|  |
| --- |
|  |
| Marketing Strategy for Banks |
| |  |  |  | | --- | --- | --- | | AKSHAY VARIK | Spring 2016 | STAT 571- ADVANCED STATISTICS FOR MANAGEMENT | |

**CONTENTS**

1. Acknowledgement……………………………………………………………………………………2
2. Project Proposal……………………………………………………………………………………….3
3. Project Report- Introduction…………………………………………………………………….4

* Background
* Goal of Study
* Background on the Data Set
* Initial Data Cleaning
* Approach Adopted
* Important Topics Touched Upon

1. Detailed Machine Learning Analysis- Model Building Process………………….8
2. Final Model Findings and Summary………………………………………………………...43
3. References………………………………………………………………………………………………47
4. Appendix: R Code…………………………………………………………………………………….48

**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](http://www.investopedia.com/terms/d/deposit.asp) held at a financial institution that has a [fixed term](http://www.investopedia.com/terms/f/fixedterm.asp). These are generally short-term with [maturities](http://www.investopedia.com/video/play/maturity-date/) ranging anywhere from a month to a few years. When a term deposit is purchased, the [lender](http://www.investopedia.com/terms/l/lender.asp) (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

1. Marital: It’s the marital status of the client (categorical)

* Divorced
* Married
* Single
* Unknown

1. Education: It’s the educations level of the client (categorical)

* basic.4y (Till 4th grade)
* basic.6y (Till 6th grade)
* basic.9y (Till 9th grade)
* high school
* illiterate
* professional.course
* university.degree
* unknown

1. Default: Has the client got any credit in default? (categorical)

* No
* Yes
* Unknown

1. Housing: Has the client got any housing loan? (categorical)

* No
* Yes
* Unknown

1. Loan: Has the client got any personal loan? (categorical)

* No
* Yes
* Unknown
* Data related with the last contact of the current campaign

1. Contact: How was the client contacted? (categorical)

* Cellular
* Telephone

1. Month: Last contact month of the year (categorical)

* January
* February
* March
* April
* May
* June
* July
* August
* September
* October
* November
* December

1. Day\_of\_week: Last contact day of the week (categorical)

* Monday
* Tuesday
* Wednesday
* Thursday
* Friday

1. Duration: It’s the last contact duration, in seconds (numeric).

* Other attributes

1. Campaign: Number of contacts performed during this campaign and for this client which includes the last contact (numeric)
2. 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)
3. Previous: Number of contacts performed before this campaign and for this client (numeric)
4. Poutcome: Outcome of the previous marketing campaign (categorical)

* Failure
* Non existent
* Succes
* Social and economic context attributes

1. Emp.var.rate: Employment variation rate - quarterly indicator (numeric)
2. Cons.price.idx: Consumer price index - monthly indicator (numeric)
3. Cons.conf.idx: Consumer confidence index - monthly indicator (numeric)
4. 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.

1. 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.

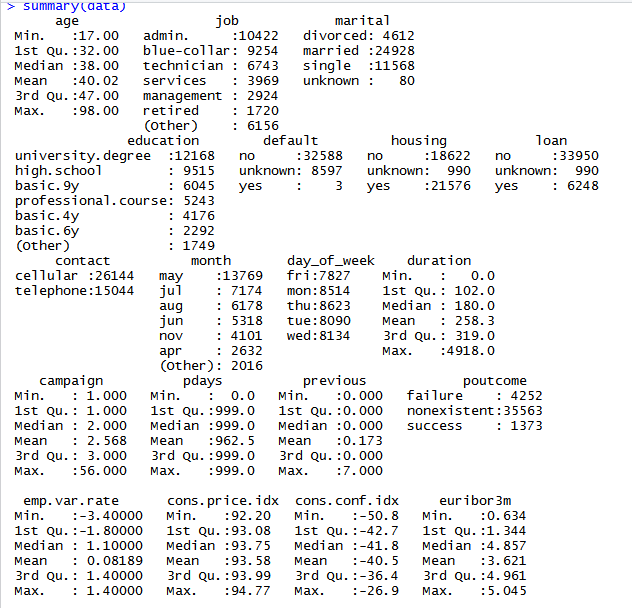
**Important Topics Touched Upon:**

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

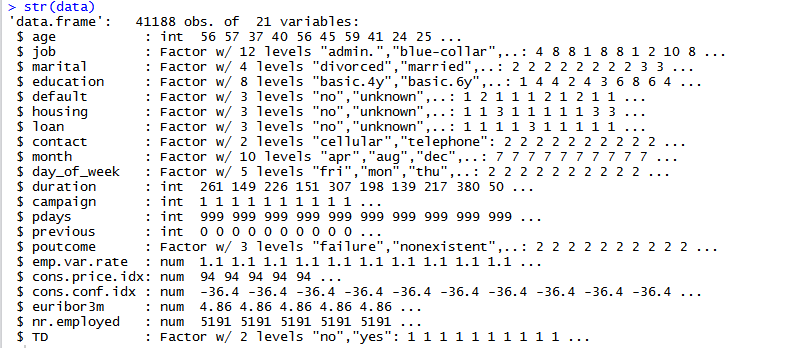
**DETAILED MACHINE LEARNING ANALYSIS**

**The Study (of Original Data):**

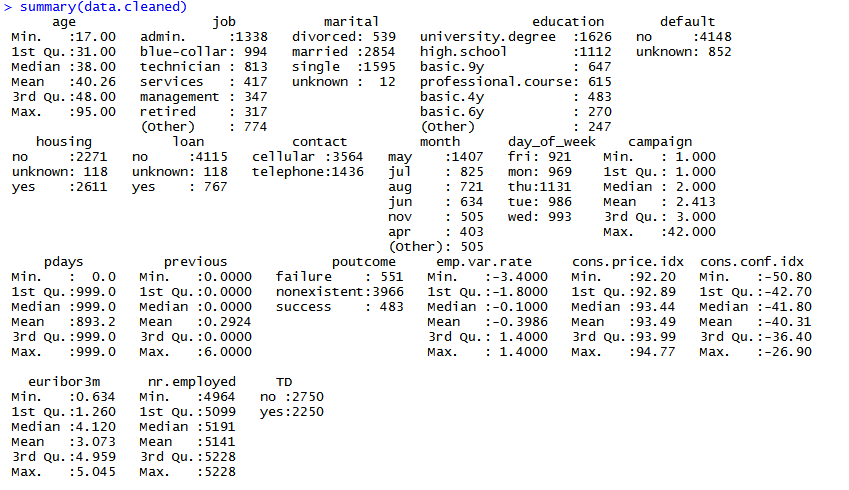
Firstly, I tried to understand the original data, the summary of which has been presented below. As can be seen the data is pleasant and has no missing values.



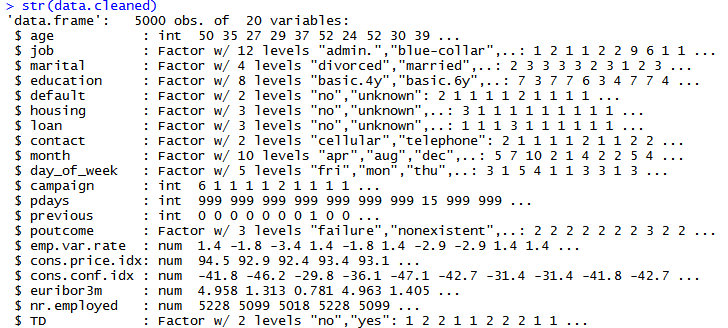




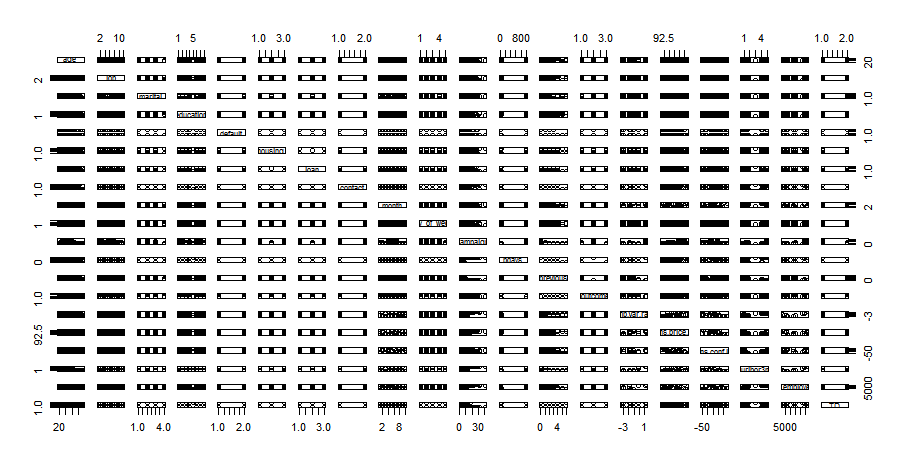
**The Study (of Cleaned Data):**



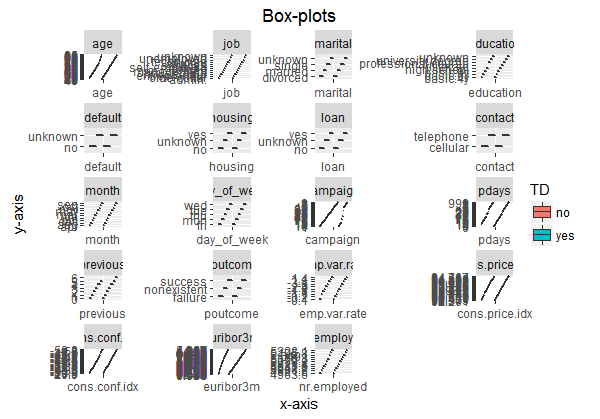
Below we can see how the ‘duration’ variable has been removed from the dataset.

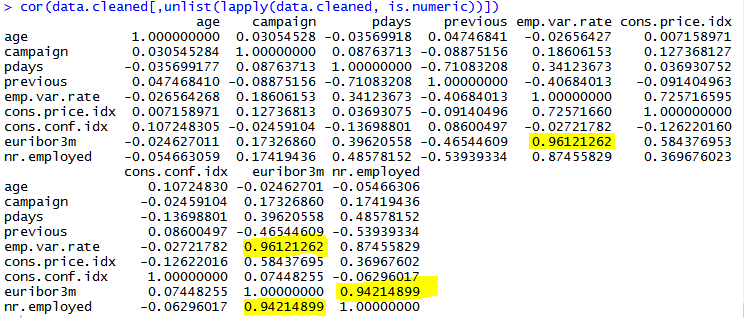


The correlations among the variables can be seen from the pair plot as can be seen below.



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.





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).

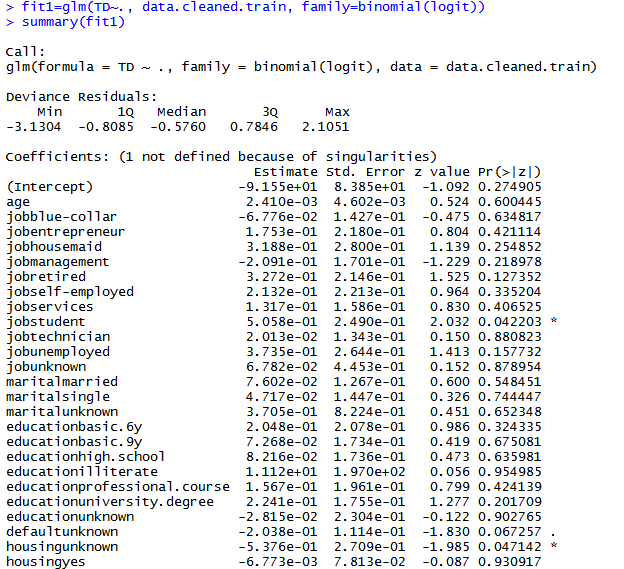
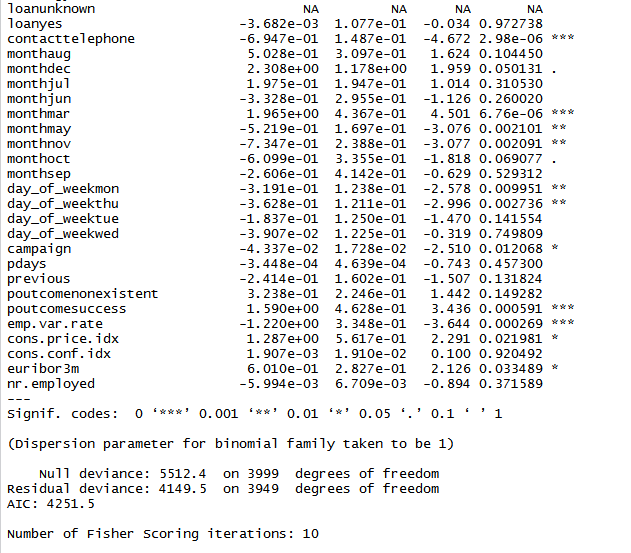
Next, thing I divided the dataset into training and testing data, so that I can train my model on the training set and then test it on the hidden test set to check for accuracy. This helps in understanding model accuracies and to pick amongst the best models. For this purpose I am taking the 4000 values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

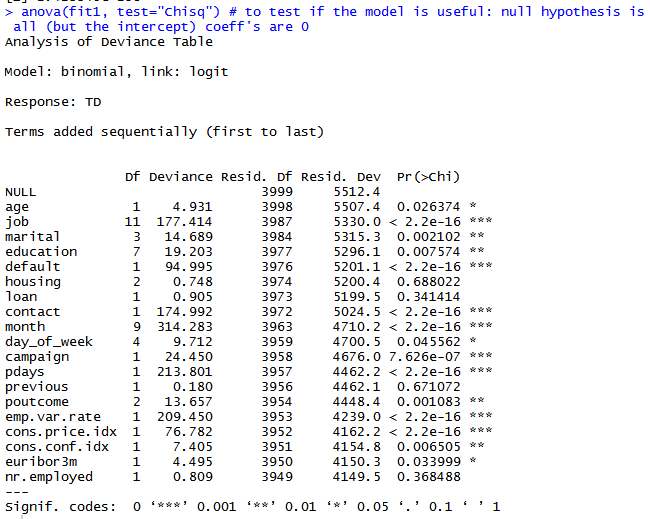
* Model 1:

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

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.

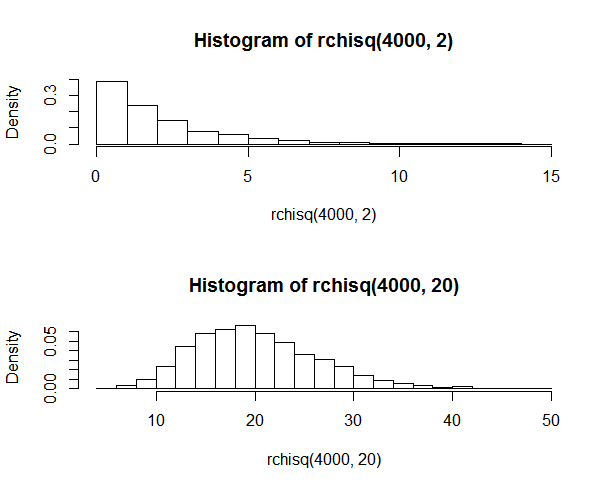


Here it is seen that the adding job variable on top of the age variable is helpful since the significance is high. AIC value of Model 1 is 4251.5. A model will be better than this if its AIC value is lower.

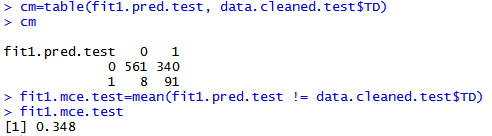
So to select the variables I use Regsubsets function. To keep my model size feasible I choose 8 variables (There are 8 variables whose significance level is <0.001). This is a good way to limit the model size. Exhaustive, forward and backward searches could be done to obtain the 8 variables.

This has been done in subsequent models.

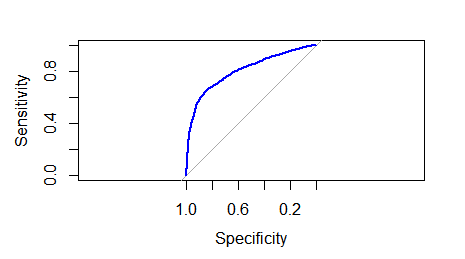
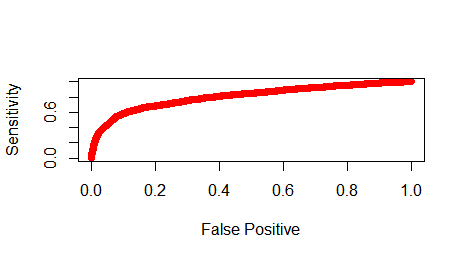
The chi-square distribution is as follows:



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:



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

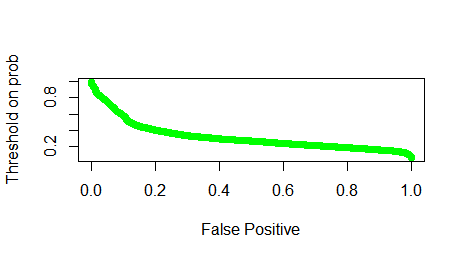
 

The above are the ROC curves. The Area under this curve (AUC) is 0.8063. For model selection higher the AUC better is the model.

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.

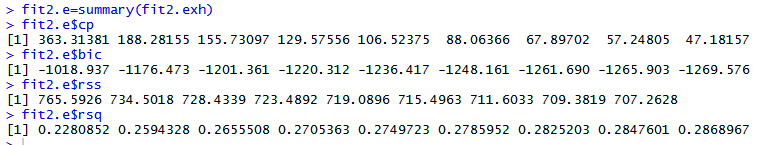
**TD = - 9.155e+01 + sum across all variables (Corresponding Coeff \* Correspoding Variable)**

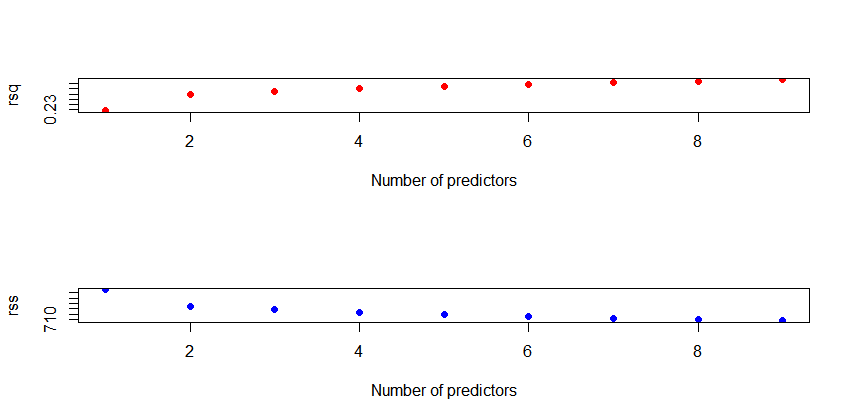
* Model 2:

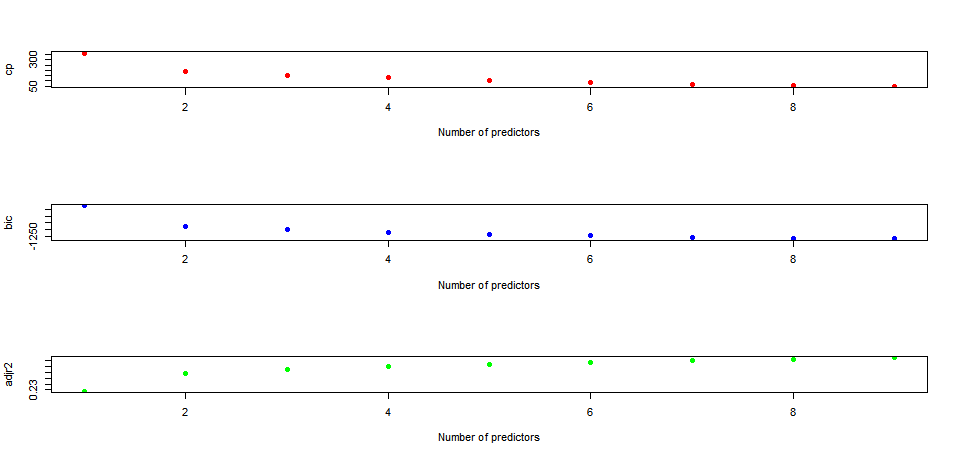
Here I used model selection method to get 8 variables using Regsubsets.

1. Exhaustive Search

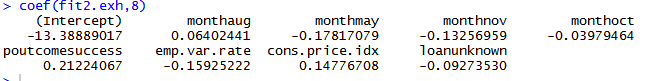
The values of the different criterions that were obtained is given below. RSS and RSQ and not the best criterions to be used for model selection. RSS will of course reduce when more variables are used



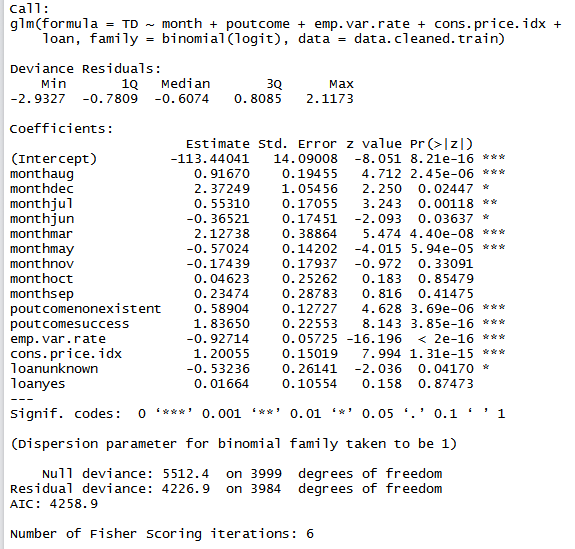


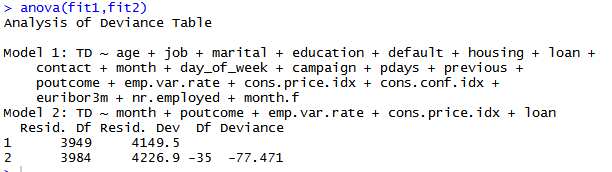


The 8 variables and their corresponding coefficients that we obtain are



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

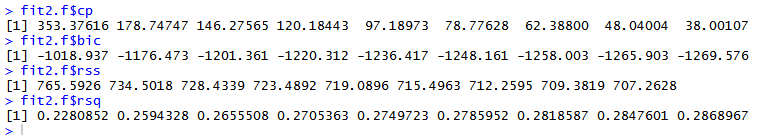




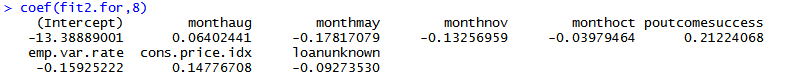
The above compares the 2 models generated so far. Surprisingly model 2 doesn’t perform as well than model 1 because we see that the deviance actually increased as compared to model 1.

1. Forward Selection

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



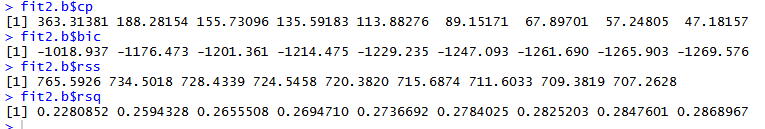
The 8 variables and their corresponding coefficients that we obtain are,



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

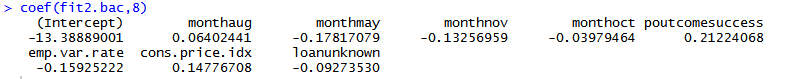
1. Backward Selection

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



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,

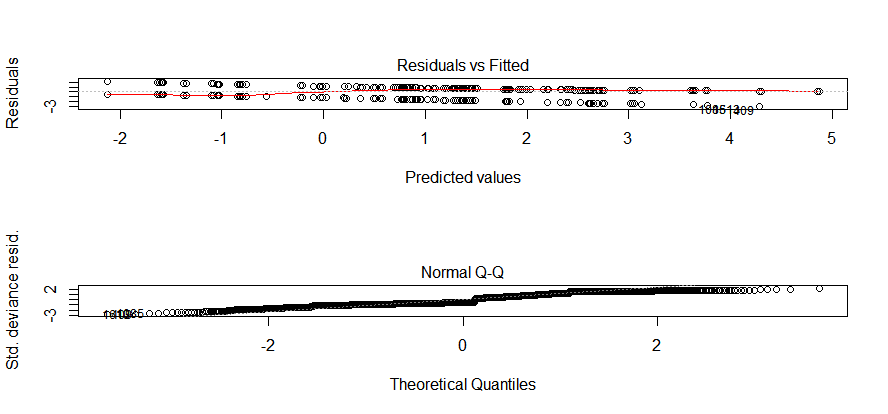


The coefficient values of this is also the same as the above 2 methods.

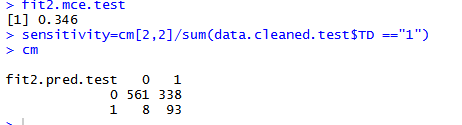
Thus the model (Model 2) we get is on performing logistic regression and retaining variables with significance level of \*\*\*:

**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**

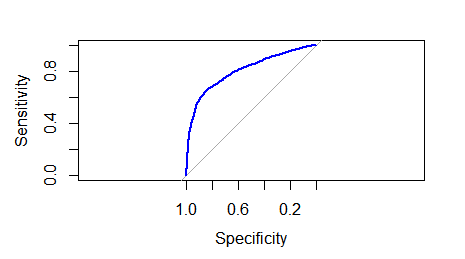
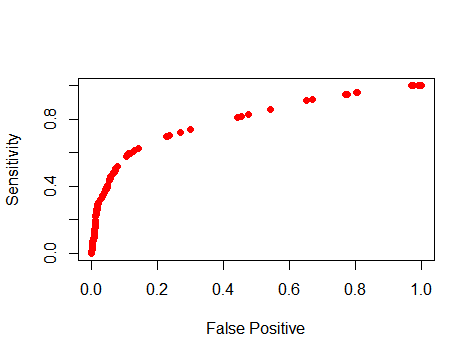
The above fit can be seen graphically below.



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

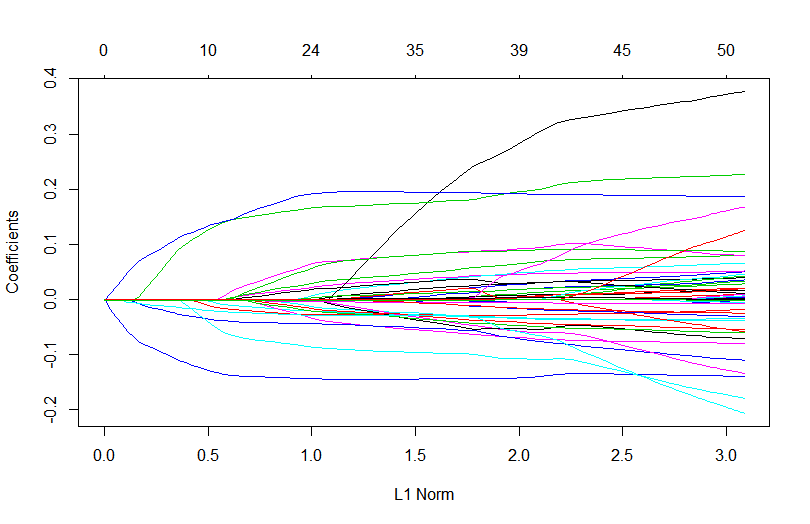


The ROC curve obtained is show below. The AUC value is 0.7967.

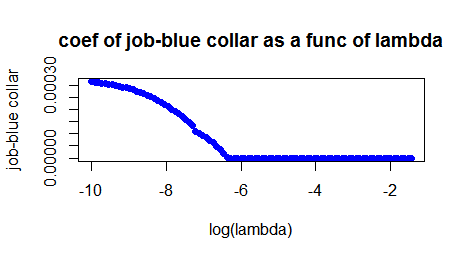
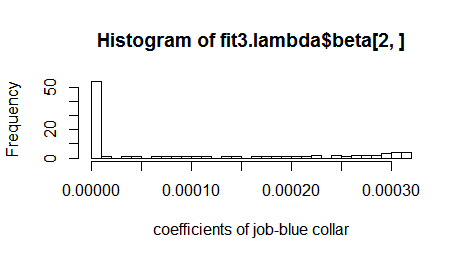
 

* Model 3:

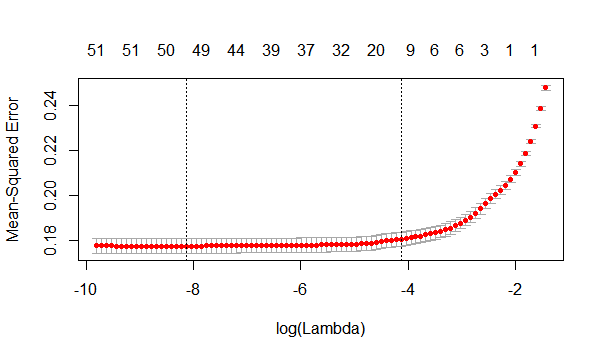
Here the model has been built by using regularization methods. I stick with LASSO than ridge since LASSO has the ability to do feature selection. When the lambda value is not specified in glmnet, the output consists of 100 outputs, one for each lambda. A plot of it shows that each hat beta is shrinking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



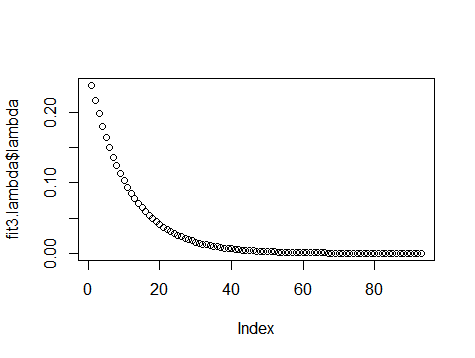
Also lambda values here have huge variability and hence log value is applied. The coefficients also have variability as shown below.

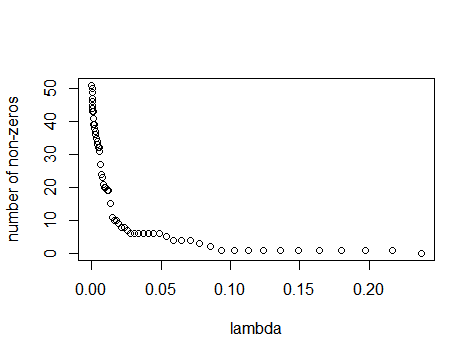
The below graph is an important one on basis of which I can choose the lambda value. Low lambda value means less penalty and the beta values will be high and vice versa.



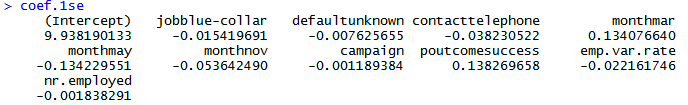
The 100 lambda values used are shown, and the min lambda value is 0.00029318.



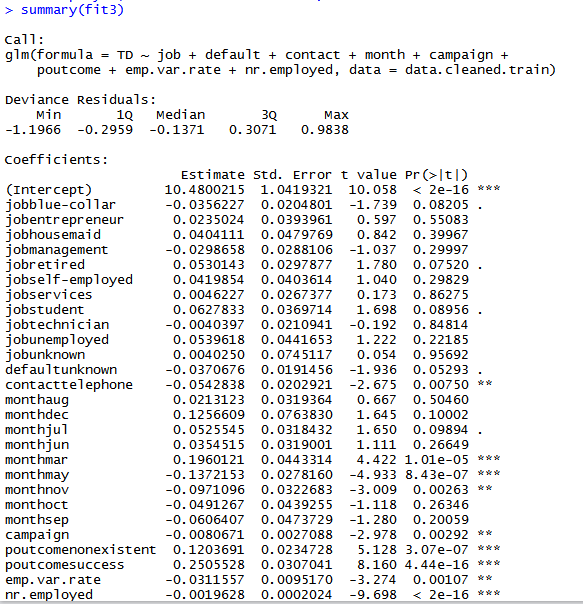
The number of non-zero terms as a function of the lambda used is shown below. As can be seen, higher the value of lambda lesser are the number of non-zeros.

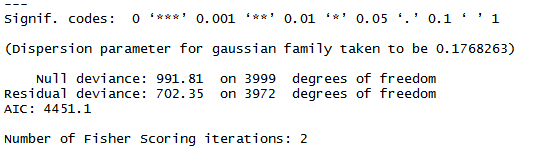


Here I used lambda.1se so that I can keep a tight hold on the number of features selected. The variables (10 make it) and their respective coefficients that make it are,

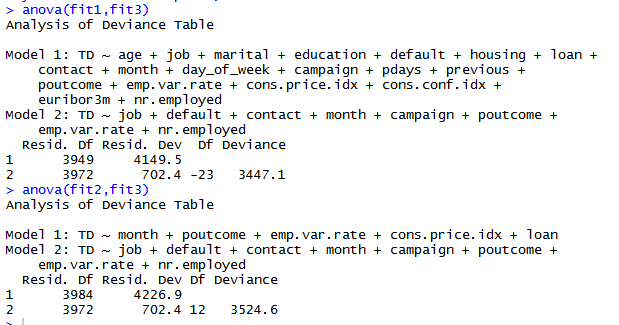


Now fitting a logistic regression model on the chosen variables,





Comparing the three models we have thus far

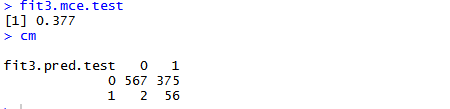


It can be seen that the deviance of model 3 from model 1 is lesser than its deviance from model 2. Hence Model 3 is the best so far amongst the three models.

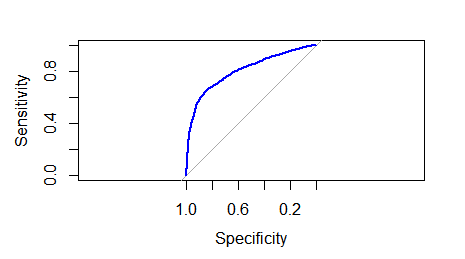
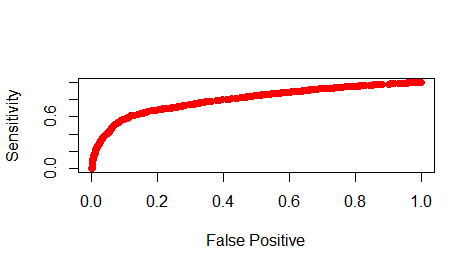
Thus the model (Model 3) we get is on performing logistic regression and retaining variables with significance level of \*\*\* and \*\*:

**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.



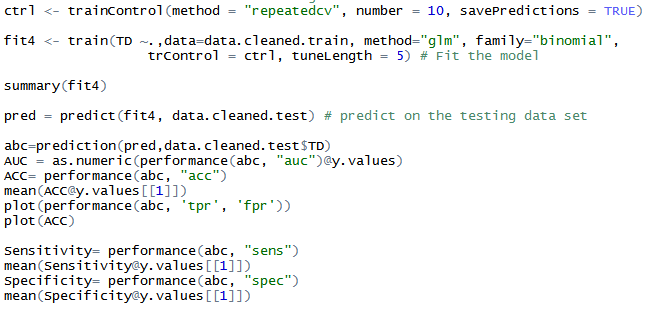
The specificity is 0.996485, sensitivity is 0.12993 and the false positive value is 0.0035149. The ROC curve obtained is show below. The AUC value is 0.7999.

 ****

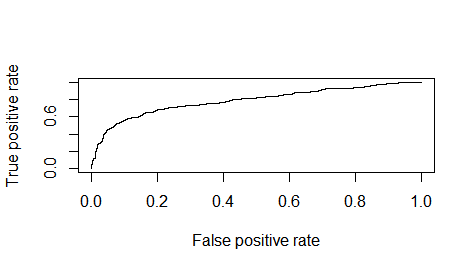
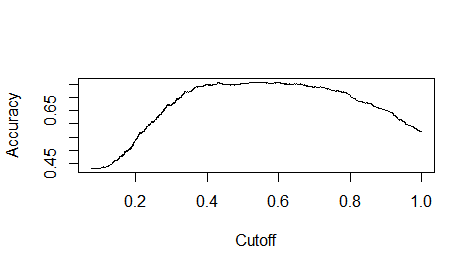
So far we have seen three models in which the training set constituted of 4000 data points and the testing set constitutes of 1000 data points. There was no Cross Validation done. The next three models have cross validation done in order to improve the training of the model and thereby reduce overfitting.

* Model 4:

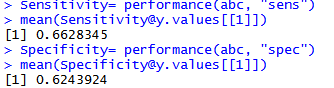
Here I basically use the same model as Model 1 but with cross validation done. I do a 10 fold cross validation. The code snippet is as shown below.



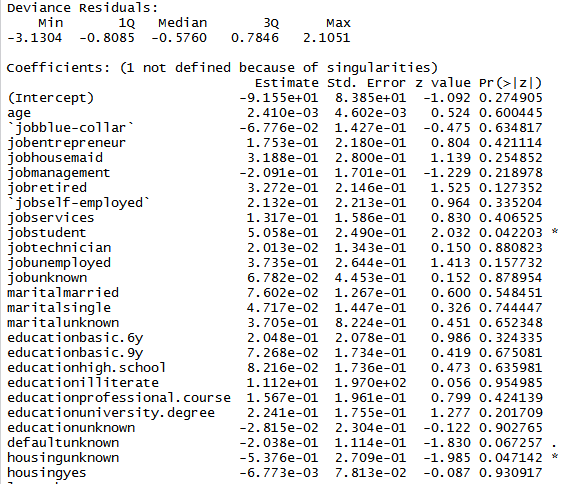
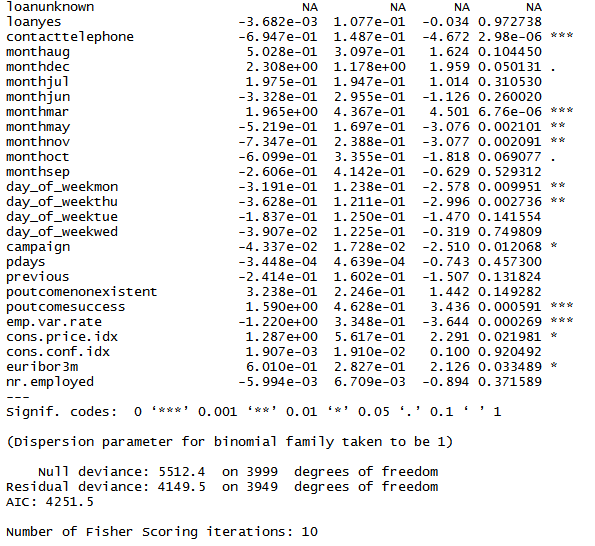
The AUC value is 0.787322. The mean accuracy is 0.641 while the max accuracy obtained is 0.757.

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,



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

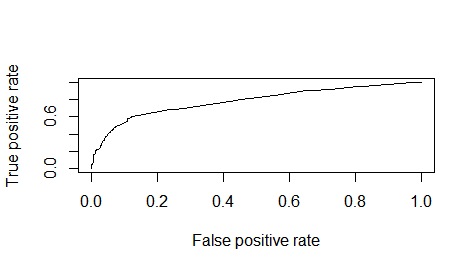
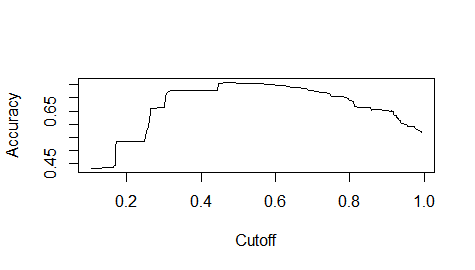
**TD = - 9.155e+01 + sum across all variables (Corresponding Coeff \* Correspoding Variable)**

* Model 5:

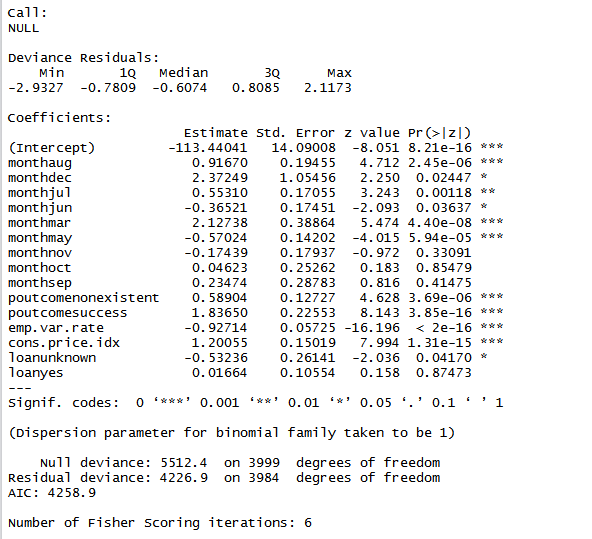
Here I basically use the same model as Model 2 but with cross validation done. I do a 10 fold cross validation. The predictor variables that are chosen here are the same as that of Model 2.

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,

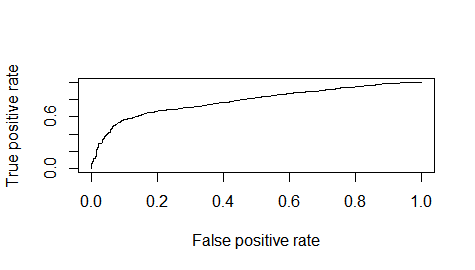
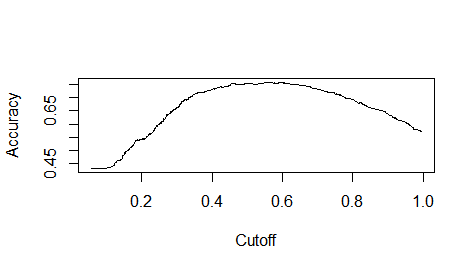


**TD = -113.44 + 0.9167\*monthaug + 2.127\*monthmar - 0.57\*monthmay + 0.589\*poutcomenonexistent + 1.8365\*poutcomesuccess - 0.927\*emp.var.rate + 1.2\*cons.price.idx**

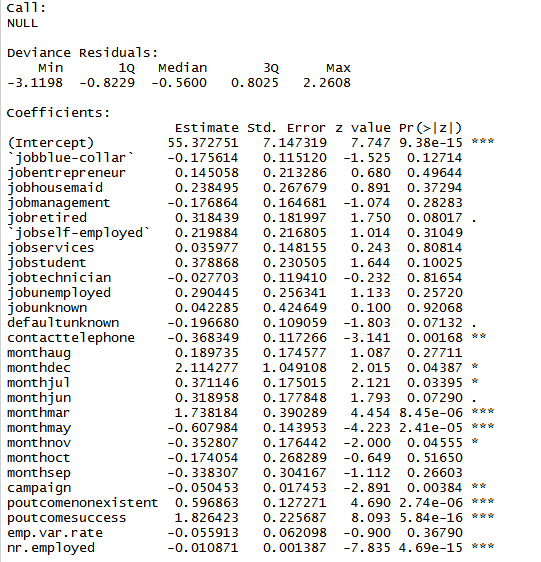
* Model 6:

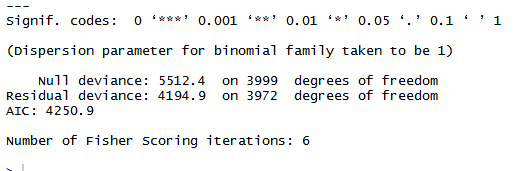
Here I basically use the same model as Model 3 but with cross validation done. I do a 10 fold cross validation. The predictor variables that are chosen here are the same as that of Model 3.

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,



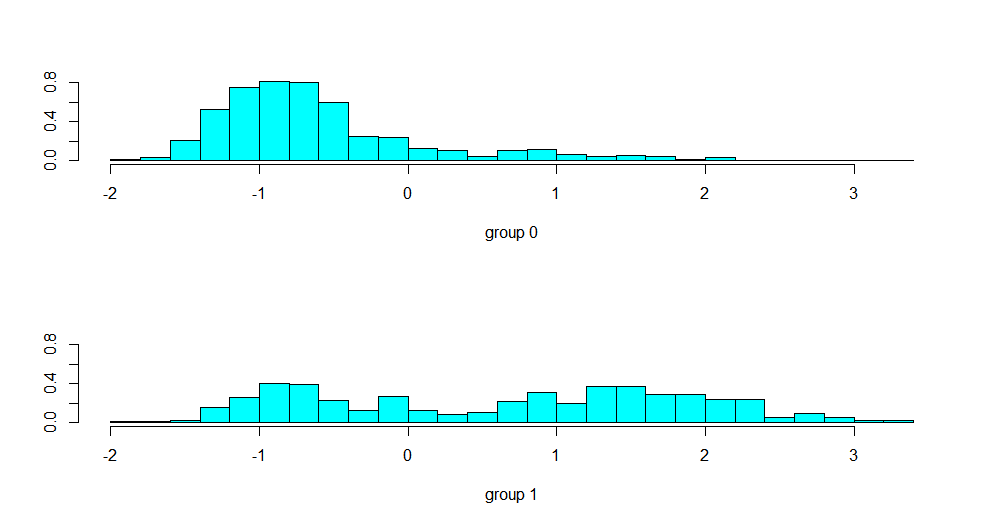


**TD = 10.48 + 0.196\*monthmar - 0.1372\*monthmay + 0.12037\*poutcomenonexistent + 0.25055\*poutcomesuccess - 0.00196\*nr.employed - 0.543\*contacttelephone - 0.097\*monthnov - 0.0311\*emp.var.rate - 0.0081\*campaign**

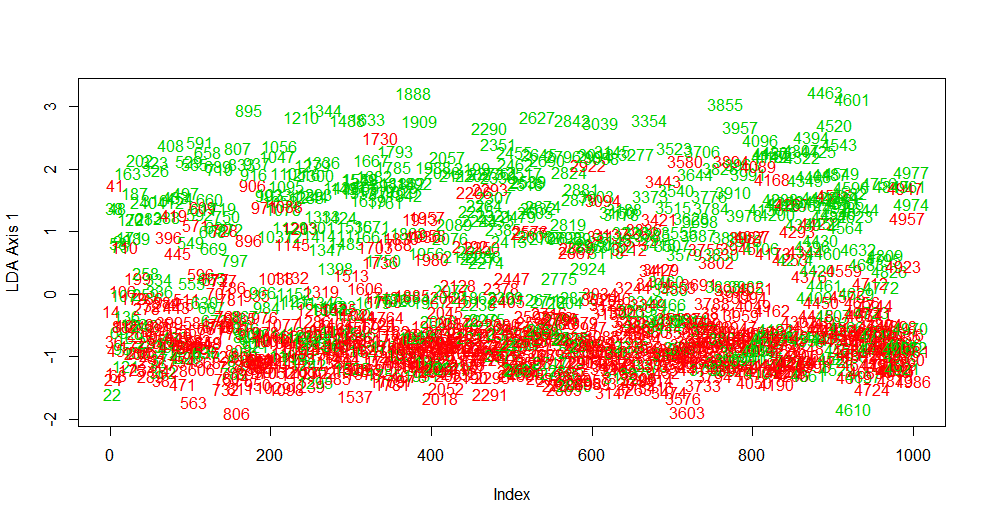
* Model 7:

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).

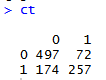
The histogram of the discriminant functions is shown below.



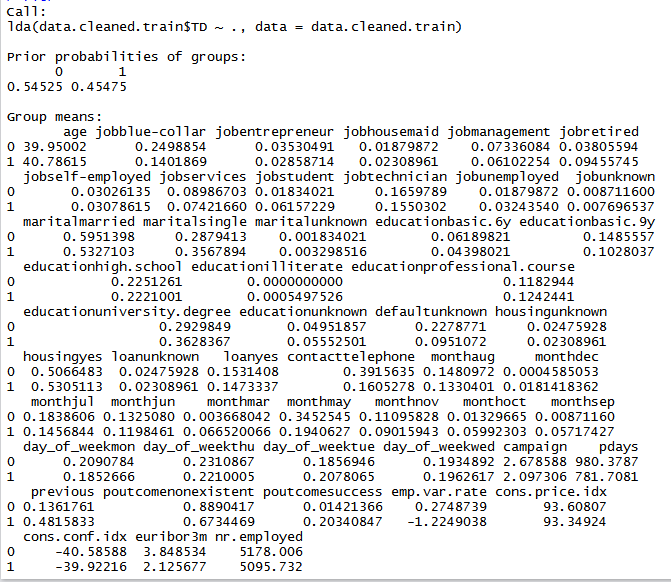
The scatter plot of the discriminant function values is,



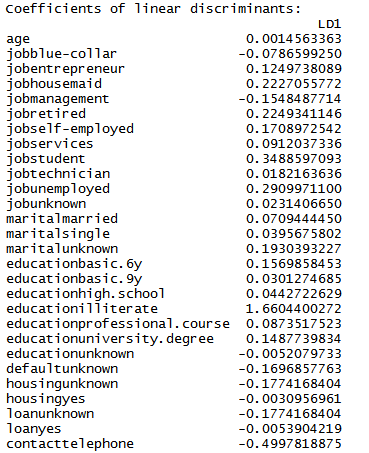
As can be seen we can separate the data along LD1 fairly well. The confusion matrix post prediction on test data is obtained as,

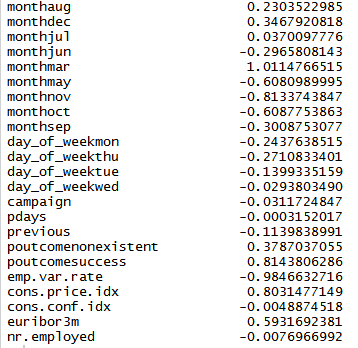


The MCE is 0.754. Clearly, this method does not do as well with high miss-classifications. The group means obtained are as follows,



The LD1 values are as follows,





Hence the model we get is

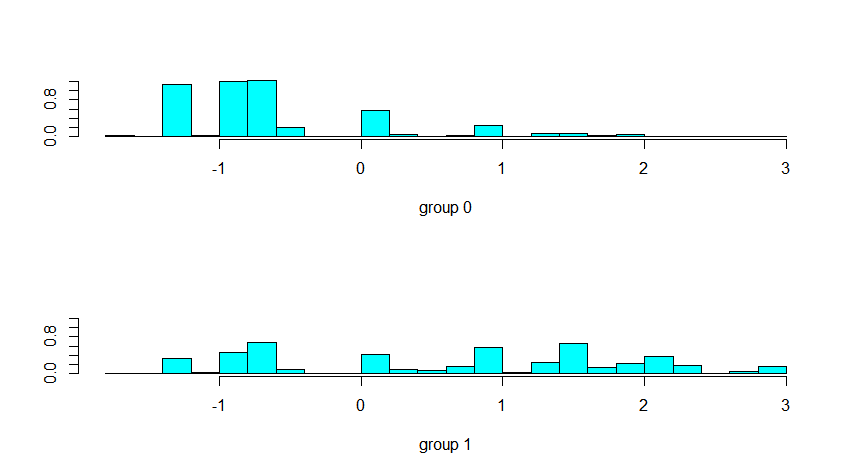
**TD= sum across all the variables (Variable \* Corresponding LD1)**

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

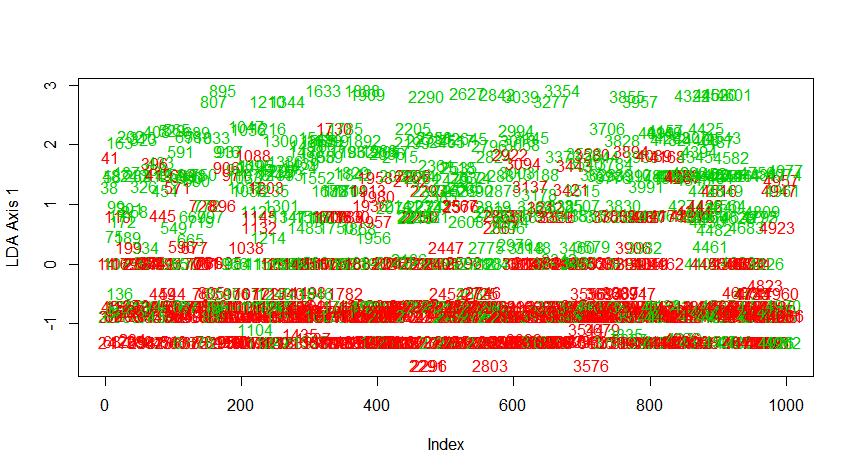
* Model 8:

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).

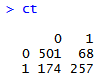
The histogram of the discriminant functions is shown below.



The scatter plot of the discriminant function values is,

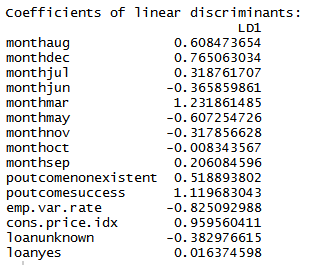


The data separation in this case along LD1 is not as well as the previous case. The confusion matrix post prediction on test data is obtained as,



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,

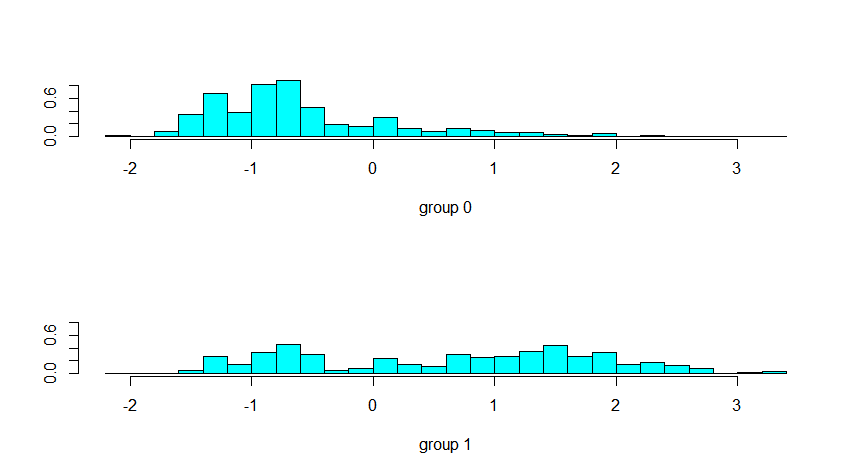


**TD= sum across all the variables above (Variable \* Corresponding LD1)**

* 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).

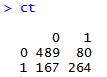
The histogram of the discriminant functions is shown below.



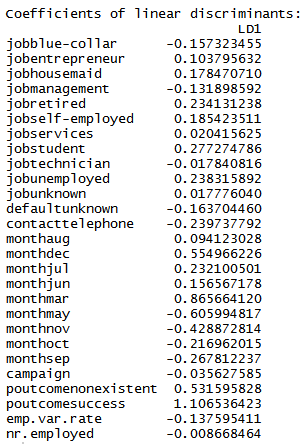
The scatter plot of the discriminant function values is,



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,



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,



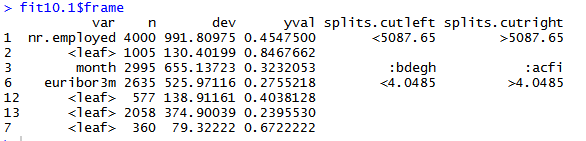
**TD= sum across all the variables above (Variable \* Corresponding LD1)**

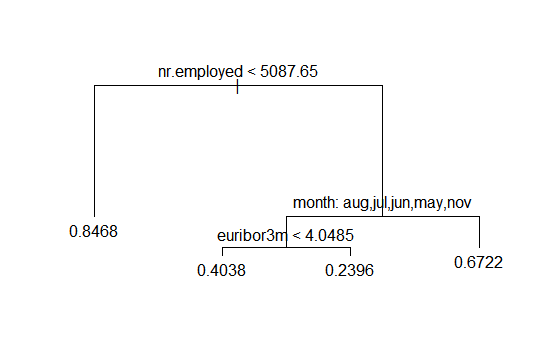
What can be summarized out of the last 3 models that we obtain from LDA is that the MCE has significantly higher as compared to logistic regressions. The models don’t do well to classify the dataset.

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

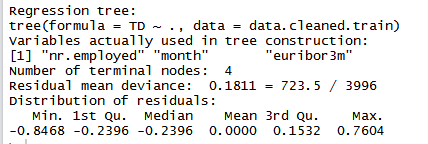
* Model 10:

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.





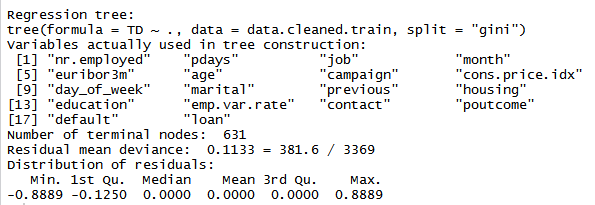
There are 3 variables that the tree is split on and it leads to 4 leaves. Basically they are the mean values in the 4 regions the entire data set get divided into. The summary of this tree is given below,



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

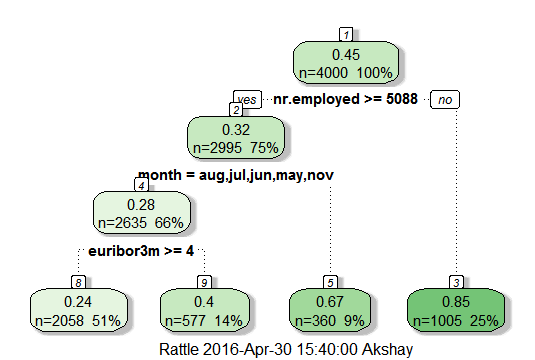


The summary of this tree is as shown below,

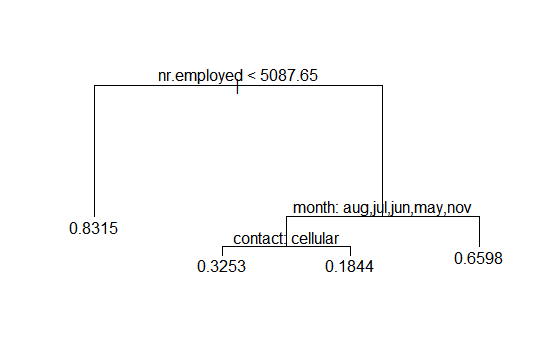


As can be seen in the gini split, the number of tree leaves are far more and also the residual mean deviance is lesser than when the split occurs on deviance.

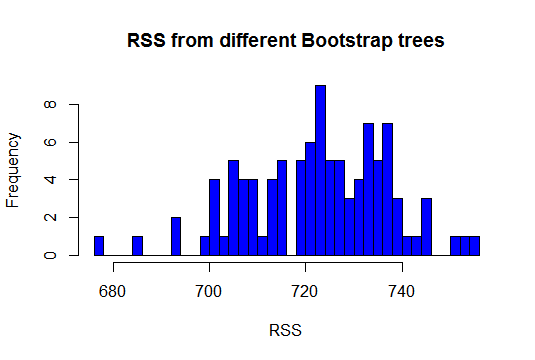
A fancier version of the plot is shown below. Here the number of observation in a particular node and its accuracy is shown.



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.

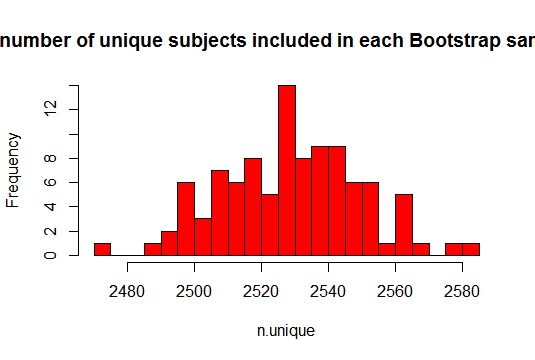


A plot of the RSS that we get across the Bootstrap samples is shown by the below histogram.



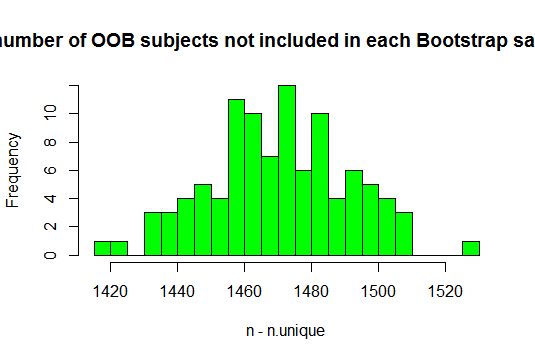
Here it can be seen that the RSS varies from 680 to 740.

The number of unique data points in each bootstrap sample is shown in the below histogram.



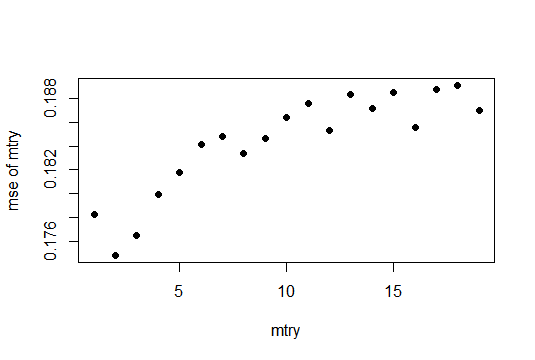
This shows that anywhere between 2480 and 2580 samples are unique out of 4000, while the rest are repeated.

Number of Out of Bag (OOB) samples not included in each Bootstrap sample is shown below.

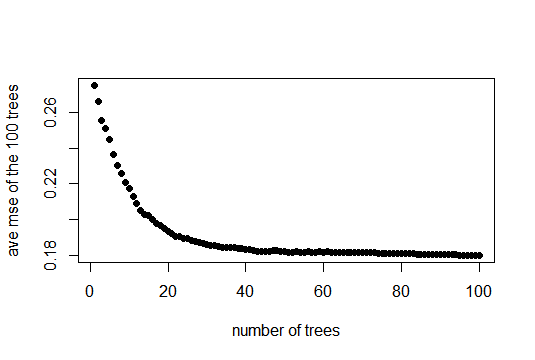


Basically while bootstrapping two thirds of the observations are used to make the tree. The rest one third are out of bag samples. The response of the i’th observation in the OOB was predicted using each of the tree in which that observation was OOB. This meant that I got roughly around B/3 responses 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.

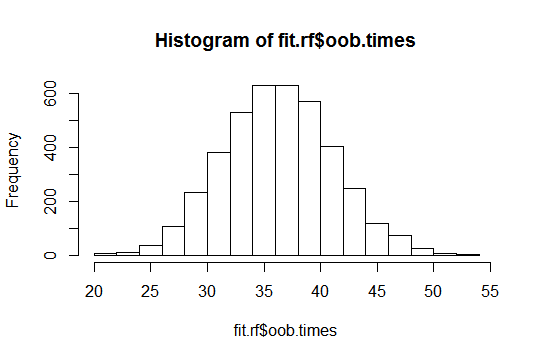
Additionally to improve performance, I built random forest on top of each bootstrapped sample. The effect of mtry parameter, which is the number of random split at each leaf can be seen below.



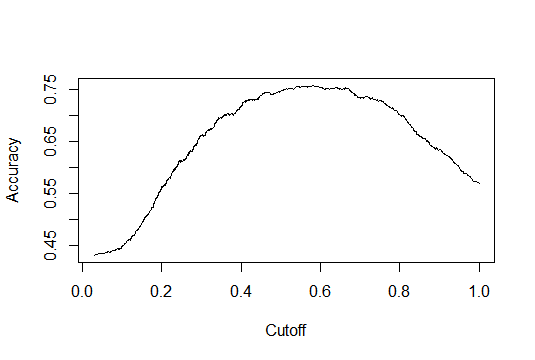
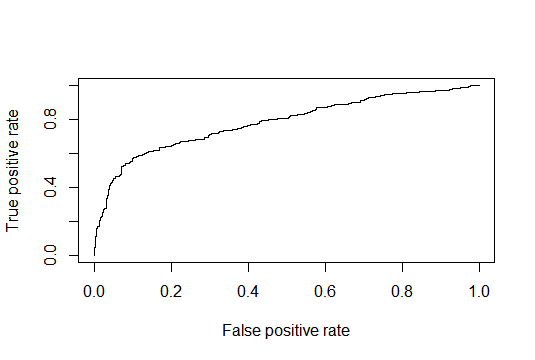
As mtry increases the MSE also increases. Hence I used a value like mtry =4 (square root of 20 roughly) to obtain the Random Forest. The average MSE that I get for 100 trees is shown. Clearly, more the number of trees, less is the MSE.



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



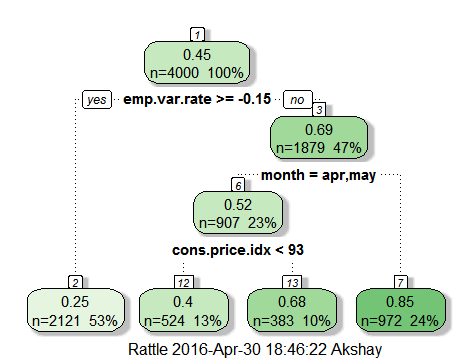
The training error I obtain is 0.18 (18%). This means a mean accuracy of 82% on the training dataset (OOB accuracy). Upon prediction using the test data, I got the mean test accuracy of 63.81% and the max test accuracy to be 75.8%. The accuracy curve on the test data is given below. The ROC curve is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

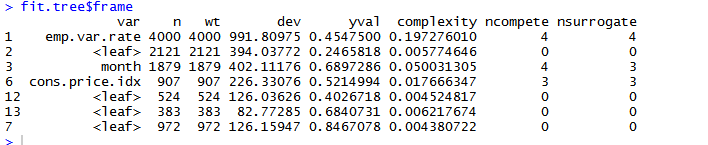
 

The mean sensitivity which is the probability of a point being classified as positive when it is indeed positive is 0.6626 while the specificity which is the probability of a point being classified as negative when it is indeed negative is 0.6196.

* Model 11:

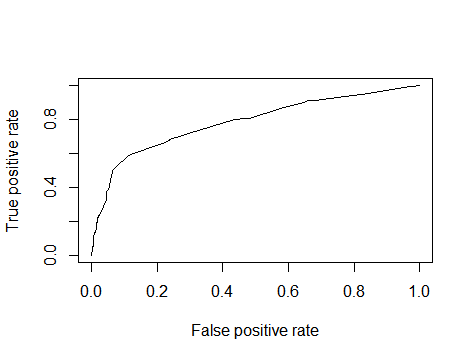
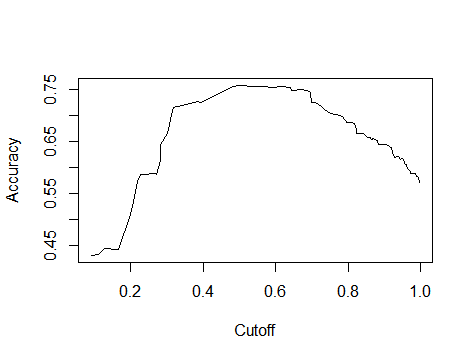
This is a tree based approach on all the variables that I got from regsubsets. The frame showing the splits and the tree itself is shown below. Note I use the default deviance split here on.





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.

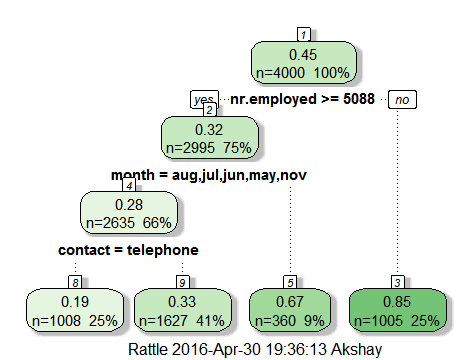
The training error I obtain is 0.17738 (17.738%). This means a mean accuracy of over 82% on the training dataset (OOB accuracy). Upon prediction using the test data, I got the mean test accuracy of 64.85377% and the max test accuracy to be 75.8%. The accuracy curve on the test data is given below. The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

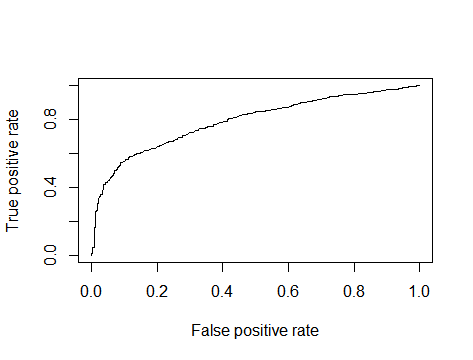
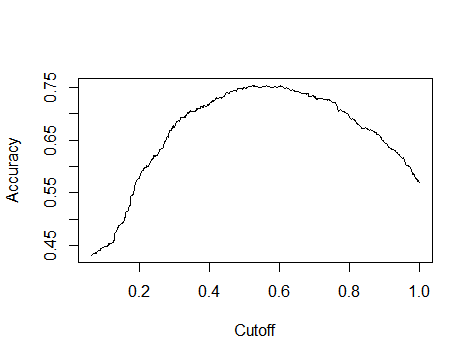
* Model 12:

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.

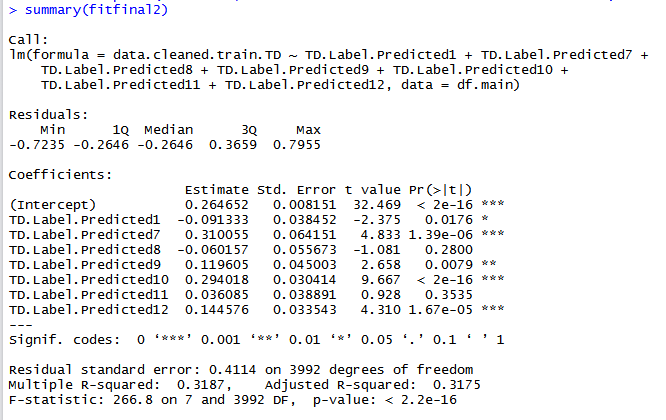
The mean sensitivity which is the probability of a point being classified as positive when it is indeed positive is 0.6146369 while the specificity which is the probability of a point being classified as negative when it is indeed negative is 0.6629753.

* 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.



**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

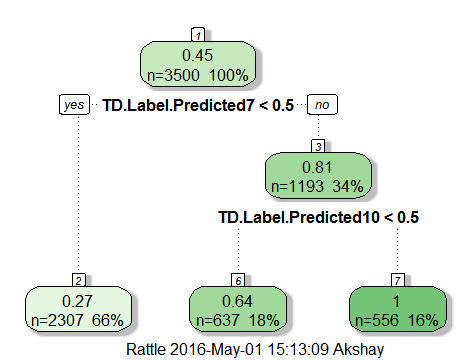
The above is the mean MCE. The least MCE (Max Accuracy) computed is around 75% for the test data set and 82% for the training data set. LDA doesn’t perform well as compared to Logistic Regression. Cross Validation also surprisingly does not go a long way in improving accuracy. Random Forest performs 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.

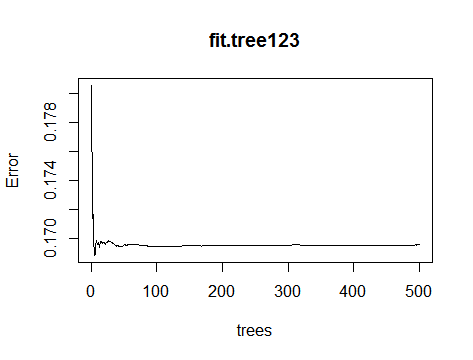
Data.cleaned.train.TD = 0.264652 – 0.091333\*TD.Label.Predicted1 + 0.31\* TD.Label.Predicted7 – 0.060157\* TD.Label.Predicted8 + 0.1196\* TD.Label.Predicted9 + 0.294\* TD.Label.Predicted10 + 0.0361\* TD.Label.Predicted11 + 0.144576\* TD.Label.Predicted12

So then the final model!!

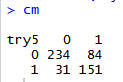
Random Forest did the best of all the models. And hence I used the prediction data frame and performed a Random Forest on them. I again divided the data frame into training and testing and performed my analysis. I then fit a random forest tree (fit.tree123 in code) on my training data set shown below,



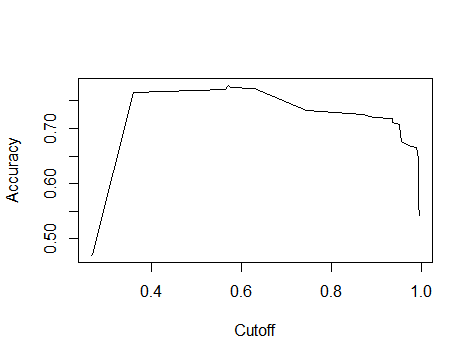
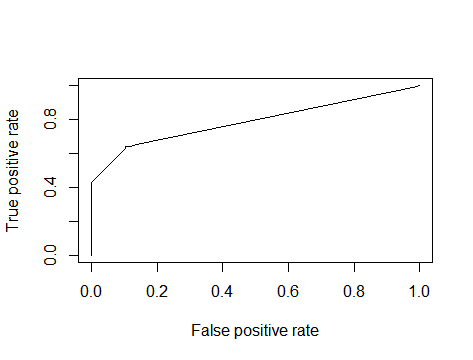
The plot of the final tree is as follows:



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



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))*

*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]])*