LoanPrediction

Yatharth Malik

January 3, 2017

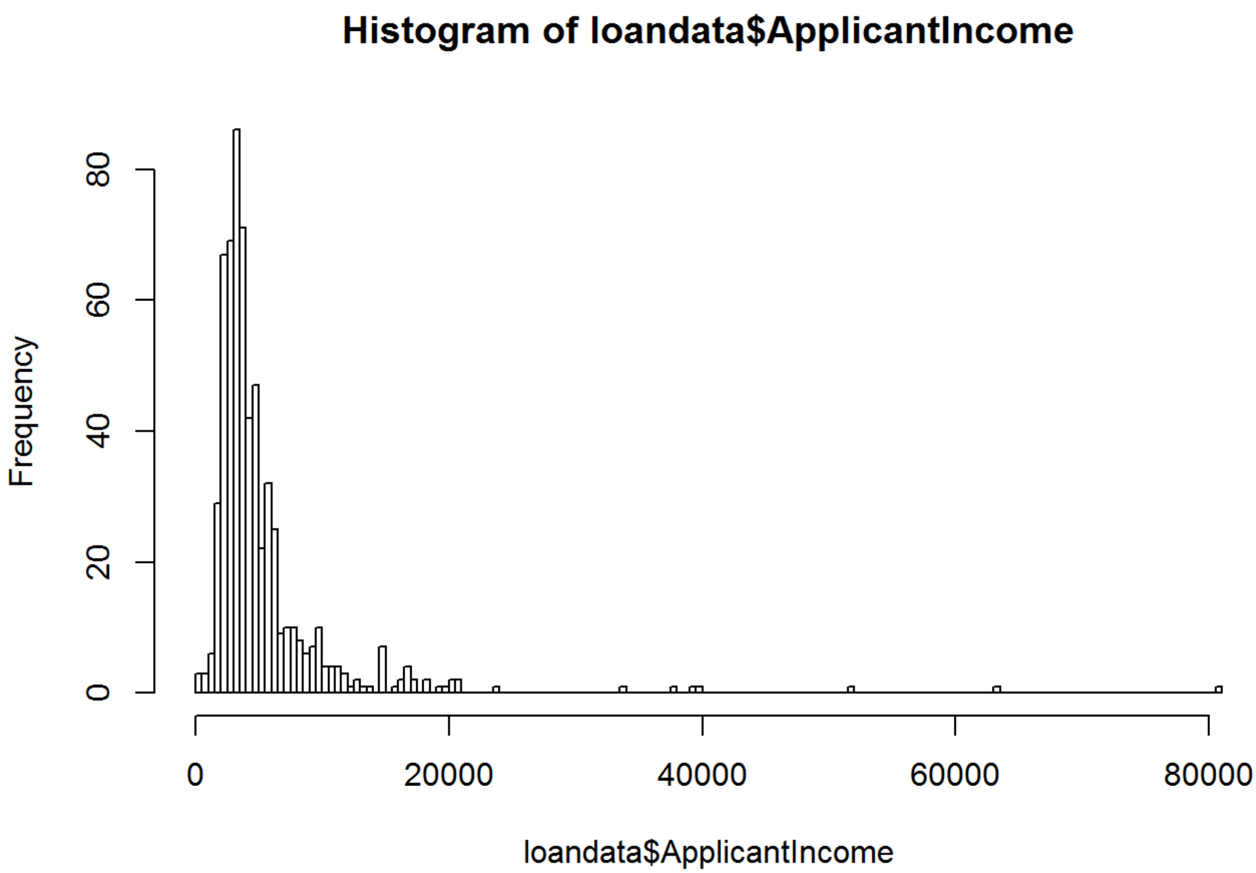
```
library(mlr)
library(caTools)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
set.seed(121)
```

Loading and exploring data

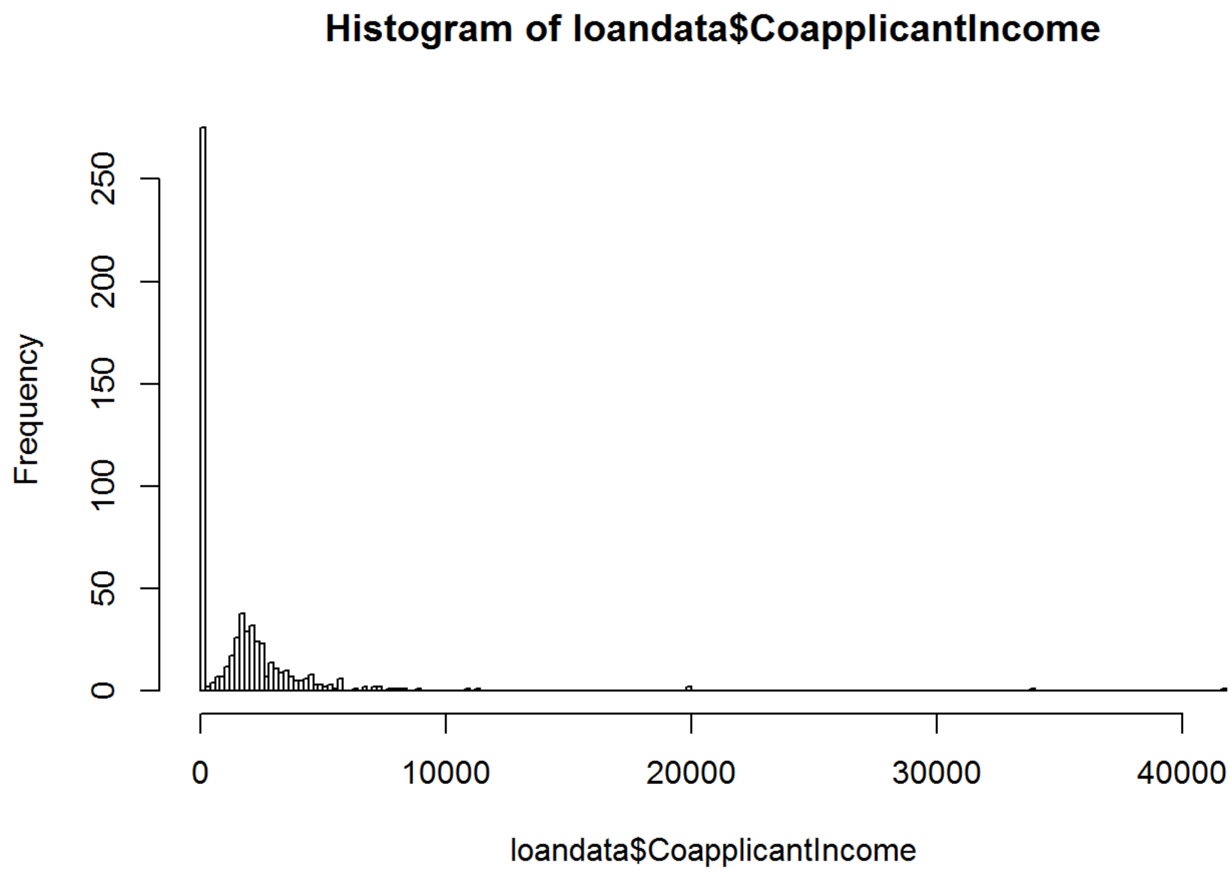
```
loandata = read.csv("train.csv",na.strings = c(""," ",NA))
summarizeColumns(loandata)
```

```
##                  name    type na         mean         disp median       mad
## 1             Loan_ID  factor  0           NA    0.9983713     NA        NA
## 2              Gender  factor 13           NA           NA     NA        NA
## 3             Married  factor  3           NA           NA     NA        NA
## 4          Dependents  factor 15           NA           NA     NA        NA
## 5           Education  factor  0           NA    0.2182410     NA        NA
## 6       Self_Employed  factor 32           NA           NA     NA        NA
## 7     ApplicantIncome integer  0 5403.4592834 6109.0416734 3812.5 1822.8567
## 8   CoapplicantIncome numeric  0 1621.2457980 2926.2483692 1188.5 1762.0701
## 9          LoanAmount integer 22  146.4121622   85.5873252  128.0   47.4432
## 10   Loan_Amount_Term integer 14  342.0000000   65.1204099  360.0    0.0000
## 11     Credit_History integer 50    0.8421986    0.3648783    1.0    0.0000
## 12      Property_Area  factor  0           NA    0.6205212     NA        NA
## 13        Loan_Status  factor  0           NA    0.3127036     NA        NA
##    min   max nlevs
## 1    1     1   614
## 2  112   489     2
## 3  213   398     2
## 4   51   345     4
## 5  134   480     2
## 6   82   500     2
## 7  150 81000     0
## 8    0 41667     0
## 9    9   700     0
## 10  12   480     0
## 11   0     1     0
## 12 179   233     3
## 13 192   422     2
```

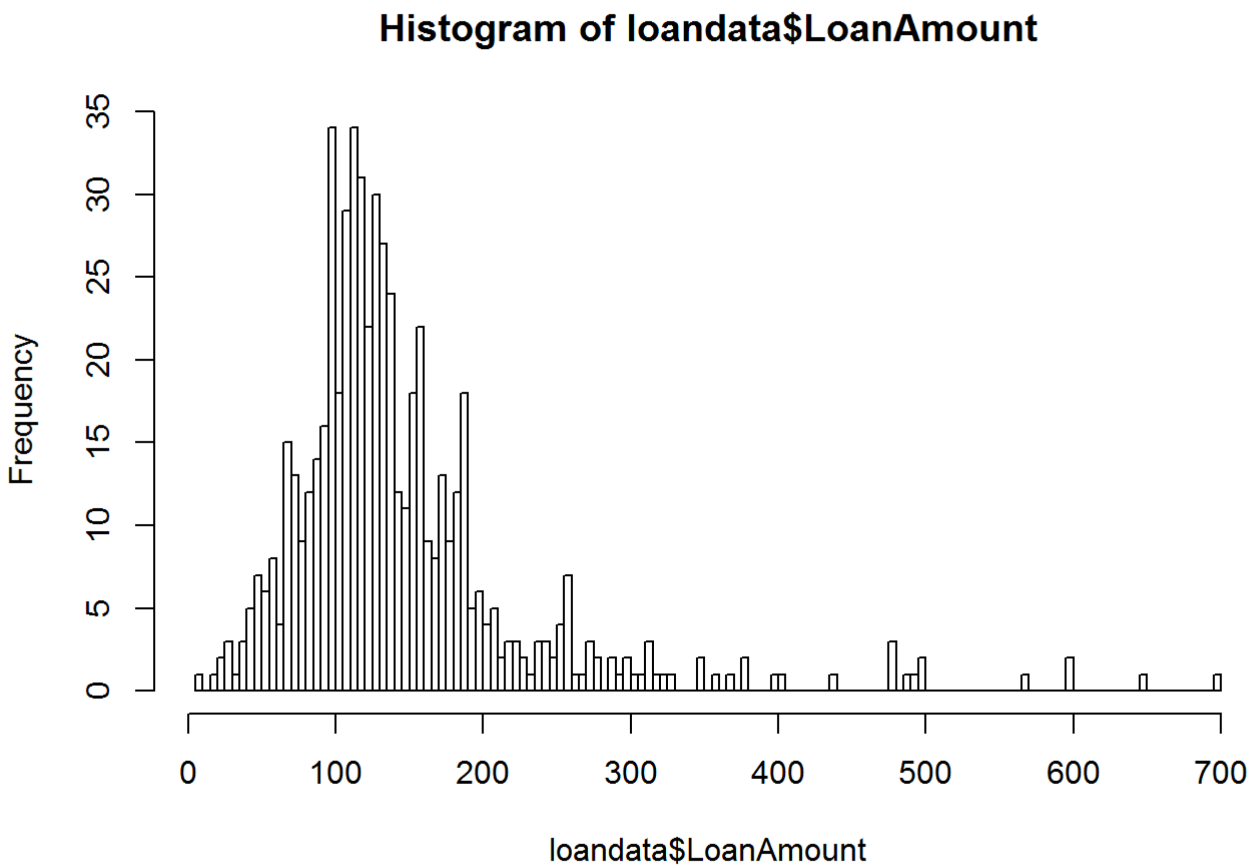
```
hist(loandata$ApplicantIncome,breaks = 200)
```




```
hist(loandata$CoapplicantIncome,breaks = 200)
```

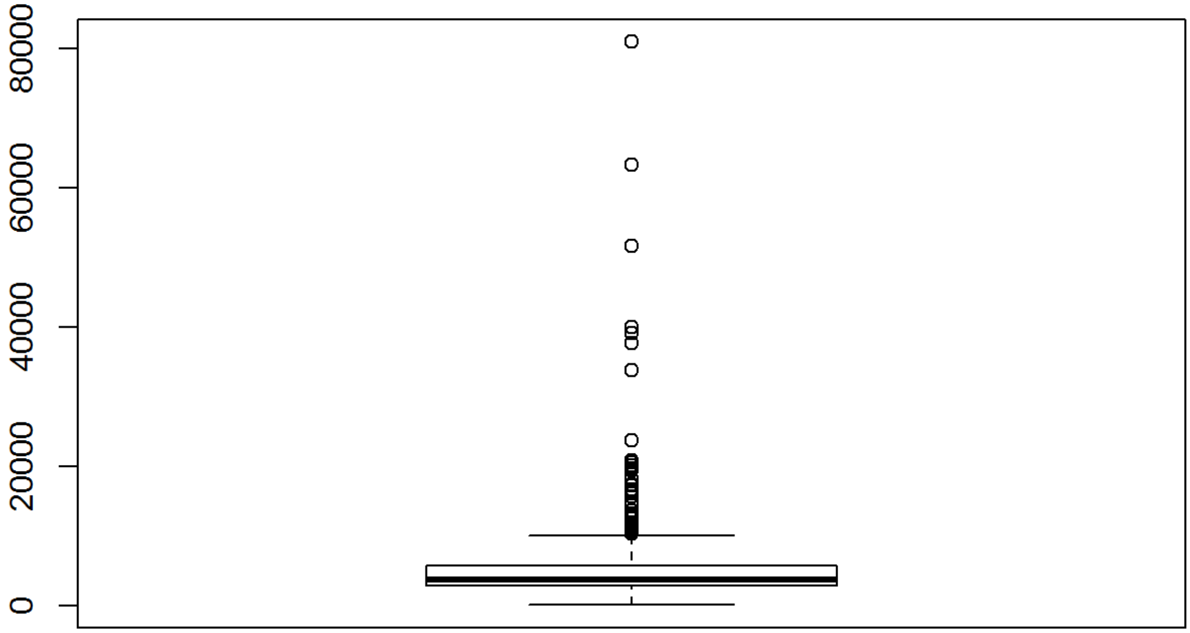



```
hist(loandata$LoanAmount,breaks = 200)
```

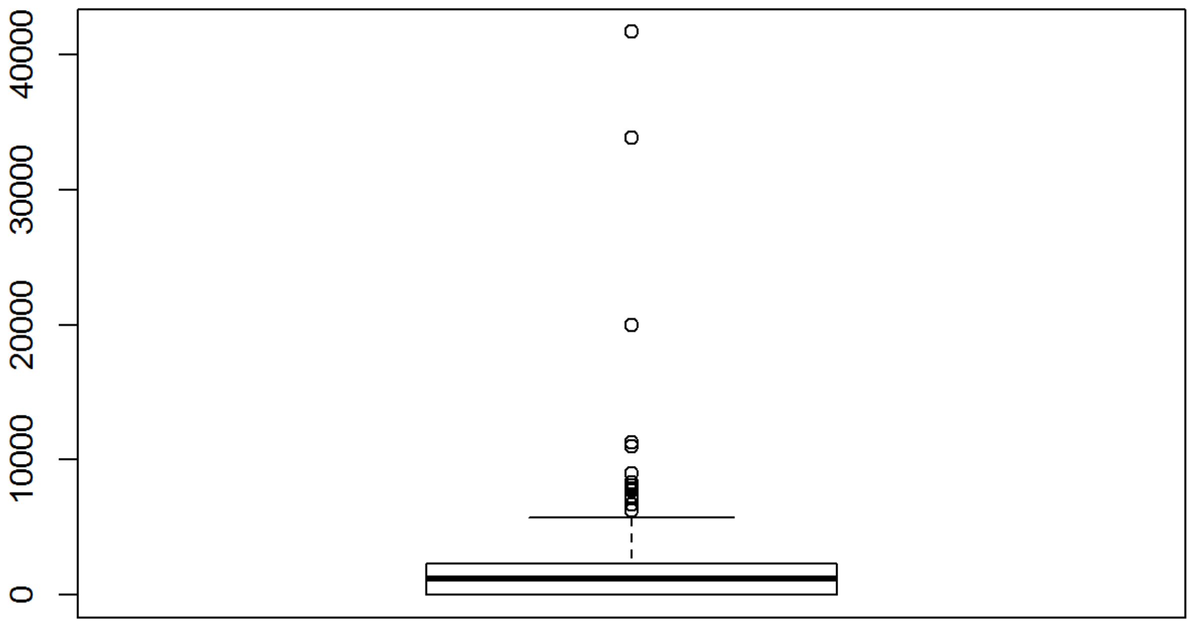


By looking at above plots, we find out that ApplicantIncome and CoapplicantIncome are highly skewed and hence we have to normalize them.

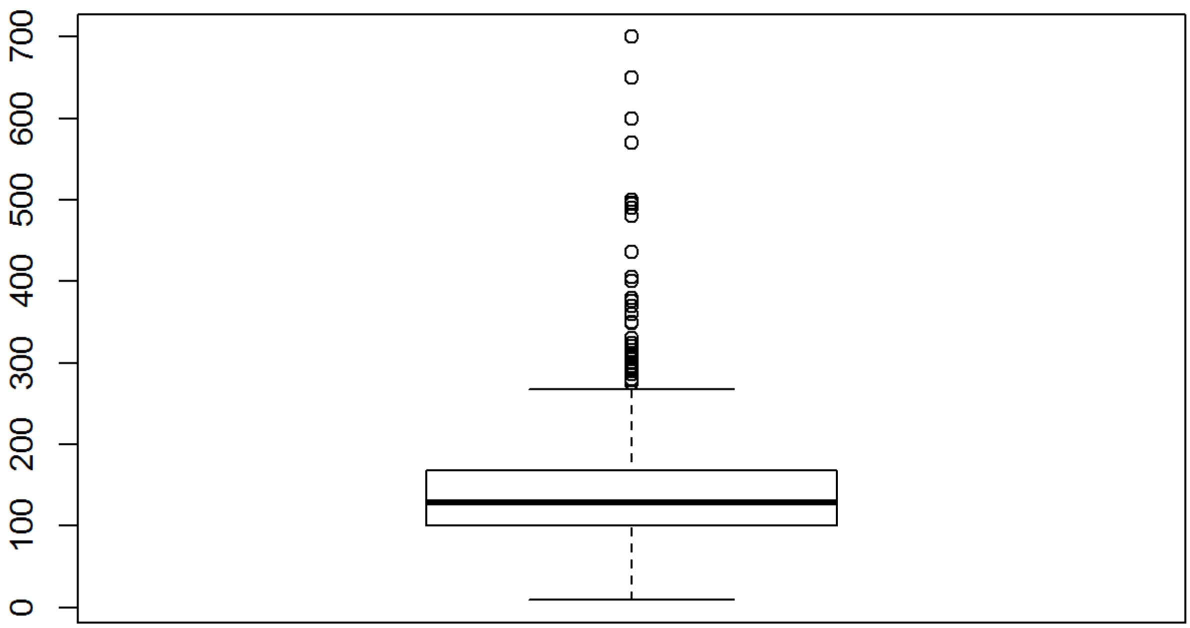
```
boxplot(loandata$ApplicantIncome)
```




```
boxplot(loandata$CoapplicantIncome)
```




```
boxplot(loandata$LoanAmount)
```



All these variables have outliers which need to be deal seperately.

Also we need to change Credit_History to factor and “3+” level to “3” in Dependents variable.

```
loandata$Credit_History = as.factor(loandata$Credit_History)
levels(loandata$Dependents)[4]  =  "3"
```

Missing Value Imputation

There are missing values in our dataset.So,we’ll have to deal with them first. We will replace the numeric missing values with the mean of that variable(Mean Imputation) and factor missing values with mode of that variable(Mode Imputation). For missing value imputation, we will use “impute” function from the package “mlr”

```
imp = impute(loandata,classes = list(factor = imputeMode(),integer = imputeMean()))
completedata = imp$data
```

Removing Outlier

To remove outliers we will use “capLargeValues” function from package “mlr”.

```
cd = capLargeValues(completedata,target = "Loan_Status",cols = c("ApplicantIncome"),threshold =
40000)
cd = capLargeValues(cd,target = "Loan_Status",cols = c("CoapplicantIncome"),threshold = 21000)
cd = capLargeValues(cd,target = "Loan_Status",cols =c("LoanAmount"),threshold = 520)
cappedData = cd
```

Creating new variables

```
cappedData$TotalIncome = cappedData$ApplicantIncome + cappedData$CoapplicantIncome
cappedData$IncomeLoan = cappedData$TotalIncome/cappedData$LoanAmount
```

Normalizing data

To normalize our dataset,we will use “preProcess” function from package “caret”

```
preproc = preProcess(cappedData)
dataNorm = predict(preproc,cappedData)
```

Correlation

Variables which are highly correlated do not contribute to accuracy of the model.Hence one of the two highly correlated variables can be ignored safely. To check the correlation of different numeric variables

```
az = split(names(dataNorm),sapply(dataNorm,function(x){class(x)}))
xs = dataNorm[az$numeric]
cor(xs)
```

```
##                   ApplicantIncome CoapplicantIncome  LoanAmount
## ApplicantIncome        1.00000000       -0.14117527  0.58388949
## CoapplicantIncome     -0.14117527        1.00000000  0.22253459
## LoanAmount             0.58388949        0.22253459  1.00000000
## Loan_Amount_Term      -0.03745779       -0.04086784  0.04108567
## TotalIncome            0.89527867        0.31465345  0.65998236
## IncomeLoan             0.41950702        0.18621087 -0.17732606
##                   Loan_Amount_Term TotalIncome IncomeLoan
## ApplicantIncome        -0.03745779  0.89527867  0.4195070
## CoapplicantIncome      -0.04086784  0.31465345  0.1862109
## LoanAmount              0.04108567  0.65998236 -0.1773261
## Loan_Amount_Term        1.00000000 -0.05430597 -0.1033059
## TotalIncome            -0.05430597  1.00000000  0.4860247
## IncomeLoan             -0.10330594  0.48602473  1.0000000
```

Since ApplicantIncome and TotalIncome are highly correlated,we will ignore TotalIncome variable.

```
dataNorm$TotalIncome = NULL
dataNorm$Loan_ID = NULL
```

Creating training and testing dataset

We will use package caTools to split the data into training and testing dataset.70% of original dataset will be training dataset and 30% will be testing.

```
set.seed(121)
split = sample.split(dataNorm$Loan_Status,SplitRatio = 0.70)
train = subset(dataNorm,split==T)
test =  subset(dataNorm,split==F)
```

Predictive Models

1. Logistic Regression

```
logmodel = glm(Loan_Status ~ . ,data = train,family = "binomial")
logpreds = predict(logmodel,newdata = test, type = "response")
```

To create confusion matrix

```
table(test$Loan_Status,logpreds>0.55)
```

```
##
##     FALSE TRUE
##   N    19   39
##   Y     2  125
```

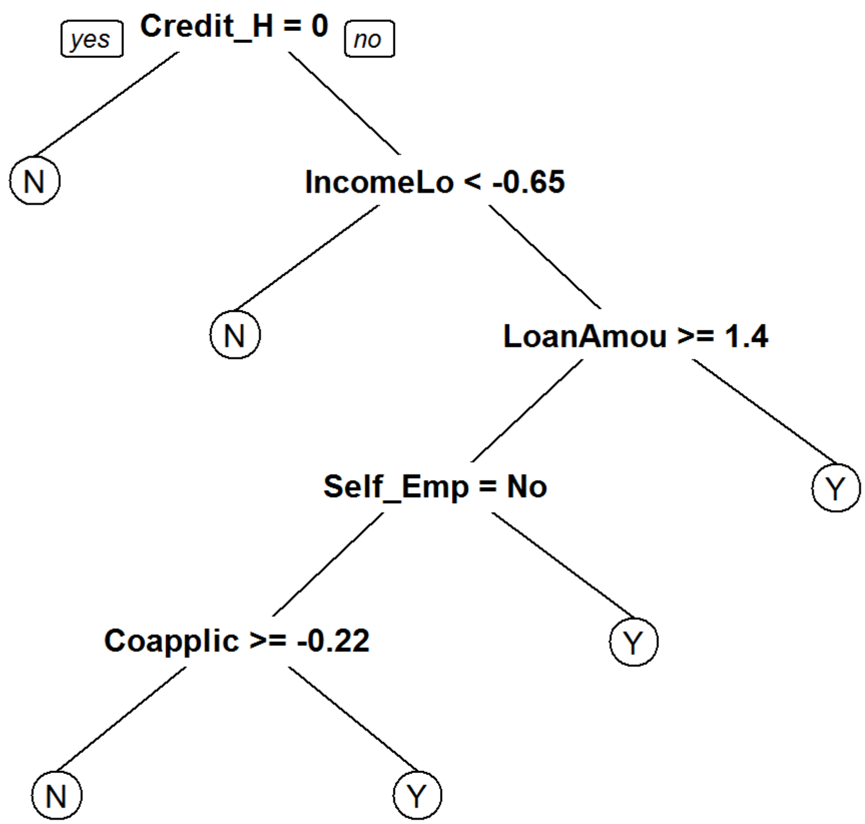
We get an accuracy of

```
((19+125)/nrow(test))*100
```

```
## [1] 77.83784
```

2.Decision Trees

```
set.seed(121)
tree = rpart(Loan_Status ~ .,data = train,method = "class")
prp(tree)
```




```
treepreds = predict(tree,newdata = test , type = "class")
```

Confusion Matrix

```
table(test$Loan_Status,treepreds)
```

```
##    treepreds
##        N   Y
##   N  27  31
##   Y  12 115
```

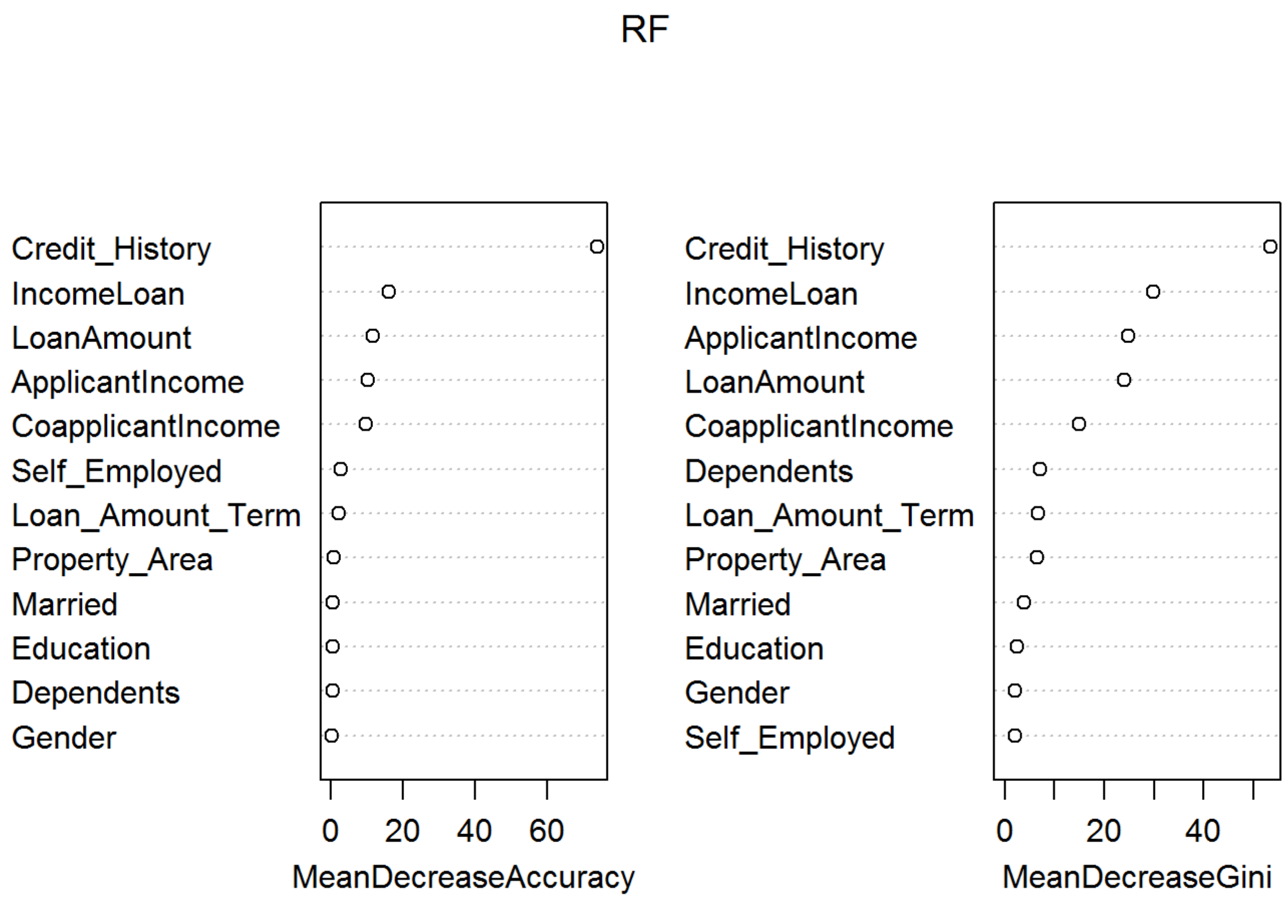
Accuracy of our model is given by :-

```
((27+115)/nrow(test))*100
```

```
## [1] 76.75676
```

3. Random Forest

```
set.seed(121)
RF = randomForest(Loan_Status ~ . ,data =train,importance= T)
varImpPlot(RF)
```




```
RFpreds = predict(RF,newdata = test,type = "class")
```

Confusion Matrix

```
table(test$Loan_Status,RFpreds)
```

```
##    RFpreds
##       N   Y
##   N  26  32
##   Y  13 114
```

Accuracy of our model is given by :-

```
((26+114)/nrow(test))*100
```

```
## [1] 75.67568
```

Out of all 3 models,Logistic Model gives us best accuracy.