**Report**

**BY:**

**ASHUTOSH MISHRA**

**16BIT0110**

**Loan Default Prediction on Large Imbalanced Data Using Random Forests**

**INTRODUCTION:**

Predicting whether a borrower would default on his/her loan is of vital importance for bankers as default prediction accuracy will have great impact on their profitability. From the machine learning perspective, loan default prediction (also called credit scoring) can be viewed as a binary classification problem. Previous approaches focus on prediction using ensemble methods or fuzzy systems. Neural networks, successfully applied to various fields, also finds applications in the default prediction problem. Most of them assume that “good” cases and “bad” cases of the data sets are equally distributed or simply ignore the class distribution of data sets. However, most available loan defaults data sets are highly skewed, i.e. there are many cases in one class, few in another. We try to make loan default prediction on imbalanced data sets with an improved random forests approach which employs weighted majority votes in tree aggregation. The weights assigned to each tree in the forest are based on OOB (out-of-bag) errors which are easy to obtain during the forest construction process. Also, due to the fact that random forests can be parallel in nature, we employ the each package in the statistical software R to make random forest parallel and greatly reduce the learning time.

**Random Forest: Overview**

Random Forest is an ensemble learning (both classification and regression) technique. It is one of the commonly used predictive modelling and machine learning technique.

In a normal decision tree, one decision tree is built and in a random forest algorithm number of decision trees are built during the process. A vote from each of the decision trees is considered in deciding the final class of a case or an object, this is called ensemble process. This is a democratic process. Since, many decision trees are built and used in a process of Random Forest algorithm, it is called a forest.

Now, why is it “random”? A data frame (or SAS dataset) has two dimensions - observation (or rows) and variables (or columns). For a building a decision tree, samples of a data frame are selected with replacement along with selecting a subset of variables for each of the decision tree. Both sampling of data frame and selection of subset of the variables are done randomly, so first word “random” is arrived.

Key advantages of using Random Forest

* Reduce chances of over-fitting
* Higher model performance or accuracy

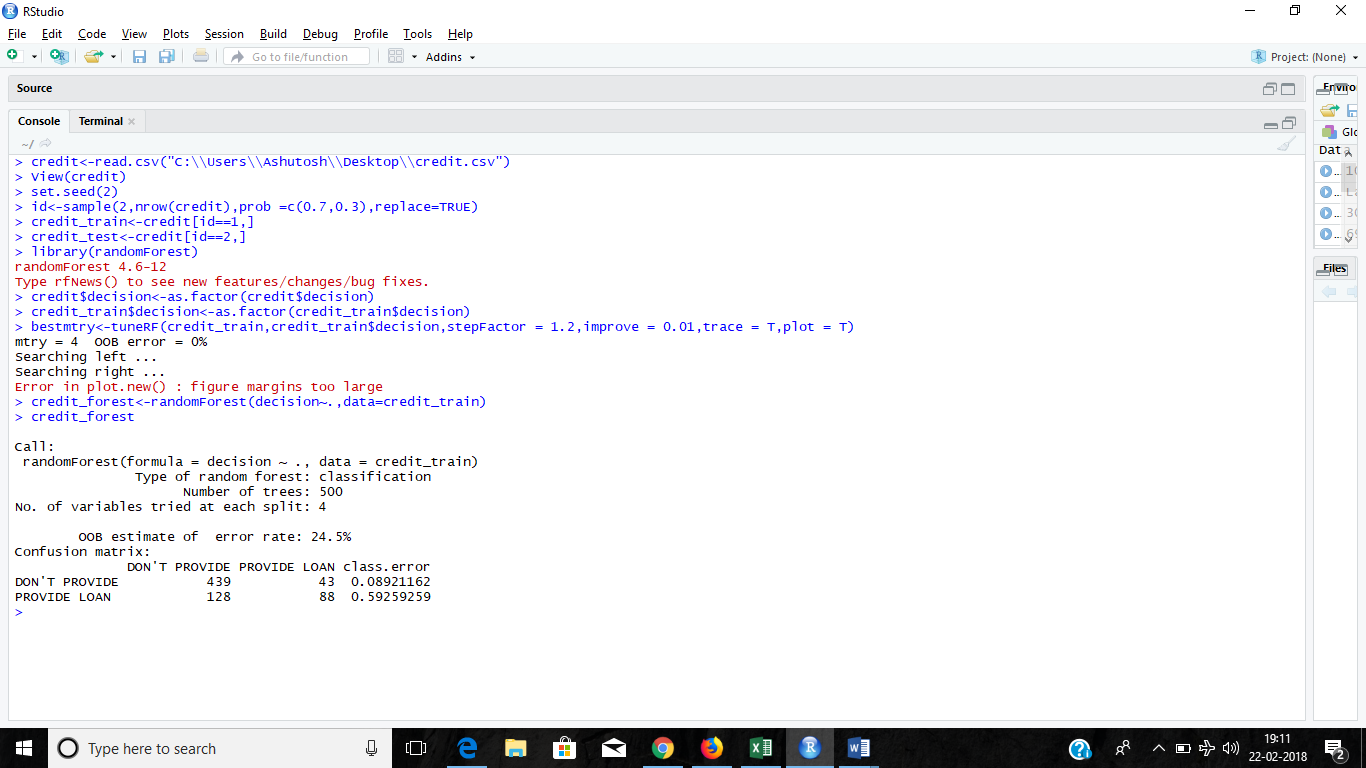
Random Forest uses Gini Index based impurity measures for building decision tree. Gini Index is also used for building Classification and Regression Tree (CART). In earlier blogs we have explained working of CART Decision Tree and a worked out example of Gini Index calculation.

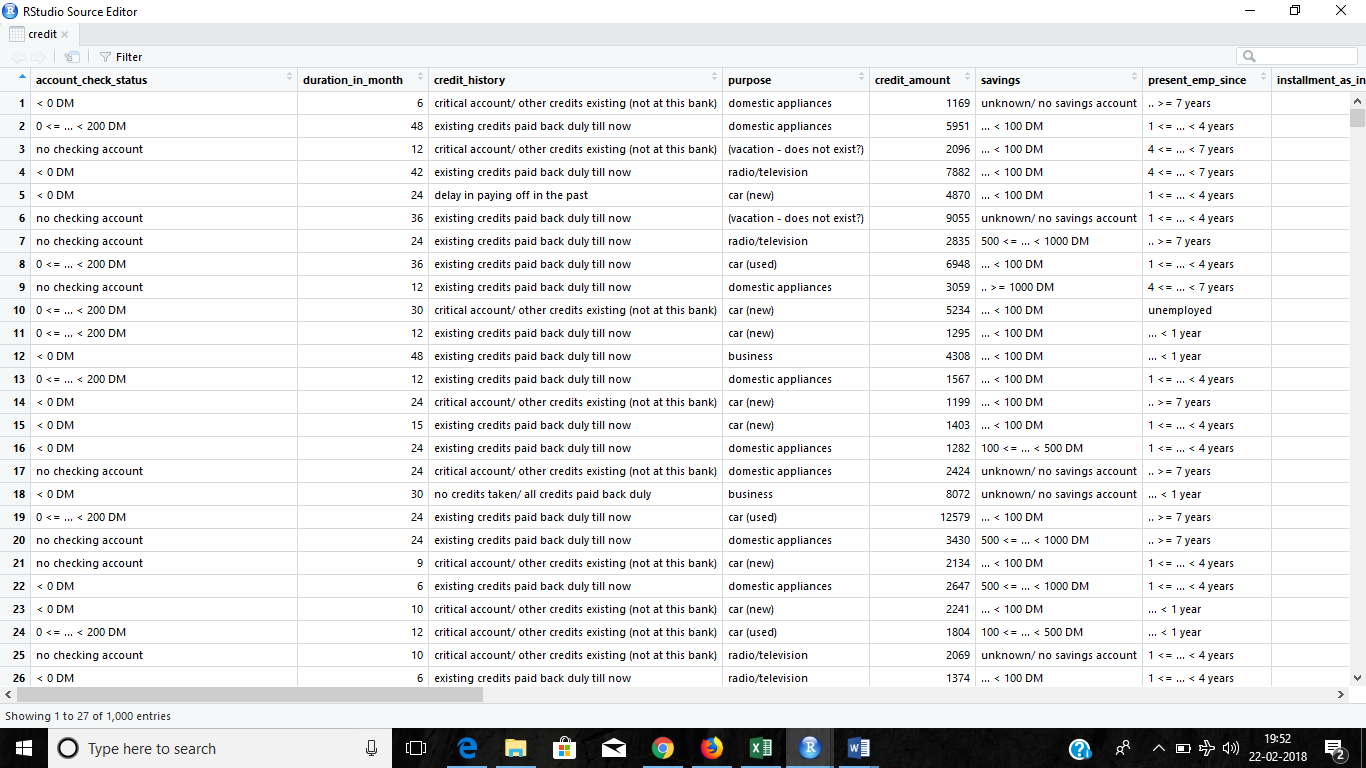
Random Forest algorithm can be used for both classification and regression applications.

**Method:**

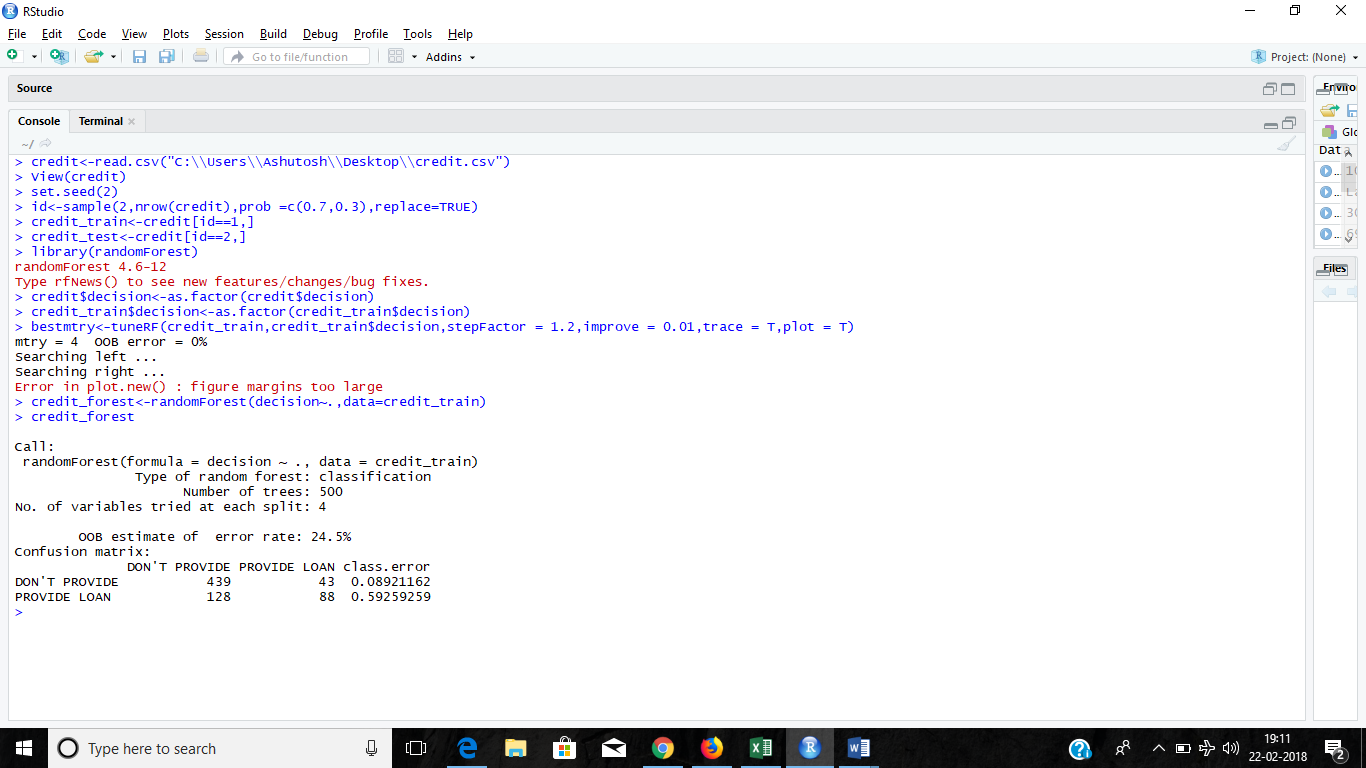
**Data acquisition:**

In this section data will be read by system on which we have to train the system.

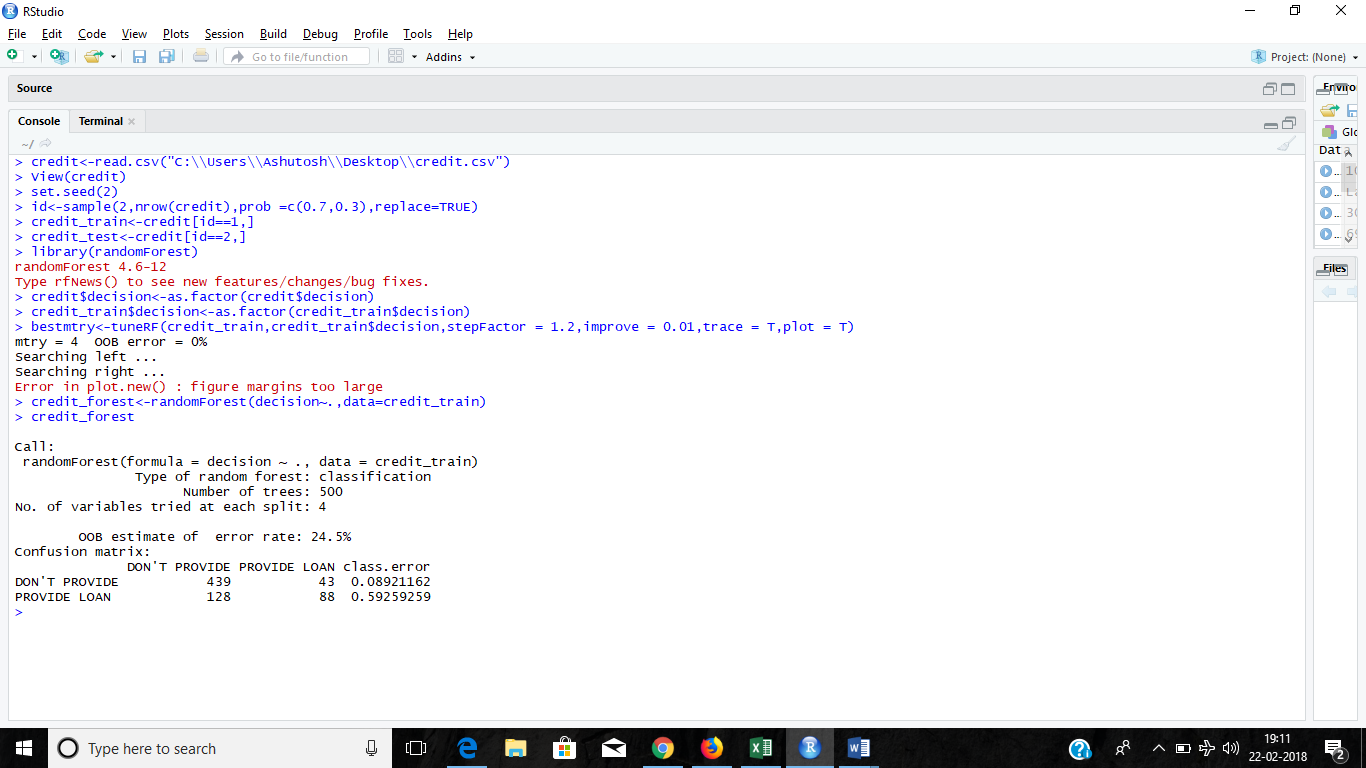
****

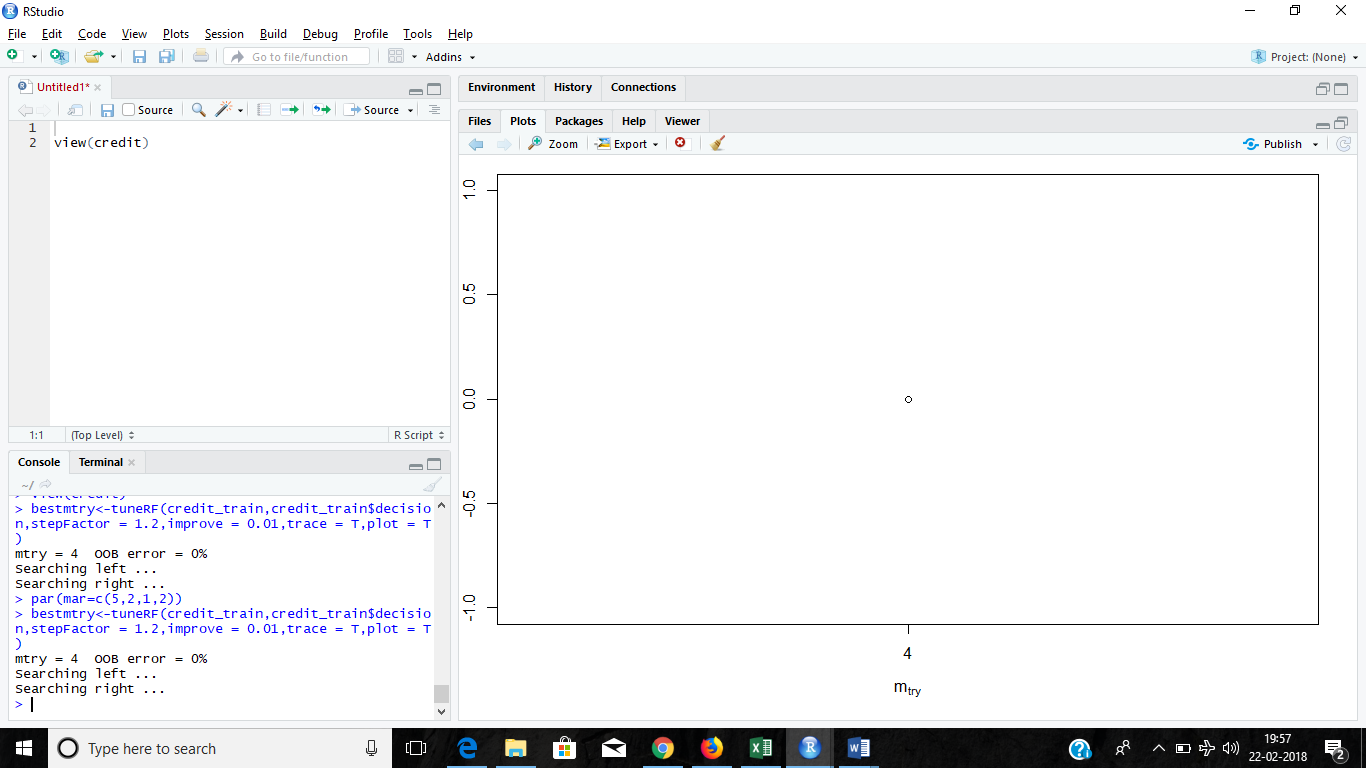


**Devide dataset:**

****

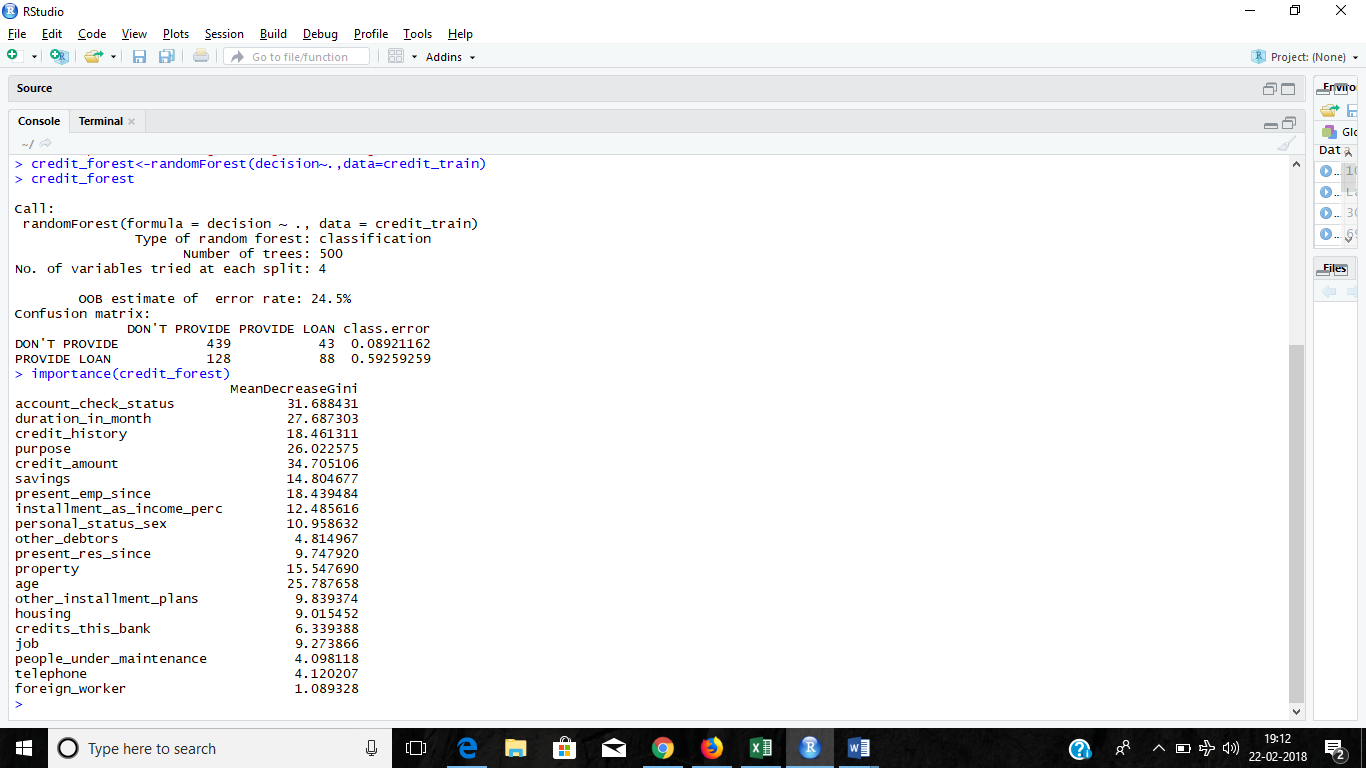
**Implement Model:**

****

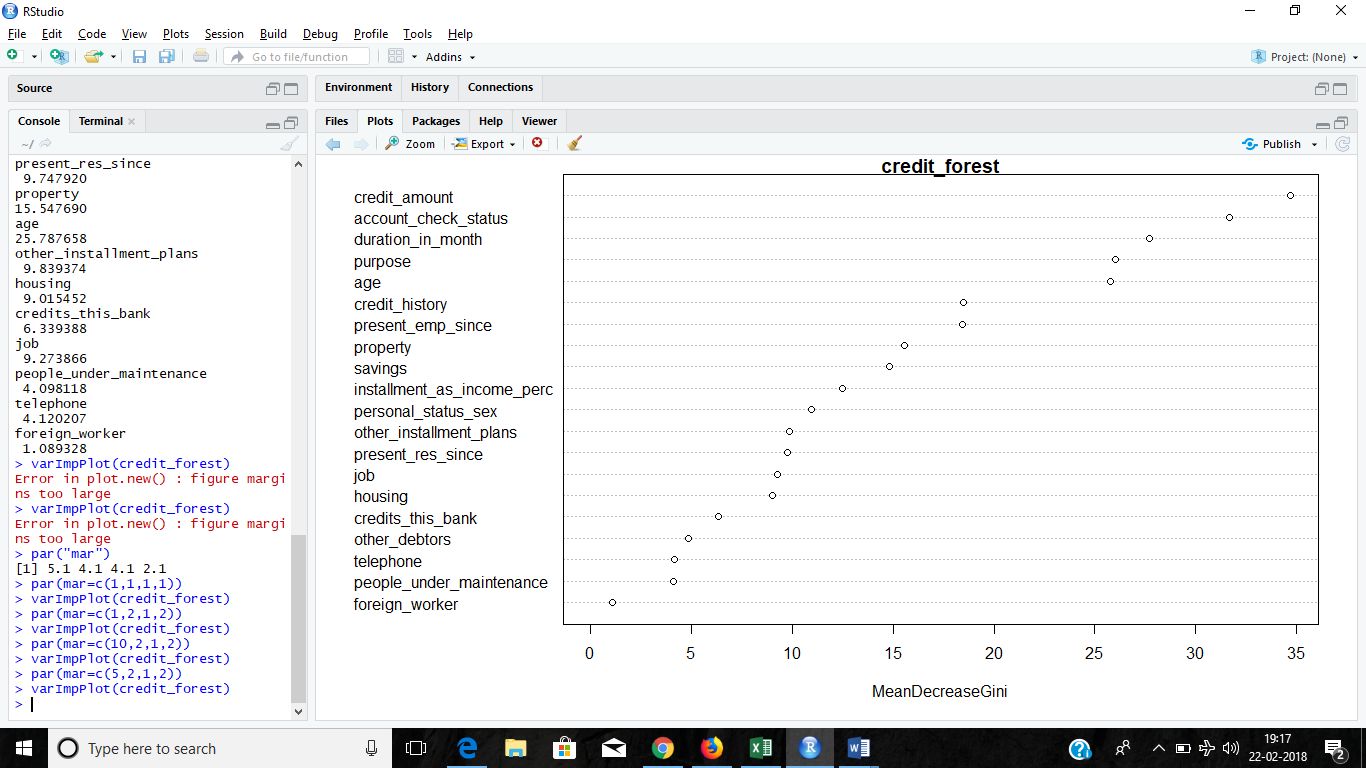


**Visualisation:**

**Gini-Index of variables:**

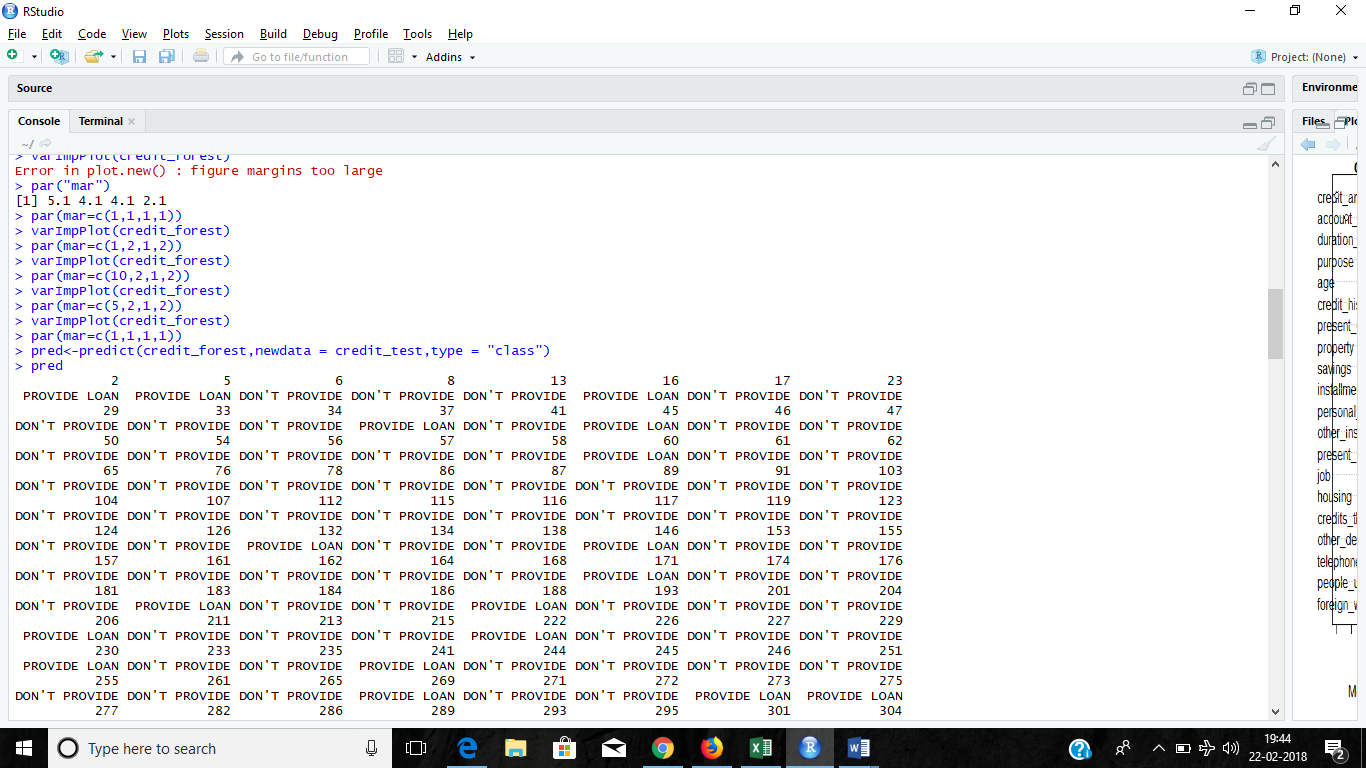
****

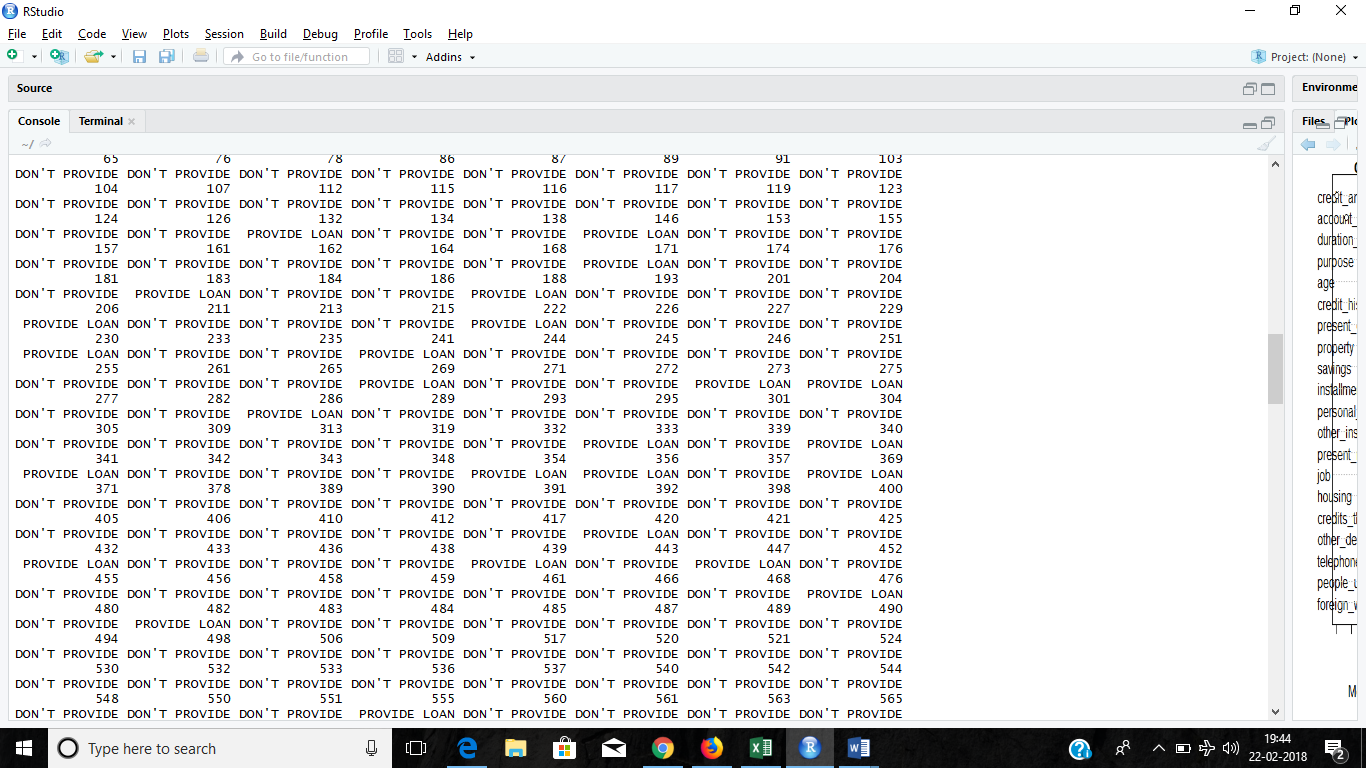
**Graphical representation of Gini Index:**

****

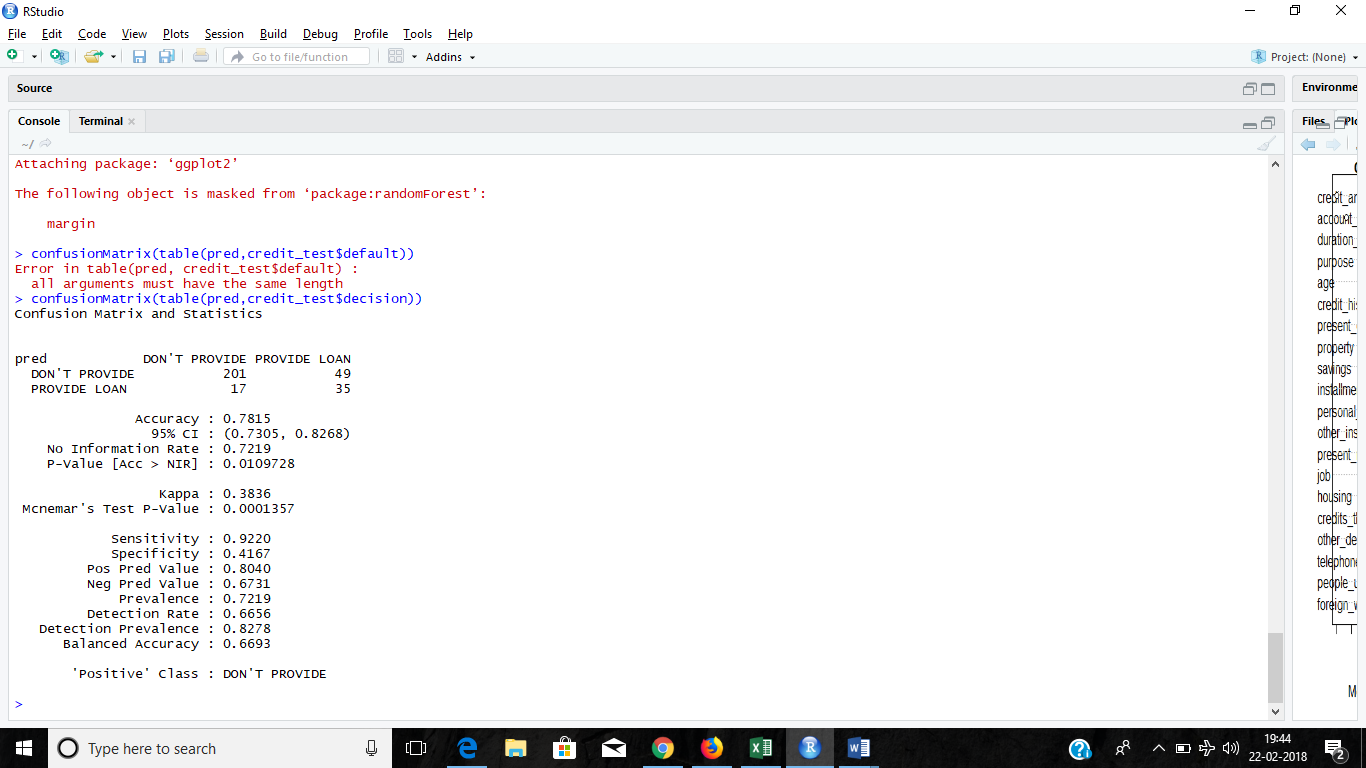
**Making Predictions:**

**.**

****

****

**Model Validation:**

****

**Code:**

> credit<-read.csv("C:\\Users\\Ashutosh\\Desktop\\credit.csv")

> View(credit)

> set.seed(2)

> id<-sample(2,nrow(credit),prob =c(0.7,0.3),replace=TRUE)

> credit\_train<-credit[id==1,]

> credit\_test<-credit[id==2,]

> library(randomForest)

> credit$decision<-as.factor(credit$decision)

> credit\_train$decision<-as.factor(credit\_train$decision)

> bestmtry<-tuneRF(credit\_train,credit\_train$decision,stepFactor = 1.2,improve = 0.01,trace = T,plot = T)

> credit\_forest<-randomForest(decision~.,data=credit\_train)

> credit\_forest

> varImpPlot(credit\_forest)

> pred<-predict(credit\_forest,newdata = credit\_test,type = "class")

> pred

> library(caret)

> confusionMatrix(table(pred,credit\_test$decision))

**Conclusion:**

In the dataset used there are many parameters available about a party which has applied for loan. These variables are:

account\_check\_status

duration\_in\_month

credit\_history

purpose

credit\_amount

savings

present\_emp\_since

installment\_as\_income\_perc

personal\_status\_sex

other\_debtors

present\_res\_since

property

age

other\_installment\_plans

housing

credits\_this\_bank

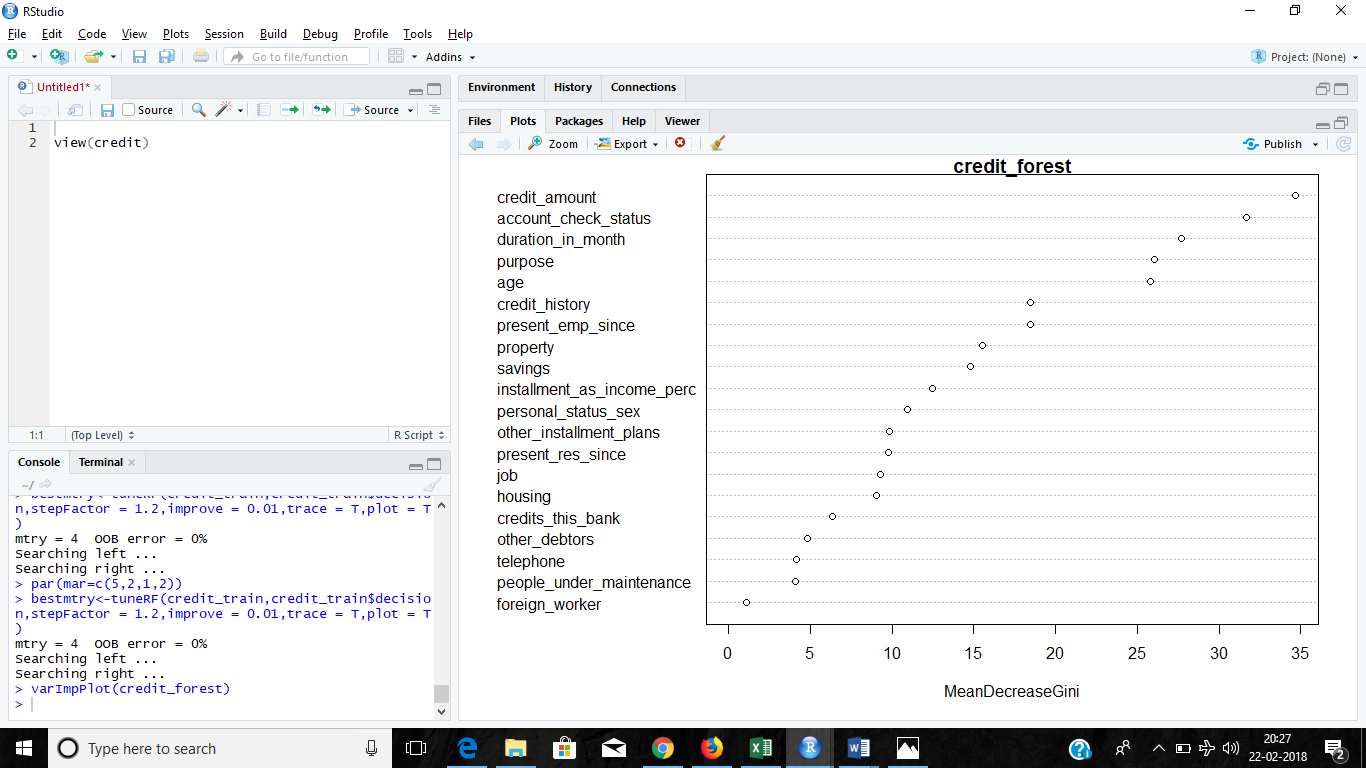
job

people\_under\_maintenance

telephone

foreign\_worker

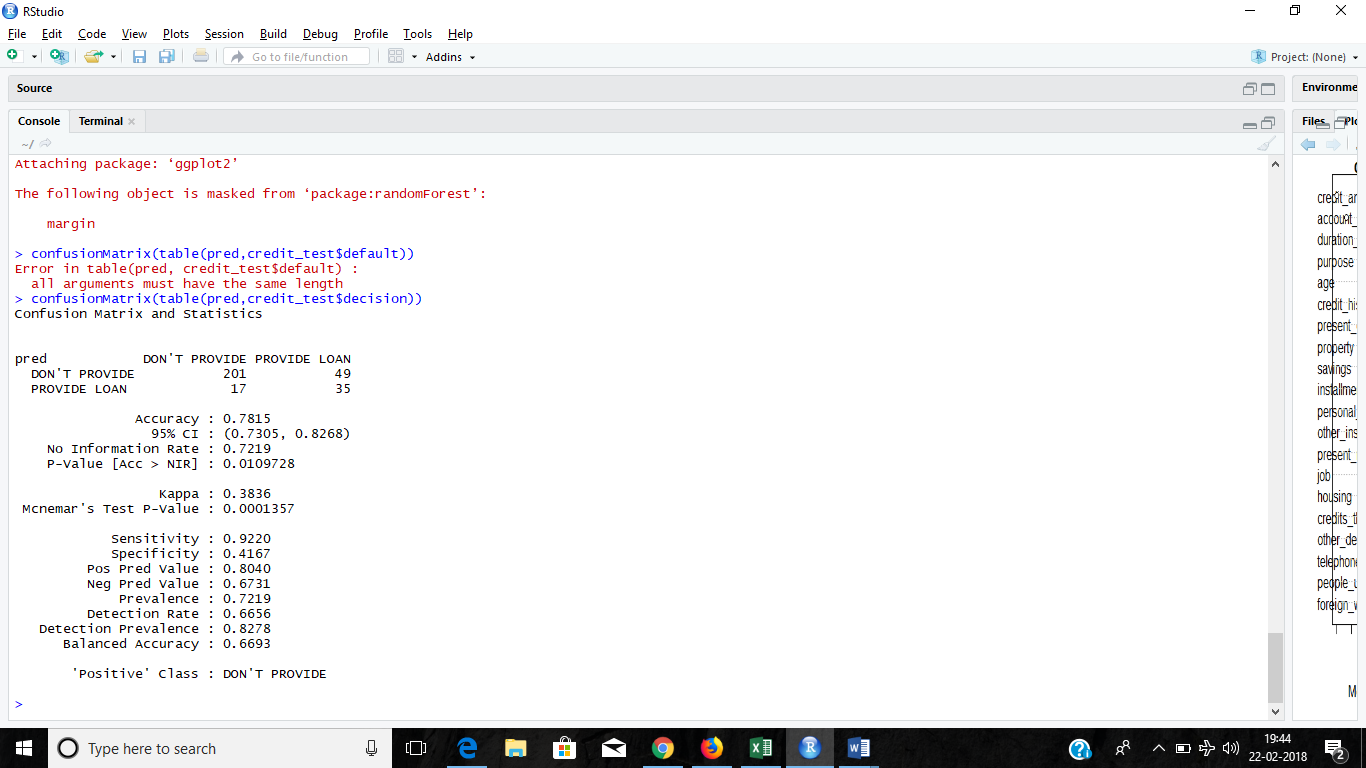
In these variables the most important variable which is most deciding for providing a loan are as below:



So in the algorithm used in random forest technique which is 80% accurate the prediction is done on the basis of these variables using many trees which will use subsets of variabes.

In our algorithm, tree weights are easily calculated based on their past performance, namely, OOB (out-of-bag) errors in training. To deal with the loan default prediction problem in question, we adopt a balanced random forests approach. We also try to speed up random forests’ learning process using parallelism to cope with the large data problem. We try to make loan default prediction on imbalanced data sets with an improved random forests approach which employs weighted majority votes in tree aggregation. The weights assigned to each tree in the forest are based on OOB (out-of-bag) errors which are easy to obtain during the forest construction process. Also, due to the fact that random forests can be parallel in nature, we employ the each package in the statistical software R to make random forest parallel and greatly reduce the learning time.

On these basis the prediction is done for the testing data. The accuracy of data predicted may be viewed by confusion matrix as:



Experiments also show that random forests with parallelism can greatly reduce the learning time random forests and this technique should be considered as a standard practice on learning large datasets.