DATA 622 - Homework3 - Loan Approval Prediction

Group1: Diego Correa, Amanda Arce, Soumya Ghosh & Atina Karim

October 10, 2021

Table of Contents

[Libraries 2](#_Toc85971535)

[Background 2](#_Toc85971536)

[Data Dictionary 3](#_Toc85971537)

[Problem Statement 3](#_Toc85971538)

[Dataset 3](#_Toc85971539)

[Descriptive Dataset Summary 6](#_Toc85971540)

[Pre-Processing 8](#_Toc85971541)

[Missing Value Analysis 8](#_Toc85971542)

[Data Imputation 11](#_Toc85971543)

[Exploratory Data Analysis 13](#_Toc85971544)

[Categorical Variables 13](#_Toc85971545)

[Continuous Variables 14](#_Toc85971546)

[Observations: 15](#_Toc85971547)

[Further Analysis 15](#_Toc85971548)

[Feature Engineering 17](#_Toc85971549)

[Correlation Plot: Multicollinearity Check 20](#_Toc85971550)

[Model Building 21](#_Toc85971551)

[Splitting Data: Train/Test 21](#_Toc85971552)

[Model1: LDA Algorithm 21](#_Toc85971553)

[Model Summary 22](#_Toc85971554)

[Model2: K-nearest neighbor (KNN) algorithm 23](#_Toc85971555)

[Model Optimization 24](#_Toc85971556)

[Accuracy Plot 25](#_Toc85971557)

[Model Summary 27](#_Toc85971558)

[Model3: Decision Tree 28](#_Toc85971559)

[Converting Categorical Variables to Factors in Train 28](#_Toc85971560)

[Converting Categorical Variables to Factors in Test 28](#_Toc85971561)

[Initial Decision Tree 28](#_Toc85971562)

[Predicting on the Test Data 31](#_Toc85971563)

[Predicting Pruned Tree on Test Data 33](#_Toc85971564)

[Model Summary 34](#_Toc85971565)

[Model4: Random Forest 34](#_Toc85971566)

[Model Summary 37](#_Toc85971567)

[Model Performance Comparision 37](#_Toc85971568)

[Conclusion 39](#_Toc85971569)

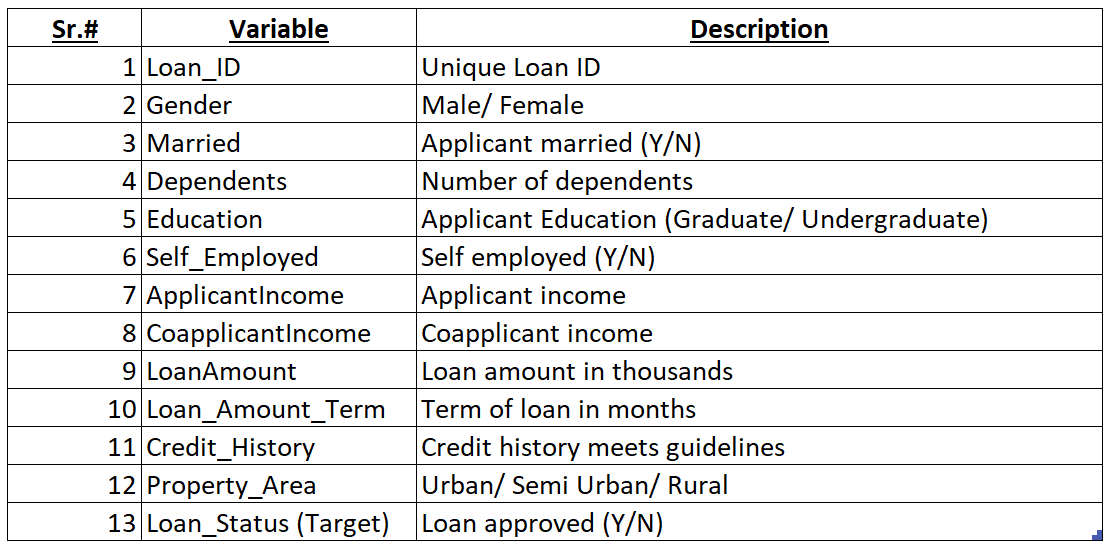
## Libraries

library(kableExtra)  
library(tidyverse)  
library(ggplot2)  
library(dplyr)  
library(psych)  
library(caret)  
library(mice)  
library(randomForest)  
library(caTools)  
library(corrplot)  
library(class)  
library(rpart)  
library(rpart.plot)  
library(naniar)

## Background

For this assignment, we will be working with a dataset on loan approval status. The **‘Loan\_Status’** is the target variable here -

### Data Dictionary



**Loan Approval Status Data Dictionary**

### Problem Statement

1. As we begin working with the dataset, we will conduct a thorough exploratory data analysis. This step is necessary as we figure out which variables should be included in models. (10 points)
2. We will use the LDA algorithm to predict the loan approval status. This will include the walk through for the steps we took, and how we decided on the key variables. (40 points)
3. Use K-nearest neighbor (KNN) algorithm to predict the loan approval status variable. Please be sure to walk through the steps you took. This includes talking about what value for ‘k’ you settled on and why. (40 points)
4. Use Decision Trees to predict on loan approval status. (40 points)
5. Use Random Forests to predict on loan approval status. (40 points)
6. Model performance: Comparison of the models we settled on in problem # 2- 5. Comment on their relative performance. Which one would you prefer the most? Why? (5 points)

## Dataset

dataset <- dataset <- read.csv('https://raw.githubusercontent.com/dcorrea614/DATA622/main/HW3/Loan\_approval.csv')  
head(dataset)%>% kable() %>% kable\_styling(bootstrap\_options = c("striped", "hover", "condensed", "responsive")) %>%   
 scroll\_box(width="100%",height="300px")

Loan\_ID

Gender

Married

Dependents

Education

Self\_Employed

ApplicantIncome

CoapplicantIncome

LoanAmount

Loan\_Amount\_Term

Credit\_History

Property\_Area

Loan\_Status

LP001002

Male

No

0

Graduate

No

5849

0

NA

360

1

Urban

Y

LP001003

Male

Yes

1

Graduate

No

4583

1508

128

360

1

Rural

N

LP001005

Male

Yes

0

Graduate

Yes

3000

0

66

360

1

Urban

Y

LP001006

Male

Yes

0

Not Graduate

No

2583

2358

120

360

1

Urban

Y

LP001008

Male

No

0

Graduate

No

6000

0

141

360

1

Urban

Y

LP001011

Male

Yes

2

Graduate

Yes

5417

4196

267

360

1

Urban

Y

### Descriptive Dataset Summary

summary(dataset)%>% kable() %>% kable\_styling(bootstrap\_options = c("striped", "hover", "condensed", "responsive")) %>% scroll\_box(width="100%",height="400px")

Loan\_ID

Gender

Married

Dependents

Education

Self\_Employed

ApplicantIncome

CoapplicantIncome

LoanAmount

Loan\_Amount\_Term

Credit\_History

Property\_Area

Loan\_Status

Length:614

Length:614

Length:614

Length:614

Length:614

Length:614

Min. : 150

Min. : 0

Min. : 9.0

Min. : 12

Min. :0.0000

Length:614

Length:614

Class :character

Class :character

Class :character

Class :character

Class :character

Class :character

1st Qu.: 2878

1st Qu.: 0

1st Qu.:100.0

1st Qu.:360

1st Qu.:1.0000

Class :character

Class :character

Mode :character

Mode :character

Mode :character

Mode :character

Mode :character

Mode :character

Median : 3812

Median : 1188

Median :128.0

Median :360

Median :1.0000

Mode :character

Mode :character

NA

NA

NA

NA

NA

NA

Mean : 5403

Mean : 1621

Mean :146.4

Mean :342

Mean :0.8422

NA

NA

NA

NA

NA

NA

NA

NA

3rd Qu.: 5795

3rd Qu.: 2297

3rd Qu.:168.0

3rd Qu.:360

3rd Qu.:1.0000

NA

NA

NA

NA

NA

NA

NA

NA

Max. :81000

Max. :41667

Max. :700.0

Max. :480

Max. :1.0000

NA

NA

NA

NA

NA

NA

NA

NA

NA

NA

NA’s :22

NA’s :14

NA’s :50

NA

NA

## Pre-Processing

### Missing Value Analysis

Based on the above descriptive data summary, there are quite a few variables with missing values. So we conducted an analysis of all missing values in various attributes to identify proper imputation technique.

## Counts of missing data per feature  
dataset\_missing\_counts <- data.frame(apply(dataset, 2, function(x) length(which(is.na(x)))))  
dataset\_missing\_pct <- data.frame(apply(dataset, 2,function(x) {sum(is.na(x)) / length(x) \* 100}))  
  
dataset\_missing\_counts <- cbind(Feature = rownames(dataset\_missing\_counts), dataset\_missing\_counts, dataset\_missing\_pct)  
colnames(dataset\_missing\_counts) <- c('Feature','NA\_Count','NA\_Percentage')  
rownames(dataset\_missing\_counts) <- NULL  
  
dataset\_missing\_counts %>% filter(`NA\_Count` != 0) %>% arrange(desc(`NA\_Count`)) %>% kable() %>% kable\_styling(bootstrap\_options = c("striped", "hover", "condensed", "responsive")) %>% scroll\_box(width="100%",height="300px")

Feature

NA\_Count

NA\_Percentage

Credit\_History

50

8.143323

LoanAmount

22

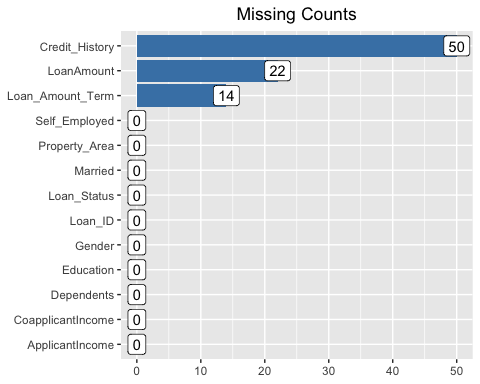
3.583062

Loan\_Amount\_Term

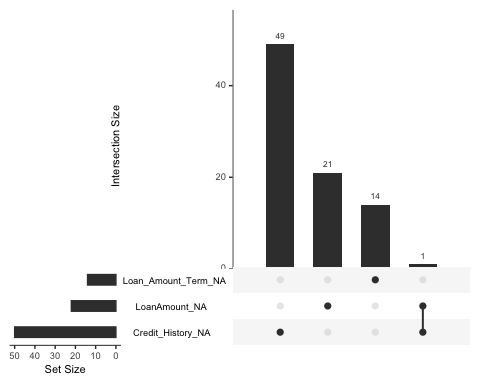
14

2.280130

ggplot(dataset\_missing\_counts, aes(x = NA\_Count, y = reorder(Feature, NA\_Count))) +   
 geom\_bar(stat = 'identity', fill = 'steelblue') +  
 geom\_label(aes(label = NA\_Count)) +  
 labs(title = 'Missing Counts') +  
 theme(plot.title = element\_text(hjust = 0.5), axis.title.y = element\_blank(), axis.title.x = element\_blank())



# Use nanair package to plot missing value patterns  
gg\_miss\_upset(dataset)



### Data Imputation

Based on above missing value analysis, we are going to perform data imputation using the **mice** package following Random Forest method. But before that, we converted all categorical variables into factors -

#transformation  
#Loan\_ID should be removed before imputing data  
#mice uses all data to impute  
dataset <- dataset %>%  
 select(-'Loan\_ID') %>%  
 mutate(  
 Gender = as.factor(Gender),  
 Married = as.factor(Married),  
 Dependents = as.factor(Dependents),  
 Education = as.factor(Education),  
 Self\_Employed = as.factor(Self\_Employed),  
 Credit\_History = as.factor(Credit\_History),  
 Property\_Area = as.factor(Property\_Area),  
 Loan\_Status = as.factor(Loan\_Status)  
 )

#imputation by using the random forest method ('rf')  
init <- mice(dataset, maxit = 0)  
predM <- init$predictorMatrix  
set.seed(123)  
imputed <- mice(dataset, method = 'rf', predictorMatrix = predM, m=5)

##   
## iter imp variable  
## 1 1 LoanAmount Loan\_Amount\_Term Credit\_History  
## 1 2 LoanAmount Loan\_Amount\_Term Credit\_History  
## 1 3 LoanAmount Loan\_Amount\_Term Credit\_History  
## 1 4 LoanAmount Loan\_Amount\_Term Credit\_History  
## 1 5 LoanAmount Loan\_Amount\_Term Credit\_History  
## 2 1 LoanAmount Loan\_Amount\_Term Credit\_History  
## 2 2 LoanAmount Loan\_Amount\_Term Credit\_History  
## 2 3 LoanAmount Loan\_Amount\_Term Credit\_History  
## 2 4 LoanAmount Loan\_Amount\_Term Credit\_History  
## 2 5 LoanAmount Loan\_Amount\_Term Credit\_History  
## 3 1 LoanAmount Loan\_Amount\_Term Credit\_History  
## 3 2 LoanAmount Loan\_Amount\_Term Credit\_History  
## 3 3 LoanAmount Loan\_Amount\_Term Credit\_History  
## 3 4 LoanAmount Loan\_Amount\_Term Credit\_History  
## 3 5 LoanAmount Loan\_Amount\_Term Credit\_History  
## 4 1 LoanAmount Loan\_Amount\_Term Credit\_History  
## 4 2 LoanAmount Loan\_Amount\_Term Credit\_History  
## 4 3 LoanAmount Loan\_Amount\_Term Credit\_History  
## 4 4 LoanAmount Loan\_Amount\_Term Credit\_History  
## 4 5 LoanAmount Loan\_Amount\_Term Credit\_History  
## 5 1 LoanAmount Loan\_Amount\_Term Credit\_History  
## 5 2 LoanAmount Loan\_Amount\_Term Credit\_History  
## 5 3 LoanAmount Loan\_Amount\_Term Credit\_History  
## 5 4 LoanAmount Loan\_Amount\_Term Credit\_History  
## 5 5 LoanAmount Loan\_Amount\_Term Credit\_History

dataset <- complete(imputed)  
summary(dataset)

## Gender Married Dependents Education Self\_Employed  
## : 13 : 3 : 15 Graduate :480 : 32   
## Female:112 No :213 0 :345 Not Graduate:134 No :500   
## Male :489 Yes:398 1 :102 Yes: 82   
## 2 :101   
## 3+: 51   
##   
## ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term  
## Min. : 150 Min. : 0 Min. : 9.0 Min. : 12   
## 1st Qu.: 2878 1st Qu.: 0 1st Qu.:100.0 1st Qu.:360   
## Median : 3812 Median : 1188 Median :128.0 Median :360   
## Mean : 5403 Mean : 1621 Mean :146.1 Mean :342   
## 3rd Qu.: 5795 3rd Qu.: 2297 3rd Qu.:167.8 3rd Qu.:360   
## Max. :81000 Max. :41667 Max. :700.0 Max. :480   
## Credit\_History Property\_Area Loan\_Status  
## 0: 92 Rural :179 N:192   
## 1:522 Semiurban:233 Y:422   
## Urban :202   
##   
##   
##

We also checked for presence of any de-generate variables and found no such variable present in our dataset -

# none of the variables meet the condition to be a degenerate feature  
nearZeroVar(dataset)

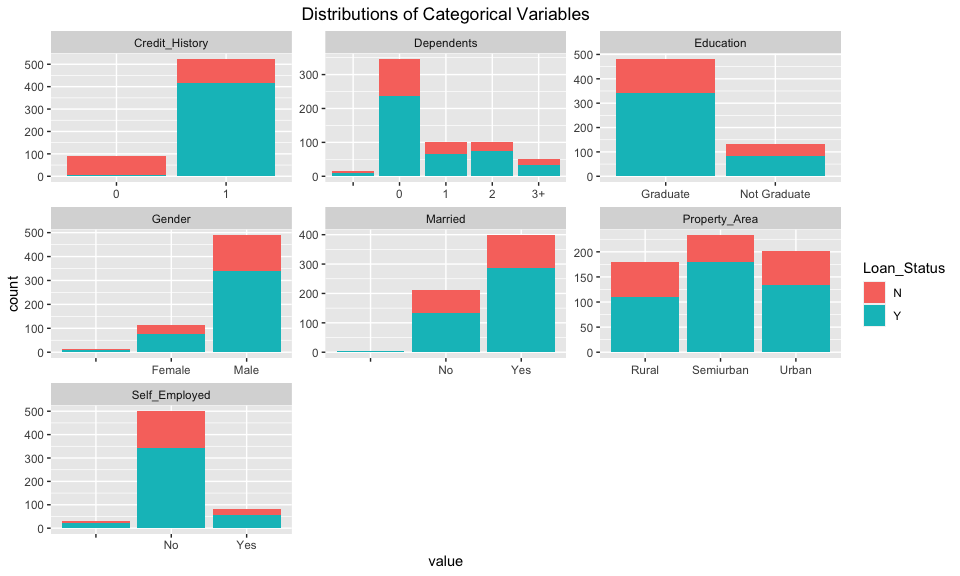
## integer(0)

## Exploratory Data Analysis

We did separate data analysis for categorical and continuous variables -

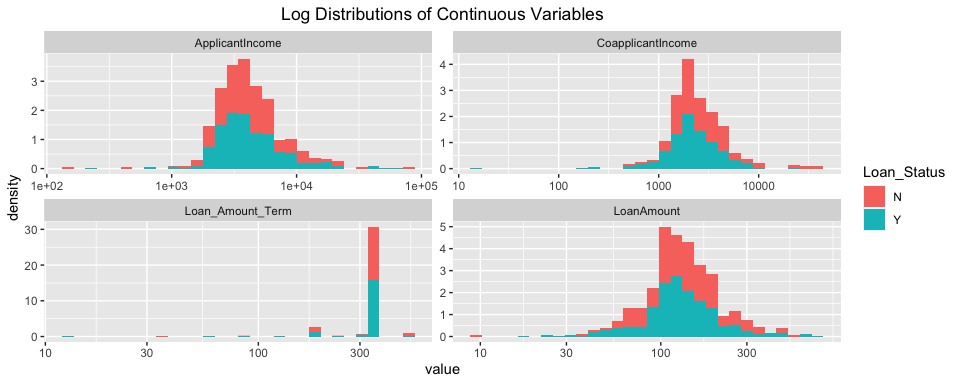
### Categorical Variables

cat\_vars <- dataset %>%  
 dplyr::select(-c('ApplicantIncome', 'CoapplicantIncome','LoanAmount','Loan\_Amount\_Term')) %>%  
 gather(key = 'predictor\_variable', value = 'value', -Loan\_Status)  
  
# Plot and print a histogram for each predictor variable.  
ggplot(cat\_vars) +  
 geom\_histogram(aes(x = value, fill = Loan\_Status),stat='count', bins = 30) +  
 labs(title = 'Distributions of Categorical Variables') +  
 theme(plot.title = element\_text(hjust = 0.5)) +  
 facet\_wrap(. ~predictor\_variable, scales = 'free', ncol = 3)



### Continuous Variables

cont\_vars <- dataset %>%  
 dplyr::select(ApplicantIncome, CoapplicantIncome, LoanAmount, Loan\_Amount\_Term, Loan\_Status) %>%  
 gather(key = 'predictor\_variable', value = 'value', -Loan\_Status)  
  
# Plot and print a histogram for each predictor variable.  
ggplot(cont\_vars) +  
 geom\_histogram(aes(x = value, y = ..density.., fill = Loan\_Status), bins = 30) +  
 labs(title = 'Log Distributions of Continuous Variables') +  
 scale\_x\_log10() +  
 theme(plot.title = element\_text(hjust = 0.5)) +  
 facet\_wrap(. ~predictor\_variable, scales = 'free', ncol = 2)

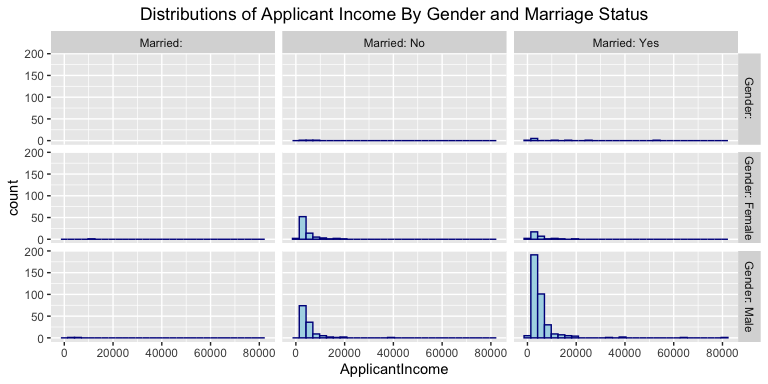


### Observations:

* More males have applied for loans than females and and they also have a higher rate of approval.
* More married couples have applied for loans.
* Self employed individuals have applied for less loans which indicates salary earners apply for and obtain more loans.
* People with better credit history guidelines are more likely to get their loans approved as they have higher chances of paying back the loan on time.
* People leaving in Semi-Urban area have most loan applications and have a higher rate of approval followed by urban and rural areas. Especially Rural loan applicants have a lower rate of loan approval.
* An extremely high number of them go for a 360 months loan term. That’s pay back within a 15 years period.
* People with *no dependents* tend to have applied for more loan applications
* People with a graduate degree have applied for more loans than w/o a graduate degree and have much higher rate of loan approvals.
* We can see that *ApplicantIncome*, *CoapplicantIncome* and *LoanAmount* are highly right-skewed, with long right tails. Conversely, it looks like *Loan\_Amount\_Term* is highly left-skewed, with a long left tail.

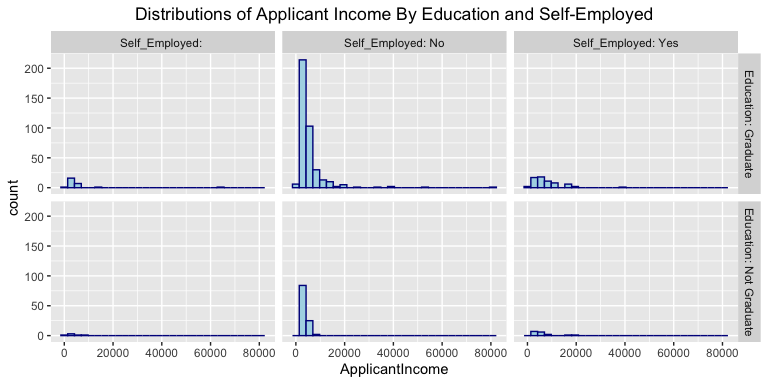
### Further Analysis

# Plot and print a histogram for a pair of predictor variables.  
bp <- ggplot(dataset, aes(x = ApplicantIncome)) +  
 geom\_histogram(bins = 30, color = "darkblue", fill = "lightblue") +  
 labs(title = 'Distributions of Applicant Income By Gender and Marriage Status') +  
 theme(plot.title = element\_text(hjust = 0.5))   
  
bp + facet\_grid(Gender ~ Married, labeller=label\_both)



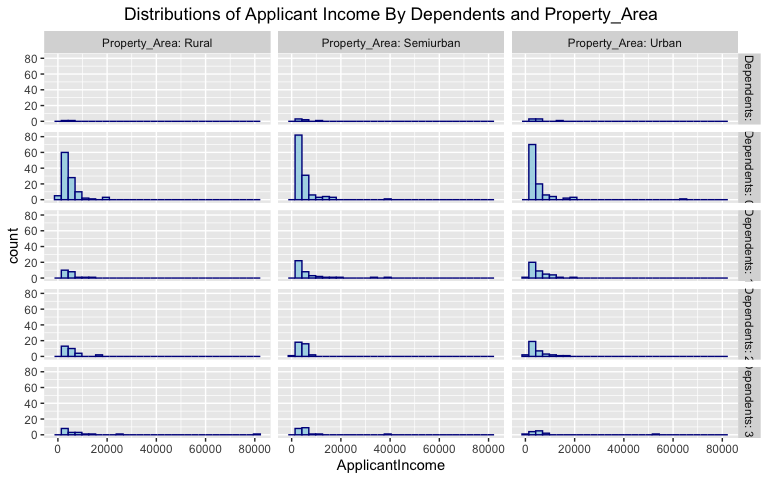
It can be observed from above plot that married males have applied for more loans and comparatively unmarried females have applied for more loans than married females.

# Plot and print a histogram for each predictor variable.  
bp <- ggplot(dataset, aes(x = ApplicantIncome)) +  
 geom\_histogram(bins = 30, color = "darkblue", fill = "lightblue") +  
 labs(title = 'Distributions of Applicant Income By Education and Self-Employed') +  
 theme(plot.title = element\_text(hjust = 0.5))   
  
bp + facet\_grid(Education ~ Self\_Employed, labeller=label\_both)



Above plot shows, people with a graduate degree and having a salaried job applied for more loans than Self-employed and non-graduate folks.

# Plot and print a histogram for each predictor variable.  
bp <- ggplot(dataset, aes(x = ApplicantIncome)) +  
 geom\_histogram(bins = 30, color = "darkblue", fill = "lightblue") +  
 labs(title = 'Distributions of Applicant Income By Dependents and Property\_Area') +  
 theme(plot.title = element\_text(hjust = 0.5))   
  
bp + facet\_grid(Dependents ~ Property\_Area, labeller=label\_both)



From the above plot, it can be observed that people with no dependents have applied for more loans and people living in semi-urban areas also applied for more loans.

## Feature Engineering

Converting all categorical variables present in the dataset to numeric codes -

# Caret package dummyVars() to do one hot encoding  
#dummy <- dummyVars(" ~ .", data=dataset)  
#newdata <- data.frame(predict(dummy, newdata = dataset))  
  
# Converting categorical variables to numeric  
newdata <- dataset %>% mutate\_if(is.factor, as.numeric)

Next, we’ll have to work through a few transformations for our highly skewed continuous data. We will use log transformation to normalize the data.

newdata <- newdata %>%  
 mutate(  
 ApplicantIncome = log(ApplicantIncome),  
 CoapplicantIncome = log(CoapplicantIncome),  
 LoanAmount = log(LoanAmount),  
 Loan\_Amount\_Term = log(Loan\_Amount\_Term))  
  
newdata <- newdata %>%  
 mutate(CoapplicantIncome = ifelse(CoapplicantIncome < 0, 0, CoapplicantIncome))  
   
head(newdata) %>% kable() %>% kable\_styling(bootstrap\_options = c("striped", "hover", "condensed", "responsive")) %>%   
 scroll\_box(width="100%",height="300px")

Gender

Married

Dependents

Education

Self\_Employed

ApplicantIncome

CoapplicantIncome

LoanAmount

Loan\_Amount\_Term

Credit\_History

Property\_Area

Loan\_Status

3

2

2

1

2

8.674026

0.000000

5.049856

5.886104

2

3

2

3

3

3

1

2

8.430109

7.318539

4.852030

5.886104

2

1

1

3

3

2

1

3

8.006368

0.000000

4.189655

5.886104

2

3

2

3

3

2

2

2

7.856707

7.765569

4.787492

5.886104

2

3

2

3

2

2

1

2

8.699515

0.000000

4.948760

5.886104

2

3

2

3

3

4

1

3

8.597297

8.341887

5.587249

5.886104

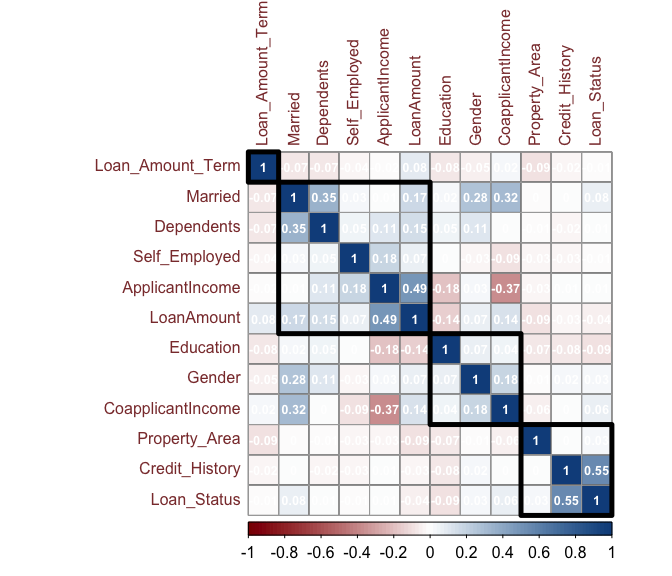
2

3

2

### Correlation Plot: Multicollinearity Check

corrMatrix <- round(cor(newdata),4)  
  
corrMatrix %>% corrplot(., method = "color", outline = T, addgrid.col = "darkgray", order="hclust", addrect = 4, rect.col = "black", rect.lwd = 5,cl.pos = "b", tl.col = "indianred4", tl.cex = 1.0, cl.cex = 1.0, addCoef.col = "white", number.digits = 2, number.cex = 0.8, col = colorRampPalette(c("darkred","white","dodgerblue4"))(100))



Based on above plot, it can be concluded there is no multicollinearity present in the dataset. There are certain variable pairs like **LoanAmount & ApplicantIncome** and **Credit\_History & Loan\_Status** that have higher correlation due to obvious reasons.

## Model Building

### Splitting Data: Train/Test

We are going to do a 75-25% split for training and test purposes.

sample = sample.split(newdata$Loan\_Status, SplitRatio = 0.75)  
train = subset(newdata, sample == TRUE)  
test = subset(newdata, sample == FALSE)  
  
#head(train)%>% kable() %>% kable\_styling(bootstrap\_options = c("striped", "hover", "condensed", "responsive")) %>% scroll\_box(width="100%",height="300px")  
  
#Creating seperate dataframe for 'Loan\_Status' feature which is our target.  
train.loan\_labels <- train[,12]  
test.loan\_labels <- test[,12]  
  
train1 <- train[,-12]  
test1 <- test[,-12]

### Model1: LDA Algorithm

Before we start building the LDA model, we applied some basic transformation.

#convert loan\_status variable as factor to work with LDA model  
train$Loan\_Status <- factor(train$Loan\_Status)

We built the LDA model using caret’s train() method -

library(caret)  
#create lda model using caret's train()  
lda <- caret::train(Loan\_Status ~ .,  
 train, method = 'lda', trControl = trainControl(method = "cv"))  
#output lda  
lda

## Linear Discriminant Analysis   
##   
## 460 samples  
## 11 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 415, 414, 414, 414, 413, 414, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8174232 0.4945496

Below is the summary of the generated model -

#run summary on lda model using lda variable  
summary(lda)

## Length Class Mode   
## prior 2 -none- numeric   
## counts 2 -none- numeric   
## means 22 -none- numeric   
## scaling 11 -none- numeric   
## lev 2 -none- character  
## svd 1 -none- numeric   
## N 1 -none- numeric   
## call 3 -none- call   
## xNames 11 -none- character  
## problemType 1 -none- character  
## tuneValue 1 data.frame list   
## obsLevels 2 -none- character  
## param 0 -none- list

Then we checked the prediction against actual value in tabular form -

#create variable called pred.Loan\_status to make prediction on test data using the lda (variable/model) model we made  
pred.Loan\_Status = predict(lda, test)  
#organize results in a table  
table(pred.Loan\_Status, test$Loan\_Status)

##   
## pred.Loan\_Status 1 2  
## 1 23 5  
## 2 25 101

We calculated the prediction accuracy -

#create variable called pred.accuracy to show accuracy -   
pred.accuracy = round(mean(pred.Loan\_Status == test$Loan\_Status)\*100,2)  
pred.accuracy

## [1] 80.52

The calculated accuracy in this case being 82.47% which is a pretty good result.

#### Model Summary

We record the summary of the LDA model metrics in a data frame -

lda\_model <- confusionMatrix(table(pred.Loan\_Status ,test$Loan\_Status))$byClass  
lda\_accuracy <- confusionMatrix(table(pred.Loan\_Status ,test$Loan\_Status))$overall['Accuracy']  
lda\_model <- data.frame(lda\_model)  
lda\_model <- rbind("Accuracy" = lda\_accuracy, lda\_model)

### Model2: K-nearest neighbor (KNN) algorithm

For this model, we will need to identify the appropriate value for *K*. K represents the number of nearby neighbors used to determine the sample data point class. A rule of thumb is to start with the square root of the number of observations (i.e., rows or records). This varies depending on the number of records, and typically this rule of thumb leads to slightly higher than optimal values, but it’s a good starting point. Later we will use the caret package to help find the optimal value for K.

#Find the number of observation  
NROW(train$Loan\_Status)

## [1] 460

So, we have 460 observations in our training data set.

# initial value for K  
sqrt(nrow(train))

## [1] 21.44761

The square root of 460 is around 21.45, therefore we’ll create two models. One with ‘K’ value as 21 and the other model with a ‘K’ value as 22.

knn.21 <- knn(train=train1, test=test1, cl=train.loan\_labels, k=21)  
knn.22 <- knn(train=train1, test=test1, cl=train.loan\_labels, k=22)  
  
#Calculate the proportion of correct classification for k = 21, 22  
ACC.21 <- 100 \* sum(test.loan\_labels == knn.21)/NROW(test.loan\_labels)  
ACC.22 <- 100 \* sum(test.loan\_labels == knn.22)/NROW(test.loan\_labels)  
  
# Print Accuracy Scores  
ACC.21

## [1] 70.12987

ACC.22

## [1] 70.12987

# Check prediction against actual value in tabular form for k=21  
table(knn.21 ,test.loan\_labels)

## test.loan\_labels  
## knn.21 1 2  
## 1 2 0  
## 2 46 106

# Check prediction against actual value in tabular form for k=22  
confusionMatrix(table(knn.22 ,test.loan\_labels))

## Confusion Matrix and Statistics  
##   
## test.loan\_labels  
## knn.22 1 2  
## 1 2 0  
## 2 46 106  
##   
## Accuracy : 0.7013   
## 95% CI : (0.6224, 0.7723)  
## No Information Rate : 0.6883   
## P-Value [Acc > NIR] : 0.4011   
##   
## Kappa : 0.0565   
##   
## Mcnemar's Test P-Value : 3.247e-11   
##   
## Sensitivity : 0.04167   
## Specificity : 1.00000   
## Pos Pred Value : 1.00000   
## Neg Pred Value : 0.69737   
## Prevalence : 0.31169   
## Detection Rate : 0.01299   
## Detection Prevalence : 0.01299   
## Balanced Accuracy : 0.52083   
##   
## 'Positive' Class : 1   
##

Our resulting confusion matrix shows the kNN predicted loan\_approval\_status compared to the actual loan status. We can see that K=22 performed well on our test set with an overall Accuracy=70.78%. While this model performed well, our choice of K=22 might be sub-optimal. Next, we will try to find the optimal value for K through testing.

#### Model Optimization

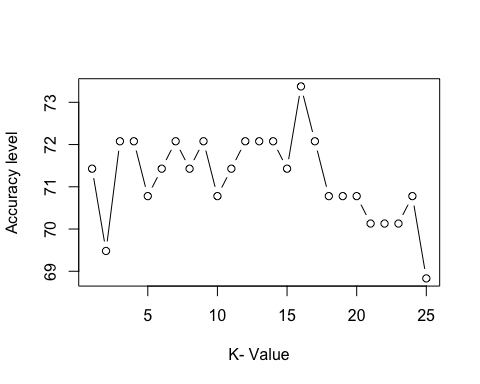
In order to improve the accuracy of the model, you can use n number of techniques such as the Elbow method and maximum percentage accuracy graph. In the below code snippet, I’ve created a loop that calculates the accuracy of the KNN model for ‘K’ values ranging from 1 to 25. This way you can check which ‘K’ value will result in the most accurate model:

i=1  
k.optm=1  
for (i in 1:25){  
 knn.mod <- knn(train=train1, test=test1, cl=train.loan\_labels, k=i)  
 k.optm[i] <- 100 \* sum(test.loan\_labels == knn.mod)/NROW(test.loan\_labels)  
 k=i  
 cat(k,'=',k.optm[i],'  
')  
}

## 1 = 71.42857   
## 2 = 69.48052   
## 3 = 72.07792   
## 4 = 72.07792   
## 5 = 70.77922   
## 6 = 71.42857   
## 7 = 72.07792   
## 8 = 71.42857   
## 9 = 72.07792   
## 10 = 70.77922   
## 11 = 71.42857   
## 12 = 72.07792   
## 13 = 72.07792   
## 14 = 72.07792   
## 15 = 71.42857   
## 16 = 73.37662   
## 17 = 72.07792   
## 18 = 70.77922   
## 19 = 70.77922   
## 20 = 70.77922   
## 21 = 70.12987   
## 22 = 70.12987   
## 23 = 70.12987   
## 24 = 70.77922   
## 25 = 68.83117

#### Accuracy Plot

#Accuracy plot  
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")



Based on the above plot, K=7 seems to perform best in terms of accuracy.

knn.7 <- knn(train=train1, test=test1, cl=train.loan\_labels, k=7)  
  
#Calculate the proportion of correct classification for k = 21, 22  
ACC.7 <- 100 \* sum(test.loan\_labels == knn.7)/NROW(test.loan\_labels)  
  
# Print Accuracy Scores  
ACC.7

## [1] 72.07792

# Check prediction against actual value in tabular form for k=21  
table(knn.7 ,test.loan\_labels)

## test.loan\_labels  
## knn.7 1 2  
## 1 9 4  
## 2 39 102

# Check prediction against actual value in tabular form for k=22  
confusionMatrix(table(knn.7 ,test.loan\_labels))

## Confusion Matrix and Statistics  
##   
## test.loan\_labels  
## knn.7 1 2  
## 1 9 4  
## 2 39 102  
##   
## Accuracy : 0.7208   
## 95% CI : (0.6429, 0.79)  
## No Information Rate : 0.6883   
## P-Value [Acc > NIR] : 0.2181   
##   
## Kappa : 0.1871   
##   
## Mcnemar's Test P-Value : 2.161e-07   
##   
## Sensitivity : 0.18750   
## Specificity : 0.96226   
## Pos Pred Value : 0.69231   
## Neg Pred Value : 0.72340   
## Prevalence : 0.31169   
## Detection Rate : 0.05844   
## Detection Prevalence : 0.08442   
## Balanced Accuracy : 0.57488   
##   
## 'Positive' Class : 1   
##

From our output, we can see that a K=7 was found to be the optimal value for our model based on the calculated Accuracy and Kappa values. Given this, a K=7 may ultimately perform better than our initial KNN model with K=22, especially on new data.

#### Model Summary

We record the summary of the kNN model metrics in a data frame -

knn\_model <- confusionMatrix(table(knn.7 ,test.loan\_labels))$byClass  
knn\_accuracy <- confusionMatrix(table(knn.7 ,test.loan\_labels))$overall['Accuracy']  
knn\_model <- data.frame(knn\_model)  
knn\_model <- rbind("Accuracy" = knn\_accuracy, knn\_model)  
  
knn\_model

## knn\_model  
## Accuracy 0.72077922  
## Sensitivity 0.18750000  
## Specificity 0.96226415  
## Pos Pred Value 0.69230769  
## Neg Pred Value 0.72340426  
## Precision 0.69230769  
## Recall 0.18750000  
## F1 0.29508197  
## Prevalence 0.31168831  
## Detection Rate 0.05844156  
## Detection Prevalence 0.08441558  
## Balanced Accuracy 0.57488208

### Model3: Decision Tree

Prior to building our decision tree - we will need to ensure that all categorical variables in our train and test set are coded as factors:

#### Converting Categorical Variables to Factors in Train

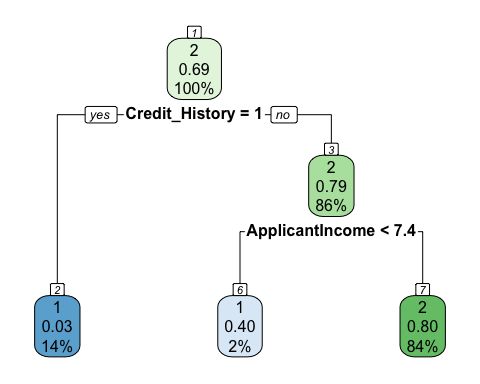
train2 <- train %>%  
   
 mutate(  
 Gender = as.factor(Gender),  
 Married = as.factor(Married),  
 Dependents = as.factor(Dependents),  
 Education = as.factor(Education),  
 Self\_Employed = as.factor(Self\_Employed),  
 Credit\_History = as.factor(Credit\_History),  
 Property\_Area = as.factor(Property\_Area),  
 Loan\_Status = as.factor(Loan\_Status))

#### Converting Categorical Variables to Factors in Test

test2 <- test %>%  
   
 mutate(  
 Gender = as.factor(Gender),  
 Married = as.factor(Married),  
 Dependents = as.factor(Dependents),  
 Education = as.factor(Education),  
 Self\_Employed = as.factor(Self\_Employed),  
 Credit\_History = as.factor(Credit\_History),  
 Property\_Area = as.factor(Property\_Area),  
 Loan\_Status = as.factor(Loan\_Status))

#### Initial Decision Tree

dt <- rpart(Loan\_Status ~ .,   
 data=train2, method="class")  
rpart.plot(dt, nn=TRUE)



Here are some of the details of the tree:

summary (dt)

## Call:  
## rpart(formula = Loan\_Status ~ ., data = train2, method = "class")  
## n= 460   
##   
## CP nsplit rel error xerror xstd  
## 1 0.41666667 0 1.0000000 1.0000000 0.06906903  
## 2 0.01388889 1 0.5833333 0.5833333 0.05754295  
## 3 0.01000000 2 0.5694444 0.6458333 0.05981817  
##   
## Variable importance  
## Credit\_History ApplicantIncome   
## 95 5   
##   
## Node number 1: 460 observations, complexity param=0.4166667  
## predicted class=2 expected loss=0.3130435 P(node) =1  
## class counts: 144 316  
## probabilities: 0.313 0.687   
## left son=2 (64 obs) right son=3 (396 obs)  
## Primary splits:  
## Credit\_History splits as LR, improve=63.928070, (0 missing)  
## Property\_Area splits as LRL, improve= 3.833137, (0 missing)  
## ApplicantIncome < 7.551114 to the left, improve= 2.646995, (0 missing)  
## Education splits as RL, improve= 2.381620, (0 missing)  
## LoanAmount < 4.085941 to the right, improve= 1.790726, (0 missing)  
##   
## Node number 2: 64 observations  
## predicted class=1 expected loss=0.03125 P(node) =0.1391304  
## class counts: 62 2  
## probabilities: 0.969 0.031   
##   
## Node number 3: 396 observations, complexity param=0.01388889  
## predicted class=2 expected loss=0.2070707 P(node) =0.8608696  
## class counts: 82 314  
## probabilities: 0.207 0.793   
## left son=6 (10 obs) right son=7 (386 obs)  
## Primary splits:  
## ApplicantIncome < 7.385511 to the left, improve=3.1678650, (0 missing)  
## Property\_Area splits as LRL, improve=2.4458090, (0 missing)  
## LoanAmount < 5.06575 to the right, improve=1.9042060, (0 missing)  
## Married splits as RLR, improve=0.9447752, (0 missing)  
## CoapplicantIncome < 8.015491 to the left, improve=0.8313020, (0 missing)  
## Surrogate splits:  
## CoapplicantIncome < 9.465733 to the right, agree=0.977, adj=0.1, (0 split)  
##   
## Node number 6: 10 observations  
## predicted class=1 expected loss=0.4 P(node) =0.02173913  
## class counts: 6 4  
## probabilities: 0.600 0.400   
##   
## Node number 7: 386 observations  
## predicted class=2 expected loss=0.1968912 P(node) =0.8391304  
## class counts: 76 310  
## probabilities: 0.197 0.803

Credit history and income seem to be some of the important predictors for loan approval.

#### Predicting on the Test Data

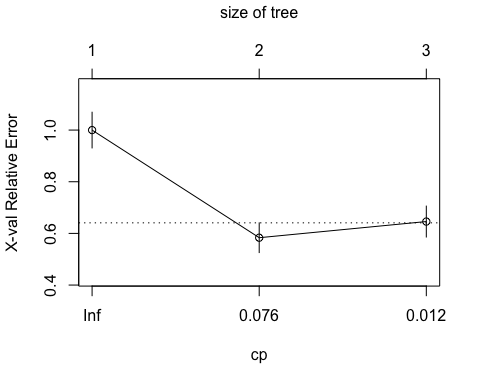
We are going to apply our tree to the test data and create a confusion table to evaluate the accuracy of the classifications.

tree.pred = predict(dt,test2,type="class") #because we want to predict the class labels  
# Confusion Tree  
confusionMatrix(predict(dt,type="class"), train2$Loan\_Status)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2  
## 1 68 6  
## 2 76 310  
##   
## Accuracy : 0.8217   
## 95% CI : (0.7836, 0.8556)  
## No Information Rate : 0.687   
## P-Value [Acc > NIR] : 4.015e-11   
##   
## Kappa : 0.5223   
##   
## Mcnemar's Test P-Value : 2.541e-14   
##   
## Sensitivity : 0.4722   
## Specificity : 0.9810   
## Pos Pred Value : 0.9189   
## Neg Pred Value : 0.8031   
## Prevalence : 0.3130   
## Detection Rate : 0.1478   
## Detection Prevalence : 0.1609   
## Balanced Accuracy : 0.7266   
##   
## 'Positive' Class : 1   
##

Our model is 82% accurate. This tree was grown to full depth and therefore there might be too many variables. To achieve improved accuracy, we need to prune the tree using the cross-validation:

plotcp(dt)

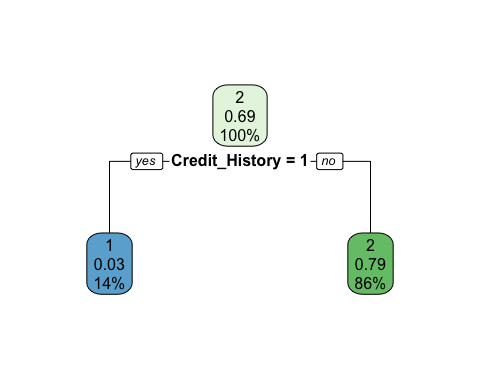


dt$cptable

## CP nsplit rel error xerror xstd  
## 1 0.41666667 0 1.0000000 1.0000000 0.06906903  
## 2 0.01388889 1 0.5833333 0.5833333 0.05754295  
## 3 0.01000000 2 0.5694444 0.6458333 0.05981817

The plot above shows the cross validated errors against the complexity parameters. The curve is at its lowest at 2, so we will prune our tree to a size of 2. At size 2, the error is ~0.58 and cp is 0.0138

prune\_dt=prune(dt,cp=0.0139)  
rpart.plot(prune\_dt)



#### Predicting Pruned Tree on Test Data

Prune\_pred <- predict(prune\_dt,   
 test2,   
 type="class")  
confusionMatrix(Prune\_pred, test2$Loan\_Status)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2  
## 1 23 5  
## 2 25 101  
##   
## Accuracy : 0.8052   
## 95% CI : (0.7337, 0.8645)  
## No Information Rate : 0.6883   
## P-Value [Acc > NIR] : 0.0007844   
##   
## Kappa : 0.4876   
##   
## Mcnemar's Test P-Value : 0.0005226   
##   
## Sensitivity : 0.4792   
## Specificity : 0.9528   
## Pos Pred Value : 0.8214   
## Neg Pred Value : 0.8016   
## Prevalence : 0.3117   
## Detection Rate : 0.1494   
## Detection Prevalence : 0.1818   
## Balanced Accuracy : 0.7160   
##   
## 'Positive' Class : 1   
##

Seems like pruning the tree reduced model accuracy to 81%. So we will stick to base decision tree model.

#### Model Summary

We record the summary of the Decision Tree model metrics in a data frame -

dtree\_model <- confusionMatrix(table(tree.pred, test2$Loan\_Status))$byClass  
dtree\_accuracy <- confusionMatrix(table(tree.pred, test2$Loan\_Status))$overall['Accuracy']  
dtree\_model <- data.frame(dtree\_model)  
dtree\_model <- rbind("Accuracy" = dtree\_accuracy, dtree\_model)

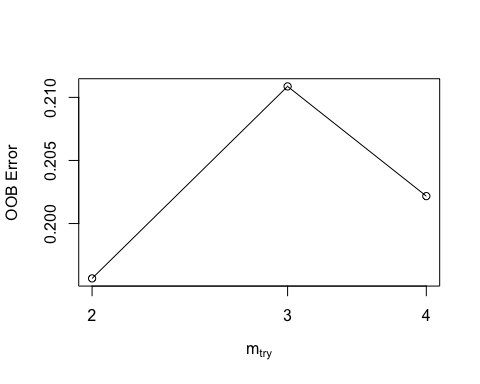
However, more often than not, trees do not give very good prediction errors. Therefore, we will build out a random forest models which tend to outperform trees in terms of prediction and misclassification errors.

### Model4: Random Forest

In creating the best random forest model, we want to minimize the OOB error rate by finding the optimal number of variables selected at each split, known as the mtry. The below code finds the optimal mtry to use in our random forest model.

# Finding best mtry to use in random forest model by evaluating using the lowest OOB error  
mtry <- randomForest::tuneRF(train2[-12],train2$Loan\_Status, ntreeTry=500,  
 stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE)

## mtry = 3 OOB error = 21.09%   
## Searching left ...  
## mtry = 2 OOB error = 19.57%   
## 0.07216495 0.01   
## Searching right ...  
## mtry = 4 OOB error = 20.22%   
## -0.03333333 0.01



best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]  
print(mtry)

## mtry OOBError  
## 2.OOB 2 0.1956522  
## 3.OOB 3 0.2108696  
## 4.OOB 4 0.2021739

print(best.m)

## [1] 2

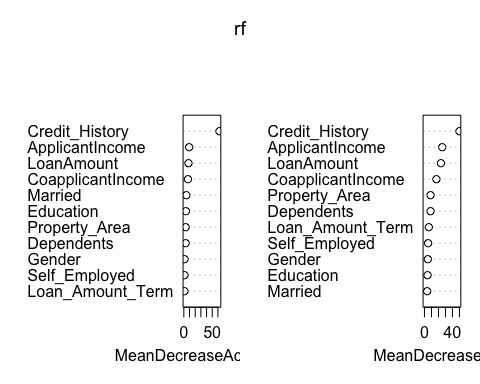
Once the optimal mtry value is found, we apply it to our model.

# Using best mtry in model, plotting importance  
set.seed(71)  
rf <-randomForest(Loan\_Status~.,data=train2, mtry=best.m, importance=TRUE,ntree=500)  
print(rf)

##   
## Call:  
## randomForest(formula = Loan\_Status ~ ., data = train2, mtry = best.m, importance = TRUE, ntree = 500)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 18.91%  
## Confusion matrix:  
## 1 2 class.error  
## 1 62 82 0.56944444  
## 2 5 311 0.01582278

The below graph illustrates the importance of the variables used to predict the Loan Status. The Mean Decrease Accuracy displays how much the model accuracy decreases if we drop the variable. Here, Credit History is regarded as the most important variable by a wide margin. The Mean Decrease Gini graph displays the variable importance on the Gini impurity index used for splitting trees. Again, Credit History is the clear leader but with a narrower gap followed by Loan Amount.

#Evaluate variable importance  
varImpPlot(rf)



The random forest model we end up using has a accuracy of 81.82% on the test dataset. Futhermore, the model has a 97.15 speficity rate and 47.92% sensitivity rate.

rf\_predict <- predict(rf, newdata = test2)  
rf\_conf\_matrix <- confusionMatrix(rf\_predict, test2$Loan\_Status)  
print(rf\_conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2  
## 1 23 6  
## 2 25 100  
##   
## Accuracy : 0.7987   
## 95% CI : (0.7266, 0.8589)  
## No Information Rate : 0.6883   
## P-Value [Acc > NIR] : 0.001482   
##   
## Kappa : 0.4739   
##   
## Mcnemar's Test P-Value : 0.001225   
##   
## Sensitivity : 0.4792   
## Specificity : 0.9434   
## Pos Pred Value : 0.7931   
## Neg Pred Value : 0.8000   
## Prevalence : 0.3117   
## Detection Rate : 0.1494   
## Detection Prevalence : 0.1883   
## Balanced Accuracy : 0.7113   
##   
## 'Positive' Class : 1   
##

#### Model Summary

We record the summary of the Random Forest model metrics in a data frame -

rf\_model <- confusionMatrix(table(rf\_predict, test2$Loan\_Status))$byClass  
rf\_accuracy <- confusionMatrix(table(rf\_predict, test2$Loan\_Status))$overall['Accuracy']  
rf\_model <- data.frame(rf\_model)  
rf\_model <- rbind("Accuracy" = rf\_accuracy, rf\_model)

## Model Performance Comparision

After running various LDA, kNN, decision trees and random forest models, we can take a look at the overall evaluation metrics for these techniques on the loan approval dataset. By creating a dataframe to store all of our metrics, we can visualize the outcomes below:

model\_summary <- data.frame(lda\_model, knn\_model, dtree\_model, rf\_model)  
  
model\_summary %>% kable() %>% kable\_styling(bootstrap\_options = c("striped", "hover", "condensed", "responsive")) %>% scroll\_box(width="100%",height="450px")

lda\_model

knn\_model

dtree\_model

rf\_model

Accuracy

0.8051948

0.7207792

0.7857143

0.7987013

Sensitivity

0.4791667

0.1875000

0.4791667

0.4791667

Specificity

0.9528302

0.9622642

0.9245283

0.9433962

Pos Pred Value

0.8214286

0.6923077

0.7419355

0.7931034

Neg Pred Value

0.8015873

0.7234043

0.7967480

0.8000000

Precision

0.8214286

0.6923077

0.7419355

0.7931034

Recall

0.4791667

0.1875000

0.4791667

0.4791667

F1

0.6052632

0.2950820

0.5822785

0.5974026

Prevalence

0.3116883

0.3116883

0.3116883

0.3116883

Detection Rate

0.1493506

0.0584416

0.1493506

0.1493506

Detection Prevalence

0.1818182

0.0844156

0.2012987

0.1883117

Balanced Accuracy

0.7159984

0.5748821

0.7018475

0.7112814

Consulting the above output table, we observe that LDA Model has the strongest performance for Accuracy, Specificity, Pos Pred Value, Precision and F1 etc.

## Conclusion

Decision tree’s often raise concerns regarding over-fitting, bias and variance error because of their simplicity, and random forests are meant to address these concerns by accounting for a collection of decision trees to come to a single, aggregated result. We found it surprising that the LDA outperformed the random forest model for many metrics (ie. Balanced Accuracy). This may have been because of how we implemented the model or it may have simply been a poor situation for random forests. Also we’re dealing with imbalanced classes (recall 192 N’s, 422 Y’s) this might have had an implication in model performance.