Table of Contents

Abstract

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).Total have 21columns of features, the last column: term deposit (target variable)

Key words:

age:numerical
job:categorical
education:categorical
marital: categorical
default: categorical
housing: categorical
loan: categorical
contact: categorical
month: categorical
day_of_week: categorical
duration: numerical
campaign: numerical
pday: numerical
previous:numerical
poutcome:categorical
emp.var.rate:numerical
cons.price.idx:numerical
cons.conf.idx:numerical
euribor3m:numerical
nr.employed: numerical
term_deposit: binary(target)

1.0 Introduction

With the increasing power of computer technology, companies and institutions cannowadays store large amounts of data at reduced cost. The amount of available datais increasing exponentially and cheap disk storage makes it easy to store data thatpreviously was thrown away. There is a huge amount of information locked up indatabases that is potentially important but has not yet been explored. The growing sizeand complexity of the databases makes it hard to analyse the data manually, so it isimportant to have automated systems to support the process. Hence there is the needof computational tools able to treat these large amounts of data and extract valuableinformation.

In this context, Data Mining provides automated systems capable of processinglarge amounts of data that are already present in databases. Data Mining is used toautomatically extract important patterns and trends from databases seeking regularitiesor patterns that can reveal the structure of the data and answer business problems. DataMining includes learning techniques that fall into the field of Machine learning. Thegrowth of databases in recent years brings data mining at the forefront of new businesstechnologies.

A key challenge for the insurance industry is to charge each customer an appropriate price for the risk they represent. Risk varies widely from customer to customer and a deep understanding of different risk factors helps predict the likelihood and cost of insurance claims. The goal of this program is to see how well various statistical methods perform in predicting auto Insurance claims based on the characteristics of the driver, vehicle and driver / vehicle coverage details.

A number of factors will determine BI claims prediction among them a driver's age, past accident history, and domicile, etc. However, this contest focused on the relationship between claims and vehicle characteristics well as other characteristics associated with the auto insurance policies.

1.1. What are the different types of Machine Learning?

There three types of machine learning:

- SUPERVISED MACHINE LEARNING
- UNSUPERVISED MACHINE LEARNING
- REIFORCEMENT LEARNING

- **SUPERVISED MACHINE LEARNING:**

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher. Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**. In the real-world, supervised learning can be used for **Risk Assessment, Image classification, Fraud Detection, spam filtering**, etc. There are two types of Supervised Learning techniques: Regression and Classification. It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.

- **UNSUPEREVISED MACHINE LEARNING:**

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things.

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.

The unsupervised learning algorithm can be further categorized into two types of problems:

- Clustering
- Association

Below is the list of some popular unsupervised learning algorithms:

- **K-means clustering**
- **KNN (knearest neighbours)**
- **Hierarchal clustering**

REINFORCEMENT LEARNING

Reinforcement learning is a machine learning training method based on rewarding desired behaviours and/or punishing undesired ones. In general, a reinforcement learning agent is able to perceive and interpret its environment, take actions and learn through trial and error.

1.2. Benefits of Using Machine Learning in Bank Marketing:

- Greater Automation and Improved Productivity
- Personalized Customer Service More precise Risk Assessment.
- Advanced Fraud Detection and Prevention.
- Improving Customer Experience.
- Chatbots.
- Personalized Offers.
- Customer Retention.

1.3. About Industry

Bank marketing is known for its nature of developing a unique brand image, which is treated as the capital reputation of the financial academy. It is very important for a bank to develop good relationship with valued customers accompanied by innovative ideas which can be used as measures to meet their requirements.

Customers expect quality services and returns. There are good chances that the quality factor will be the sole determinant of successful banking corporations. Therefore, Indian banks need to acknowledge the imperative of proactive Bank Marketing and Customer Relationship Management and also take systematic steps in this direction.

The banking industry provides different types of banking and allied services to its clients. Bank customers are mostly people and enterprises that have surplus or lack of funds and those who require various types of financial and related services. These customers are from different strata of the economy, they belong to different geographical regions, areas and are into different professions and businesses.

It is quite natural for the requirement of each individual group of customers to be unique from the requirements of other groups. Thus, it is important to acknowledge distinct homogenous groups and even sub-groups of customers, and then with maximum precision conclude their requirements, design schemes to suit their particular requirements, and deliver them most efficiently.

Basically, banks engage in transaction of products and services through their retail outlets known as branches to different customers at the grassroots level. This is referred as the 'top to bottom' approach. It should be 'bottom to top' approach with customers at the grassroots level as the target point to work out with different products or schemes to match the requirements of various homogenous groups of customers. Hence, bank marketing approach, is considered as a group or "collective" approach.

1.3.1 AI / ML Role in term deposit prediction

Machine Learning is a sub-set of artificial intelligence where computer algorithms are used to autonomously learn from data. Machine learning (ML) is getting more and more attention and is becoming increasingly popular in many other industries. Within the insurance industry, there is more application of ML regarding the prediction.

2.0 Term Deposit prediction

Term deposits are a major source of income for a bank. A term deposit is a cash investment held at a financial institution. Your 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 term deposits to their customers such as email marketing, advertisements, telephonic marketing and digital marketing. Telephonic marketing campaigns still remain one of the most effective ways to reach out to 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. You are provided with the client data such as age of the client, their job type, their marital status, etc.

2.1. Internship Project - Data Link

The internship project data has taken from UCI machine learning repository and the link is https://archive.ics.uci.edu/ml/datasets/Bank+Marketing#

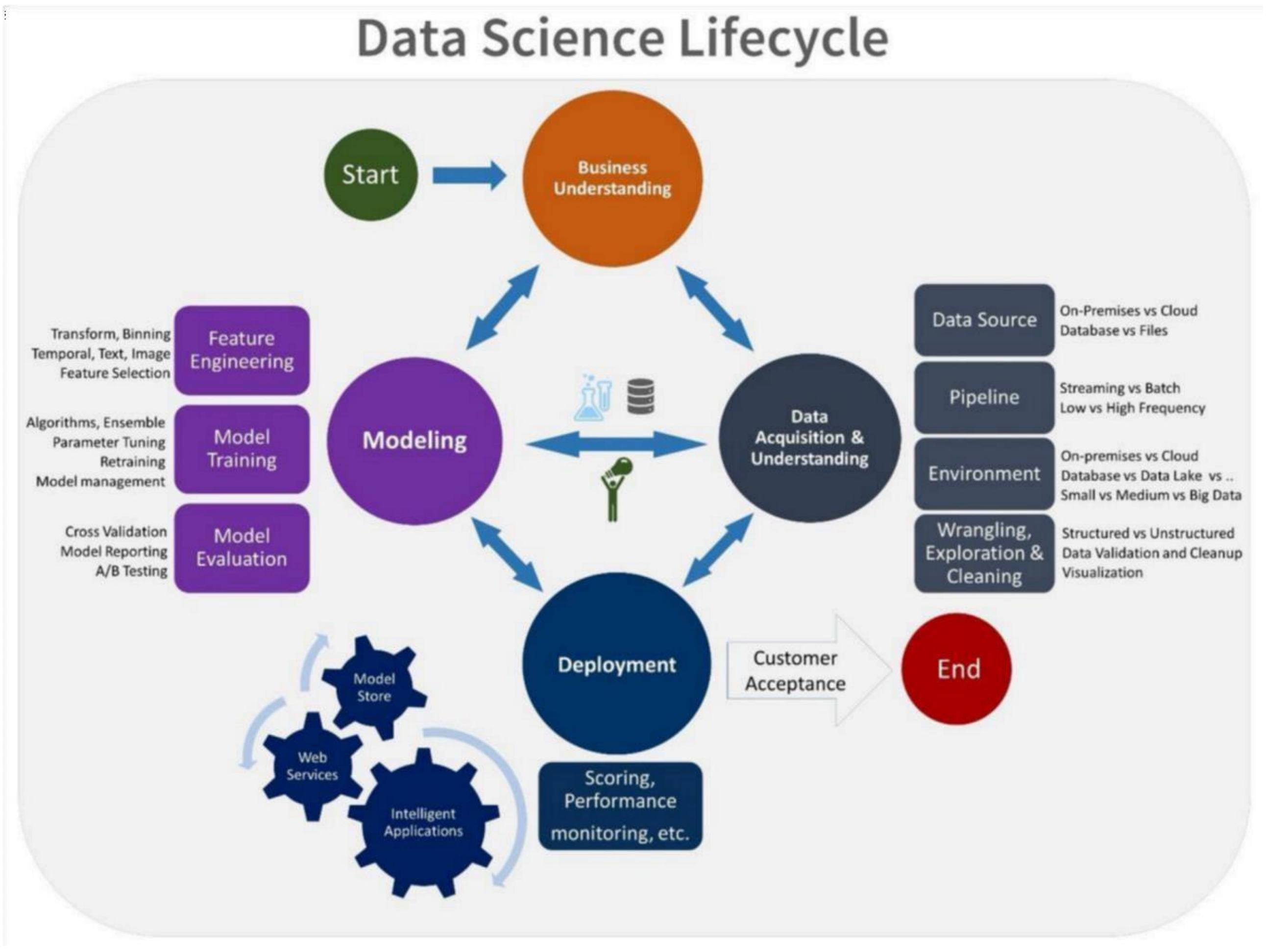
3.0 AI / ML Modelling and Results

3.1. Your Problem of Statement

You have been provided with an Excel dataset that has 21 columns and 41189 rows. Our task is to analyze the dataset and predict if the client will subscribe a term deposit or not by developing a supervised machine learning model.

3.2. Data Science Project Life Cycle

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data.



3.2.1 Data Exploratory Analysis

Exploratory data analysis has been done on the data to look for relationship and correlation between different variables and to understand how they impact or target variable.

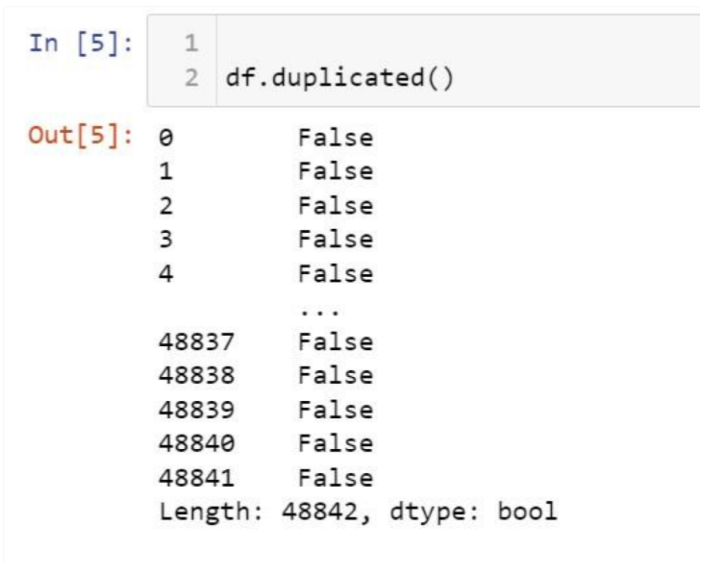
The exploratory analysis is done for term deposit prediction with different parameters and all the charts are presented in **Appendices 6.2 - List of charts (6.2.1 to 6.2.5)**

3.2.2 Data Pre-processing

We removed variables which does not affect our target variable as they may add noise and also increase our computation time.We checked the data for anomalous data points and outliers.We did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

1. Check the Duplicate and low variation data

No duplicates found in the data set.



```
In [5]:
df.duplicated()

Out[5]: 0        False
        1        False
        2        False
        3        False
        4        False
                 ...
        48837    False
        48838    False
        48839    False
        48840    False
        48841    False
        Length: 48842, dtype: bool
```

2 Identify and address the missing variables

No Missing values are identified in the data set

3. Handling of Outliers

4. Categorical data and Encoding Techniques

As we know categorical variables represent types of data which may be further divided into groups.

Categorical variables in the data set are:

job:categorical
education:categorical
marital: categorical
default: categorical
housing: categorical
loan: categorical
contact: categorical
month: categorical
day_of_week: categorical
poutcome: categorical
term_deposit: binary(target)

The Encoding technique used in the code of this project is LabelEncoder.

5. Feature Scaling

It is important to bring the data to a common scale, since there are certain columns with very small values and some columns with high values. This process is important as values on a similar scale allow the model to learn better.

For scaling purpose MinMaxdscalar technique is used for term deposit prediction .

3.2.3 Selection of Dependent and Independent variables

On observing the data it is clear that except from term deposit attribute remaining all independent variables hence all independent variables are assigned to variable “X” and target or dependent variable term deposit is assigned to variable”y”.

3.2.4 Data Sampling Methods

The data we have is highly unbalanced data so we used some sampling methods which are used to balance the target variable so we our model will be developed with good accuracy and precision. Simple random sampling method is used.

1. Simple random sampling

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the dataset which can occur if data is selected manually without randomizing the dataset.

We used this method to split the dataset into train dataset which contains 70% of the total data and test dataset with the remaining 30% of the data.

3.2.5 Models Used for Development

We built our predictive models by using the following first three algorithms and remainng can be applied future for more better accuracy.

1. MODEL 01: Logistic regression

Logistic uses logit link function to convert the likelihood values to probabilities so we can get a good estimate on the probability of a particular observation to be positive class or negative class.The also gives us p-value of the variables which tells us about significance of each independent variable.

2. MODEL 02: Random forest

Random forest is an algorithm that consists of many decision trees. It was first developedby Leo Breiman and Adele Cutler. The idea behind it is to build several trees,to have the instance classified by each tree, and to give a "vote" at each class.The model uses a "bagging" approach and the random selection of features to build acollection of decision trees with controlled variance. The instance's class is to the classwith the highest number of votes, the class that occurs the most within the leaf in whichthe instance is placed.

The error of the forest depends on:

- Trees correlation: the higher the correlation, the higher the forest error rate.
- The strength of each tree in the forest. A strong tree is a tree with low error. By using trees that classify the instances with low error the error rate of the forest decreases.

3. MODEL 03: Decision tree

Artificial neural networks can theoretically solve any problem. ANNs can identify hidden patterns between the variables and can find how different combinations of variables can affect the target variable. The error correction is done by gradient descent algorithm which can reduce the error rate as much as possible for the given data

.

4. MODEL 04: KNN

Model description is one of the only simplest ML models and complies with supervised learning as well. It predicts the similarities between the data for which we are predicting the class and the existing classes and at the end; it put the new case into the category of the class which is very similar to the record or data. It can be used in regression and in classification as well. However, it is mostly used in the classification. Apart from that. It is also a Lazy learner as it is not learning or using a training dataset for memorization, it actually starts performing as soon as it gets data which it needs to predict and that is why it is not using any memory. It is also good at outliers for example here the new data point can be considered as the outlier and can be easily classified into the class A as we are using KNN here. Talking about the advantages

First of all it is very easy to implement Data can be added at any time as it is a lazy learner It requires no training period so it is time effective.

5. MODEL 05 : SVM

The Support Vector Machine (SVM) is a supervised machine learning technology that may be used to handle classification and regression issues. It is, however, mostly used in classification difficulties. Each data item is represented as a point in n-dimensional space, with the value of each feature being the value of a specific coordinate in the SVM algorithm. Then, we achieve classification by selecting the hyper-plane that best separates the two classes. Individual observation coordinates are utilised to compute support vectors. The SVM classifier is a frontier that best differentiates between the two classes (hyper-plane/line

6. MODEL 06: Extreme Gradient Boosting

The XGBoost (eXtreme Gradient Boosting) approach is well-known and successful. Gradient boosting is a supervised learning approach that combines estimates from a series of simpler and weaker models to properly predict a target variable. Because of its strong handling of a wide range of data kinds, relationships, and distributions, as well as the huge range of hyperparameters that can be fine-tuned, the XGBoost technique performs well in machine learning issues. XGBoost iscapable of dealing with regression, classification (binary and multiclass), and ranking issues

7. MODEL 07: CatBüst Classifier

CatBoost means Categorical Boosting because it is designed to work on categorical data flawlessly. If you have Categorical data in your dataset,

Here are some features of the CatBoost, which makes it stand apart from all the other Boosting Algorithm.

- High Quality without parameter tuning
- Categorical Features support
- The fast and scalable GPU version
- Improved accuracy by reducing overfitting
- Fast Predictions
- Works well with less data

8. MODEL 08:LGBM

Light GBM may be a fast, distributed high-performance gradient boosting framework supported decision tree algorithm, used for ranking, classification and lots of other machine learning tasks. Since it's supported decision tree algorithms, it splits the tree leaf wise with the simplest fit whereas other boosting algorithms split the tree depth wise or level wise instead of leaf-wise. So when growing on an equivalent leaf in Light GBM, the leafwise algorithm can reduce more loss than the level-wise algorithm and hence leads to far better accuracy which may rarely be achieved by any of the prevailing boosting algorithms. Also, it's surprisingly in no time , hence the word 'Light'

9. MODEL 09: EXTRA TREES

Extra trees classifier is type of ensemble learning techniques that aggregates the classification results of several non-correlated decision trees gathered in "Forest" to obtain its classification results. It is conceptually very similar to Random Forest Classifier and varies mostly in the manner the decision trees in the forest are formed The Extra Trees Forest's Decision Trees are constructed from training samples. Then, at each test node, each tree is given random samples of k feature from the feature sets. From which each of the decision tree must select the best features to divide data using important mathematical criterion. This random selection of features results in the construction of several de correlated decision trees.

10. MODEL 10: Naïve bias

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem. Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

There are three event models:

Multivariate Bernoulli Event Model Multivariate Event Model Gaussian

Naive Bayes classification

3.3 AI / ML ModelsAnalysis and Final Results

We used our train dataset to build the above models and used our test data to check the accuracy and performance of our models.

We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model for given auto dataset of size ~ 48842 policies. 3.3.1

3.3.1 Different Model codes

- The Python code for models with simple random sampling technique as follows:

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X, y,test_size=0.3,random_state=42)
```

3.3.2 Random Forest Python Code

The Python code for models with simple random sampling technique as follows:

```
from sklearn.ensemble import RandomForestClassifier


ModelRF  =  RandomForestClassifier(n_estimators=100,  criterion='gini',
max_depth=None, min_samples_split=2,                              min_samples_leaf=1,
min_weight_fraction_leaf=0.0,  max_features='sqrt',
max_leaf_nodes=None,    min_impurity_decrease=0.0,  bootstrap=True,
oob_score=False,                              n_jobs=None,  random_state=None,
     verbose=0, warm_start=False, class_weight=None,
ccp_alpha=0.0, max_samples=None)



# Train the model with train data


ModelRF.fit(X_train,y_train)


# Predict the model with test data set


y_pred = ModelRF.predict(X_test) y_pred_prob =
ModelRF.predict_proba(X_test)



# Confusion matrix in sklearn


from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report



# actual values


actual = y_test


# predicted values


predicted = y_pred

# confusion matrix


matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None,
normalize=None)    print('Confusion matrix : \n', matrix)


# outcome values order in sklearn


tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)   print('Outcome
values : \n', tp, fn, fp, tn)
```

```
# classification report for precision, recall f1-score and accuracy

C_Report = classification_report(actual,predicted,labels=[1,0])

print('Classification report : \n', C_Report)
# calculating the metrics

sensitivity = round(tp/(tp+fn), 3); specificity =   round(tn/(tn+fp), 3); accuracy =

round((tp+tn)/(tp+fp+tn+fn), 3); balanced_accuracy =
round((sensitivity+specificity)/2, 3); precision =
round(tp/(tp+fp), 3); f1Score = round((2*tp/(2*tp + fp + fn)), 3);

# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to+1.

# A model with a score of +1 is a perfect model and -1 is a poor model        from

math import sqrt

mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn) MCC = round(((tp* tn)
- (fp * fn)) / sqrt(mx), 3)

print('Accuracy :', round(accuracy*100, 2),'%') print('Precision :',
round(precision*100, 2),'%') print('Recall :', round(sensitivity*100,2), '%')  print('F1
Score :', f1Score) print('Specificity or True Negative Rate :',
round(specificity*100,2), '%'  ) print('Balanced Accuracy :',
round(balanced_accuracy*100, 2),'%') print('MCC :', MCC)

# Area under ROC curve

from sklearn.metrics import roc_curve, roc_auc_score

print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))

# ROC Curve

from sklearn.metrics import roc_auc_score from sklearn.metrics import   roc_curve
logit_roc_auc = roc_auc_score(y_test, y_pred) fpr, tpr, thresholds  =
roc_curve(y_test,ModelRF.predict_proba(X_test)[:,1]) plt.figure()

#------------------------------------------------------------------- plt.plot(fpr, tpr, label= 'Classification
Model' % logit_roc_auc) plt.plot([0, 1], [0, 1],'r--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate') plt.title('Receiver operating
```

characteristic') plt.legend(loc="lower right") plt.savefig('Log_ROC') plt.show()   print('-----
--')

3.3.3 Decision Trees Python code

 The Python code for models with simple random sampling technique as follows: # To build the 'Decision Tree' model with random sampling

```
from sklearn.tree import DecisionTreeClassifier

dfDT = DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
max_depth=None, max_features=None,
                              max_leaf_nodes=None, min_impurity_decrease=0.0,
min_samples_leaf=1,   min_samples_split=2,min_weight_fraction_leaf=0.0,

                              random_state=None, splitter='best')

# Train the model with train data

dfDT.fit(X_train,y_train)

# Predict the model with test data set

y_pred = dfDT.predict(X_test) y_pred_prob = dfDT.predict_proba(X_test)

# Confusion matrix in sklearn

from sklearn.metrics import confusion_matrix from  sklearn.metrics  import
classification_report

# actual values

actual = y_test

# predicted values
```

```
predicted = y_pred

# confusion matrix

matrix   =    confusion_matrix(actual,predicted,   labels=[1,0],sample_weight=None,
normalize=None)  print('Confusion matrix
: \n', matrix)

  # outcome values order in sklearn

tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)   print('Outcome
values : \n', tp, fn, fp, tn)


# classification report for precision, recall f1-score and accuracy


C_Report = classification_report(actual,predicted,labels=[1,0])


print('Classification report : \n', C_Report)


# calculating the metrics


sensitivity = round(tp/(tp+fn), 3); specificity =  round(tn/(tn+fp),
3); accuracy =  round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy =  round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3); f1Score = round((2*tp/(2*tp + fp
+ fn)),

3);



# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to
+1.

# A model with a score of +1 is a perfect model and -1 is a poor model


from math import sqrt


mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn) MCC =  round(((tp *
tn) - (fp * fn)) / sqrt(mx), 3)
```

```
print('Accuracy :', round(accuracy*100, 2),'%')  print('Precision
:', round(precision*100, 2),'%') print('Recall :',  round(sensitivity*100,2),
'%') print('F1 Score :', f1Score)   print('Specificity or True Negative Rate :',
round(specificity*100,2), '%'  )   print('Balanced Accuracy :',
round(balanced_accuracy*100, 2),'%') print('MCC  :', MCC)


# Area under ROC curve


from sklearn.metrics import roc_curve, roc_auc_score


print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))


# ROC Curve


from sklearn.metrics import roc_auc_score from sklearn.metrics import  roc_curve
logit_roc_auc = roc_auc_score(y_test, y_pred) fpr, tpr,  thresholds = roc_curve(y_test,
dfDT.predict_proba(X_test)[:,1]) plt.figure()

# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc) plt.plot(fpr, tpr,
label= 'Classification Model' % logit_roc_auc) plt.plot([0, 1],
[0, 1],'r--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate') plt.title('Receiver operating characteristic')
plt.legend(loc="lower right") plt.show()

print('----------------------------------------------------------------------------------------------------')
```

3.3.4 logistic regression python Code

```
from sklearn.linear_model import LogisticRegression


# Create an object for LR model


ModelLR= LogisticRegression()


# Train the model with training data


ModelLR = ModelLR.fit(X_train,y_train)
```

```
# Predict the model with test data set

y_pred = ModelLR.predict(X_test)     #
confusion matrix in sklearn


from sklearn.metrics import confusion_matrix from   sklearn.metrics
import classification_report



# actual values


actual = y_test


# predicted values


predicted = y_pred


# confusion matrix


matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)   print('Confusion
matrix : \n', matrix)



# outcome values order in sklearn


tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1) print('Outcome  values
: \n', tp, fn, fp, tn)



# classification report for precision, recall f1-score and accuracy


C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)


# calculating the metrics
```

```
 sensitivity = round(tp/(tp+fn), 3); specificity =  round(tn/(tn+fp),
3);       accuracy      =            round((tp+tn)/(tp+fp+tn+fn),      3);
balanced_accuracy    =    round((sensitivity+specificity)/2,    3);
precision = round(tp/(tp+fp), 3); f1Score =  round((2*tp/(2*tp +
fp + fn)), 3);

# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to
+1.  # A model with a score of +1 is a perfect model and -1 is a poor model

from math import sqrt     mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn) MCC = round((((tp * tn) -
(fp * fn)) / sqrt(mx), 3)

print('Accuracy :', round(accuracy*100, 2),'%') print('Precision :',   round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%') print('F1

Score :', f1Score) print('Specificity or True Negative Rate :', round(specificity*100,2),
'%'  ) print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%') print('MCC :', MCC)

 # Area under ROC curve      from sklearn.metrics import
roc_curve, roc_auc_score     print('roc_auc_score:',
round(roc_auc_score(y_test, y_pred), 3))

# ROC Curve
 from sklearn.metrics import roc_auc_score from sklearn.metrics import  roc_curve
logit_roc_auc = roc_auc_score(y_test, y_pred) fpr, tpr, thresholds  =
roc_curve(y_test,ModelLR.predict_proba(X_test)[:,1]) plt.figure()

# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc) plt.plot(fpr, tpr, label= 'Classification
Model' % logit_roc_auc) plt.plot([0, 1],

[0, 1],'r--')  plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate') plt.ylabel('True Positive
Rate') plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")  #plt.savefig('Log_ROC')
plt.show()

print('----------------------------------------------------------------------------------------------------')
```

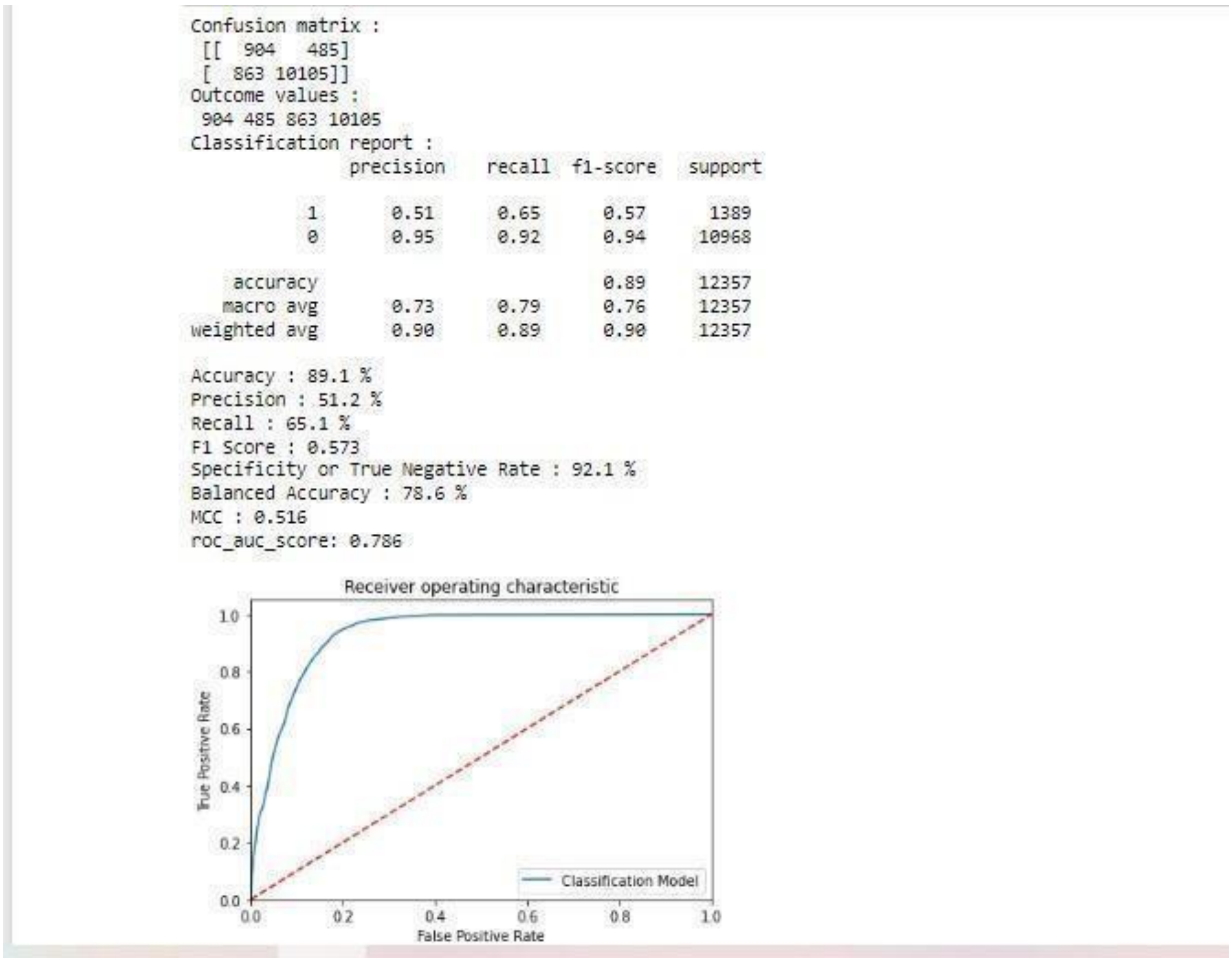
4.0 Conclusions and Future work

The model results in the following order by considering the model accuracy, F1 score and RoC AUC score.

1) **Random Forest** with Simple Random Sampling
2) **Logistic Regression** with Simple Random Sampling
3) **Decision tree** with Simple Random Sampling

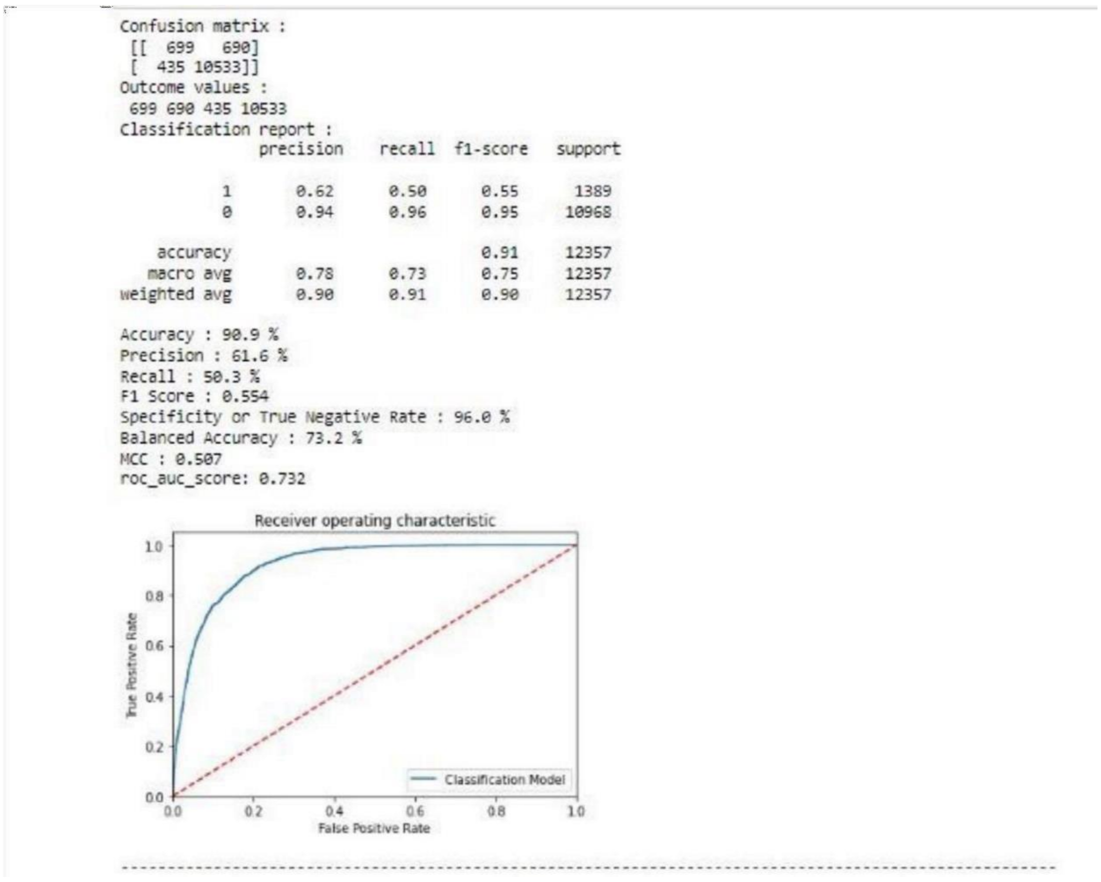
We recommend model - **Logistic Regression** with Simple Random Sampling technique as a best fit for the given bank marketing dataset. We considered **Logistic Regression** because it uses bootstrap aggregation which can reduce bias and variance in the data and can leads to good predictions with bank marketing dataset.

Random forest:



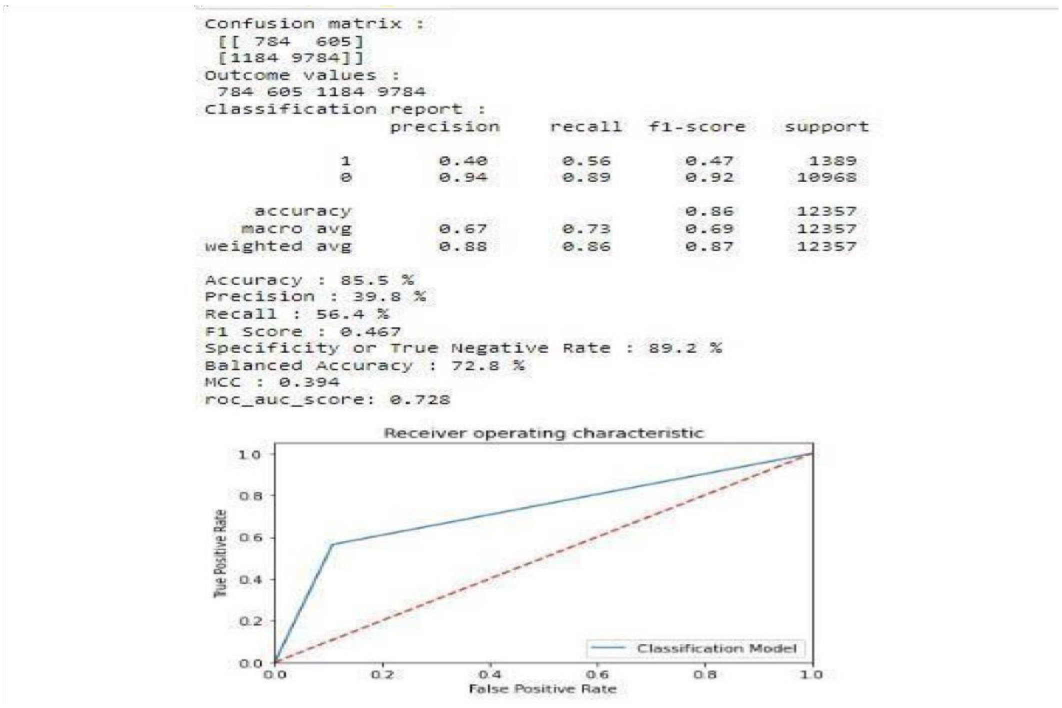
```
Confusion matrix :
 [[  904   485]
 [  863 10105]]
Outcome values :
 904 485 863 10105
Classification report :
              precision    recall  f1-score   support

           1       0.51      0.65      0.57      1389
           0       0.95      0.92      0.94     10968

    accuracy                           0.89     12357
   macro avg       0.73      0.79      0.76     12357
weighted avg       0.90      0.89      0.90     12357

Accuracy : 89.1 %
Precision : 51.2 %
Recall : 65.1 %
F1 Score : 0.573
Specificity or True Negative Rate : 92.1 %
Balanced Accuracy : 78.6 %
MCC : 0.516
roc_auc_score: 0.786
```

Logistic regression:



```
Confusion matrix :
 [[  699   690]
 [  435 10533]]
Outcome values :
 699 690 435 10533
Classification report :
               precision    recall  f1-score   support

           1       0.62      0.50      0.55      1389
           0       0.94      0.96      0.95     10968

    accuracy                           0.91     12357
   macro avg       0.78      0.73      0.75     12357
weighted avg       0.90      0.91      0.90     12357

Accuracy : 90.9 %
Precision : 61.6 %
Recall : 50.3 %
F1 Score : 0.554
Specificity or True Negative Rate : 96.0 %
Balanced Accuracy : 73.2 %
MCC : 0.507
roc_auc_score: 0.732
```

Decision tree:



```
Confusion matrix :
 [[ 784  605]
 [1184 9784]]
Outcome values :
 784 605 1184 9784
Classification report :
               precision    recall  f1-score   support

           1       0.40      0.56      0.47      1389
           0       0.94      0.89      0.92     10968

    accuracy                           0.86     12357
   macro avg       0.67      0.73      0.69     12357
weighted avg       0.88      0.86      0.87     12357

Accuracy : 85.5 %
Precision : 39.8 %
Recall : 56.4 %
F1 Score : 0.467
Specificity or True Negative Rate : 89.2 %
Balanced Accuracy : 72.8 %
MCC : 0.394
roc_auc_score: 0.728
```

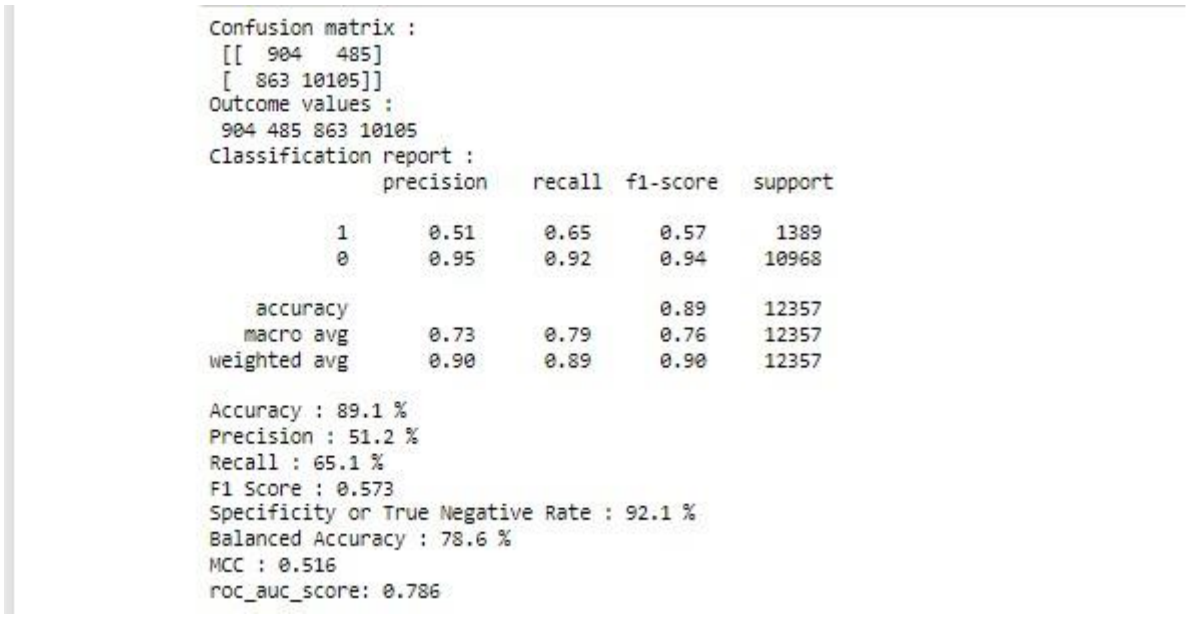
5.0 References

- Data set taken from **UCI machine learning repository**.
- Algorithms random forest ,logistic regression ,Decision tree Classifier are referenced from **JAVA TUTORIAL WEBSITE.**
- Think python:A book wriiten by Allen B. Downey (O'Reilly, 2015) is referenced for python packages and charts plotting.

6.0 Appendices

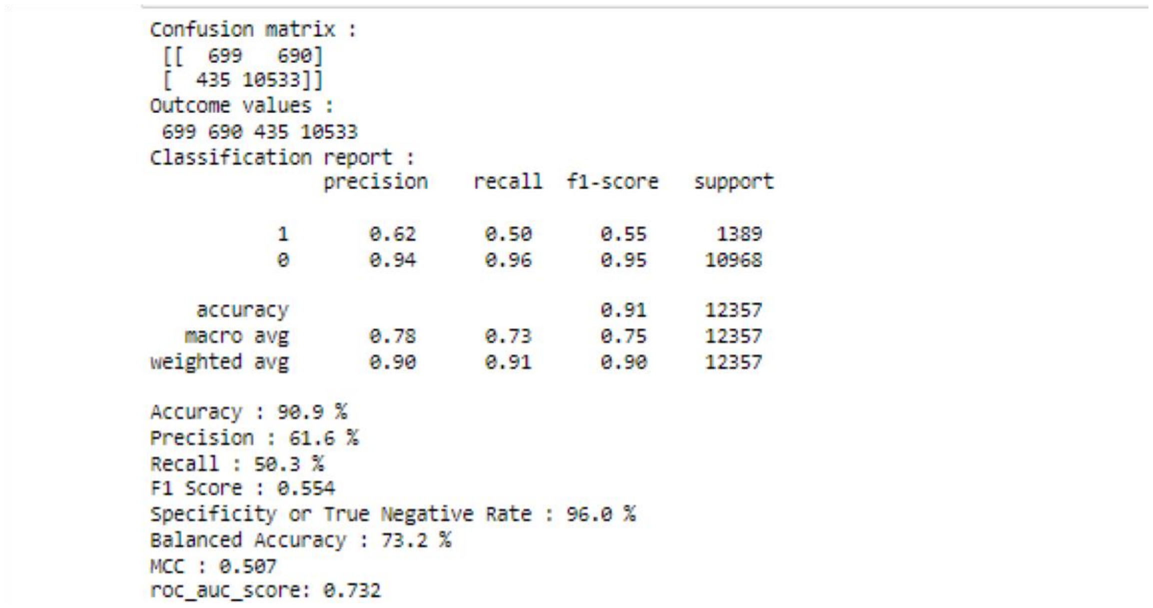
6.1. Python code Results

Random forest:



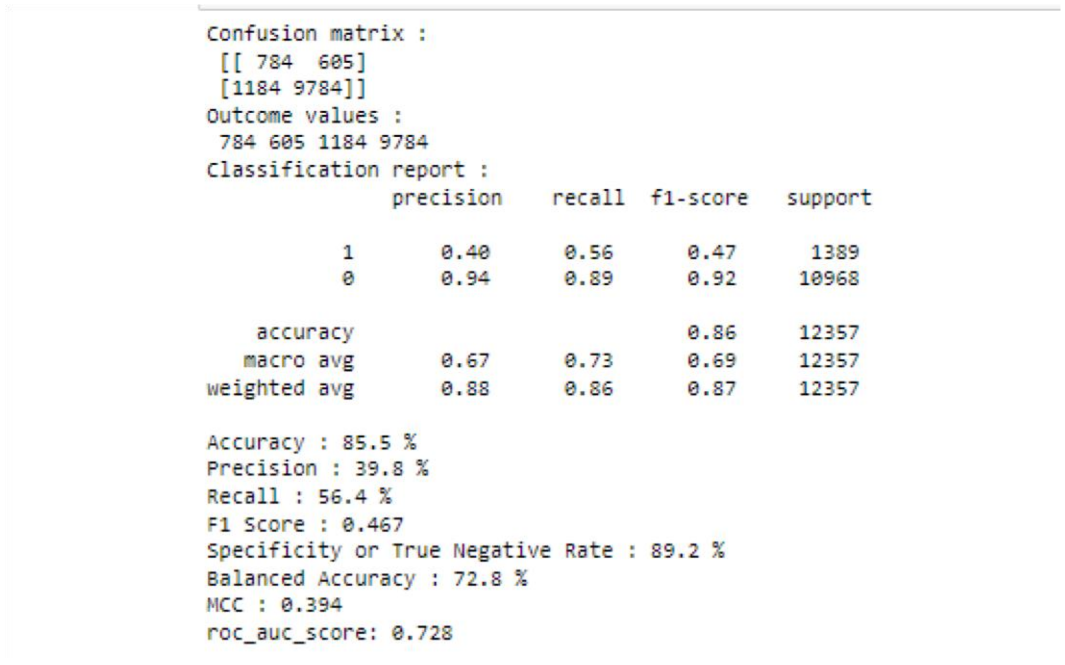
```
Confusion matrix :
 [[  904   485]
 [  863 10105]]
Outcome values :
 904 485 863 10105
Classification report :
              precision    recall  f1-score   support

           1       0.51      0.65      0.57      1389
           0       0.95      0.92      0.94     10968

    accuracy                           0.89     12357
   macro avg       0.73      0.79      0.76     12357
weighted avg       0.90      0.89      0.90     12357

Accuracy : 89.1 %
Precision : 51.2 %
Recall : 65.1 %
F1 Score : 0.573
Specificity or True Negative Rate : 92.1 %
Balanced Accuracy : 78.6 %
MCC : 0.516
roc_auc_score: 0.786
```

Logistic regression:



```
Confusion matrix :
 [[  699   690]
 [  435 10533]]
Outcome values :
 699 690 435 10533
Classification report :
              precision    recall  f1-score   support

           1       0.62      0.50      0.55      1389
           0       0.94      0.96      0.95     10968

    accuracy                           0.91     12357
   macro avg       0.78      0.73      0.75     12357
weighted avg       0.90      0.91      0.90     12357

Accuracy : 90.9 %
Precision : 61.6 %
Recall : 50.3 %
F1 Score : 0.554
Specificity or True Negative Rate : 96.0 %
Balanced Accuracy : 73.2 %
MCC : 0.507
roc_auc_score: 0.732
```

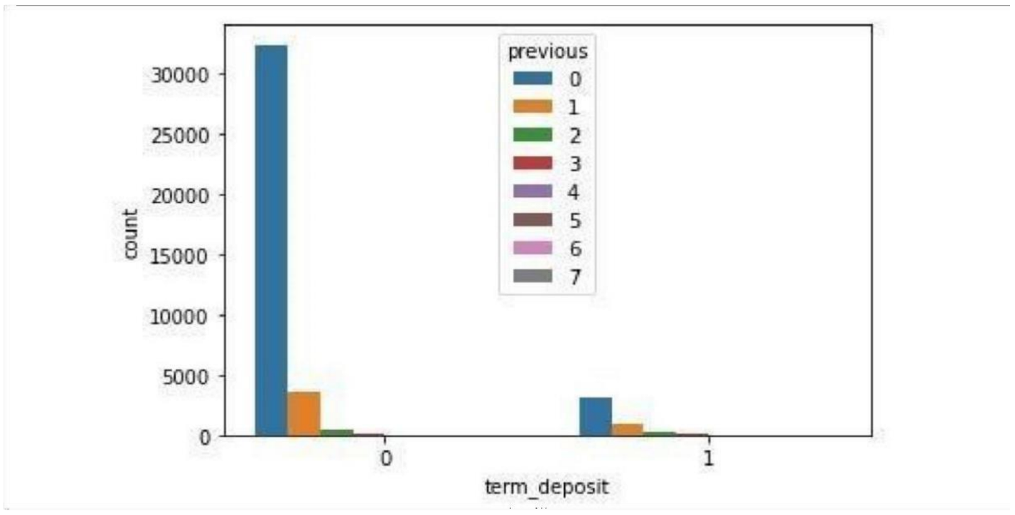
Decision tree:



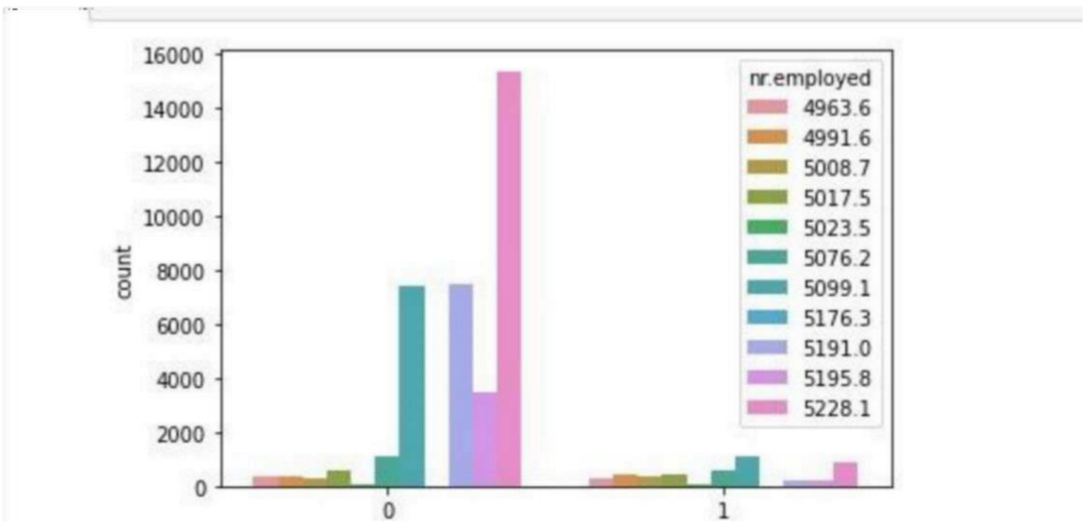
```
Confusion matrix :
 [[ 784  605]
 [1184 9784]]
Outcome values :
 784 605 1184 9784
Classification report :
              precision    recall  f1-score   support

           1       0.40      0.56      0.47      1389
           0       0.94      0.89      0.92     10968

    accuracy                           0.86     12357
   macro avg       0.67      0.73      0.69     12357
weighted avg       0.88      0.86      0.87     12357

Accuracy : 85.5 %
Precision : 39.8 %
Recall : 56.4 %
F1 Score : 0.467
Specificity or True Negative Rate : 89.2 %
Balanced Accuracy : 72.8 %
MCC : 0.394
roc_auc_score: 0.728
```

6.2. List of Charts

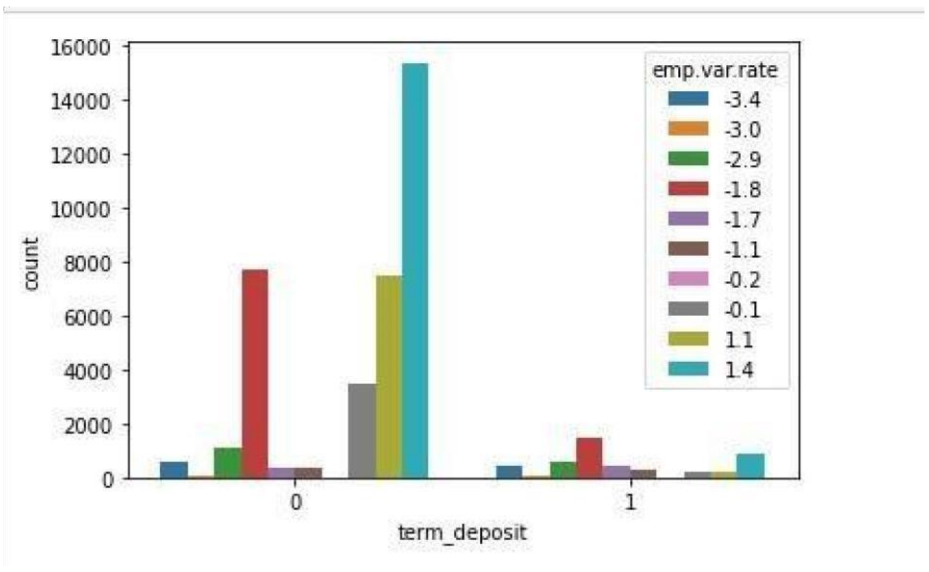
6.2.1 Chart 01: Previous:



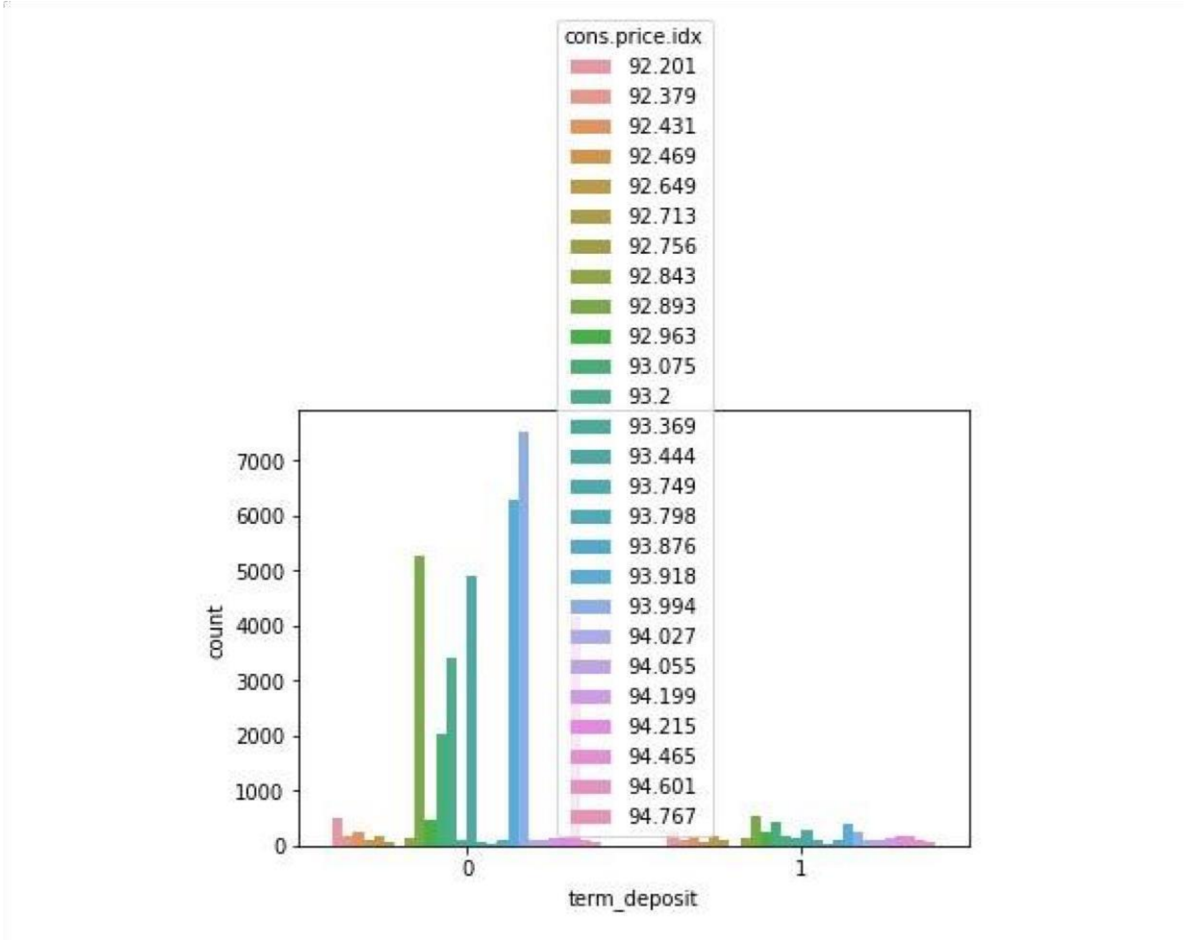
6.2.2 Chart 02: nr.employed



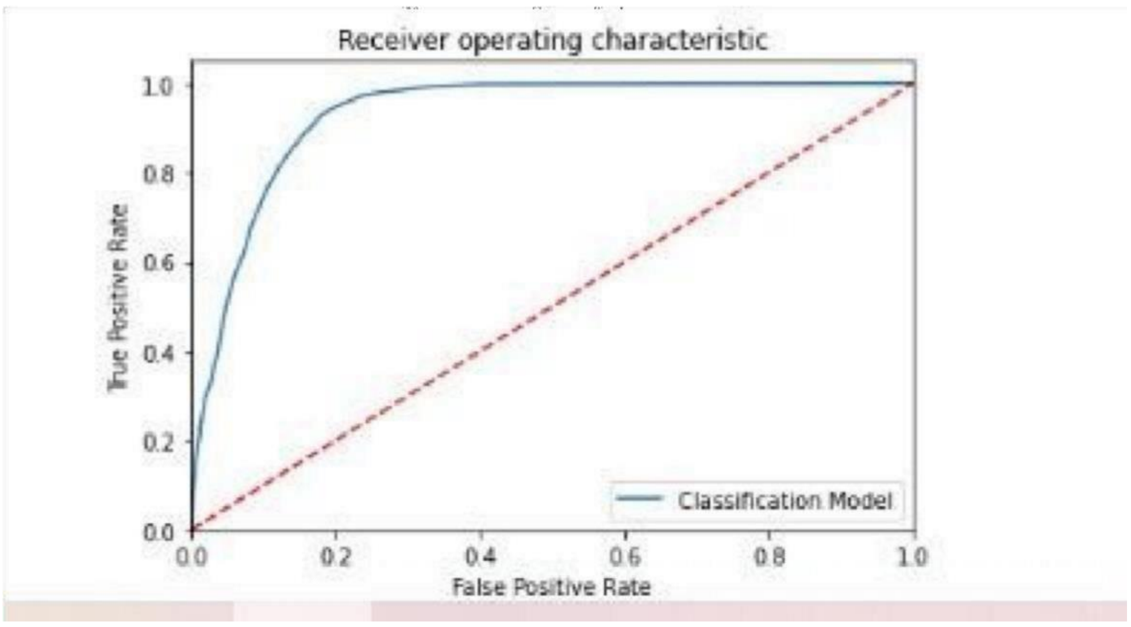
6.2.3 Chart 03: emp.var.rate



6.2.4 Chart 04: cons.price.idx:



6.2.5 Chart 05: Roc curve for Random forest:


Receiver operating characteristic
True Positive Rate
False Positive Rate
Classification Model