: 0.3634 - accuracy: 0.8818
Epoch 10/10
792/792 [==============================] - 1s 1ms/step - loss: 0.3634 - accuracy: 0.8818
<tensorflow.python.keras.callbacks.History at 0x7fb6b1ad1240>
```

Next, we try to predict the accuracy on the testing dataset.

```
[ ] # Evaluate the model
    model.evaluate(np.array(X_val),np.array(y_val))
```

```
198/198 [==============================] - 0s 1ms/step - loss: 0.3551 - accuracy: 0.8859
[0.3550862967967987, 0.8859399557113647]
```

Selecting the best algorithm



From the various algorithms that we have considered we can infer that Decision Tree has the highest accuracy on the both training and testing data. So our predicted values will be calculated through that model.

Solution Checker

We can use solution_checker.xlsx to generate score (accuracy) of our
predictions.

This is an excel sheet where we are provided with the test IDs and we
have to submit your predictions in the “subscribed” column.
Below are the steps to submit your predictions and generate score:

a. Save the predictions on test.csv file in a new csv file (submission.csv).

b. Open the generated csv file, copy the predictions and paste them in the subscribed column of solution_checker.xlsx file.

c. Your score will be generated automatically and will be shown in **Your Accuracy Score** column.

REFERENCES

1. https://www.kaggle.com/
2. https://towardsdatascience.com/
3. https://anaconda.org/anaconda/pydotplus
4. https://keras.io/
5. https://www.tensorflow.org/