DIRECT BANK MARKETING

ISDS 574: Data Mining for Business Applications

Team Members:

ARINDAM ROY

BENU BANSAL

JAPPREET SINGH GILL

RAVI KUMAR

SEPEHR EFTEKHARIAN

Executive summary

A Financial Institution (Bank) in Portugal have multiple Marketing team to sell their financial products known as Term Deposit. For this purpose, the marketing team do campaigning using direct call methodology. The purpose of this analysis to increase the number of customers through the identification of the factors have the most impact on a customer's decision.

The Data have been used in this analysis consist of two full years data from 2008 to 2010, which includes 41,188 observations which are split into 21 Variables-10 numerical,10-categorical and one Binary response variable. To distinguish the factors that are significant and help to predict more accurate, it was decided to use four models (Classification, KNN, CART, Logistic Regression) among all methods.

It was also decided to use the confusion matrix in order to see the prediction accuracy. Furthermore, the confusion matrix which is used for prediction accuracy, has been implemented. Besides, Sensitivity, Specificity, and misclassification error were considered as an index for identifying the best model.

The criteria that the analysts decided to use in order to choose the best model was the model which has the highest sensitivity Low specificity, high accuracy and a high predictive confusion matrix. The best model was identified as the model with high sensitivity. With that said, XGBoost model was selected as the best model which has resulted in 0.49972 as sensitivity, 0.96609 as specificity, 0.9139 as accuracy.

Contents

Introduction: 4
Question of Interest: 4
Variables Explanation: 4
Data Pre-Processing and Exploration 6
KNN 13
Classification Tree 17
Logistic Regression 22
Gradient Boosting 24
Results 27
Conclusion 27
Future Scope and Recommendations. 27
References 29
APPENDIX: 30

Introduction:

This report explains the model that was set and developed using various methods and techniques to increase the sales of a Marketing campaign of a Financial Institution in Portugal through direct Marketing approach which means that Marketers will call the customers directly to encourage them to choose the product which is a term deposit. The data gathered for this analysis is for two years period from 2008 to 2010 which has been found on the UCI Machine Learning Repository. From data type perspective, the set consisted of 21 variables which 10 were numerical, 10 categorical and one Binary variable summing to a total of 41188 observations. Among these variables, some are directly related to the customer whereas some variables are related to the market. The programing tool that the team used for analysis and performing the result was R Programming.

Question of Interest:

The purpose of this analysis is to identify the variables or factors that play a crucial role in the customer's acceptance or denial for having the product. As mentioned above, there are certain factors related to the customer which requires further investigation on the relation of those factors with the market factors and how the combination of these two-factor types affect the decision. Once these patterns identified, the bank can use those for the future strategy of marketing campaigns to absorb the customers.

Variables Explanation:

There are 10 categorical variables, 10 numerical variables, and one binary variable, which is the output variable. The binary variable indicates the outcome of a marketing campaign. In other words, yes if a client purchased a bank term deposit or no if a client did not buy a bank term deposit. All the variables are outlined below. Also, to have better visibility on the Data set, a new column has been added to show the relation of the variable to either Market or Individual.

Variable Name	Description	Type of Variable
Age	Age	Numeric
Job	Type of Job	Categorical
Marital	Marital status	Categorical
Education	Education	Categorical
Default	does client have credit	Categorical
Housing	does client have housing loan	Categorical
Loan	does client have personal loan	Categorical
Contact	contact communication type	Categorical
Month	last contact month of year	Categorical
Day of week	last contact day of week	Categorical
Duration	Last contact duration in seconds	Numerical
Campaign	number of times a client was contacted during the campaign	Numerical
Pdays	number of days that passed after client was contacted from previous campaign	Numerical
Previous	number of times client was contacted before campaign	Numerical
Emp.var.rate	Employment variation rate-quarterly indicator	Numerical
Cons.price.idx	Consumer price index-monthly indicator	Numerical
Cons.conf.idx	Consumer confidence index-monthly indicator	Numerical
Euribor3m	Euribor 3 month rate – daily indicator	Numerical
Nr.employed	Number of employees – quarterly indicator	Numerical
Poutcome	outcome of previous marketing campaign	Categorical
Y	did client purchase bank term deposit	Binary

Also, some of the Market variables have been defined below for further clarification.

Emp.Var.Rate: Employment variation rate is a cyclical variation which explains how many people are being hired or fired due to the changes in the condition of the economy.

Cons.Price.idx: Consumer Price Index: is an indicator which explains the trend or changes on the price of a weighted average basket of consumer goods and services which are purchased by households on a monthly or annually basis.

Cons.Conf.idx: Consumer Confidence Index is an index which explains the confidence on the state of the U.S economy that consumers are expressing through their activities of savings and spending.

Euribor3M: Euribor 3 month is a rate which is published by European Money Market based on the rates of the interest that European banks offer to lend unsecured funds to other banks.

Nr. Employed: Number of Employees is an indicator which shows how many Employees were hired quarterly.

Data Pre-Processing and Exploration

There are 41,188 observation and 21 variables in the Data Set. There are no missing continuous values due to which no imputation was necessary. There are ten continuous variables and ten categorical variables. The target variable, Y, is a binary response which indicates whether the client will subscribe or not. Data visualization techniques are used to describe the behavior of continuous and categorical variables against the target variable (Y)

Continuous Variables

Continuous variables are also known as numeric variables. Although, they have an infinite number of values between any two values. For instance, age in our data lies between 20-60 where it has a lot of values between these two numbers.

The boxplots and frequency graphs were created for the continuous variables such as age, campaign, consumer confidence index, monthly consumer price index, duration, employee variation rate, Euribor 3-month rate, number of employees and number of days that passed after a client was contacted (pdays). The following boxplots and frequency graphs are showing us the behavior of continuous variables against a binary variable in our data set.

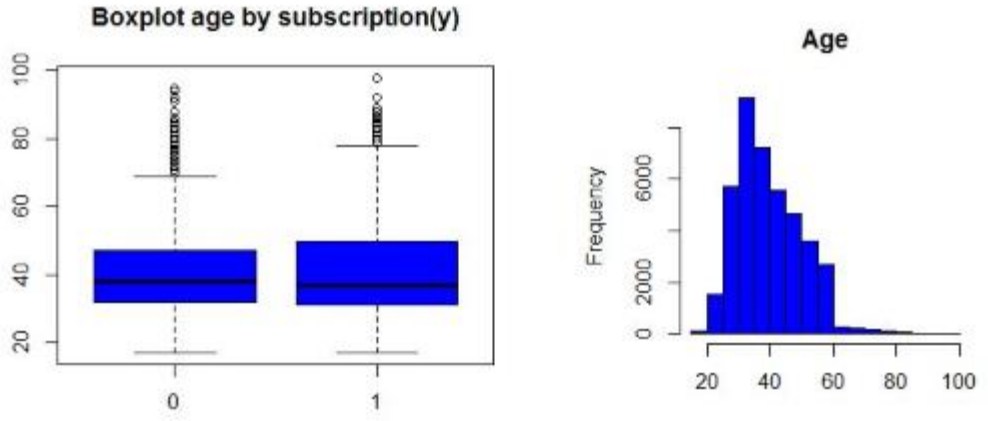

Figure 1: Boxplot and Frequency graph for a continuous variable, age.

Figure 1 describes that age does not have an impact on the target variable (y). The histograms are centered around the same region and have very similar variance regardless of the outcome variable. It means any age group can be a possible subscriber.

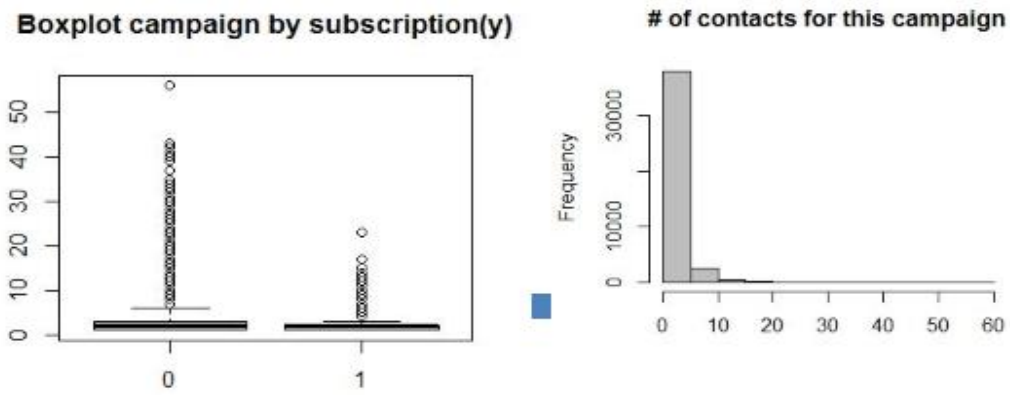

Figure 2: Boxplot and Frequency graph for a continuous variable, # of contacts.

Figure 2 depicts that 97% of the data for the campaign is in the lowest levels for both 0 and 1 responses. It shows that if the customers are contacted more than five times, they likely did not subscribe to the bank deposit term during the campaign.

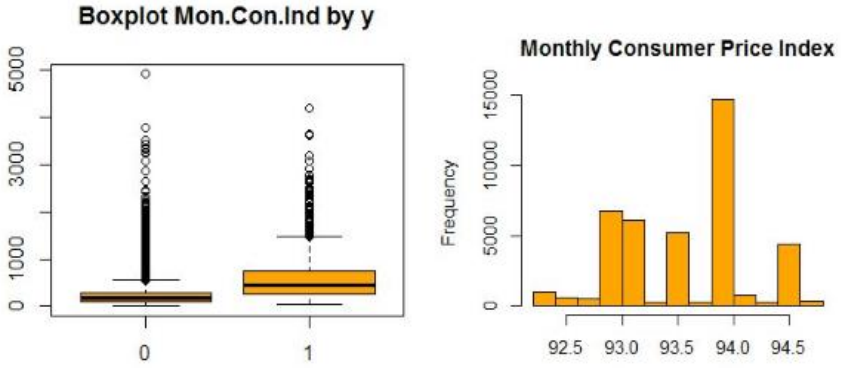

Figure 3: Boxplot and Frequency graph for a continuous variable, Monthly Consumer Price Index

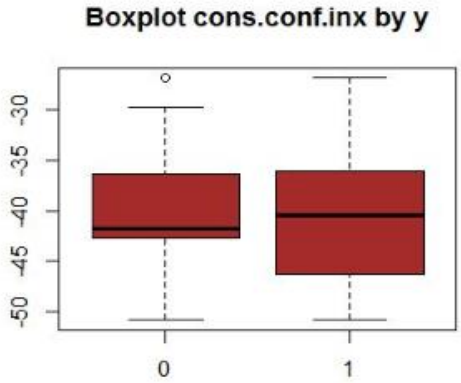

Figure4: Boxplot for continuous variable, consumer confidence Index.

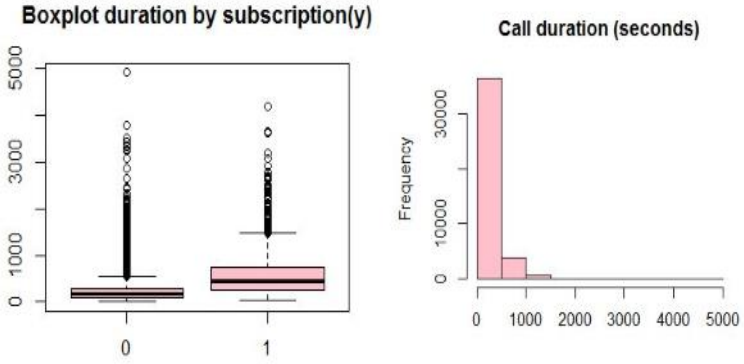

Figure 5: Boxplot and Frequency graph for a continuous variable, Call duration.

Figure 3 and 4 describes that data is multi-model and spread out evenly. Figure 5 briefly depicts the call duration during the campaign. It shows that with increasing call duration, customers less likely to subscribe to the bank deposits. It shows how to call duration is an active variable against the target variable.

The following figures depict the behavior of variables such as Employment variation rate, Euribor 3-month rate and several days passed (pdays).

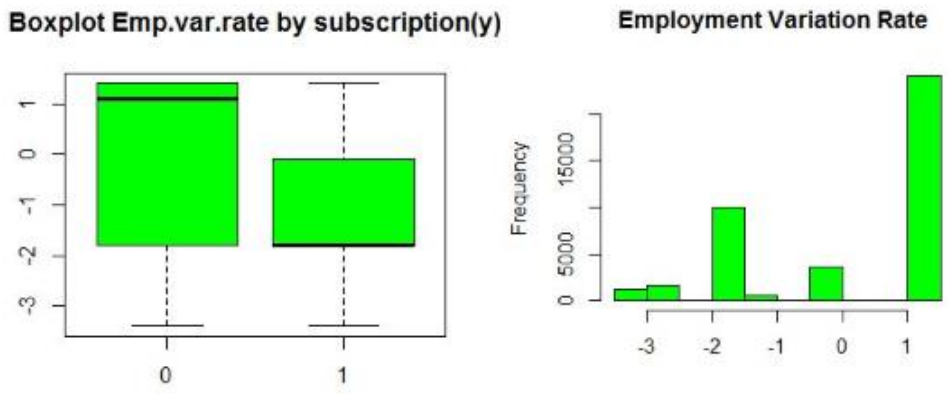

Figure 6: Boxplot and Frequency graph for a continuous variable, Employment Variation Rate.

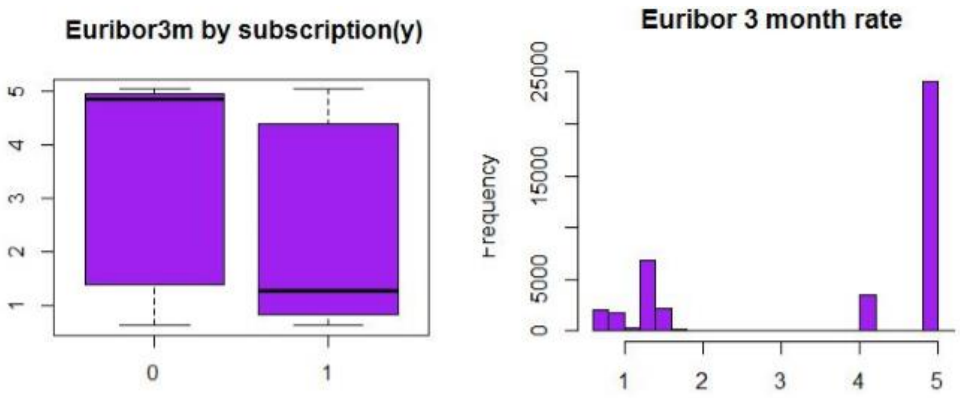

Figure 7: Boxplot and Frequency graph for a continuous variable, Euribor 3-month rate.

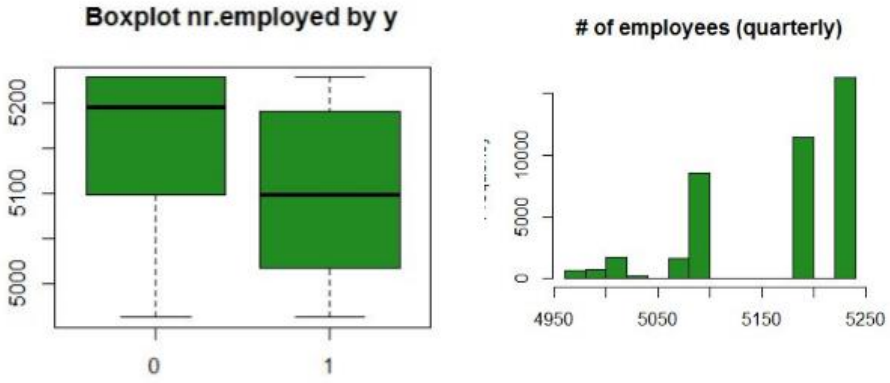

Figure 8: Boxplot and Frequency graph for a continuous variable, # of employees.

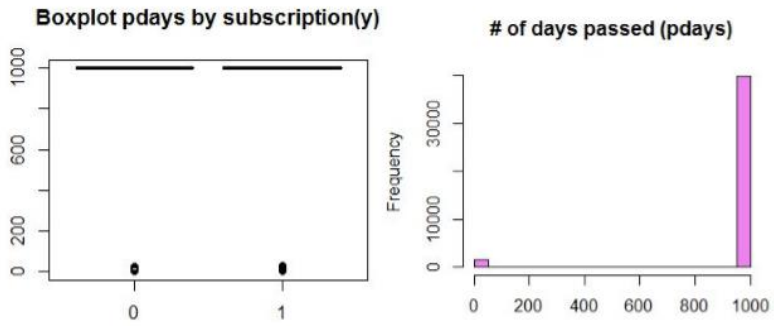

Figure 9: Boxplot and Frequency graph for a continuous variable, pdays.

Above figures being multi-model shows that they are nor spread out evenly. The variable duration is skewed to the right. Number of employees is skewed to the right as well. Figure 6,7 and 8 shows that data is not spread out evenly and to capture the market during the campaign, all the variables needs to be in consideration. On the other hand, Figure 9 depicts the number of days passed after the campaign the client has been contacted. Approximately, 80% of the clients were not contacted after the campaign according to the frequency graph.

Correlation Matrix for Continuous variables

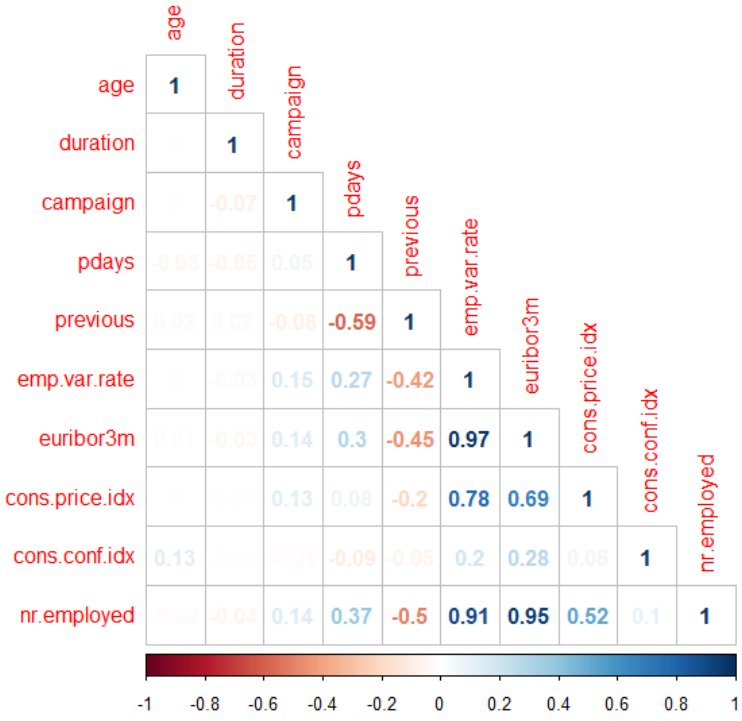

Figure 10: Correlation Matrix.

The above figure is showing correlation coefficients among variables. Each cell in the figure has the correlation between two variables. On our data, it shows that variables are perfectly correlated with each other as all the diagonal values are 1. Correlation matrix helped us to summarize our data where our goals were to see patterns and analyze the behavior of the variables.

Categorical Variables

Categorical variable is a variable which takes limited and fixed values to a particular group based on quantitative property. For instance, race, sex, educational level, married status are all categorical variables.

Graphs were created for the categorical variables job, education, marital, loan default, housing, contact, loan, p outcome, month, and day of the week.

Following graphs examine the categorical variables against the target binary variable (Y). The following graph helped us to determine the behavior of categorical variables in the data set. There were no missing values in the data set, so we did not do any imputation with categorical variables as well.

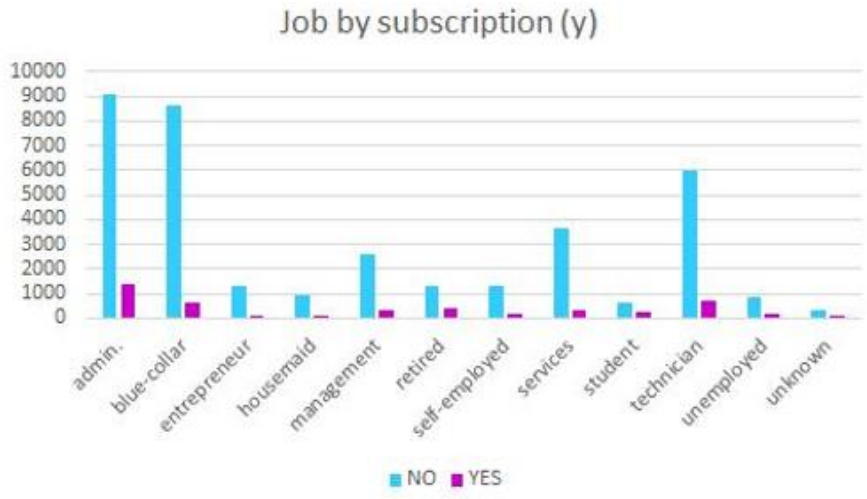

Figure 11: Graph for the categorical variable, Job.

Figure 11 shows the job of the customers subscribed during the campaign against binary variable. After examining, 1.6% of total observations moved into the category of admin being largest. It can be advantageous to reduce the number of categories but there is no commonality amongst the levels to be combined into lesser categories. Although, admin and blue-collar being the largest categories as shown in the figure will be used to analyze further.

Figure 12: Graph for the categorical variable, Marital Status.

Figure 12 depicts the client's marital status against the target variable, Y. After analyzing, we realized 45% of total observations are in the category of 'married'. It shows that the greatest number of subscriptions are from married people followed by single subscribers.

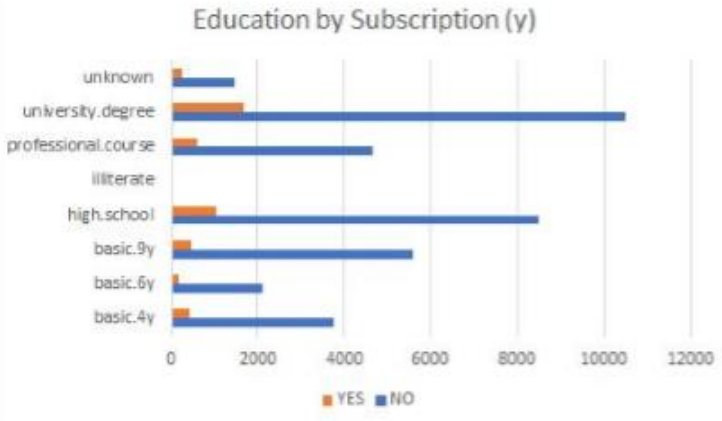

Figure 13: Graph for the categorical variable, Education.

Figure 13 depicts the subscriber's education which will help us to analyze that which clients are more interested in the subscription. It shows that 9.46% of the subscribers roll into the category of university degree holders. There were only 18 observations for Illiterate, so it can be dropped for further examining purposes as 18 observations are not enough to make proper interpretation.

Figure 14: Graph for the categorical variable, Month.

Figure 14 shows the months of the most and least subscription made during the campaigns. It shows that campaigns in the month of May, June, July and August were most successful.

The above graphs for continuous and categorical variables helped us to know the behavior of the data set. Now we know how the variables behave against the target variable, we will use Binary Logistic Regression, Classification and Regression trees and K- nearest neighbors' algorithms will be used to classify the data.

KNN

The K-nearest neighbor method is simplest classification method that classifies based on distance measures. The positive side of this method is that it requires no fitting and it is purely data driven not model driven. Therefore, no assumptions are needed for this method. **Accuracy of this model is 90.42%** with a **misclassification error rate** for **k = 17 is 0.0958**.

K-NN Confusion Matrix

actual default	predicted default 0	1	Total
0	14262	357	14619
	0.866	0.022	
1	1222	634	1856
	0.074	0.038	
Total	15484	991	16475

Training the Knn model (Explanation):

We are using Caret package – train method for training our data for knn algorithm.
Before train() method, we will first use trainControl() method. It controls the computational nuances of the train() method.

We are setting 3 parameters of trainControl() method. The "method" parameter holds the details about resampling method. We can set "method" with many values like "boot", "boot632", "cv", "repeatedcv", "LOOCV", "LGOCV" etc. For this project, we used repeatedcv i.e., repeated cross-validation.

The "number" parameter holds the number of resampling iterations. The "repeats " parameter contains the complete sets of folds to compute for our repeated cross-validation. We are using setting number =10 and repeats =3. This trainControl() methods returns a list. We are going to pass this on our train() method.
Before training our knn classifier, set.seed().

For training knn classifier, train() method should be passed with "method" parameter as "knn". We are passing our target variable y. The y ~. denotes a formula for using all attributes in our classifier and y as the target variable. The "trControl" parameter should be passed with results from our trianControl() method. The "preProcess" parameter is for preprocessing our training data.

As discussed earlier for our data, preprocessing is a mandatory task. We are passing 2 values in our "preProcess" parameter "center" & "scale". These two helps for centering and scaling the data. After preProcessing these convert our training data with mean value as approximately "0" and standard deviation as "1". The "tuneLength" parameter holds an integer value. This is for tuning our algorithm.

Trained Knn model result:

Based on the Accuracy and Kappa metrics result for different k value, our training model is choosing **k = 17** as its final value.

```
k-Nearest Neighbors

24713 samples
   20 predictor
    2 classes: '0', '1'

Pre-processing: centered (20), scaled (20)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 22241, 22243, 22242, 22242, 22241, 22242, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.8997558  0.4113874
   7  0.9022379  0.4151427
   9  0.9033439  0.4150406
  11  0.9042610  0.4126522
  13  0.9051781  0.4136986
  15  0.9055693  0.4125744
  17  0.9058525  0.4106262
  19  0.9056771  0.4070219
  21  0.9054207  0.4025357
  23  0.9053534  0.3985682

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 17.
```

We can see variation in Accuracy w.r.t K value by plotting these in a graph.

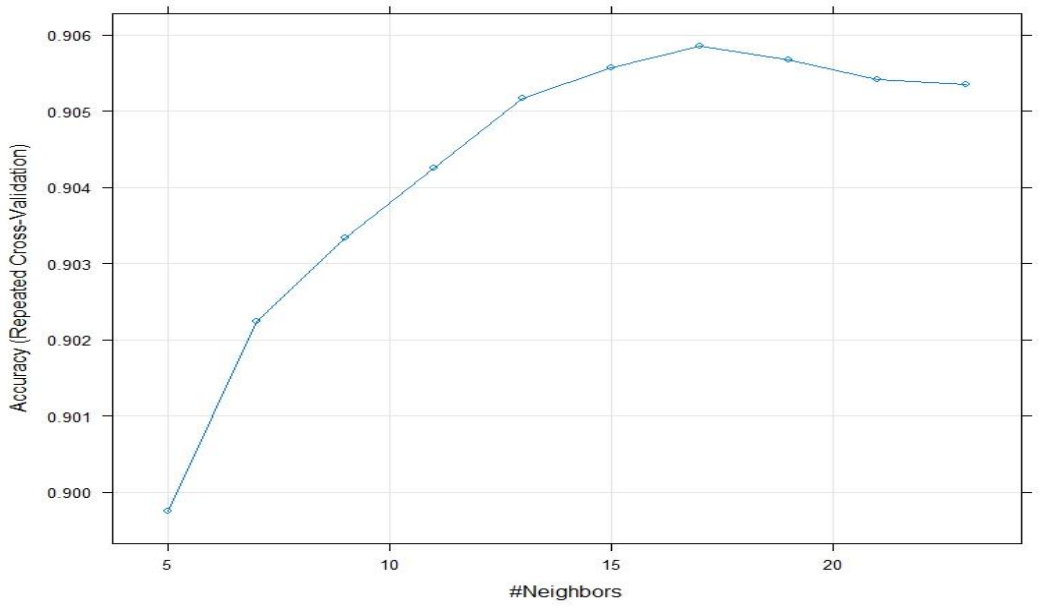

Figure 15: #Plotting yields Number of Neighbors Vs accuracy (based on repeated cross validation)

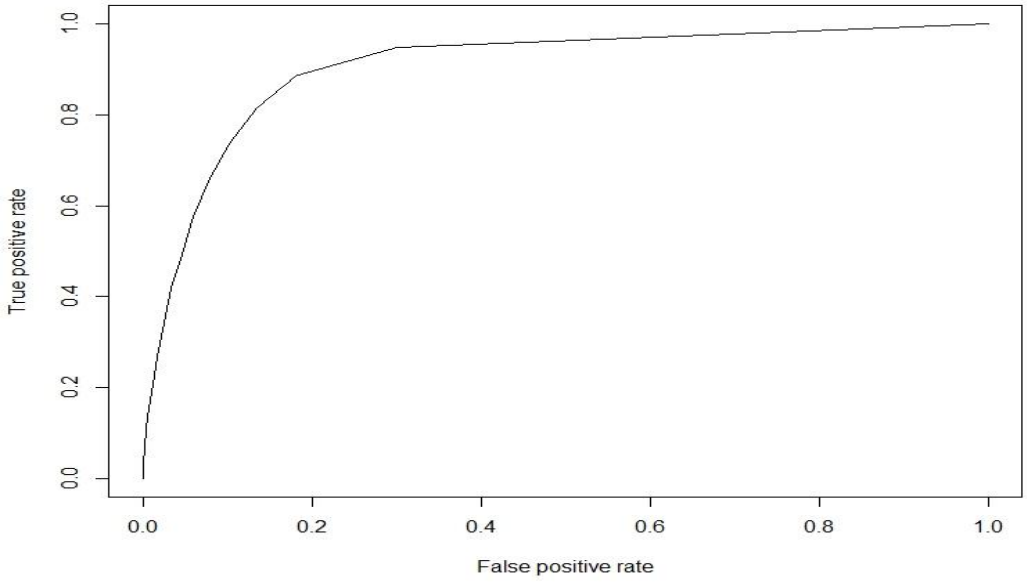

Figure 16: Area Under Curve

Below table gives the detailed output of the result.

Algo.	Cut-off	Accuracy	Misclassification	Sensitivity	Specificity	Predictions
	0.3	0.8987	0.1013	0.9397	0.5754	predicted default actual default 0 1 Total 0 13738 881 14619 0.834 0.053 1 788 1068 1856 0.048 0.065 Total 14526 1949 16475
KNN	0.4	0.9021	0.0979	0.9540	0.4930	predicted default actual default 0 1 Total 0 13947 672 14619 0.847 0.041 1 941 915 1856 0.057 0.056 Total 14888 1587 16475
	0.5	0.9042	0.0958	0.9756	0.3416	predicted default actual default 0 1 Total 0 14262 357 14619 0.866 0.022 1 1222 634 1856 0.074 0.038 Total 15484 991 16475

SUMMARY:

Using confusion matrix analysis, for **cut-off = 0.5**, we found that the best KNN works best with **90.42 % Accuracy** and with misclassification rate **9.58%.**

Classification Tree

1. THE FULL TREE

The training data set is used to grow the full tree using the "rpart" function for the method class. The full tree is grown as follows:

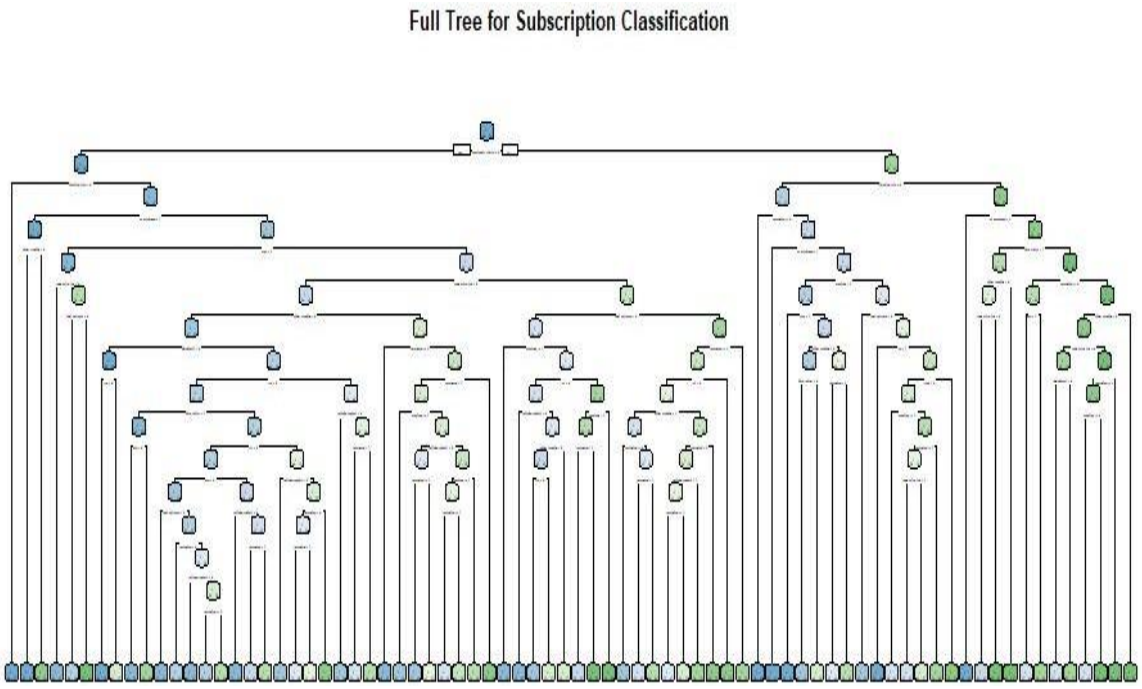

Figure 17: Full Tree for Subscription Classification

The full tree is fully grown and includes all predictor variables as decision nodes, the full tree includes 75 splits. Due to the large number of decision nodes, the data is over plotted and thus, is hard to interpret. Further investigation of the tree's predictive power is required to assess its performance. Using the validation data set, the confusion matrix below is generated using the table function:

p	0	1
0	14200	1219
1	317	595

Figure 18: Confusion matrix using the table function

Based on the confusion matrix, the full tree successfully predicted 14200 records of class 0 as class 0, and successfully predicted 595 records of class 1 as class 1. Nevertheless, the full tree misclassified 1219 records of class 1 as class 0, and additionally misclassified 317 records of class 0 as class 1. Therefore, the accuracy of the full tree is 90.6%, while the sensitivity and specificity are respectively equal to 65.2% and 96%.

2. THE MINIMUM ERROR TREE

The full tree is pruned using the prune function from the *"rpart"* package to prune the full tree based on the lowest complexity parameter that yields the lowest error.

	CP	nsplit	rel error	xerror	xstd
1	0.0990634006	0	1.0000000	1.0000000	0.01786913
2	0.0252161383	1	0.9009366	0.9009366	0.01706829
3	0.0054935159	2	0.8757205	0.8757205	0.01685457
4	0.0036023055	7	0.8364553	0.8454611	0.01659240
5	0.0032420749	9	0.8292507	0.8364553	0.01651313
6	0.0028818444	10	0.8260086	0.8339337	0.01649083
7	0.0021613833	12	0.8202450	0.8299712	0.01645569
8	0.0016210375	13	0.8180836	0.8288905	0.01644609
9	0.0012608069	15	0.8148415	0.8263689	0.01642365
10	0.0011707493	17	0.8123199	0.8260086	0.01642044
11	0.0009005764	21	0.8076369	0.8256484	0.01641723

Based on the complexity parameter table, the minimum error tree contains 21 splits, has a complexity parameter of 0.0009 and yields an error of 0.8076369. In order to grow the most optimal minimum error tree, the complexity parameter table is used to select the number of splits for which the complexity parameter is the lowest and for which the sum of the tree's relative error and the standard error is less than the x-val relative error. Indeed, the minimum error tree error is equal to 0.8076369 and standard error is equal to 0.01641723. The sum of these two values is equal to 0.82405413 which is less than the x-val relative error of 0.8256484. Therefore, the minimum error tree contains 21 splits and 22 terminal nodes as shown below:

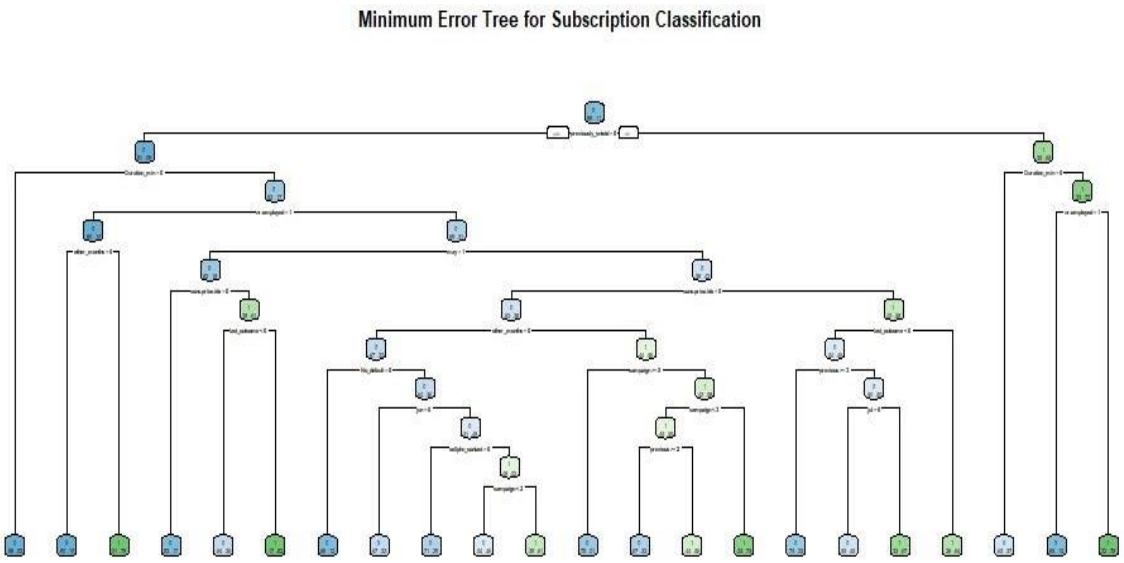

Figure 19: Minimum Error Tree for Subscription Classification

The minimum tree contains fewer decision nodes and fewer terminal nodes than the full tree. Nonetheless, multiple predictor variables with strict rules of sparse data are still included which might increase the risk of overfitting. Further investigation of the tree's predictive power is required to assess its performance. Using the validation data set, the confusion matrix below is generated using the table function:

p1	0	1
0	14230	1213
1	287	601

Based on the confusion matrix, the minimum error tree successfully predicted 14230 records of class 0 as class 0, and successfully predicted 601 records of class 1 as class 1. Nevertheless, the minimum error tree misclassified 1213 records of class 1 as class 0, and additionally misclassified 287 records of class 0 as class 1. Therefore, the accuracy of the minimum error tree is 90.8%, while the sensitivity and specificity are respectively equal to 67.7% and 96%.

3. THE BEST PRUNED TREE

The best pruned tree yields an error which is within one standard error of the minimum error tree. In order to select the best pruned tree, the complexity parameter plot of the minimum error tree is used to identify the location of the lowest error within one standard error of the minimum error tree.

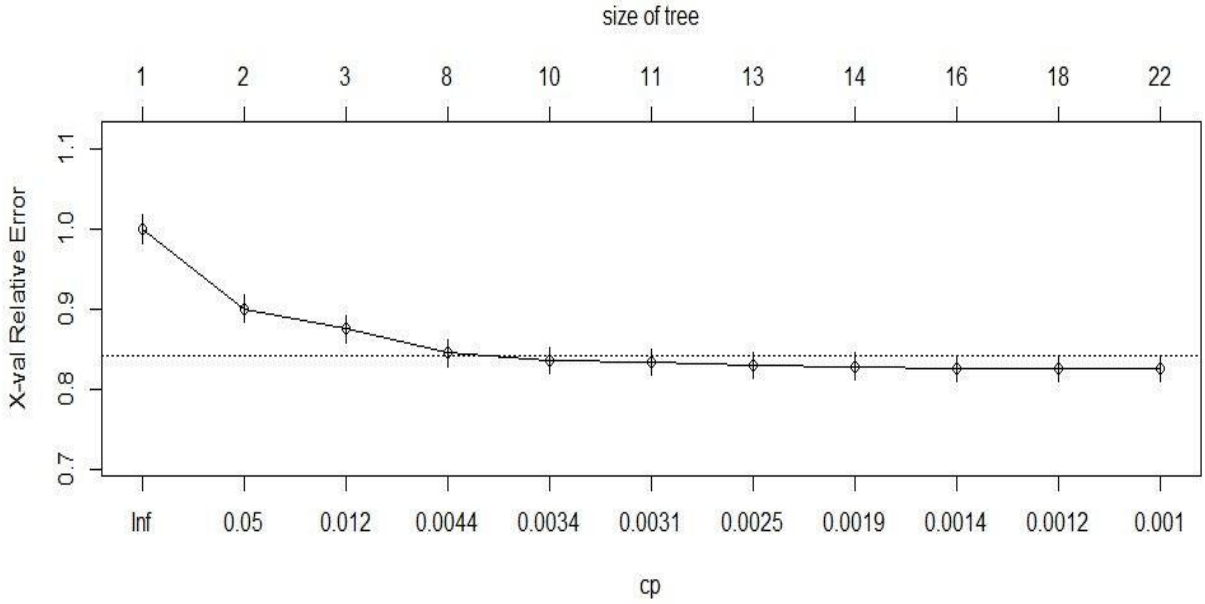

Figure 19: CP Plot

Based on the plot, it seems that the tree containing 18 splits has the lowest error which is within one standard error of the minimum error tree. In order to grow the best pruned tree, the complexity parameter of 0.0012 will be used to control the *"cp"* feature in the *"rpart"* function. The best pruned tree is grown as follows:

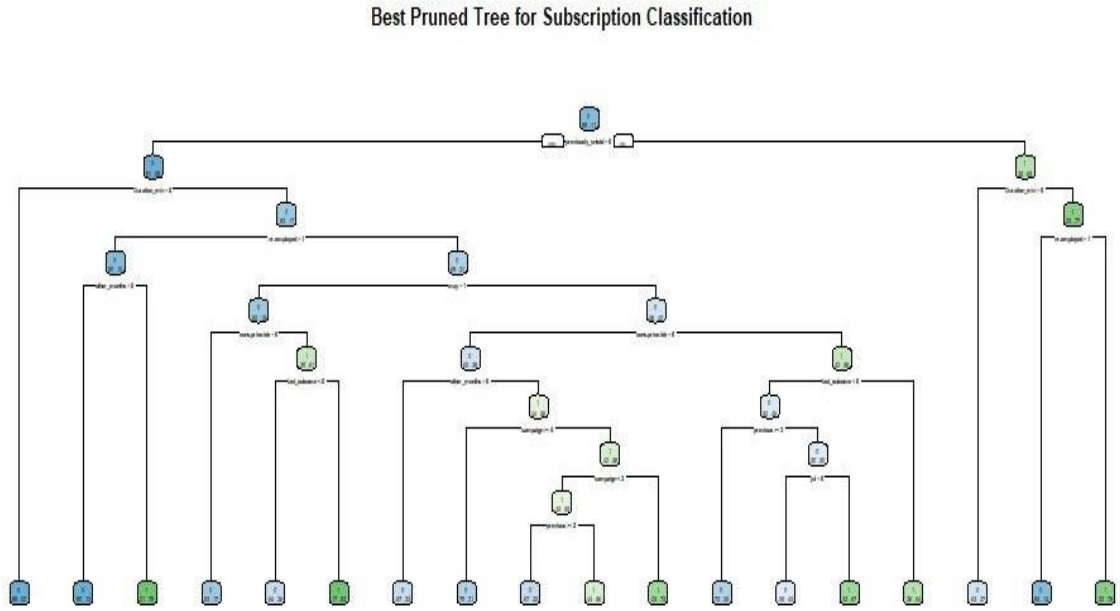

Figure 20 : Best Pruned Tree for Subscription Classification

In order to assess the performance of the best pruned tree, the confusion matrix is generated using the validation data set as follows:

p2	0	1
0	14251	1246
1	266	568

Based on the confusion matrix, the best pruned tree successfully predicted 14251 records of class 0 as class 0, and successfully predicted 568 records of class 1 as class 1. Nevertheless, the best pruned tree misclassified 1246 records of class 1 as class 0, and additionally misclassified 266 records of class 0 as class 1. Therefore, the accuracy of the best pruned tree is 90.7%, while the sensitivity and specificity are respectively equal to 68.1% and 96.2%.

4. SUMMARY

The analyses of the full tree, the minimum error tree and the best pruned tree are summarized in the table below:

	FULL TREE	MINIMUM ERROR TREE	BEST PRUNED TREE
#OF SPLITS	75	21	18
ACCURACY	90.60%	90.80%	90.70%
SENSITIVITY	65.20%	67.70%	68.10%
SPECIFICITY	96%	96%	96.20%

Logistic Regression

Logistic regression is used for the classification as the output is binary. It is a predictive, model based supervised learning algorithm. It does allow variable reduction. Since our data has many variables we would need to take a prudent and informed decision based on different tools and analysis such as histograms, box plots etc. The purpose of this process is to identify the significant variables apply the algorithms to get suggestion on significant variables, calculate, accuracy, misclassification error, sensitivity and specificity.

Out of 40,766 records and 15 variables, following variables are found as significant.

1. Duration
2. Number of employees
3. Pdays
4. Consumer price index
5. Poutcome
6. Job – blue collar category
7. Default - category
8. Contact – cellphone category
9. Other Months
10. May
11. June
12. July
13. August
14. Previous
15. Y (Output)

Three parameter selection methods were employed and each were run on three cut-offs i.e. 0.3, 0.4, 0.5. First one is being forward selection. In this each step begins with 0 predictors and do not add significance. The purpose of the addition of each predictor is to get the better R^2

In the backward selection, we start with all the predictors together, then eliminate the predictor at each step. The method keeps eliminating the predictor until the elements that are statistically significant are present. Backward elimination generally takes more time than other variable selection methods.

The third and last one is Stepwise selection which is a combination of forward and backward methods. It can be combined with both of the prior methods but mostly used with backward elimination.

Below table gives the detailed output of the result.

SL No.	Cut-off	Accuracy	Misclassification	Sensitivity	Specificity	Predictions
	0.3	0.a09409244	0.09409244	0.8867615	0.8389743	0 1 0 13703 733 1 804 1095
Forward	0.4	0.09146006	0.09146006	0.8867615	0.8389743	0 1 0 13949 936 1 558 892
	0.5	0.09139884	0.09139884	0.8867615	0.8389743	0 1 0 14112 1098 1 395 730
	0.3	0.09409244	0.09409244	0.8867615	0.8389743	0 1 0 13703 733 1 804 1095
Backward	0.4	0.09146006	0.09146006	0.8867615	0.8389743	0 1 0 13949 936 1 558 892
	0.5	0.09139884	0.09139884	0.8867615	0.8389743	0 1 0 14112 1098 1 395 730
	0.3	0.09409244	0.09409244	0.8867615	0.8389743	0 1 0 13703 733 1 804 1095
Stepwise	0.4	0.09146006	0.09146006	0.8867615	0.8389743	0 1 0 13949 936 1 558 892
	0.5	0.09139884	0.09139884	0.8867615	0.8389743	0 1 0 14112 1098 1 395 730

We can see the prediction matrix and data are getting repeated for all the three forward selection, backward elimination, and stepwise selection. The detailed observations can be calculated from Prediction matrix.

Gradient Boosting

Boosting is an ensembling method. Literal meaning of ensembling is orchestra. Here the outcome is symphony or melody of all the participating instruments. Following the same principle and adding the randomness, this model randomly picks some records. The weights are the same, the probability of any record to be picked also is the same. First decision tree is made, and misclassified records are calculated. Now, before creating a second decision trees the weights of all the misclassified records are increased. As the weights of the misclassified records in the previous decision tree is increased, the probability of picking that records by the algorithm are high. All the subsequent decision trees that are made are called the weak classifiers. After creating N decision trees or the weak classifiers, all these weak classifiers are combined to generate on final classifier.Making sequential decision trees help XGBoost based model to produce higher accuracy than CART in most of the cases.

The Bernoulli distribution was used to specify the data and assume 1000 trees. The boosted model used 23,512 observations, 14 predictors and 2 classes (yes and no). For this model 6 variables were centered, 6 were scaled and 8 were ignored. The data was resampled with cross validation by 3 fold for higher accuracy as shown in figure 18. The corresponding sample sizes were 15,674, 15,675 and 15,674. The tuning parameter "shrinkage" was held constant at a value of 0.1, "n.minobsinnode" was held constant at a value of 10. ROC was used to select the optimal model using the largest value. Overall the results shown below collected with a model were n.tree equaled 150, interaction.depth was 3, shrinkage was 0.1 and n.minobsinnode was 10.

interaction.depth	n.trees	ROC	Sens	Spec
1	50	0.9292165	0.2410715	0.9876118
1	100	0.9341969	0.3420408	0.9795762
1	150	0.9361241	0.4180463	0.9745061
2	50	0.9392583	0.4099725	0.9745061
2	100	0.9418854	0.4671847	0.9700100
2	150	0.9429962	0.4863780	0.9684316
3	50	0.9425032	0.4668077	0.9694839
3	100	0.9439473	0.5063428	0.9669489
3	150	0.9448675	0.5051894	0.9666619

Fig. Boosting model resampling result

The test value results are as below:

SL No.	Variables	Value
1	Sensitivity	0.50256
2	Specificity	0.96645
3	Positive Prediction Value	0.65382
4	Negative Prediction value	0.93907
5	Prevalence	0.11195
6	Detection rate	0.05626
7	Detection prevalence	0.08606
8	Balanced accuracy	0.73451

Sl No.	Variable	Rel.inf
duration	duration	47.68998256
nr.employed	nr.employed	26.37933879
month	month	8.81503443
pdays	pdays	5.82604150
poutcome	poutcome	3.77269490
cons.conf.idx	cons.conf.idx	2.21543708
job	job	1.63186685
day_of_week	day_of_week	1.33801406
age	age	0.79604965

contact	contact	0.71980846
education	education	0.61039431
default	default	0.10991110
campaign	campaign	0.05536722
marital	marital	0.04005908

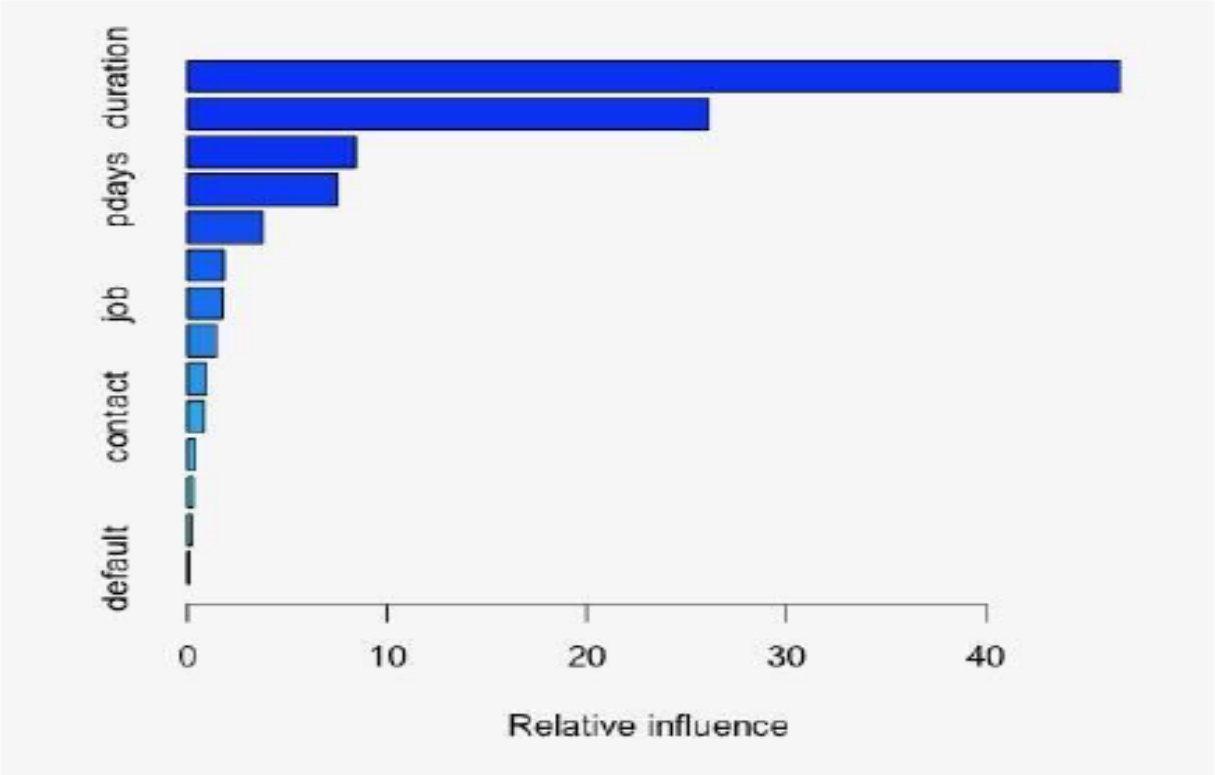

Fig 21: Boosted model

To determine correlation for categorical variables goodness of fit was used. This resulted in showing that variables loan, day of week, and housing were not significant.

Results

SL No.	Model	Sensitivity	Specificity	Accuracy	Misclassification	Best Model	
1	Logistics	0.8867615	0.8389743	0.90860116	0.09139884		
2	XGBoost	0.49972	0.96609	0.9139	0.0861	**Best**	
3	Best Pruned Tree	0.6520	0.6770	0.9060	0.094		
4	KNN	0.9756	0.3416	0.9042	0.0958		

Conclusion

The purpose of this study was to find the best model to predict the factors that contributed the most for every term deposit account opened between 2008 to 2010 for a Portuguese bank. All four models were run on the same dataset and with the same seed to get accurate results. The three models that were tested are KNN, CART, XGBoost and logistic regression. Confusion matrix, sensitivity, specificity, accuracy, and misclassification error rates are used to judge the performance of each model. For this dataset we need high sensitivity and low specificity.

Future Scope and Recommendations.

1. C 5.0

C 5.0 algorithm is the successor to C 4.5. It is the industry standard and used very often in place of CART. This is because CART uses GINI index as its splitting criteria and C 5.0 uses information gain and entropy as its splitting criteria. Using information gain and entropy gives better results in most cases. Entropy in common language can be defined as the messiness of data. After each split in the decision trees, the reduction in the entropy is called the information gain. We want to achieve maximum information gain at each split. Testing C 5.0 algorithm is also recommended as it might perform better than CART.

2. Tuning Hyperparameter

In machine learning hyperparameters are the parameters whose value control the learning process and are set before the model is put to learn. For the purpose of this report we mostly used the default parameters. The problem with setting the parameters is that there are a lot of possible

combinations and it is difficult to choose the most optimal combination for a model. There are three ways in which the hyperparameters can be tuned:

a. Manual Tuning - We can manually tune the parameters and set the best combination based on the previous experience with that model. It is difficult to choose the right combination as each dataset behaves differently with a specific set of combination.
b. Full Search - There are a couple of libraries like caret that allow you to run all the possible combinations of the parameters and return the best performing combination. This is a good option if the dataset is small. But the problem comes when the dataset is large. We can not afford the time it takes to try all the combination on a large dataset.
c. Randomized search - To solve the problem of efficiency with full search, randomized search is used to find a good combination of hyperparameters. Randomized search is not as effective as full search because it may skip on some good combination, but it is very efficient. If the dataset is large it is recommended to use the randomized search.

For the purpose of our project randomized hyperparameter search is recommended. The dataset is large so it will be difficult to try all possible combinations with full search. Randomized search will be more effective and efficient.

References

- Akwei, John., “Bank Marketing Data Classification” (November 2017). http://rstudio-pubs- static.s3.amazonaws.com/330635_587540af094c461d9e58a5b9b697e1db.html

- Colaianni, Gina., Magdangal., & Mitchell, Matthew., “Factors Determining Term Deposit Purchases” (December 2016). https://support.sas.com/resources/papers/proceedings17/2029-2017.pdf

- Data from: UCI Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/bank+marketing

APPENDIX:

CODE SNIPPETS

KNN:-

```
rm(list=ls()); gc()
usePackage <- function(p) {
  if (!is.element(p, installed.packages()[,1]))
    install.packages(p, dep = TRUE)
  require(p, character.only = TRUE)
}
usePackage('ggplot2')
# For Accuracy Calculation
usePackage('caret')

usePackage('e1071')
usePackage('rpart.plot')
usePackage('rattle')
usePackage('randomForest')
usePackage('caTools')
usePackage('descr')
usePackage('rpart')
usePackage('RColorBrewer')
usePackage('knitr')
usePackage('tidyr')
usePackage('dplyr')
usePackage('ROCR')
```

```
#Data Extraction

data <- read.csv(file="M:/Semester 3/ISDS 574/FInal Project/bank-additional-full.csv", header=TRUE, sep=";")


str(data) #structure of our data frame


dim(data) # To check dimension of the data-set


#summary before cleaning

summary(data)

##############################################

# Exploratory Data Analysis & Cleaning Data:

##############################################


# Check for any missing values:

sum(is.na(data)) # 0 which means we have a clean data set


# Transforming character data into numeric format

data$job <- as.numeric(as.factor(data$job))

data$marital <- as.numeric(as.factor(data$marital))

data$education <- as.numeric(as.factor(data$education))

data$default<- ifelse(data$default == "yes", 1, 0)

data$housing <- ifelse(data$housing== "yes", 1, 0)

data$loan<- ifelse(data$loan== "yes", 1, 0)

data$month <- as.numeric(as.factor(data$month))

data$contact <- as.numeric(as.factor(data$contact))

data$poutcome <- as.numeric(as.factor(data$poutcome))

data$age <- as.numeric(as.factor(data$age))

data$duration <- as.numeric(as.factor(data$duration))
```

```
data$campaign <- as.numeric(as.factor(data$campaign))
data$pdays <- as.numeric(as.factor(data$pdays))
data$previous <- as.numeric(as.factor(data$previous))
data$emp.var.rate <- as.numeric(as.factor(data$emp.var.rate))
data$cons.price.idx <- as.numeric(as.factor(data$cons.price.idx))
data$cons.conf.idx <- as.numeric(as.factor(data$cons.conf.idx))
data$nr.employed <- as.numeric(as.factor(data$nr.employed))
data$day_of_week <- as.numeric(as.factor(data$day_of_week))

#Converting output variable to factor
data$y <-as.factor(ifelse(data$y== "yes", 1, 0))

#checking classes of attributes after transformation
sapply(data,class)



#Data partitioning
set.seed(3033)
intrain <- createDataPartition(y = data$y, p= 0.6, list = FALSE)
training <- data[intrain,]
testing <- data[-intrain,]

#check dimensions of train & test set
dim(training)
dim(testing)
#check whether any NA value exists or not
anyNA(data)

summary(data) #summary stats of our data
```

```
training[["y"]] = factor(training[["y"]]) #conversion of V1 integer variable to factor variable

#Training & Preprocessing
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
knn_fit <- train(y ~., data = training, method = "knn",
                 trControl=trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)

#knn classifier
knn_fit

#plot accuracy vs K Value graph
plot(knn_fit)

#predict classes for test set using knn classifier
#test_pred <- predict(knn_fit, newdata = testing)
test_pred <-predict(knn_fit, testing, "prob")[,2]

#- Area Under Curve
plot(performance(prediction(test_pred, testing$y),
                 "tpr", "fpr"))

# use probability cut off  for classification
test_pred = ifelse(test_pred > 0.3, 1,0)
test_pred = ifelse(test_pred > 0.4, 1,0)
test_pred = ifelse(test_pred > 0.5, 1,0) # best accuracy
#- confusion matrix
```

```
confusionMatrix(factor(test_pred),
                factor(testing$y))

#Computing Misclassification rate for CART
knn_miss <- 1- 0.9042 #1 - Accuracy

# In KNN we have  90.42 % Accuracy with misclassification rate  9.58%



### Cross table validation for KNN
CrossTable(factor(testing$y), factor(test_pred),
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

Classification Tree:-

```
rm(list=ls()); gc()
usePackage <- function(p) {
  if (!is.element(p, installed.packages()[,1]))
    install.packages(p, dep = TRUE)
  require(p, character.only = TRUE)
}
usePackage('rpart')
usePackage('rpart.plot')
usePackage('caret')
usePackage('pROC')
usePackage('mlbench')
```

```
#Data Extraction
clean_bank<-read.csv("bank-additional-full.csv",sep=";", stringsAsFactors = FALSE)
set.seed(1234)
pd = sample(2,nrow(clean_bank), replace = TRUE, prob = c(0.6,0.4))
training = clean_bank[pd==1,]
validation = clean_bank[pd==2,]
#Full Tree
tree = rpart( y ~ . , training , method = "class" , cp = 0 )
rpart.plot( tree , main = 'Full Tree for Subscription Classification' , extra = 4 )
tree
tree$cptable
plotcp(tree)
p = predict( tree , validation , type = "class" )
tab =table( p , validation $ y )
tab

#Minimum Error Tree
tree2 = prune( tree , cp = tree $ cptable [which.min( tree $ cptable [, "xerror" ]), "CP" ])
rpart.plot( tree2 , main = 'Minimum Error Tree for Subscription Classification' , extra = 4 )
tree2

p1 = predict( tree2 , validation , type = "class" )
tab1 =table( p1 , validation $ y )

tab1
tree2$cptable
plotcp( tree2 )

#Best Pruned Tree
```

```
tree3 = rpart( y ~ . , training , method = "class" , cp = 0.0012 )
rpart.plot( tree3 , main = 'Best Pruned Tree for Subscription Classification' , extra = 4 )
p2 = predict( tree3 , validation , type = "class" )
tab1 =table( p2 , validation $ y )
tab1
```

XGBoost :-

```
bank.additional.full <- read.csv("bank-additional-full.csv",
                                 header=TRUE, sep=";")
#Loading the packages
library(MASS)
library(ggplot2)
library(caret)
library(corrplot)
library(e1071)
#Viewing the dataset
head(bank.additional.full,3)
str(bank.additional.full)
##preprocessing

##separating continuous and categorical variables for checking significance of the variables
bank_continous = subset(bank.additional.full, select = c(age,
                                                         duration,campaign,pdays,previous,emp.var.rate,euribor3m,
cons.price.idx,cons.conf.idx,
                                                         nr.employed) )


bank_categorical=subset(bank.additional.full, select=-c(age,
                                                        duration,campaign,pdays,previous, emp.var.rate,euribor3m,
```

```
                                    cons.price.idx,cons.conf.idx, nr.employed))



corr <- cor(bank_continous)
corrplot(corr = corr,method='square', type = 'full')
###nr.employed is correlated with euribor3m, emp.var.rate
##negative correlation btw previous and pdays
##we will remove euribor3m, emp.var.rate and previous
##cons.price.idx is correlated with euribor3m, emp.var.rate
bank_removed_correlations=bank.additional.full[,-c(14,16,17,19)]
#Goodness of fit for categorical variables
chisq.test(bank_categorical$housing,bank_categorical$y)
##remove housing p value is very less for both loan and housing
chisq.test(bank_categorical$job,bank_categorical$y)
chisq.test(bank_categorical$loan,bank_categorical$y)
##loan is not significant
chisq.test(bank_categorical$marital,bank_categorical$y)
chisq.test(bank_categorical$month,bank_categorical$y)
chisq.test(bank_categorical$contact,bank_categorical$y)
chisq.test(bank_categorical$day_of_week,bank_categorical$y)
chisq.test(bank_categorical$poutcome,bank_categorical$y)



bank_removed_correlations=bank_removed_correlations[,-c(6,7)]
str(bank_removed_correlations)
```

```
#Removing unknown rows
table(bank_removed_correlations$job)
unknownjob= which(bank_removed_correlations$job %in% c('unknown'))
bank_updated=bank_removed_correlations[-(unknownjob),]
table(bank_updated$job)


table(bank_removed_correlations$marital)
unknownmarital= which(bank_updated$marital %in% c('unknown'))
bank_updated=bank_updated[-(unknownmarital),]


table(bank_removed_correlations$education)
unknowneducation= which(bank_updated$education %in% c('unknown'))
bank_updated=bank_updated[-(unknowneducation),]


table(bank_removed_correlations$default)
unknowndefault= which(bank_updated$default%in% c('yes'))
##the people who have default cannot take bank term deposit
bank_updated=bank_updated[-(unknowndefault),]
table(bank_removed_correlations$contact)
table(bank_removed_correlations$month)
table(bank_removed_correlations$day_of_week)
table(bank_removed_correlations$poutcome)
#View(bank_updated)
#########################################################################
```

```
set.seed(1)
##60% training and 40% test
training_size <- floor(0.60 * nrow(bank_updated))
train_ind <- sample(seq_len(nrow(bank_updated)), size = training_size)
training <- bank_updated[train_ind, ]
testing <- bank_updated[-train_ind, ]
#####################################################################

library(gbm)
is.factor(bank_updated$y)
bank_updated$y=as.numeric(bank_updated$y)
bank_updated$y=factor(x=bank_updated$y, levels = c(2,1),labels = c("yes","no"))
bank_updated$y



?trainControl
outcomeName <- 'y'
predictorsNames <- names(bank_updated)[names(bank_updated) != outcomeName]
##performing cross validation for higher accuracy
objControl <- trainControl(method='cv', number=3, returnResamp='none',
                    summaryFunction = twoClassSummary, classProbs = TRUE, )

objModel <- train(training[,predictorsNames], training[,outcomeName],
              method='gbm',
              trControl=objControl,
              metric = "ROC", preProc = c("center", "scale"))
```

```
?postResample
summary(objModel)
print(objModel)
predictions <- predict(object=objModel, testing[,predictorsNames], type="raw")
head(predictions)
print(postResample(pred=predictions, obs=as.factor(testing[,outcomeName])))
confusionMatrix(predictions, testing$y, positive='yes')
```

Logistic Regression:-

```
rm(list=ls())
library(corrplot)
library(MASS)
library(ggplot2)
library(caret)
library(corrplot)
rm(list=ls())
#setwd('C:\\Users\\Priyanka\\Desktop\\bank\\bank-additional')
bank1<-read.csv("bank-additional-full.csv",sep=";", stringsAsFactors = FALSE)
#exporting the well-ordered datafile
write.csv(bank1,"bank1.csv")
dim(bank1) #41188 obs and 21 variables---20 independent variables (10 char,5 int, 5 num)
str(bank1) # there are
summary(bank1)
colnames(bank1)
#"age" "job" "marital" "education" "default" "housing"
# "loan" "contact" "month" "day_of_week" "duration" "campaign"
```

```
# "pdays" "previous" "poutcome" "emp.var.rate" "cons.price.idx" "cons.conf.idx"

# "euribor3m" "nr.employed" "y"

# numeric variables: age, duration,campaign,pdays,previous,emp.var.rate,

#euribor3m, cons.price.idx,cons.conf.idx

#10 character var : job, marital, education, default, housing, loan, contact, month, day_of_week,

# poutcome

# outcome variable = y (0/1)

bank1$y= ifelse(bank1$y=='yes',1,0)

bank2= as.data.frame(bank1)

# converting character variables to levels/factor

bank2$job=as.factor(bank2$job)

bank2$marital=as.factor(bank2$marital)


bank2$education=as.factor(bank2$education)

bank2$default=as.factor(bank2$default)

bank2$housing=as.factor(bank2$housing)

bank2$loan=as.factor(bank2$loan)

bank2$contact=as.factor(bank2$contact)

bank2$month=as.factor(bank2$month)

bank2$day_of_week=as.factor(bank2$day_of_week)

bank2$poutcome=as.factor(bank2$poutcome)

bank2$y=as.factor(bank2$y)



bank_cont = subset(bank1, select = c(age, duration,campaign,pdays,previous,
```

```
                    emp.var.rate,euribor3m, cons.price.idx,cons.conf.idx,
                    nr.employed) )


cor_cat=cor(bank_cont)
corrplot(cor_cat, method="number", type='lower')

cor_cat_m= bank_cont[,-c(6,7)] # updated continuous dataframe

corrplot(cor(cor_cat), method='number', type='upper',number.cex = 0.7)

bank2=bank2[-c(16,19)] #removed emp.var.rate and euribor3m as highly correlated
bank2=bank2[-c(6,7)]# removed housing and loan from categorical var
colnames(bank2)

#checking for any more correlations
corrplot(cor(cor_cat_m), method='number', type='upper',number.cex = 0.7)
#bank3 = subset(bank1, select = c(y,age, duration,campaign,pdays,previous,
#emp.var.rate,euribor3m, cons.price.idx,cons.conf.idx)


table(is.na(bank1)) # total no missing values
sapply(bank1,function(x)sum(is.na(x))) # no of missing values by column
table(bank1$y) # has the output as no and yes, hence converted as 0 and 1 respectively
str(bank2$y) # to check the structure of the outcome variable
```

```
########################################

##############################VISUALIZATION OF CONTINUOUS
##VARIABLES############
hist(bank1$age, col="blue", main="Age", xlab = " ")
boxplot(age~y, data=bank1, main= 'Boxplot age by subscription(y)', col="blue")
hist(bank1$duration, col="pink", main= "Call duration (seconds)", xlab="")
boxplot(duration~y, data=bank1, main= 'Boxplot duration by subscription(y)', col="pink")
hist(bank1$campaign, col="grey", main= "# of contacts for this campaign", xlab="")
boxplot(campaign~y, data=bank1, main= 'Boxplot campaign by subscription(y)', col="grey")

hist(bank1$pdays, col="violet", main= "# of days passed (pdays)", xlab="")
boxplot(pdays~y, data=bank1, main= 'Boxplot pdays by subscription(y)',col="violet")
hist(bank1$previous, col="aquamarine", main= " previous contact", xlab="")
boxplot(previous~y, data=bank1, main= 'Boxplot of previous by
subscription(y)',col="aquamarine")

hist(bank1$cons.conf.idx, col="brown", main= "Monthly cons.conf.inx", xlab="")
boxplot(cons.conf.idx~y, data=bank1, main= 'Boxplot cons.conf.inx by y', col="brown")
hist(bank1$emp.var.rate, col="green", main= "Employment Variation Rate ", xlab="")
```

```
boxplot(emp.var.rate~y, data=bank1, main= 'Boxplot Emp.var.rate by subscription(y)',
      col="green")


hist(bank1$euribor3m, col="purple", main= "Euribor 3 month rate", xlab="")
boxplot(euribor3m~y, data=bank1, main= ' Euribor3m by subscription(y)', col="purple")
hist(bank1$cons.price.idx, col="orange", main= "Monthly Consumer Price Index", xlab="")
boxplot(duration~y, data=bank1, main= 'Boxplot Mon.Con.Ind by y', col="orange")
hist(bank1$nr.employed, col="forestgreen", main= "# of employees (quarterly)", xlab="")
boxplot(nr.employed~y, data=bank1, main= 'Boxplot nr.employed by y', col="forestgreen")



####################VISA
library(ggplot2)
rm(job1)
with(bank1, table(job, y))
with(bank1, table(marital, y))
with(bank1, table(education, y))

with(bank1, table(default, y))
with(bank1, table(housing, y))
with(bank1, table(loan, y))
with(bank1, table(contact, y))
```

```
with(bank1, table(month, y))
with(bank1, table(day_of_week, y))
with(bank1, table(poutcome, y))
#job1=as.data.frame(job1)
#ggplot(job1, aes(x= job, fill=y)) + geom_bar(color="red", fill="red", width = 0.3)
# ggplot(bank1, aes(x= marital, fill=y)) + geom_bar(color="blue", fill="blue")
#ggplot(bank1, aes(x= marital, fill=y)) + geom_bar(color="blue", fill="blue")



########################checking the significant
###variables#############################################################



#significant categorical variables
bank_cat = subset(bank2, select = -c(age, duration,campaign,pdays,previous,
                                     cons.price.idx,cons.conf.idx,nr.employed) )
chisq.test(bank_cat$job, bank2$y) #p-value < 2.2e-16
chisq.test(bank_cat$marital, bank2$y) #p-value < 2.2e-16
chisq.test(bank_cat$education, bank2$y, simulate.p.value = TRUE) #p value: 0.0004
chisq.test(bank_cat$default, bank2$y, simulate.p.value = TRUE) #p-value = 0.0004998
chisq.test(bank_cat$housing, bank2$y) #not significant p-value = 0.05829
chisq.test(bank_cat$loan, bank2$y) #not significant p-value = 0.5787
chisq.test(bank_cat$contact, bank2$y) #p-value < 2.2e-16
chisq.test(bank_cat$month, bank2$y) #p-value < 2.2e-16
```

```
chisq.test(bank_cat$day_of_week, bank2$y) #p-value = 2.958e-05
chisq.test(bank_cat$poutcome, bank2$y) #p-value < 2.2e-16
#On the basis of this, we are going to remove housing and loan from the data
#bank2=bank2[-c(6,7)]


colnames(bank2)
table(bank2$job)
table(bank2$marital)
table(bank2$education)


table(bank2$default)
table(bank2$contact)
table(bank2$month)
table(bank2$day_of_week)
table(bank2$poutcome)
table(bank2$job)


#admin. blue-collar entrepreneur housemaid management retired
#10422 9254 1456 1060 2924 1720
#self-employed services student technician unemployed unknown
#1421 3969 875 6743 1014 330
#since unknown category is less, can't infer much hence dropping unknown
```

```
#total obs:41188-330 =40858
rm(bank3)
bank3=as.data.frame(bank2)

ind_job= which(bank2$job %in% c('unknown'))
bank3=bank2[-(ind_job),]
table(bank3$job)

table(bank2$marital)
#divorced married single unknown
#4612 24928 11568 80
table(bank3$marital)
#updated tables on bank3: divorced married single unknown
#4599 24694 11494 71
#since unknown category is less, can't infer much hence dropping unknown
#total obs:40858-71 =40787


table(bank2$education)
#basic.4y basic.6y basic.9y high.school
#4176 2292 6045 9515
#illiterate professional.course university.degree unknown
#18 5243 12168 1731
table(bank3$education)
#updated tables on bank3:
```

```
# basic.4y basic.6y basic.9y high.school illiterate
#4172 2289 6033 9501 18
#professional.course university.degree unknown


#5238 12139 1727
# illiterate will not infer anything hence, removed total obs= 40787-18=40769
ind_edu= which(bank3$education %in% c('illiterate'))
bank3=bank3[-(ind_edu),]
table(bank2$default)
#no unknown yes
#32588 8597 3
table(bank3$default)
#no unknown yes
#32337 8429 3
#deleting yes rows on default: total obs:40769-3=40766


ind_def= which(bank3$default %in% c('yes'))
bank3=bank3[-(ind_def),]
table(bank2$contact)
#cellular telephone
#26144 15044
table(bank2$month)
```

```
#apr aug dec jul jun mar may nov oct sep
#2632 6178 182 7174 5318 546 13769 4101 718 570
table(bank2$day_of_week)
#fri mon thu tue wed
#7827 8514 8623 8090 8134
table(bank2$poutcome)

#failure nonexistent success
#4252 35563 1373
##########################################################################
##########
##################
############################CREATING DUMMY
##VARIABLES#####################################
summary(bank3)
bank4=as.data.frame(bank3)



# retired is the reference total 6 variables.total=23 vars
bank4$Admin=bank4$Blue_collar=bank4$Technician=bank4$Services=bank4$M
anagement=bank4$Other_jobs=0

bank4$Admin[which(bank4$job=='admin.')]=1
```

```
bank4$Blue_collar[which(bank4$job=='blue-collar')]=1
bank4$Technician[which(bank4$job=='technician')]=1
bank4$Services[which(bank4$job=='services')]=1
bank4$Management[which(bank4$job=='management')]=1
bank4$Other_jobs[which(bank4$job=='entrepreneur')]=1
bank4$Other_jobs[which(bank4$job=='housemaid')]=1
bank4$Other_jobs[which(bank4$job=='self-employed')]=1
bank4$Other_jobs[which(bank4$job=='student')]=1
bank4$Other_jobs[which(bank4$job=='unemployed')]=1
table(bank4$Other_jobs)


summary(bank3)
#ind1=which(bank4$job %in% c('entrepreneur','housemaid','self-employed','student','unemployed'))
# bank4$job[ind1] = 'other_jobs'
## for marital
table(bank4$marital)
#divorced married single
# 4597 24676 11493
#divorce is reference
bank4$married=bank4$single=0
bank4$married[which(bank4$marital=='married')]=1
bank4$single[which(bank4$marital=='single')]=1
table(bank4$marital)
table(bank4$single)
# contact: no cellphone or no
```

```
table(bank4$contact)
bank4$cellphn_contact=0
bank4$cellphn_contact[which(bank4$contact=='cellular')]=1


table(bank4$cellphn_contact)
#month
table(bank4$month)
bank4$may=bank4$jun=bank4$jul=bank4$aug=bank4$nov=bank4$other_months
=0
bank4$may[which(bank4$month=='may')]=1
bank4$jun[which(bank4$month=='jun')]=1
bank4$jul[which(bank4$month=='jul')]=1
bank4$aug[which(bank4$month=='aug')]=1
bank4$nov[which(bank4$month=='nov')]=1
bank4$other_months[which(bank4$month=='dec')]=1
bank4$other_months[which(bank4$month=='mar')]=1
bank4$other_months[which(bank4$month=='oct')]=1


bank4$other_months[which(bank4$month=='sep')]=1
table(bank4$jun)
# not including day of week as it doesnt sound to have realistic approach on
subscription,
#duration in minute.
bank5=as.data.frame(bank4)
```

```
bank5$Duration_min=(bank5$duration/60)
#pdays: prev_c_contact or not
ind_999 = which(bank5$pdays==999)
bank5$previously_cntctd[ind_999] = 0
ind_less999 = which(bank5$pday<999)
bank5$previously_cntctd[ind_less999] = 1
# c_c_contact derived from previous variable: contacted previously for this campaign
bank5$c_c_contact=0
ind_0 = which(bank5$previous==0)
bank5$c_c_contact[ind_0]=0
ind_1=which(bank5$previous>0)
bank5$c_c_contact[ind_0]=1
table(bank5$c_c_contact)
bank6=bank5[,-c(2,3,6,7,8,9,11)]


#DEFAULT VARIABLE
bank6$No_default=0
bank6$No_default[which(bank6$default=='no')]=1
bank6=bank6[,-c(3)]
#previous outcome:poutcome....new variable last_outcome
bank6$last_outcome=0
ind_fail=which(bank6$poutcome=='failure')
bank6$last_outcome[ind_fail]=-1
ind_success=which(bank6$poutcome=='success')
```

```
bank6$last_outcome[ind_success]=1
ind_noexist=which(bank6$poutcome=='nonexistent')
bank6$last_outcome[ind_noexist]=0
table(bank6$last_outcome)
bank6=bank6[,-c(5)]



## EDUCATION VARIABLE
summary(bank6)
bank7=as.data.frame(bank6)
bank7$education=as.character(bank7$education)
table(bank7$education)
ind_basic = which(bank7$education %in% c('basic.4y', 'basic.6y', 'basic.9y'))
bank7$education[ind_basic] = 'basic_edu'
bank7$basic_edu=bank7$high_school=bank7$prof_course=bank7$university_degr
ee=0
bank7$basic_edu[which(bank7$education=='basic_edu')]=1
bank7$high_school[which(bank7$education=='high.school')]=1
bank7$university_degree[which(bank7$education=='university.degree')]=1
bank7$prof_course[which(bank7$education=='professional.course')]=1

final=bank7[,-c(2)]
#####################################################################
library(caret)
nearZeroVar(final)
```

```
###################LOGISTIC REGRESSION############



# NOVEMBER AND SERVICES ARE INSIGNIFICANT VARIABLES:
#model1 with 40766 observations and 15 independent variables
model1= subset(final,

select=c(y,Duration_min,nr.employed,may,previously_cntctd,cons.price.idx,last_o
utcome,

Blue_collar,No_default,cellphn_contact,campaign,other_months,jun,aug,jul,
                    previous))



#ACCURACY : FORWARD
n=nrow(model1)

set.seed(123)
id.train = sample(1:n, n*0.60) # ncol() gives number of columns
id.test = setdiff(1:n, id.train)
dat.train = model1[id.train,]
dat.test = model1[id.test,]
```

```
min.model1 = glm(y~ 1, data = dat.train, family = 'binomial')
max.model1 = glm(y ~ ., data = dat.train, family = 'binomial')
max.formula.model1 = formula(max.model1)
obj_model1 = step(min.model1, direction='forward', scope=max.formula.model1) # it will printout models in each step
summary(obj_model1)

#yhat_step = predict(obj_model1, newdata = dat.test, type='response')
predictions <- predict.glm(obj_model1, newdata=dat.test, type= "response")
predictions[predictions > 0.5] <- 1
predictions[predictions <= 0.5] <- 0
acc <- 1 - (length(predictions[predictions == dat.test$y]) / length(predictions))
acc
table(predictions, dat.test$y)
yhat = predict(obj_model1, newdata = dat.test, type='response')
# if type = response, we get the probability.
hist(yhat)

dichotomize = function(yhat, cutoff=.5) {
  out = rep(0, length(yhat))
  out[yhat > cutoff] = 1
  out
}

yhat.class = dichotomize(yhat, .1)
err = mean(yhat.class != dat.test$y) # misclassification error rate
```

```
err

sen = function(ytrue, yhat) {
  ind.true1 = which(ytrue == 1)
  mean( ytrue[ind.true1] == yhat[ind.true1] )
}

spe = function(ytrue, yhat) {
  ind.true0 = which(ytrue == 0)
  mean( ytrue[ind.true0] == yhat[ind.true0] )
}

sen(dat.test$y, yhat.class)
spe(dat.test$y, yhat.class)




######### ACCURACY: BACKWARD

n=nrow(model1)
set.seed(123)
id.train = sample(1:n, n*0.60) # ncol() gives number of columns
id.test = setdiff(1:n, id.train)
dat.train = model1[id.train,]
```

```
dat.test = model1[id.test,]
min.model1 = glm(y~ 1, data = dat.train, family = 'binomial')
max.model1 = glm(y ~ ., data = dat.train, family = 'binomial')
max.formula.model1 = formula(max.model1)


obj_model1_back = step(max.model1, direction='backward', scope=max.formula.model1) # it will print out models in each step
summary(obj_model1_back)
predictions_back <- predict.glm(obj_model1_back, newdata=dat.test, type= "response")
predictions_back[predictions_back > 0.5] <- 1
predictions_back[predictions_back <= 0.5] <- 0


1 - length(predictions_back[predictions_back == dat.test$y]) / length(predictions_back)


table(predictions_back, dat.test$y)


yhat = predict(obj_model1_back, newdata = dat.test, type='response')
# if type = response, we get the probability.
hist(yhat)


dichotomize = function(yhat, cutoff=.5) {
  out = rep(0, length(yhat))
  out[yhat > cutoff] = 1
```

```
  out
}

yhat.class = dichotomize(yhat, .1)
err = mean(yhat.class != dat.test$y) # misclassification error rate
err

sen = function(ytrue, yhat) {
  ind.true1 = which(ytrue == 1)
  mean( ytrue[ind.true1] == yhat[ind.true1] )
}

spe = function(ytrue, yhat) {
  ind.true0 = which(ytrue == 0)
  mean( ytrue[ind.true0] == yhat[ind.true0] )
}

sen(dat.test$y, yhat.class)
spe(dat.test$y, yhat.class)

#predictions 0 1
# 0 14086 1109
# 1 363 749
```

########## STEPWISE

```
n=nrow(model1)
set.seed(123)
id.train = sample(1:n, n*0.60) # ncol() gives number of columns
id.test = setdiff(1:n, id.train)
dat.train = model1[id.train,]
dat.test = model1[id.test,]
min.model1 = glm(y~ 1, data = dat.train, family = 'binomial')
max.model1 = glm(y ~ ., data = dat.train, family = 'binomial')
max.formula.model1 = formula(max.model1)

obj_model1_step = step(min.model1, scope=list(lower=min.model1,
                                              upper=max.formula.model1), direction='both')
summary(obj_model1_step)
yhat_step = predict(obj_model1_step, newdata = dat.test, type='response')
hist(yhat_step)
predictions_step <- predict.glm(obj_model1_step, newdata=dat.test, type= "response")
predictions_step[predictions_step > 0.5] <- 1
predictions_step[predictions_step <= 0.5] <- 0
1 - length(predictions_step[predictions_step == dat.test$y]) / length(predictions_step)
table(predictions_step, dat.test$y)
```

```
yhat = predict(obj_model1_step, newdata = dat.test, type='response')
# if type = response, we get the probability.
hist(yhat)

dichotomize = function(yhat, cutoff=.5) {
  out = rep(0, length(yhat))
  out[yhat > cutoff] = 1
  out
}

yhat.class = dichotomize(yhat, .1)
err = mean(yhat.class != dat.test$y) # misclassification error rate
err

sen = function(ytrue, yhat) {
  ind.true1 = which(ytrue == 1)
  mean( ytrue[ind.true1] == yhat[ind.true1] )
}

spe = function(ytrue, yhat) {
  ind.true0 = which(ytrue == 0)
  mean( ytrue[ind.true0] == yhat[ind.true0] )
}

sen(dat.test$y, yhat.class)
```

```
spe(dat.test$y, yhat.class)
```