EVALUATION OF CLASSIFICATION AND ENSEMBLE ALGORITHMS FOR BANK CUSTOMER MARKETINGRESPONSE PREDICTION

EXECUTIVE SUMMARY

This project attempts to improve the performance of classification algorithms used in the bank customer marketing response prediction of an unnamed bank using the Random Forest ensemble.

A thorough exploratory data analysis (EDA) was conducted on the data in order to ascertain the presence of anomalies such as outliers and extreme values. The EDA revealed that the bank data had 41, 118 instances and 21 features, with 11.7% positive responses. This was in addition to the detection of outliers and extreme values.

Classification algorithms used for modelling the bank dataset include; Logistic Regression, Decision Tree, Naïve Bayes and the Random Forest ensemble. These algorithms were applied to both the balanced and original bank data using the ten-fold cross-validation method. Results from the experimental methods revealed that the performance of the Random Forest ensemble improved when the data was balanced. Results also showed that the features duration, poutcome, contact, month and housing were the most important features that contribute to the success of the bank customer marketing campaign for deposit subscription. The study also revealed that the duration of call to clients, response to past promotions, and the use of cell phone contribute positively to the success of the campaign. While the months of September, November, March and April recorded higher subscription rates. Those in management cadre and technicians were found to have responded more positively to the campaign than those in other job categories.

Table of Contents

1- Introduction to Bank Marketing Campaigns 8

1.1 **Case Study for this Project:** 8

1.1.1 Background 8

2- Data Cleaning and Exploratory Analysis 10

2.4. Graphical Analysis 12

2.5. Data Cleaning and Feature Engineering 15

3- Machine Learning Model Selection 19

4- Model Evaluation and Performance 22

4.1 Evaluating the performance metrics for our dataset: 24

Summary of Model Performances 28

Conclusion 30

Recommendations to the Marketing Team for Future Campaigns: 31

Bibliography 31

1- Introduction to Bank Marketing Campaigns

With continued emphasis on cost reduction while fierce competition within the global financial services market is steadily growing, banks are faced with more challenging marketing goals than ever before. Growing revenue through cross-selling along with focusing on the acquisition of new customers is a fundamental necessity, meaning that the need for additional marketing campaigns is real. These campaigns however, cannot merely be delivered at any cost, Banks need to assess how they can best meet their marketing goals in a cost effective and timely manner. More than ever before, the digital nature of today's banking market is helping retail Banks to reach out to customers in more efficient and appropriate ways.

Personal finances topics have a more sensitive connotation and Banks need to remain aware of the very privileged relationship they hold with their customers, based on confidentiality, something which differs from any other industry. Banking is a necessity which makes these relationships so special due to their nature. Banking relationships are based on confidentially shared personal data and account management that go back many years for some customers and it certainly implies that these should be protected against potential damage from digital marketing tactics used at a high frequency. This is precisely why a marketing campaign run by a Bank cannot be intrusive. Banks need to work hard to find the right balance between selling and protecting customer privacy. In order to successfully navigate this, Banks should consider a number of factors to ensure they approach customers in just the right way.

1.1 Case Study for this Project:

1.1.1 Background:

The increasing number of marketing campaigns over time has reduced their effects on the general public. First, due to competition, positive response rate to mass campaigns are typically very low, according to a recent study, less than 1% of the contacts will subscribe a term deposit. Second, direct marketing has drawbacks, such as causing negative attitude towards banks due to intrusion of

privacy. In order to save costs and time, it is important to filter the contacts but keep a certain success rate.

1.1.2 Objective:

The objective is to build a classifier to predict whether or not a client will subscribe a term deposit. If the classifier has high accuracy, the banks can arrange a better management of available resources by focusing on the potential customers "picked" by the classifier, which will improve their efficiency a lot. Besides, we plan to find out which factors are influential to customers' decision, so that a more efficient and precise campaign strategy can be designed to help to reduce the costs and improve the profits.

A **Term deposit** is a deposit that a bank or a financial institution offers with a fixed rate (often better than just opening deposit account) in which the money will be returned back at a specific maturity time.

1.1.3 Data Source:

Our data were collected from a Portuguese marketing campaign related with bank deposit subscription for 41188 clients and 21 features, and the response is whether the client has subscribed a term deposit.

This data was obtained from the UC Irvine Machine Learning Repository, and relates to the direct marketing campaigns of a Portuguese banking institution attempting to get its clients to subscribe to a term deposit. The marketing campaigns were conducted by making multiple phone calls to the clients. The client responses and predictor variable information are used to assess whether the subject will subscribe to the bank term deposit or not.

2- Data Cleaning and Exploratory Analysis

2.1. Programming in Python:

Python provides a number of packages and libraries for the convenience of the programmer. The whole project is coded using Python 3. Packages/libraries used are numpy for array manipulation, pandas for dataframe operations, and matplotlib and seaborn for visualization. The sklearn libraries were also critical in providing packages for machine learning algorithms, tasks, and by giving the user the control to set important attributes of those algorithms as they wished. The dataset is stored in a dataframe and is intensively queried and manipulated using facilities provided by the Python 3 environment. Other data structures such as arrays, lists, and dictionaries are used as needed.

2.2. Exploratory Data Analysis:

Data cleaning and preparation is a critical first step in any machine learning project. Before we start cleaning data for a machine learning project, it is vital to understand what the data is, and what we want to achieve. Without that understanding, we have no basis from which to make decisions about what data is relevant as we clean and prepare our data.

2.2.1 Attribute Description:

Input variables:

Bank Client Data:

1. Age: (numeric)
2. Job: type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','selfemployed','services','student','technician','unemployed','unknown')
3. Marital: marital status (categorical: 'divorced','married','single','unknown';)
4. Education:(categorical:'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5. Default: has credit in default? (categorical: 'no','yes','unknown')
6. Housing: has housing loan? (categorical: 'no','yes','unknown')
7. Loan: has personal loan? (categorical: 'no','yes','unknown')
8. Contact: contact communication type (categorical: 'cellular', 'telephone')
9. Month: last contact month of year (categorical: 'jan', 'feb'..., 'nov', 'dec')

10. Day_Of_Week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11. Duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

Variables related to Campaign:

12. Campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13. Pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14. Previous: number of contacts performed before this campaign and for this client (numeric)
15. Poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

Social And Economic Attributes:

16. Emp.Var.Rate: employment variation rate - quarterly indicator (numeric)
17. Cons.Price.Idx: consumer price index - monthly indicator (numeric)
18. Cons.Conf.Idx: consumer confidence index - monthly indicator (numeric)
19. Euribor3m: euribor 3 month rate - daily indicator (numeric)
20. Nr.Employed: number of employees - quarterly indicator (numeric)

Output Variable :

21. y - has the client subscribed a term deposit? (binary: 'yes','no')

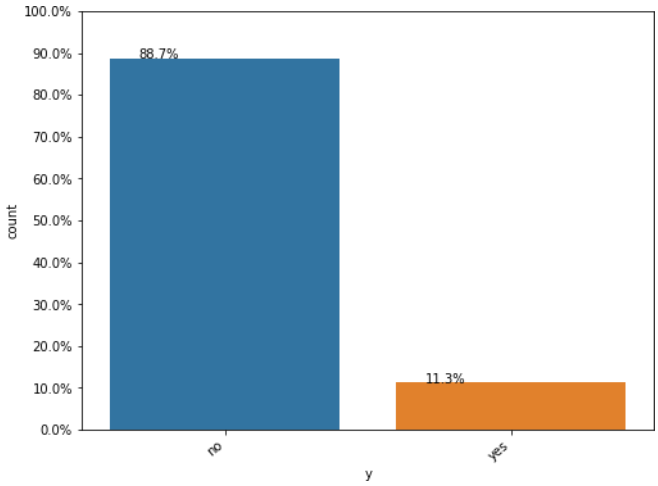
2.3. Snapshot of The Data:

Fig : 1 Bank Marketing Data Snapshot

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent
5	45	services	married	basic.9y	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent
6	59	admin.	married	professional.course	no	no	no	telephone	may	mon	...	1	999	0	nonexistent
7	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent
8	24	technician	single	professional.course	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent
9	25	services	single	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent
10	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent

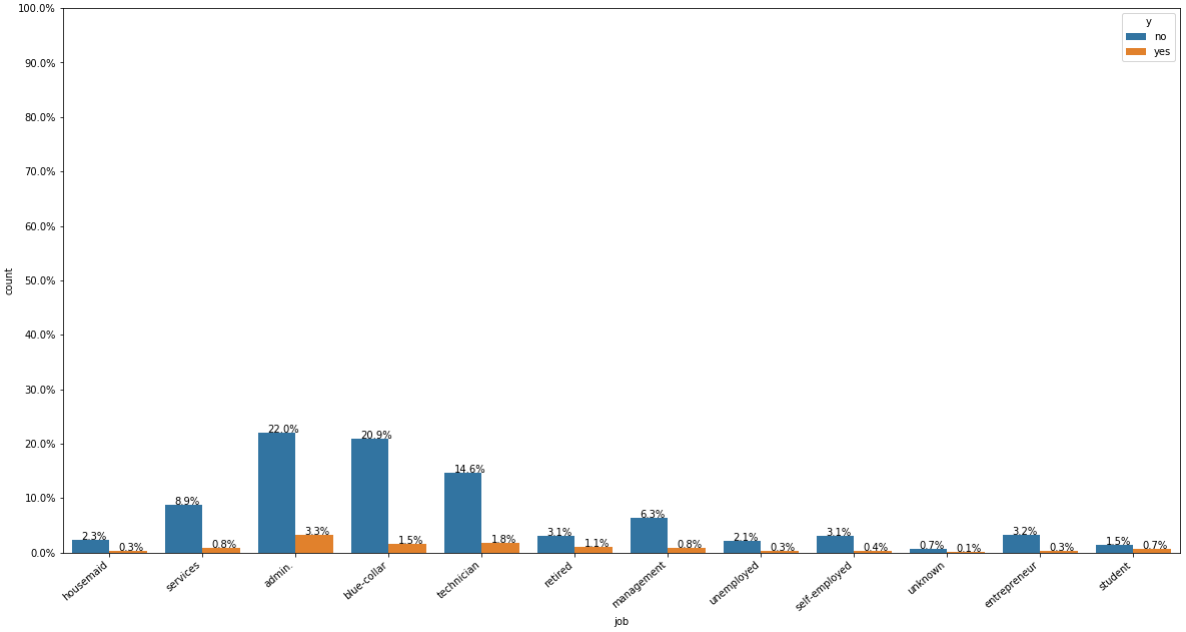
2.4. Graphical Analysis:

Distribution of the Target Variable:

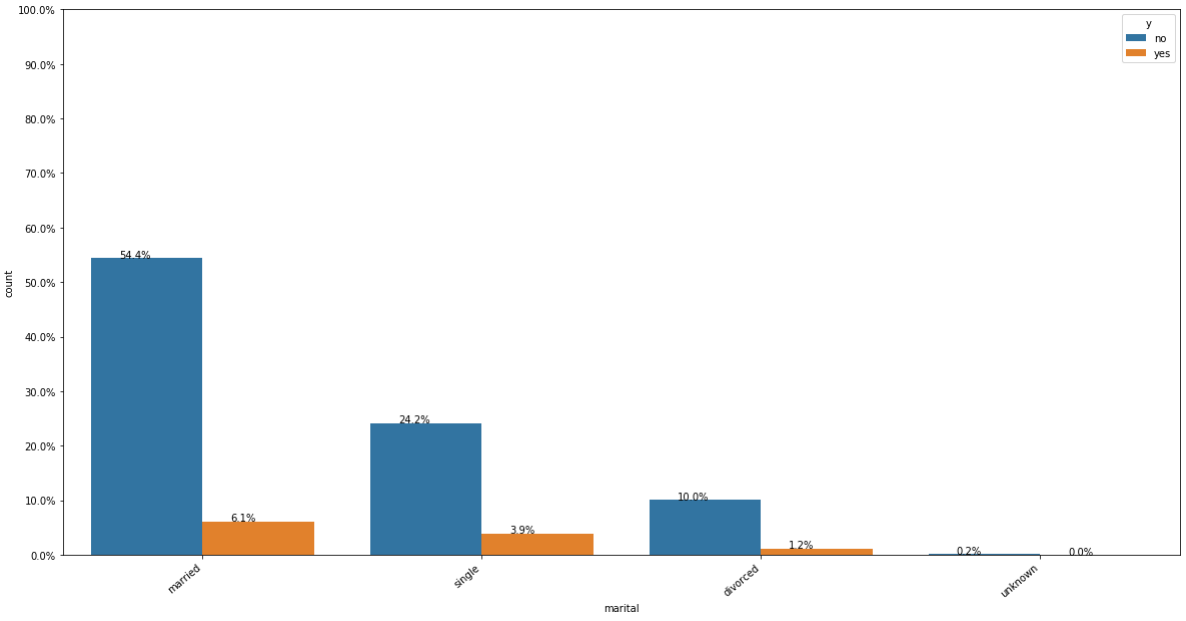


Univariate Analysis:

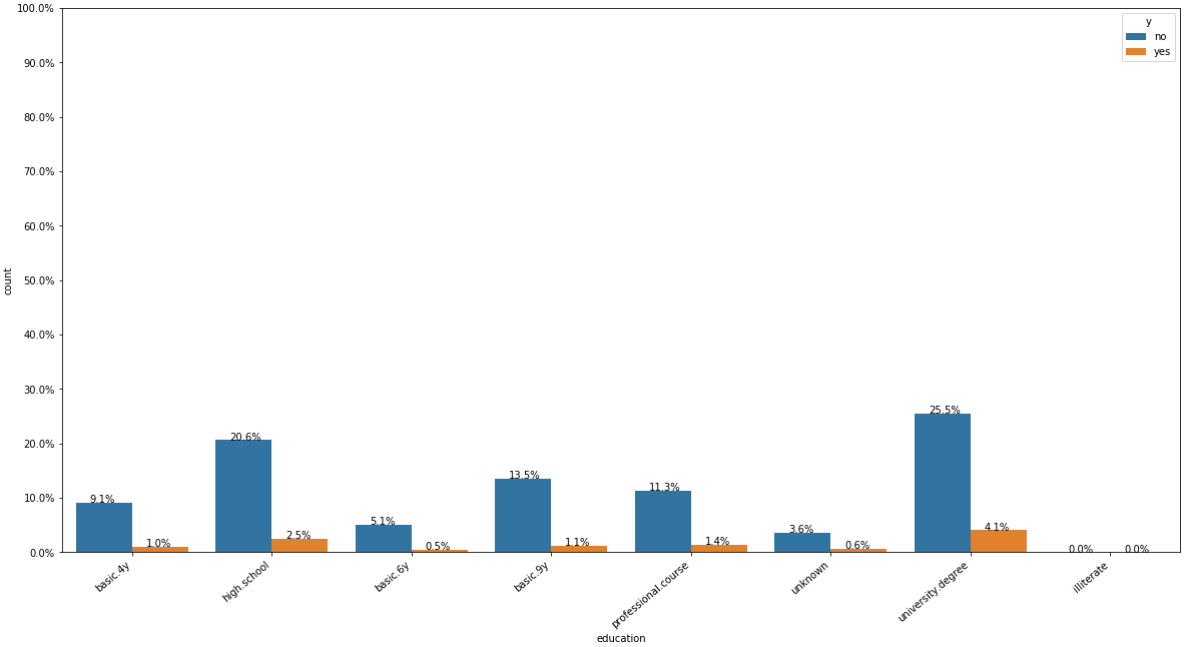
1. Categorical Variable: Job



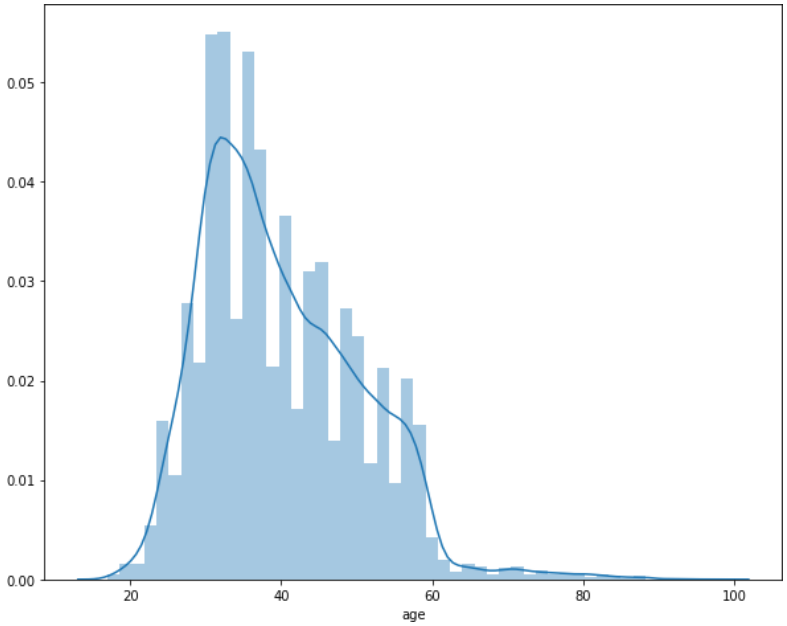
2. Categorical Variable: Marital



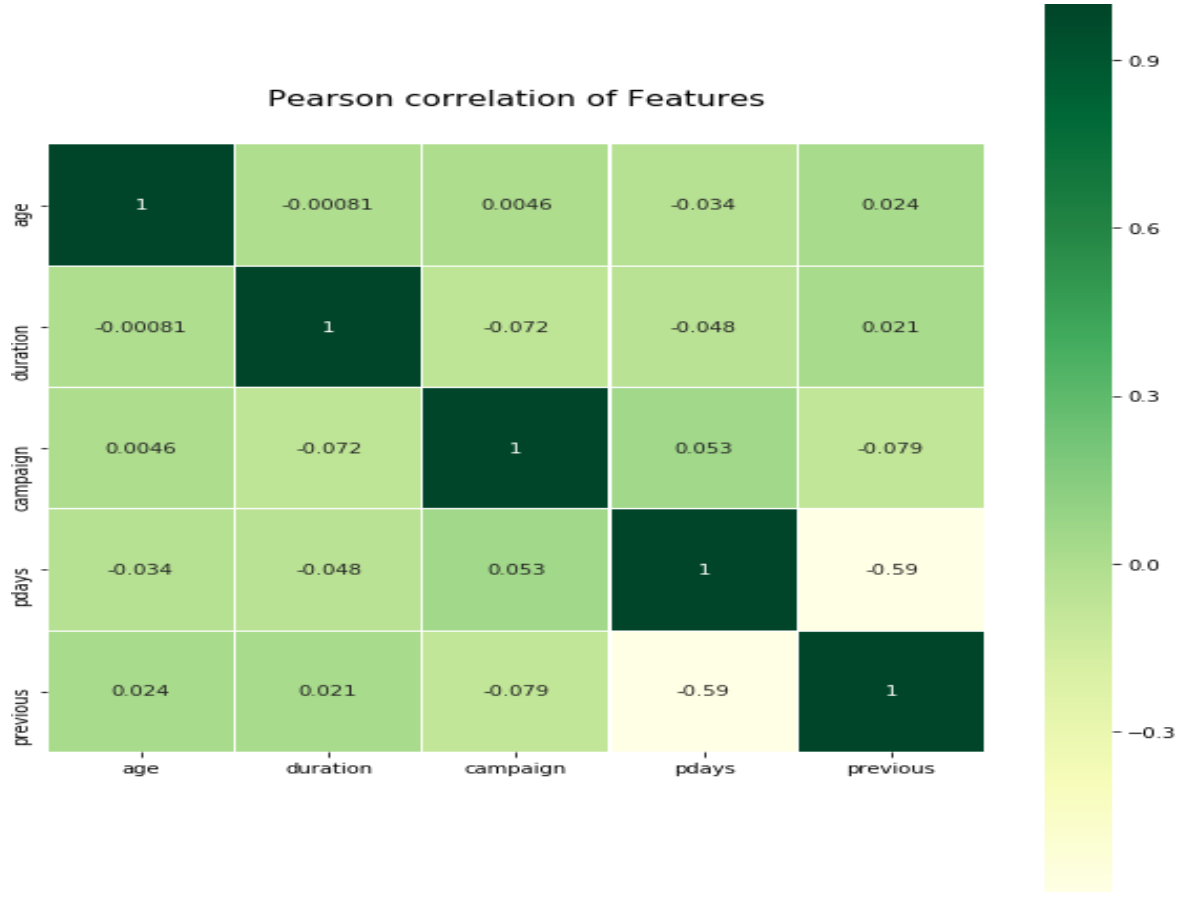
3. Categorical Feature: Education



Distribution of Age : Numeric



Correlation Matrix for numerical variables:



2.5. Data Cleaning and Feature Engineering:

There are two main issues with the dataset:

1. Missing Data:

Since our data were collected from phone call interviews, many clients refused to provide their personal information due to the privacy issue. There are several missing values in some categorical attributes, all coded with the "unknown" label.

```
In [85]: df_cleaned=df.copy()

         #Dropping the unknown job level
         df_cleaned = df_cleaned[df_cleaned.job != 'unknown']
         #Dropping the unknown marital status
         df_cleaned = df_cleaned[df_cleaned.marital != 'unknown']
         #Dropping the unknown and illiterate education level
         df_cleaned = df_cleaned[df_cleaned.education != 'unknown']
         df_cleaned = df_cleaned[df_cleaned.education != 'illiterate']
         #Deleting the 'default' column
         del df_cleaned['default']
         #Deleting the 'duration' column
         del df_cleaned['duration']
         #Dropping the unknown housing loan status
         df_cleaned = df_cleaned[df_cleaned.housing != 'unknown']
         #Dropping the unknown personal loan status
         df_cleaned = df_cleaned[df_cleaned.loan != 'unknown']
```

All the rows with unknown labels were dropped.

2. Combining Sparse Variables:

```
#Combining entrepreneurs and self-employed into self-employed
df_cleaned.job.replace(['entrepreneur', 'self-employed'], 'self-employed', inplace=True)

#Combining administrative and management jobs into admin_management
df_cleaned.job.replace(['admin.', 'management'], 'administration_management', inplace=True)

#Combining blue-collar and tecnician jobs into blue-collar
df_cleaned.job.replace(['blue-collar', 'technician'], 'blue-collar', inplace=True)

#Combining retired and unemployed into no_active_income
df_cleaned.job.replace(['retired', 'unemployed'], 'no_active_income', inplace=True)

#Combining services and housemaid into services
df_cleaned.job.replace(['services', 'housemaid'], 'services', inplace=True)

#Combining single and divorced into single
df_cleaned.marital.replace(['single', 'divorced'], 'single', inplace=True)

#Combining basic school degrees
df_cleaned.education.replace(['basic.9y', 'basic.6y', 'basic.4y'], 'basic_school', inplace=True)
```

After combining the sparse variables, removing the unknowns and duplicates the shape of our final dataset was (38216,14).

3. Chi Square Test for Testing the Independence of Categorical Variables:

Based on the results of the chi-squared tests, we conclude that:

Job level depends on the level of education.

Job level is independent from the absence or presence of a housing loan

Job level depends on the marital status.

Job level is independent from the absence or presence of a personal loan

Job level depends on the type of contact.

Marital status depends on the level of education.

Marital status is independent from the absence or presence of a housing loan.

Marital status is independent from the absence or presence of a personal loan.

Marital status depends on the type of contact.

Level of education is independent from the absence or presence of a housing loan.

Level of education is independent from the absence or presence of a personal loan.

Level of education depends on the type of contact.

Absence or presence of a housing loan depends on the absence or presence of a personal loan.

Absence or presence of a housing loan depends on the type of contact.

Absence or presence of a personal loan is independent from the type of contact.

4. Imbalanced Data:

The presence of imbalanced data may distort the algorithms and its predicting performance. This problem often happens in real world dataset, since people with some certain behaviours account for relatively smaller part. In this case, the responses in the training data are 90% “no” and 10% “yes”, which is surly a significantly imbalanced dataset. How to deal with this problem can be divided into two parts.

First, change the way of measuring algorithm’s performance. As a traditional and common measurement of performance, the test accuracy rate cannot be simply used here because the model will tend to fit the majority class better to improve the overall accuracy. However, we prefer to be more successful in identifying people who will subscribe a term deposit than the overall power of prediction. Therefore, we will use ROC (Receiver Operating Characteristic) curve and AUC (Area Under Curve) as the performance measurement.

Second, change the dataset using resampling method or apply different weights to the observations in objective function. The most commonly used resampling method is oversampling (sample with replacement from the group with less data until the number equals to the larger group), under sampling (sample with replacement from the group with more data until the number equals to the lesser group) and mix sampling (mixture of oversampling and under sampling). The method with reweighting can vary a lot by applying different weights. Therefore, how to deal with the imbalanced data can vary according to the algorithms and the real situations.

To handle imbalanced data in our dataset, we have used the SMOTE TECHNIQUE.

SMOTE TECHNIQUE AND ONE HOT ENCODING:

One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class,

although these examples don’t add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short.

SMOTe is a technique based on nearest neighbours judged by Euclidean Distance between data points in feature space. There is a percentage of Over-Sampling which indicates the number of synthetic samples to be created and this percentage parameter of Over-sampling is always a multiple of 100.

Before applying the SMOTE Technique, we label encoded the data in order to convert all the categorical variables into numerical variables. This conversion is a pre requisite in building efficient Predictive models.

A critical step before one hot encoding is to split the dataset into Train and Test.

```
In [98]: from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test=train_test_split(df_scaled,df['y'],test_size=0.30)
```

We apply the SMOTE technique on the train dataset and test the accuracy of the algorithms with the test dataset.

3- Machine Learning Model Selection

A. Logistic Regression Algorithm

Logistic Regression (LR) algorithm is used for predicting variables with finite set of values. In LR the output is in the form of probability distribution with a value less than one. Logistic Regression is based on maximum probability estimation rather than the least squares estimation used in traditional multiple regression analysis, and hence requires more input data for better results.

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
logpred = logmodel.predict(X_test)


print(confusion_matrix(y_test, logpred))
print(round(accuracy_score(y_test, logpred),2)*100)
LOGCV = (cross_val_score(logmodel, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

B. Decision Tree Algorithm :

Decision Trees: Python provides the package sklearn.tree.DecisionTreeClassifier for the decision tree classifier. Decision trees are a simple yet effective method for classification. Using a tree structure, this algorithm splits the data set based on one feature at every node until all the data in the leaf belongs to the same class. The criterion used for splitting is called information gain, which is based on a purity measure called entropy, a measure of disorder. The set with the highest impurity will have higher entropy whereas the set which has higher purity will have lower entropy. Information gain measures the change in entropy due to the amount of information added. The higher the information gain, the more information that feature provides about the target variable. By default, the decision tree grows deep and complex until every leaf is pure and hence it is prone to overfitting.

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(criterion='gini') #criterion = entopy, gini
dtree.fit(X_train, y_train)
dtreepred = dtree.predict(X_test)

print(confusion_matrix(y_test, dtreepred))
print(round(accuracy_score(y_test, dtreepred),2)*100)
DTREECV = (cross_val_score(dtree, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

C. Support Vector Machine Algorithm:

SVM is a learning algorithm used in regression tasks. However, SVM is preferable in classification tasks. This algorithm is based on the following idea: if a classifier is effective in separating convergent non-linearly separable data points, then it should perform well on dispersed ones. SVM finds the best separating line that maximizes the distance between the hyperplanes of decision boundaries.

```
#SVC Algorithm
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
from sklearn.svm import SVC
svc= SVC(kernel = 'sigmoid')
svc.fit(X_train, y_train)
svcpred = svc.predict(X_test)
print(confusion_matrix(y_test, svcpred))
print(round(accuracy_score(y_test, svcpred),2)*100)
SVCCV = (cross_val_score(svc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

D. Random Forests:

Random forests are an ensemble learning method that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests differ in only one way from tree bagging: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. Tree bagging repeatedly selects a random sample with replacement of the training set and fits trees to these samples. This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias.

```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 200)#criterion = entopy,gini
rfc.fit(X_train, y_train)
rfcpred = rfc.predict(X_test)

print(confusion_matrix(y_test, rfcpred ))
print(round(accuracy_score(y_test, rfcpred),2)*100)
RFCCV = (cross_val_score(rfc, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

E. Naïve Bayes Classifier:

Naive Bayes is a direct and powerful classifier that uses the Bayes theorem. It predicts the probability that a given record or data point belongs to a particular class. The class with the highest probability is considered to be the most likely class. This algorithm assumes that all features are independent and unrelated. The Naive Bayes model is simple and easy to build and particularly useful for large data sets. This model is known to outperform even highly sophisticated classification methods.

```
from sklearn.naive_bayes import GaussianNB
gaussiannb= GaussianNB()
gaussiannb.fit(X_train, y_train)
gaussiannbpred = gaussiannb.predict(X_test)
probs = gaussiannb.predict(X_test)

print(confusion_matrix(y_test, gaussiannbpred ))
print(round(accuracy_score(y_test, gaussiannbpred),2)*100)
GAUSIAN = (cross_val_score(gaussiannb, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

F. Gradient Boost Algorithm:

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

```
from sklearn.ensemble import GradientBoostingClassifier
gbk = GradientBoostingClassifier()
gbk.fit(X_train, y_train)
gbkpred = gbk.predict(X_test)
print(confusion_matrix(y_test, gbkpred ))
print(round(accuracy_score(y_test, gbkpred),2)*100)
GBKCV = (cross_val_score(gbk, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

G. XGBOOST Algorithm:

In XGBoost, we fit a model on the gradient of loss generated from the previous step. In XGBoost, we just modified our gradient boosting algorithm so that it works with any differentiable loss function.

```
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
xgbprd = xgb.predict(X_test)

print(confusion_matrix(y_test, xgbprd ))
print(round(accuracy_score(y_test, xgbprd),2)*100)
XGB = (cross_val_score(estimator = xgb, X = X_train, y = y_train, cv = 10).mean())
```

4- Model Evaluation and Performance

Our dataset is a highly imbalanced one and so we have looked at various performance metrics to analyse the performance of our models. Since accuracy doesn't work well with imbalanced datasets we have tried comparing other metrics such as Precision, Recall and F1-Scores as well. Our final evaluation is performed using the AUC metric.

We will first briefly look into the performance characteristics:

1. Confusion Matrix:

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

2. Accuracy:

It is defined as the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives.

3. Precision:

Precision is the ratio of True positives and Total positives predicted.

$0<P<1$

Precision metric focusses on Type-I error (FP). A Type-I error occurs when we reject a true null Hypothesis(H^0). Precision score towards 1 will signify that the model didn't miss any True Positives. What it cannot measure is the existence of Type-II error which is False.

Low precision score (<0.5) means the classifier has a high number of False positives which can be an outcome of imbalanced class or untuned model hyper parameters. In an imbalanced class problem, we have to prepare the data beforehand with Over/Under-Sampling or Focal Loss in order to curb FP/FN.

4. Recall:

Recall is essentially the ratio of True Positives to all the Positives in Ground Truth.

$0<R<1$

Recall metric focussed on Type-II errors(FN). A Type-II error occurs when we accept a false null Hypothesis(H^0). Recall score towards 1 will signify that our model didn't miss any True Positives and is able to classify well. What it cannot measure is the existence of Type-I error which is False. Low recall score (<0.5) means the classifier has a high number of False negatives which can be an outcome of imbalanced class or untuned model hyper parameters. In an imbalanced class problem, we have to prepare the data beforehand with Over/Under-Sampling or Focal Loss in order to curb FP/FN.

5. F1- Score:

F1-score metric uses a combination of precision and recall both. In fact, F1-score is the harmonic mean of the two.

Now, a high F1-score symbolizes a high precision as well as high recall. It presents a good balance between precision and recall and gives good results on imbalanced classification problems.

6. AUROC (Area under Receiver operating characteristics):

A Receiver Operator Characteristic (ROC) curve is a graphical plot used to show the diagnostic ability of binary classifiers. It was first used in signal detection theory but is now used in many other areas such as medicine, radiology, natural hazards and machine learning. In this post I'll show you how a ROC curve is created and how to interpret the ROC curve.

A ROC curve is constructed by plotting the true positive rate (TPR) against the false positive rate (FPR). The true positive rate is the proportion of observations that were correctly predicted to be positive out of all positive observations (TP/ (TP + FN)). Similarly, the false positive rate is the proportion of observations that are incorrectly predicted to be positive out of all negative observations (FP/ (TN + FP)). For example, in medical testing, the true positive rate is the rate in which people are correctly identified to test positive for the disease in question.

4.1 Evaluating the performance metrics for our dataset:

Classification Report and Confusion Matrix for Logistic Regression:

```
[[9990  185]
 [1034  256]]
89.0
```

```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,logpred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     10175
           1       0.58      0.20      0.30      1290

    accuracy                           0.89     11465
   macro avg       0.74      0.59      0.62     11465
weighted avg       0.87      0.89      0.87     11465
```

The accuracy is 89% which is good. The precision is quite high which indicates that a good amount of TRUE POSITIVES are being predicted accurately. A low recall score indicates that the algorithm gives a larger amount of False Negatives and these should be reduced to minimise the Type II error.

Classification Report and Confusion Matrix for SVM Algorithm:

```
[[10017   158]
 [ 1045   245]]
90.0
```

```
print(classification_report(y_test,svcpred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     10175
           1       0.61      0.19      0.29      1290

    accuracy                           0.90     11465
   macro avg       0.76      0.59      0.62     11465
weighted avg       0.87      0.90      0.87     11465
```

The accuracy is high and the False Positives are minimised to a great extent.

Classification Report and Confusion Matrix for Decision Tree Algorithm:

```
[[9224  951]
 [ 944  346]]
83.0
```

```
print(classification_report(y_test,dtreepred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.91      0.91     10175
           1       0.27      0.27      0.27      1290

    accuracy                           0.83     11465
   macro avg       0.59      0.59      0.59     11465
weighted avg       0.84      0.83      0.83     11465
```

The Decision Tree algorithm gave an accuracy of 83%. Higher precision and recall scores indicate that the model is working well and has a lesser number of False Positives and False Negatives as desired.

Classification Report and Confusion Matrix for Random Forest Classifier:

```
[[9673  502]
 [ 910  380]]
88.0
```

```
print(classification_report(y_test,rfcpred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93     10175
           1       0.43      0.29      0.35      1290

    accuracy                           0.88     11465
   macro avg       0.67      0.62      0.64     11465
weighted avg       0.86      0.88      0.87     11465
```

The random forest classifier gave an accuracy of 88% which is great. Higher precision and recall scores indicate that the model is working well and has a lesser number of False Positives and False Negatives as desired.

Classification Report and Confusion Matrix for Naïve Bayes Classifier:

```
[[9152 1023]
 [ 767  523]]
84.0
```

```
print(classification_report(y_test,probs))
```

```
              precision    recall  f1-score   support

           0       0.92      0.90      0.91     10175
           1       0.34      0.41      0.37      1290

    accuracy                           0.84     11465
   macro avg       0.63      0.65      0.64     11465
weighted avg       0.86      0.84      0.85     11465
```

The accuracy received from Naïve Bayes was relatively good. Also the higher precision and recall scores indicated that the model predicted a lesser number of False Positives and False Negatives.

Classification Report and Confusion Matrix for Gradient Boosting Classifier:

```
[[9565  610]
 [ 745  545]]
88.0
```

```
print(classification_report(y_test,gbkpred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.94      0.93     10175
           1       0.47      0.42      0.45      1290

    accuracy                           0.88     11465
   macro avg       0.70      0.68      0.69     11465
weighted avg       0.88      0.88      0.88     11465
```

The scores for accuracy, precision and recall are on the higher side indicating that the model has worked well.

Classification Report and Confusion Matrix for Extreme Gradient Boosting Classifier:

```
[[9902  273]
 [ 971  319]]
89.0
```

```
print(classification_report(y_test,xgbprd))
```

```
              precision    recall  f1-score   support

           0       0.91      0.97      0.94     10175
           1       0.54      0.25      0.34      1290

    accuracy                           0.89     11465
   macro avg       0.72      0.61      0.64     11465
weighted avg       0.87      0.89      0.87     11465
```

The accuracy, precision and recall are good for this model.

Summary of Model Performances:

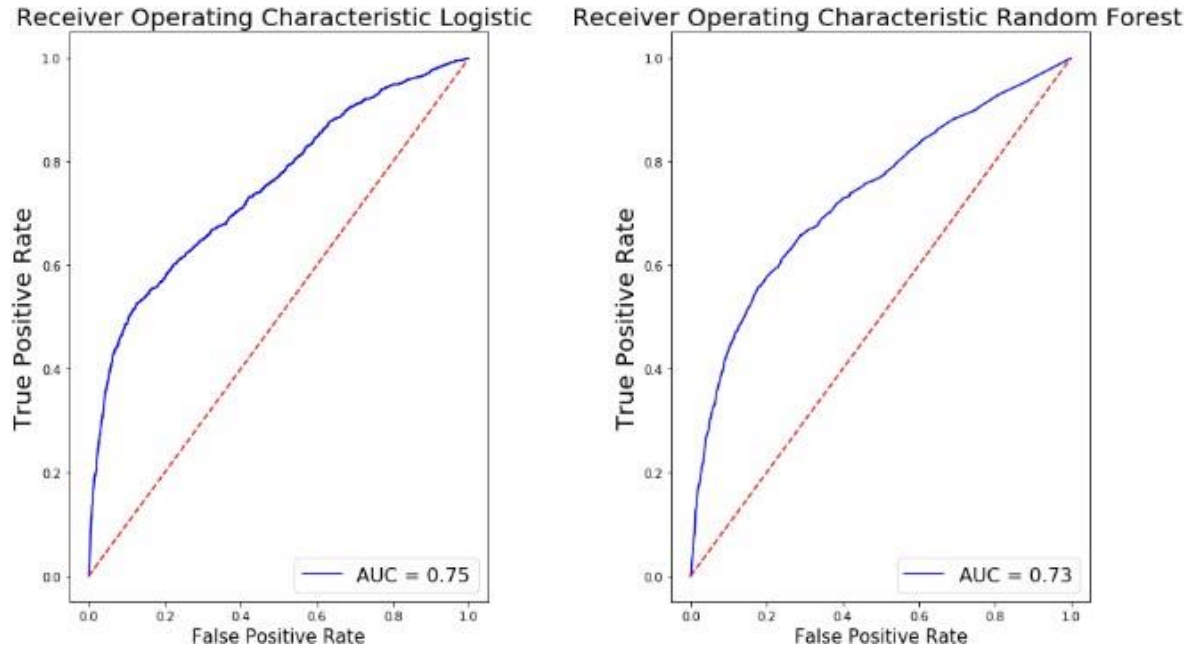
	Models	Score
0	Random Forest Classifier	0.946084
3	Logistic Model	0.942657
5	XGBoost	0.934566
1	Decision Tree Classifier	0.921430
6	Gradient Boosting	0.908641
4	Gausian NB	0.698720
2	Support Vector Machine	0.598364

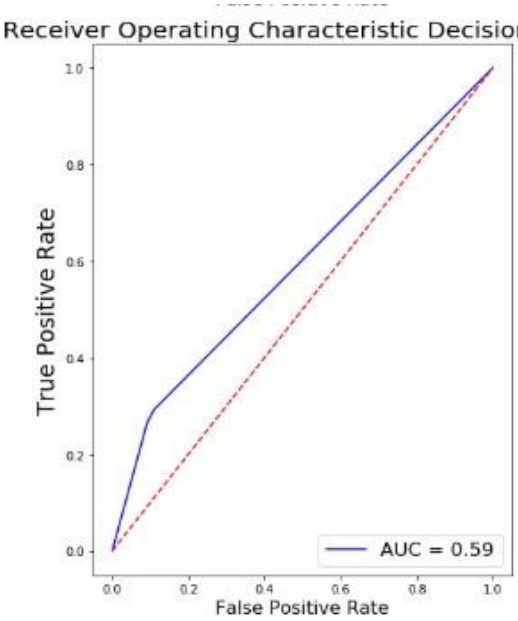
Random forest classifier performed the best with our dataset with a Cross Value Score of 94 %. In terms of reducing the False Positives Extreme Gradient Boosting Algorithm performed the best.

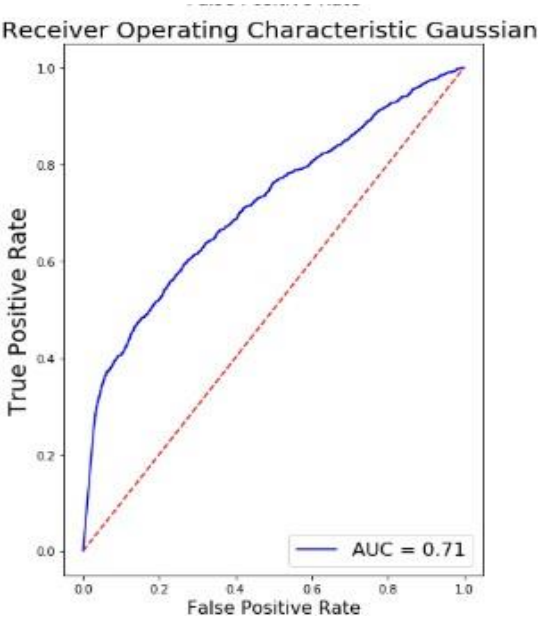
Therefore, in cases where the target distribution is highly imbalanced, we can balance the data using various re sampling techniques. The top 3 algorithms for this kind of classification problem as per our analysis are:

1. Random Forest Classifier
2. Logistic Model
3. XG Boost

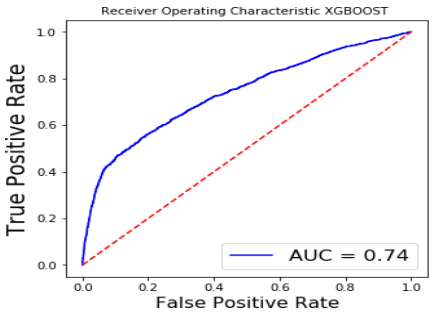
ROC Curves to support the analysis are as follows:

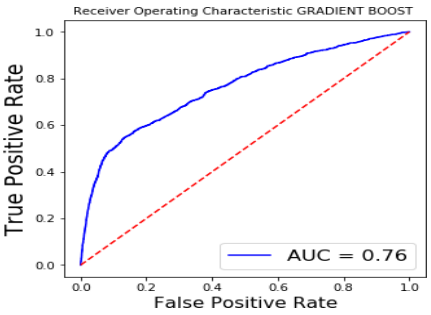




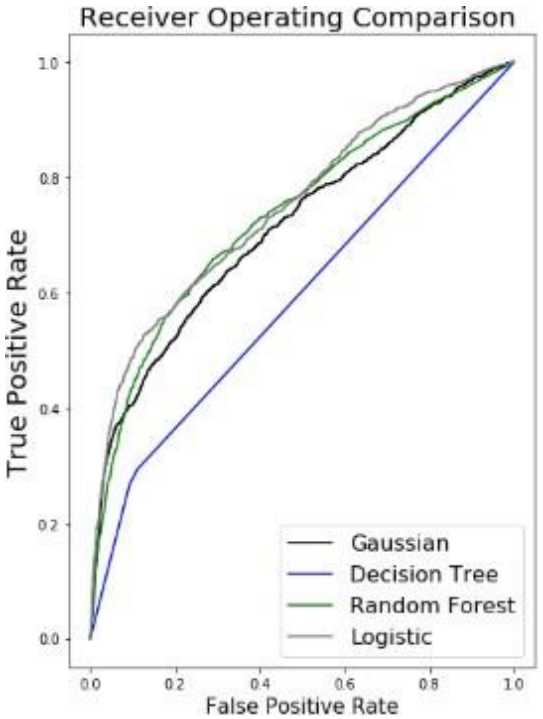


The more the AUC score is close to 1, the more optimal the model is.





AUC Scores Comparison:



An ROC curve demonstrates several things:

1. It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
2. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
3. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.
4. The slope of the tangent line at a cutpoint gives the likelihood ratio (LR) for that value of the test. curve.
5. The area under the curve is a measure of text accuracy.

Conclusion:

The AUC Score for Gradient Boost Algorithm is highest followed by XGBOOST and Logistic Regression Algorithm. Although in terms of accuracy Random Forest algorithm performed best, accuracy cannot be the sole performance metric for imbalanced datasets.

AUC Score is a better performance metric for imbalanced datasets.

A feature importance test on the Random Forest Model gave the following insights :

1. Age : The product was mostly bought by people above the age of 60 although the median age of the dataset was around 31.

2. Contact : The month of May saw the most number of subscriptions as it is the month where most people receive bonuses/increments.

3. Previous Campaign Features : The prediction was highly dependent on features that represented previous campaign details revealing that people who had taken products earlier or were contacted frequently were more inclined to subscribe.

4. Job : Blue collar, Services and Retired people took more subscriptions.

5. Education : The education level of people affected the most in terms of subscriptions as people with a high school and university degree or above were more inclined to subscribe.

Recommendations to the Marketing Team for Future Campaigns:

1. Future Campaigns should target older people and students as they look forward to a less risky form of investment.
2. People who have participated in previous campaigns should be given first priority as they will be more inclined to subscribe to the product.
3. Since only people who had a high school degree or above were able to subscribe it indicates that the product description or review might not have been clear or straightforward for people who are less educated to understand. Future Campaigns should work on improvising the product description so that it can reach a wider audience.

Bibliography

1. Evaluation of Classification and Ensemble Algorithms for Bank Customer Marketing Response Prediction, O. Amampa Journal of International Technology and Information Management
2. www.Kaggle.com
3. Predicting Bank Marketing Campaign Success using Machine Learning, Chaitra Hegde
4. www.TowardsDataScience.com
5.www.Medium.com