Project Report on

LOAN ELIGIBLITY PREDICTION

Submitted By
Kalyani Avhale - 242062001

Guided By

Prof. Ashwini Matange

Post-Graduation Diploma in

Data Science & Artificial Intelligence

CONTENT

PROBLEM STATEMENT

Dream Housing Finance company deals in all home loans. They have a presence across all urban, semi-urban, and rural areas. Customer-first applies for a home loan after that company validates the customer eligibility for a loan.

The company wants to automate the loan eligibility process (real-time) based on customer detail provided while filling the online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History, and others. To automate this process, they have given a problem to identify the customer's segments, those are eligible for loan amount so that they can specifically target these customers. Here they have provided a partial data set.

DATA DICTIONARY

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi Urban/ Rural
Loan_Status	(Target) Loan approved (Y/N)

#Loan_Status : is target feature

R LIBRARIES USED

```
# @Script for Loan Eligibility Prediction Script
# @Author : Kalyani Avhale
# @Language : R
# @Dataset source : https://www.kaggle.com/vikasukani/loan-eligible-dataset
# @Date : 17th April , 2021

setwd("D:/Trisem_2/R/Project/packages")
#Install Required Packages
# install.packages('tidyverse') # metapackage of all tidyverse packages
# install.packages('dplyr')    #data manuplation(included in tidyverse)
# install.packages('caret')    #for Classification and regression
# install.packages('ggpubr')   #arranging plots into grids
# install.packages('modeest')  #Estimation of the mode
# install.packages('ggplot2')  #for plotting graphs(included in tidyverse)
# install.packages('ggcorrplot')#for plotting correlation matrix
# install.packages('randomForest')#Random forest
# install.packages('xgboost')     #Gradient Boosting

#import packages
```

- library('tidyverse')
- library('caret')
- library('ggpubr')
- library('modeest')
- library('ggcorrplot')
- library('randomForest')
- library('xgboost')

UNDERSTANDING THE DATA

- data=read.csv('loan-train.csv',na.strings=c("")) #Load the dataset
- head(data)

```
   Loan_ID Gender Married Dependents     Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History
1 LP001002   Male      No          0      Graduate            No            5849                 0         NA              360              1
2 LP001003   Male     Yes          1      Graduate            No            4583              1508        128              360              1
3 LP001005   Male     Yes          0      Graduate           Yes            3000                 0         66              360              1
4 LP001006   Male     Yes          0  Not Graduate            No            2583              2358        120              360              1
5 LP001008   Male      No          0      Graduate            No            6000                 0        141              360              1
6 LP001011   Male     Yes          2      Graduate           Yes            5417              4196        267              360              1
  Property_Area Loan_Status
1         Urban           Y
2         Rural           N
3         Urban           Y
4         Urban           Y
5         Urban           Y
6         Urban           Y
>
```

- dim(data) #Dimension of dataset

  # [1] 614 13

- str(data) #returns type of attribute along with firstfew values

```
'data.frame':   614 obs. of  13 variables:
 $ Loan_ID          : chr  "LP001002" "LP001003" "LP001005" "LP001006" ...
 $ Gender           : chr  "Male" "Male" "Male" "Male" ...
 $ Married          : chr  "No" "Yes" "Yes" "Yes" ...
 $ Dependents       : chr  "0" "1" "0" "0" ...
 $ Education        : chr  "Graduate" "Graduate" "Graduate" "Not Graduate" ...
 $ Self_Employed    : chr  "No" "No" "Yes" "No" ...
 $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
 $ CoapplicantIncome: num  0 1508 0 2358 0 ...
 $ LoanAmount       : int  NA 128 66 120 141 267 95 158 168 349 ...
 $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
 $ Credit_History   : int  1 1 1 1 1 1 1 0 1 1 ...
 $ Property_Area    : chr  "Urban" "Rural" "Urban" "Urban" ...
 $ Loan_Status      : chr  "Y" "N" "Y" "Y" ...
>
```

- attr_type <- sapply(data,class) # list types for each attribute

```
                  attr_type
Loan_ID           character
Gender            character
Married           character
Dependents        character
Education         character
Self_Employed     character
ApplicantIncome     integer
CoapplicantIncome   numeric
LoanAmount          integer
Loan_Amount_Term    integer
Credit_History      integer
Property_Area     character
Loan_Status       character
```

#Levels of classes

- unique(data$Loan_Status) # Since we have 2 classes it's a binary classification problem

  [1] "Y" "N"

#class distribution

- percent = prop.table(table(data$Loan_Status))*100
- cbind(freq=table(data$Loan_Status),percentage=percent)

```
  freq percentage
N  192   31.27036
Y  422   68.72964
>
```

#We see that only 31% of all the people in the dataset had a loan being approved. This means that our baseline model has an accuracy of 69%. An important measure when evaluating our model we be the sensitivity (aka recall aka the probability of detection as positive). If this value is low then our model is not very good at detecting true positive cases, even if the accuracy is very high. There are several ways to deal with imbalance

#Statistical Summary

- summary(data)

```
> summary(data)
   Loan_ID             Gender             Married            Dependents          Education          Self_Employed      ApplicantIncome
 Length:614         Length:614         Length:614         Length:614         Length:614         Length:614         Min.   :  150
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   1st Qu.: 2878
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median : 3812
                                                                                                                   Mean   : 5403
                                                                                                                   3rd Qu.: 5795
                                                                                                                   Max.   :81000

 CoapplicantIncome   LoanAmount     Loan_Amount_Term Credit_History   Property_Area       Loan_Status
 Min.   :    0     Min.   :  9.0   Min.   : 12      Min.   :0.0000   Length:614         Length:614
 1st Qu.:    0     1st Qu.:100.0   1st Qu.:360      1st Qu.:1.0000   Class :character   Class :character
 Median : 1188     Median :128.0   Median :360      Median :1.0000   Mode  :character   Mode  :character
 Mean   : 1621     Mean   :146.4   Mean   :342      Mean   :0.8422
 3rd Qu.: 2297     3rd Qu.:168.0   3rd Qu.:360      3rd Qu.:1.0000
 Max.   :41667     Max.   :700.0   Max.   :480      Max.   :1.0000
                   NA's   :22      NA's   :14       NA's   :50
>
```

we can see few cols has NA's and the scale for variables differ

We can perform scaling in later steps

We can see few outlier values

PRE-PROCESSING/CLEANING

#check for duplicate rows

- dim(loan_dataset[duplicated(loan_dataset$Loan_ID),])
  [1] 0 13

#we have 0 duplicates across 13 columns for Loan_Id

#check for all unique values across dataset (probably for char type values)

- Col_names=c("Gender","Married","Dependents","Education",

  "Self_Employed","Loan_Amount_Term",

  "Credit_History","Property_Area","Loan_Status")
- lapply(loan_dataset[Col_names], function(x) unique(x))

```
$Gender
[1] "Male"   "Female" NA

$Married
[1] "No"  "Yes" NA

$Dependents
[1] "0"   "1"   "2"   "3+" NA

$Education
[1] "Graduate"     "Not Graduate"

$Self_Employed
[1] "No"  "Yes" NA

$Loan_Amount_Term
 [1] 360 120 240  NA 180  60 300 480  36  84  12

$Credit_History
[1]  1  0 NA

$Property_Area
[1] "Urban"     "Rural"     "Semiurban"

$Loan_Status
[1] "Y" "N"
```

Dependents,Gender,Self_Employed has NA's and 3 suffixed with +

Loan_Amount_Term,Credit_History has NA's

#check for null values count for numerical type cols

- colSums(is.na(loan_dataset))

```
          Loan_ID            Gender           Married        Dependents         Education     Self_Employed   ApplicantIncome CoapplicantIncome
                0                13                 3                15                 0                32                 0                 0
       LoanAmount  Loan_Amount_Term    Credit_History     Property_Area       Loan_Status
               22                14                50                 0                 0
```

#replace 3+ with 3

- loan_dataset$Dependents <-
  replace(loan_dataset$Dependents,loan_dataset$Dependents=='3+',3)

HANDLING MISSING VALUES

- numeric_cols <- c('ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term') #numerical cols
- cat_cols <- c('Gender','Married','Dependents','Self_Employed','Credit_History') #categorical cols

#numeric NA : mean

- data_beforeImputation[numeric_cols] <- sapply(data_beforeImputation[numeric_cols], function(x)ifelse(is.na(x), mean(x, na.rm=TRUE), x))
- colSums(is.na(data_beforeImputation))

#categorical NA : fill with mode

- data_beforeImputation[cat_cols] <- sapply(data_beforeImputation[cat_cols], function(x)ifelse(is.na(x), mfv(x), x))
- colSums(is.na(data_beforeImputation))

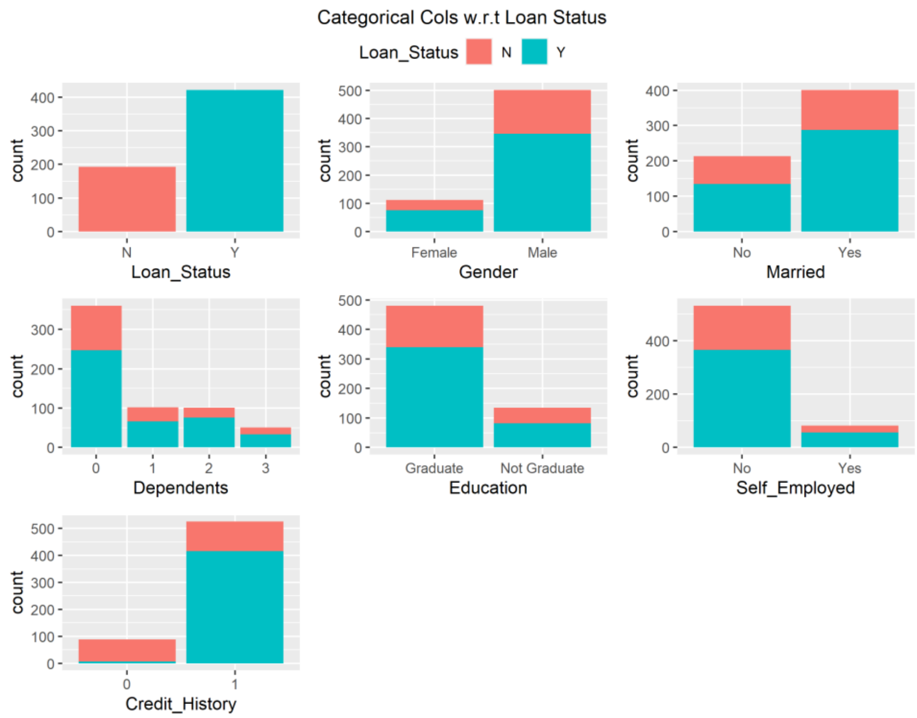
```
          Loan_ID            Gender           Married        Dependents         Education     Self_Employed   ApplicantIncome CoapplicantIncome
                0                 0                 0                 0                 0                 0                 0                 0
       LoanAmount  Loan_Amount_Term    Credit_History     Property_Area       Loan_Status
                0                 0                 0                 0                 0
```

EXPLORATORY DATA ANALYSIS

- setwd("D:/Trisem_2/R/Project/Plots") #derictory to save plots

#visualizaing categorical variables first with respect to Loan Status

- ls_plt <- ggplot(data = data_afterImputation) + geom_bar(mapping = aes(x = Loan_Status,fill=Loan_Status))
- g_plt <- ggplot(data = data_afterImputation) + geom_bar(mapping = aes(x = Gender,fill=Loan_Status))
- m_plt <- ggplot(data = data_afterImputation) + geom_bar(mapping = aes(x = Married,fill=Loan_Status))
- d_plt <- ggplot(data = data_afterImputation) + geom_bar(mapping = aes(x = Dependents,fill=Loan_Status))
- e_plt <- ggplot(data = data_afterImputation) + geom_bar(mapping = aes(x = Education,fill=Loan_Status))
- se_plt <- ggplot(data = data_afterImputation) + geom_bar(mapping = aes(x = Self_Employed,fill=Loan_Status))
- ch_plt <- ggplot(data = data_afterImputation,aes(x=Credit_History,fill=Loan_Status)) +geom_bar()
- ggarrange(ls_plt,g_plt,m_plt,d_plt,e_plt,se_plt,ch_plt,nrow=3,ncol=3,common.legend =TRUE)

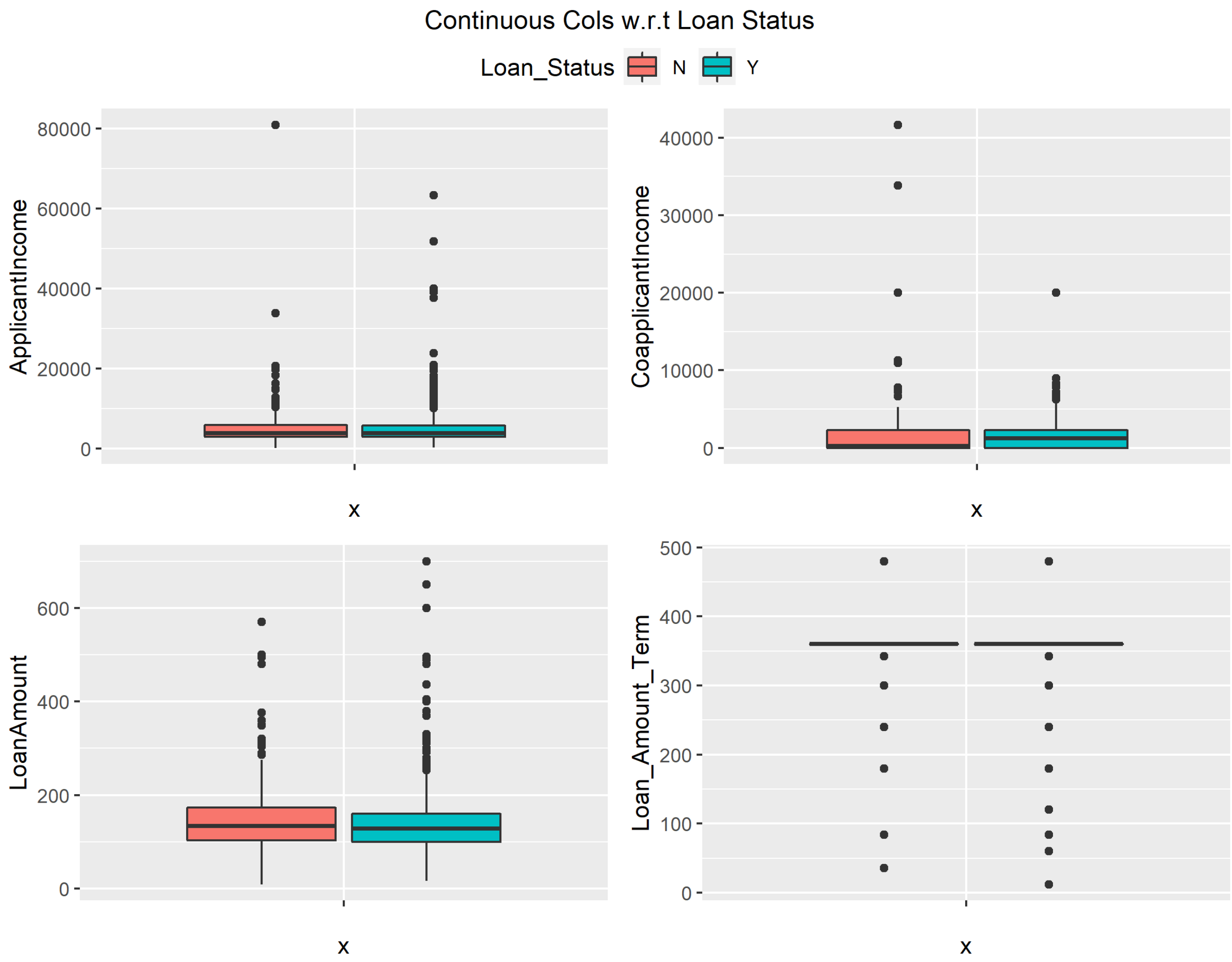


Inference :Male applicant has high loan approval,Applicant with 0 dependents has been approved with loan as compared to applicants with dependents. Self-employed applicants with loan approval e low as compared with which are not self employed(can be with other profession such as jobs,business,etc). Applicant with Credit History has highest loan approval

#Analyzing the three continuous variables w.r.t Loan_Status:

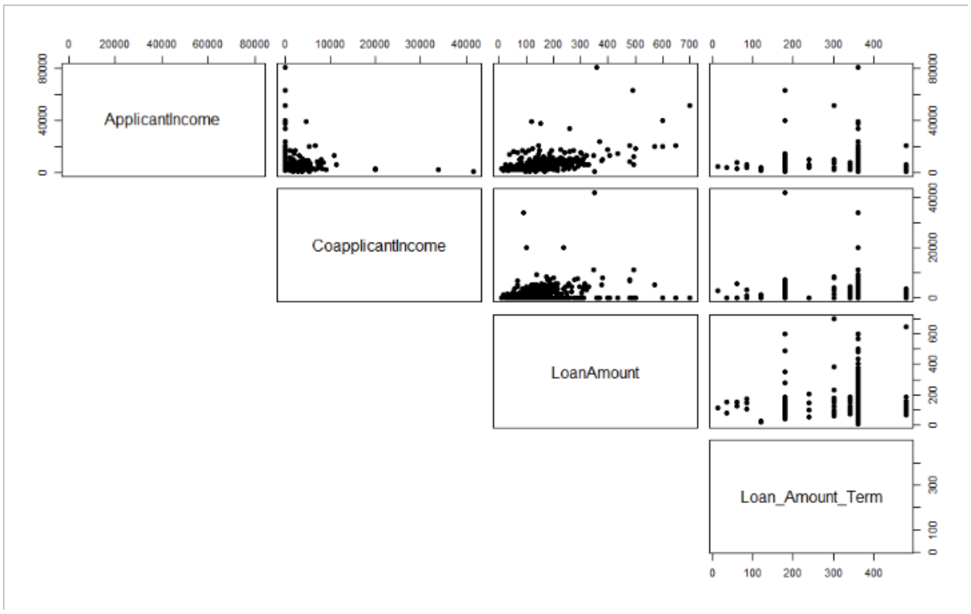
#ApplicantIncome,CoapplicantIncome,LoanAmount,Loan_Amount_Term

- ai_plt <- ggplot(data_afterImputation, aes(y= ApplicantIncome, x = "", fill = Loan_Status)) + geom_boxplot()
- cai_plt <- ggplot(data_afterImputation, aes(y= CoapplicantIncome, x = "", fill = Loan_Status)) + geom_boxplot()
- la_plt <- ggplot(data_afterImputation, aes(y= LoanAmount, x = "", fill = Loan_Status)) + geom_boxplot()
- lat_plt <- ggplot(data_afterImputation, aes(y= Loan_Amount_Term, x = "", fill = Loan_Status)) + geom_boxplot()
- figure <- ggarrange(ai_plt,cai_plt,la_plt,lat_plt,nrow=2,ncol=2,common.legend=TRUE)
- annotate_figure(figure,top = "Continuous Cols w.r.t Loan Status")
- ggsave('Numerical_col_plot.png')



#Pair Plot

- numeric_cols <- c('ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term')
- options(repr.plot.width =10, repr.plot.height = 10)   #adjust size of plot
- pairs(data_afterImputation[numeric_cols], pch = 19,cex.labels=1.5,lower.panel=NULL)



#We have positive correlations: LoanAmount and ApplicantIncome and LoanAmount and CoapplicantIncome

- options(repr.plot.width =10, repr.plot.height = 10) #adjust size of plot
- par(mfrow=c(2,2)) #arrange plot (matrix grid)
- boxplot(data_afterImputation$LoanAmount,horizontal=TRUE,main="Loan Amount",col='red')
- plot(density(data_afterImputation$LoanAmount),main="Density Graph",col='red')
- boxplot(log(data_afterImputation$LoanAmount),horizontal=TRUE,main="Loan Amount after log",col='blue')
- plot(density(log(data_afterImputation$LoanAmount)),main="Density Graph after log",col='blue')

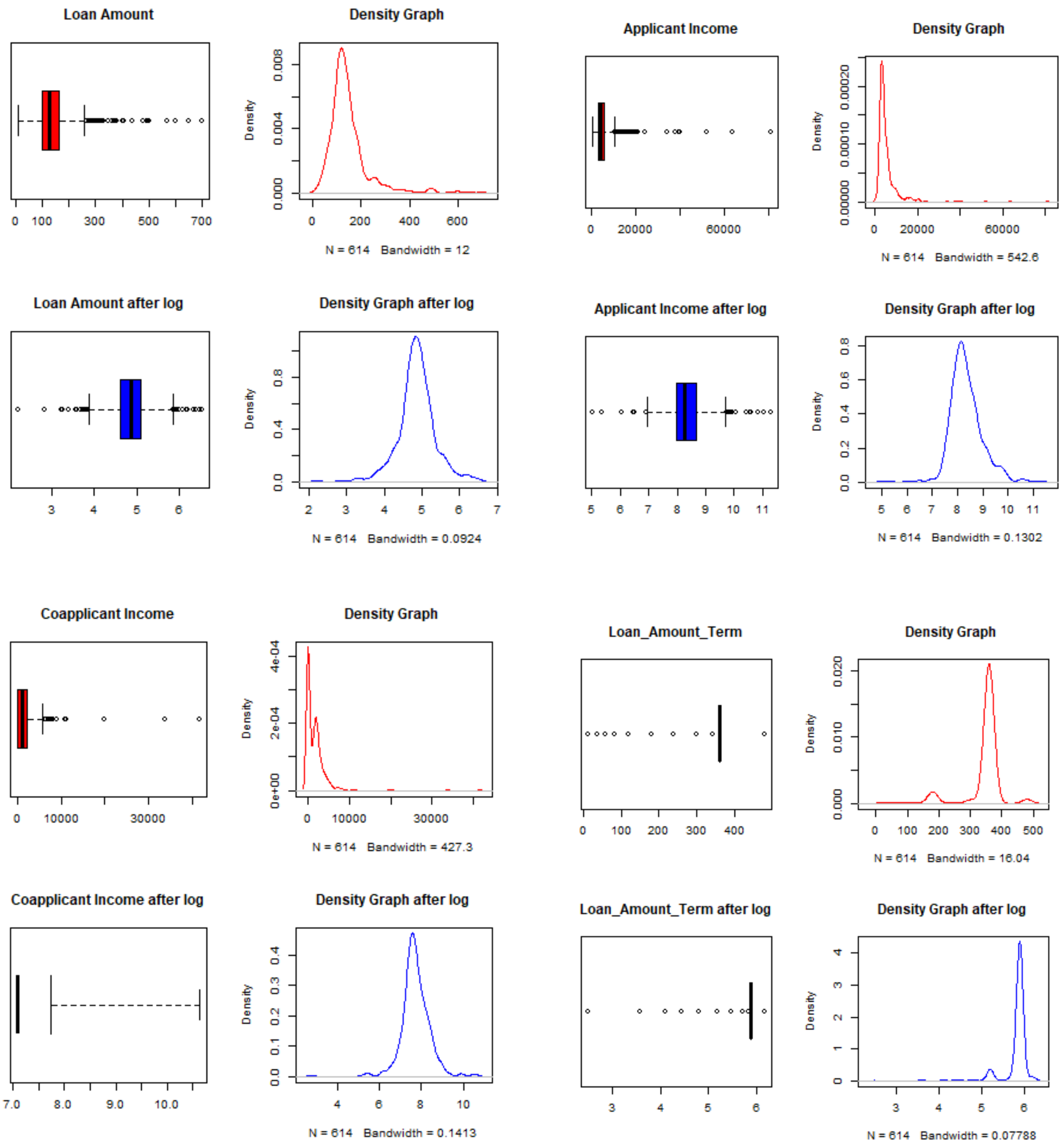
#LoanAmount log(removed the skewness)

#Similar plots for :

#ApplicantIncome

#CoapplicantIncome

#Loan_Amount_Term



#add log values to data

- data_afterImputation$ApplicantIncome_log = log(data_afterImputation$ApplicantIncome)
- data_afterImputation$LoanAmount_log = log(data_afterImputation$LoanAmount)
- data_afterImputation$CoapplicantIncome_log = log(data_afterImputation$CoapplicantIncome)

#log(0) is -Inf , so replace -Inf to 0

- data_afterImputation[data_afterImputation== -Inf]<-0

#lets check the covariance metrics

- round(cor(data_afterImputation[numeric_cols]),3)

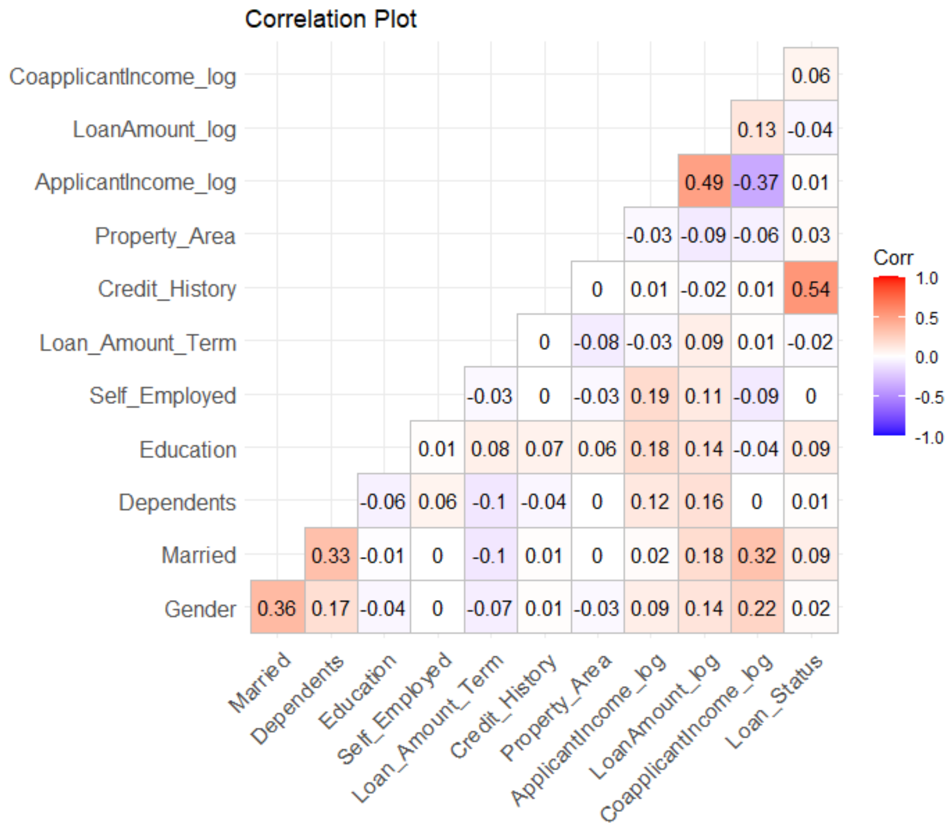
we have +ve correlations : LoanAmount and ApplicantIncome and LoanAmount and coapplicationIncome

As Loan Amount depends on Income of applicant ,the more the Income has high probablity of getting more Loan amount

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
ApplicantIncome	1.000	-0.117	0.566	-0.045
CoapplicantIncome	-0.117	1.000	0.188	-0.060
LoanAmount	0.566	0.188	1.000	0.039
Loan_Amount_Term	-0.045	-0.060	0.039	1.000

#co-relation Plot after scaling data

- options(repr.plot.width =20, repr.plot.height = 20)#adjust size of plot
- loan_corr <- round(cor(af_scale),3) #get corr matrix
- ggcorrplot(loan_corr, title = "Correlation Plot",type = "lower",lab=TRUE,insig = "blank")



Credit History has Positive correlation with target feature Loan Status

Dependents and Married , Loan Amount and Applicant Income are positivly correlated

Applicant Income and CoApplicant Income has negative correlation

CATEGORICAL FEATURES

#drop log cols

- loanData <- subset(data_afterImputation,select = -c(ApplicantIncome,LoanAmount,CoapplicantIncome))

#Education

- loanData$Education <- sapply(loanData$Education,function(x) ifelse(x=='Graduate',1,0)) #replace "Graduate" with 1 and "Not Graduate" with 0

#Loan_status

- loanData$Loan_Status <- sapply(loanData$Loan_Status,function(x) ifelse(x=='Y',1,0))

#Gender

- loanData$Gender <- sapply(loanData$Gender,function(x) ifelse(x=='Male',1,0)) #replace "Y" with 1 and "N" with 0

#Married

- loanData$Married <- sapply(loanData$Married,function(x) ifelse(x=='Yes',1,0)) #replace "Y" with 1 and "N" with 0

#Self_Employed

- loanData$Self_Employed <- sapply(loanData$Self_Employed,function(x) ifelse(x=='Yes',1,0)) #replace "Yes" with 1 and "No" with 0

#Property_Area

- loanData$Property_Area <- as.integer(factor(loanData$Property_Area))

#Credit History and Dependents convert to numeric

- loanData$Credit_History<-as.integer(loanData$Credit_History)
- loanData$Dependents<-as.integer(loanData$Dependents)
- cleanData_bfscale <- subset(loanData,select = -c(Loan_ID,Loan_Status))

#Scaling

- af_scale <- data.frame(scale(cleanData_bfscale))
- af_scale$Loan_Status <-loanData$Loan_Status

MODEL BUILDING

1. Train Test Split

#Splitting the dataset into the Training set and Test set

- set.seed(100) #randomization`
- train_sample <- sample(nrow(af_scale), 0.75 * nrow(af_scale))

#splitting data into training/testing data using the trainIndex object

- trainData <- af_scale[train_sample,] #training data (75% of data)
- testData <- af_scale[-train_sample,] #testing data (25% of data)

Check whether data set fairly even split

- prop.table(table(trainData$Loan_Status))#train
- prop.table(table(testData$Loan_Status))#test

```
> # Check whether data set fairly even split
> prop.table(table(trainData$Loan_Status))#train

        0         1 
0.3108696 0.6891304 
> prop.table(table(testData$Loan_Status))#test

        0         1 
0.3181818 0.6818182 
> 
```

2. LOGISTIC REGRESSION

glm() --> for generalized linear model and can be used to compute Logistic Regression

family = binomial is specified to perform binary classification

Predictions can be easily made using the function predict(). Use the option type = “response” to directly obtain the probabilities

glm model with all Features

- model_all <- glm(Loan_Status ~., data = trainData, family = binomial)
- pred_all <- predict.glm(model_all,testData[-12],type = 'response')
- pclass_all <- ifelse(pred_all<0.5,0,1)
- confusionMatrix(table(as.factor(testData$Loan_Status),pclass_all),positive = '1')

```
Confusion Matrix and Statistics

   pclass_all
      0   1
  0  18  31
  1   1 104

               Accuracy : 0.7922
                 95% CI : (0.7195, 0.8533)
    No Information Rate : 0.8766
    P-Value [Acc > NIR] : 0.9989

                  Kappa : 0.4276

 Mcnemar's Test P-Value : 2.951e-07

            Sensitivity : 0.7704
            Specificity : 0.9474
         Pos Pred Value : 0.9905
         Neg Pred Value : 0.3673
             Prevalence : 0.8766
         Detection Rate : 0.6753
   Detection Prevalence : 0.6818
      Balanced Accuracy : 0.8589

       'Positive' Class : 1
```

Features will minimum p value are good features

glm model with single feature Credit_History

- model1ch <- glm(Loan_Status ~Credit_History, data = trainData, family = binomial)
- pred_m1 <- predict.glm(model1ch,testData[-12],type = 'response')
- pclass_m1 <- ifelse(pred_m1<0.5,0,1)
- confusionMatrix(table(as.factor(testData$Loan_Status),pclass_m1),positive = '1')

```
Confusion Matrix and Statistics

   pclass_m1
      0   1
  0  18  31
  1   1 104

               Accuracy : 0.7922
                 95% CI : (0.7195, 0.8533)
    No Information Rate : 0.8766
    P-Value [Acc > NIR] : 0.9989

                  Kappa : 0.4276

 Mcnemar's Test P-Value : 2.951e-07

            Sensitivity : 0.7704
            Specificity : 0.9474
         Pos Pred Value : 0.9905
         Neg Pred Value : 0.3673
             Prevalence : 0.8766
         Detection Rate : 0.6753
   Detection Prevalence : 0.6818
      Balanced Accuracy : 0.8589

       'Positive' Class : 1
```

#Logistic curve

- ggplot(data=trainData,aes(Credit_History, Loan_Status)) +

  geom_point(alpha = 0.2) +

  geom_smooth(method = "glm", method.args = list(family = "binomial")) +

  labs(

   title = "Logistic Regression Model",

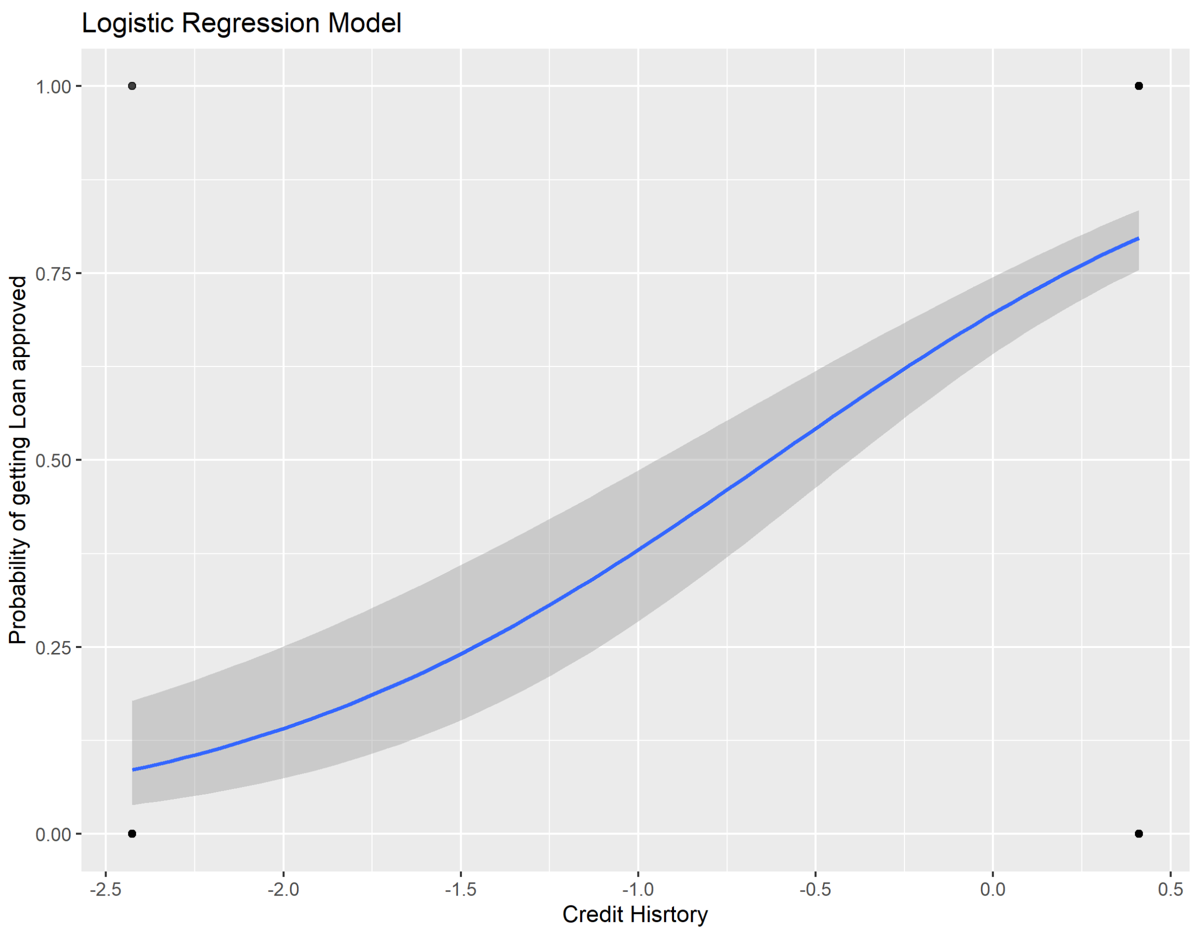
   x = "Credit Hisrtory",

   y = "Probability of getting Loan approved"

)

- ggsave('Logistic_curve.png')

#No improvement in model accuracy



3. Random Forest

Bagging stands for bootstrap aggregating. It consists of building multiple different decision tree models from a single training data set by repeatedly using multiple bootstrapped subsets of the data and averaging the models. Here, each tree is build independently to the others.

Random Forest algorithm, is one of the most commonly used and the most powerful machine learning techniques. It is a special type of bagging applied to decision trees.

#random forest with default parameters and all features

- set.seed(100)
- original_rf<-randomForest(as.factor(Loan_Status)~ ., trainData,OOB=TRUE)
- original_rf

```
Call:
 randomForest(formula = as.factor(Loan_Status) ~ ., data = trainData,      OOB = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 20.22%
Confusion matrix:
   0   1 class.error
0 66  77  0.53846154
1 16 301  0.05047319
>
```

- pred <- predict(original_rf,testData[-12])
- confusionMatrix(as.factor(testData$Loan_Status),pred)

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  19  30
         1   2 103

               Accuracy : 0.7922
                 95% CI : (0.7195, 0.8533)
    No Information Rate : 0.8636
    P-Value [Acc > NIR] : 0.9947

                  Kappa : 0.435

 Mcnemar's Test P-Value : 1.815e-06

            Sensitivity : 0.9048
            Specificity : 0.7744
         Pos Pred Value : 0.3878
         Neg Pred Value : 0.9810
             Prevalence : 0.1364
         Detection Rate : 0.1234
   Detection Prevalence : 0.3182
      Balanced Accuracy : 0.8396

       'Positive' Class : 0
```

- accuracy = sum(testData$Loan_Status == pred)/length(testData$Loan_Status)
  #0.7922078
  #no improvement observed as compared to glm model

Hyper Parameter Tunning – Random Forest

Hyperparameter tunning for Random Forest

- set.seed(10)
- tune_grid<-expand.grid(mtry=c(1:10), ntree=c(500,1000,1500,2000,2500,3000)) #expand a grid of parameters
- mtry<-tune_grid[[1]]
- ntree<-tune_grid[[2]] #using vectors instead of dataframe to subset is faster in for loop
- OOB<-NULL #use to store calculated OOB error estimate

- for(i in 1:nrow(tune_grid)){

  rf<-randomForest(as.factor(Loan_Status)~. ,trainData, mtry=mtry[i], ntree=ntree[i])

  confusion<-rf$confusion

  temp<-(confusion[2]+confusion[3])/614 #calculate the OOB error estimate

  OOB<-append(OOB,temp)

  }

- tune_grid$OOB<-OOB
- head(tune_grid[order(tune_grid["OOB"]),], 4) #order the results

```
#       mtry   ntree      OOB
# 22     2  1500 0.1400651
# 52     2  3000 0.1400651
# 2      2   500 0.1416938
# 32     2  2000 0.1416938
```

#We have optimal paramater with lowest OOB score mtry:2 and ntree:1500

#fit model with optimal Parameters

- final_rf <- randomForest(as.factor(Loan_Status)~. ,trainData, mtry=2, ntree=1500)

```
> final_rf

Call:
 randomForest(formula = as.factor(Loan_Status) ~ ., data = trainData,      mtry = 2, ntree = 1500)
               Type of random forest: classification
                     Number of trees: 1500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 18.91%
Confusion matrix:
   0   1 class.error
0 64  79  0.55244755
1  8 309  0.02523659
>
```

#Comparing the above random forest models :

#model original_rf has better train accuracy but low test accuracy (its getting over fitted)

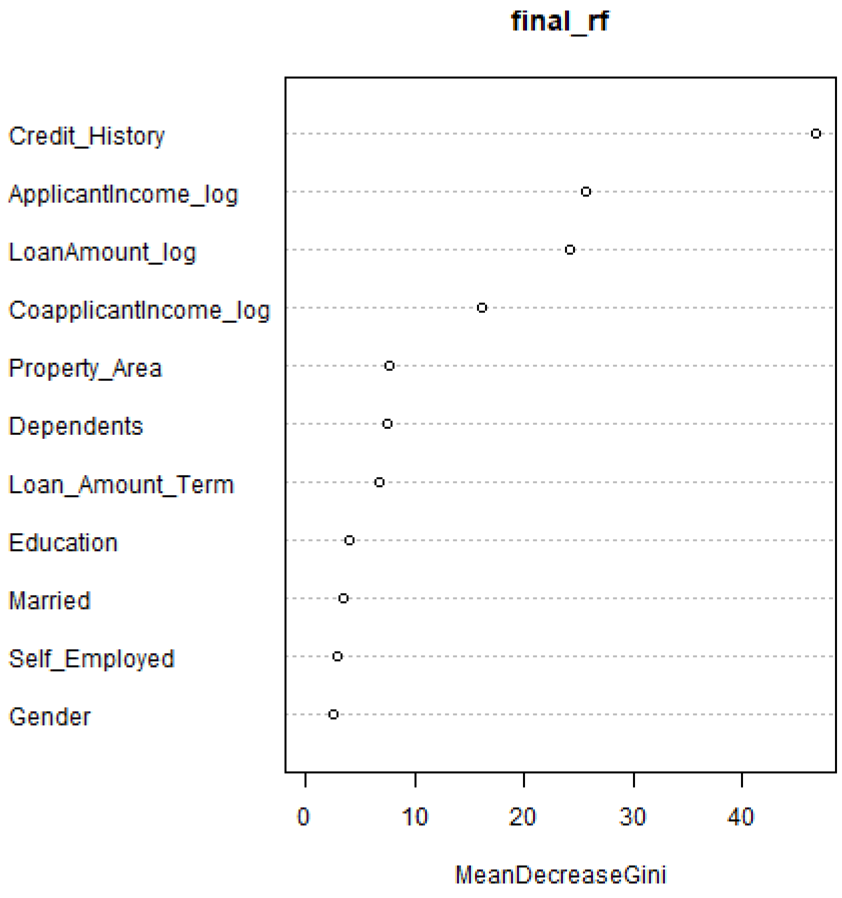
#model obtained with parameter tunning is the best fitted model as compared

Important Features

- png("FeatureImpPlot.png")
- featureImp_plt <- varImpPlot(final_rf)
- dev.off()

#Important Features are :

#Credit_History(strongest of all) , ApplicantIncome_log ,LoanAmount_log , CoapplicantIncome_log



4. GRADIENT BOOSTING

- set.seed(101)
- xgmat <-data.matrix(trainData[-12]) #convert dataframe to matrix
- xgmat_test <- data.matrix(testData[-12])
- xgb <- xgboost(data=xgmat,label = trainData$Loan_Status,nrounds =100)

```
> xgb <- xgboost(data=xgmat,label = trainData$Loan_Status,nrounds =100)
[1]     train-rmse:0.422693
[2]     train-rmse:0.370246
[3]     train-rmse:0.331519
[4]     train-rmse:0.305919
[5]     train-rmse:0.283536
[6]     train-rmse:0.270067
[7]     train-rmse:0.259062
[8]     train-rmse:0.248201
[9]     train-rmse:0.242699
[10]    train-rmse:0.233021
[11]    train-rmse:0.225370
[12]    train-rmse:0.207365
[13]    train-rmse:0.200783
[14]    train-rmse:0.195978
```

- xgb

```
> xgb
##### xgb.Booster
raw: 383.2 Kb
call:
  xgb.train(params = params, data = dtrain, nrounds = nrounds,
    watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
    early_stopping_rounds = early_stopping_rounds, maximize = maximize,
    save_period = save_period, save_name = save_name, xgb_model = xgb_model,
    callbacks = callbacks)
params (as set within xgb.train):
  validate_parameters = "TRUE"
xgb.attributes:
  niter
callbacks:
  cb.print.evaluation(period = print_every_n)
  cb.evaluation.log()
# of features: 11
niter: 100
nfeatures : 11
evaluation_log:
    iter train_rmse
       1   0.422693
       2   0.370246
---
      99   0.013028
     100   0.012902
>
```

- xgb_pred <- predict(xgb,newdata = xgmat_test)
- xgb_pclass <- ifelse(xgb_pred<0.5,0,1)
- confusionMatrix(table(as.factor(testData$Loan_Status),xgb_pclass),positive = '1')

```
Confusion Matrix and Statistics

   xgb_pclass
     0  1
  0 24 25
  1 13 92

               Accuracy : 0.7532
                 95% CI : (0.6774, 0.8191)
    No Information Rate : 0.7597
    P-Value [Acc > NIR] : 0.61707

                  Kappa : 0.3916

 Mcnemar's Test P-Value : 0.07435

            Sensitivity : 0.7863
            Specificity : 0.6486
         Pos Pred Value : 0.8762
         Neg Pred Value : 0.4898
             Prevalence : 0.7597
         Detection Rate : 0.5974
   Detection Prevalence : 0.6818
      Balanced Accuracy : 0.7175

       'Positive' Class : 1
```

5. PREDICTIONS

#prediction

- predictions <- predict(final_rf,af_scale)
- predClass <- ifelse(predictions ==1,'Y','N')

#Final Submission score 0.770833333333333

SUMMARY

- This is the end of the analysis, we started from data cleaning and processing, missing value imputation , then exploratory analysis and feature engineering, and finally model building and evaluation.

- The best accuracy we obtained on our validation data is 0.7792 ,Since dataset is not huge enough models performance does not vary but Random forest performed with hyperparameter tunning.

- The insights about loan approval status from the analysis is:

    - Applicants with credit history not passing guidelines mostly fails to get approved, probably because that they have a higher probability of not paying back.
    - Most of the time, applicants with high income, loaning low amount is more likely to get approved, which makes sense, those applicants are more likely to pay back their loans.
    - Having a strong co-applicant can be a plus to the probability of getting approve.