SrinivasRengarajan\_DA\_Part1

SrinivasRengarajan\_R00183599

30/11/2019

\*The training and scoring data set has been loaded from the excel file using readxl package’s read\_xlsx function.

\*The read tibbles are then casted into data frames using data.frame() for automatic cleaning of the column names or the variable names from a non-standard name into a standard one (removing spaces by replacing with dots,special characters).

\*The missing value columns are found then (Housing, Employment and Personal.Status) and the number of missing values in each column has been obtained. Considering the proportion of missing values in each of the above variable (not more than 6.5% of the total records) and it’s type (categorical), mode imputation is chosen (as in this case, I believe, it won’t introduce much bias to the model) to deal with the missing values.

\*The categorical variables are then factorized in the training and scoring set using factor() functio with lapply() function

\*A seperate column or variable has been created called Credit.Standing.Class which has either 0 or 1 according to whether the Credit.Standing is bad or good. This variable is used as the response variable in Gradient Boosting model as this algorithm works only with numbers or regression kind of problems.

\*The entire structure of the training and scoring set has been verified after the data cleaning process using the str function.

## -- Attaching packages ------------------------------------------------------------------------ tidyverse 1.3.0 --

## v ggplot2 3.2.1 v purrr 0.3.3  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts --------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

## The columns which contain missing values:

## [1] "Employment" "Personal.Status" "Housing"

## Number of missing values in Housing:

## [1] 5

## Number of missing values in Employment:

## [1] 33

## Number of missing values in Personal.Status:

## [1] 6

## Structure of Training Dataset

## 'data.frame': 780 obs. of 15 variables:  
## $ ID : num 1 2 3 4 5 6 7 8 9 10 ...  
## $ Checking.Acct : Factor w/ 4 levels "0Balance","High",..: 4 1 1 1 4 3 1 3 1 4 ...  
## $ Credit.History : Factor w/ 5 levels "All Paid","Bank Paid",..: 1 4 4 4 1 4 1 2 1 5 ...  
## $ Loan.Reason : Factor w/ 10 levels "Business","Car New",..: 2 2 2 5 10 2 2 4 3 3 ...  
## $ Savings.Acct : Factor w/ 5 levels "High","Low","MedHigh",..: 2 2 5 5 5 4 2 2 2 2 ...  
## $ Employment : Factor w/ 6 levels "Long","Medium",..: 2 4 1 1 1 6 1 2 1 4 ...  
## $ Personal.Status : Factor w/ 3 levels "Divorced","Married",..: 3 1 1 3 3 1 2 1 3 2 ...  
## $ Housing : Factor w/ 3 levels "Other","Own",..: 2 2 2 2 1 2 2 1 2 3 ...  
## $ Job.Type : Factor w/ 4 levels "Management","Skilled",..: 1 2 2 2 2 4 2 4 2 2 ...  
## $ Foreign.National : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 2 1 2 2 ...  
## $ Months.since.Checking.Acct.opened: num 7 16 25 31 7 13 22 25 25 13 ...  
## $ Residence.Time : num 3 2 2 4 4 2 3 4 4 4 ...  
## $ Age : num 44 28 28 30 35 22 29 33 62 40 ...  
## $ Credit.Standing : Factor w/ 2 levels "Bad","Good": 2 1 1 2 2 2 2 2 2 2 ...  
## $ Credit.Standing.Class : num 1 0 0 1 1 1 1 1 1 1 ...

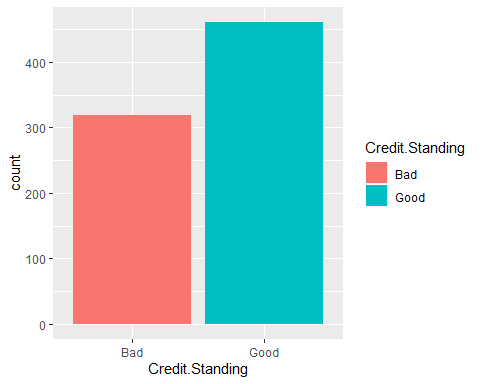
## Structure of Scoring Dataset

## 'data.frame': 13 obs. of 13 variables:  
## $ ID : num 781 782 783 784 785 786 787 788 789 790 ...  
## $ Checking.Acct : Factor w/ 4 levels "0Balance","High",..: 4 3 4 2 3 3 2 3 4 4 ...  
## $ Credit.History : Factor w/ 3 levels "All Paid","Critical",..: 1 3 3 3 3 3 3 1 3 3 ...  
## $ Loan.Reason : Factor w/ 6 levels "Business","Car New",..: 2 6 6 1 6 5 6 4 6 4 ...  
## $ Savings.Acct : Factor w/ 4 levels "High","Low","MedHigh",..: 3 2 2 2 2 4 2 1 2 4 ...  
## $ Employment : Factor w/ 5 levels "Long","Medium",..: 3 2 5 2 2 4 1 5 5 2 ...  
## $ Personal.Status : Factor w/ 3 levels "Divorced","Married",..: 3 3 1 3 1 3 3 1 2 1 ...  
## $ Housing : Factor w/ 3 levels "Other","Own",..: 3 3 2 3 2 1 2 2 2 3 ...  
## $ Job.Type : Factor w/ 4 levels "Management","Skilled",..: 4 2 2 2 4 3 2 2 2 4 ...  
## $ Foreign.National : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 2 2 2 2 ...  
## $ Months.since.Checking.Acct.opened: num 11 37 13 16 9 49 37 12 19 16 ...  
## $ Residence.Time : num 2 4 2 4 2 4 2 1 4 4 ...  
## $ Age : num 39 23 28 25 43 39 30 19 38 32 ...

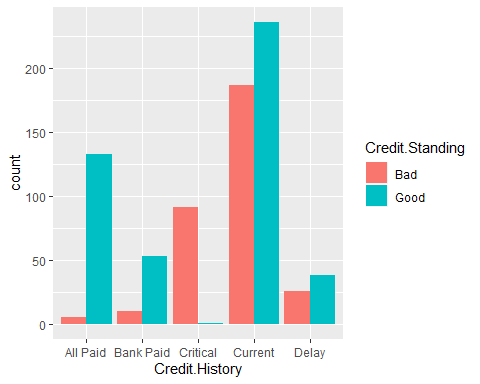
1. EDA

## Bad Good  
##   
## Business Long 0.50 0.50  
## Medium 0.05 0.95  
## Retired NaN NaN  
## Short 0.50 0.50  
## Unemployed 0.50 0.50  
## Very Short 0.38 0.62  
## Car New Long 0.43 0.57  
## Medium 0.39 0.61  
## Retired 1.00 0.00  
## Short 0.55 0.45  
## Unemployed 0.22 0.78  
## Very Short 0.34 0.66  
## Car Used Long 0.38 0.62  
## Medium 0.19 0.81  
## Retired NaN NaN  
## Short 0.38 0.62  
## Unemployed 0.38 0.62  
## Very Short 0.67 0.33  
## Education Long 0.42 0.58  
## Medium 0.00 1.00  
## Retired NaN NaN  
## Short 0.68 0.32  
## Unemployed 1.00 0.00  
## Very Short 0.50 0.50  
## Furniture Long 0.31 0.69  
## Medium 0.33 0.67  
## Retired 0.00 1.00  
## Short 0.70 0.30  
## Unemployed 0.33 0.67  
## Very Short 0.42 0.58  
## Large Appliance Long NaN NaN  
## Medium 0.00 1.00  
## Retired NaN NaN  
## Short 0.75 0.25  
## Unemployed 0.00 1.00  
## Very Short NaN NaN  
## Other Long 0.25 0.75  
## Medium 0.00 1.00  
## Retired NaN NaN  
## Short 0.60 0.40  
## Unemployed 0.00 1.00  
## Very Short NaN NaN  
## Repairs Long 0.00 1.00  
## Medium 0.00 1.00  
## Retired NaN NaN  
## Short 0.00 1.00  
## Unemployed 0.00 1.00  
## Very Short 0.67 0.33  
## Retraining Long NaN NaN  
## Medium 0.00 1.00  
## Retired NaN NaN  
## Short NaN NaN  
## Unemployed NaN NaN  
## Very Short NaN NaN  
## Small Appliance Long 0.20 0.80  
## Medium 0.22 0.78  
## Retired NaN NaN  
## Short 0.52 0.48  
## Unemployed 0.30 0.70  
## Very Short 0.45 0.55

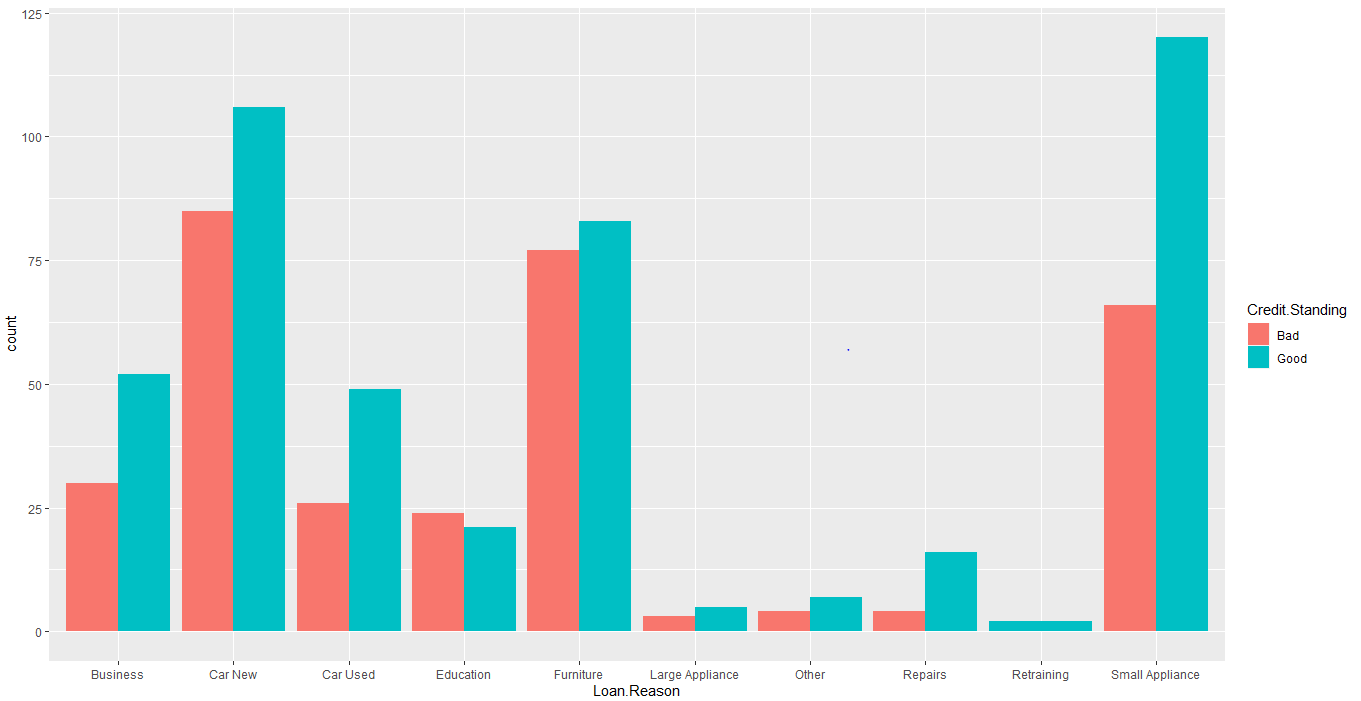
Analysing LoanReason, Employment Type and Credit Standing variable using ftable,it could be observed that People who got educational loan and are unemployed have a bad Credit Standing.Similarly, people who got new car loan and were retired had only bad Credit Standing.



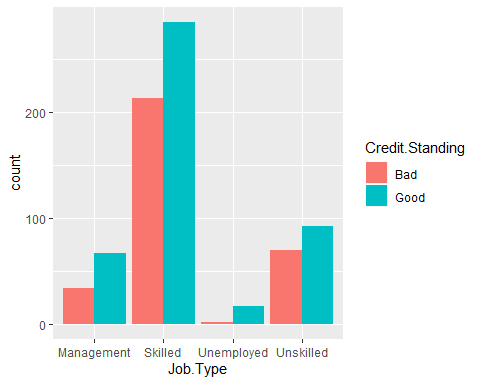
Visualising the proportions of good and bad credit standing, more than 50% of the customers in Kate’s financial institution had only good loans.



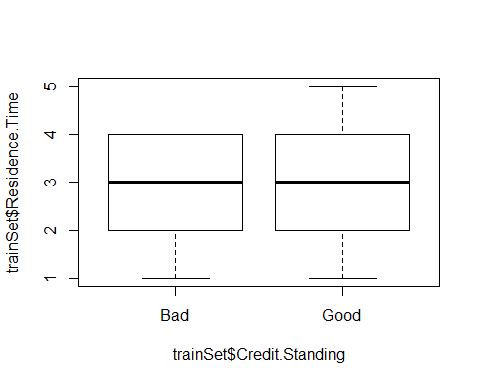
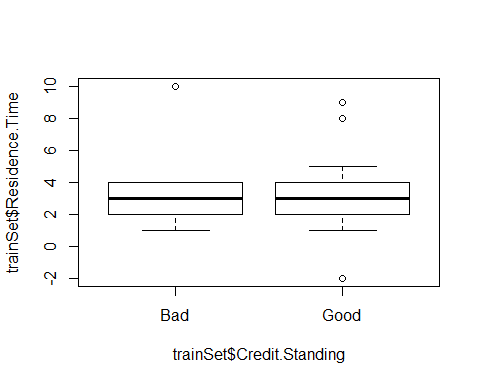
Examining the relationship between Credit.Standing and Credit.History, People having Critical Credit History mostly had a bad Credit Standing



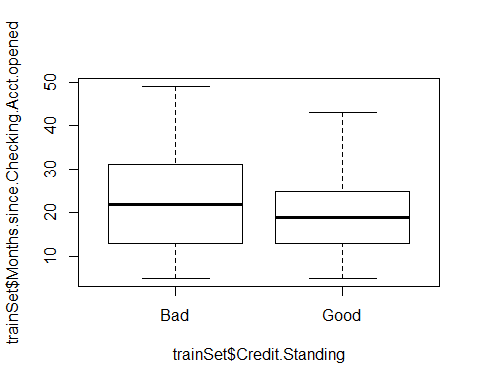
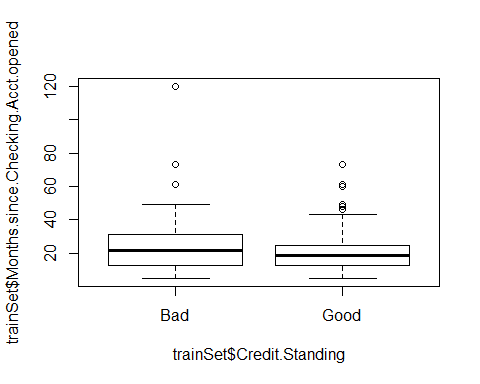
Examining the relationship between Loan.Reason and Credit.Standing, Only people who have got loans for educational reason has a higher percentage of bad Credit Standing than good Credit.Standing.



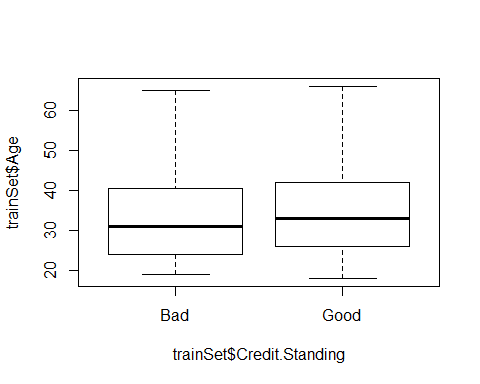
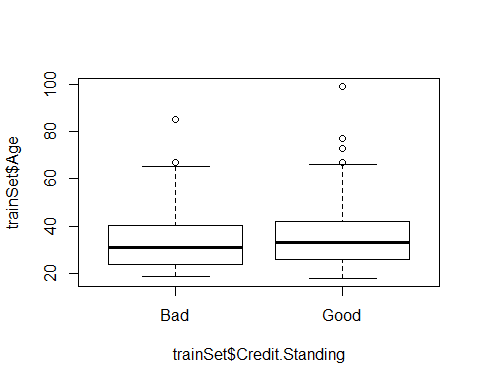
Examining the relationship between Job.Type and Credit.History, There is not a huge relationship between them as in all job types, the number of good and bad loans don’t have a huge difference.



Boxplot for Credit Standing against Residence time depicts no significant relationship as there is no difference in medians. There were a few outliers in the residence time data.



Boxplot for Credit Standing against Months.since.Checking.Acct.opened depicts no significant difference in medians.There were only a few outliers in the data.



Boxplot for Credit Standing against Age has no difference in the medians. There were a few outliers in the Age data.

##   
## Pearson's Chi-squared test  
##   
## data: trainSet$Foreign.National and trainSet$Credit.Standing  
## X-squared = 0.15486, df = 1, p-value = 0.6939

##   
## Pearson's Chi-squared test  
##   
## data: trainSet$Credit.History and trainSet$Credit.Standing  
## X-squared = 225.67, df = 4, p-value < 2.2e-16

##   
## Pearson's Chi-squared test  
##   
## data: trainSet$Personal.Status and trainSet$Credit.Standing  
## X-squared = 7.1355, df = 2, p-value = 0.02822

##   
## Pearson's Chi-squared test  
##   
## data: trainSet$Housing and trainSet$Credit.Standing  
## X-squared = 9.2181, df = 2, p-value = 0.009961

Analysing the relationship between two categorical variables using chi-square test learnt in the stats module

Performing chi-square test to analyse the reltionship between Foreign National and Credit.Standing

Null Hypothesis: No relationship exists between Foreign National and Credit Standing Alternate Hypothesis: Foreign National and Credit Standing has relationship between them

p-value=0.6939 which is greater than 0.05(5% significance level), so failed to reject the null hypothesis. Therefore, there is not enough evidence to say that there is a relationship between Foreign National and Credit Standing.

Performing chi-square test to analyse the reltionship between Credit.History and Credit.Standing

Null Hypothesis: No relationship exists between Credit.History and Credit Standing Alternate Hypothesis: Credit.History and Credit Standing has relationship between them

p-value<2.2\*10^-16 which is very less than 0.05(5% significance level), so rejecting the null hypothesis. Therefore, with 95% confidence, we could say that there is a strong relationship between Credit.History and Credit Standing

Performing chi-square test to analyse the reltionship between Personal.Status and Credit.Standing

Null Hypothesis: No relationship exists between Personal.Status and Credit Standing Alternate Hypothesis: Personal.Status and Credit Standing has relationship between them

p-value is 0.02822 which is lesser than 0.05,so rejecting the null hypothesis but the significance of relationship is poor. i.e, not strongly correlated.

Performing chi-square test to analyse the reltionship between Housing and Credit.Standing

Null Hypothesis: No relationship exists between Housing and Credit Standing Alternate Hypothesis: Housing and Credit Standing has relationship between them

p-value is 0.0099 which is lesser than 0.05,so rejecting the null hypothesis but the significance of relationship is poor. i.e, not strongly correlated.

EDA Actions & Proposals:

With the help of all the Exploratory Data Analysis done above, we can observe that, Residence.Time,Foreign National,Months.Since.Checking.Acct.Opened,Personal.Status,Job.Type,Housing and Age are not so influential in predicting the result. So,this could introduce noise in the model which might lead to low bias(more accuracy in training) and high variance(bad predictions with unseen data) i.e., overfitting. Thus, not considering those features while building the model.

Splitting the training data into train and validation set for checking the accuracy (70:30 split) on the validation set.Setting the seed to get reproducible results (seed value - last three digits of my ID number-R00183599). Out of 780 records, 546 records is taken for training and 234 records for validation purpose. Accuracies for all the models built below are evaluated on the validation data(234).

1. DecisionTree Model Building using rpart package

Decision tree is a simple but easily interpretable machine learning model. The tree starts splitting from the root node (most important node) and further splits into the internal nodes until it reaches the leaf nodes or label nodes. The nodes chosen for splitting could be found either using information gain or gini index. The variable having the higest information gain is selected as the root node. Info Gain is found by subtracting the entropy (impurity or uncertainity) of the independent variable from the entropy of the target variable. Lower the probablity of a variable, higher the entropy and higher its probability, lesser the entropy.

The accuracy of the decision tree model on the validation set before tuning is (122+44)/234=0.709 i.e., 70.9%

10-fold Cross-Validation is done for hypertuning of the parameters in the decision tree. It is also used to avoid overfitting.(good performance on training and poor performance on unseen data).

1. The training dataset would be randomly shuffled.
2. The training dataset would then be split into 10 equal groups.
3. For each group: That group woud be taken as the test set and the model would be trained on all other groups and tested on this test set. An accuracy is obtained.
4. Compare the accuracies obtained above for getting the optimal model

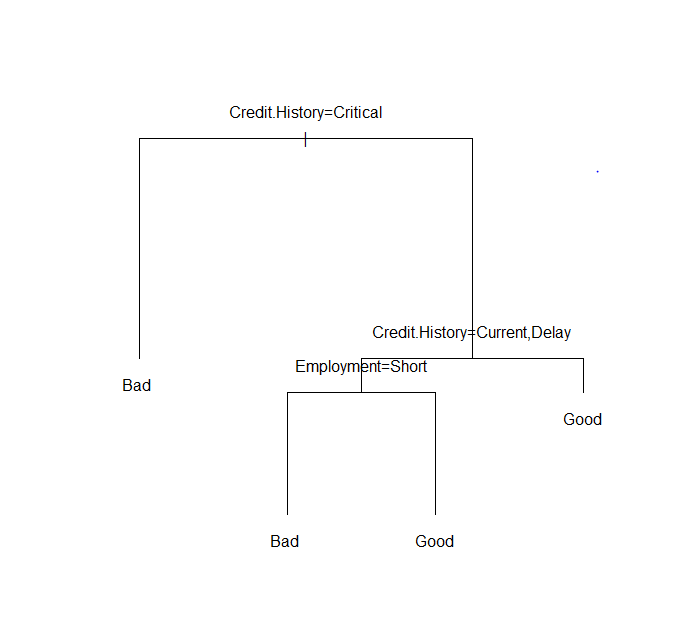
The accuracy of the cross-validated tree on the validation set is 72.6% (126+44)/234.

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

##   
## Bad Good   
## 0.4089744 0.5910256



##   
## prediction Bad Good  
## Bad 44 27  
## Good 41 122

##   
## tree.pred Bad Good  
## Bad 44 23  
## Good 41 126

1. Predicting the unseen data with the decision tree model built.

We take 5 records to explain to Kate, how the decision tree works and predicts the result for the unseen data.

In General, Decision Tree is a basic conditional tree which predicts the class of the record by parsing down the conditions among the variables (features) of a record. If the condition is evaluated to true, the tree goes down the left and if the condition is false, the tree goes down the right. The tree keeps on parsing or going down till it reaches or predicts the class for that record or observation or row.

1. For ID 781,first the decision tree checks for the root node whether the customer’s credit history is critical or not. In this case it’s not critical (i.e., false), so the tree goes down the right hand side and checks whether the credit history is within Current,delay or not. In our case, it’s not (false),so the decision tree goes down the right hand side and predicts this as a Good Loan.
2. For ID 791,the decision tree checks for the root node whether the customer’s credit history is critical or not. In this case it’s critical (i.e., true), so the decision tree goes down the left hand side and predicts this as a Bad Loan.
3. Again, For ID 792,the decision tree checks for the root node whether the customer’s credit history is critical or not. In this case it’s critical (i.e., true), so the decision tree goes down the left hand side and predicts this as a Bad Loan.
4. For ID 788,first the decision tree checks for the root node whether the customer’s credit history is critical or not. In this case it’s not critical (i.e., false), so the tree goes down the right hand side and checks whether the credit history is within Current,delay or not. In our case, it’s not (false),so the decision tree goes down the right hand side and predicts this as a Good Loan.
5. For ID 793,first the decision tree checks for the root node whether the customer’s credit history is critical or not. In this case it’s not critical (i.e., false), so the tree goes down the right hand side and checks whether the credit history is within Current,delay or not. In our case, it’s Current (true),so the decision tree goes down the left hand side and checks whether the Employment is short or not, in our case, it’s true (Short), so it goes down the left hand side and predicts this as a Bad Loan.

The accuracies of these being a good/bad loan is calculated by the use of constructing a confusion matrix. The accuracy is calculated by dividing the sum of True Positives and True negatives by the sum of True Positives,False Positives,False Negatives and True Negatives.

True Positive - Actual and predicted results are the same where the results are positive like Good,Yes… True Negative - Actual and predicted results are the same where the results are negative like Bad,No… False Positive - Actual and predicted results are different where the actual result is negative but the predicted result is positive. False Negative - Actual and predicted results are different where the actual result is positive but the predicted result is negative. This could be more dangerous.

In our case, TP=126,TN=44,FP=23 and FN=41, so accuracy=(126+44)/(126+23+41+44)=170/234=72.64%

## [1] Good Good Good Good Good Good Good Good Good Good Bad Bad Bad   
## Levels: Bad Good

1. Trying to improve my model performance using Random Forest (Bagging), GBM and AdaBoost(Boosting):

Random Forest is an Ensemble technique (combination of two or more models) which uses the principle of Bagging(Bootstrapping and Aggregating) where the dataset is randomly sampled of equal observations and many number of independent trees are grown, all are modelled using decision tree and the maximum accuracy tree is selected from majority voting as this is a classification problem, otherwise results would be averaged (regression problems).

The main parameter is mtry that is the number of features by which each tree should be split at random. We arrive at the best parameter for our combination of predictor variables by running a loop for mtry values ranging from 2 to 5. We obtain good accuracy at mtry=2 comparatively to other values of mtry.

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

## [1] 0.7393162 0.7179487 0.7307692 0.7222222

##   
## randomforestPrediction Bad Good  
## Bad 57 33  
## Good 28 116

Accuracy for randomForest model is (116+57)/234=73.9 percent on the validation set which is better than decision tree which was 72.6%

Boosting:

Using AdaBoost - Ensemble technique where trees (weak learners or decision stumps) are grown sequentially one after the other giving more weight to the misclassified data points from the preceeding trees and reducing the misclassification rate to get a strong learner after multiple rounds of iterations.

The main parameter is mfinal which is the number of iterations for which the boosting has to be run. We got the best accuracy for that comparatively with mfinal = 30.

## Loading required package: foreach

##   
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':  
##   
## accumulate, when

## Loading required package: doParallel

## Loading required package: iterators

## Loading required package: parallel

Unable to display the output for adaboost here as the output is having more than 10 pages. Kindly refer the markdown for the implementation output and accuracy verification.

The Adaboost algorithm has 73.5% (122+50)/234 accuracy on my validation set, which is less than GBM & RandomForest.

Gradient Boosting Machine:

GBM boosting algorithm tries to find the minimum loss function using gradient descent. It uses the learning rate which when given a higher value might skip the global minima or when given a very low value,might increase the accuracy by exactly finding the minima but would take a very long time to find that thereby increasing the time taken to build the model.

A GridSearch has been done to find the best combination of parameters(for which the accuracy will be more comparatively) for gbm algorithm by running the algorithm for every possible combination of parameters supplied in the grid.

Citation: <https://www.rdocumentation.org/packages/NMOF/versions/1.6-0/topics/gridSearch> (For Understanding GridSearch)

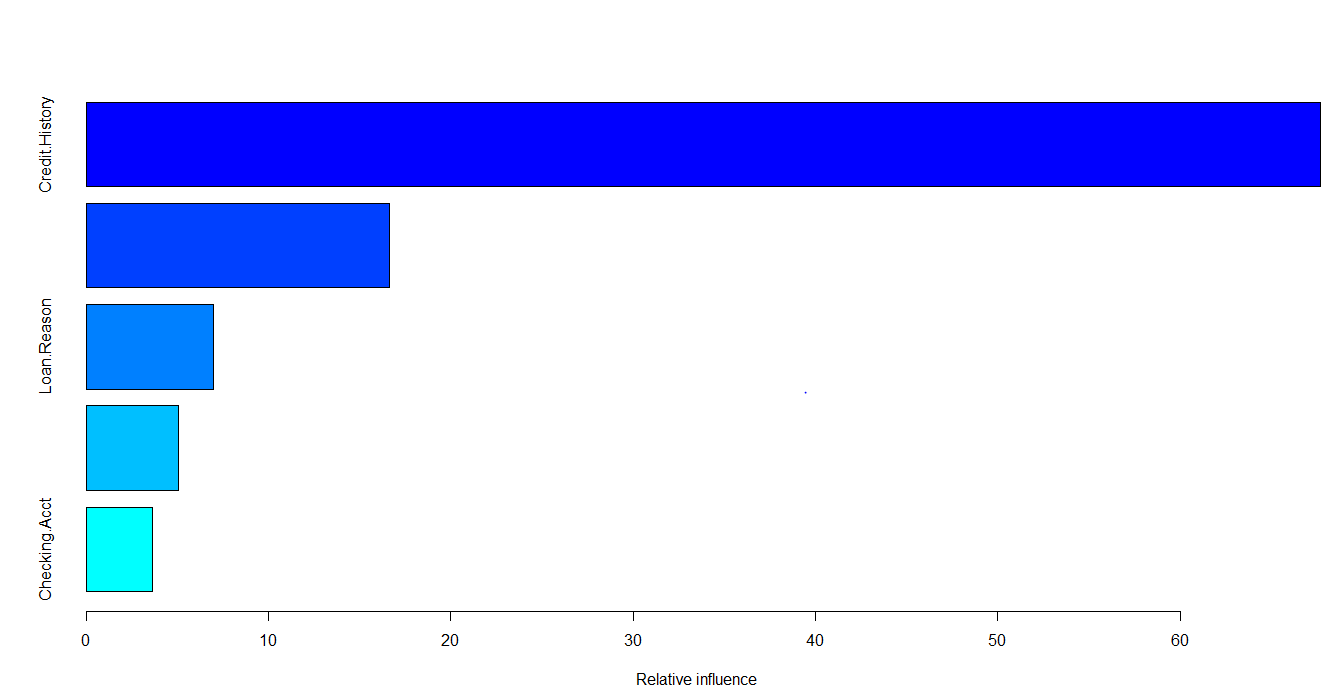
The combination of parameters tried are:

shrinkage = c(.01, .001, .0001), #learning rates interaction.depth = c(1, 3, 5), #maximum depths of each tree n.minobsinnode = c(5, 10, 15), #minimum no of observations in the leaf nodes bag.fraction = c(.5, .7, 1), #fractions of the training data set observations selected for building the #consequent tree n.trees = 5000 #max number of trees to be built

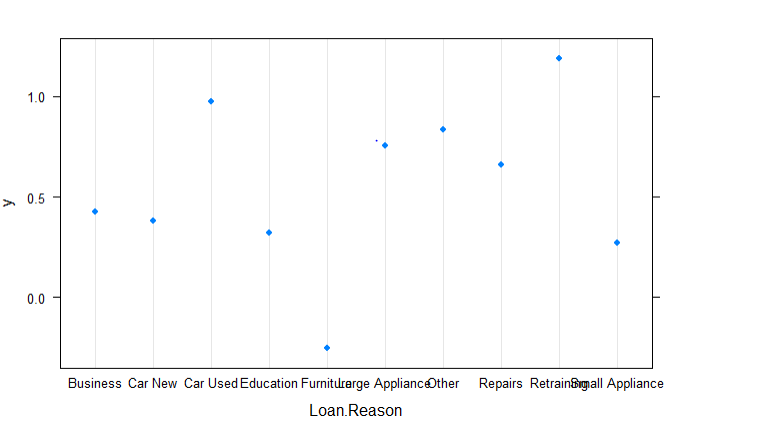
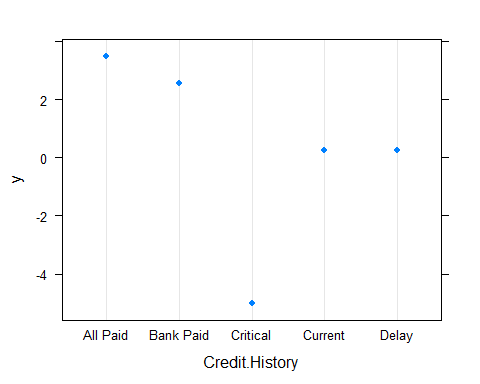
The best combination of parameters found from grid search is learning rate at 0.01, tree depth at 3, no.of trees at 1938 and all the data is taken from the given dataset (bag.fraction is 1). The best combination is the one for the min\_RMSE (Root Mean Squared Error) is minimum. The drawback of gridsearch is its time and cost complexity since it runs through every single possible combination.

## Loaded gbm 2.1.5

## shrinkage interaction.depth n.minobsinnode bag.fraction optimal\_trees  
## 1 0.01 3 5 1 1938  
## 2 0.01 3 10 1 1784  
## 3 0.01 5 5 1 1358  
## 4 0.01 5 10 1 1283  
## 5 0.01 3 15 1 1852  
## min\_RMSE  
## 1 0.3926942  
## 2 0.3951426  
## 3 0.3972441  
## 4 0.3975198  
## 5 0.4016099

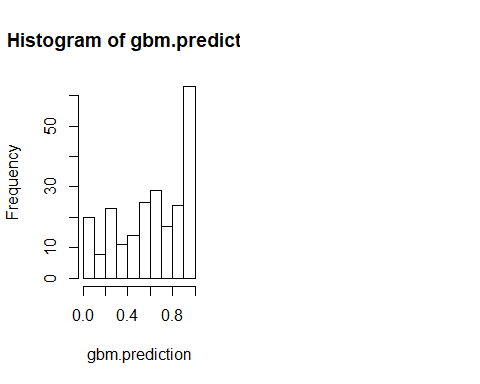


## var rel.inf  
## Credit.History Credit.History 67.707168  
## Employment Employment 16.635469  
## Loan.Reason Loan.Reason 6.968974  
## Savings.Acct Savings.Acct 5.064360  
## Checking.Acct Checking.Acct 3.624028



##   
## predict\_class Bad Good  
## Bad 53 23  
## Good 32 126

## [1] 0.7649573



We get 76.5% accuracy for GBM which is 2.6 % more than random forest and 3% more than Adaboost

Using GBM, we found that Credit.History and Loan.Reason have high relative importance.

i.e.,Higher the proportion of Credit.History being all paid,the higher it is classified as good loan and the higher it is critical,the higher it is classified as bad loan.

Higher the loan reason is for education, the higher it is highly classified as bad whereas the higher it’s for retraining the higher it is classified as good

GBM has 76.5% accuracy i.e, 2.6 % more than random forest and 3% more than Adabag.

With Decision Tree (72.6%), RandomForest (73.9%), Adabag (73.5%) and Gradient Boosting (76.5%) models built,validated and after hyper-tuning the parameters, GBM stands out to be the better model amongst them.

1. To find the suspiciously incorrect pattern in the training set data which contains the details of the customer’s loan, let’s apply the best model obtained above for the training set, predict the labels with that model,compare the results with the actual labels and find any consecutive ID’s or nearly consecutive ID’s of circa 10 or more which shows incorrect patterns. When analysed, the customer ID starting from 304 to 311 which is 8 consecutive customer ID’s had incorrect patterns.

##   
## best.predict\_class Bad Good  
## Bad 229 57  
## Good 90 404

## Suspiciously Incorrect Pattern found in consecutive customer ID's of circa 8 is: 304 305 306 307 308 309 310 311