Marketing Strategy for Banks

AKSHAY VARIK

Spring 2016

STAT 571- ADVANCED STATISTICS FOR MANAGEMENT

CONTENTS

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to Dr. Linda Zhao. Professor STAT 471/571/701 Wharton School, University of Pennsylvania, Philadelphia for providing me an opportunity to do this project by myself. Without her guidance, support and assurance this would definitely would not have been successful.

I would also like to thank all the TA's of the course for patiently assisting me during times when I need clarifications. I would also like to thank my colleagues without whom this course would not have been the same experience.

This is exactly how I feel having had this course experience!!

Akshay Varik
Graduate Student
Mechanical Engineering and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104

STAT 471/571/701
Final Project Report
Spring 2016
University of Pennsylvania

Name: AKSHAY VARIK
Professor: Dr. LINDA ZHAO

PROJECT PROPOSAL

Objective:

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. The classification goal is to predict if the client will subscribe (yes/no) a term deposit.

Data:

The data set characteristic is of type multivariate. The data has 41188 observations, 20 predictor variables and the response variable (term deposit- variable 'y'). The data is ordered by date (from May 2008 to November 2010), and is very close to the data analyzed in [Moro et al., 2014]. On quick exploration of the data it is seen that around 89% of the response variable is 'no' while 11% is 'yes'.

Source:

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

Link:

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

PROJECT REPORT- INTRODUCTION

Background:

At the banking institutions it's a very common practice to place calls to their customers to encourage them to open a term deposit with them. A term deposit is a deposit held at a financial institution that has a fixed term. These are generally short-term with maturities ranging anywhere from a month to a few years. When a term deposit is purchased, the lender (the customer) understands that the money can only be withdrawn after the term has ended or by giving a predetermined number of days' notice. These types of financial products are sold by banks, thrift institutions and credit unions.

Goal of the Study:

The goal of the study here is to determine the likelihood of a customer to open the term deposit and thereby classify him/her into two levels (yes or no).

Background on the Data Set:

Source of the data:
The data is available at the Machine Learning Repository of the Centre for Machine Learning and Intelligent Systems. It was originally obtained from the Elsevier journal scientific paper written in June 2014 by Moro et al.

Characteristics of the Data Set:

The data set has 41188 values which are essentially phone calls placed to these many clients. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. There are 20 predictor variables used and the response variable is y (has the client subscribed a term deposit? (Binary: 'yes', 'no')).

Description of variables:

- Bank Client Data

1. Age of the client (numeric)
2. Job: The type of job the client performs (categorical)

- Admin
- Blue collar
- Entrepreneur
- Housemaid
- Management
- Retired
- Self-employed
- Services
- Student
- Technician
- Unemployed
- Unknown

3. Marital: It's the marital status of the client (categorical)
    - Divorced
    - Married
    - Single
    - Unknown
4. Education: It's the educations level of the client (categorical)
    - basic.4y (Till 4^{th} grade)
    - basic.6y (Till 6^{th} grade)
    - basic.9y (Till 9^{th} grade)
    - high school
    - illiterate
    - professional.course
    - university.degree
    - unknown
5. Default: Has the client got any credit in default? (categorical)
    - No
    - Yes
    - Unknown
6. Housing: Has the client got any housing loan? (categorical)
    - No
    - Yes
    - Unknown
7. Loan: Has the client got any personal loan? (categorical)
    - No
    - Yes
    - Unknown

- Data related with the last contact of the current campaign

8. Contact: How was the client contacted? (categorical)
    - Cellular
    - Telephone
9. Month: Last contact month of the year (categorical)
    - January
    - February
    - March
    - April
    - May
    - June
    - July
    - August
    - September
    - October
    - November
    - December
10. Day_of_week: Last contact day of the week (categorical)
    - Monday
    - Tuesday
    - Wednesday
    - Thursday
    - Friday
11. Duration: It's the last contact duration, in seconds (numeric).

- Other attributes

12. Campaign: Number of contacts performed during this campaign and for this client which includes the last contact (numeric)
13. Pdays: Number of days that passed by after the client was last contacted from a previous campaign and 999 means client was not previously contacted (numeric)
14. Previous: Number of contacts performed before this campaign and for this client (numeric)
15. Poutcome: Outcome of the previous marketing campaign (categorical)
    - Failure
    - Non existent
    - Succes

- Social and economic context attributes

16. Emp.var.rate: Employment variation rate - quarterly indicator (numeric)
17. Cons.price.idx: Consumer price index - monthly indicator (numeric)
18. Cons.conf.idx: Consumer confidence index - monthly indicator (numeric)
19. Euribor3m: Euribor 3 month rate - daily indicator (numeric)
    Euribor is short for Euro Interbank Offered Rate. The Euribor rates are based on the interest rates at which a panel of European banks borrow funds from one another.
20. Nr.employed: Number of employees - quarterly indicator (numeric)

Initial Data Cleaning:

The data that we have is already clean and has no missing values. Variable 'duration' has been removed because this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

I have renamed the response variable as TD. The dimension of the data is 41188 into 21. Since it's important to keep in mind the computational speed to build models and train data, I have used 5000 data values of the 41188 available. In the original data, what's noticeable is that the percentage of TD=yes is just 11% roughly. This is highly skewed and hence to build a strong classifier, I have maintained a 55% TD=no to 45% TD=yes ratio amongst the randomly sub-setted dataset of 5000.

All the study has been performed on this cleaned dataset (data3=data.cleaned). The remaining data could be very well used for testing the classifier.

Approach adopted:

Essentially, I adopt a simple approach to build my final and the best model. First 1 build 3 models by logistic regression. The first model in it will contain all the variables. The second will have variables I obtain using the function Regsubsets. In this I explore all the types that include exhaustive search, forward selection and backward selection Last, I build a logistic regression model containing models I get from regularization (LASSO- L1 norm).

The next three models are built using the same approach as above but with 10 fold cross validation. This way we get three more models (again primarily by using Logistic Regression).

The next 3 models are built by Linear Discriminant Approach (LDA). The first of them is by using all the variables, while the second contains the same variables as obtained from Regsubsets used from above. The last one is built using variables I get from the LASSO regularization.

Finally, I explore the Random Forest method for classification. Again three models are built here, one having all the variables, the other having variables obtained from Regsubsets function and last one from the LASSO regularization. While studying this I also performed Bagging and Bootstrapping to build strong models.

Having obtained 12 models, I compare all the models through a common parameter like MCE on the test data and see which the best model out of them was. Lastly, I try to use an ensembled approach and use the models to formulate a common model. I use three ensemble approaches, just a mean of the predictions of all methods, fitting regression model on the model and then classifying them and finally fitting a random forest tree on the predictions of other models (this eventually was the selected model).

This ensembled model constituted to be the best model and was the chosen classifier.

Important Topics Touched Upon:

Logistic Regression, Linear Discriminant Analysis, Random Forest, Decision Trees, Bagging, Bootstrap, Cross Validation, LASSO, Subsets, Ensemble, Multiple Linear Regression

DETAILED MACHINE LEARNING ANALYSIS

The Study (of Original Data):

Firstly, I tried to understand the original data, the summary of which has been presented below. As can be seen the data is pleasant and has no missing values.

```
> summary(data)
      age                 job            marital     
 Min.   :17.00   admin.     :10422   divorced: 4612  
 1st Qu.:32.00   blue-collar: 9254   married :24928  
 Median :38.00   technician : 6743   single  :11568  
 Mean   :40.02   services   : 3969   unknown :   80  
 3rd Qu.:47.00   management : 2924                   
 Max.   :98.00   retired    : 1720                   
                 (Other)    : 6156                   
               education        default         housing           loan      
 university.degree  :12168   no     :32588   no     :18622   no     :33950  
 high.school        : 9515   unknown: 8597   unknown:  990   unknown:  990  
 basic.9y           : 6045   yes    :    3   yes    :21576   yes    : 6248  
 professional.course: 5243                                                  
 basic.4y           : 4176                                                  
 basic.6y           : 2292                                                  
 (Other)            : 1749                                                  
      contact          month        day_of_week    duration     
 cellular :26144   may    :13769   fri:7827    Min.   :   0.0  
 telephone:15044   jul    : 7174   mon:8514    1st Qu.: 102.0  
                   aug    : 6178   thu:8623    Median : 180.0  
                   jun    : 5318   tue:8090    Mean   : 258.3  
                   nov    : 4101   wed:8134    3rd Qu.: 319.0  
                   apr    : 2632               Max.   :4918.0  
                   (Other): 2016                                
    campaign          pdays          previous            poutcome    
 Min.   : 1.000   Min.   :  0.0   Min.   :0.000   failure    : 4252  
 1st Qu.: 1.000   1st Qu.:999.0   1st Qu.:0.000   nonexistent:35563  
 Median : 2.000   Median :999.0   Median :0.000   success    : 1373  
 Mean   : 2.568   Mean   :962.5   Mean   :0.173                      
 3rd Qu.: 3.000   3rd Qu.:999.0   3rd Qu.:0.000                      
 Max.   :56.000   Max.   :999.0   Max.   :7.000                      

  emp.var.rate      cons.price.idx  cons.conf.idx     euribor3m    
 Min.   :-3.40000   Min.   :92.20   Min.   :-50.8   Min.   :0.634  
 1st Qu.:-1.80000   1st Qu.:93.08   1st Qu.:-42.7   1st Qu.:1.344  
 Median : 1.10000   Median :93.75   Median :-41.8   Median :4.857  
 Mean   : 0.08189   Mean   :93.58   Mean   :-40.5   Mean   :3.621  
 3rd Qu.: 1.40000   3rd Qu.:93.99   3rd Qu.:-36.4   3rd Qu.:4.961  
 Max.   : 1.40000   Max.   :94.77   Max.   :-26.9   Max.   :5.045  

  nr.employed      TD       
 Min.   :4964   no :36548  
 1st Qu.:5099   yes: 4640  
 Median :5191              
 Mean   :5167              
 3rd Qu.:5228              
 Max.   :5228              
```

```
> str(data)
'data.frame':   41188 obs. of  21 variables:
 $ age           : int  56 57 37 40 56 45 59 41 24 25 ...
 $ 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 ...
 $ duration      : int  261 149 226 151 307 198 139 217 380 50 ...
 $ campaign      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ pdays         : int  999 999 999 999 999 999 999 999 999 999 ...
 $ previous      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ poutcome      : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ emp.var.rate  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
 $ cons.price.idx: num  94 94 94 94 94 ...
 $ cons.conf.idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
 $ euribor3m     : num  4.86 4.86 4.86 4.86 4.86 ...
 $ nr.employed   : num  5191 5191 5191 5191 5191 ...
 $ TD            : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

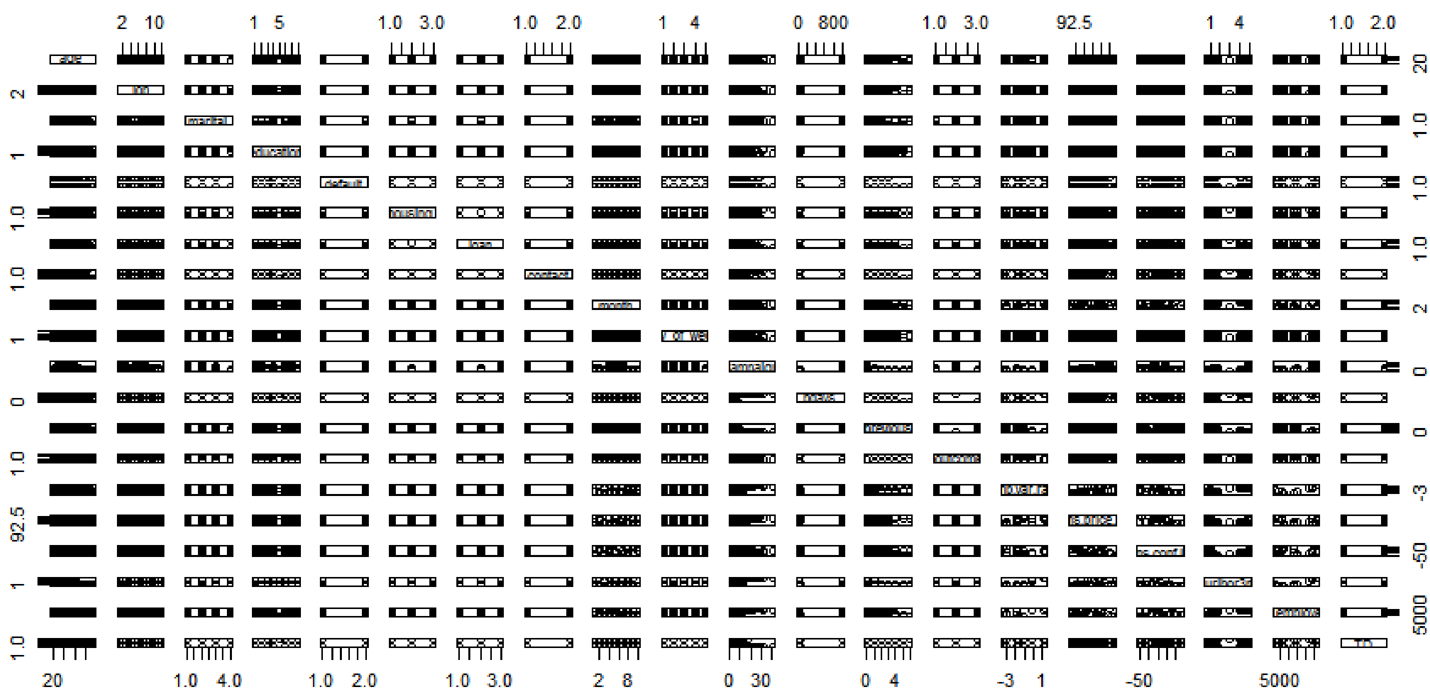
The Study (of Cleaned Data):

```
> summary(data.cleaned)
      age                 job            marital                    education        default    
 Min.   :17.00   admin.      :1338   divorced: 539   university.degree  :1626   no     :4148  
 1st Qu.:31.00   blue-collar: 994   married :2854   high.school        :1112   unknown: 852  
 Median :38.00   technician : 813   single  :1595   basic.9y           : 647                 
 Mean   :40.26   services   : 417   unknown :  12   professional.course: 615                 
 3rd Qu.:48.00   management : 347                   basic.4y           : 483                 
 Max.   :95.00   retired    : 317                   basic.6y           : 270                 
                 (Other)    : 774                   (Other)            : 247                 
    housing           loan            contact          month      day_of_week    campaign     
 no     :2271   no     :4115   cellular :3564   may    :1407   fri: 921    Min.   : 1.000  
 unknown: 118   unknown: 118   telephone:1436   jul    : 825   mon: 969    1st Qu.: 1.000  
 yes    :2611   yes    : 767                    aug    : 721   thu:1131    Median : 2.000  
                                                jun    : 634   tue: 986    Mean   : 2.413  
                                                nov    : 505   wed: 993    3rd Qu.: 3.000  
                                                apr    : 403               Max.   :42.000  
                                                (Other): 505                                
     pdays          previous              poutcome      emp.var.rate      cons.price.idx  cons.conf.idx  
 Min.   :  0.0   Min.   :0.0000   failure    : 551   Min.   :-3.4000   Min.   :92.20   Min.   :-50.80  
 1st Qu.:999.0   1st Qu.:0.0000   nonexistent:3966   1st Qu.:-1.8000   1st Qu.:92.89   1st Qu.:-42.70  
 Median :999.0   Median :0.0000   success    : 483   Median :-0.1000   Median :93.44   Median :-41.80  
 Mean   :893.2   Mean   :0.2924                      Mean   :-0.3986   Mean   :93.49   Mean   :-40.31  
 3rd Qu.:999.0   3rd Qu.:0.0000                      3rd Qu.: 1.4000   3rd Qu.:93.99   3rd Qu.:-36.40  
 Max.   :999.0   Max.   :6.0000                      Max.   : 1.4000   Max.   :94.77   Max.   :-26.90  

   euribor3m      nr.employed      TD      
 Min.   :0.634   Min.   :4964   no :2750  
 1st Qu.:1.260   1st Qu.:5099   yes:2250  
 Median :4.120   Median :5191             
 Mean   :3.073   Mean   :5141             
 3rd Qu.:4.959   3rd Qu.:5228             
 Max.   :5.045   Max.   :5228             
```

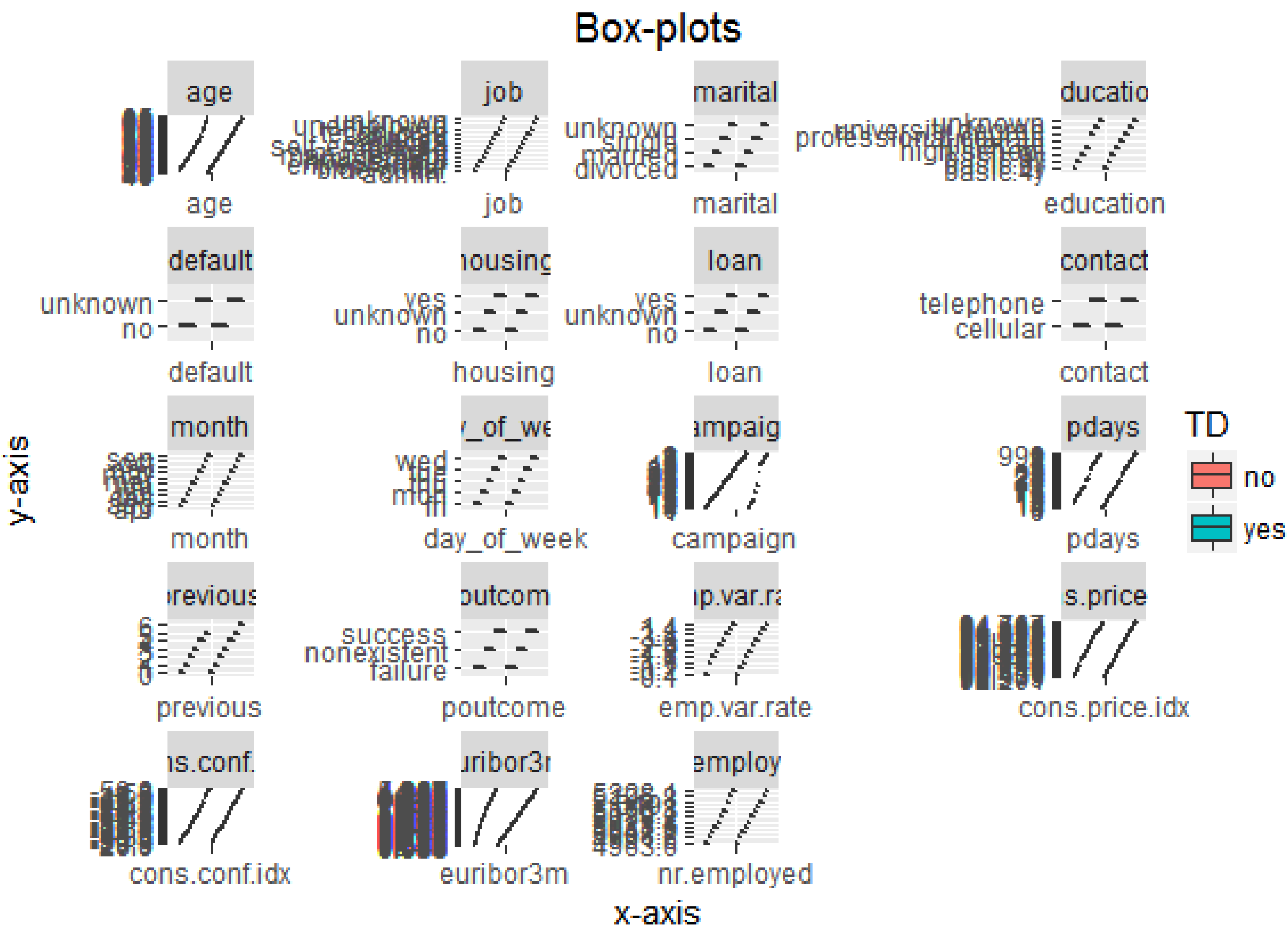
Below we can see how the ‘duration’ variable has been removed from the dataset.

```
> str(data.cleaned)
'data.frame':	5000 obs. of  20 variables:
 $ age           : int  50 35 27 29 37 52 24 52 30 39 ...
 $ job           : Factor w/ 12 levels "admin.","blue-collar",..: 1 2 1 1 2 2 9 6 1 1 ...
 $ marital       : Factor w/ 4 levels "divorced","married",..: 2 3 3 3 3 2 3 1 2 3 ...
 $ education     : Factor w/ 8 levels "basic.4y","basic.6y",..: 7 3 7 7 6 3 4 7 7 4 ...
 $ default       : Factor w/ 2 levels "no","unknown": 2 1 1 1 1 2 1 1 1 1 ...
 $ housing       : Factor w/ 3 levels "no","unknown",..: 3 1 1 1 1 1 1 1 1 1 ...
 $ loan          : Factor w/ 3 levels "no","unknown",..: 1 1 1 3 1 1 1 1 1 1 ...
 $ contact       : Factor w/ 2 levels "cellular","telephone": 2 1 1 1 1 2 1 1 2 2 ...
 $ month         : Factor w/ 10 levels "apr","aug","dec",..: 5 7 10 2 1 4 2 2 5 4 ...
 $ day_of_week   : Factor w/ 5 levels "fri","mon","thu",..: 3 1 5 4 1 1 3 3 1 3 ...
 $ campaign      : int  6 1 1 1 1 2 1 1 1 1 ...
 $ pdays         : int  999 999 999 999 999 999 999 15 999 999 ...
 $ previous      : int  0 0 0 0 0 0 0 1 0 0 ...
 $ poutcome      : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 3 2 2 ...
 $ emp.var.rate  : num  1.4 -1.8 -3.4 1.4 -1.8 1.4 -2.9 -2.9 1.4 1.4 ...
 $ cons.price.idx: num  94.5 92.9 92.4 93.4 93.1 ...
 $ cons.conf.idx : num  -41.8 -46.2 -29.8 -36.1 -47.1 -42.7 -31.4 -31.4 -41.8 -42.7 ...
 $ euribor3m     : num  4.958 1.313 0.781 4.963 1.405 ...
 $ nr.employed   : num  5228 5099 5018 5228 5099 ...
 $ TD            : Factor w/ 2 levels "no","yes": 1 2 2 1 1 2 2 2 1 1 ...
```

The correlations among the variables can be seen from the pair plot as can be seen below.



Most importantly what could be observed is that duration is highly correlated with TD and hence was removed to build more robust model (as suggested on the website as well). The box plot below shows visually the categorical and numerical variables.




```
> cor(data.cleaned[,unlist(lapply(data.cleaned, is.numeric))])
                       age    campaign       pdays    previous emp.var.rate cons.price.idx
age             1.000000000  0.03054528 -0.03569918  0.04746841  -0.02656427    0.007158971
campaign        0.030545284  1.00000000  0.08763713 -0.08875156   0.18606153    0.127368127
pdays          -0.035699177  0.08763713  1.00000000 -0.71083208   0.34123673    0.036930752
previous        0.047468410 -0.08875156 -0.71083208  1.00000000  -0.40684013   -0.091404963
emp.var.rate   -0.026564268  0.18606153  0.34123673 -0.40684013   1.00000000    0.725716595
cons.price.idx  0.007158971  0.12736813  0.03693075 -0.09140496   0.72571660    1.000000000
cons.conf.idx   0.107248305 -0.02459104 -0.13698801  0.08600497  -0.02721782   -0.126220160
euribor3m      -0.024627011  0.17326860  0.39620558 -0.46544609   0.96121262    0.584376953
nr.employed    -0.054663059  0.17419436  0.48578152 -0.53939334   0.87455829    0.369676023
               cons.conf.idx   euribor3m nr.employed
age               0.10724830 -0.02462701 -0.05466306
campaign         -0.02459104  0.17326860  0.17419436
pdays            -0.13698801  0.39620558  0.48578152
previous          0.08600497 -0.46544609 -0.53939334
emp.var.rate     -0.02721782  0.96121262  0.87455829
cons.price.idx   -0.12622016  0.58437695  0.36967602
cons.conf.idx     1.00000000  0.07448255 -0.06296017
euribor3m         0.07448255  1.00000000  0.94214899
nr.employed      -0.06296017  0.94214899  1.00000000
```

The above shows in numbers how the variables are correlated. This has been done only for numeric variables. It is logical that the euribor rate (rate of interest) would be correlated to how often employment is changed by people in the market. More change suggests instability and hence the rate would be lower (low job security).

Next, thing I divided the dataset into training and testing data, so that I can train my model on the training set and then test it on the hidden test set to check for accuracy. This helps in understanding model accuracies and to pick amongst the best models. For this purpose I am taking the 4000 values to be training set and the remaining 1000 to be the test set.

Building the classifier:

The most important task to build the classifier is to build several models of different types and see their prediction performance on the dataset. The final classifier will be the one that performs the best amongst the lot.

- Model 1:

Here I performed logistic regression to build a model by considering all the variables. The summary of this fit is given below.

```
> fit1=glm(TD~., data.cleaned.train, family=binomial(logit))
> summary(fit1)

Call:
glm(formula = TD ~ ., family = binomial(logit), data = data.cleaned.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1304  -0.8085  -0.5760   0.7846   2.1051

Coefficients: (1 not defined because of singularities)
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                 -9.155e+01  8.385e+01  -1.092 0.274905
age                          2.410e-03  4.602e-03   0.524 0.600445
jobblue-collar              -6.776e-02  1.427e-01  -0.475 0.634817
jobentrepreneur              1.753e-01  2.180e-01   0.804 0.421114
jobhousemaid                 3.188e-01  2.800e-01   1.139 0.254852
jobmanagement               -2.091e-01  1.701e-01  -1.229 0.218978
jobretired                   3.272e-01  2.146e-01   1.525 0.127352
jobself-employed             2.132e-01  2.213e-01   0.964 0.335204
jobservices                  1.317e-01  1.586e-01   0.830 0.406525
jobstudent                   5.058e-01  2.490e-01   2.032 0.042203 *
jobtechnician                2.013e-02  1.343e-01   0.150 0.880823
jobunemployed                3.735e-01  2.644e-01   1.413 0.157732
jobunknown                   6.782e-02  4.453e-01   0.152 0.878954
maritalmarried               7.602e-02  1.267e-01   0.600 0.548451
maritalsingle                4.717e-02  1.447e-01   0.326 0.744447
maritalunknown               3.705e-01  8.224e-01   0.451 0.652348
educationbasic.6y            2.048e-01  2.078e-01   0.986 0.324335
educationbasic.9y            7.268e-02  1.734e-01   0.419 0.675081
educationhigh.school         8.216e-02  1.736e-01   0.473 0.635981
educationilliterate          1.112e+01  1.970e+02   0.056 0.954985
educationprofessional.course 1.567e-01  1.961e-01   0.799 0.424139
educationuniversity.degree   2.241e-01  1.755e-01   1.277 0.201709
educationunknown            -2.815e-02  2.304e-01  -0.122 0.902765
defaultunknown              -2.038e-01  1.114e-01  -1.830 0.067257 .
housingunknown              -5.376e-01  2.709e-01  -1.985 0.047142 *
housingyes                  -6.773e-03  7.813e-02  -0.087 0.930917
loanunknown                         NA         NA      NA       NA
loanyes                     -3.682e-03  1.077e-01  -0.034 0.972738
contacttelephone            -6.947e-01  1.487e-01  -4.672 2.98e-06 ***
monthaug                     5.028e-01  3.097e-01   1.624 0.104450
monthdec                     2.308e+00  1.178e+00   1.959 0.050131 .
monthjul                     1.975e-01  1.947e-01   1.014 0.310530
monthjun                    -3.328e-01  2.955e-01  -1.126 0.260020
monthmar                     1.965e+00  4.367e-01   4.501 6.76e-06 ***
monthmay                    -5.219e-01  1.697e-01  -3.076 0.002101 **
monthnov                    -7.347e-01  2.388e-01  -3.077 0.002091 **
monthoct                    -6.099e-01  3.355e-01  -1.818 0.069077 .
monthsep                    -2.606e-01  4.142e-01  -0.629 0.529312
day_of_weekmon              -3.191e-01  1.238e-01  -2.578 0.009951 **
day_of_weekthu              -3.628e-01  1.211e-01  -2.996 0.002736 **
day_of_weektue              -1.837e-01  1.250e-01  -1.470 0.141554
day_of_weekwed              -3.907e-02  1.225e-01  -0.319 0.749809
campaign                    -4.337e-02  1.728e-02  -2.510 0.012068 *
pdays                       -3.448e-04  4.639e-04  -0.743 0.457300
previous                    -2.414e-01  1.602e-01  -1.507 0.131824
poutcomenonexistent          3.238e-01  2.246e-01   1.442 0.149282
poutcomesuccess              1.590e+00  4.628e-01   3.436 0.000591 ***
emp.var.rate                -1.220e+00  3.348e-01  -3.644 0.000269 ***
cons.price.idx               1.287e+00  5.617e-01   2.291 0.021981 *
cons.conf.idx                1.907e-03  1.910e-02   0.100 0.920492
euribor3m                    6.010e-01  2.827e-01   2.126 0.033489 *
nr.employed                 -5.994e-03  6.709e-03  -0.894 0.371589
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5512.4  on 3999  degrees of freedom
Residual deviance: 4149.5  on 3949  degrees of freedom
AIC: 4251.5

Number of Fisher Scoring iterations: 10
```

What can be seen here is that a lot of variables have very low significance.

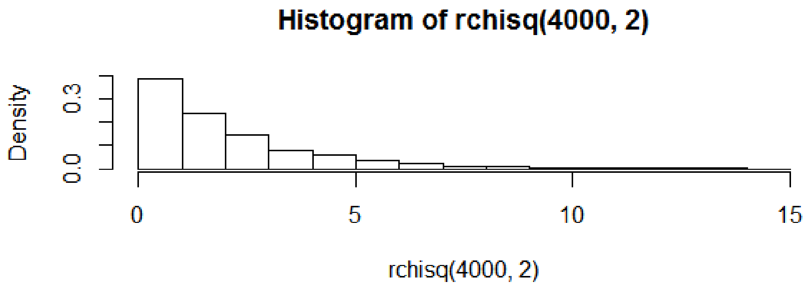
Now, the next thing I do is the chi-square test. This was done using anova function. This is type 1 test where each variable gets added sequentially. If a particular variable is of high significance then the reduction in the residual deviance will be great. This could be seen in the AIC value.

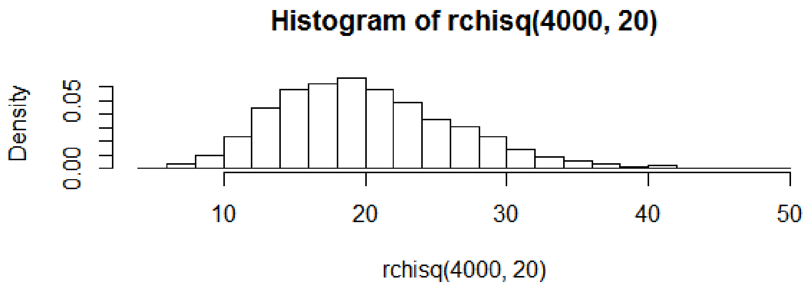
```
> anova(fit1, test="Chisq") # to test if the model is useful: null hypothesis is
 all (but the intercept) coeff's are 0
Analysis of Deviance Table

Model: binomial, link: logit

Response: TD

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)    
NULL                            3999     5512.4              
age             1    4.931      3998     5507.4  0.026374 *  
job            11  177.414      3987     5330.0 < 2.2e-16 ***
marital         3   14.689      3984     5315.3  0.002102 ** 
education       7   19.203      3977     5296.1  0.007574 ** 
default         1   94.995      3976     5201.1 < 2.2e-16 ***
housing         2    0.748      3974     5200.4  0.688022    
loan            1    0.905      3973     5199.5  0.341414    
contact         1  174.992      3972     5024.5 < 2.2e-16 ***
month           9  314.283      3963     4710.2 < 2.2e-16 ***
day_of_week     4    9.712      3959     4700.5  0.045562 *  
campaign        1   24.450      3958     4676.0 7.626e-07 ***
pdays           1  213.801      3957     4462.2 < 2.2e-16 ***
previous        1    0.180      3956     4462.1  0.671072    
poutcome        2   13.657      3954     4448.4  0.001083 ** 
emp.var.rate    1  209.450      3953     4239.0 < 2.2e-16 ***
cons.price.idx  1   76.782      3952     4162.2 < 2.2e-16 ***
cons.conf.idx   1    7.405      3951     4154.8  0.006505 ** 
euribor3m       1    4.495      3950     4150.3  0.033999 *  
nr.employed     1    0.809      3949     4149.5  0.368488    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Here it is seen that the adding job variable on top of the age variable is helpful since the significance is high. AIC value of Model 1 is 4251.5. A model will be better than this if its AIC value is lower.

So to select the variables I use Regsubsets function. To keep my model size feasible I choose 8 variables (There are 8 variables whose significance level is <0.001). This is a good way to limit the model size. Exhaustive, forward and backward searches could be done to obtain the 8 variables. This has been done in subsequent models.

The chi-square distribution is as follows:

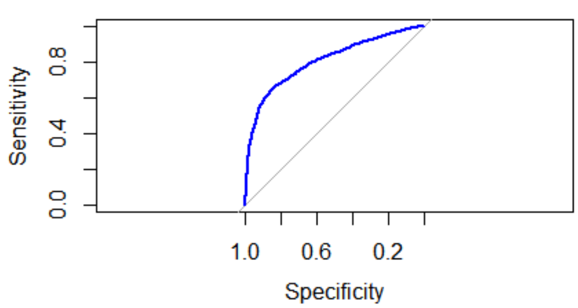


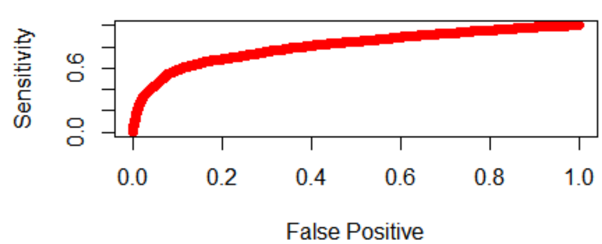


Continuing with the process of prediction using this classifier and calculating the miss-calculation error and the Area under the ROC curve as a performance measure of this model, I get the MCE on the testing data to be 0.348. The confusion matrix I obtain for the test data is as follows:

```
> cm=table(fit1.pred.test, data.cleaned.test$TD)
> cm

fit1.pred.test   0   1
             0 561 340
             1   8  91
> fit1.mce.test=mean(fit1.pred.test != data.cleaned.test$TD)
> fit1.mce.test
[1] 0.348
```

The specificity is 0.98594, sensitivity is 0.2111 and the false positive value is 0.01406.



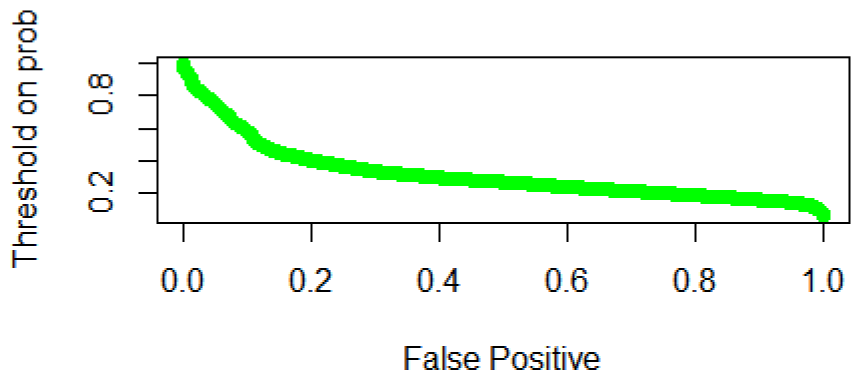


The above are the ROC curves. The Area under this curve (AUC) is 0.8063. For model selection higher the AUC better is the model.

To tune it we can use the correct threshhold value for probabilty which has been kept to 0.9. This makes sure that the false.positive value is very small. This is the Bayes rule. Essentially if the loss (cost) of making a '1' to a '0' is given by a_{1,0} and if the loss (cost) of making a '0' to a '1' is given by a_{0,1}, then

P(Y=1|X)> (a_{0,1}/a_{1,0})/(1+(a_{0,1}/a_{1,0}))

In this case, I have taken a_{0,1}/a_{1,0}=9



To summarize the model here, it constitutes all the variables whose coefficients are shown before through the summary of the fit. Its not a great model. The classification boundary is essentially the log odds ratio of the hat Y=1 to hat Y=0 equal to the model. Using a different threshhold on probabilities (different loss function) we get different classifiers and overlaying the ROC curves together we can see which one does better (It's a challenge to visualize the classification boundary for such high dimension). Here we stick to 0.9 because of the lot this perfoms the best.

TD = - 9.155e+01 + sum across all variables (Corresponding Coeff * Correspoding Variable)

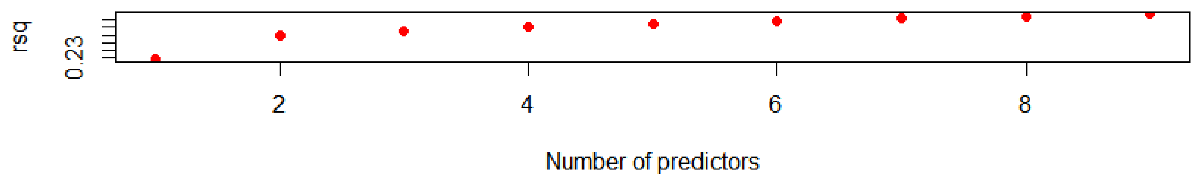
Model 2:

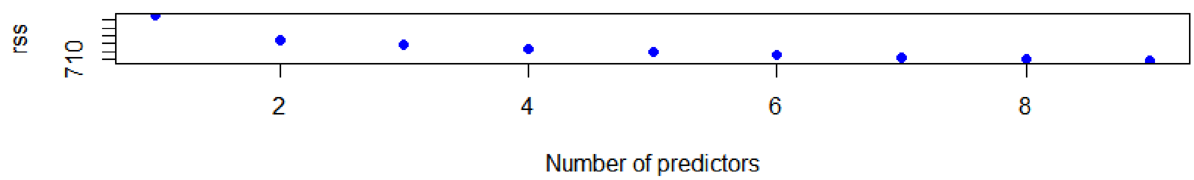
Here I used model selection method to get 8 variables using Regsubsets.

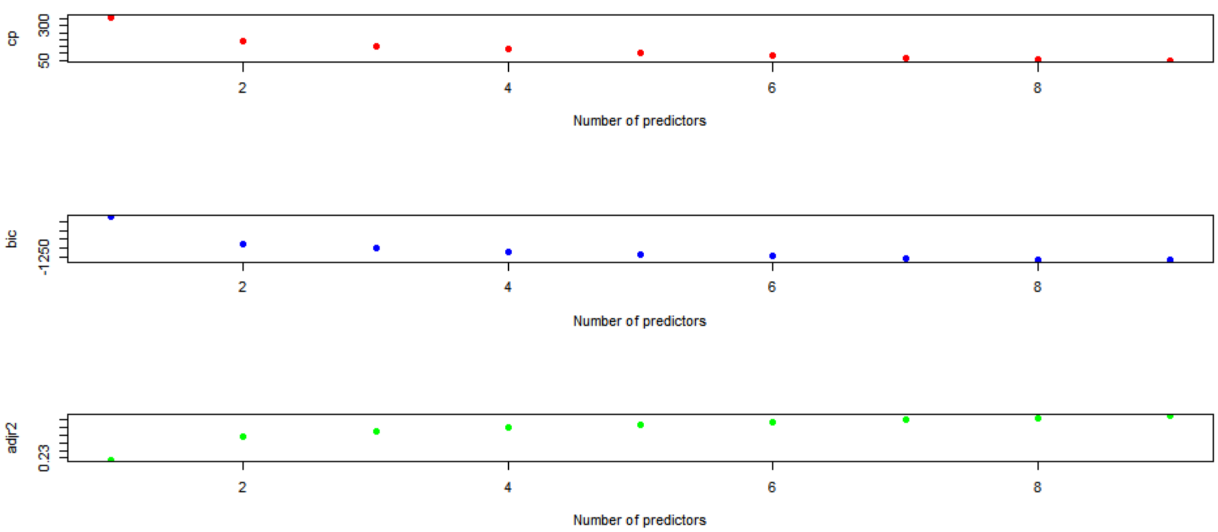
1. Exhaustive Search

The values of the different criterions that were obtained is given below. RSS and RSQ and not the best criterions to be used for model selection. RSS will of course reduce when more variables are used

```
> fit2.e=summary(fit2.exh)
> fit2.e$cp
[1] 363.31381 188.28155 155.73097 129.57556 106.52375  88.06366  67.89702  57.24805  47.18157
> fit2.e$bic
[1] -1018.937 -1176.473 -1201.361 -1220.312 -1236.417 -1248.161 -1261.690 -1265.903 -1269.576
> fit2.e$rss
[1] 765.5926 734.5018 728.4339 723.4892 719.0896 715.4963 711.6033 709.3819 707.2628
> fit2.e$rsq
[1] 0.2280852 0.2594328 0.2655508 0.2705363 0.2749723 0.2785952 0.2825203 0.2847601 0.2868967
```







The 8 variables and their corresponding coefficients that we obtain are

```
> coef(fit2.exh,8)
    (Intercept)         monthaug         monthmay         monthnov         monthoct
   -13.38889017       0.06402441      -0.17817079      -0.13256959      -0.03979464
poutcomesuccess     emp.var.rate   cons.price.idx      loanunknown
     0.21224067      -0.15925222       0.14776708      -0.09273530
```

Now fitting a logistic regression model on the chosen variables, we get

```
Call:
glm(formula = TD ~ month + poutcome + emp.var.rate + cons.price.idx +
    loan, family = binomial(logit), data = data.cleaned.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9327  -0.7809  -0.6074   0.8085   2.1173

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -113.44041   14.09008  -8.051 8.21e-16 ***
monthaug               0.91670    0.19455   4.712 2.45e-06 ***
monthdec               2.37249    1.05456   2.250  0.02447 *
monthjul               0.55310    0.17055   3.243  0.00118 **
monthjun              -0.36521    0.17451  -2.093  0.03637 *
monthmar               2.12738    0.38864   5.474 4.40e-08 ***
monthmay              -0.57024    0.14202  -4.015 5.94e-05 ***
monthnov              -0.17439    0.17937  -0.972  0.33091
monthoct               0.04623    0.25262   0.183  0.85479
monthsep               0.23474    0.28783   0.816  0.41475
poutcomenonexistent    0.58904    0.12727   4.628 3.69e-06 ***
poutcomesuccess        1.83650    0.22553   8.143 3.85e-16 ***
emp.var.rate          -0.92714    0.05725 -16.196  < 2e-16 ***
cons.price.idx         1.20055    0.15019   7.994 1.31e-15 ***
loanunknown           -0.53236    0.26141  -2.036  0.04170 *
loanyes                0.01664    0.10554   0.158  0.87473
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5512.4  on 3999  degrees of freedom
Residual deviance: 4226.9  on 3984  degrees of freedom
AIC: 4258.9

Number of Fisher Scoring iterations: 6
```

```
> anova(fit1,fit2)
Analysis of Deviance Table

Model 1: TD ~ 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 + month.f
Model 2: TD ~ month + poutcome + emp.var.rate + cons.price.idx + loan
  Resid. Df Resid. Dev  Df Deviance
1      3949     4149.5
2      3984     4226.9 -35  -77.471
```

The above compares the 2 models generated so far. Surprisingly model 2 doesn't perform as well than model 1 because we see that the deviance actually increased as compared to model 1.

2. Forward Selection

The same practice as done for Exhaustive search was performed for Forward selection under Regsubsets.

```
> fit2.f$cp
[1] 353.37616 178.74747 146.27565 120.18443  97.18973  78.77628  62.38800  48.04004  38.00107
> fit2.f$bic
[1] -1018.937 -1176.473 -1201.361 -1220.312 -1236.417 -1248.161 -1258.003 -1265.903 -1269.576
> fit2.f$rss
[1] 765.5926 734.5018 728.4339 723.4892 719.0896 715.4963 712.2595 709.3819 707.2628
> fit2.f$rsq
[1] 0.2280852 0.2594328 0.2655508 0.2705363 0.2749723 0.2785952 0.2818587 0.2847601 0.2868967
> 
```

The 8 variables and their corresponding coefficients that we obtain are,

```
> coef(fit2.for,8)
  (Intercept)       monthaug       monthmay       monthnov       monthoct poutcomesuccess
 -13.38889001     0.06402441    -0.17817079    -0.13256959    -0.03979464      0.21224068
 emp.var.rate cons.price.idx    loanunknown
  -0.15925222     0.14776708    -0.09273530
```

We see that forward selection produces the exact same model as exhaustive search.

3. Backward Selection

Finally, backward selection was performed under Regsubsets. The corresponding criterion values we get are:

```
> fit2.b$cp
[1] 363.31381 188.28154 155.73096 135.59183 113.88276  89.15171  67.89701  57.24805  47.18157
> fit2.b$bic
[1] -1018.937 -1176.473 -1201.361 -1214.475 -1229.235 -1247.093 -1261.690 -1265.903 -1269.576
> fit2.b$rss
[1] 765.5926 734.5018 728.4339 724.5458 720.3820 715.6874 711.6033 709.3819 707.2628
> fit2.b$rsq
[1] 0.2280852 0.2594328 0.2655508 0.2694710 0.2736692 0.2784025 0.2825203 0.2847601 0.2868967
> 
```

Observe that the criterion values for the backward and forward selection are the same.

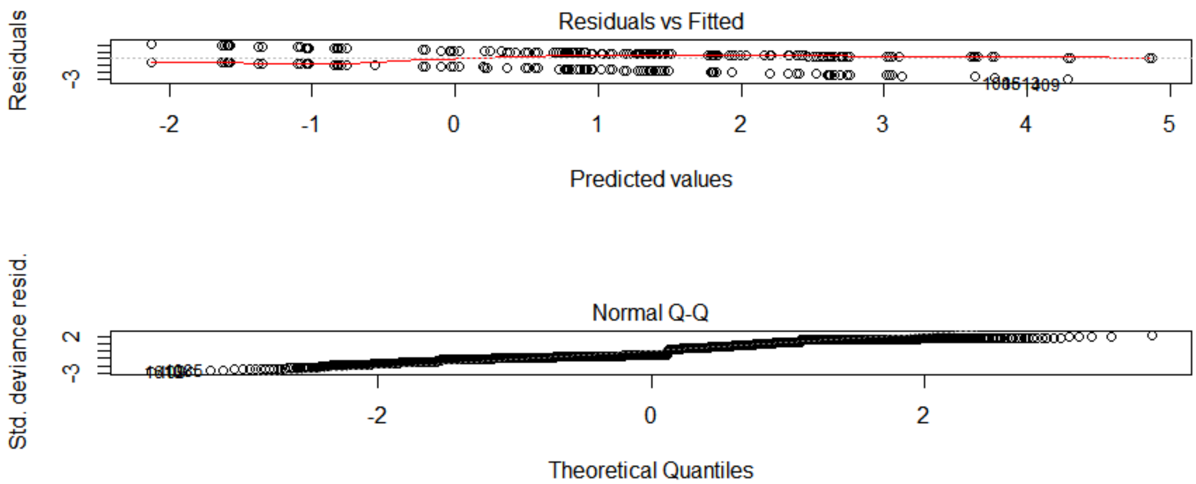
The 8 variables and their corresponding coefficients that we obtain are,

```
> coef(fit2.bac,8)
  (Intercept)       monthaug       monthmay       monthnov       monthoct poutcomesuccess
 -13.38889001     0.06402441    -0.17817079    -0.13256959    -0.03979464      0.21224068
 emp.var.rate cons.price.idx    loanunknown
  -0.15925222     0.14776708    -0.09273530
```

The coefficient values of this is also the same as the above 2 methods.
Thus the model (Model 2) we get is on performing logistic regression and retaining variables with significance level of ***:

TD = -113.44 + 0.9167*monthaug + 2.127*monthmar - 0.57*monthmay + 0.589*poutcomenonexistent + 1.8365*poutcomesuccess - 0.927*emp.var.rate + 1.2*cons.price.idx

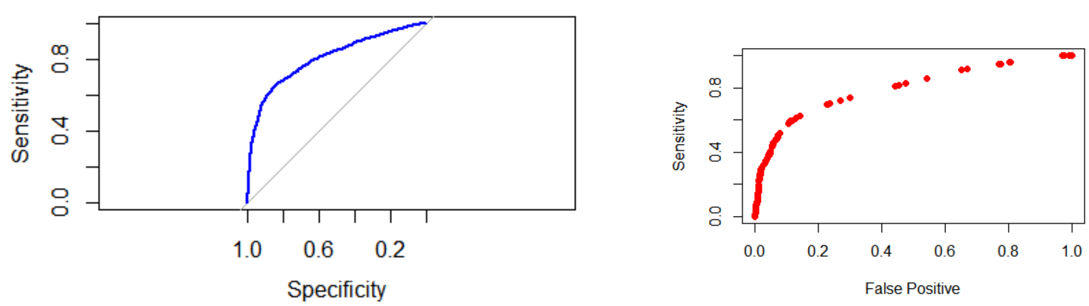
The above fit can be seen graphically below.



The MCE of this model on the test data is 0.346 and the confusion matrix is given below.

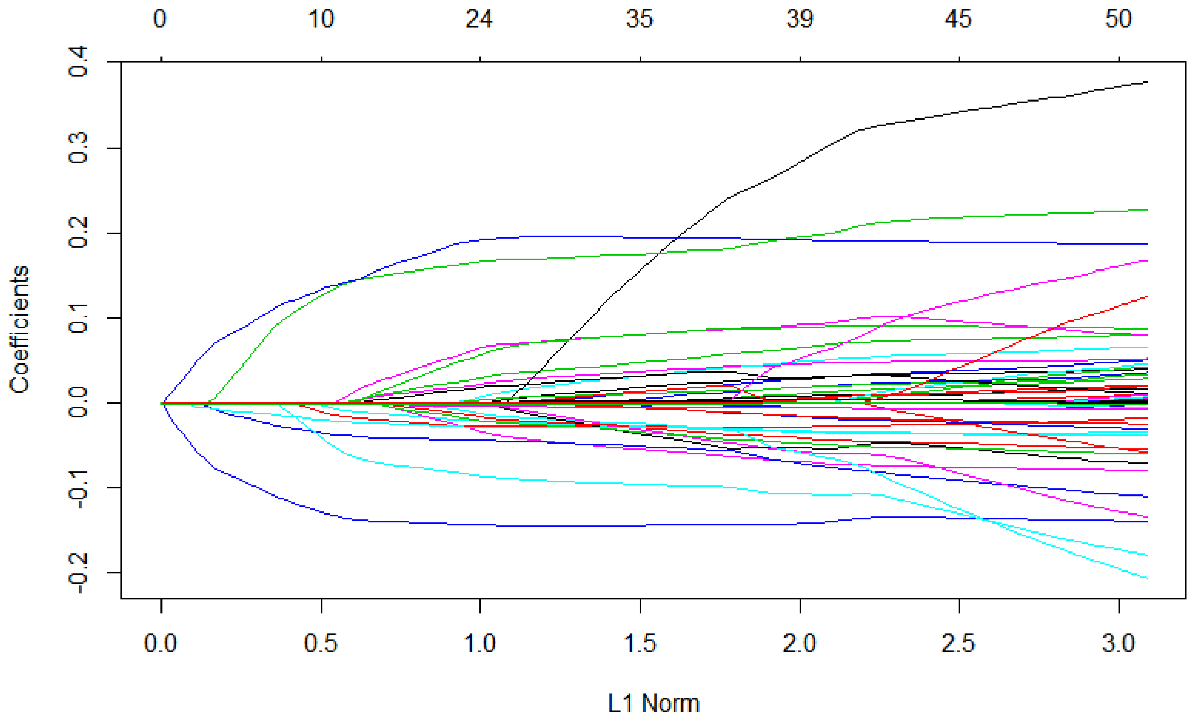
```
> fit2.mce.test
[1] 0.346
> sensitivity=cm[2,2]/sum(data.cleaned.test$TD =="1")
> cm

fit2.pred.test   0   1
             0 561 338
             1   8  93
```

The ROC curve obtained is show below. The AUC value is 0.7967.

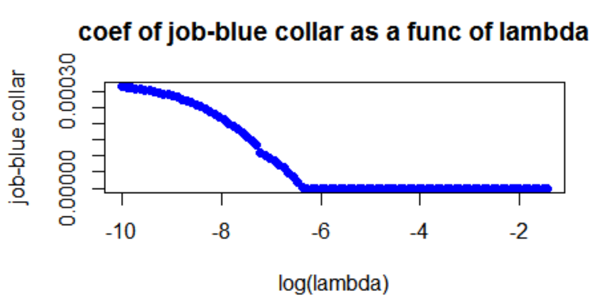


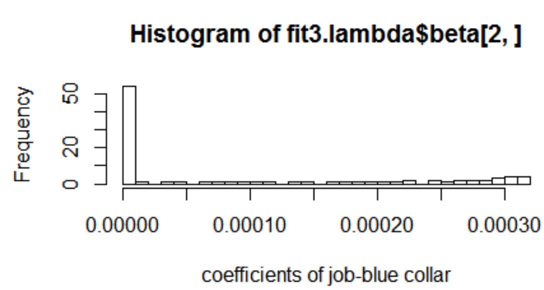
- <u>Model 3:</u>

Here the model has been built by using regularization methods. I stick with LASSO than ridge since LASSO has the ability to do feature selection. When the lambda value is not specified in glmnet, the output consists of 100 outputs, one for each lambda. A plot of it shows that each hat beta is shrinking towards a 0 as L2 norm of beta is smaller, which is equivalent to lambda getting larger as can be seen below.

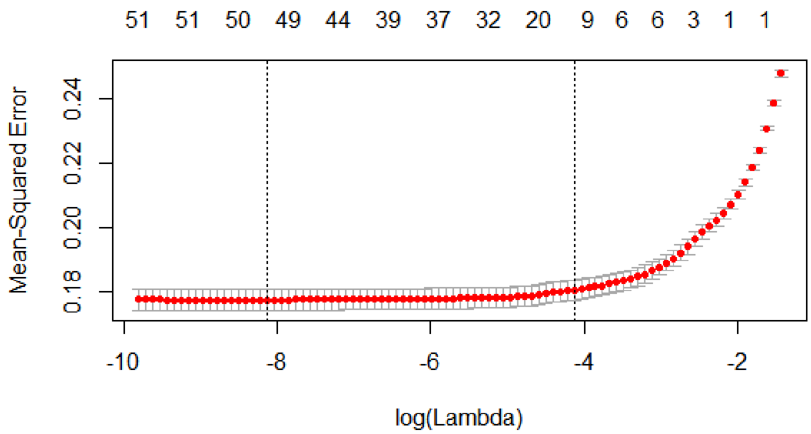


Also lambda values here have huge variability and hence log value is applied. The coefficients also have variability as shown below.

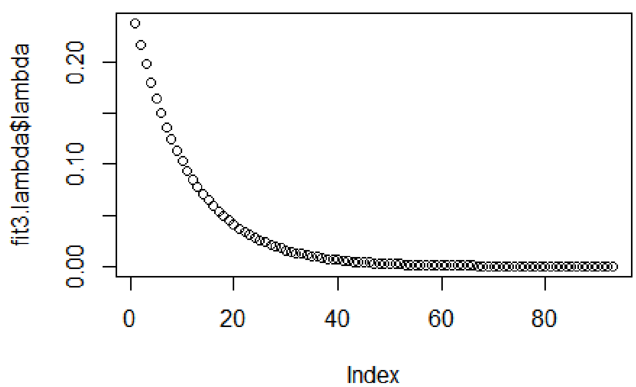




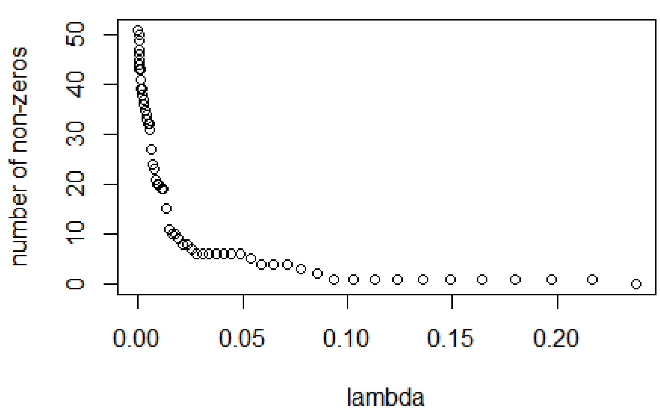
The below graph is an important one on basis of which I can choose the lambda value. Low lambda value means less penalty and the beta values will be high and vice versa.



The 100 lambda values used are shown, and the min lambda value is 0.00029318.



The number of non-zero terms as a function of the lambda used is shown below. As can be seen, higher the value of lambda lesser are the number of non-zeros.



Here I used lambda.1se so that I can keep a tight hold on the number of features selected. The variables (10 make it) and their respective coefficients that make it are,

```
> coef.1se
    (Intercept)   jobblue-collar   defaultunknown contacttelephone         monthmar
    9.938190133     -0.015419691     -0.007625655     -0.038230522      0.134076640
       monthmay         monthnov         campaign  poutcomesuccess     emp.var.rate
   -0.134229551     -0.053642490     -0.001189384      0.138269658     -0.022161746
    nr.employed
   -0.001838291
```

Now fitting a logistic regression model on the chosen variables,

```
> summary(fit3)

Call:
glm(formula = TD ~ job + default + contact + month + campaign +
    poutcome + emp.var.rate + nr.employed, data = data.cleaned.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1966  -0.2959  -0.1371   0.3071   0.9838

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)         10.4800215  1.0419321  10.058  < 2e-16 ***
jobblue-collar      -0.0356227  0.0204801  -1.739  0.08205 .
jobentrepreneur      0.0235024  0.0393961   0.597  0.55083
jobhousemaid         0.0404111  0.0479769   0.842  0.39967
jobmanagement       -0.0298658  0.0288106  -1.037  0.29997
jobretired           0.0530143  0.0297877   1.780  0.07520 .
jobself-employed     0.0419854  0.0403614   1.040  0.29829
jobservices          0.0046227  0.0267377   0.173  0.86275
jobstudent           0.0627833  0.0369714   1.698  0.08956 .
jobtechnician       -0.0040397  0.0210941  -0.192  0.84814
jobunemployed        0.0539618  0.0441653   1.222  0.22185
jobunknown           0.0040250  0.0745117   0.054  0.95692
defaultunknown      -0.0370676  0.0191456  -1.936  0.05293 .
contacttelephone    -0.0542838  0.0202921  -2.675  0.00750 **
monthaug             0.0213123  0.0319364   0.667  0.50460
monthdec             0.1256609  0.0763830   1.645  0.10002
monthjul             0.0525545  0.0318432   1.650  0.09894 .
monthjun             0.0354515  0.0319001   1.111  0.26649
monthmar             0.1960121  0.0443314   4.422 1.01e-05 ***
monthmay            -0.1372153  0.0278160  -4.933 8.43e-07 ***
monthnov            -0.0971096  0.0322683  -3.009  0.00263 **
monthoct            -0.0491267  0.0439255  -1.118  0.26346
monthsep            -0.0606407  0.0473729  -1.280  0.20059
campaign            -0.0080671  0.0027088  -2.978  0.00292 **
poutcomenonexistent  0.1203691  0.0234728   5.128 3.07e-07 ***
poutcomesuccess      0.2505528  0.0307041   8.160 4.44e-16 ***
emp.var.rate        -0.0311557  0.0095170  -3.274  0.00107 **
nr.employed         -0.0019628  0.0002024  -9.698  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.1768263)

    Null deviance: 991.81  on 3999  degrees of freedom
Residual deviance: 702.35  on 3972  degrees of freedom
AIC: 4451.1

Number of Fisher Scoring iterations: 2
```

Comparing the three models we have thus far

```
> anova(fit1,fit3)
Analysis of Deviance Table

Model 1: TD ~ 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
Model 2: TD ~ job + default + contact + month + campaign + poutcome + 
    emp.var.rate + nr.employed
  Resid. Df Resid. Dev  Df Deviance
1      3949     4149.5             
2      3972      702.4 -23   3447.1
> anova(fit2,fit3)
Analysis of Deviance Table

Model 1: TD ~ month + poutcome + emp.var.rate + cons.price.idx + loan
Model 2: TD ~ job + default + contact + month + campaign + poutcome + 
    emp.var.rate + nr.employed
  Resid. Df Resid. Dev Df Deviance
1      3984     4226.9            
2      3972      702.4 12   3524.6
```

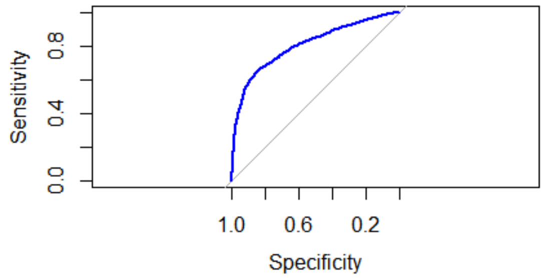
It can be seen that the deviance of model 3 from model 1 is lesser than its deviance from model 2. Hence Model 3 is the best so far amongst the three models.

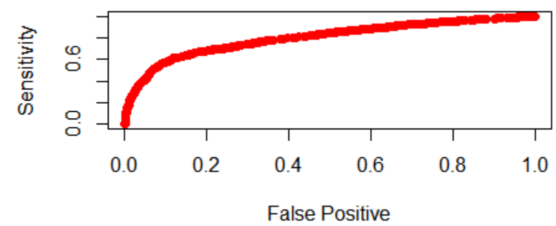
Thus the model (Model 3) we get is on performing logistic regression and retaining variables with significance level of *** and **:
TD = 10.48 + 0.196*monthmar – 0.1372*monthmay + 0.12037*poutcomenonexistent + 0.25055*poutcomesuccess - 0.00196*nr.employed – 0.543*contacttelephone – 0.097*monthnov – 0.0311*emp.var.rate – 0.0081*campaign

The MCE of this model on the test data is 0.377 and the confusion matrix is given below.

```
> fit3.mce.test
[1] 0.377
> cm

fit3.pred.test   0   1
             0 567 375
             1   2  56
```

The specificity is 0.996485, sensitivity is 0.12993 and the false positive value is 0.0035149. The ROC curve obtained is show below. The AUC value is 0.7999.





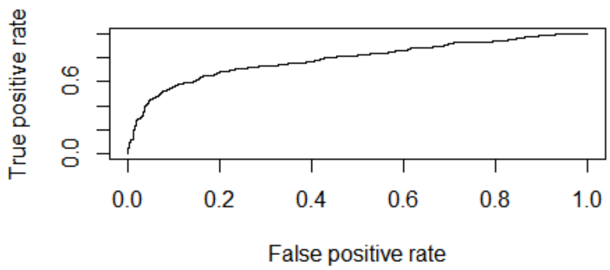
So far we have seen three models in which the training set constituted of 4000 data points and the testing set constitutes of 1000 data points. There was no Cross Validation done. The next three models have cross validation done in order to improve the training of the model and thereby reduce overfitting.

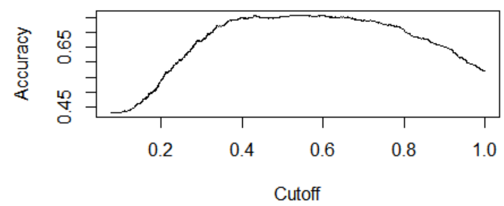
- Model 4:

Here I basically use the same model as Model 1 but with cross validation done. I do a 10 fold cross validation. The code snippet is as shown below.

```
ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions = TRUE)

fit4 <- train(TD ~.,data=data.cleaned.train, method="glm", family="binomial",
                 trControl = ctrl, tuneLength = 5) # Fit the model

summary(fit4)

pred = predict(fit4, data.cleaned.test) # predict on the testing data set

abc=prediction(pred,data.cleaned.test$TD)
AUC = as.numeric(performance(abc, "auc")@y.values)
ACC= performance(abc, "acc")
mean(ACC@y.values[[1]])
plot(performance(abc, 'tpr', 'fpr'))
plot(ACC)

Sensitivity= performance(abc, "sens")
mean(Sensitivity@y.values[[1]])
Specificity= performance(abc, "spec")
mean(Specificity@y.values[[1]])
```

The AUC value is 0.787322. The mean accuracy is 0.641 while the max accuracy obtained is 0.757.





Above I have the ROC curve and the accuracy curve which corresponds to the values of the accuracy reported before. The sensitivity and specificity are as follows,

```
> Sensitivity= performance(abc, "sens")
> mean(Sensitivity@y.values[[1]])
[1] 0.6628345
> Specificity= performance(abc, "spec")
> mean(Specificity@y.values[[1]])
[1] 0.6243924
```

Hence the MCE mean is 1-0.641=0.359. The model parameters are the same as that of Model 1. The summary is,

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1304  -0.8085  -0.5760   0.7846   2.1051

Coefficients: (1 not defined because of singularities)
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -9.155e+01  8.385e+01  -1.092 0.274905
age                           2.410e-03  4.602e-03   0.524 0.600445
`jobblue-collar`             -6.776e-02  1.427e-01  -0.475 0.634817
jobentrepreneur               1.753e-01  2.180e-01   0.804 0.421114
jobhousemaid                  3.188e-01  2.800e-01   1.139 0.254852
jobmanagement                -2.091e-01  1.701e-01  -1.229 0.218978
jobretired                    3.272e-01  2.146e-01   1.525 0.127352
`jobself-employed`            2.132e-01  2.213e-01   0.964 0.335204
jobservices                   1.317e-01  1.586e-01   0.830 0.406525
jobstudent                    5.058e-01  2.490e-01   2.032 0.042203 *
jobtechnician                 2.013e-02  1.343e-01   0.150 0.880823
jobunemployed                 3.735e-01  2.644e-01   1.413 0.157732
jobunknown                    6.782e-02  4.453e-01   0.152 0.878954
maritalmarried                7.602e-02  1.267e-01   0.600 0.548451
maritalsingle                 4.717e-02  1.447e-01   0.326 0.744447
maritalunknown                3.705e-01  8.224e-01   0.451 0.652348
educationbasic.6y             2.048e-01  2.078e-01   0.986 0.324335
educationbasic.9y             7.268e-02  1.734e-01   0.419 0.675081
educationhigh.school          8.216e-02  1.736e-01   0.473 0.635981
educationilliterate           1.112e+01  1.970e+02   0.056 0.954985
educationprofessional.course  1.567e-01  1.961e-01   0.799 0.424139
educationuniversity.degree    2.241e-01  1.755e-01   1.277 0.201709
educationunknown             -2.815e-02  2.304e-01  -0.122 0.902765
defaultunknown               -2.038e-01  1.114e-01  -1.830 0.067257 .
housingunknown               -5.376e-01  2.709e-01  -1.985 0.047142 *
housingyes                   -6.773e-03  7.813e-02  -0.087 0.930917
```

```
loanunknown                          NA         NA      NA       NA
loanyes                      -3.682e-03  1.077e-01  -0.034 0.972738
contacttelephone             -6.947e-01  1.487e-01  -4.672 2.98e-06 ***
monthaug                      5.028e-01  3.097e-01   1.624 0.104450
monthdec                      2.308e+00  1.178e+00   1.959 0.050131 .
monthjul                      1.975e-01  1.947e-01   1.014 0.310530
monthjun                     -3.328e-01  2.955e-01  -1.126 0.260020
monthmar                      1.965e+00  4.367e-01   4.501 6.76e-06 ***
monthmay                     -5.219e-01  1.697e-01  -3.076 0.002101 **
monthnov                     -7.347e-01  2.388e-01  -3.077 0.002091 **
monthoct                     -6.099e-01  3.355e-01  -1.818 0.069077 .
monthsep                     -2.606e-01  4.142e-01  -0.629 0.529312
day_of_weekmon               -3.191e-01  1.238e-01  -2.578 0.009951 **
day_of_weekthu               -3.628e-01  1.211e-01  -2.996 0.002736 **
day_of_weektue               -1.837e-01  1.250e-01  -1.470 0.141554
day_of_weekwed               -3.907e-02  1.225e-01  -0.319 0.749809
campaign                     -4.337e-02  1.728e-02  -2.510 0.012068 *
pdays                        -3.448e-04  4.639e-04  -0.743 0.457300
previous                     -2.414e-01  1.602e-01  -1.507 0.131824
poutcomenonexistent           3.238e-01  2.246e-01   1.442 0.149282
poutcomesuccess               1.590e+00  4.628e-01   3.436 0.000591 ***
emp.var.rate                 -1.220e+00  3.348e-01  -3.644 0.000269 ***
cons.price.idx                1.287e+00  5.617e-01   2.291 0.021981 *
cons.conf.idx                 1.907e-03  1.910e-02   0.100 0.920492
euribor3m                     6.010e-01  2.827e-01   2.126 0.033489 *
nr.employed                  -5.994e-03  6.709e-03  -0.894 0.371589
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5512.4  on 3999  degrees of freedom
Residual deviance: 4149.5  on 3949  degrees of freedom
AIC: 4251.5

Number of Fisher Scoring iterations: 10
```

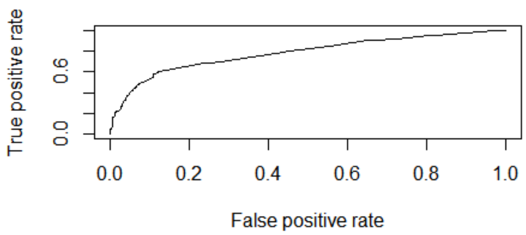
TD = - 9.155e+01 + sum across all variables (Corresponding Coeff * Correspoding Variable)

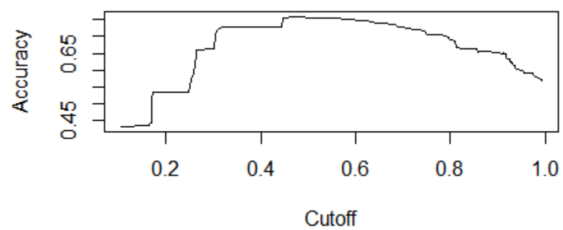
- <u>Model 5:</u>

Here I basically use the same model as Model 2 but with cross validation done. I do a 10 fold cross validation. The predictor variables that are chosen here are the same as that of Model 2.

The AUC value is 0.781377. The mean accuracy is 0.65098 while the max accuracy obtained is 0.76.

Below I have the ROC curve and the accuracy curve which corresponds to the values of the accuracy reported before. The sensitivity and specificity are 0.40627 and 0.83634 respectively.





Hence the MCE mean is 1-0.65098=0.34902. The model parameters are the same as that of Model 2. The summary is,

```
Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9327  -0.7809  -0.6074   0.8085   2.1173

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -113.44041   14.09008  -8.051 8.21e-16 ***
monthaug               0.91670    0.19455   4.712 2.45e-06 ***
monthdec               2.37249    1.05456   2.250  0.02447 *
monthjul               0.55310    0.17055   3.243  0.00118 **
monthjun              -0.36521    0.17451  -2.093  0.03637 *
monthmar               2.12738    0.38864   5.474 4.40e-08 ***
monthmay              -0.57024    0.14202  -4.015 5.94e-05 ***
monthnov              -0.17439    0.17937  -0.972  0.33091
monthoct               0.04623    0.25262   0.183  0.85479
monthsep               0.23474    0.28783   0.816  0.41475
poutcomenonexistent    0.58904    0.12727   4.628 3.69e-06 ***
poutcomesuccess        1.83650    0.22553   8.143 3.85e-16 ***
emp.var.rate          -0.92714    0.05725 -16.196  < 2e-16 ***
cons.price.idx         1.20055    0.15019   7.994 1.31e-15 ***
loanunknown           -0.53236    0.26141  -2.036  0.04170 *
loanyes                0.01664    0.10554   0.158  0.87473
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5512.4  on 3999  degrees of freedom
Residual deviance: 4226.9  on 3984  degrees of freedom
AIC: 4258.9

Number of Fisher Scoring iterations: 6
```

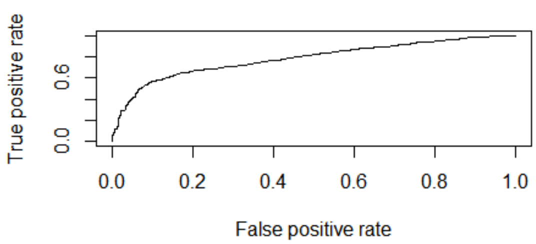
TD = -113.44 + 0.9167*monthaug + 2.127*monthmar - 0.57*monthmay + 0.589*poutcomenonexistent + 1.8365*poutcomesuccess - 0.927*emp.var.rate + 1.2*cons.price.idx

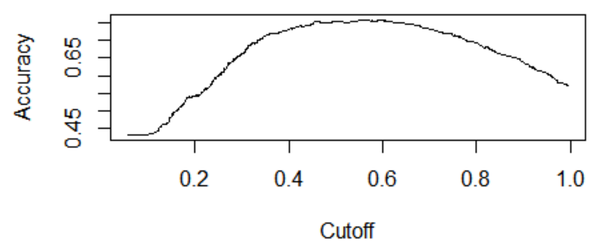
- <u>Model 6:</u>

Here I basically use the same model as Model 3 but with cross validation done. I do a 10 fold cross validation. The predictor variables that are chosen here are the same as that of Model 3.

The AUC value is 0.7834. The mean accuracy is 0.63986 while the max accuracy obtained is 0.759.

Below I have the ROC curve and the accuracy curve which corresponds to the values of the accuracy reported before. The sensitivity and specificity are 0.6136 and 0.6597 respectively.





Hence the MCE mean is 1-0.63986=0.36014. The model parameters are the same as that of Model 3. The summary is,

```
Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1198  -0.8229  -0.5600   0.8025   2.2608

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         55.372751   7.147319   7.747 9.38e-15 ***
`jobblue-collar`    -0.175614   0.115120  -1.525  0.12714
jobentrepreneur      0.145058   0.213286   0.680  0.49644
jobhousemaid         0.238495   0.267679   0.891  0.37294
jobmanagement       -0.176864   0.164681  -1.074  0.28283
jobretired           0.318439   0.181997   1.750  0.08017 .
`jobself-employed`   0.219884   0.216805   1.014  0.31049
jobservices          0.035977   0.148155   0.243  0.80814
jobstudent           0.378868   0.230505   1.644  0.10025
jobtechnician       -0.027703   0.119410  -0.232  0.81654
jobunemployed        0.290445   0.256341   1.133  0.25720
jobunknown           0.042285   0.424649   0.100  0.92068
defaultunknown      -0.196680   0.109059  -1.803  0.07132 .
contacttelephone    -0.368349   0.117266  -3.141  0.00168 **
monthaug             0.189735   0.174577   1.087  0.27711
monthdec             2.114277   1.049108   2.015  0.04387 *
monthjul             0.371146   0.175015   2.121  0.03395 *
monthjun             0.318958   0.177848   1.793  0.07290 .
monthmar             1.738184   0.390289   4.454 8.45e-06 ***
monthmay            -0.607984   0.143953  -4.223 2.41e-05 ***
monthnov            -0.352807   0.176442  -2.000  0.04555 *
monthoct            -0.174054   0.268289  -0.649  0.51650
monthsep            -0.338307   0.304167  -1.112  0.26603
campaign            -0.050453   0.017453  -2.891  0.00384 **
poutcomenonexistent  0.596863   0.127271   4.690 2.74e-06 ***
poutcomesuccess      1.826423   0.225687   8.093 5.84e-16 ***
emp.var.rate        -0.055913   0.062098  -0.900  0.36790
nr.employed         -0.010871   0.001387  -7.835 4.69e-15 ***
```

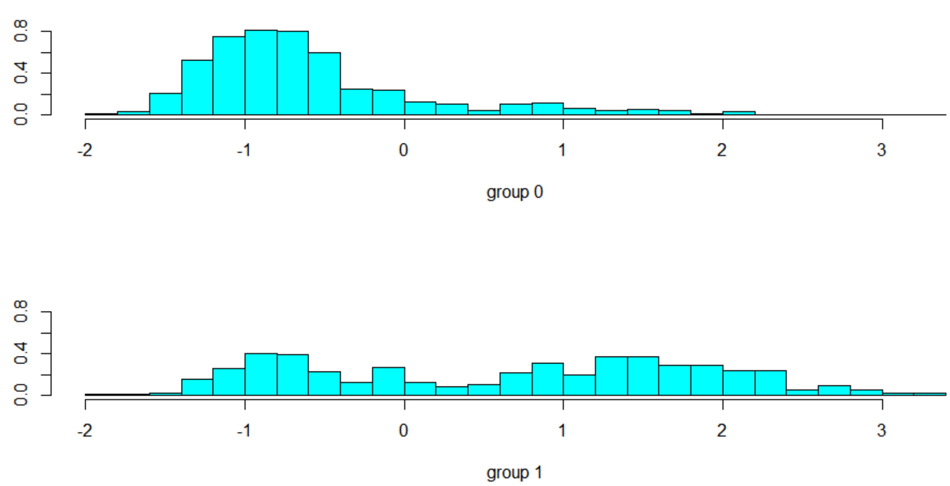
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5512.4  on 3999  degrees of freedom
Residual deviance: 4194.9  on 3972  degrees of freedom
AIC: 4250.9

Number of Fisher Scoring iterations: 6
```

TD = 10.48 + 0.196*monthmar - 0.1372*monthmay + 0.12037*poutcomenonexistent + 0.25055*poutcomesuccess - 0.00196*nr.employed - 0.543*contacttelephone - 0.097*monthnov - 0.0311*emp.var.rate - 0.0081*campaign

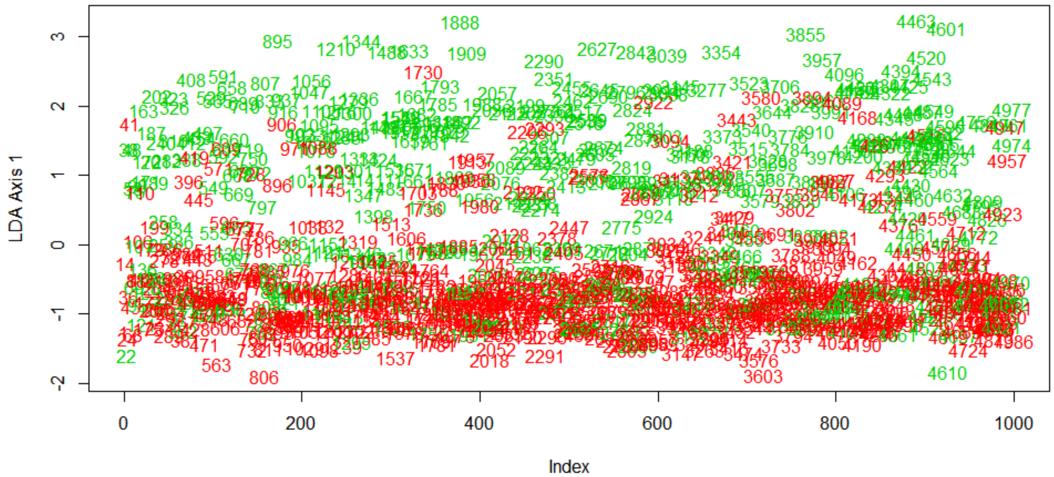
- Model 7:

In this method I use Linear Discriminant Analysis (LDA) which is a dimensionality reduction method in supervised learning. Here I use all the variables. Since the response variable has only 2 levels –yes or no, we have only one principal axis (LD1).

The histogram of the discriminant functions is shown below.



The scatter plot of the discriminant function values is,



As can be seen we can separate the data along LD1 fairly well. The confusion matrix post prediction on test data is obtained as,

```
> ct

     0   1
  0 497  72
  1 174 257
```

The MCE is 0.754. Clearly, this method does not do as well with high miss-classifications. The group means obtained are as follows,

```
Call:
lda(data.cleaned.train$TD ~ ., data = data.cleaned.train)

Prior probabilities of groups:
      0       1 
0.54525 0.45475 

Group means:
       age jobblue-collar jobentrepreneur jobhousemaid jobmanagement jobretired
0 39.95002      0.2498854      0.03530491   0.01879872    0.07336084 0.03805594
1 40.78615      0.1401869      0.02858714   0.02308961    0.06102254 0.09455745
  jobself-employed jobservices jobstudent jobtechnician jobunemployed  jobunknown
0       0.03026135  0.08986703 0.01834021     0.1659789    0.01879872 0.008711600
1       0.03078615  0.07421660 0.06157229     0.1550302    0.03243540 0.007696537
  maritalmarried maritalsingle maritalunknown educationbasic.6y educationbasic.9y
0      0.5951398     0.2879413    0.001834021        0.06189821         0.1485557
1      0.5327103     0.3567894    0.003298516        0.04398021         0.1028037
  educationhigh.school educationilliterate educationprofessional.course
0            0.2251261        0.0000000000                    0.1182944
1            0.2221001        0.0005497526                    0.1242441
  educationuniversity.degree educationunknown defaultunknown housingunknown
0                  0.2929849       0.04951857      0.2278771     0.02475928
1                  0.3628367       0.05552501      0.0951072     0.02308961
  housingyes loanunknown    loanyes contacttelephone  monthaug     monthdec
0  0.5066483  0.02475928 0.1531408        0.3915635 0.1480972 0.0004585053
1  0.5305113  0.02308961 0.1473337        0.1605278 0.1330401 0.0181418362
   monthjul  monthjun    monthmar  monthmay   monthnov   monthoct   monthsep
0 0.1838606 0.1325080 0.003668042 0.3452545 0.11095828 0.01329665 0.00871160
1 0.1456844 0.1198461 0.066520066 0.1940627 0.09015943 0.05992303 0.05717427
  day_of_weekmon day_of_weekthu day_of_weektue day_of_weekwed campaign    pdays
0      0.2090784      0.2310867      0.1856946      0.1934892 2.678588 980.3787
1      0.1852666      0.2210005      0.2078065      0.1962617 2.097306 781.7081
   previous poutcomenonexistent poutcomesuccess emp.var.rate cons.price.idx
0 0.1361761           0.8890417      0.01421366    0.2748739       93.60807
1 0.4815833           0.6734469      0.20340847   -1.2249038       93.34924
  cons.conf.idx euribor3m nr.employed
0     -40.58588  3.848534    5178.006
1     -39.92216  2.125677    5095.732
```

The LD1 values are as follows,

```
Coefficients of linear discriminants:
                                       LD1
age                           0.0014563363
jobblue-collar               -0.0786599250
jobentrepreneur               0.1249738089
jobhousemaid                  0.2227055772
jobmanagement                -0.1548487714
jobretired                    0.2249341146
jobself-employed              0.1708972542
jobservices                   0.0912037336
jobstudent                    0.3488597093
jobtechnician                 0.0182163636
jobunemployed                 0.2909971100
jobunknown                    0.0231406650
maritalmarried                0.0709444450
maritalsingle                 0.0395675802
maritalunknown                0.1930393227
educationbasic.6y             0.1569858453
educationbasic.9y             0.0301274685
educationhigh.school          0.0442722629
educationilliterate           1.6604400272
educationprofessional.course  0.0873517523
educationuniversity.degree    0.1487739834
educationunknown             -0.0052079733
defaultunknown               -0.1696857763
housingunknown               -0.1774168404
housingyes                   -0.0030956961
loanunknown                  -0.1774168404
loanyes                      -0.0053904219
contacttelephone             -0.4997818875
monthaug                      0.2303522985
monthdec                      0.3467920818
monthjul                      0.0370097776
monthjun                     -0.2965808143
monthmar                      1.0114766515
monthmay                     -0.6080989995
monthnov                     -0.8133743847
monthoct                     -0.6087753863
monthsep                     -0.3008753077
day_of_weekmon               -0.2437638515
day_of_weekthu               -0.2710833401
day_of_weektue               -0.1399335159
day_of_weekwed               -0.0293803490
campaign                     -0.0311724847
pdays                        -0.0003152017
previous                     -0.1139838991
poutcomenonexistent           0.3787037055
poutcomesuccess               0.8143806286
emp.var.rate                 -0.9846632716
cons.price.idx                0.8031477149
cons.conf.idx                -0.0048874518
euribor3m                     0.5931692381
nr.employed                  -0.0076966992
```

Hence the model we get is

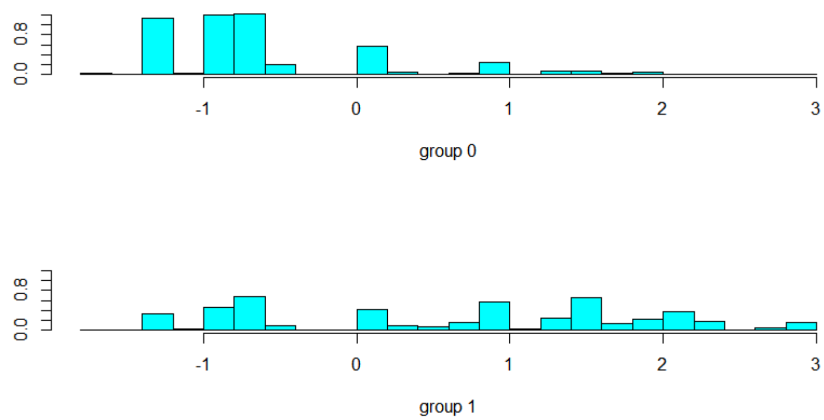
TD= sum across all the variables (Variable * Corresponding LD1)

Source: http://www.talkstats.com/showthread.php/39958-discriminant-linear-analysis-coefficient-interpretation

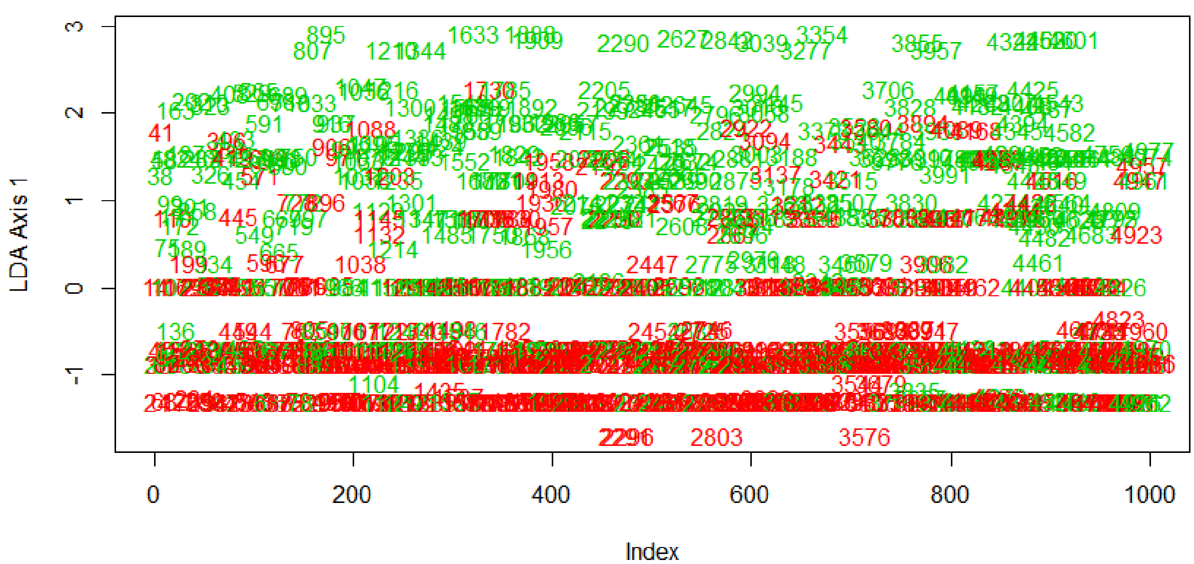
Model 8:

In this method I use Linear Discriminant Analysis (LDA) which is a dimensionality reduction method in supervised learning. Here I use only those variables I had gotten from Regsubsets. These are the same variables used in Model 2 and Model 5. Since the response variable has only 2 levels –yes or no, we have only one principal axis (LD1).

The histogram of the discriminant functions is shown below.



The scatter plot of the discriminant function values is,



The data separation in this case along LD1 is not as well as the previous case. The confusion matrix post prediction on test data is obtained as,

```
> ct

     0   1
  0 501  68
  1 174 257
```

The MCE is 0.758. Clearly, this method does not do as well with high miss-classifications just like the above.

The LD1 values are as follows,

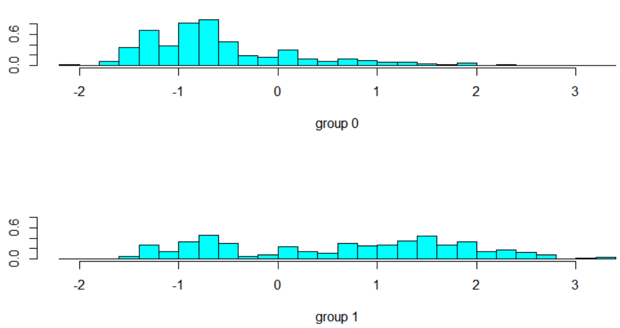
```
Coefficients of linear discriminants:
                             LD1
monthaug             0.608473654
monthdec             0.765063034
monthjul             0.318761707
monthjun            -0.365859861
monthmar             1.231861485
monthmay            -0.607254726
monthnov            -0.317856628
monthoct            -0.008343567
monthsep             0.206084596
poutcomenonexistent  0.518893802
poutcomesuccess      1.119683043
emp.var.rate        -0.825092988
cons.price.idx       0.959560411
loanunknown         -0.382976615
loanyes              0.016374598
```

TD= sum across all the variables above (Variable * Corresponding LD1)

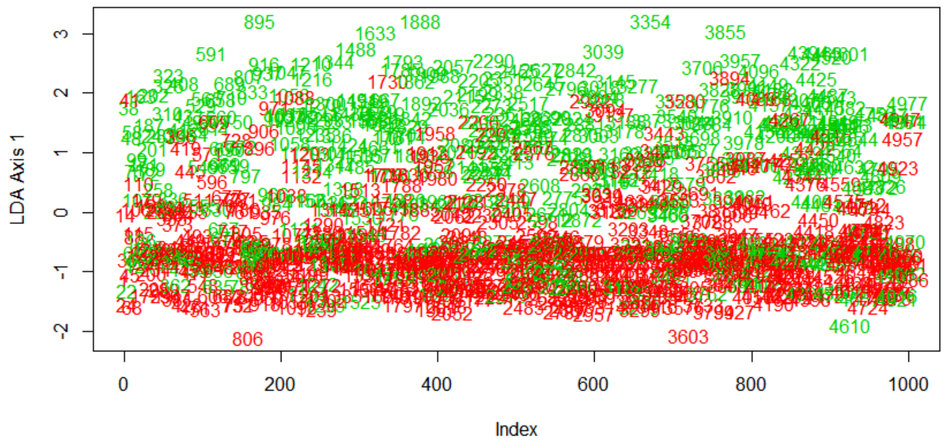
- Model 9:

In this method I use Linear Discriminant Analysis (LDA) which is a dimensionality reduction method in supervised learning. Here I use only those variables I had gotten from LASSO regularization. These are the same variables used in Model 3 and Model 6. Since the response variable has only 2 levels –yes or no, we have only one principal axis (LD1).

The histogram of the discriminant functions is shown below.



The scatter plot of the discriminant function values is,



The data separation in this case along LD1 is decent with still some values abruptly present farther from the cluster. The confusion matrix post prediction on test data is obtained as,

```
> ct

    0   1
0 489  80
1 167 264
```

The MCE is 0.753. Clearly, this method does not do as well with high miss-classifications just like the above. The LD1 values are as follows,

```
Coefficients of linear discriminants:
                              LD1
jobblue-collar       -0.157323455
jobentrepreneur       0.103795632
jobhousemaid          0.178470710
jobmanagement        -0.131898592
jobretired            0.234131238
jobself-employed      0.185423511
jobservices           0.020415625
jobstudent            0.277274786
jobtechnician        -0.017840816
jobunemployed         0.238315892
jobunknown            0.017776040
defaultunknown       -0.163704460
contacttelephone     -0.239737792
monthaug              0.094123028
monthdec              0.554966226
monthjul              0.232100501
monthjun              0.156567178
monthmar              0.865664120
monthmay             -0.605994817
monthnov             -0.428872814
monthoct             -0.216962015
monthsep             -0.267812237
campaign             -0.035627585
poutcomenonexistent   0.531595828
poutcomesuccess       1.106536423
emp.var.rate         -0.137595411
nr.employed          -0.008668464
```

TD= sum across all the variables above (Variable * Corresponding LD1)

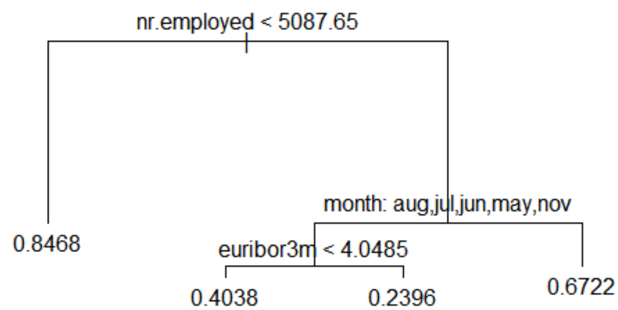
What can be summarized out of the last 3 models that we obtain from LDA is that the MCE has significantly higher as compared to logistic regressions. The models don't do well to classify the dataset.

Finally, I explore the Tree methods along with Random Forest to generate 3 more models.

- Model 10:

This is a tree based approach on all the variables that are there in the dataset. The frame showing the splits and the tree itself is shown below.

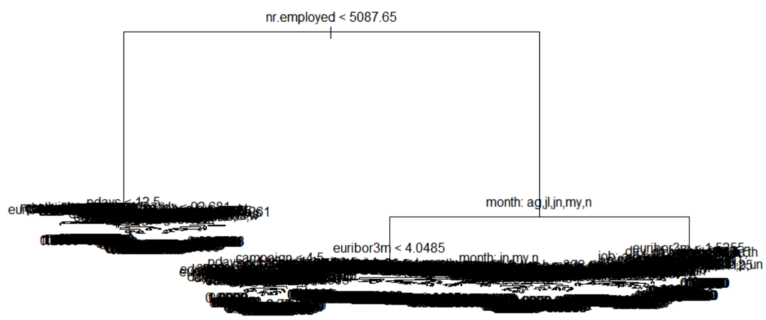
```
> fit10.1$frame
           var    n       dev      yval splits.cutleft splits.cutright
1  nr.employed 4000 991.80975 0.4547500       <5087.65        >5087.65
2       <leaf> 1005 130.40199 0.8467662
3        month 2995 655.13723 0.3232053         :bdegh           :acfi
6    euribor3m 2635 525.97116 0.2755218        <4.0485         >4.0485
12      <leaf>  577 138.91161 0.4038128
13      <leaf> 2058 374.90039 0.2395530
7       <leaf>  360  79.32222 0.6722222
```



There are 3 variables that the tree is split on and it leads to 4 leaves. Basically they are the mean values in the 4 regions the entire data set get divided into. The summary of this tree is given below,

```
Regression tree:
tree(formula = TD ~ ., data = data.cleaned.train)
Variables actually used in tree construction:
[1] "nr.employed" "month"       "euribor3m"
Number of terminal nodes:  4
Residual mean deviance:  0.1811 = 723.5 / 3996
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.8468 -0.2396 -0.2396  0.0000  0.1532  0.7604
```

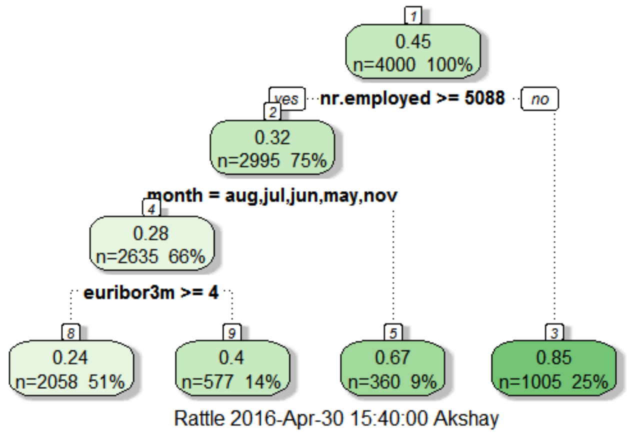
The default split is on deviance. The other split is on Gini. The tree obtained from that is,



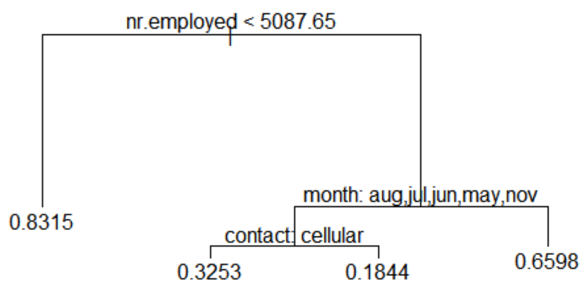
The summary of this tree is as shown below,

```
Regression tree:
tree(formula = TD ~ ., data = data.cleaned.train, split = "gini")
Variables actually used in tree construction:
 [1] "nr.employed"    "pdays"          "job"            "month"
 [5] "euribor3m"      "age"            "campaign"       "cons.price.idx"
 [9] "day_of_week"    "marital"        "previous"       "housing"
[13] "education"      "emp.var.rate"   "contact"        "poutcome"
[17] "default"        "loan"
Number of terminal nodes:  631
Residual mean deviance:  0.1133 = 381.6 / 3369
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.8889 -0.1250  0.0000  0.0000  0.0000  0.8889
```

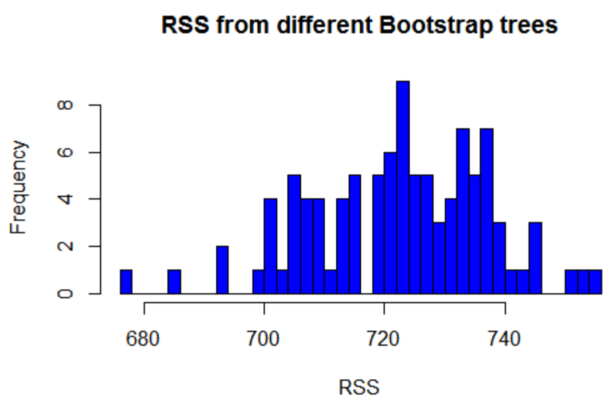
As can be seen in the gini split, the number of tree leaves are far more and also the residual mean deviance is lesser than when the split occurs on deviance.
A fancier version of the plot is shown below. Here the number of observation in a particular node and its accuracy is shown.



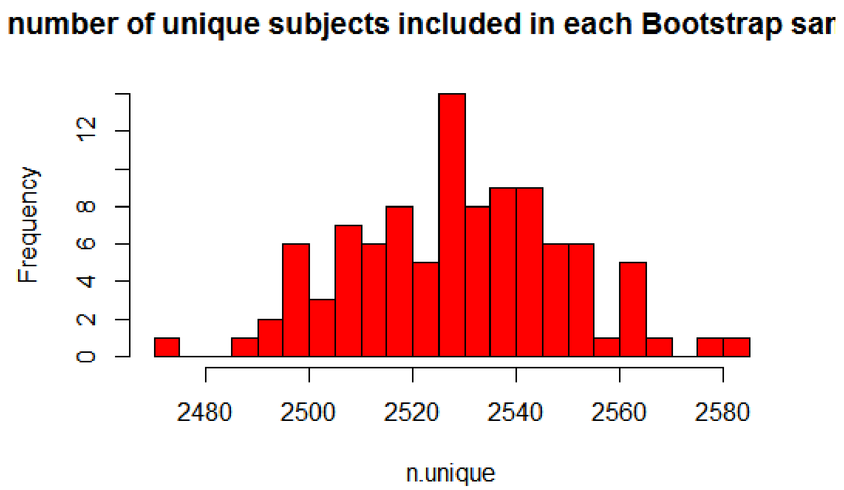
The RSS that has been computed is723.5 for the default split. A quick insight into the RSS of the logistic model shows that for the same variables the deviance is much larger. Next I performed Bootstrap to get more sampled dataset and get better trees. I get 100 trees, one for each bootstrap sample. One such tree is given below.



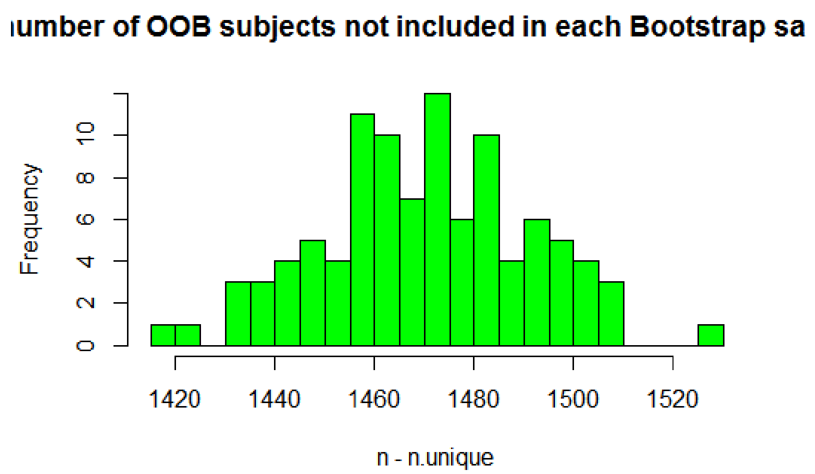
A plot of the RSS that we get across the Bootstrap samples is shown by the below histogram.



Here it can be seen that the RSS varies from 680 to 740.
The number of unique data points in each bootstrap sample is shown in the below histogram.

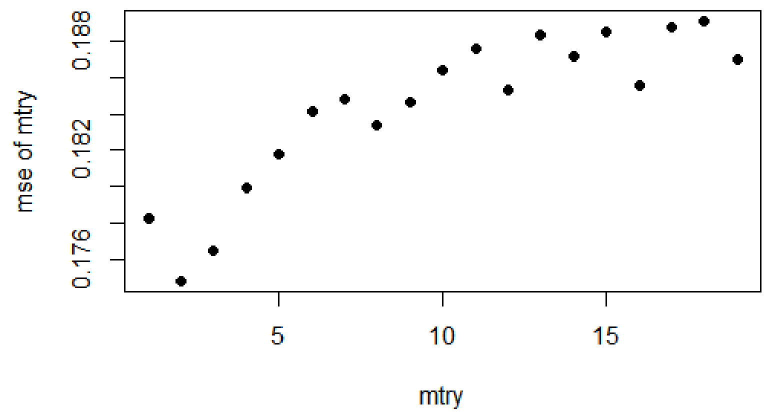


This shows that anywhere between 2480 and 2580 samples are unique out of 4000, while the rest are repeated.
Number of Out of Bag (OOB) samples not included in each Bootstrap sample is shown below.

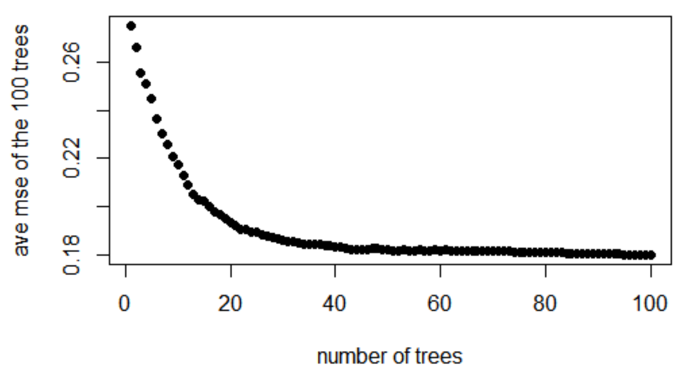


Basically while bootstrapping two thirds of the observations are used to make the tree. The rest one third are out of bag samples. The response of the i'th observation in the OOB was predicted using each of the tree in which that observation was OOB. This meant that I got roughly around B/3 responses for each observation, where B is the number of Bootstrap sample and then I averaged those responses to be able to finally obtain one response for each observation. This entire process is called bagging.

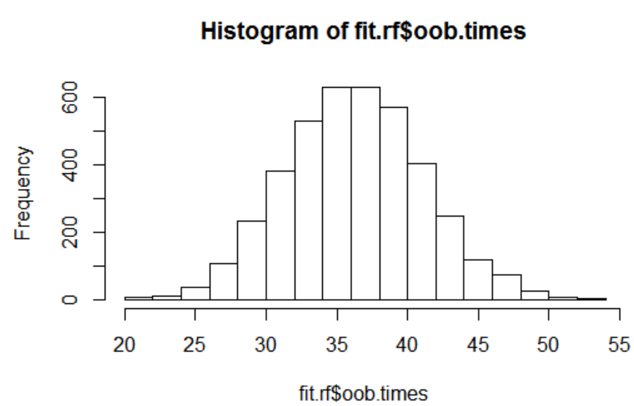
Additionally to improve performance, I built random forest on top of each bootstrapped sample. The effect of mtry parameter, which is the number of random split at each leaf can be seen below.



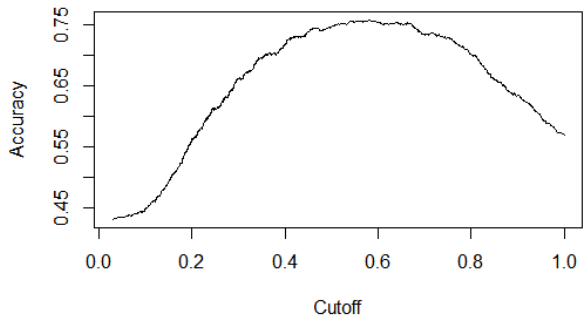
As mtry increases the MSE also increases. Hence I used a value like mtry =4 (square root of 20 roughly) to obtain the Random Forest. The average MSE that I get for 100 trees is shown. Clearly, more the number of trees, less is the MSE.

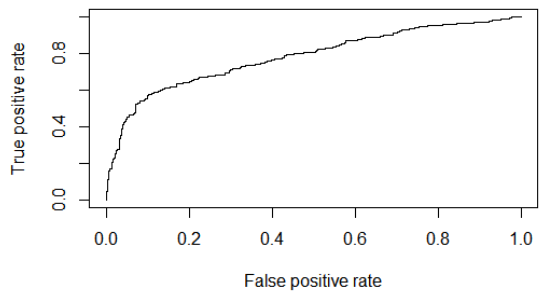


The histogram below how many times each observation is OOB. Most of the observations are OOB roughly 30-40% times.



The training error I obtain is 0.18 (18%). This means a mean accuracy of 82% on the training dataset (OOB accuracy). Upon prediction using the test data, I got the mean test accuracy of 63.81% and the max test accuracy to be 75.8%. The accuracy curve on the test data is given below. The ROC curve is also shown.

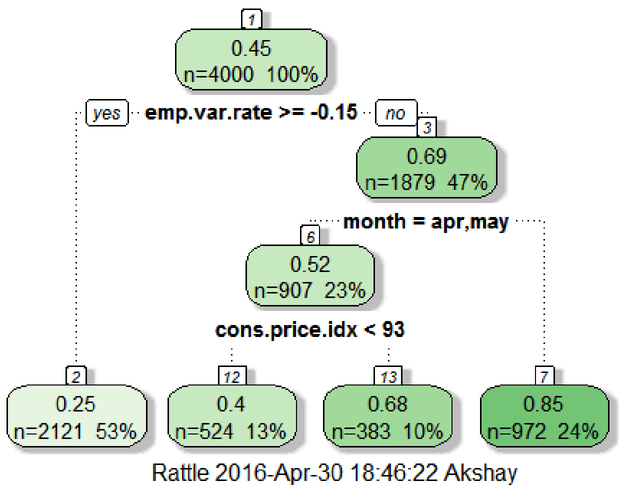




The mean sensitivity which is the probability of a point being classified as positive when it is indeed positive is 0.6626 while the specificity which is the probability of a point being classified as negative when it is indeed negative is 0.6196.

- <u>Model 11:</u>

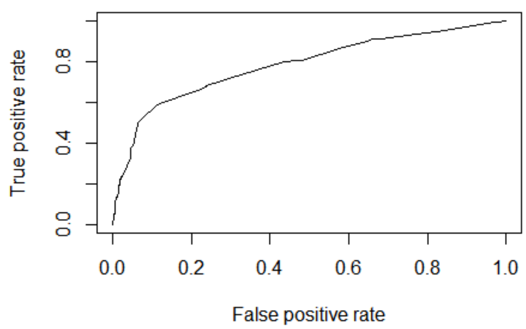
This is a tree based approach on all the variables that I got from regsubsets. The frame showing the splits and the tree itself is shown below. Note I use the default deviance split here on.

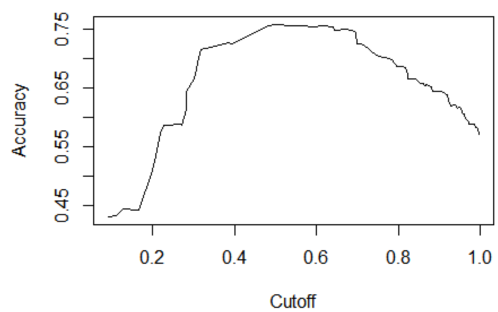


```
> fit.tree$frame
              var    n   wt       dev      yval  complexity ncompete nsurrogate
1    emp.var.rate 4000 4000 991.80975 0.4547500 0.197276010        4          4
2          <leaf> 2121 2121 394.03772 0.2465818 0.005774646        0          0
3           month 1879 1879 402.11176 0.6897286 0.050031305        4          3
6  cons.price.idx  907  907 226.33076 0.5214994 0.017666347        3          3
12         <leaf>  524  524 126.03626 0.4026718 0.004524817        0          0
13         <leaf>  383  383  82.77285 0.6840731 0.006217674        0          0
7          <leaf>  972  972 126.15947 0.8467078 0.004380722        0          0
>
```

Like the previous model, I use bootstrap to improve the model performance. Additionally to improve performance, I built random forest on top of each bootstrapped sample. As mtry increases the MSE also increases. Hence I used a value like mtry =2 (square root of 5 roughly) to obtain the Random Forest. Also I build 500 trees for a good result.

The training error I obtain is 0.17738 (17.738%). This means a mean accuracy of over 82% on the training dataset (OOB accuracy). Upon prediction using the test data, I got the mean test accuracy of 64.85377% and the max test accuracy to be 75.8%. The accuracy curve on the test data is given below. The ROC curve is also shown.

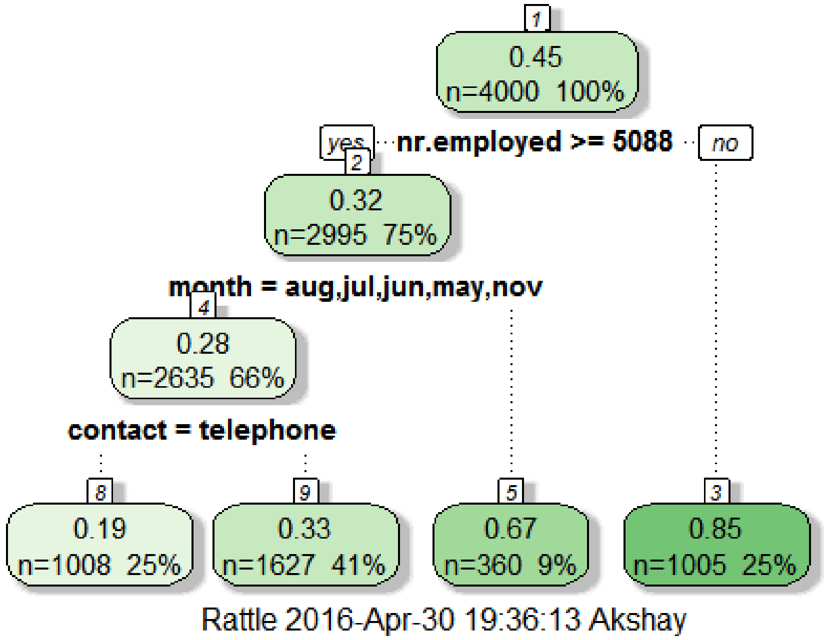




The mean sensitivity which is the probability of a point being classified as positive when it is indeed positive is 0.395329 while the specificity which is the probability of a point being classified as negative when it is indeed negative is 0.8403356.

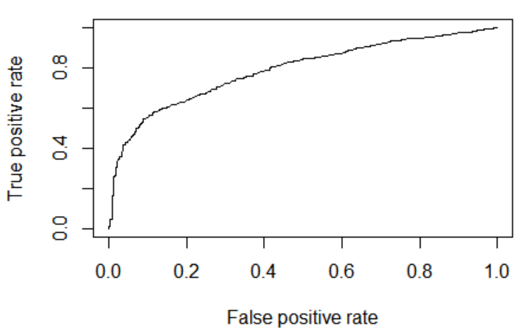
- Model 12:

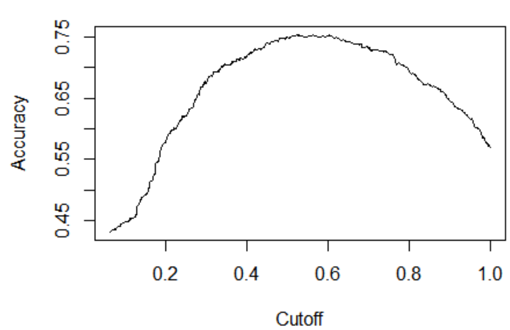
This is a tree based approach on all the variables that I got from LASSO regularization. The frame showing the splits and the tree itself is shown below. The trees have been split with the default deviance parameter.



Like the previous model, I use bootstrap to improve the model performance. Additionally to improve performance, I built random forest on top of each bootstrapped sample. As mtry increases the MSE also increases. Hence I used a value like mtry =3 (square root of 8 roughly) to obtain the Random Forest. Also I build 500 trees for a good result.

The training error I obtain is 0.18154 (18.154%). This means a mean accuracy of over 81% on the training dataset (OOB accuracy). Upon prediction using the test data, I got the mean test accuracy of 64.21414% and the max test accuracy to be 75.4%. The accuracy curve on the test data is given below. The ROC curve is also shown.





The mean sensitivity which is the probability of a point being classified as positive when it is indeed positive is 0.6146369 while the specificity which is the probability of a point being classified as negative when it is indeed negative is 0.6629753.

- Ensemble Model:

Having obtained all the above models, I ensemble them to boost the performance of the classifier. First I obtained all the predictions made by the models on the train data set. From the above analysis it was clear that cross validation did not provide much difference in test scores from what was reported in the respective first three models and hence it is not considered here to build the ensemble model.

Using the 9 models, I created a common data frame and obtained the mean values and classified them as 0 if it was lesser than 0.5 or 1 if it was greater. This way I got the training accuracy to be around 69%.

Lastly, I built a linear regression model on top of all the models considering these models as predictors. This helped me weigh the models.

```
> summary(fitfinal2)

Call:
lm(formula = data.cleaned.train.TD ~ TD.Label.Predicted1 + TD.Label.Predicted7 +
    TD.Label.Predicted8 + TD.Label.Predicted9 + TD.Label.Predicted10 +
    TD.Label.Predicted11 + TD.Label.Predicted12, data = df.main)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7235 -0.2646 -0.2646  0.3659  0.7955

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           0.264652   0.008151  32.469  < 2e-16 ***
TD.Label.Predicted1  -0.091333   0.038452  -2.375   0.0176 *
TD.Label.Predicted7   0.310055   0.064151   4.833 1.39e-06 ***
TD.Label.Predicted8  -0.060157   0.055673  -1.081   0.2800
TD.Label.Predicted9   0.119605   0.045003   2.658   0.0079 **
TD.Label.Predicted10  0.294018   0.030414   9.667  < 2e-16 ***
TD.Label.Predicted11  0.036085   0.038891   0.928   0.3535
TD.Label.Predicted12  0.144576   0.033543   4.310 1.67e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4114 on 3992 degrees of freedom
Multiple R-squared:  0.3187,    Adjusted R-squared:  0.3175
F-statistic: 266.8 on 7 and 3992 DF,  p-value: < 2.2e-16
```

FINAL MODEL FINDINGS AND SUMMARY

To summarize, the entire study began the idea to create multiple models and compare them and at the end ensemble them. Data was initially cleaned and subsetted for training and testing purposes. A summary of the model is as follows:

Model Number	Model Type	Test MCE
1	Logistic Regression on all variables	0.348
2	Logistic Regression on variables selected from Backward Selection of Regsubsets	0.346
3	Logistic Regression after feature selection by LASSO	0.377
4	Logistic Regression on all variables along with 10 fold Cross Validation	0.359
5	Logistic Regression on variables selected from Backward Selection of Regsubsets along with 10 fold Cross Validation	0.349
6	Logistic Regression after feature selection by LASSO along with 10 fold Cross Validation	0.360
7	LDA on all variables	0.754
8	LDA on variables selected from Backward Selection of Regsubsets	0.758
9	LDA on variables left after feature selection by LASSO	0.753
10	Random Forest after bagging and bootsrapping on all variables	0.362
11	Random Forest after bagging and bootsrapping on variables selected from Backward Selection of Regsubsets	0.351
12	Random Forest after bagging and bootsrapping on variables left after feature selection by LASSO	0.358

The variables that get selected from Regsubsets are:

month + poutcome + emp.var.rate + cons.price.idx + loan

The variables that get selected from LASSO are:

Job + default + contact + month + campaign + poutcome + emp.var.rate +nr.employed

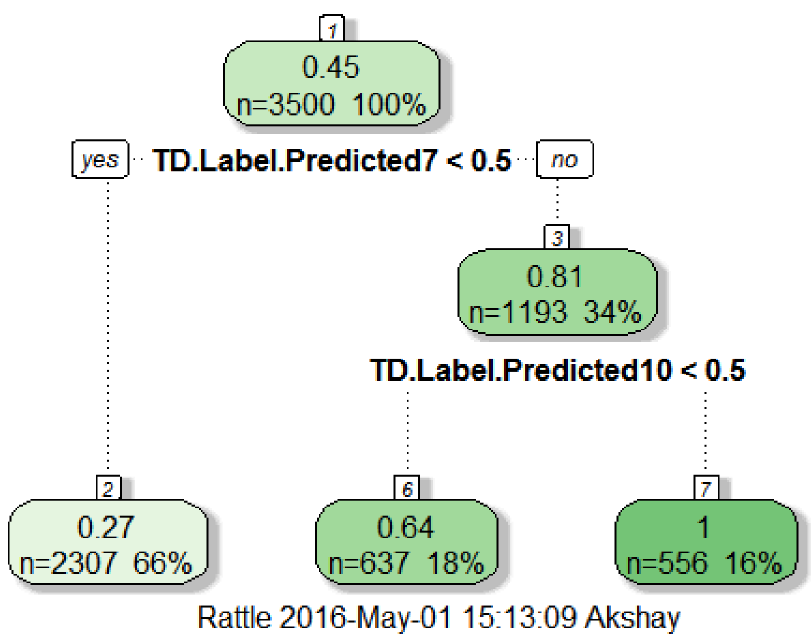
The above is the mean MCE. The least MCE (Max Accuracy) computed is around 75% for the test data set and 82% for the training data set. LDA doesn't perform well as compared to Logistic Regression. Cross Validation also surprisingly does not go a long way in improving accuracy. Random Forest performs the best. Some trees do really well but on an average the accuracy is comparable to those of logistic regression.

I then took the 9 models barring the cross validated ones and ensembled them in three ways. First by taking mean of the predictions made on test data and then classifying them and the other was to fit a linear regression model on the 7 models itself to weigh the models for classifications. For this I made a data frame consisting of predictions of all the 7 models and the response variable and fit a linear model. The data frame is made of 0's and 1's. A coupe of the models were correlated and hence those are removed and the fit is updated. Upon regression, if the response obtained is <0.5 then the classification would be 0 and if more than it would be 1.

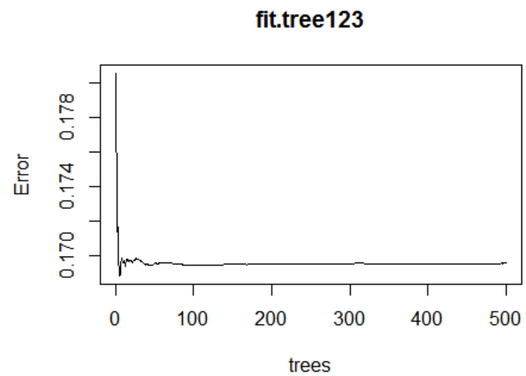
Data.cleaned.train.TD = 0.264652 – 0.091333*TD.Label.Predicted1 + 0.31* TD.Label.Predicted7 – 0.060157* TD.Label.Predicted8 + 0.1196* TD.Label.Predicted9 + 0.294* TD.Label.Predicted10 + 0.0361* TD.Label.Predicted11 + 0.144576* TD.Label.Predicted12

So then the final model!!

Random Forest did the best of all the models. And hence I used the prediction data frame and performed a Random Forest on them. I again divided the data frame into training and testing and performed my analysis. I then fit a random forest tree (fit.tree123 in code) on my training data set shown below,



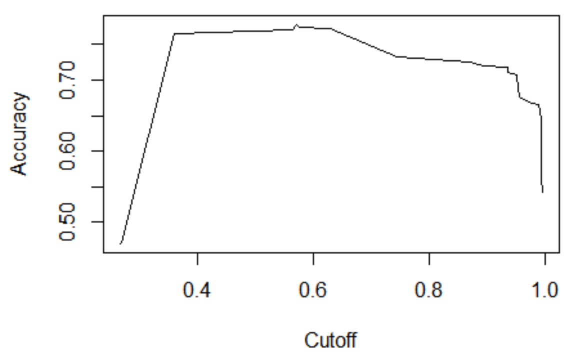
The plot of the final tree is as follows:

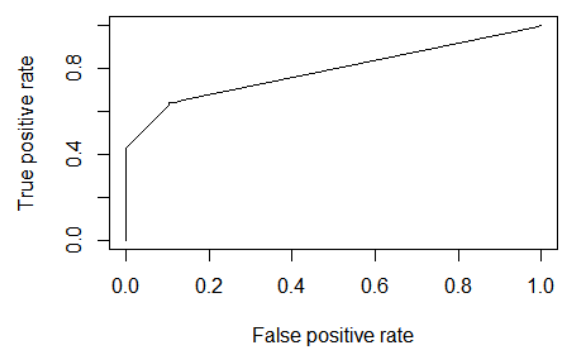


I do the prediction on the test data and the confusion matrix I get is as shown,

```
> cm

try5   0   1
   0 234  84
   1  31 151
```

The test MCE that I finally get is 0.23!!!! This means an accuracy of 77% on the test data which is far more than the accuracy we had from the individual models. The max accuracy reported is 77.8%. The roc curve and the accuracy curve is as shown below. The mean sensitivity is 0.4487 and the mean specificity is 0.879





What I also noticed when building these models is that some variables were consistently part of it and had significance level in the order of ***. These are:

month, poutcome, emp.var.rate and cons.price.idx

Even among the levels of the month, March and May were of great significance. These make sense because essentially these months constitute the end of Quarter 1 and midway Quarter 2. The primary reason behind a lot of customers opening the term deposit with the bank during these months could be because of reasons like the onset of spring, to the schooling season of Portugal is midway and parents have to make savings to pay for the schooling of the next year.

Also most government holidays in Portugal are in December. Opening a half yearly term deposit is a good investment option to quickly save money to splurge on the holiday season. 81% of the population in Portugal is Roman Catholic. Expenditures during Christmas and the holiday season could be a lot. Hence could be the reason. Also, historically, the banks have done well in the first two quarters in Portugal and this might be another reason.

Poutcome is an important predictor. This is because if a client has not responded positively in the previous marketing campaigns it is highly likely that he would not do so in the future. It is usual that an average person would gets calls and emails from various sources and on an average people unsubscribe/subscribe to such notifications out of sheer disinterest/interest for them. Marketing calls similarly are prone to such response from the customer. Hence naturally it's a strong predictor. If a person in the past has responded favorably, is it highly probable that he would lend an ear to such calls in the future as well.

Em.var.rate indicates the quarterly variation in the employment of the client. This is a strong indicator on the financial status/background of the client. Hence naturally this becomes an important predictor for the client opening a term deposit.

Cons.price.index is a very important predictor because this represents the consumer price index, another financial determinant of a customer to be able to open (or have the willingness) or not open a term deposit with the bank.

Thus as an overview of how I obtain my final model:

I take the predictions on the train data (4000 of them) of my models (Logistic Regression, LASSO and Random Forest ones- Totally 7 of them) and create a data frame of them with the corresponding true label. I take 3500 of them as the new training set and train a Random Forest and use this model to predict on the 500 to get my final model having superior accuracy and least MCE.

This is how I built my classifier to be able to classify if a client called by the bank will end up opening up a term deposit with them or not. Hence task achieved!

REFERENCES

- The word cloud on the acknowledgement was obtained by understanding and running the Professor Zhao's code lectured in class. I just wanted to do it since I find it cool!
- http://stackoverflow.com/questions/14604439/plot-multiple-boxplot-in-one-graph
- http://www.r-bloggers.com/side-by-side-box-plots-with-patterns-from-data-sets-stacked-by-reshape2-and-melt-in-r/
- http://www.r-bloggers.com/ggplot2-multiple-boxplots-with-metadata/
- http://www.exegetic.biz/blog/2013/05/introducing-r-plottin-categorical-variables/
- http://stats.stackexchange.com/questions/82497/can-the-scaling-values-in-a-linear-discriminant-analysis-lda-be-used-to-plot-e
- https://tgmstat.wordpress.com/2014/01/15/computing-and-visualizing-lda-in-r/
- http://rstudio-pubs-static.s3.amazonaws.com/35817_2552e05f1d4e4db8ba87b334101a43da.html
- http://www.inside-r.org/packages/cran/adabag/docs/adaboost.M1
- http://www.r-bloggers.com/an-intro-to-ensemble-learning-in-r/
- http://stackoverflow.com/questions/1299871/how-to-join-merge-data-frames-inner-outer-left-right
- http://stackoverflow.com/questions/18275639/remove-highly-correlated-variables

APPENDIX: R CODE

```
# Advanced Statistics for Management (STAT 471/571/701)
# Spring 2016
# Final Project
# Prof: Dr. Linda Zhao
# Name: Akshay Varik
# Penn ID: 73531118


# Task:
# The data is related with direct marketing campaigns of a Portuguese banking
# institution. The marketing campaigns were based on phone calls. Often, more than one
# contact to the same client was required, in order to access if the product (bank term
# deposit) would be ('yes') or not ('no') subscribed. The classification goal is to predict
# if the client will subscribe (yes/no) a term deposit.


# Loading the file and set the working directory
rm(list=ls()) # Remove all the existing variables
dir=c("E:/Data Mining- STAT 571")   # my laptop
setwd(dir)


# Include all the libraries
library(plyr)
library(dplyr)
library(leaps)
library(glmnet)
library(pROC)
library(MASS)
library(car)
library(data.table)
library(rockchalk)
library(caret)
library(e1071)
library(ROCR)
library(calibrate)
library(gridExtra)
library(boot)
library(rpart)
library(tree)
library(randomForest)
library(rattle)
library(rpart.plot)

# Read the data file
data=read.csv("Data_FinalProject.csv",header=T)


# Preliminary familiarity with the data and cleaning the data
dim(data)
```

```
names(data)[21]="TD" # Rename the response varible as TD (Term deposit)
levels(data$TD)  # Get the levels in the response variable

a=length(which(data$TD == "yes")) # Checking out the proportion of the response
yes.percentage=a/dim(data)[1]*100


data$TD=as.factor(data$TD)
summary(data)
str(data)

data1=data[,-11] # Removed the duration variable (as per the suggestion on the site to
            # obtain realistic predictive model)
            # Since after the call the outcome of it would be known

data1$Sl_no=1:nrow(data1) # Created a column of serial number

sum(is.na(data)) # Check for missing values

# To subset data randomly i.e. extract 5000 rows
# data2=data1[sample(1:nrow(data1), 5000, replace=FALSE),]

# But I am creating a new file of 55% no and 45% yes in 5000 random subset
set.seed(1)
data_TD_no=subset(data1, TD=="no") # obtain all TD=no rows
data_TD_yes=subset(data1, TD=="yes") # obtain all TD=yes rows
data2_TD_no=data_TD_no[sample(1:nrow(data_TD_no), 2750, replace=FALSE),] # 3500 random values of
TD=no
data2_TD_yes=data_TD_yes[sample(1:nrow(data_TD_yes), 2250, replace=FALSE),] # 1500 random values
of TD=yes
data2= rbind(data2_TD_no, data2_TD_yes) # concatenated the two dataframes to get 1 dataframe
                        # This data frame will kind of constitue our entire dataset.

a=length(which(data2$TD == "yes")) # Cross checking out the proportion of the response
yes.percentage=a/dim(data2)[1]*100

data2=data2[sample(nrow(data2)),] # Randomly shuffled the rows of the dataframe

data3=data1[!(data1$Sl_no %in% data2$Sl_no),] # Data1-Data2 (All the rows we did not pick)


# Data: Original Dataset as I downloaded
# Data1: Here I have added the serial no column and removed the Duration column from Data
# Data2: I will work on this. Contains 55% no and 45% yes TD response. Randomly obtained from Data1
# Data3: All the rows of Data2 not included in Data1
write.csv(data1,file="E:/Data Mining- STAT 571/Data1.csv") # save the data files
write.csv(data2,file="E:/Data Mining- STAT 571/Data2.csv")
write.csv(data3,file="E:/Data Mining- STAT 571/Data3.csv")



data.cleaned=read.csv("Data2.csv",header=T) # read Data2 that I will be using
data.cleaned$TD=as.factor(data.cleaned$TD) # generate levels for categorical variable
```

```
data.cleaned$TD = ifelse(data.cleaned$TD=="yes", 1, 0) # add new comun of 1 for TD=yes and for TD=no
a=length(which(data.cleaned$TD == 1)) # Cross checking out the proportion of the response
yes.percentage=a/dim(data.cleaned)[1]*100

summary(data.cleaned)
str(data.cleaned)

data.cleaned=data.cleaned[-c(1,22)] # Dropped the unnecessary serial number columns

cor(data.cleaned[,unlist(lapply(data.cleaned, is.numeric))]) # correlation between
                              # numeric varaibles in the dataset

# Preliminary Visualization
require(ggplot2)
# pairs(data.cleaned[1:20], pch = 21)
# df.m = melt(data.cleaned, id.var = "TD")
# p <- ggplot(data = df.m, aes(x=variable, y=value))
# p <- p + geom_boxplot(aes(fill=TD))
# p <- p + facet_wrap( ~ variable, scales="free", ncol=4)
# p <- p + xlab("x-axis") + ylab("y-axis") + ggtitle("Box-plots")
# p <- p + guides(fill=guide_legend(title="TD"))
# p




# Now in this I divide the dataset into training data-80% and testing data-20%
data.cleaned=na.omit(data.cleaned)
set.seed(1)  # set a random seed so that we will be able to reproduce the random sample
index.train=sample(dim(data.cleaned)[1], 4000) # Take a random sample of n=4000 from 1 to N=5000
data.cleaned.train=data.cleaned[index.train,] # Set the 1000 randomly chosen subjects as a training data
data.cleaned.test=data.cleaned[-index.train,] # The remaining subjects will be reserved for testing purposes.
dim(data.cleaned.train)
dim(data.cleaned.test)




# Model 1: Performing Logistic Regression with all variables.
fit1=glm(TD~., data.cleaned.train, family=binomial(logit))
summary(fit1)
chi.sq= 5512.4-4149.5     # get the Chi-square stat
pchisq(chi.sq, 1, lower.tail=FALSE)  # p-value: from the likelihood Ratio test
anova(fit1, test="Chisq") # to test if the model is useful: null hypothesis is all (but the intercept) coeff's are 0
confint.default(fit1) # obtain the confidence level of the coefficient of
                 # the variables in this model.

# The chi-square distribution
par(mfrow=c(2,1))
hist(rchisq(4000, 2), freq=FALSE, breaks=20)
hist(rchisq(4000, 20), freq=FALSE, breaks=20)
```

```
# When DF is getting larger, Chi-Squared dis is approx. normal

#prediction on training data
fit1.pred.train=rep("0", 4000)   # prediction step 1
fit1.pred.train[fit1$fitted > 0.9]="1"  # prediction step 2 to get a classifier
fit1.pred.train=as.factor(fit1.pred.train)
cm.train=table(fit1.pred.train, data.cleaned.train$TD)
#Training error
fit1.mce.train=mean(fit1.pred.train != data.cleaned.train$TD)

#prediction on test data
fit1.predict=predict(fit1, data.cleaned.test, type="response", interval="confidence", se.fit=T)
fit1.pred.test=rep("0", 1000)   # prediction step 1
fit1.pred.test[fit1.predict$fit > 0.9]="1"  # prediction step 2 to get a classifier
fit1.pred.test=as.factor(fit1.pred.test)
data.frame(data.cleaned.test$TD, fit1.pred.test) # put observed y and predicted y's together
cm=table(fit1.pred.test, data.cleaned.test$TD)
confusionMatrix(data=fit1.pred.test, data.cleaned.test$TD)
#Testing error
fit1.mce.test=mean(fit1.pred.test != data.cleaned.test$TD)

sensitivity=cm[2,2]/sum(data.cleaned.test$TD =="1")
specificity=cm[1,1]/ sum(data.cleaned.test$TD == "0")
false.positive=cm[2,1]/sum(data.cleaned.test$TD == "0")

#ROC Curve
fit1.roc=roc(data.cleaned.train$TD, fit1$fitted, plot=T, col="blue")
names(fit1.roc)
auc(fit1.roc)

##### False Positive vs. Sensitivity curve is called ROC
plot(1-fit1.roc$specificities, fit1.roc$sensitivities, col="red", pch=16,
    xlab="False Positive",
    ylab="Sensitivity")

#### Given a False positive rate, locate the prob threshold
plot(1-fit1.roc$specificities, fit1.roc$thresholds, col="green", pch=16,
    xlab="False Positive",
    ylab="Threshold on prob")

# Tried to plot classifier boundary, but due to high dimension its hard!. Visualization
# would be hard.




#Model 2:Done regsubset generation to obtain 8 variables and logistic model fit
#Exhaustive search
fit2.exh=regsubsets(data.cleaned.train$TD~.,data.cleaned.train,        nvmax=8,        method="exhaustive",
really.big=T)
fit2.e=summary(fit2.exh)
```

```
fit2.e$bic

par(mfrow=c(2,1))     # Compare different criterions: as expected rsq ^ when p is larger
plot(fit2.e$rsq, xlab="Number of predictors", ylab="rsq", col="red", type="p", pch=16)
plot(fit2.e$rss, xlab="Number of predictors", ylab="rss", col="blue", type="p", pch=16)

coef(fit2.exh,8)

par(mfrow=c(3,1))
plot(fit2.e$cp, xlab="Number of predictors",
    ylab="cp", col="red", type="p", pch=16)
plot(fit2.e$bic, xlab="Number of predictors",
    ylab="bic", col="blue", type="p", pch=16)
plot(fit2.e$adjr2, xlab="Number of predictors",
    ylab="adjr2", col="green", type="p", pch=16)

Reg.var=rownames(as.matrix(coef(fit2.exh,8))) # variables chosen

fit2.1=glm(TD~month+poutcome+emp.var.rate+cons.price.idx      #Building a logistic regression model
       +loan, data.cleaned.train, family=binomial(logit))
summary(fit2.1)

anova(fit1,fit2.1) # Compare Model 1 and Model 2.1


# Forward selection
fit2.for=regsubsets(data.cleaned.train$TD~.,data.cleaned.train, nvmax=8, method="forward", really.big=T)
fit2.f=summary(fit2.for)

fit2.f$cp

coef(fit2.for,8)

Reg.var=rownames(as.matrix(coef(fit2.for,8)))

fit2.2=glm(TD~month+poutcome+emp.var.rate+cons.price.idx      #Building a logistic regression model
        +loan, data.cleaned.train, family=binomial(logit))
summary(fit2.2)


# Backward Selection
fit2.bac=regsubsets(data.cleaned.train$TD~.,data.cleaned.train,       nvmax=8,       method="backward",
really.big=T)
fit2.b=summary(fit2.bac)

fit2.b$rsq

coef(fit2.bac,8)

Reg.var=rownames(as.matrix(coef(fit2.bac,8)))

fit2.3=glm(TD~month+poutcome+emp.var.rate+cons.price.idx      #Building a logistic regression model
        +loan, data.cleaned.train, family=binomial(logit))
```

```r
summary(fit2.3)

fit2=fit2.3

par(mfrow=c(2,1))
plot(fit2,1)
plot(fit2,2)

chi.sq= 5512.4-4226.9     # get the Chi-square stat
pchisq(chi.sq, 1, lower.tail=FALSE)  # p-value: from the likelihood Ratio test
anova(fit2, test="Chisq") # to test if the model is useful: null hypothesis is all (but the intercept) coeff's are 0


#prediction on training data
fit2.pred.train=rep("0", 4000)   # prediction step 1
fit2.pred.train[fit1$fitted > 0.9]="1"  # prediction step 2 to get a classifier
fit2.pred.train=as.factor(fit2.pred.train)
cm.train=table(fit2.pred.train, data.cleaned.train$TD)
#Training error
fit2.mce.train=mean(fit2.pred.train != data.cleaned.train$TD)

#prediction on test data
fit2.predict=predict(fit2, data.cleaned.test, type="response", interval="confidence", se.fit=T)
fit2.pred.test=rep("0", 1000)   # prediction step 1
fit2.pred.test[fit2.predict$fit > 0.9]="1"  # prediction step 2 to get a classifier
fit2.pred.test=as.factor(fit2.pred.test)
data.frame(data.cleaned.test$TD, fit2.pred.test) # put observed y and predicted y's together
cm=table(fit2.pred.test, data.cleaned.test$TD)
confusionMatrix(data=fit2.pred.test, data.cleaned.test$TD)
#Testing error
fit2.mce.test=mean(fit2.pred.test != data.cleaned.test$TD)

sensitivity=cm[2,2]/sum(data.cleaned.test$TD =="1")
specificity=cm[1,1]/ sum(data.cleaned.test$TD == "0")
false.positive=cm[2,1]/sum(data.cleaned.test$TD == "0")

#ROC Curve
fit2.roc=roc(data.cleaned.train$TD, fit2$fitted, plot=T, col="blue")
names(fit2.roc)
auc(fit2.roc)

##### False Positive vs. Sensitivity curve is called ROC
plot(1-fit2.roc$specificities, fit2.roc$sensitivities, col="red", pch=16,
    xlab="False Positive",
    ylab="Sensitivity")

#### Given a False positive rate, locate the prob threshold
plot(1-fit2.roc$specificities, fit2.roc$thresholds, col="green", pch=16,
    xlab="False Positive",
    ylab="Threshold on prob")

# Tried to plot classifier boundary, but due to high dimension its hard!. Visualization
# would be hard.
```

```
# Model3: Using Regularization Techniques
X.fl=model.matrix(~., data.cleaned.train)   # put data.frame into a matrix
colnames(X.fl)
Y=X.fl[, 53]  # extract y
X.fl=X.fl[, -c(53)]
fit3.lambda=cv.glmnet(X.fl, Y, alpha=1,nfolds=10)

names(fit3.lambda)
plot(fit3.lambda)
plot(fit3.lambda$lambda)
meancverror=fit3.lambda$cvm              # the mean cv error
plot(fit3.lambda$lambda, fit3.lambda$cvm, xlab="lambda", ylab="mean cv errors")
fit3.lambda$lambda.min        # min lambda changes a lot as a function of nfolds!
nonzeros=fit3.lambda$nzero
plot(fit3.lambda$lambda, fit3.lambda$nzero, xlab="lambda", ylab="number of non-zeros")

#output beta's from lambda.1se (this way we use smaller set of variables.)
coef.1se=coef(fit3.lambda, s="lambda.1se")
coef.1se=coef.1se[which(coef.1se !=0),]
pvariables=rownames(as.matrix(coef.1se))


# Fit the model
glm.input=as.formula(paste("TD", "~", paste(pvariables[-1], collapse = "+"))) # formula
fit3=glm(TD~job+default+contact+month+campaign+poutcome+emp.var.rate
        +nr.employed, data=data.cleaned.train)
summary(fit3)

anova(fit1,fit3)
anova(fit2,fit3)

chi.sq= 991.81-702.35     # get the Chi-square stat
pchisq(chi.sq, 1, lower.tail=FALSE)  # p-value: from the likelihood Ratio test
anova(fit3, test="Chisq") # to test if the model is useful: null hypothesis is all (but the intercept) coeff's are 0


#prediction on training data
fit3.pred.train=rep("0", 4000)   # prediction step 1
fit3.pred.train[fit1$fitted > 0.9]="1"  # prediction step 2 to get a classifier
fit3.pred.train=as.factor(fit3.pred.train)
cm.train=table(fit3.pred.train, data.cleaned.train$TD)
#Training error
fit3.mce.train=mean(fit3.pred.train != data.cleaned.train$TD)

#prediction on test data
```

```
fit3.predict=predict(fit3, data.cleaned.test, type="response", interval="confidence", se.fit=T)
fit3.pred.test=rep("0", 1000)   # prediction step 1
fit3.pred.test[fit3.predict$fit > 0.9]="1"  # prediction step 2 to get a classifier
fit3.pred.test=as.factor(fit3.pred.test)
data.frame(data.cleaned.test$TD, fit3.pred.test) # put observed y and predicted y's together
cm=table(fit3.pred.test, data.cleaned.test$TD)
confusionMatrix(data=fit3.pred.test, data.cleaned.test$TD)
#Testing error
fit3.mce.test=mean(fit3.pred.test != data.cleaned.test$TD)

sensitivity=cm[2,2]/sum(data.cleaned.test$TD =="1")
specificity=cm[1,1]/ sum(data.cleaned.test$TD == "0")
false.positive=cm[2,1]/sum(data.cleaned.test$TD == "0")

#ROC Curve
fit3.roc=roc(data.cleaned.train$TD, fit3$fitted, plot=T, col="blue")
names(fit3.roc)
auc(fit3.roc)

##### False Positive vs. Sensitivity curve is called ROC
plot(1-fit3.roc$specificities, fit3.roc$sensitivities, col="red", pch=16,
    xlab="False Positive",
    ylab="Sensitivity")

#### Given a False positive rate, locate the prob threshold
plot(1-fit3.roc$specificities, fit3.roc$thresholds, col="green", pch=16,
    xlab="False Positive",
    ylab="Threshold on prob")

# Tried to plot classifier boundary, but due to high dimension its hard!. Visualization
# would be hard.




# Model 4: Cross Validation on training set of Model 1
ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions = TRUE)

fit4 <- train(TD ~.,data=data.cleaned.train, method="glm", family="binomial",
              trControl = ctrl, tuneLength = 5) # Fit the model

summary(fit4)

pred = predict(fit4, data.cleaned.test) # predict on the testing data set

# try=rep("0", 1000)   # prediction step 1
# try[pred > 0.9]="1"  # prediction step 2 to get a classifier
# try=as.factor(try)
```

```
# data.frame(data.cleaned.test$TD, try) # put observed y and predicted y's together
# cm=table(try, data.cleaned.test$TD)
# confusionMatrix(data=try, data.cleaned.test$TD)
# try.mce=mean(try != data.cleaned.test$TD)

abc=prediction(pred,data.cleaned.test$TD)
AUC = as.numeric(performance(abc, "auc")@y.values)
ACC= performance(abc, "acc")
mean(ACC@y.values[[1]])
plot(performance(abc, 'tpr', 'fpr'))
plot(ACC)

Sensitivity= performance(abc, "sens")
mean(Sensitivity@y.values[[1]])
Specificity= performance(abc, "spec")
mean(Specificity@y.values[[1]])
```

```
# Model 5: Cross Validation on training set of Model 2
ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions = TRUE)

fit5 <- train(TD ~month+poutcome+emp.var.rate+cons.price.idx+loan,
        data=data.cleaned.train, method="glm", family="binomial",
        trControl = ctrl, tuneLength = 5) # Fit the model

summary(fit5)

pred = predict(fit5, data.cleaned.test) # predict on the testing data set

abc=prediction(pred,data.cleaned.test$TD)
AUC = as.numeric(performance(abc, "auc")@y.values)
ACC= performance(abc, "acc")
max(ACC@y.values[[1]])
plot(performance(abc, 'tpr', 'fpr'))
plot(ACC)

Sensitivity= performance(abc, "sens")
mean(Sensitivity@y.values[[1]])
Specificity= performance(abc, "spec")
mean(Specificity@y.values[[1]])
```

```
# Model 6: Cross Validation on training set of Model 3
ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions = TRUE)

fit6 <- train(TD ~job+default+contact+month+campaign+poutcome+emp.var.rate
        +nr.employed,  data=data.cleaned.train, method="glm", family="binomial",
        trControl = ctrl, tuneLength = 5) # Fit the model

summary(fit6)

pred = predict(fit6, data.cleaned.test) # predict on the testing data set

abc=prediction(pred,data.cleaned.test$TD)
AUC = as.numeric(performance(abc, "auc")@y.values)
ACC= performance(abc, "acc")
max(ACC@y.values[[1]])
plot(performance(abc, 'tpr', 'fpr'))
plot(ACC)

Sensitivity= performance(abc, "sens")
mean(Sensitivity@y.values[[1]])
Specificity= performance(abc, "spec")
mean(Specificity@y.values[[1]])
```

```
# Model 7: LDA on all variables
fit7 <- lda(data.cleaned.train$TD ~., data=data.cleaned.train) # fit the lda model
plda <- predict(object = fit7,newdata = data.cleaned.test) #predict on test data
summary(fit7)

plda.class.1=predict(fit7, data.cleaned.test)$class # gives the class of the test data
plda.class.train.1=predict(fit7, data.cleaned.train)$class # gives the class of the train data

# create a histogram of the discriminant function values
ldahist(data = plda$x[,1], g=data.cleaned.test$TD)

# create a scatterplot of the discriminant function values
plot(plda$x[,1], type="n", ylab=c("LDA Axis 1"))
text(plda$x[,1], row.names(data.cleaned.test),  col=c(as.numeric(data.cleaned.test$TD)+10))

# Compute the misclasification error of the model
ct <- table(data.cleaned.test$TD, plda$class)
(ct[1,1]+ct[2,2])/sum(ct)
```

```
# Model 8: LDA on variables otained from regsubsets
fit8 <- lda(data.cleaned.train$TD ~ month+poutcome+emp.var.rate+cons.price.idx
        +loan, data=data.cleaned.train) # fit the lda model
plda <- predict(object = fit8,newdata = data.cleaned.test) #predict on test data
summary(fit8)

plda.class.2=predict(fit8,data.cleaned.test)$class # gives the class of the test data
plda.class.train.2=predict(fit8,data.cleaned.train)$class # gives the class of the train data

# create a histogram of the discriminant function values
ldahist(data = plda$x[,1], g=data.cleaned.test$TD)

# create a scatterplot of the discriminant function values
plot(plda$x[,1], type="n", ylab=c("LDA Axis 1"))
text(plda$x[,1], row.names(data.cleaned.test),  col=c(as.numeric(data.cleaned.test$TD)+10))

# Compute the misclasification error of the model
ct <- table(data.cleaned.test$TD, plda$class)
(ct[1,1]+ct[2,2])/sum(ct)
```

```
# Model 9: LDA on variables obtained from LASSO
fit9 <- lda(data.cleaned.train$TD ~job+default+contact+month+campaign+poutcome+emp.var.rate
        +nr.employed, data=data.cleaned.train) # fit the lda model
plda <- predict(object = fit9,newdata = data.cleaned.test) #predict on test data
summary(fit7)

plda.class.3=predict(fit9, data.cleaned.test)$class # gives the class of the test data
plda.class.train.3=predict(fit9, data.cleaned.train)$class # gives the class of the train data

# create a histogram of the discriminant function values
ldahist(data = plda$x[,1], g=data.cleaned.test$TD)

# create a scatterplot of the discriminant function values
plot(plda$x[,1], type="n", ylab=c("LDA Axis 1"))
text(plda$x[,1], row.names(data.cleaned.test),  col=c(as.numeric(data.cleaned.test$TD)+10))

# Compute the misclasification error of the model
ct <- table(data.cleaned.test$TD, plda$class)
(ct[1,1]+ct[2,2])/sum(ct)
```

```
# Model 10: Random Forest on all variables
fit10.1= tree(TD~., data=data.cleaned.train)
plot(fit10.1)
text(fit10.1, pretty=0)
fit10.1$frame
fit10.1.result=summary(fit10.1)
fit10.1.result$dev
#xyz=summary(glm(TD~nr.employed+euribor3m+month,data.cleaned.train, family=binomial(logit)))
#names(xyz)
#RSS.LogReg=(4000-4)*((xyz)$deviance)^2

fit.tree=rpart(TD~., data.cleaned.train)
fancyRpartPlot(fit.tree)   # The plot shows the split together with more information
fit.tree$frame

# Split on gini
fit10.1.gini=tree(TD~., data.cleaned.train, split="gini")
plot(fit10.1.gini)
text(fit10.1.gini, pretty=TRUE)  # plot the labels
fit10.1.gini$frame
summary(fit10.1.gini)$dev


#Bootstrap
RSS=0  # initial values
n.unique=0
n=nrow(data.cleaned.train)
for (i in 1:100)
{
  index1=sample(n, n, replace=TRUE)
  Sample1=data.cleaned.train[index1, ] # Take a bootstrap sample
  fit1.boot=tree(TD~., Sample1)  # Get a tree fit
  plot(fit1.boot,
       title="Trees with a Bootstrap sample")
  text(fit1.boot, pretty=0)
  RSS[i]=summary(fit1.boot)$dev  # output RSS for each bootstrap tree
  n.unique[i]=length(unique(index1))
  Sys.sleep(2)                    # Pause for 2 seconds before running for next round
}

hist(RSS, breaks=30,
     col="blue",
     main="RSS from different Bootstrap trees")
```

```
hist(n.unique, breaks=30,
    col="red",
    main="number of unique subjects included in each Bootstrap sample")

hist(n-n.unique, breaks=30,
    col="green",
    main="number of OOB subjects not included in each Bootstrap sample")


#Random Forest
rf.error.p=1:19
for (p in 1:19)
{
  fit.rf=randomForest(TD~., data.cleaned.train, mtry=p, ntree=100)
  rf.error.p[p]=fit.rf$mse[100]
}
rf.error.p
plot(1:19, rf.error.p, pch=16,
    xlab="mtry",
    ylab="mse of mtry")

# For a fixed mtry= 4
fit10.2=randomForest(TD~., data.cleaned.train, mtry=4, ntree=100)
str(fit10.2)
plot(fit10.2)
summary(fit10.2)

plot(fit10.2$mse, xlab="number of trees",
    ylab="ave mse of the 100 trees",
    pch=16)

# oob times for each obs'n
fit10.2$oob.times  # Out of bags for each observation.
hist(fit10.2$oob.times)

trainingerror=mean((fit10.2$y-fit10.2$predicted)^2) #  this will output the oob errors

pred1=predict(fit10.2, data.cleaned.test) # make predictions on the test data
pred1.train=predict(fit10.2, data.cleaned.train) # predictions on trained data

try1=rep("0", 1000)
try1[pred1 > 0.9]="1"
try1=as.factor(try1)
data.frame(data.cleaned.test$TD, try1) # put observed y and predicted y's together
cm=table(try1, data.cleaned.test$TD)
confusionMatrix(data=try1, data.cleaned.test$TD)
try1.mce=mean(try1 != data.cleaned.test$TD)

try1.train=rep("0", 4000)
try1.train[pred1.train > 0.9]="1"
try1.train=as.factor(try1.train)

abc=prediction(pred1,data.cleaned.test$TD)
```

```
AUC = as.numeric(performance(abc, "auc")@y.values)
ACC= performance(abc, "acc")
mean(ACC@y.values[[1]])
plot(performance(abc, 'tpr', 'fpr'))
plot(ACC)

Sensitivity= performance(abc, "sens")
mean(Sensitivity@y.values[[1]])
Specificity= performance(abc, "spec")
mean(Specificity@y.values[[1]])
```

```
# Model 11: Random Forest on variables obtained from Regsubsets
fit11.1= tree(TD~month+poutcome+emp.var.rate+cons.price.idx
          +loan, data=data.cleaned.train)
plot(fit11.1)
text(fit11.1, pretty=0)
fit11.1$frame
fit11.1.result=summary(fit11.1)
fit11.1.result$dev

fit.tree=rpart(TD~month+poutcome+emp.var.rate+cons.price.idx
           +loan, data.cleaned.train)
fancyRpartPlot(fit.tree)   # The plot shows the split together with more information
fit.tree$frame

# Split on gini
fit11.1.gini=tree(TD~month+poutcome+emp.var.rate+cons.price.idx
              +loan, data.cleaned.train, split="gini")
plot(fit11.1.gini)
text(fit11.1.gini, pretty=TRUE)  # plot the labels
fit11.1.gini$frame
summary(fit11.1.gini)$dev


#Bootstrap
RSS=0  # initial values
n.unique=0
n=nrow(data.cleaned.train)
for (i in 1:100)
{
  index1=sample(n, n, replace=TRUE)
  Sample1=data.cleaned.train[index1, ]  # Take a bootstrap sample
  fit1.boot=tree(TD~month+poutcome+emp.var.rate+cons.price.idx
```

```
          +loan, Sample1)  # Get a tree fit
  plot(fit1.boot,
     title="Trees with a Bootstrap sample")
  text(fit1.boot, pretty=0)
  RSS[i]=summary(fit1.boot)$dev  # output RSS for each bootstrap tree
  n.unique[i]=length(unique(index1))
  Sys.sleep(2)                    # Pause for 2 seconds before running for next round
}

hist(RSS, breaks=30,
    col="blue",
    main="RSS from different Bootstrap trees")

hist(n.unique, breaks=30,
    col="red",
    main="number of unique subjects included in each Bootstrap sample")

hist(n-n.unique, breaks=30,
    col="green",
    main="number of OOB subjects not included in each Bootstrap sample")


#Random Forest
rf.error.p=1:4
for (p in 1:4)
{
  fit.rf=randomForest(TD~month+poutcome+emp.var.rate+cons.price.idx
               +loan, data.cleaned.train, mtry=p, ntree=500)
  rf.error.p[p]=fit.rf$mse[500]
}
rf.error.p
plot(1:4, rf.error.p, pch=16,
    xlab="mtry",
    ylab="mse of mtry")

# For a fixed mtry= 2
fit11=randomForest(TD~month+poutcome+emp.var.rate+cons.price.idx
            +loan, data.cleaned.train, mtry=2, ntree=500)
str(fit11)
plot(fit11)
summary(fit11)

plot(fit11$mse, xlab="number of trees",
    ylab="ave mse of the 500 trees",
    pch=16)

# oob times for each obs'n
fit11$oob.times  # Out of bags for each observation.
hist(fit11$oob.times)

trainingerror=mean((fit11$y-fit11$predicted)^2) #  this will output the oob errors

pred2=predict(fit11, data.cleaned.test) # make predictions on the test data
```

```
pred2.train=predict(fit11, data.cleaned.train) # make predictions on the train data

try2=rep("0", 1000)
try2[pred2 > 0.9]="1"
try2=as.factor(try2)
data.frame(data.cleaned.test$TD, try2) # put observed y and predicted y's together
cm=table(try2, data.cleaned.test$TD)
confusionMatrix(data=try2, data.cleaned.test$TD)
try2.mce=mean(try2 != data.cleaned.test$TD)

try2.train=rep("0", 4000)
try2.train[pred2.train > 0.9]="1"
try2.train=as.factor(try2.train)

abc=prediction(pred2,data.cleaned.test$TD)
AUC = as.numeric(performance(abc, "auc")@y.values)
ACC= performance(abc, "acc")
mean(ACC@y.values[[1]])
max(ACC@y.values[[1]])
plot(performance(abc, 'tpr', 'fpr'))
plot(ACC)

Sensitivity= performance(abc, "sens")
mean(Sensitivity@y.values[[1]])
Specificity= performance(abc, "spec")
mean(Specificity@y.values[[1]])
```

```
# Model 12: Random Forest on variables obtained from LASSO
fit12.1= tree(TD~job+default+contact+month+campaign+poutcome+emp.var.rate
        +nr.employed, data=data.cleaned.train)
plot(fit12.1)
text(fit12.1, pretty=0)
fit12.1$frame
fit12.1.result=summary(fit12.1)
fit12.1.result$dev

fit.tree=rpart(TD~job+default+contact+month+campaign+poutcome+emp.var.rate
        +nr.employed, data.cleaned.train)
fancyRpartPlot(fit.tree)   # The plot shows the split together with more information
fit.tree$frame

# Split on gini
fit12.1.gini=tree(TD~job+default+contact+month+campaign+poutcome+emp.var.rate
        +nr.employed, data.cleaned.train, split="gini")
```

```
plot(fit12.1.gini)
text(fit12.1.gini, pretty=TRUE)  # plot the labels
fit12.1.gini$frame
summary(fit12.1.gini)$dev


#Bootstrap
RSS=0  # initial values
n.unique=0
n=nrow(data.cleaned.train)
for (i in 1:100)
{
  index1=sample(n, n, replace=TRUE)
  Sample1=data.cleaned.train[index1, ]  # Take a bootstrap sample
  fit1.boot=tree(TD~job+default+contact+month+campaign+poutcome+emp.var.rate
               +nr.employed, Sample1)  # Get a tree fit
  plot(fit1.boot,
      title="Trees with a Bootstrap sample")
  text(fit1.boot, pretty=0)
  RSS[i]=summary(fit1.boot)$dev  # output RSS for each bootstrap tree
  n.unique[i]=length(unique(index1))
  Sys.sleep(2)                     # Pause for 2 seconds before running for next round
}

hist(RSS, breaks=30,
    col="blue",
    main="RSS from different Bootstrap trees")

hist(n.unique, breaks=30,
    col="red",
    main="number of unique subjects included in each Bootstrap sample")

hist(n-n.unique, breaks=30,
    col="green",
    main="number of OOB subjects not included in each Bootstrap sample")


#Random Forest
rf.error.p=1:7
for (p in 1:7)
{
  fit.rf=randomForest(TD~job+default+contact+month+campaign+poutcome+emp.var.rate
                  +nr.employed, data.cleaned.train, mtry=p, ntree=500)
  rf.error.p[p]=fit.rf$mse[500]
}
rf.error.p
plot(1:7, rf.error.p, pch=16,
    xlab="mtry",
    ylab="mse of mtry")

# For a fixed mtry= 3
fit12=randomForest(TD~job+default+contact+month+campaign+poutcome+emp.var.rate
                 +nr.employed, data.cleaned.train, mtry=3, ntree=500)
```

```
str(fit12)
plot(fit12)
summary(fit12)

plot(fit12$mse, xlab="number of trees",
    ylab="ave mse of the 500 trees",
    pch=16)

# oob times for each obs'n
fit12$oob.times  # Out of bags for each observation.
hist(fit12$oob.times)

trainingerror=mean((fit12$y-fit12$predicted)^2) #  this will output the oob errors

pred3=predict(fit12, data.cleaned.test) # make predictions on the test data
pred3.train=predict(fit12, data.cleaned.train) # make predictions on the train data

try3=rep("0", 1000)
try3[pred3 > 0.9]="1"
try3=as.factor(try3)
data.frame(data.cleaned.test$TD, try3) # put observed y and predicted y's together
cm=table(try3, data.cleaned.test$TD)
confusionMatrix(data=try3, data.cleaned.test$TD)
try3.mce=mean(try3 != data.cleaned.test$TD)

try3.train=rep("0", 4000)
try3.train[pred3.train > 0.9]="1"
try3.train=as.factor(try3.train)

abc=prediction(pred3,data.cleaned.test$TD)
AUC = as.numeric(performance(abc, "auc")@y.values)
ACC= performance(abc, "acc")
max(ACC@y.values[[1]])
plot(performance(abc, 'tpr', 'fpr'))
plot(ACC)

Sensitivity= performance(abc, "sens")
mean(Sensitivity@y.values[[1]])
Specificity= performance(abc, "spec")
mean(Specificity@y.values[[1]])




# Ensemble of all the above models
#Model1
df1.1=data.frame(fit1.pred.train)
colnames(df1.1)="TD.Label.Predicted1"
```

```
#Model2
df2.1=data.frame(fit2.pred.train)
colnames(df2.1)="TD.Label.Predicted2"
#Model3
df3.1=data.frame(fit3.pred.train)
colnames(df3.1)="TD.Label.Predicted3"
#Model7
df7.1=data.frame(plda.class.train.1)
colnames(df7.1)="TD.Label.Predicted7"
#Model8
df8.1=data.frame(plda.class.train.2)
colnames(df8.1)="TD.Label.Predicted8"
#Model9
df9.1=data.frame(plda.class.train.3)
colnames(df9.1)="TD.Label.Predicted9"
#Model10
df10.1=data.frame(try1.train)
colnames(df10.1)="TD.Label.Predicted10"
#Model11
df11.1=data.frame(try2.train)
colnames(df11.1)="TD.Label.Predicted11"
#Model12
df12.1=data.frame(try3.train)
colnames(df12.1)="TD.Label.Predicted12"

# combine all the dataframes
df.predictions=cbind(df1.1, df2.1, df3.1, df7.1, df8.1, df9.1, df10.1, df11.1, df12.1)
str(df.predictions)
# convert the dataframe to numeric type
indx <- sapply(df.predictions, is.factor)
df.predictions[indx] <- lapply(df.predictions[indx], function(x) as.numeric(as.character(x)))
#get mean value of predictions
df.predictions.mean=data.frame(Mean.Prediction=rowMeans(df.predictions))

# Final Classifier By Equal Weights
try.final1=rep("0", 4000)
try.final1[df.predictions.mean > 0.5]="1"
try.final1=as.factor(try.final1)
data.frame(data.cleaned.train$TD, try.final1) # put observed y and predicted y's together
cm=table(try.final1, data.cleaned.train$TD)
confusionMatrix(data=try.final1, data.cleaned.train$TD)
try.final1.mce=mean(try.final1 != data.cleaned.train$TD)

# Linear Regression
df.main=data.frame(data.cleaned.train$TD, df.predictions)

fitfinal1=lm(data.cleaned.train.TD~., data=df.main)
summary(fitfinal1)

fitfinal2=update(fitfinal1, .~. -TD.Label.Predicted2 -TD.Label.Predicted3) # Removed coorelated variables
summary(fitfinal2)
```

```
# FINAL MODEL!!!!!!!!!!!!!!!!!!
# Random Forest

#Divide the dataset into training and testing
set.seed(1)  # set a random seed so that we will be able to reproduce the random sample
index.train1=sample(dim(df.main)[1], 3500) # Take a random sample of n=3500 from 1 to N=4000
df.main.train=df.main[index.train1,] # Set the 500 randomly chosen subjects as a training data
df.main.test=df.main[-index.train1,]

# Fit a random forest tree
fit.tree123=randomForest(data.cleaned.train.TD~., df.main.train, mtry=3, ntree=500)
fit.tree1234=rpart(data.cleaned.train.TD~., df.main.train)
fancyRpartPlot(fit.tree1234)   # The plot shows the split together with more information
fit.tree1234$frame
summary(fit.tree123)
plot(fit.tree123)

finaltree.pred=predict(fit.tree123, df.main.test)
try5=rep("0", 500)
try5[finaltree.pred > 0.5]="1"
try5=as.factor(try5)
data.frame(df.main.test$data.cleaned.train.TD, try5) # put observed y and predicted y's together
cm=table(try5, df.main.test$data.cleaned.train.TD)
confusionMatrix(data=try5, df.main.test$data.cleaned.train.TD)
try5.mce=mean(try5 != df.main.test$data.cleaned.train.TD)
abc1=prediction(finaltree.pred, df.main.test$data.cleaned.train.TD)
AUC1 = as.numeric(performance(abc1, "auc")@y.values)
ACC1= performance(abc1, "acc")
max(ACC1@y.values[[1]])
plot(performance(abc1, 'tpr', 'fpr'))
plot(ACC1)

Sensitivity= performance(abc1, "sens")
mean(Sensitivity@y.values[[1]])
Specificity= performance(abc1, "spec")
mean(Specificity@y.values[[1]])
```