Group 213: Bank Marketing Analysis

First Name	Last Name	Email (hawk.iit.edu)	Student ID
Nupur	Singh	nsingh26@hawk.iit.edu	A20431624
Vaishali	Pachisiya	vpachisiya@hawk.iit.edu	A20411068
Somendra	Chaudhary	schaudhary3@hawk.iit.edu	A20432563

Table of Contents

1. Introduction

Implementation of the marketing concept in business practices is not possible without the application of marketing management, which is regarded as the art and science of choosing target markets and the ability to attract, retain and increase the number of buyers by creating, delivering and communicating superior customer value.

This project is based on the direct marketing campaigns of a Portuguese banking institution. This marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to assess if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

In order to meet our goal, we developed several classification models with the balanced ability to give a correct prediction on both consumer sign-ups as opening a deposit account and no sign-ups. We found the best model based on accuracy which is suitable enough to help marketing personnel to focus on those promising leads and reduce the efforts on those who are unlikely to conduct business with them.

2. Data

The dataset we have used for our project to predict the term deposit subscription problem "Bank Marketing Data" from www.UCI.edu. In this dataset, the variable 'y' is the response variable which depicts the term deposit subscription. There are 20 attributes in the dataset. Here is a breakdown of all 20 variables in the dataset along with variable data type. The target variable is y which has two values: ‘yes’ (customer opens a bank account) and ‘no’ (customer does not open an account). The following table represents the columns in the data along with the datatypes and the representation of the data.

Website: http://archive.ics.uci.edu/ml/datasets/Bank+Marketing#

Columns in the datasets	Datatypes	Representation
age	numeric	age of clients
job	categorical	type of job
marital	categorical	marital status
education	categorical	marital status
default	categorical	has credit in default?
housing	categorical	has housing loan?
loan	categorical	has personal loan?
contact	categorical	contact communication type
month	categorical	last contact month of year
day_of_week	categorical	last contact day of the week
duration	numeric	last contact duration, in seconds
campaign	numeric	number of contacts performed during this campaign and for this client
pdays	numeric	number of days that passed by after the client was last contacted from a previous campaign
previous	numeric	number of contacts performed before this campaign and for this client
poutcome	categorical	outcome of the previous marketing campaign
emp.var.rate	numeric	employment variation rate - quarterly indicator
cons.price.idx	numeric	consumer price index - monthly indicator
cons.conf.idx	numeric	consumer confidence index - monthly indicator
euribor3m	numeric	euribor 3 month rate - daily indicator
nr.employed	numeric	number of employees - quarterly indicator
y	binary: 'yes', 'no'	the client subscribed a term deposit

3. Problems to be Solved

We need to implement multiple classification algorithms on this data to build a model that predicts the success of telemarketing calls for selling bank long term deposits based on a set of predictor variables.

Potential Implications: The model generated would help banks and financial institutions for the following domains:

1. Marketing Analytics: Identify which factors the bank needs to work on to market their products more effectively to their potential customers.

2. Behavioral Analytics: Identify worthy customers and estimate how likely a consumer would respond to a telephone call asking them to subscribe to a long-term deposit account.

4. Solutions

We will be making different models and test for their accuracy to decide which model provides the best solution to improve the revenue by opening deposit accounts in the bank. The various models include:

- Logistic Regression
- Naïve Bayes'
- K-Nearest Neighbor algorithm
- Decision Trees
- Random Forest

We aim to determine the model with the best accuracy to improve the performance of their telmarketing campaigns and identify potential customers.

5. Experiments and Results

5.1 Methods and Process

Model Selection:

- Lot of work has been done on the impact of Bank Marketing Analysis. We have come across a lot of research work which involves predictive modeling using supervised learning models like Logistic Regression Model (which include full, backward, forward), C5.0 Naïve Bayes Model, Combination of models. We have listed several journals and publications that helped us in model selection for our project:

  1. Myron L. Kwast, Martha Starr-McCluer and John D. Wolken, "Market Definition and the Analysis of Antitrust in Banking", The Antitrust Bulletin, 44 (1997), pp. 973-995, and Marianne P. Bitler, Alicia M. Robb and John D. Wolken. http://www.rpubs.com/johnakwei/330635

  2. Wikipedia contributors. (2018, November 26). Naive Bayes classifier. In *Wikipedia, The Free Encyclopedia*. Retrieved 08:20, December 1, 2018, from http://en.wikipedia.org/wiki/Naïve_Bayes_classifier

  3. The UCI Machine Learning Repository is a collection of databases. http://archive.ics.uci.edu/ml

  After doing the review and research about classification techniques and our dataset, we decided to work 5 models that is Logistic Regression (full, stepwise, backward, forward) Random forest, Decision trees, Naïve Bayes' and KNN. By applying different models, we can identify which model gives us the best results in terms of the relevance of the outcome.

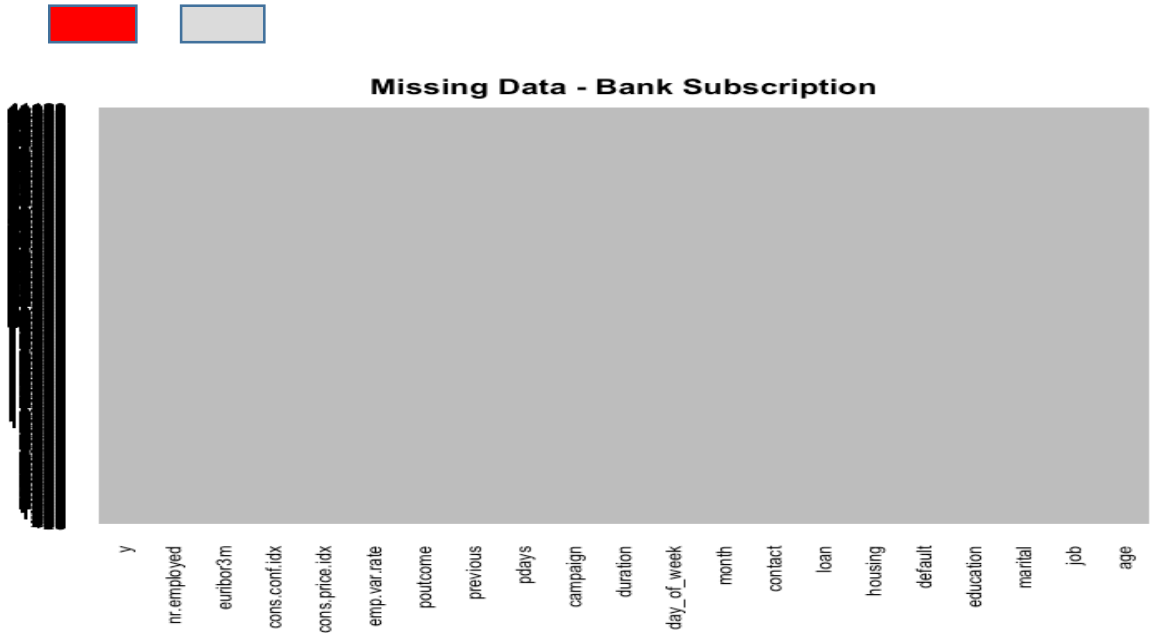
5.1. a) Data Processing

We analyzed the dataset, and identified that the dataset does not require any cleaning. The graph below explains the total missing values in the given dataset. The gray box represents all the observations and the red box in the graph represents the missing values in the variables. As there is no red lines in graph no missing values.

Missing Values Graph:

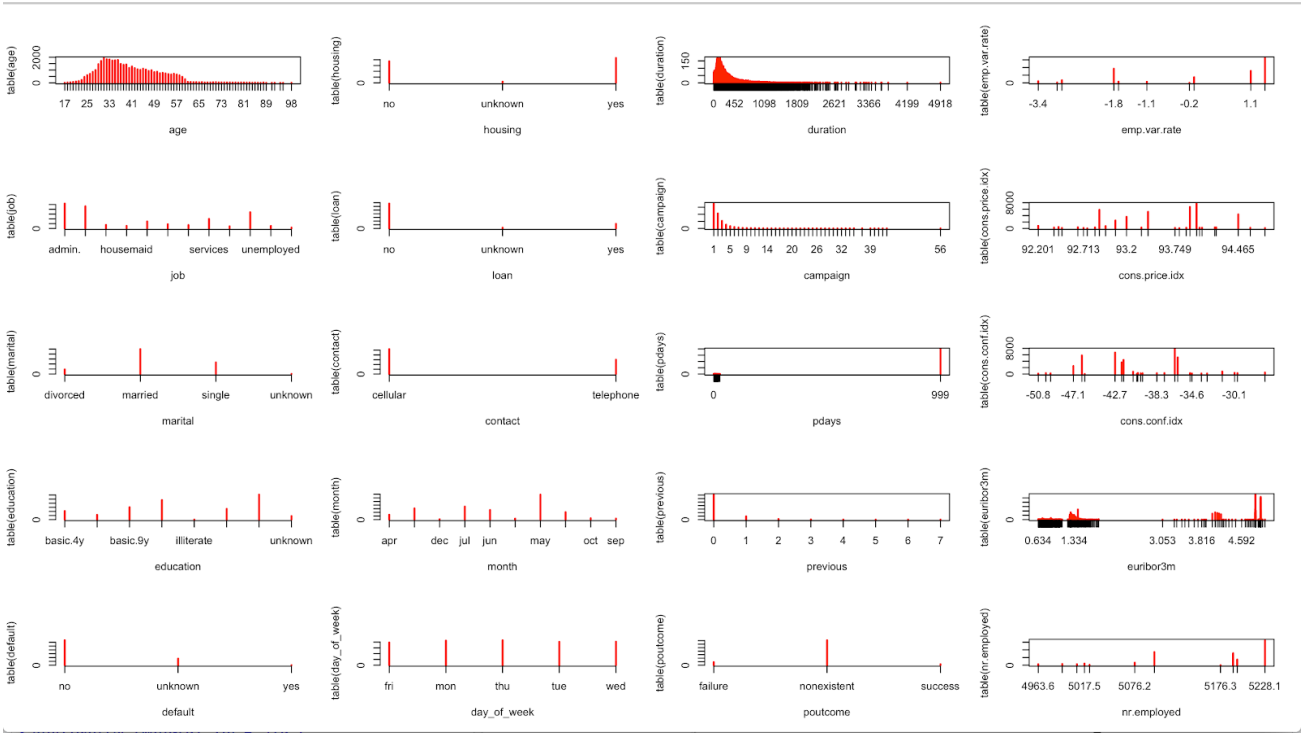
```
> missmap(alldata,main="Missing Data - Bank Subscription", col=c("re
d","grey"),legend=FALSE)
>
```

Missing Values No missing values



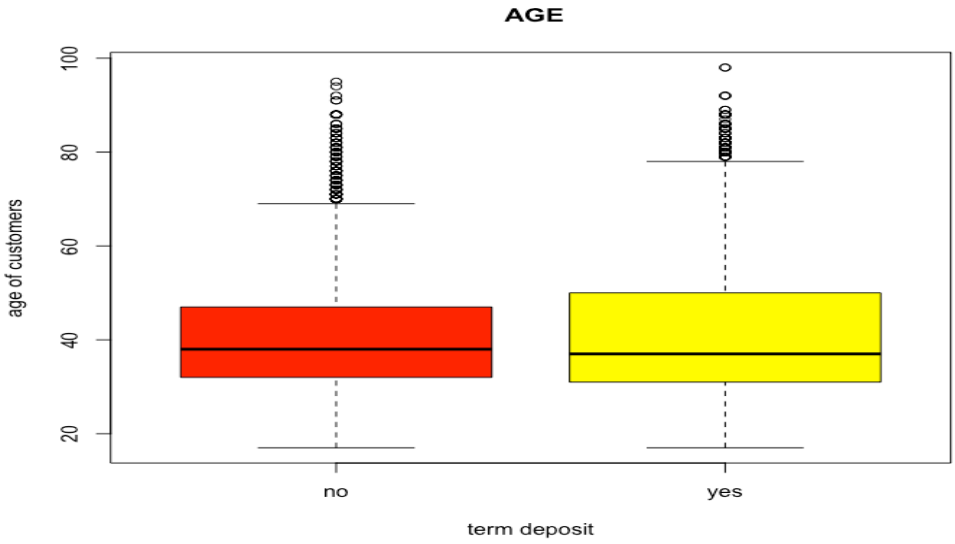
For a better understanding of the dataset, below we have explained the data analysis process and the appropriate steps taken to deal with the missing values and the outliers for each of the independent variables by analyzing them through R codes, histograms and boxplots.

Here, we have shown the spread of data in each of the x variables.



Box plots for numerical variables:

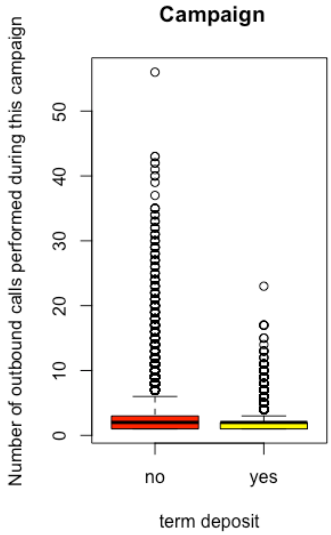
1. **Age**



From the age box plot we were not sure that what is the average age of people saying 'yes' to term deposit and who are saying 'no', so we performed hypothesis testing on this box plot.

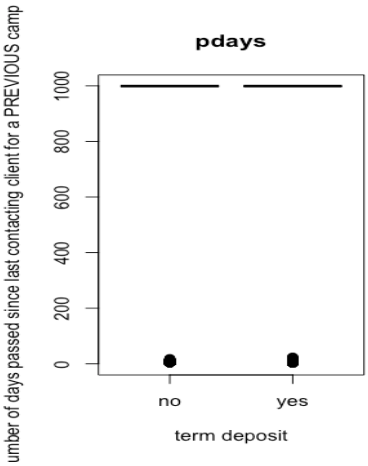
2. Campaign

This box plots gives us an idea of number of contacts performed during this campaign and for a particular client. Here, we can see very clearly that number of clients saying no are those who have been contacted more.



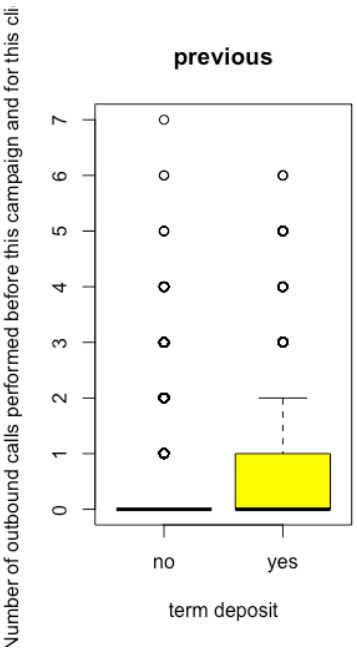
3. pdays

This box plot gives a spread of number of days that passed by after the client was last contacted from a previous campaign. Here, either the clients have been contacted recently or they have been contacted a long back. Hence the percentage of clients saying yes and no are almost equivalent here so we did not perform any transformation on the data.



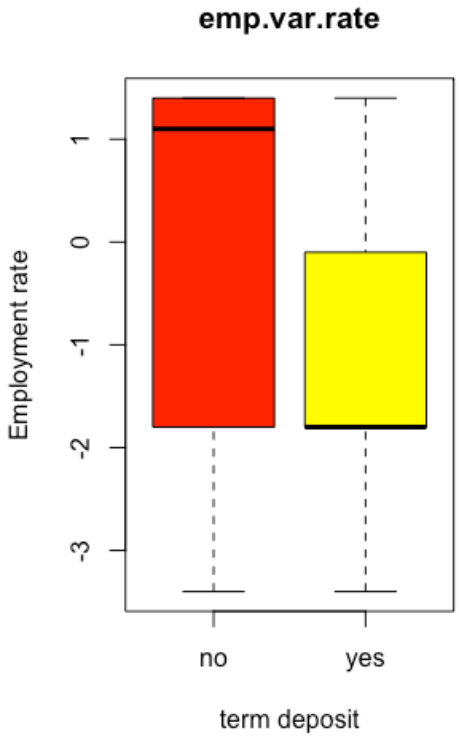
4. previous

From this box plot, we can see the number outbound calls performed before this campaign and for a particular client. We can see that the median value for both yes and no is almost equal. There are few outliers for both yes and no, but the data was quite normalized.



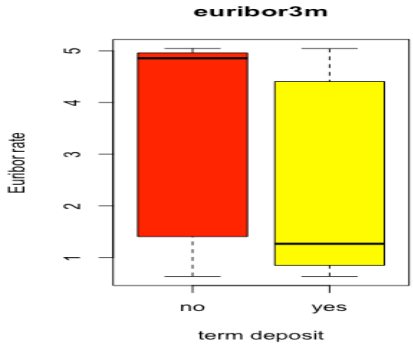
5. emp.var.rate

This box plot indicates the employment variation rate and it is measured quarterly. The spread of this was skewed a bit, so we made log transformation on this to make it normal.



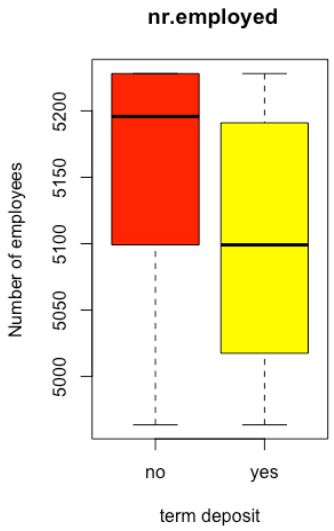
6. euribor3m

This box plot represents the Euribor rates which is based on the average interest rates at which a large panel of European banks borrow funds from one another. This box plot shows that the amount of people saying yes was more when the Euribor rates were low, but as the rates increased the majority of clients said no for the term deposit. The spread of this variable was skewed too.



7. nr.employed

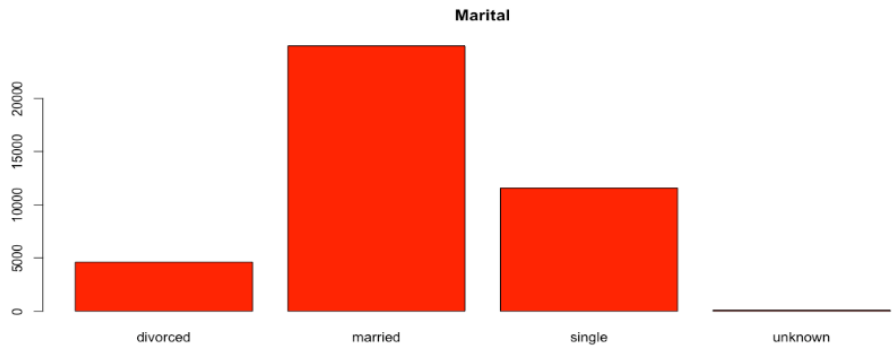
This represents the box plot for number of employees. We can see that the number of employees for clients saying yes is normally distributed. For the clients saying no, it is normally distributed too but with some extent of skewness. Hence we did not make any such changes in this variable.



Bar plots for categorical variables:

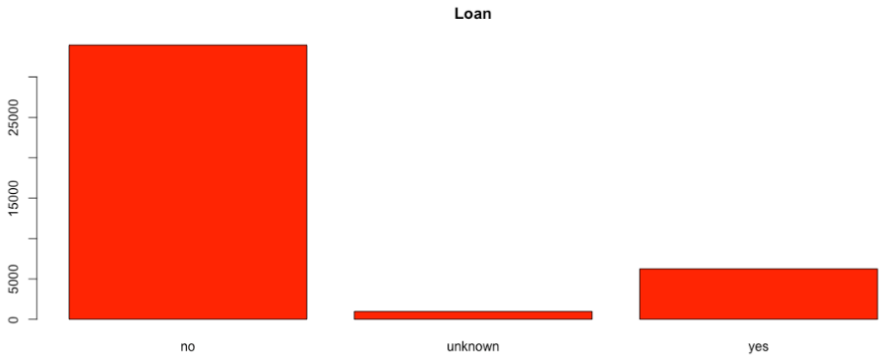
1. **Marital**

From the bar plot, we can see that there is some missing information here, which is “unknown” but the percentage of data not known is quite less, hence we did not replace it with the most used value in that variable.



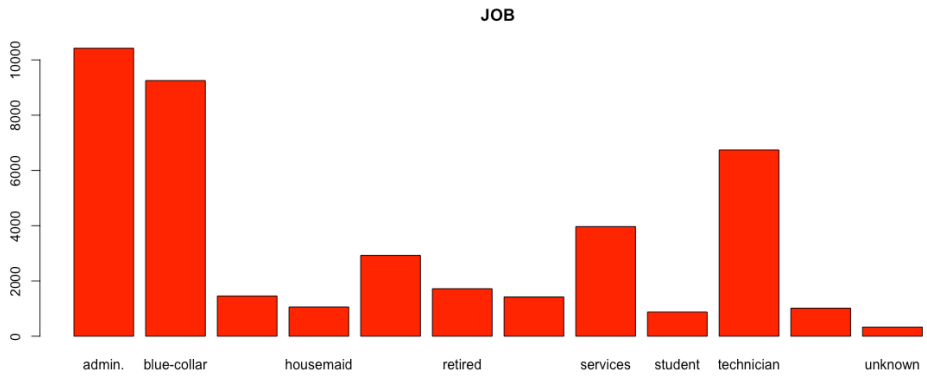
2. **Loan**

From the bar plot, we can see that there is some missing information here, which is “unknown” but the percentage of data not known is quite less, hence we did not replace it with the most used value in that variable. Maximum people in the data don’t have a loan, while some people do.



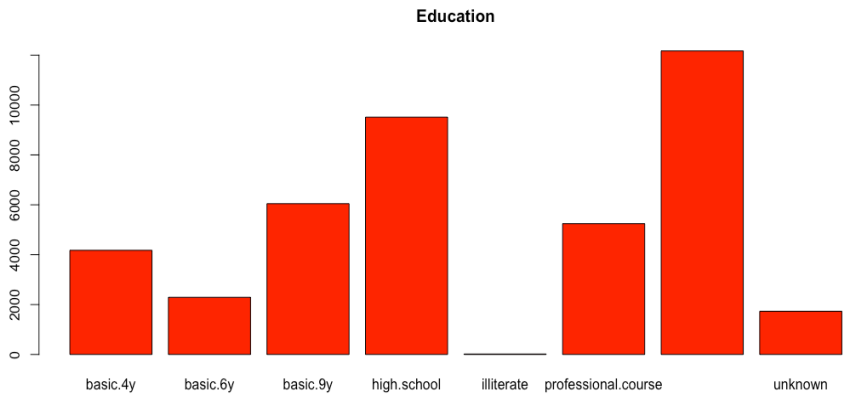
3. **Job:**

From the bar plot, we can see that there is some missing information here, which is "unknown" but the percentage of data not known is quite less, hence we did not replace it with the most used value in that variable. Most people that the calls are made to are administrators or have blue-collar jobs.

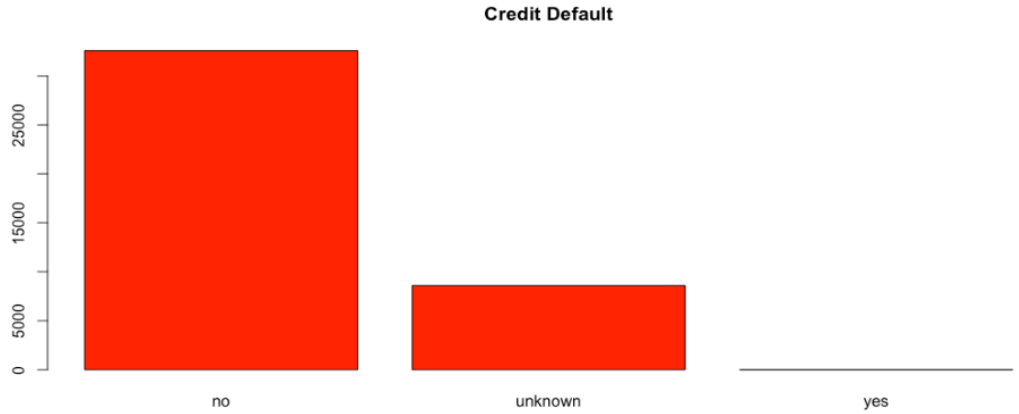


4. **Education:**

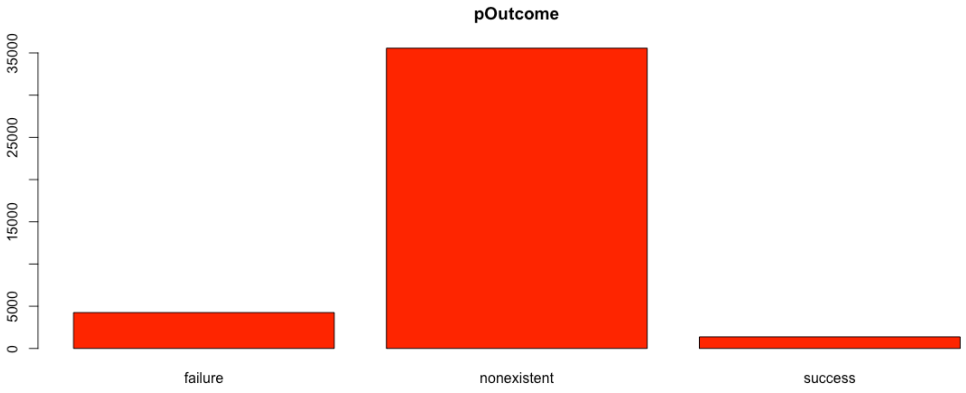
From the bar plot, we can see that there is some missing information here, which is "unknown" but the percentage of data not known is quite less, hence we did not replace it with the most used value in that variable. We didn't require any transformations here.



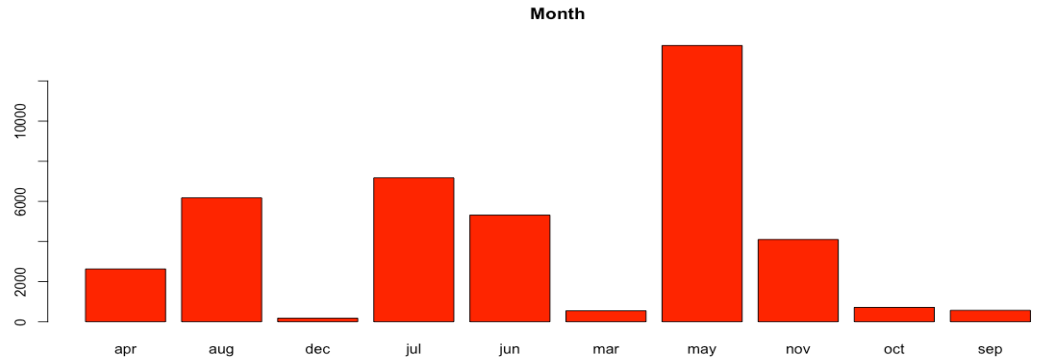
5. **Default:**
   From the bar plot, we can see that there is some missing information here, which is "unknown" but the percentage of data not known is quite less but more than the percentage of 'yes', we can replace this one with 'no' but there would be bias in this variable towards 'no' as data for 'yes' is very few.



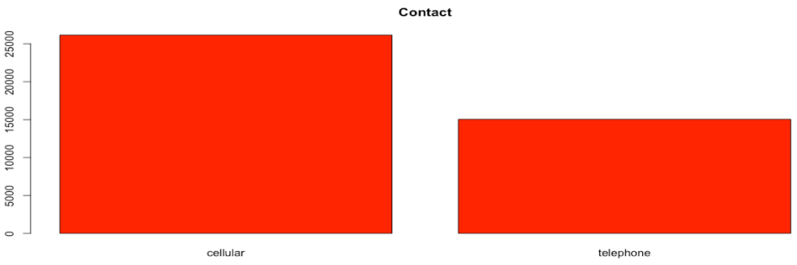
6. **pOutcome:**
   pOutcome recorded the success or failure of previous telmarketing campaigns to a customer. From the bar plot, we can see that there is no missing information here. We further observe that the calls were being made the first time to most of the customers. For the customers to whom the calls had been made previously, most of them had been failures, with few successes.



7. **Month:**
   From the bar plot, we can see that there is no missing information here. The other observations here are that most customers have been contacted in May, while very few, al most negligible attempt to reach customers was made in December.



8. **Contact:**
   From the bar plot, we can see that there is no missing information here. The data is quite balanced.



We removed duration column from our data, as the duration variable records the amount of time the telemarketer spends speaking with the customers, and therefore this isn't a factor in real time prediction of likelihood of obtaining a term deposit.

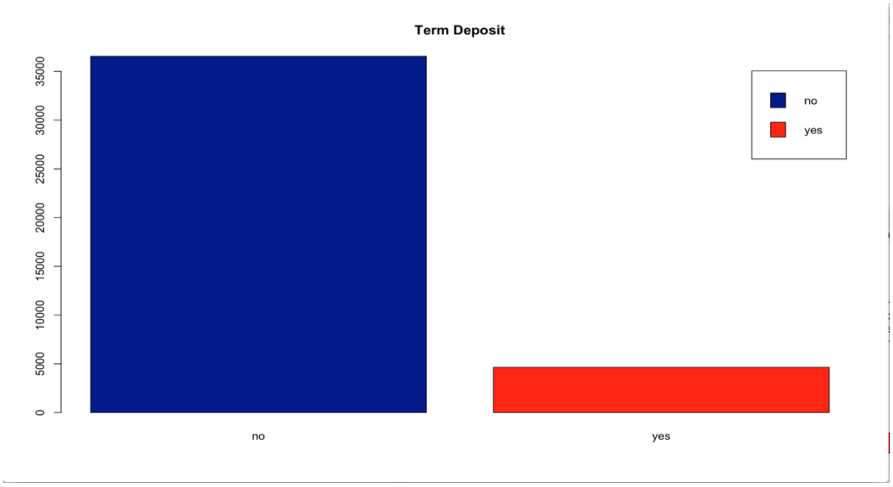
Correlation:

We also checked the correlation value between all the numerical variables here and the correlation values were fine for the numeric x variables,

```
> alldata.cont<-data.frame(alldata$age,alldata$campaign,alldata$pdays,alldata$previous,alldata$emp.var.rate,alldata$cons.price.idx,alldata$cons.conf.idx, alldata$euribor3m,
alldata$nr.employed)
Warning messages:
1: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
2: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
3: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
> cor(alldata.cont)
                      alldata.age alldata.campaign alldata.pdays alldata.previous alldata.emp.var.rate alldata.cons.price.idx
alldata.age           1.0000000000      0.00459358   -0.03436895       0.02436474       -0.0003706855           0.000856715
alldata.campaign      0.0045935805      1.00000000    0.05258357      -0.07914147        0.1507538056           0.127835912
alldata.pdays        -0.0343689512      0.05258357    1.00000000      -0.58751386        0.2710041743           0.078889109
alldata.previous      0.0243647409     -0.07914147   -0.58751386       1.00000000       -0.4204891094          -0.203129967
alldata.emp.var.rate -0.0003706855      0.15075381    0.27100417      -0.42048911        1.0000000000           0.775334171
alldata.cons.price.idx 0.0008567150     0.12783591    0.07888911      -0.20312997        0.7753341708           1.000000000
alldata.cons.conf.idx  0.1293716142    -0.01373310   -0.09134235      -0.05093635        0.1960412681           0.058986182
alldata.euribor3m     0.0107674295      0.13513251    0.29689911      -0.45449365        0.9722446712           0.688230107
alldata.nr.employed  -0.0177251319      0.14409489    0.37260474      -0.50133293        0.9069701013           0.522033977
                     alldata.cons.conf.idx alldata.euribor3m alldata.nr.employed
alldata.age                     0.12937161        0.01076743        -0.01772513
alldata.campaign               -0.01373310        0.13513251         0.14409489
alldata.pdays                  -0.09134235        0.29689911         0.37260474
alldata.previous               -0.05093635       -0.45449365        -0.50133293
alldata.emp.var.rate            0.19604127        0.97224467         0.90697010
alldata.cons.price.idx          0.05898618        0.68823011         0.52203398
alldata.cons.conf.idx           1.00000000        0.27768622         0.10051343
alldata.euribor3m               0.27768622        1.00000000         0.94515443
alldata.nr.employed             0.10051343        0.94515443         1.00000000
```

DATA BALANCE TESTING

```
> table(alldata$y)

   no   yes
36548  4640
```



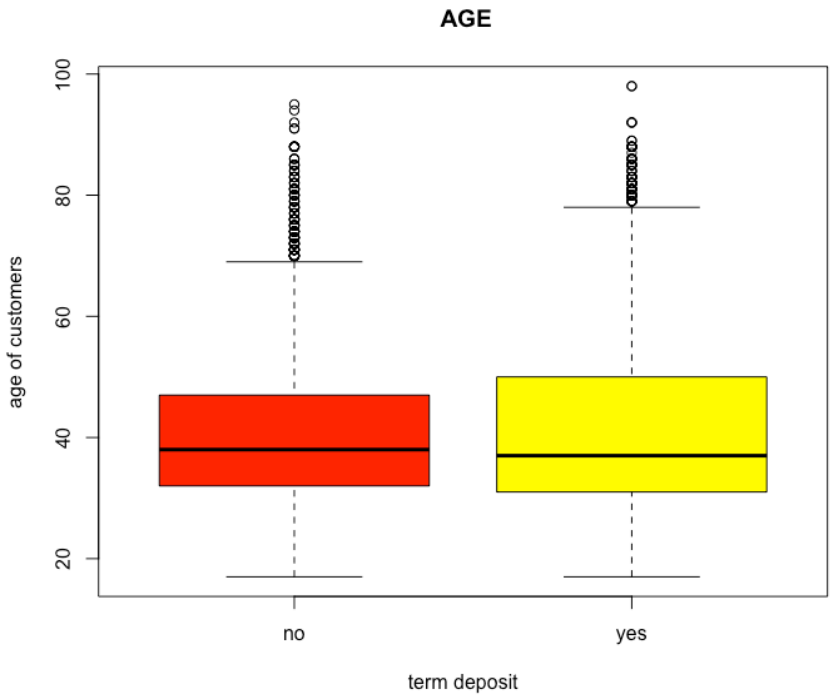
We checked the label and we saw that amount of people saying 'no' are much more than the people saying 'yes'. Hence our data is heavily biased towards 'no'. To deal with this situation we used up sampling, such that our label has equal amount of yes and no. We mainly performed this for two reasons:

1. **Our data was not very large, hence down sampling it would have reduced the data to a very small level which was not a feasible option.**
2. **Up sampling allows us to keep our main data intact, hence we were not losing any important information here.**

```
> alldata_n=upSample(x, y, list = FALSE, yname = "y")
> table(alldata_n$y)

   no   yes
36548 36548
>
```

Hypothesis Testing:

1. Two sample One-tailed Hypothesis Testing



```
> hypo1 <- z.test(age_y, age_n, alternative= "less", mu = 0, sigma.x=sd(age_y), sigma.y=sd(age_n), conf.level=0.95)
> hypo1

        Two-sample z-Test

data:  age_y and age_n
z = 4.7795, p-value = 1
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       NA 1.346781
sample estimates:
mean of x mean of y
 40.91315  39.91119
```

Null Hypothesis: Average age of people saying yes to term deposit is greater than people saying no

Alternative Hypothesis: Average age of people saying yes is less than the average age of people saying no to term deposit.

From the hypothesis test, we can see that the p value is greater than 0.05 for 95% confidence level. Hence we reject the null hypothesis.

5.1.b Classification Models

Model 1: Logistic Regression

Logistic regression is a predictive analysis method to analyze dataset and predict the dependent variable. Logistic regression model is used when the dependent variable or the target variable is binary. It explains the relationship between the dependent variable and one or more nominal, ordinal, interval or ratio level independent variables. Basically, there is no requirement for logistic model we can use any kind of data. But, here to build our model we convert all the categorical data into numerical dataset.

a) **Full model:**

Here class is our y variable, datasetu = alldata_n(i.e. our up sampled data)

```
> fullu = glm(class~., data = datasetu, family = binomial() )
```

```
> summary(fullu)

Call:
glm(formula = Class ~ ., family = binomial(), data = datasetu)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.87945  -0.85952   0.07715   0.80935   2.05917

Coefficients: (10 not defined because of singularities)
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -4.408e-01  1.532e+01  -0.029 0.977044
age                          -5.232e-03  1.122e-02  -0.466 0.641027
jobadmin.                    -4.209e-03  4.512e-02  -0.093 0.925674
`jobblue-collar`             -4.351e-02  4.350e-02  -1.000 0.317112
jobentrepreneur               6.224e-03  2.070e-02   0.301 0.763669
jobhousemaid                 -3.156e-02  1.858e-02  -1.699 0.089307 .
jobmanagement                -1.137e-02  2.762e-02  -0.412 0.680635
jobretired                    6.367e-02  2.215e-02   2.874 0.004047 **
`jobself-employed`            7.765e-03  2.084e-02   0.373 0.709430
jobservices                   6.531e-04  3.198e-02   0.020 0.983709
jobstudent                    4.551e-02  1.699e-02   2.679 0.007380 **
jobtechnician                 2.602e-02  3.959e-02   0.657 0.511063
jobunemployed                -1.829e-03  1.821e-02  -0.100 0.919993
jobunknown                           NA         NA      NA       NA
maritaldivorced              -1.677e-01  5.837e-02  -2.874 0.004058 **
maritalmarried               -2.452e-01  8.969e-02  -2.734 0.006249 **
maritalsingle                -1.806e-01  8.267e-02  -2.185 0.028909 *
maritalunknown                       NA         NA      NA       NA
educationbasic.4y            -2.063e-02  1.639e-02  -1.259 0.207993
educationbasic.6y             2.981e-02  1.361e-02   2.190 0.028521 *
educationbasic.9y             2.591e-03  1.806e-02   0.143 0.885924
educationhigh.school          2.504e-03  2.036e-02   0.123 0.902123
educationilliterate           7.838e-03  8.250e-03   0.950 0.342100
educationprofessional.course  9.178e-03  1.744e-02   0.526 0.598744
educationuniversity.degree    4.328e-02  2.200e-02   1.967 0.049172 *
educationunknown                     NA         NA      NA       NA
defaultno                    -1.950e+11  6.220e+11  -0.313 0.753922
defaultunknown               -1.949e+11  6.219e+11  -0.313 0.753922
defaultyes                   -4.094e+09  1.306e+10  -0.313 0.753922
housingno                     1.534e-02  8.891e-03   1.725 0.084489 .
housingunknown               -6.048e-03  9.496e-03  -0.637 0.524202
housingyes                           NA         NA      NA       NA
loanno                        7.513e-03  9.331e-03   0.805 0.420701
loanunknown                          NA         NA      NA       NA
loanyes                              NA         NA      NA       NA
contactcellular               3.074e-01  1.644e-02  18.705  < 2e-16 ***
contacttelephone                     NA         NA      NA       NA
monthapr                     -7.342e-02  2.256e-02  -3.254 0.001137 **
monthaug                      1.451e-01  2.940e-02   4.935 8.00e-07 ***
monthdec                      2.411e-02  9.251e-03   2.606 0.009160 **
monthjul                     -3.268e-02  3.429e-02  -0.953 0.340623
monthjun                     -3.019e-01  4.093e-02  -7.376 1.63e-13 ***
monthmar                      1.397e-01  1.106e-02  12.622  < 2e-16 ***
monthmay                     -3.227e-01  3.798e-02  -8.497  < 2e-16 ***
monthnov                     -1.986e-01  2.538e-02  -7.826 5.03e-15 ***
monthoct                     -2.983e-03  1.224e-02  -0.244 0.807489
monthsep                             NA         NA      NA       NA
day_of_weekfri               -4.217e-02  1.103e-02  -3.824 0.000131 ***
day_of_weekmon               -1.119e-01  1.122e-02  -9.968  < 2e-16 ***
day_of_weekthu               -3.303e-02  1.103e-02  -2.994 0.002755 **
day_of_weektue               -5.405e-02  1.104e-02  -4.894 9.90e-07 ***
day_of_weekwed                       NA         NA      NA       NA
campaign                     -1.034e-01  1.106e-02  -9.354  < 2e-16 ***
pdays                        -2.318e-01  2.739e-02  -8.464  < 2e-16 ***
previous                     -6.290e-02  2.022e-02  -3.111 0.001865 **
poutcomefailure              -2.024e-01  4.438e-02  -4.562 5.07e-06 ***
poutcomenonexistent          -1.063e-01  4.959e-02  -2.143 0.032137 *
poutcomesuccess                      NA         NA      NA       NA
```

```
emp.var.rate                -2.468e+00  1.099e-01 -22.462  < 2e-16 ***
cons.price.idx               1.152e+00  6.954e-02  16.571  < 2e-16 ***
cons.conf.idx                8.165e-02  1.980e-02   4.124 3.73e-05 ***
euribor3m                    6.676e-01  1.075e-01   6.207 5.38e-10 ***
nr.employed                  2.969e-01  1.040e-01   2.856 0.004291 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 101333  on 73095  degrees of freedom
Residual deviance:  78139  on 73043  degrees of freedom
AIC: 78245

Number of Fisher Scoring iterations: 25
```

b) Stepwise Forward Model:

Stepwise method adds the most significant variable and removes the least significant variables for each step to identify the useful subsets of variables. It checks the p-value with alpha and according to it removes/adds the variable to the model fit.

Building a base model with one x-variable and building a stepwise forward model:

Here class is our y variable, datasetu = alldata_n(i.e. our up sampled data)

```
> #Building a base model with one x-variable and building a step-wise forward model using it: upsampli
ng
> baseu = glm(class~age, datasetu, family = binomial())
> forwardu = step(baseu, scope = list(upper = fullu, lower = ~1), direction = 'forward', trace = F)
> summary(forwardu)

Call:
glm(formula = class ~ age + nr.employed + monthmay + pdays +
    poutcomefailure + monthmar + contactcellular + monthnov +
    campaign + defaultno + day_of_weekmon + jobretired + emp.var.rate +
    euribor3m + cons.price.idx + educationuniversity.degree +
    monthjun + monthaug + monthapr + jobstudent + cons.conf.idx +
    day_of_weekwed + educationbasic.4y + maritalunknown + maritalsingle +
    previous + educationbasic.9y + jobhousemaid + monthdec +
    educationilliterate + housingno + day_of_weektue + maritaldivorced +
    poutcomenonexistent + jobadmin. + jobentrepreneur + educationbasic.6y +
    `jobblue-collar` + housingunknown + jobservices, family = binomial(),
    data = datasetu)
```

```
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.87644  -0.86009  -0.05521   0.81072   2.07771

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                -0.438914   0.009141 -48.016  < 2e-16 ***
age                        -0.005015   0.011040  -0.454 0.649656
nr.employed                 0.299134   0.083971   3.562 0.000368 ***
monthmay                   -0.289625   0.015286 -18.947  < 2e-16 ***
pdays                      -0.228261   0.027452  -8.315  < 2e-16 ***
poutcomefailure            -0.206170   0.044477  -4.635 3.56e-06 ***
monthmar                    0.152997   0.008711  17.565  < 2e-16 ***
contactcellular             0.303586   0.015946  19.039  < 2e-16 ***
monthnov                   -0.198289   0.012146 -16.325  < 2e-16 ***
campaign                   -0.088899   0.010873  -8.176 2.94e-16 ***
defaultno                   0.071389   0.010186   7.008 2.41e-12 ***
day_of_weekmon             -0.080131   0.009770  -8.201 2.37e-16 ***
jobretired                  0.071682   0.009861   7.269 3.62e-13 ***
emp.var.rate               -2.440700   0.098078 -24.885  < 2e-16 ***
euribor3m                   0.643467   0.095719   6.722 1.79e-11 ***
cons.price.idx              1.146279   0.059725  19.193  < 2e-16 ***
educationuniversity.degree  0.041085   0.010147   4.049 5.14e-05 ***
monthjun                   -0.280262   0.019998 -14.014  < 2e-16 ***
monthaug                    0.177570   0.017881   9.931  < 2e-16 ***
monthapr                   -0.053222   0.010350  -5.142 2.71e-07 ***
jobstudent                  0.032371   0.008498   3.809 0.000139 ***
cons.conf.idx               0.084095   0.018924   4.444 8.84e-06 ***
day_of_weekwed              0.028868   0.009503   3.038 0.002384 **
educationbasic.4y          -0.026555   0.011129  -2.386 0.017026 *
maritalunknown              0.026685   0.007718   3.457 0.000546 ***
maritalsingle               0.029027   0.009983   2.908 0.003642 **
previous                   -0.057250   0.020207  -2.833 0.004609 **
educationbasic.9y          -0.013308   0.010868  -1.225 0.220761
jobhousemaid               -0.025197   0.009867  -2.554 0.010660 *
monthdec                    0.017927   0.008113   2.210 0.027134 *
educationilliterate         0.016203   0.007370   2.199 0.027906 *
housingno                   0.017908   0.008887   2.015 0.043895 *
day_of_weektue             -0.020986   0.009671  -2.170 0.029999 *
maritaldivorced            -0.019623   0.009271  -2.117 0.034290 *
poutcomenonexistent        -0.101359   0.049515  -2.047 0.040654 *
jobadmin.                   0.013545   0.010197   1.328 0.184071
jobentrepreneur             0.013636   0.009147   1.491 0.136040
educationbasic.6y           0.024767   0.009866   2.510 0.012065 *
`jobblue-collar`           -0.027343   0.012841  -2.129 0.033222 *
housingunknown             -0.015256   0.008996  -1.696 0.089917 .
jobservices                -0.015674   0.010161  -1.543 0.122912
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 101333  on 73095  degrees of freedom
Residual deviance:  78146  on 73055  degrees of freedom
AIC: 78228

Number of Fisher Scoring iterations: 5
```

c) Stepwise Backward Model:

Building a stepwise backward model:

Here class is our y variable, datasetu = alldata_n(i.e. our up sampled data)

```
> backwardu = step(fullu, direction = 'backward', trace = F)
There were 50 or more warnings (use warnings() to see the first 50)
> summary(backwardu)

Call:
glm(formula = Class ~ `jobblue-collar` + jobhousemaid + jobretired +
    `jobself-employed` + jobservices + jobstudent + maritaldivorced +
    maritalmarried + maritalsingle + educationbasic.4y + educationbasic.6y +
    educationilliterate + educationuniversity.degree + defaultunknown +
    housingno + housingunknown + contactcellular + monthapr +
    monthaug + monthdec + monthjun + monthmar + monthmay + monthnov +
    day_of_weekfri + day_of_weekmon + day_of_weekthu + day_of_weektue +
    campaign + pdays + previous + poutcomefailure + poutcomenonexistent +
    emp.var.rate + cons.price.idx + cons.conf.idx + euribor3m +
    nr.employed, family = binomial(), data = datasetu)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.87619  -0.86006  -0.05503   0.80913   2.07987

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -0.438855   0.009141 -48.012  < 2e-16 ***
`jobblue-collar`         -0.041662   0.011147  -3.738 0.000186 ***
jobhousemaid             -0.029602   0.009697  -3.053 0.002268 **
jobretired                0.064923   0.008758   7.413 1.23e-13 ***
`jobself-employed`       -0.013070   0.008870  -1.473 0.140618
jobservices              -0.021147   0.009689  -2.183 0.029066 *
jobstudent                0.029952   0.008209   3.649 0.000264 ***
maritaldivorced          -0.209271   0.055790  -3.751 0.000176 ***
maritalmarried           -0.293765   0.085659  -3.429 0.000605 ***
maritalsingle            -0.239197   0.078937  -3.030 0.002444 **
educationbasic.4y        -0.022169   0.010385  -2.135 0.032787 *
educationbasic.6y         0.028597   0.009405   3.041 0.002362 **
educationilliterate       0.016901   0.007372   2.292 0.021879 *
educationuniversity.degree 0.046353  0.009824   4.718 2.38e-06 ***
defaultunknown           -0.071765   0.010098  -7.107 1.19e-12 ***
housingno                 0.017924   0.008886   2.017 0.043681 *
housingunknown           -0.015445   0.008994  -1.717 0.085921 .
contactcellular           0.302913   0.015943  19.000  < 2e-16 ***
monthapr                 -0.053941   0.010348  -5.213 1.86e-07 ***
monthaug                  0.177288   0.017860   9.927  < 2e-16 ***
monthdec                  0.017709   0.008113   2.183 0.029049 *
monthjun                 -0.281430   0.019975 -14.089  < 2e-16 ***
monthmar                  0.153092   0.008707  17.582  < 2e-16 ***
monthmay                 -0.289583   0.015288 -18.942  < 2e-16 ***
monthnov                 -0.198476   0.012126 -16.368  < 2e-16 ***
day_of_weekfri           -0.035107   0.011021  -3.185 0.001446 **
day_of_weekmon           -0.109617   0.011254  -9.741  < 2e-16 ***
day_of_weekthu           -0.023597   0.011025  -2.140 0.032337 *
day_of_weektue           -0.050132   0.011060  -4.533 5.82e-06 ***
campaign                 -0.088609   0.010867  -8.154 3.53e-16 ***
pdays                    -0.228440   0.027447  -8.323  < 2e-16 ***
previous                 -0.057455   0.020210  -2.843 0.004471 **
poutcomefailure          -0.204981   0.044465  -4.610 4.03e-06 ***
poutcomenonexistent      -0.100058   0.049497  -2.022 0.043227 *
emp.var.rate             -2.446752   0.098013 -24.964  < 2e-16 ***
cons.price.idx            1.149875   0.059665  19.272  < 2e-16 ***
cons.conf.idx             0.084068   0.018884   4.452 8.52e-06 ***
euribor3m                 0.641371   0.095667   6.704 2.03e-11 ***
nr.employed               0.304499   0.083894   3.630 0.000284 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 101333  on 73095  degrees of freedom
Residual deviance:  78148  on 73057  degrees of freedom
AIC: 78226

Number of Fisher Scoring iterations: 5
```

Model	AIC
Full	78245
Stepwise forward	78228
Stepwise backward	78226

We find that the Stepwise backward model has the lowest AIC value, but it is close to Stepwise forward, so we can't determine which model is best based on the AIC values alone. So, we perform cross-validation.

Performing cross-validation:

```
> cv.glm(fullu, data = datasetu, K=10)$delta
[1] 0.1787567 0.1787413
There were 29 warnings (use warnings() to see them)
> cv.glm(forwardu, data = datasetu, K=10)$delta
[1] 0.1787156 0.1787043
> cv.glm(backwardu, data = datasetu, K=10)$delta
[1] 0.1786781 0.1786692
```

Based on cross-validation, we calculate accuracy as follows:

Model	Accuracy
Full	82.12433
Stepwise forward	82.12844
Stepwise backward	82.13219

Model 2: Naïve Bayes'

The next model is Naïve Bayes'. This model is based on the Baynesian theorem and it is more applicable when the input dimentionality is high. It is used in various fields, as it is considered easy to implement and provides accurate results. Even though we speak about its simplicity, often Naïve Bayes' can outperform certain sophisticated classification methods. Basic requirement for NB is that all the variables should be categorial. For our Naïve Bayes' model, we transformed our dataset such that all the x-variables are categorical. We first examined the summary of all of the numerical x-variables and made appropriate groupings of the data in each of the x variable.

```
> summary(age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  17.00   32.00   38.00   40.02   47.00   98.00
> summary(campaign)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   2.000   2.568   3.000  56.000
> summary(pdays)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0   999.0   999.0   962.5   999.0   999.0
> summary(previous)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   0.173   0.000   7.000
> summary(emp.var.rate)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-3.40000 -1.80000  1.10000  0.08189  1.40000  1.40000
> summary(cons.conf.idx)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -50.8   -42.7   -41.8   -40.5   -36.4   -26.9
> summary(cons.price.idx)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  92.20   93.08   93.75   93.58   93.99   94.77
> summary(euribor3m)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.634   1.344   4.857   3.621   4.961   5.045
> summary(nr.employed)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4964    5099    5191    5167    5228    5228
>
```

```
> data_nb = alldata_n
> data_nb$age <- cut(data_nb$age, breaks = c(-Inf,32,38,47,Inf), labels= c("young", "middle-aged", "old", "very-old"), right = FALSE)
> data_nb$campaign <- cut(data_nb$campaign, breaks = c(-Inf,2.568,Inf), labels= c("c1", "c2"), right = FALSE)
> data_nb$pdays <- cut(data_nb$pdays, breaks = c(-Inf,962.5,Inf), labels= c("p1", "p2"), right = FALSE)
> data_nb$previous <- cut(data_nb$previous, breaks = c(-Inf,0.173,Inf), labels= c("prev1", "prev2"), right = FALSE)
> data_nb$emp.var.rate <- cut(data_nb$emp.var.rate, breaks = c(-Inf,0.08189,Inf), labels= c("emp1", "emp2"), right = FALSE)
> data_nb$cons.conf.idx <- cut(data_nb$cons.conf.idx, breaks = c(-Inf,-40.5,Inf), labels= c("conf1", "conf2"), right = FALSE)
> data_nb$cons.price.idx <- cut(data_nb$cons.price.idx, breaks = c(-Inf,93.58,Inf), labels= c("cons_price1", "cons_price2"), right = FALSE
> data_nb$euribor3m <- cut(data_nb$euribor3m, breaks = c(-Inf,3.621,Inf), labels= c("low", "high"), right = FALSE)
> data_nb$nr.employed <- cut(data_nb$nr.employed, breaks = c(-Inf,5167, Inf), labels= c("n1", "n2"), right = FALSE)
```

From here we can see that all of our data is categorical now.

```
> str(data_nb)
'data.frame':   73096 obs. of  20 variables:
 $ age           : Factor w/ 4 levels "young","middle-aged",..: 4 4 2 3 4 3 4 3 1 1 ...
 $ job           : Factor w/ 12 levels "admin.","blue-collar",..: 4 8 8 1 8 8 1 2 10 8 ...
 $ marital       : Factor w/ 4 levels "divorced","married",..: 2 2 2 2 2 2 2 2 3 3 ...
 $ education     : Factor w/ 8 levels "basic.4y","basic.6y",..: 1 4 4 2 4 3 6 8 6 4 ...
 $ default       : Factor w/ 3 levels "no","unknown",..: 1 2 1 1 1 2 1 2 1 1 ...
 $ housing       : Factor w/ 3 levels "no","unknown",..: 1 1 3 1 1 1 1 1 3 3 ...
 $ loan          : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 3 1 1 1 1 1 ...
 $ contact       : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
 $ month         : Factor w/ 10 levels "apr","aug","dec",..: 7 7 7 7 7 7 7 7 7 7 ...
 $ day_of_week   : Factor w/ 5 levels "fri","mon","thu",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ campaign      : Factor w/ 2 levels "c1","c2": 1 1 1 1 1 1 1 1 1 1 ...
 $ pdays         : Factor w/ 2 levels "p1","p2": 2 2 2 2 2 2 2 2 2 2 ...
 $ previous      : Factor w/ 2 levels "prev1","prev2": 1 1 1 1 1 1 1 1 1 1 ...
 $ poutcome      : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ emp.var.rate  : Factor w/ 2 levels "emp1","emp2": 2 2 2 2 2 2 2 2 2 2 ...
 $ cons.price.idx: Factor w/ 2 levels "cons_price1",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ cons.conf.idx : Factor w/ 2 levels "conf1","conf2": 2 2 2 2 2 2 2 2 2 2 ...
 $ euribor3m     : Factor w/ 2 levels "low","high": 2 2 2 2 2 2 2 2 2 2 ...
 $ nr.employed   : Factor w/ 2 levels "n1","n2": 2 2 2 2 2 2 2 2 2 2 ...
 $ y             : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

We built the Naïve Bayes' model here and performed 10-fold cross validation on this to test our model.

```
> x=data_nb[1:19]
> str(x)
'data.frame':   73096 obs. of  19 variables:
 $ age           : Factor w/ 4 levels "young","middle-aged",..: 4 4 2 3 4 3 4 3 1 1 ...
 $ job           : Factor w/ 12 levels "admin.","blue-collar",..: 4 8 8 1 8 8 1 2 10 8 ...
 $ marital       : Factor w/ 4 levels "divorced","married",..: 2 2 2 2 2 2 2 2 3 3 ...
 $ education     : Factor w/ 8 levels "basic.4y","basic.6y",..: 1 4 4 2 4 3 6 8 6 4 ...
 $ default       : Factor w/ 3 levels "no","unknown",..: 1 2 1 1 1 2 1 2 1 1 ...
 $ housing       : Factor w/ 3 levels "no","unknown",..: 1 1 3 1 1 1 1 1 3 3 ...
 $ loan          : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 3 1 1 1 1 1 ...
 $ contact       : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...
 $ month         : Factor w/ 10 levels "apr","aug","dec",..: 7 7 7 7 7 7 7 7 7 7 ...
 $ day_of_week   : Factor w/ 5 levels "fri","mon","thu",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ campaign      : Factor w/ 2 levels "c1","c2": 1 1 1 1 1 1 1 1 1 1 ...
 $ pdays         : Factor w/ 2 levels "p1","p2": 2 2 2 2 2 2 2 2 2 2 ...
 $ previous      : Factor w/ 2 levels "prev1","prev2": 1 1 1 1 1 1 1 1 1 1 ...
 $ poutcome      : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ emp.var.rate  : Factor w/ 2 levels "emp1","emp2": 2 2 2 2 2 2 2 2 2 2 ...
 $ cons.price.idx: Factor w/ 2 levels "cons_price1",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ cons.conf.idx : Factor w/ 2 levels "conf1","conf2": 2 2 2 2 2 2 2 2 2 2 ...
 $ euribor3m     : Factor w/ 2 levels "low","high": 2 2 2 2 2 2 2 2 2 2 ...
 $ nr.employed   : Factor w/ 2 levels "n1","n2": 2 2 2 2 2 2 2 2 2 2 ...
> y=data_nb$y
> str(y)
 Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

```
> nbmodel = train(x,y,'nb',trControl=trainControl(method='cv',number=10),na.action=na.pass)
There were 50 or more warnings (use warnings() to see the first 50)
> print(nbmodel)
Naive Bayes

73096 samples
   19 predictor
    2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 65786, 65786, 65786, 65788, 65786, 65786, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.7160718  0.4321435
   TRUE      0.7160718  0.4321435

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was
 held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust = 1.
>
```

From our model we can see that the accuracy we are getting here is 71.60%

Model 3: K- Nearest Neighbor

KNN algorithm is used for classification and regression predictive problems but it is mostly used for classification problems. In KNN object is classified based on its majority of neighbors where the object is assigned to the class as most common among its k-nearest neighbors. K being 1 state that the object is assigned to the class of the single nearest neighbor. In KNN k value must be selected to predict the accuracy. The two parameters we need to access on different k-values, small positive integers, are training error rate and validation error rate. Basic requirement for this model is we need to convert all categorial variables into numerical variable and then we normalize all the variables and build model using that dataset. After all process, we finally got best accuracy at k=1(95.17%).

```
> data_knn=dummy.data.frame(alldata_n,names = c("job","marital", "education","default",
+                                               "housing","loan","contact","month","day_of_week","poutcome"))
> head(data_knn)
  age jobadmin. jobblue-collar jobentrepreneur jobhousemaid jobmanagement jobretired jobself-employed jobservices
jobstudent jobtechnician
1  56         0              0               0            1             0          0                0           0
         0              0
2  57         0              0               0            0             0          0                0           1
         0              0
3  37         0              0               0            0             0          0                0           1
         0              0
4  40         1              0               0            0             0          0                0           0
         0              0
5  56         0              0               0            0             0          0                0           1
         0              0
6  45         0              0               0            0             0          0                0           1
         0              0
  jobunemployed jobunknown maritaldivorced maritalmarried maritalsingle maritalunknown educationbasic.4y education
basic.6y
1             0          0               0              1             0              0                1
         0
2             0          0               0              1             0              0                0
         0
3             0          0               0              1             0              0                0
         0
4             0          0               0              1             0              0                0
         1
5             0          0               0              1             0              0                0
         0
6             0          0               0              1             0              0                0
         0
  educationbasic.9y educationhigh.school educationilliterate educationprofessional.course educationuniversity.degr
ee educationunknown
1                 0                    0                   0                           0
 0                 0
2                 0                    1                   0                           0
 0                 0
3                 0                    1                   0                           0
 0                 0
4                 0                    0                   0                           0
```

```
> library(dummies)
dummies-1.5.6 provided by Decision Patterns

> data_knn=alldata_n
> data_knn=dummy.data.frame(data_knn,names=c("job","marital","education","default","housing","loan","contact", "month", "day_of_week","poutcome")
> num.vars <- sapply(data_knn,is.numeric)
> data_knn[num.vars] <- lapply(data_knn[num.vars],scale)
> head(data_knn)
          age  jobadmin. jobblue-collar jobentrepreneur jobhousemaid jobmanagement jobretired jobself-employed jobservices jobstudent
1  1.29648896 -0.6099976     -0.4793564      -0.1793503    6.2845731    -0.2745019 -0.2610638       -0.1863422  -0.3032788 -0.1969686
2  1.37945351 -0.6099976     -0.4793564      -0.1793503   -0.1591176    -0.2745019 -0.2610638       -0.1863422   3.2972514 -0.1969686
3 -0.27983735 -0.6099976     -0.4793564      -0.1793503   -0.1591176    -0.2745019 -0.2610638       -0.1863422   3.2972514 -0.1969686
4 -0.03094372  1.6393282     -0.4793564      -0.1793503   -0.1591176    -0.2745019 -0.2610638       -0.1863422  -0.3032788 -0.1969686
5  1.29648896 -0.6099976     -0.4793564      -0.1793503   -0.1591176    -0.2745019 -0.2610638       -0.1863422   3.2972514 -0.1969686
6  0.38387899 -0.6099976     -0.4793564      -0.1793503   -0.1591176    -0.2745019 -0.2610638       -0.1863422   3.2972514 -0.1969686
  jobtechnician jobunemployed  jobunknown maritaldivorced maritalmarried maritalsingle maritalunknown educationbasic.4y
1    -0.4391539    -0.1681094 -0.08935449      -0.3473554      0.8527189    -0.6721246    -0.04654239         3.0362482
2    -0.4391539    -0.1681094 -0.08935449      -0.3473554      0.8527189    -0.6721246    -0.04654239        -0.3293493
3    -0.4391539    -0.1681094 -0.08935449      -0.3473554      0.8527189    -0.6721246    -0.04654239        -0.3293493
4    -0.4391539    -0.1681094 -0.08935449      -0.3473554      0.8527189    -0.6721246    -0.04654239        -0.3293493
5    -0.4391539    -0.1681094 -0.08935449      -0.3473554      0.8527189    -0.6721246    -0.04654239        -0.3293493
6    -0.4391539    -0.1681094 -0.08935449      -0.3473554      0.8527189    -0.6721246    -0.04654239        -0.3293493
```

We built the K-Nearest Neighbor model here and performed 10-fold cross validation on this to test our model. We checked our model for K=1 till K=20. We found our best accuracy at K=1, which was **95.136%**

```
> x=data_knn[c(1:62)]
> y=data_knn$y
```

```
> knnmodel = train(x,y,'knn',trControl=trainControl(method='cv',number=10),tuneGrid=expand.grid(k = 1:20))
> print(knnmodel)
k-Nearest Neighbors

73096 samples
   62 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 65786, 65788, 65786, 65787, 65786, 65786, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9513653  0.9027306
   2  0.9156315  0.8312630
   3  0.8856026  0.7712051
   4  0.8591032  0.7182063
   5  0.8375562  0.6751122
   6  0.8182802  0.6365603
   7  0.8039429  0.6078858
   8  0.7923418  0.5846835
   9  0.7838460  0.5676920
  10  0.7763217  0.5526433
  11  0.7716565  0.5433129
  12  0.7691666  0.5383331
  13  0.7687972  0.5375944
  14  0.7674702  0.5349403
  15  0.7668955  0.5337912
  16  0.7659516  0.5319033
  17  0.7668683  0.5337366
  18  0.7647751  0.5295502
  19  0.7636671  0.5273341
  20  0.7612183  0.5224367

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 1.
>
```

5.2. Evaluations and Results

ROC curves are useful even if your predicted probabilities are not properly calculated. This is a fundamental tool for diagnostic test evaluation. AUC provides the area under curve and is a probability that a classifier will rank a randomly chosen positive observation higher than a randomly chosen negative observation. All AUC metric cares about is how well the classifier separates the two classes. It is very useful for datasets with highly unbalanced classes.

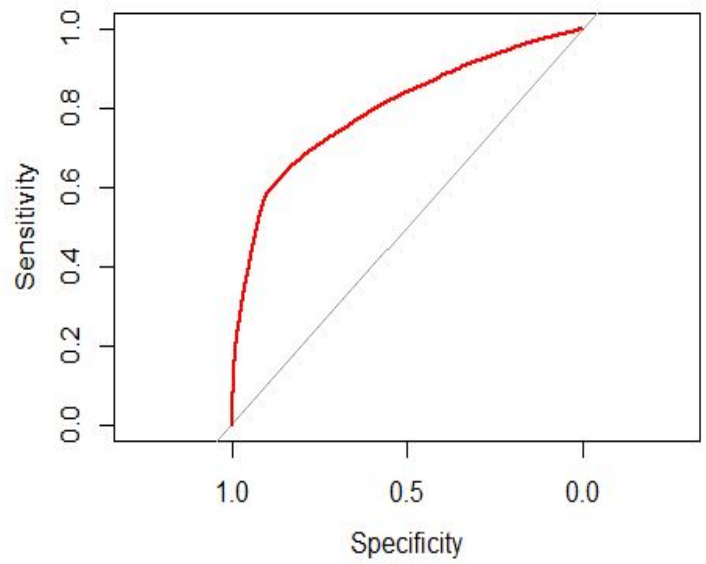
Model 1. Logistic Regression

a) Full Model

```
> pred_fullu = predict(fullu, datasetu[,-63])
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = ifelse(type ==  :
  prediction from a rank-deficient fit may be misleading
```

```
> head(lapply(pred_fullu, as.numeric))
$`1`
[1] -1.610399

$`2`
[1] -1.608782

$`3`
[1] -1.444105

$`4`
[1] -1.212664

$`5`
[1] -1.393244

$`6`
[1] -1.637224
```

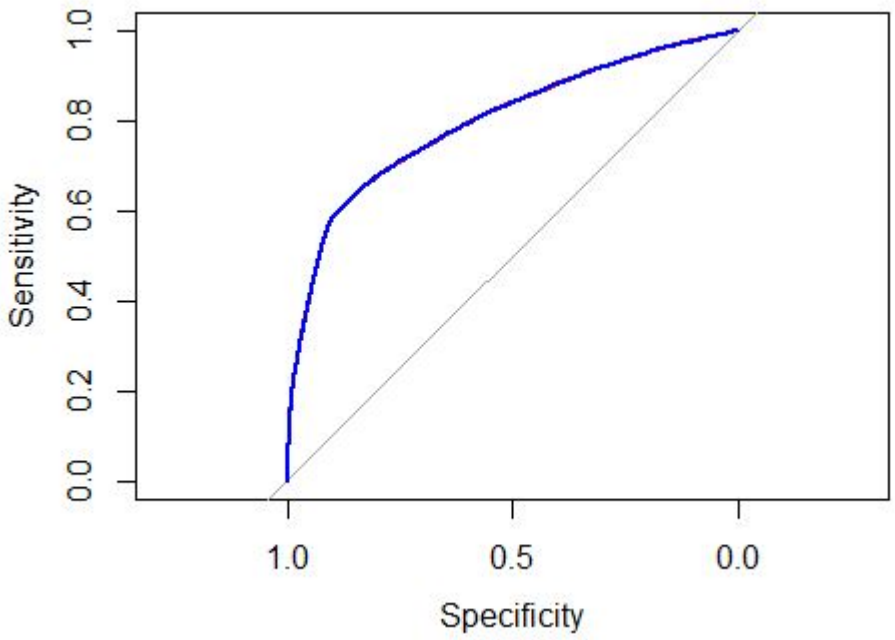
```
> roc_fullu <- roc(Class ~ as.numeric(pred_fullu), data = datasetu)
> plot(roc_fullu, col = "red")
> auc(roc_fullu)
Area under the curve: 0.7956
```



b) Logistic Stepwise Forward Model

```
> pred_forwardu = predict(forwardu, datasetu[,-63])
> head(lapply(pred_forwardu, as.numeric))
$`1`
[1] -1.613461

$`2`
[1] -1.595602

$`3`
[1] -1.446321

$`4`
[1] -1.219471

$`5`
[1] -1.419483

$`6`
[1] -1.627434
```

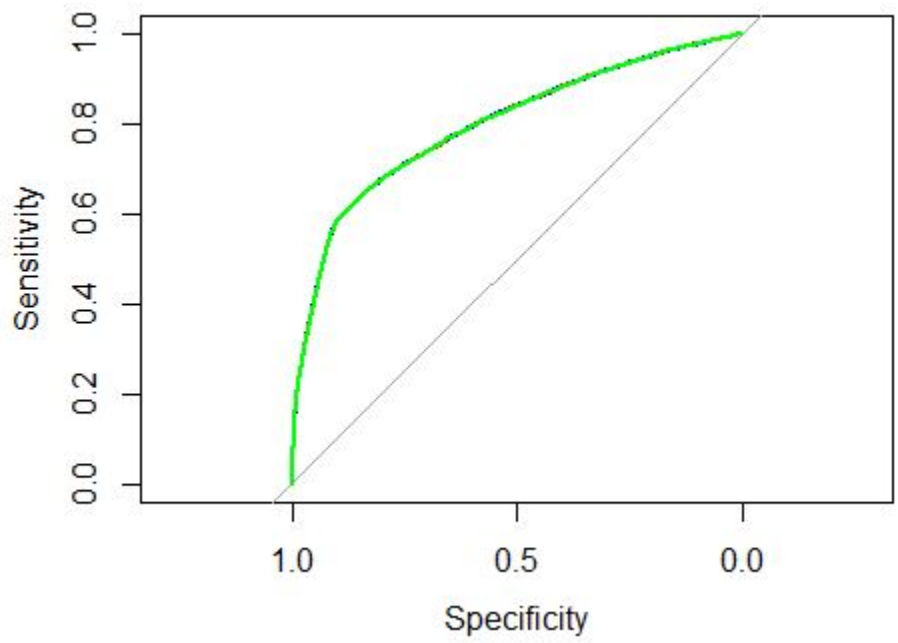
```
> roc_forwardu <- roc(Class ~ as.numeric(pred_forwardu), data = datasetu)
> lines(roc_forwardu, col = "blue")
> auc(roc_forwardu)
Area under the curve: 0.7958
```



c) Stepwise Backward Model

```
> pred_backwardu = predict(backwardu, datasetu[,-63])
> head(lapply(pred_backwardu, as.numeric))
$`1`
[1] -1.605483

$`2`
[1] -1.593344

$`3`
[1] -1.45277

$`4`
[1] -1.220352

$`5`
[1] -1.416757

$`6`
[1] -1.593344
```

```
> roc_backwardu <- roc(Class ~ as.numeric(pred_backwardu), data = datasetu)
> lines(roc_backwardu, col = "green")
> auc(roc_backwardu)
Area under the curve: 0.7957
```



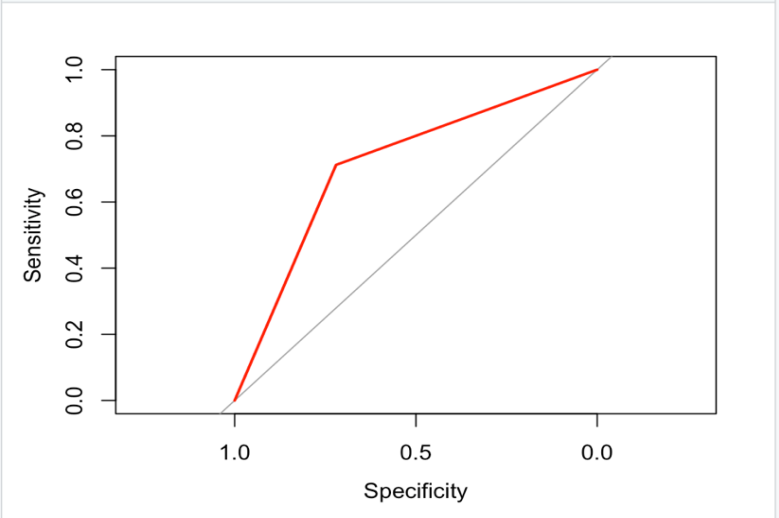
Model 2. Naïve Bayes'

Confusion Matrix:

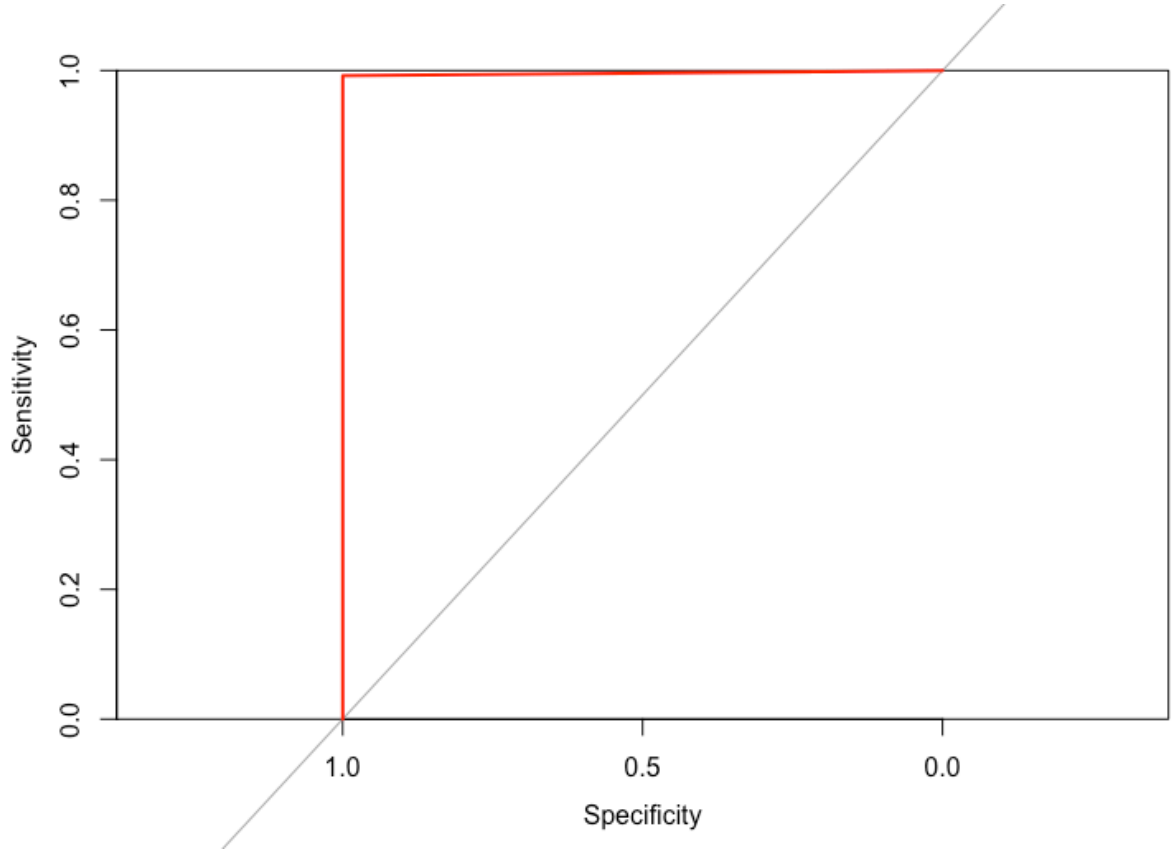
```
> cm_nb <- predict(nbmodel,x)
There were 50 or more warnings (use warnings() to see the first 50)
> tab1 <- table(cm_nb, data_nb$y)
> confusionMatrix(tab1)
Confusion Matrix and Statistics


cm_nb    no   yes
  no  26318 10519
  yes 10230 26029

               Accuracy : 0.7161
                 95% CI : (0.7129, 0.7194)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.4323
 Mcnemar's Test P-Value : 0.04557

            Sensitivity : 0.7201
            Specificity : 0.7122
         Pos Pred Value : 0.7144
         Neg Pred Value : 0.7179
             Prevalence : 0.5000
         Detection Rate : 0.3600
   Detection Prevalence : 0.5040
      Balanced Accuracy : 0.7161

       'Positive' Class : no
```

ROC and AUC:

```
> roc_nb <- roc(y ~ as.numeric(cm_nb), data = data_nb)
> plot(roc_nb, col = "red")
> auc(roc_nb)
Area under the curve: 0.7161
>
```



Model 3. K-Nearest Neighbor




```
roc_knn <- roc(y ~ as.numeric(cm_knn), data = data_knn)

plot(roc_knn, col = "red")
```

5.3. Findings

- The very first step we took was to analyze our data. We found that our data is quite good as there were no missing values but had some missing information in our categorical variable. This missing information were less and were already present in our data as "unknown". So, it was not required to replace it with any common value in that variable.

- We realized that our data is biased in terms of its class though. The data had 36548 information of 'no' and 4640 information of 'yes'. Hence, we performed up sampling to remove this biasing, such that our data increases and labels were equal in number.

- We performed three classification models, Logistic Regression, K-Nearest Neighbor and Naïve Bayes' and performed 10-fold cross-validation on them. We then compared them based on their accuracy.

- We also performed ROC curves and AUC to evaluate our model.

6. Conclusions and Future Work

6.1. Conclusions

Model	Accuracy
KNN	**95.14%**
Naive Bayes	80.47%
Logistic Full	82.12304%
Logistic Stepwise Forward	82.12915%
Logistic Stepwise Backward	82.12983%

- Analytics algorithms can help us in correctly classifying whether someone will open a deposit account with the bank or not.
- For this purpose, the best model/ algorithm is KNN as it can correctly classify 95.14% of records.

- Hence, bank telemarketers can now determine if a potential customer will open a deposit account or not with 95.14% accuracy, helping them to target customers more accurately, saving time and man hours.

6.2. Limitations

Our dataset was highly biased, hence the models we will build on that data will predict only 'no' as it will learn more from the records which have 'no'. To overcome this issue, we tried down sampling first but the data was reduced a lot and we also lost a lot of good information from our data. So finally, we had to up sample the dataset which made our dataset large but our information was left intact.

6.3. Potential Improvements or Future Work

- Another model that we want to work on is neural networks as it is a powerful computational model that is widely used for classification.
- We can improve our random forest model for different ntree and mtry values.

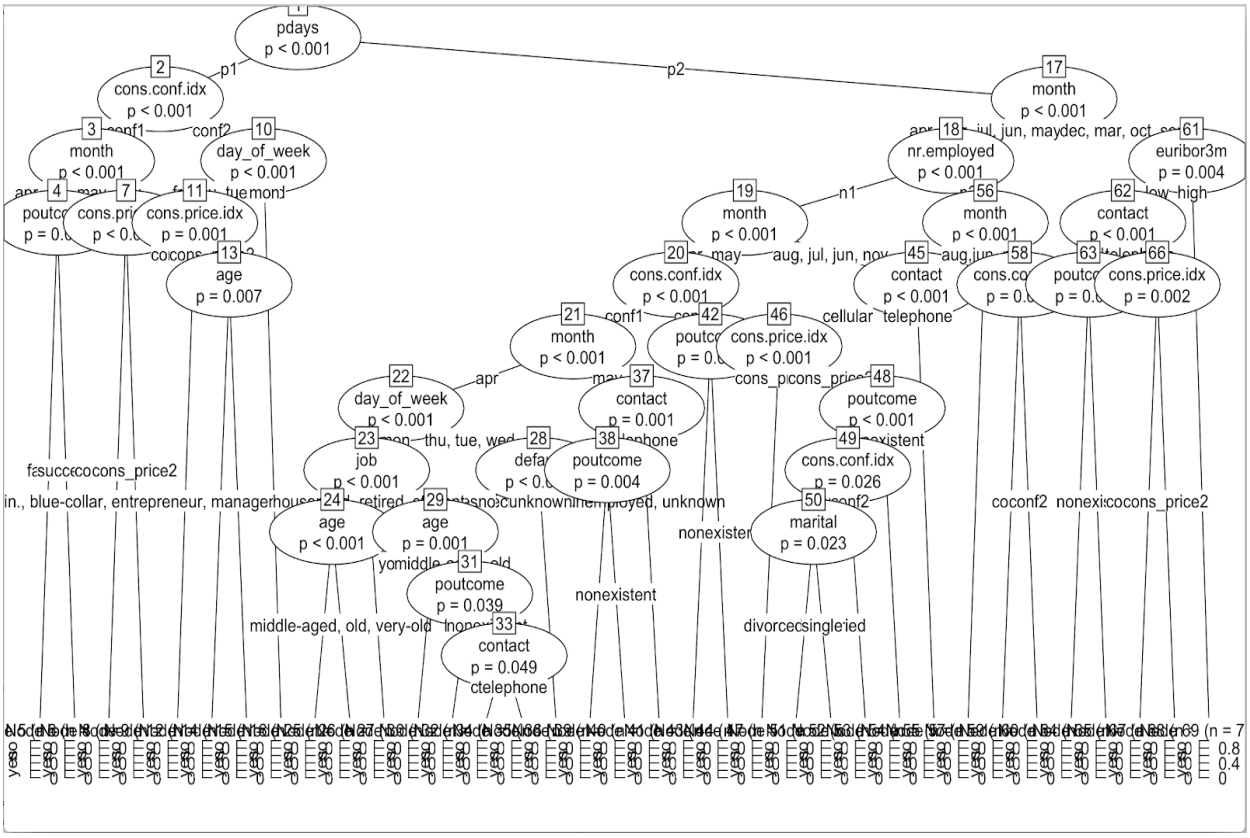
Decision Tree:

For decision tree we used ctree(). This function tests the null hypothesis of independence between any of the input variables and the response. It stops if this hypothesis cannot be rejected otherwise it selects the input variable with strongest association with the response. This association is measured by the p-value (which we can see below in the decision tree plot) corresponding to the test for the partial null hypothesis.

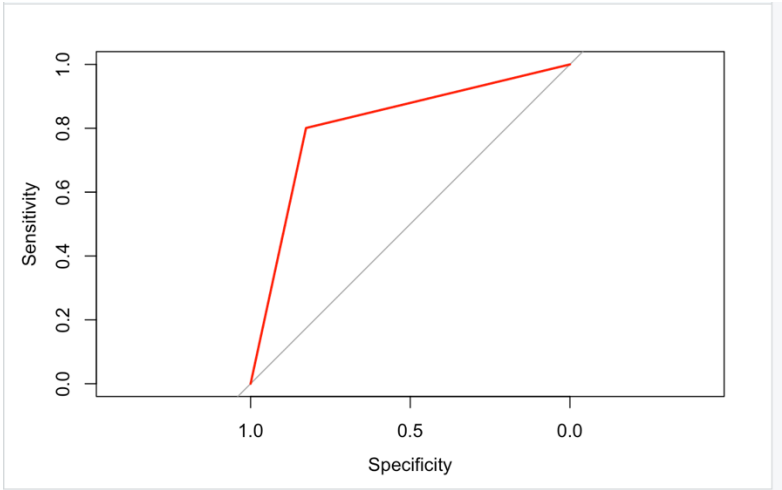
```
> tree_model <- ctree(y ~.,data = data_nb)
> ctrl_rf <- trainControl(method = "cv", number = 10)
> predict_tree <- predict(tree_model, data=data_nb, trControl=ctrl_rf)
> table(predict_tree)
predict_tree
   no   yes
39413  1775
```

```
> confusionMatrix(table(predict_tree, data_nb$y))
Confusion Matrix and Statistics


predict_tree    no   yes
         no  30196  7280
         yes  6352 29268

               Accuracy : 0.8135
                 95% CI : (0.8107, 0.8163)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.627
 Mcnemar's Test P-Value : 2.028e-15

            Sensitivity : 0.8262
            Specificity : 0.8008
         Pos Pred Value : 0.8057
         Neg Pred Value : 0.8217
             Prevalence : 0.5000
         Detection Rate : 0.4131
   Detection Prevalence : 0.5127
      Balanced Accuracy : 0.8135

       'Positive' Class : no
```

```
> print(tree_model)

Model formula:
y ~ age + job + marital + education + default + housing + loan +
    contact + month + day_of_week + campaign + pdays + previous +
    poutcome + emp.var.rate + cons.price.idx + cons.conf.idx +
    euribor3m + nr.employed

Fitted party:
[1] root
|   [2] pdays in p1
|   |   [3] cons.conf.idx in conf1
|   |   |   [4] month in apr, jun, mar, oct
|   |   |   |   [5] poutcome in failure: no (n = 20, err = 30.0%)
|   |   |   |   [6] poutcome in success: yes (n = 205, err = 36.6%)
|   |   |   [7] month in may, nov
|   |   |   |   [8] cons.price.idx in cons_price1: no (n = 199, err = 28.1%)
|   |   |   |   [9] cons.price.idx in cons_price2: yes (n = 44, err = 36.4%)
|   |   [10] cons.conf.idx in conf2
|   |   |   [11] day_of_week in fri, thu, tue, wed
|   |   |   |   [12] cons.price.idx in cons_price1: yes (n = 427, err = 31.1%)
|   |   |   |   [13] cons.price.idx in cons_price2
|   |   |   |   |   [14] age in young, middle-aged, old: yes (n = 291, err = 24.7%)
|   |   |   |   |   [15] age in very-old: yes (n = 137, err = 8.8%)
|   |   |   [16] day_of_week in mon: yes (n = 192, err = 43.2%)
|   [17] pdays in p2
|   |   [18] month in apr, aug, jul, jun, may, nov
|   |   |   [19] nr.employed in n1
|   |   |   |   [20] month in apr, may
|   |   |   |   |   [21] cons.conf.idx in conf1
|   |   |   |   |   |   [22] month in apr
|   |   |   |   |   |   |   [23] day_of_week in fri, mon
|   |   |   |   |   |   |   |   [24] job in admin., blue-collar, entrepreneur, management, self-employed, services
, technician, unemployed, unknown
|   |   |   |   |   |   |   |   |   [25] age in young: no (n = 162, err = 17.3%)
|   |   |   |   |   |   |   |   |   [26] age in middle-aged, old, very-old: no (n = 984, err = 5.7%)
|   |   |   |   |   |   |   |   |   [27] job in housemaid, retired, student: no (n = 83, err = 27.7%)
|   |   |   |   |   |   |   [28] day_of_week in thu, tue, wed
|   |   |   |   |   |   |   |   [29] default in no
|   |   |   |   |   |   |   |   |   [30] age in young, very-old: no (n = 566, err = 33.4%)
```




```
> roc_tree <- roc(y ~ as.numeric(predict_tree), data = data_nb)
> plot(roc_tree, col='red')
> auc(roc_tree)
Area under the curve: 0.8135
>
```



The decision tree gives an **accuracy of 81.35%.**

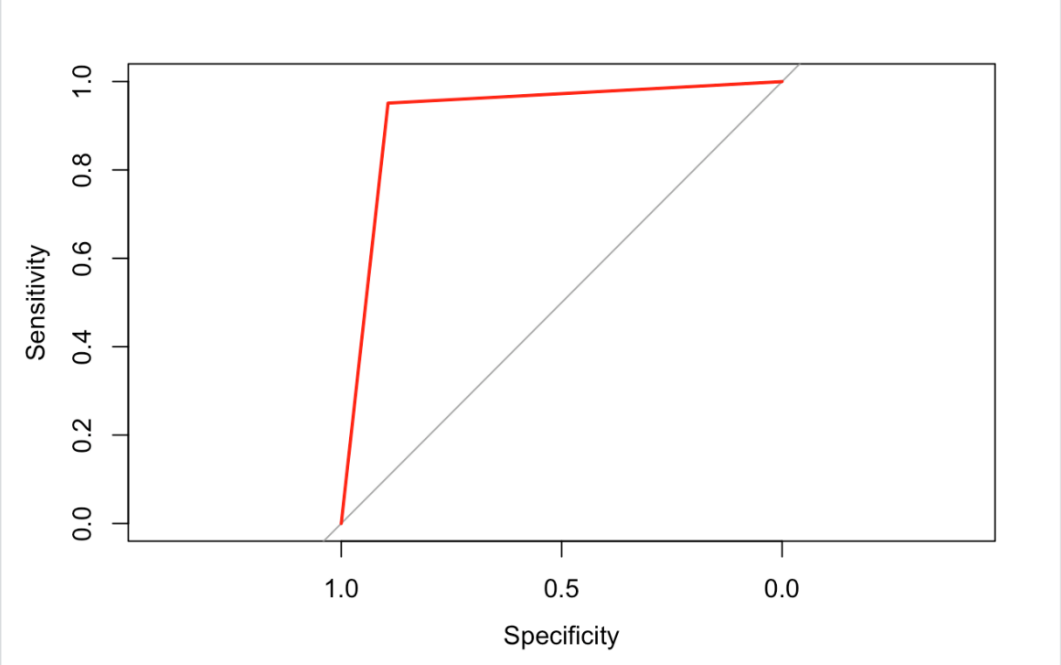
Random Forest:

For random forest we used randomForest(). Each decision tree in the forest considers a random subset of features when forming a question. This increases the diversity leading to a more robust overall accuracy.

```
> bank_rf = data_nb
> ctrl_rf <- trainControl(method = "cv", number = 10)
> model_rf2 <- randomForest(y~., data = data_nb, do.trace = TRUE, importance = TRUE, ntree = 500, mtry = 6, forest = TRUE, trControl = ctrl_rf)
ntree      OOB      1      2
    1:  14.55% 18.32% 10.79%
    2:  14.40% 17.78% 11.02%
    3:  14.06% 17.26% 10.88%
    4:  13.83% 16.89% 10.77%
    5:  13.56% 15.97% 11.15%
    6:  13.19% 15.31% 11.07%
    7:  12.65% 14.63% 10.67%
    8:  12.39% 14.28% 10.50%
    9:  11.89% 13.77% 10.02%
   10:  11.55% 13.69%  9.42%
   11:  11.31% 13.41%  9.21%
   12:  11.03% 13.12%  8.94%
   13:  10.74% 13.08%  8.41%
   14:  10.43% 12.73%  8.13%
   15:  10.19% 12.64%  7.75%
   16:  10.08% 12.53%  7.64%
   17:   9.99% 12.39%  7.58%
   18:   9.85% 12.21%  7.49%
   19:   9.70% 12.07%  7.34%
   20:   9.61% 12.08%  7.14%
   21:   9.44% 12.02%  6.85%
   22:   9.44% 11.96%  6.91%
   23:   9.30% 11.83%  6.77%
```

```
> tab3 <- table(predict_rf2, data_nb$y)
> confusionMatrix(tab3)
Confusion Matrix and Statistics


predict_rf2    no   yes
        no  32669  1786
        yes  3879 34762

               Accuracy : 0.9225
                 95% CI : (0.9205, 0.9244)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.845
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8939
            Specificity : 0.9511
         Pos Pred Value : 0.9482
         Neg Pred Value : 0.8996
             Prevalence : 0.5000
         Detection Rate : 0.4469
   Detection Prevalence : 0.4714
      Balanced Accuracy : 0.9225

       'Positive' Class : no
```

```
> predict_rf2 <- predict(model_rf2, data = data_nb, na.action = na.pass)
> roc_rf2 <- roc(y ~ as.numeric(predict_rf2), data = data_nb)
> plot(roc_rf2, col = "red")
> auc(roc_rf2) # Area under curve
Area under the curve: 0.9225
>
```



The Random Forest gives an **accuracy of 92.25%**

Model	Accuracy
KNN	**95.14%**
Naive Bayes	80.47%
Logistic Full	82.12304%
Logistic Stepwise Forward	82.12915%
Logistic Stepwise Backward	82.12983%
Decision Tree	81.35%
Random Forest	92.25%

Hence, we were still getting our KNN model as best fit.