HarvardX: PH125.9x
Data Science - Chose Your Own Project Submission:
Indian Liver Patient Records (ILPR)

Santhosh Channa

6/01/2019

Contents

1 Overview

This report is part of the capstone project of the EdX course 'HarvardX: PH125.9x Data Science: Chose Your Own'. The goal is to demonstrate that the student acquired skills with the R programming language in the field of datascience to actually solve real world problems.

1.1 Introduction

This report is for ILPR - Indian Liver Patienr Rercords.

Patients with Liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. This dataset was used to evaluate prediction algorithms in an effort to reduce burden on doctors.

1.2 Content

This data set contains 416 liver patient records and 167 non liver patient records collected from North East of Andhra Pradesh, India. The "Dataset" column is a class label used to divide groups into liver patient (liver disease) or not (no disease). This data set contains 441 male patient records and 142 female patient records.

Any patient whose age exceeded 89 is listed as being of age "90".

Columns in the dataset:

Column	Description
Age	Age of the patients
Gender	Sex of the patients
Total_Bilirubin	Total Billirubin in mg/dL
Direct_Bilirubin	Conjugated Billirubin in mg/dL
Alkaline_Phosphotase	ALP in IU/L
Alamine_Aminotransferase	ALT in IU/L
Aspartate_Aminotransferase	AST in IU/L
Total_ProtiensTotal	Proteins g/dL
Albumin	Albumin in g/dL
Albumin_and_Globulin_Ratio	A/G ratio
Dataset	(patient has liver disease or not)

Finally, the best resulting model will be used to predict the movie ratings.

1.3 Dataset

This dataset was downloaded from the UCI ML Repository:

Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

```
# Install and load libraries required
if(!require(psych)) install.packages("psych")
if(!require(randomForest)) install.packages("randomForest")
if(!require(caret)) install.packages("caret")
if(!require("pROC")) install.packages("pROC")
if(!require(corrplot)) install.packages("corrplot")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(devtools)) install.packages("devtools")
if(!require(sqldf)) install.packages("sqldf")
library(psych)
library(caret)
library('pROC')
library(tidyverse)
library(caret)
library(randomForest)
library(ggplot2)
library(dplyr)
library(corrplot)
library(rpart)
library(sqldf)
library(devtools)
devtools::install_github("collectivemedia/tictoc")
```

```
library(tictoc)

# Read the dataset
tic("Reading indian_liver_patient.csv data...")
setwd("/Users/schanna/Documents/Me/edX/ILPR-Project/ILPR-Project")

liver_df <- read.csv("data/indian_liver_patient.csv")
toc()
```

```
## Reading indian_liver_patient.csv data...: 0.011 sec elapsed
```

2 Data Analysis

2.1 Overview of dataset

Let us glance through the data set we just created to make sure it matches with the numbers mentioned in the **content** section

```
# Information about data set
class(liver_df)
```

```
## [1] "data.frame"
```

```
glimpse(liver_df)
```

```
## Observations: 583
## Variables: 11
## $ Age                       <int> 65, 62, 62, 58, 72, 46, 26, 29, 17,...
## $ Gender                    <fct> Female, Male, Male, Male, Male, Mal...
## $ Total_Bilirubin           <dbl> 0.7, 10.9, 7.3, 1.0, 3.9, 1.8, 0.9,...
## $ Direct_Bilirubin          <dbl> 0.1, 5.5, 4.1, 0.4, 2.0, 0.7, 0.2, ...
## $ Alkaline_Phosphotase      <int> 187, 699, 490, 182, 195, 208, 154, ...
## $ Alamine_Aminotransferase  <int> 16, 64, 60, 14, 27, 19, 16, 14, 22,...
## $ Aspartate_Aminotransferase <int> 18, 100, 68, 20, 59, 14, 12, 11, 19...
## $ Total_Protiens            <dbl> 6.8, 7.5, 7.0, 6.8, 7.3, 7.6, 7.0, ...
## $ Albumin                   <dbl> 3.3, 3.2, 3.3, 3.4, 2.4, 4.4, 3.5, ...
## $ Albumin_and_Globulin_Ratio <dbl> 0.90, 0.74, 0.89, 1.00, 0.40, 1.30,...
## $ Dataset                   <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,...
```

Summary information of edx dataset:

```
summary(liver_df)
```

```
##       Age           Gender    Total_Bilirubin  Direct_Bilirubin
##  Min.   : 4.00   Female:142   Min.   : 0.400   Min.   : 0.100
##  1st Qu.:33.00   Male  :441   1st Qu.: 0.800   1st Qu.: 0.200
##  Median :45.00                Median : 1.000   Median : 0.300
##  Mean   :44.75                Mean   : 3.299   Mean   : 1.486
##  3rd Qu.:58.00                3rd Qu.: 2.600   3rd Qu.: 1.300
##  Max.   :90.00                Max.   :75.000   Max.   :19.700
##
##  Alkaline_Phosphotase Alamine_Aminotransferase Aspartate_Aminotransferase
##  Min.   :  63.0       Min.   :  10.00          Min.   :  10.0
##  1st Qu.: 175.5       1st Qu.:  23.00          1st Qu.:  25.0
##  Median : 208.0       Median :  35.00          Median :  42.0
##  Mean   : 290.6       Mean   :  80.71          Mean   : 109.9
```

```
##  3rd Qu.: 298.0       3rd Qu.:  60.50          3rd Qu.:  87.0
##  Max.   :2110.0       Max.   :2000.00          Max.   :4929.0
##
##  Total_Protiens     Albumin      Albumin_and_Globulin_Ratio
##  Min.   :2.700   Min.   :0.900   Min.   :0.3000
##  1st Qu.:5.800   1st Qu.:2.600   1st Qu.:0.7000
##  Median :6.600   Median :3.100   Median :0.9300
##  Mean   :6.483   Mean   :3.142   Mean   :0.9471
##  3rd Qu.:7.200   3rd Qu.:3.800   3rd Qu.:1.1000
##  Max.   :9.600   Max.   :5.500   Max.   :2.8000
##                                  NA's   :4
##     Dataset
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.286
##  3rd Qu.:2.000
##  Max.   :2.000
##
```

2.2 Understanding the given dataset and modyfing where necessary

Rename the column "Dataset" to "Diseased" for better representation

```
colnames(liver_df)[colnames(liver_df)=='Dataset'] = 'Diseased'
colnames(liver_df)
```

```
##  [1] "Age"                         "Gender"
##  [3] "Total_Bilirubin"             "Direct_Bilirubin"
##  [5] "Alkaline_Phosphotase"        "Alamine_Aminotransferase"
##  [7] "Aspartate_Aminotransferase" "Total_Protiens"
##  [9] "Albumin"                     "Albumin_and_Globulin_Ratio"
## [11] "Diseased"
```

Showing Gender ratio in the data set:

```
tic("Gender Ratio")
table(liver_df$Gender)
```

```
##
## Female   Male
##    142    441
```

```
toc()
```

```
## Gender Ratio: 0.001 sec elapsed
```

Change values of Diseased to "0" and "1" from "1" and "2":

```
table(liver_df$Diseased)
```

```
##
##   1   2
## 416 167
```

```
liver_df$Diseased <- as.numeric(ifelse(liver_df$Diseased == 2, 1, 0))
table(liver_df$Diseased)
```

```
##
```

```
##   0   1
## 416 167
```

```
# Convert Diseased to a factor variable
class(liver_df$Diseased)
```

```
## [1] "numeric"
```

```
liver_df$Diseased <- as.factor(liver_df$Diseased)
class(liver_df$Diseased)
```

```
## [1] "factor"
```

Remove records with NA values in column(s):

```
print("Dimensions of dataset before removing NA's")
```

```
## [1] "Dimensions of dataset before removing NA's"
```

```
dim(liver_df)
```

```
## [1] 583  11
```

```
tic("Remove records with NA values in column(s)")
colSums(sapply(liver_df, is.na))
```

```
##                         Age                      Gender
##                           0                           0
##             Total_Bilirubin            Direct_Bilirubin
##                           0                           0
##       Alkaline_Phosphotase   Alamine_Aminotransferase
##                           0                           0
## Aspartate_Aminotransferase             Total_Protiens
##                           0                           0
##                     Albumin Albumin_and_Globulin_Ratio
##                           0                           4
##                    Diseased
##                           0
```

```
liver_df = na.omit(liver_df)
print("Dimensions of dataset after removing NA's")
```

```
## [1] "Dimensions of dataset after removing NA's"
```

```
dim(liver_df)
```

```
## [1] 579  11
```

```
toc()
```

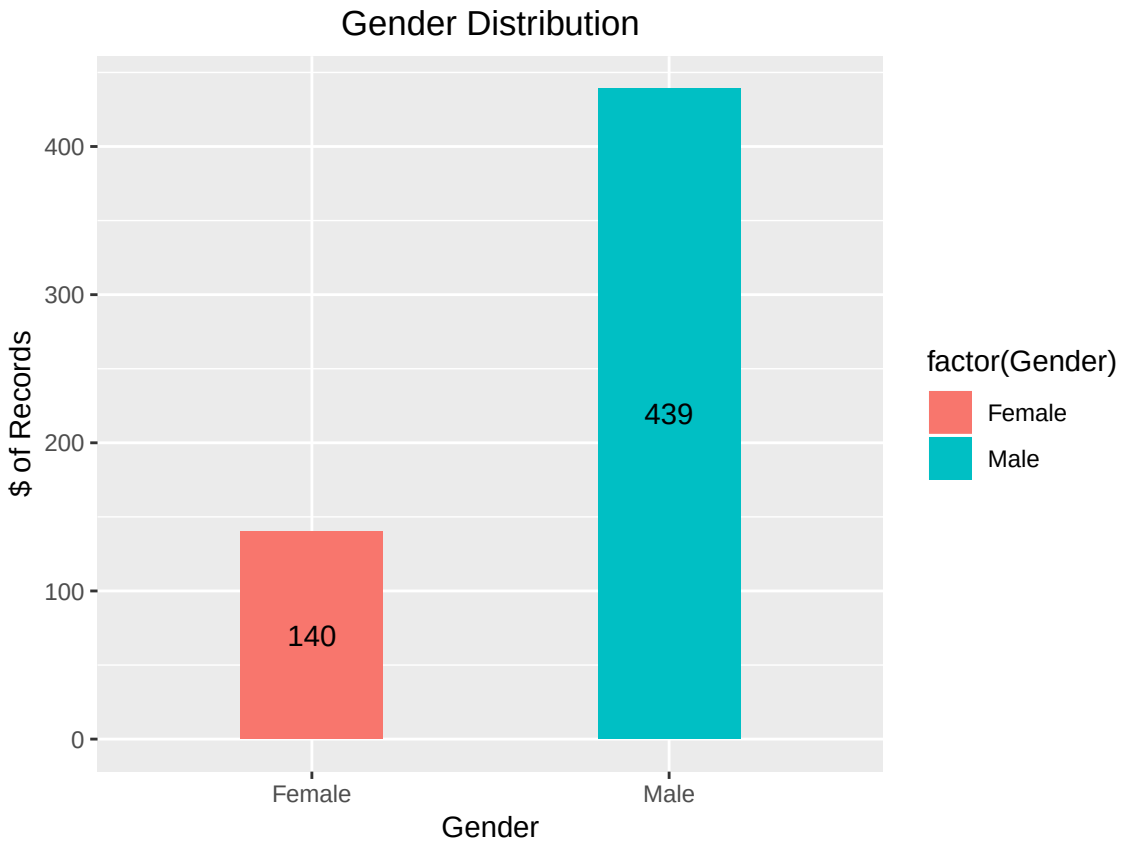
```
## Remove records with NA values in column(s): 0.005 sec elapsed
```

2.3 Visualyzing the data

Gender Distribution . . .

```
ggplot(liver_df, aes(factor(Gender))) +
geom_bar(width = 0.4, aes(fill=factor(Gender))) +
geom_text(stat = "count", check_overlap = TRUE, aes( label = ..count..),
          position = position_stack(vjust = 0.5)) +
```

```
labs(x = "Gender", y = "$ of Records", title = "Gender Distribution") +
theme(plot.title = element_text(hjust = 0.5))
```



Diseased distribution among Gender . . .

```
print("Using sqldf")
```

```
## [1] "Using sqldf"
```

```
sqldf("select count(1), Gender, Diseased from liver_df
       group by Gender, Diseased order by 2")
```

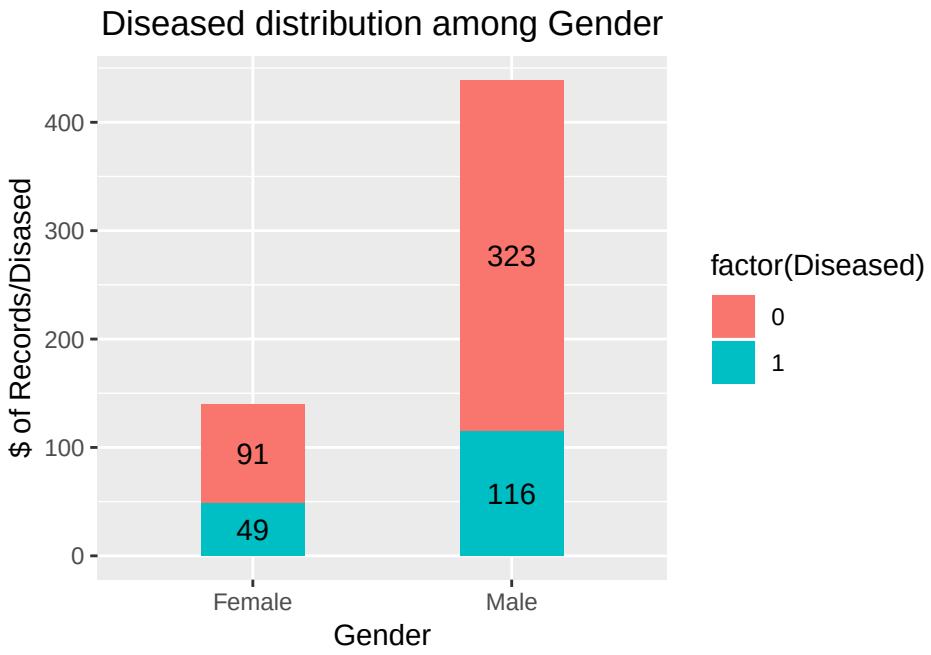
```
##   count(1) Gender Diseased
## 1       91 Female        0
## 2       49 Female        1
## 3      323   Male        0
## 4      116   Male        1
```

```
table(liver_df$Diseased, liver_df$Gender)
```

```
##
##     Female Male
##   0     91  323
##   1     49  116
```

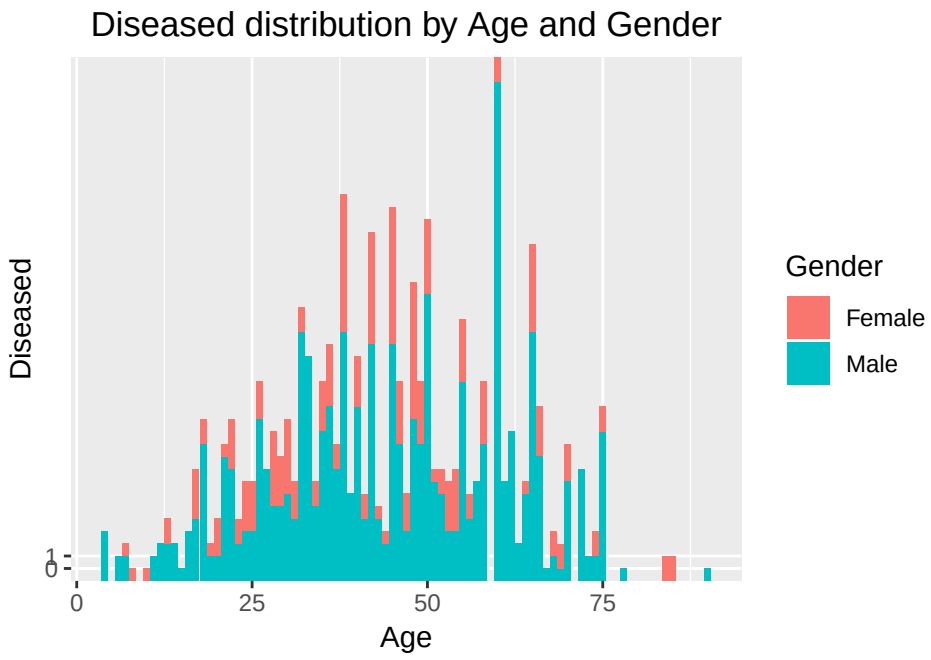
```
ggplot(liver_df, aes(factor(Gender), fill=factor(Diseased))) +
geom_bar(width = 0.4) +
geom_text(aes(label = ..count..), stat="count",
position = position_stack(0.5)) +
labs(x = "Gender", y = "$ of Records/Disased",
```

```
    title = "Diseased distribution among Gender") +
theme(plot.title = element_text(hjust = 0.5))
```



Plot by Age and Gender ...

```
ggplot(data = liver_df, aes(Age, Diseased)) +
geom_bar(stat = "identity", aes(fill = Gender)) +
ggtitle("Diseased distribution by Age and Gender") +
theme(plot.title = element_text(hjust = 0.5))
```



The graphs below will help us to understand the distribution of data points within each variable forming solid foundation for the analysis to come. In the graphs we can see how the data is extremely left or right skewed.

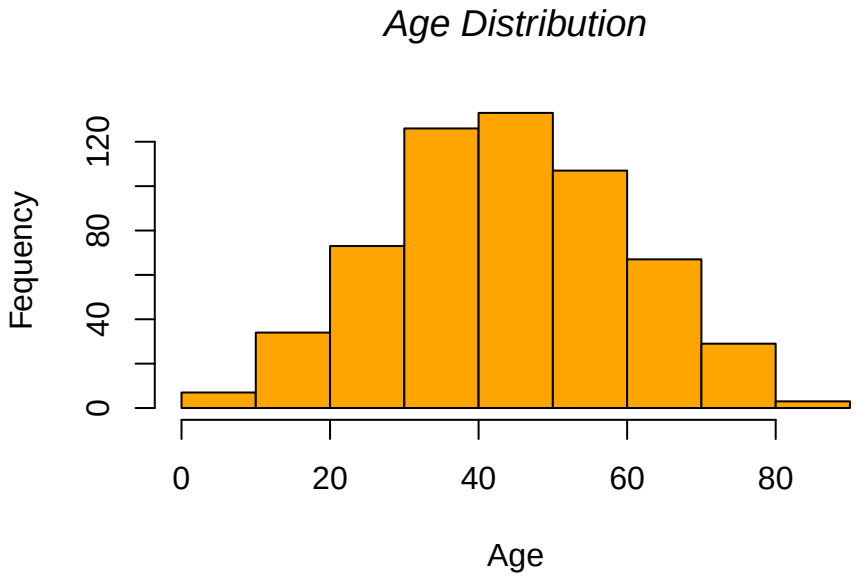
This aided in centering the data better, but still they are not completely normal, so it is important to have realistic expectations about the linear predictive power of individual variables within the model as a whole.

Histogram of Age Distribution :

```
describe(liver_df$Age)
```

```
##    vars   n  mean    sd median trimmed   mad min max range  skew kurtosis
## X1    1 579 44.78 16.22     45   44.89 17.79   4  90    86 -0.03    -0.58
##      se
## X1 0.67
```

```
hist(liver_df$Age, main = "Age Distribution", font.main = 3, col = "orange",
     xlab = "Age", ylab = "Fequency")
```

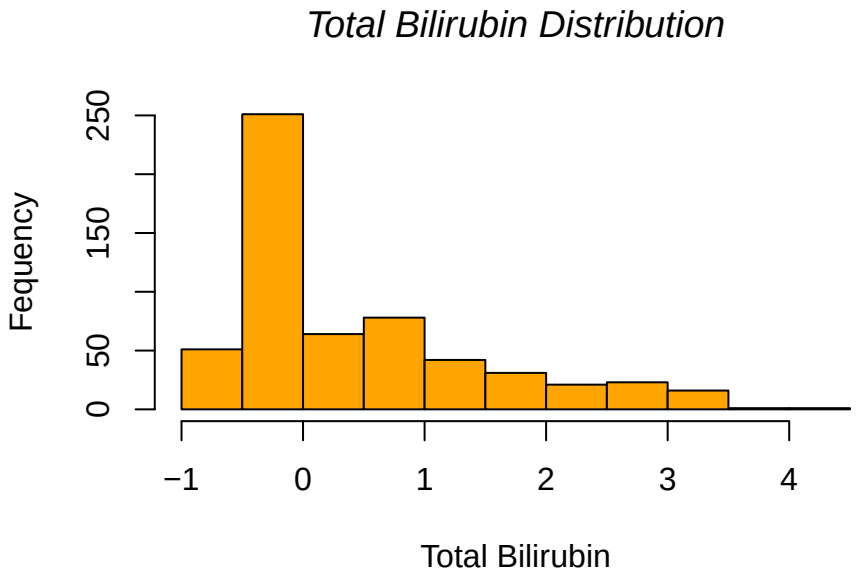


Total Bilirubin

```
describe(liver_df$Total_Bilirubin)
```

```
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 579 3.32 6.23      1    1.76 0.59 0.4  75  74.6 4.87    36.46 0.26
```

```
hist(log(liver_df$Total_Bilirubin),
     main = "Total Bilirubin Distribution", font.main = 3, col = "orange",
     xlab = "Total Bilirubin", ylab = "Fequency")
```

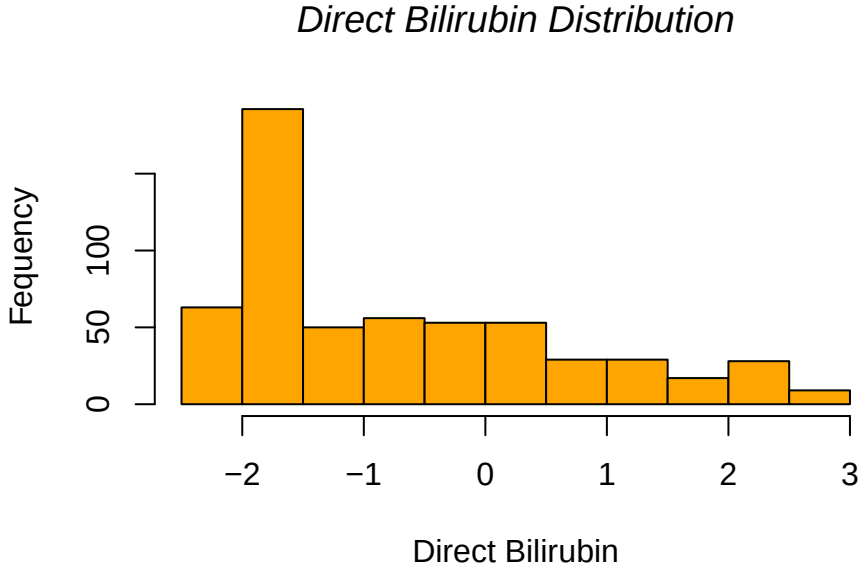


Direct Bilirubin

```
describe(liver_df$Direct_Bilirubin)
```

```
##    vars   n mean   sd median trimmed mad min  max range skew kurtosis   se
## X1    1 579 1.49 2.82    0.3    0.75 0.3 0.1 19.7  19.6 3.18     11.1 0.12
```

```
hist(log(liver_df$Direct_Bilirubin),
     main = "Direct Bilirubin Distribution", font.main = 3, col = "orange",
     xlab = "Direct Bilirubin", ylab = "Fequency")
```

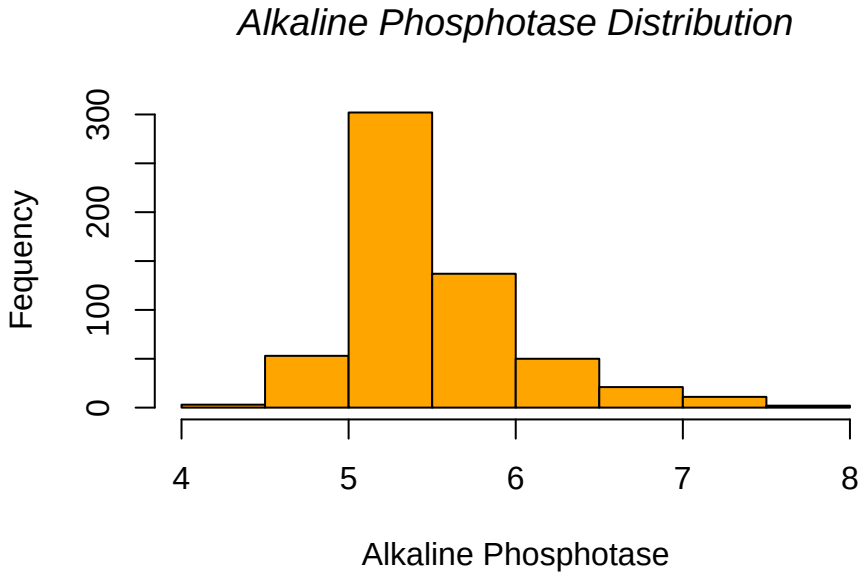


Alkaline Phosphotase

```
describe(liver_df$Alkaline_Phosphotase)
```

```
##    vars   n   mean     sd median trimmed   mad min  max range skew
## X1    1 579 291.37 243.56    208  239.24 74.13  63 2110  2047 3.73
##    kurtosis    se
## X1     17.4 10.12
```

```r
hist(log(liver_df$Alkaline_Phosphotase),
     main = "Alkaline Phosphotase Distribution", font.main = 3, col = "orange",
     xlab = "Alkaline Phosphotase", ylab = "Fequency")
```

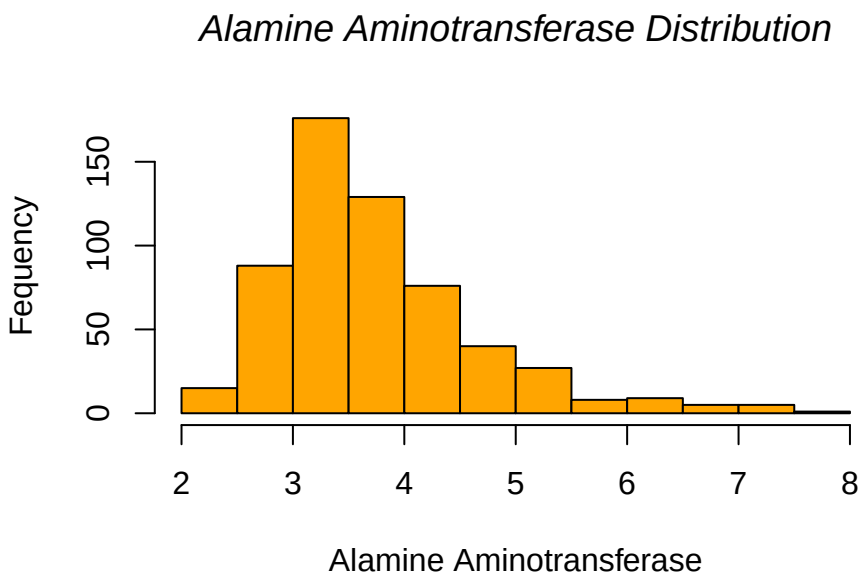


Alamine Aminotransferase

```r
describe(liver_df$Alamine_Aminotransferase)
```

```
##    vars   n  mean     sd median trimmed   mad min  max range skew kurtosis
## X1    1 579 81.13 183.18     35   44.25 22.24  10 2000  1990 6.49    49.61
##      se
## X1 7.61
```

```r
hist(log(liver_df$Alamine_Aminotransferase),
     main = "Alamine Aminotransferase Distribution", font.main = 3, col = "orange",
     xlab = "Alamine Aminotransferase", ylab = "Fequency")
```

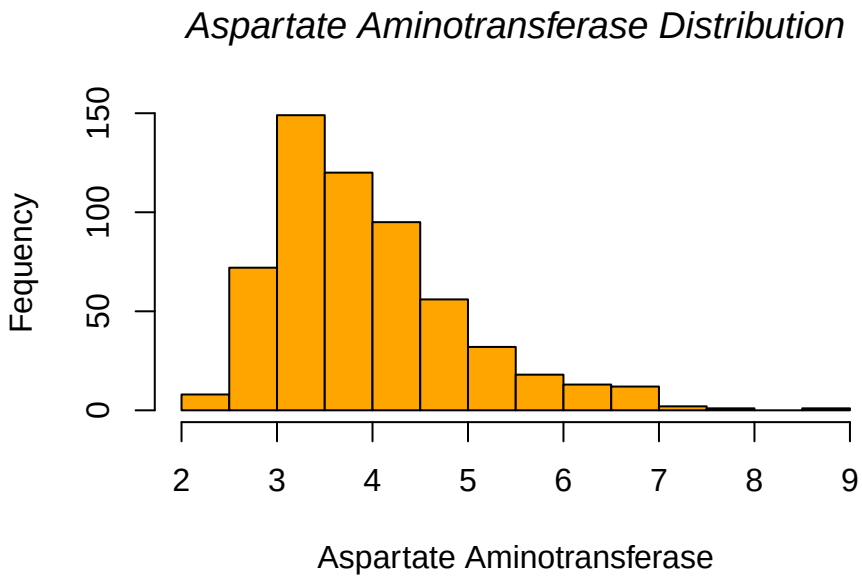


Aspartate Aminotransferase

```r
describe(liver_df$Aspartate_Aminotransferase)
```

```
##    vars   n   mean     sd median trimmed   mad min  max range  skew
## X1    1 579 110.41 289.85     42   57.18 31.13  10 4929  4919 10.46
##    kurtosis    se
## X1   148.11 12.05
```

```r
hist(log(liver_df$Aspartate_Aminotransferase),
     main = "Aspartate Aminotransferase Distribution", font.main = 3, col = "orange",
     xlab = "Aspartate Aminotransferase", ylab = "Fequency")
```

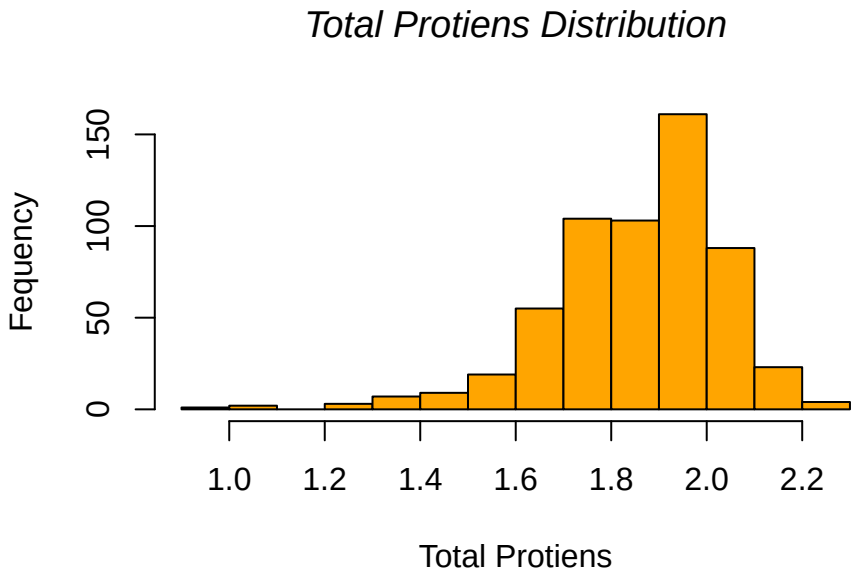


Total Protiens

```r
describe(liver_df$Total_Protiens)
```

```
##    vars   n mean   sd median trimmed  mad min max range  skew kurtosis
## X1    1 579 6.48 1.08    6.6    6.51 1.04 2.7 9.6   6.9 -0.29     0.22
##      se
## X1 0.05
```

```r
hist(log(liver_df$Total_Protiens), main = "Total Protiens Distribution",
     font.main = 3, col = "orange", xlab = "Total Protiens", ylab = "Fequency")
```

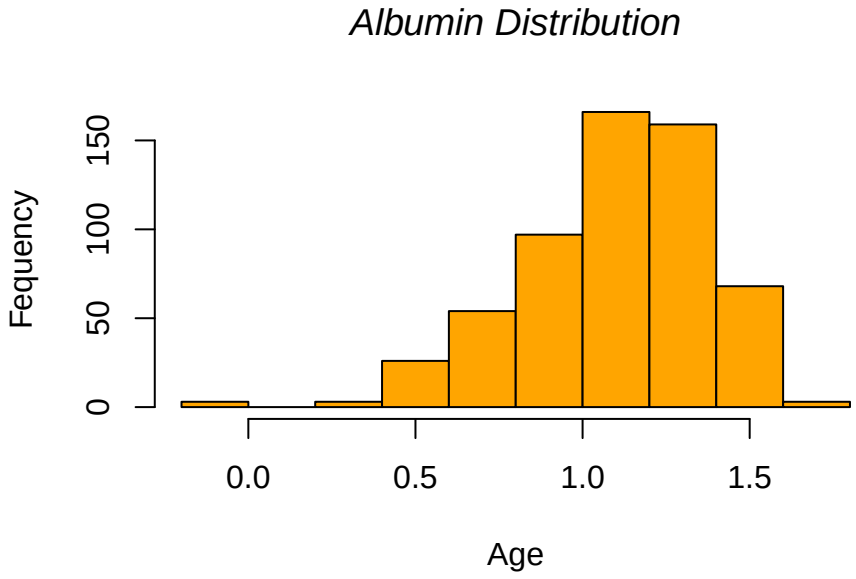


Albumin

```
describe(liver_df$Albumin)
```

```
##    vars   n mean   sd median trimmed  mad min max range  skew kurtosis
## X1    1 579 3.14 0.79    3.1    3.15 0.89 0.9 5.5   4.6 -0.05    -0.41
##      se
## X1 0.03
```

```
hist(log(liver_df$Albumin), main = "Albumin Distribution", font.main = 3,
     col = "orange", xlab = "Age", ylab = "Fequency")
```

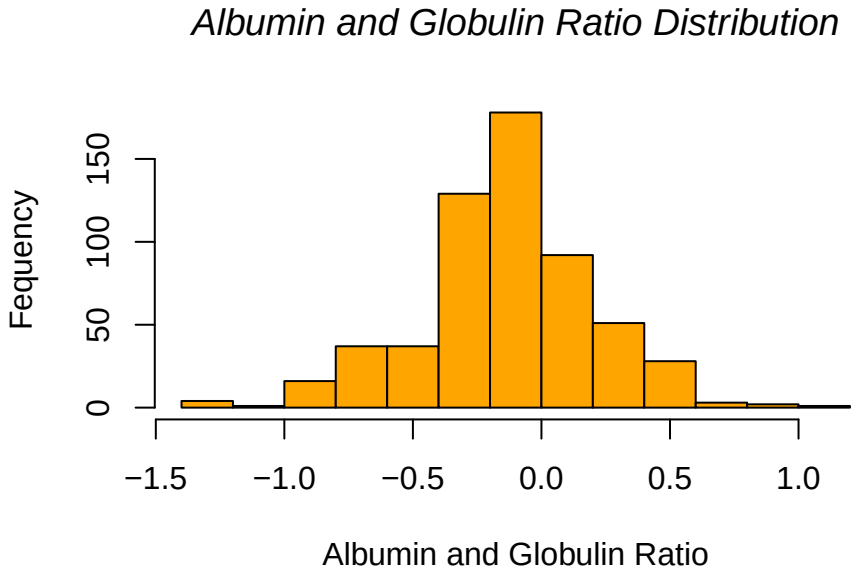


Albumin and Globulin Ratio

```
describe(liver_df$Albumin_and_Globulin_Ratio)
```

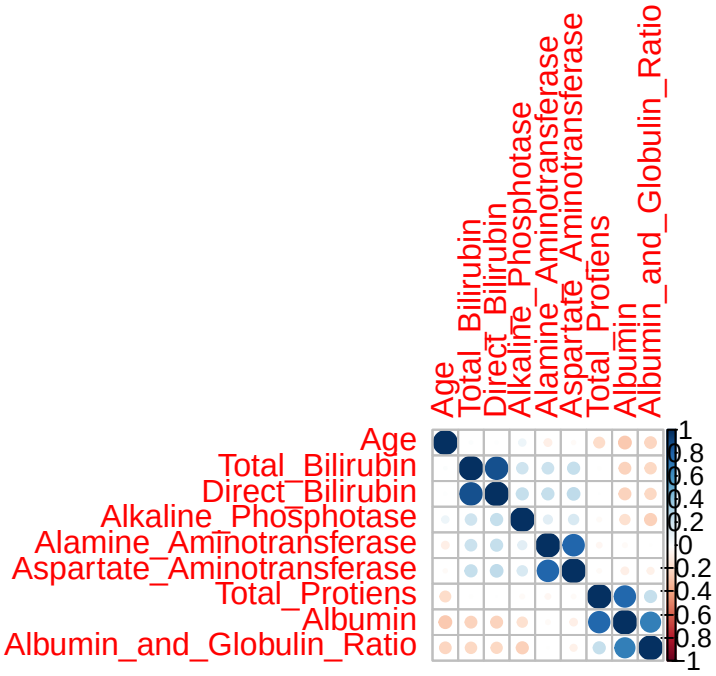
```
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 579 0.95 0.32   0.93    0.93 0.25 0.3 2.8   2.5 0.99     3.22 0.01
```

```r
hist(log(liver_df$Albumin_and_Globulin_Ratio),
     main = "Albumin and Globulin Ratio Distribution", font.main = 3,
     col = "orange", xlab = "Albumin and Globulin Ratio", ylab = "Fequency")
```



Creating Correlation Matrix

```r
non_pred_cols <- c('Gender', 'Diseased')
correlationMatrix <- cor(liver_df[, !(names(liver_df) %in% non_pred_cols)])
corrplot(correlationMatrix)
```



2.4 Modelling Approach

2.4.1 Feature Selection

Automatic feature selection methods can be used to build many models with different subsets of a dataset and identify those attributes that are and are not required to build an accurate model.

A popular automatic method for feature selection provided by the caret R package is called Recursive Feature Elimination or RFE.

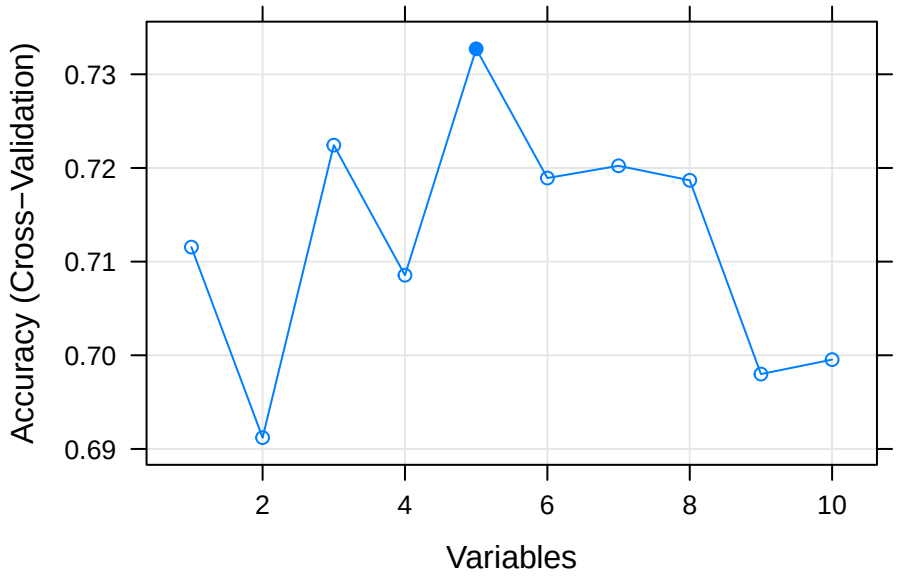
We use a RFE (Recursive Feature Elimination) to remove the unnecessary features.

```
set.seed(7)
# define the control using a random forest selection function
rfeControl <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results <- rfe(liver_df[,1:10], liver_df[,11],sizes = c(1:10), rfeControl=rfeControl)
#print(results)

# list the chosen features
predictors <- predictors(results)
print(predictors)
```

```
## [1] "Direct_Bilirubin"           "Alamine_Aminotransferase"
## [3] "Total_Bilirubin"            "Aspartate_Aminotransferase"
## [5] "Age"
```

```
# plot the results
plot(results, type=c("g", "o"))
```




```
# Resize the original dataset by removing others which are not listed from predictors

subset_liver_df <- liver_df[,c(predictors,"Diseased")]
colnames(subset_liver_df)
```

```
## [1] "Direct_Bilirubin"           "Alamine_Aminotransferase"
## [3] "Total_Bilirubin"            "Aspartate_Aminotransferase"
## [5] "Age"                        "Diseased"
```

We create a sample data using a random sample, the seed has to be set to generate same indices everytime.

Below we will generate our train and test data set samples using a 70/30 split.

```
smp_size <- floor(0.7 * nrow(subset_liver_df))
set.seed(9)
train_ind <- sample(seq_len(nrow(subset_liver_df)), size = smp_size)
train <- subset_liver_df[train_ind, ]
print("Train Data set Dimensions:")
```

```
## [1] "Train Data set Dimensions:"
```

```
dim(train)
```

```
## [1] 405   6
```

```
table(train$Diseased)
```

```
##
##   0   1
## 292 113
```

```
test = subset_liver_df[-train_ind, ]
print("Test Data set Dimensions:")
```

```
## [1] "Test Data set Dimensions:"
```

```
dim(test)
```

```
## [1] 174   6
```

```
table(test$Diseased)
```

```
##
##   0   1
## 122  52
```

2.4.2 Model#1: GLM - Generalized Liner Model

```
tic("GLM Model...")
fit_glm <- suppressWarnings(glm(Diseased ~ ., data = train, family = "binomial"))
p_hat_glm <- predict(fit_glm, test)
y_hat_glm <- factor(ifelse(p_hat_glm > 0.5, 1, 0))
glm_acc <-
suppressWarnings(confusionMatrix(data = y_hat_glm, reference = test$Diseased)$overall["Accuracy"])
print(glm_acc)
```

```
##  Accuracy
## 0.7011494
```

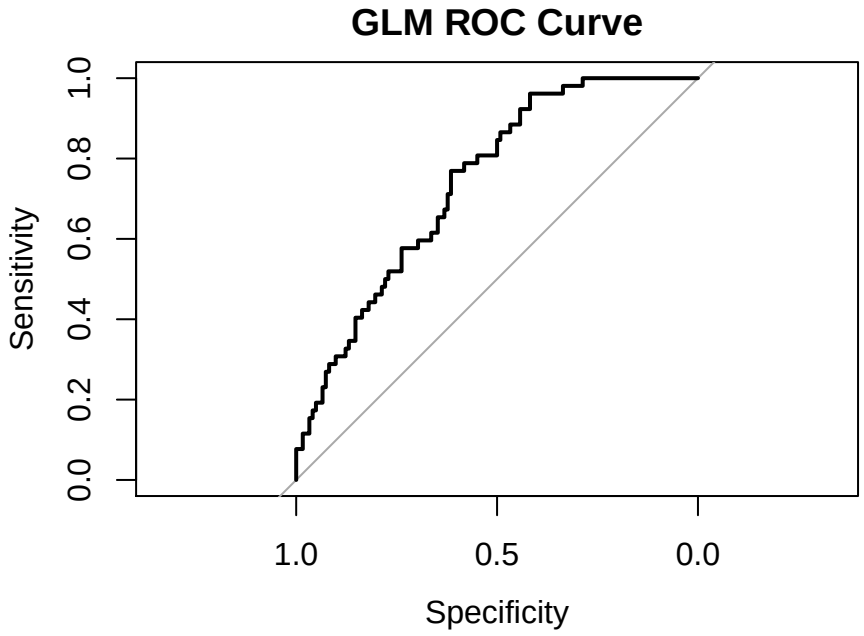
```
cat("\n")
```

```
# Persist the accuracy in accuracy_results
accuracy_results <-
  suppressWarnings(
    data_frame(
      Model = "Model#1: GLM(Generalized Liner Model) Predictions",
      Accuracy = glm_acc))
toc()
```

```
## GLM Model...: 0.024 sec elapsed
```

ROC Curve for the GLM Model

```
glm_pred_prob <- as.data.frame(predict(fit_glm, test, type = "response"))
plot(roc(test$Diseased,
         glm_pred_prob$`predict(fit_glm, test, type = "response")`),
     main = "GLM ROC Curve")
```



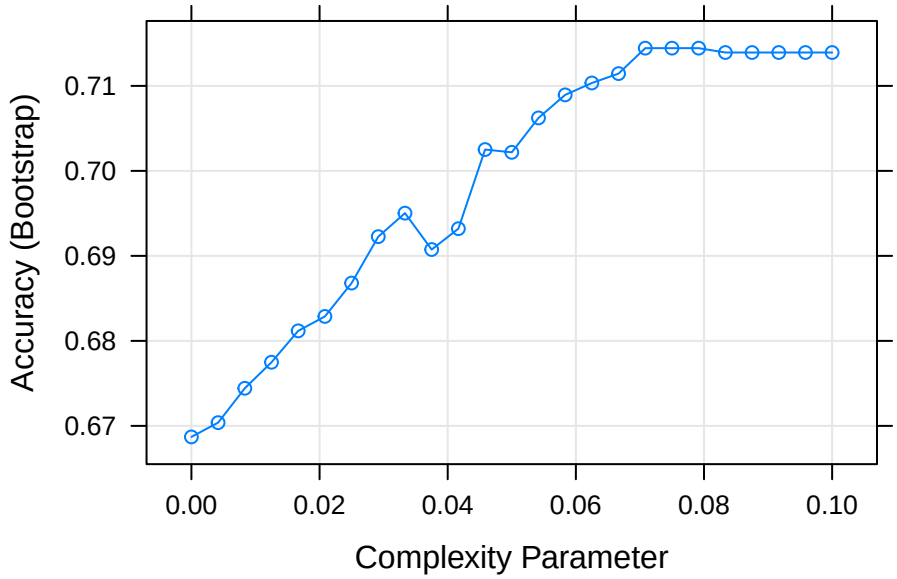

```
print(roc(test$Diseased, glm_pred_prob$`predict(fit_glm, test, type = "response")`))
```

```
##
## Call:
## roc.default(response = test$Diseased, predictor = glm_pred_prob$`predict(fit_glm, test, type = "respo
##
## Data: glm_pred_prob$`predict(fit_glm, test, type = "response")` in 122 controls (test$Diseased 0) < 5
## Area under the curve: 0.7399
```

2.4.3 Model#2 RPART

```
tic("RPART Model...")
xfit <- rpart(Diseased ~., data = subset_liver_df)
plot(xfit, margin = 0.1)
text(xfit, cex = 0.75)
```


```
train_rpart <- train(Diseased ~ ., method = "rpart",
                     tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)), data = train)
plot(train_rpart)
```




```
rpart_acc <- confusionMatrix(predict(train_rpart, test),
                             test$Diseased)$overall["Accuracy"]
print(rpart_acc)
```

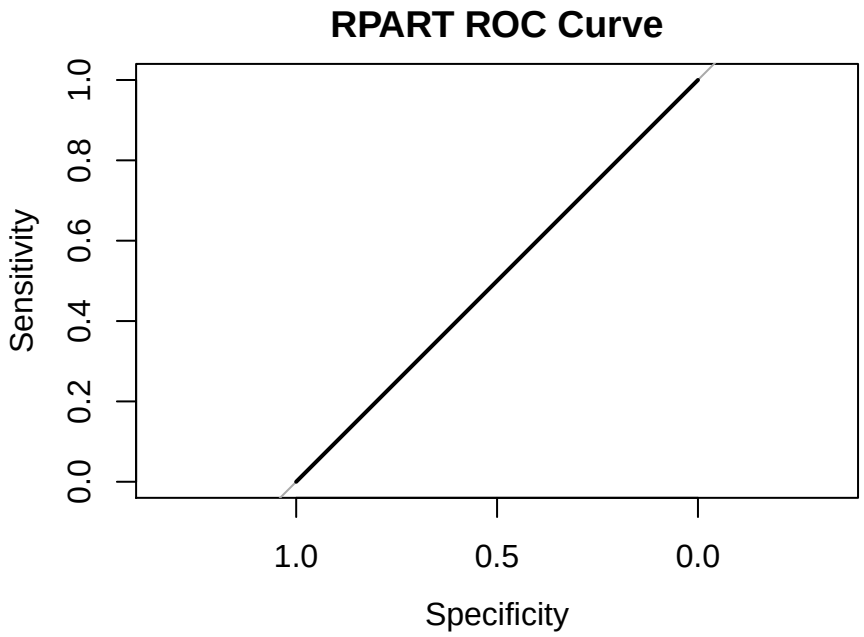
```
##  Accuracy
## 0.7011494
```

```
# Persist prediction results
accuracy_results <- bind_rows(accuracy_results,
  data_frame(Model = "Model#2: RPART Predictions",
  Accuracy = rpart_acc))
toc()
```

```
## RPART Model...: 2.506 sec elapsed
```

ROC Curve for RPART

```
rpart_pred_prob <- as.data.frame(predict(train_rpart, test,
                                         type = "prob"))
plot(roc(test$Diseased, rpart_pred_prob$`0`), main = "RPART ROC Curve")
```




```
print(roc(test$Diseased, rpart_pred_prob$`0`))
```

```
##
## Call:
## roc.default(response = test$Diseased, predictor = rpart_pred_prob$`0`)
##
## Data: rpart_pred_prob$`0` in 122 controls (test$Diseased 0) < 52 cases (test$Diseased 1).
## Area under the curve: 0.5
```

2.4.4 Model#3 Random Forest default

Random Forest can be tune up using Random Search or Grid Search. We use the Grid Search to try various combinations of parameters using Cross-Validation.

```
# First we create a K-fold cross-validation of 10 folds:
tic("Random Forest Default Model...")
trControl <- trainControl(method = "cv",
    number = 10,
    search = "grid")

# Then we run the model for the default (or not optimized) Random Forest version:

set.seed(1234)
rf_default <- train(Diseased~.,
    data = train,
    method = "rf",
    metric = "Accuracy",
    trControl = trControl)
print(rf_default)
```

```
## Random Forest
```

```
##
## 405 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 365, 365, 364, 365, 363, 365, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7086818  0.2151984
##   3     0.6986789  0.1809004
##   5     0.7013008  0.2016942
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
# Evaluate the model
default_prediction <- predict(rf_default,test)
default_acc <- confusionMatrix(default_prediction, test$Diseased)$overall["Accuracy"]
print(default_acc)
```

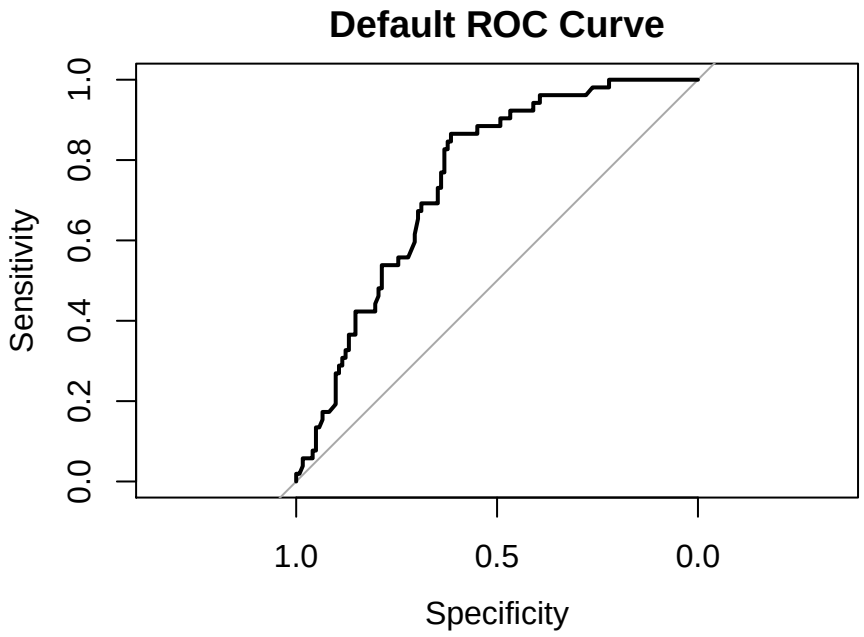
```
##  Accuracy
## 0.7241379
```

```
# Persist prediction results
accuracy_results <- bind_rows(accuracy_results,
  data_frame(Model = "Model#3: Default Predictions",
  Accuracy = default_acc))
toc()
```

```
## Random Forest Default Model...: 4.317 sec elapsed
```

ROC Curve for Random Forest Default Model

```
default_pred_prob <- as.data.frame(predict(rf_default, test, type = "prob"))
plot(roc(test$Diseased, default_pred_prob$`0`), main = "Default ROC Curve")
```




```
print(roc(test$Diseased, default_pred_prob$`0`))
```

```
##
## Call:
## roc.default(response = test$Diseased, predictor = default_pred_prob$`0`)
##
## Data: default_pred_prob$`0` in 122 controls (test$Diseased 0) > 52 cases (test$Diseased 1).
## Area under the curve: 0.751
```

2.4.5 Model#4 Hyperparameter Optimization -Best MTRY

```
tic("MTRY Model...")
set.seed(1234)
tuneGrid <- expand.grid(.mtry = c(3: 10))
rf_mtry <- suppressWarnings(train(Diseased~.,
    data = train,
    method = "rf",
    metric = "Accuracy",
    tuneGrid = tuneGrid,
    trControl = trControl,
    importance = TRUE,
    nodesize = 14,
    ntree = 300))
print(rf_mtry)
```

```
## Random Forest
##
## 405 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 365, 365, 364, 365, 363, 365, ...
```

```
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    3    0.6989866  0.1625573
##    4    0.7013647  0.1799606
##    5    0.6963676  0.1627272
##    6    0.7014257  0.1784482
##    7    0.6989257  0.1757207
##    8    0.6989866  0.1712151
##    9    0.6915447  0.1666312
##   10    0.7036266  0.1935518
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 10.
```

```
best_mtry <- rf_mtry$bestTune$mtry
max(rf_mtry$results$Accuracy)
```

```
## [1] 0.7036266
```

```
# Evaluate the model

mtry_prediction <- predict(rf_mtry, test)
mtry_acc <- confusionMatrix(mtry_prediction, test$Diseased)$overall["Accuracy"]
print(mtry_acc)
```

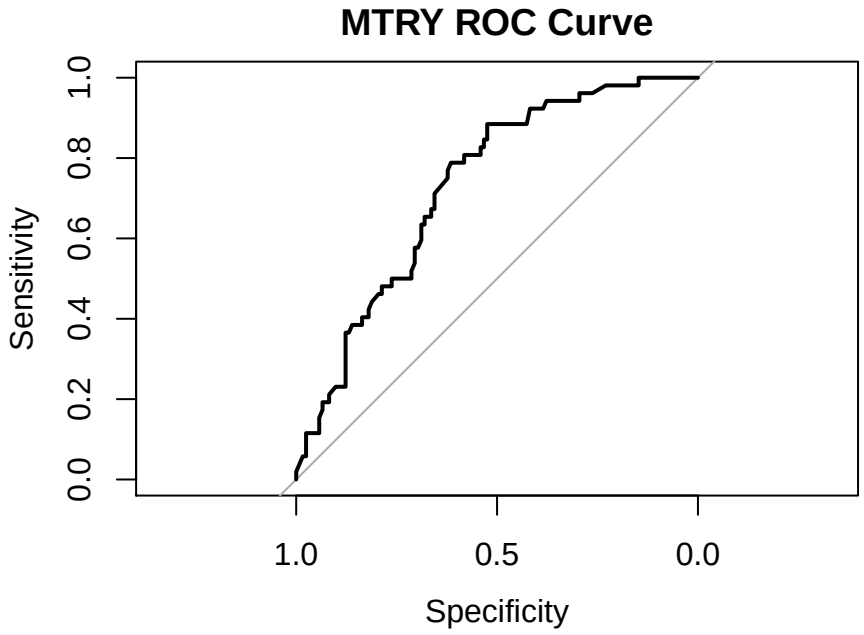
```
##  Accuracy
## 0.7011494
```

```
# Append to accuracy_results table
accuracy_results <- bind_rows(accuracy_results,
  data_frame(Model = "Model#4: Default + MTRY Predictions",
             Accuracy = mtry_acc))
toc()
```

```
## MTRY Model...: 8.604 sec elapsed
```

ROC Curve for MTRY

```
mtry_pred_prob <- as.data.frame(predict(rf_mtry, test, type = "prob"))
plot(roc(test$Diseased, mtry_pred_prob$`0`), main = "MTRY ROC Curve")
```




```
print(roc(test$Diseased, mtry_pred_prob$`0`))
```

```
##
## Call:
## roc.default(response = test$Diseased, predictor = mtry_pred_prob$`0`)
##
## Data: mtry_pred_prob$`0` in 122 controls (test$Diseased 0) > 52 cases (test$Diseased 1).
## Area under the curve: 0.7312
```

2.4.6 Model#5 Best maxnodes

This parameter sets the maximum of the terminal nodes in the forest For this parameter will need to create a list of values and then summarize the results:

```
tic("Maxnodes Model...")
#List to store the values
store_maxnode <- list()
#test only with the best mtry parameter already obtained
tuneGrid <- expand.grid(.mtry = best_mtry)
#iterate over different values
suppressWarnings(for (maxnodes in c(20: 30)) {
    set.seed(1234)
    rf_maxnode <- train(Diseased~.,
        data = train,
        method = "rf",
        metric = "Accuracy",
        tuneGrid = tuneGrid,
        trControl = trControl,
        importance = TRUE,
        nodesize = 14,
        maxnodes = maxnodes,
        ntree = 300)
    current_iteration <- toString(maxnodes)
    store_maxnode[[current_iteration]] <- rf_maxnode
})
```

```r
results_maxnode <-  resamples(store_maxnode)
summary(results_maxnode)
```

```
##
## Call:
## summary.resamples(object = results_maxnode)
##
## Models: 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
## Number of resamples: 10
##
## Accuracy
##         Min.   1st Qu.    Median      Mean   3rd Qu.  Max. NA's
## 20 0.6000000 0.6562500 0.6867015 0.6939257 0.7483232 0.775    0
## 21 0.6000000 0.6562500 0.6914634 0.6937485 0.7364983 0.775    0
## 22 0.6097561 0.6562500 0.6986063 0.6988676 0.7300305 0.800    0
## 23 0.6000000 0.6394817 0.6952381 0.6914866 0.7300305 0.775    0
## 24 0.6000000 0.6689024 0.7108014 0.7011847 0.7300305 0.775    0
## 25 0.6097561 0.6750000 0.6986063 0.7013676 0.7300305 0.800    0
## 26 0.5853659 0.6750000 0.6988966 0.7014257 0.7483232 0.800    0
## 27 0.6097561 0.6750000 0.6986063 0.7038676 0.7300305 0.800    0
## 28 0.6341463 0.6601190 0.7073171 0.7014257 0.7250000 0.775    0
## 29 0.5853659 0.6375000 0.6988966 0.6939866 0.7300305 0.800    0
## 30 0.5853659 0.6375000 0.6988966 0.6865476 0.7205793 0.775    0
##
## Kappa
##           Min.      1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 20 -0.13074205 -0.009068573 0.1272257 0.1260303 0.2843001 0.3430657    0
## 21 -0.06312292 -0.013599524 0.1834828 0.1426194 0.2519549 0.3430657    0
## 22 -0.05660377  0.006849315 0.1957463 0.1602363 0.2580432 0.4346290    0
## 23 -0.09489051 -0.020794764 0.1834828 0.1383480 0.2580432 0.3430657    0
## 24 -0.09489051  0.089127814 0.1992594 0.1601280 0.2605987 0.3430657    0
## 25 -0.01562500  0.051590809 0.1957463 0.1771509 0.2580432 0.3962264    0
## 26 -0.08736349  0.008483270 0.1780822 0.1668278 0.2965226 0.4346290    0
## 27 -0.01562500  0.051590809 0.1957463 0.1896881 0.2580432 0.4684385    0
## 28 -0.05660377  0.101256458 0.1843987 0.1791502 0.2542420 0.3430657    0
## 29 -0.09489051  0.044395569 0.1780822 0.1653450 0.2852940 0.4684385    0
## 30 -0.09489051 -0.034568470 0.1262626 0.1288243 0.2542420 0.3835616    0
```

```r
# Evaluate the model
maxnode_prediction <- predict(rf_maxnode,test)

maxnode_acc <- confusionMatrix(maxnode_prediction, test$Diseased)$overall["Accuracy"]
print(maxnode_acc)
```

```
##  Accuracy
## 0.7126437
```

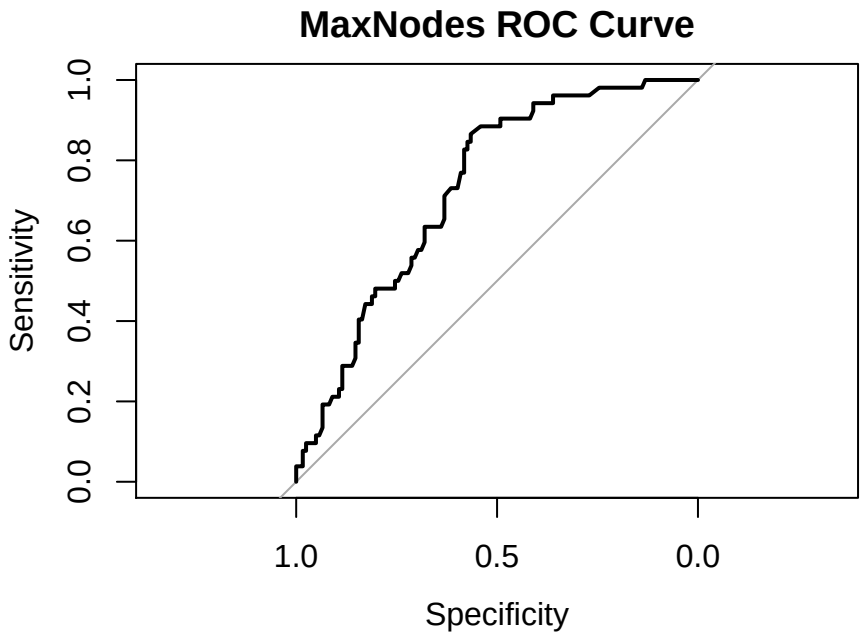
```r
# Append to accuracy_results table
accuracy_results <- bind_rows(accuracy_results,
  data_frame(Model = "Model#5: Default + MTRY + MaxNodes Predictions",
             Accuracy = maxnode_acc))

toc()
```

```
## Maxnodes Model...: 15.182 sec elapsed
```

ROC Curve for MaxNodes

```
maxnode_pred_prob <- as.data.frame(predict(rf_maxnode, test, type = "prob"))
plot(roc(test$Diseased, maxnode_pred_prob$`0`), main = "MaxNodes ROC Curve")
```




```
print(roc(test$Diseased, maxnode_pred_prob$`0`))
```

```
##
## Call:
## roc.default(response = test$Diseased, predictor = maxnode_pred_prob$`0`)
##
## Data: maxnode_pred_prob$`0` in 122 controls (test$Diseased 0) > 52 cases (test$Diseased 1).
## Area under the curve: 0.7306
```

2.4.7 Model#6 Best ntrees

This parameter is the number of trees of the ensamble. Same as maxnodes we need to iterate between various ntress values and summarize in the end. We use the already obtained values for mtry and maxnodes.

```
tic("Best ntrees Model...")
store_maxtrees <- list()
suppressWarnings(for (ntree in c(250, 300, 350, 400, 450, 500, 550, 600, 800, 1000, 2000)) {
    set.seed(1234)
    rf_maxtrees <- train(Diseased~.,
        data = train,
        method = "rf",
        metric = "Accuracy",
        tuneGrid = tuneGrid,
        trControl = trControl,
        importance = TRUE,
        nodesize = 14,
        maxnodes = 25,
        ntree = ntree)
    key <- toString(ntree)
    store_maxtrees[[key]] <- rf_maxtrees
```

```
})
results_tree <- resamples(store_maxtrees)
summary(results_tree)
```

```
##
## Call:
## summary.resamples(object = results_tree)
##
## Models: 250, 300, 350, 400, 450, 500, 550, 600, 800, 1000, 2000
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## 250  0.6000000 0.6562500 0.6867015 0.6939866 0.7300305  0.8    0
## 300  0.6097561 0.6750000 0.6986063 0.7013676 0.7300305  0.8    0
## 350  0.6000000 0.6750000 0.6986063 0.7013066 0.7300305  0.8    0
## 400  0.6000000 0.6562500 0.6986063 0.6988066 0.7300305  0.8    0
## 450  0.5750000 0.6750000 0.6986063 0.6988066 0.7300305  0.8    0
## 500  0.6000000 0.6750000 0.6986063 0.7013066 0.7300305  0.8    0
## 550  0.5750000 0.6812500 0.7108014 0.7037456 0.7300305  0.8    0
## 600  0.6000000 0.6769817 0.7071429 0.7038066 0.7300305  0.8    0
## 800  0.6250000 0.6769817 0.7071429 0.7063066 0.7300305  0.8    0
## 1000 0.6250000 0.6750000 0.6986063 0.7062456 0.7300305  0.8    0
## 2000 0.6250000 0.6562500 0.6986063 0.7013066 0.7300305  0.8    0
##
## Kappa
##             Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 250  -0.06312292 0.02474747 0.1623056 0.1539486 0.2580432 0.3962264    0
## 300  -0.01562500 0.05159081 0.1957463 0.1771509 0.2580432 0.3962264    0
## 350  -0.06312292 0.05781848 0.1957463 0.1710629 0.2580432 0.3962264    0
## 400  -0.06312292 0.05781848 0.1957463 0.1669650 0.2580432 0.3962264    0
## 450  -0.16438356 0.05781848 0.1957463 0.1609368 0.2580432 0.3962264    0
## 500  -0.06312292 0.05781848 0.1957463 0.1710629 0.2580432 0.3962264    0
## 550  -0.16438356 0.06840708 0.2115230 0.1693498 0.2605987 0.3962264    0
## 600  -0.06312292 0.06840708 0.1957463 0.1752983 0.2580432 0.3962264    0
## 800  -0.02739726 0.06840708 0.1957463 0.1788709 0.2580432 0.3962264    0
## 1000 -0.01562500 0.08527507 0.1957463 0.1842618 0.2580432 0.3962264    0
## 2000 -0.05660377 0.05781848 0.1957463 0.1765031 0.2580432 0.3962264    0
```

```
# Evaluate the model
maxtree_prediction <-predict(rf_maxtrees,test)

maxtree_acc <- confusionMatrix(maxtree_prediction, test$Diseased)$overall["Accuracy"]
print(maxtree_acc)
```

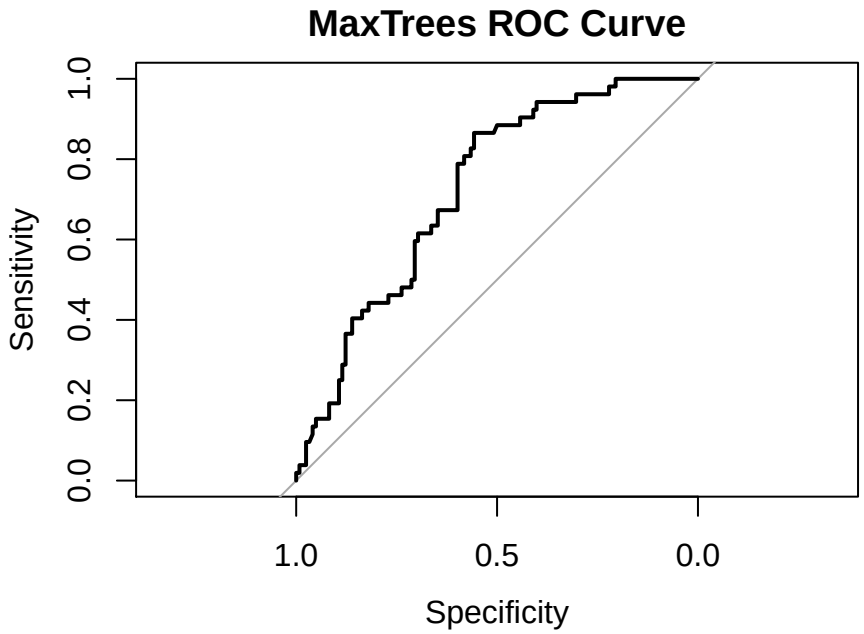
```
##  Accuracy
## 0.7241379
```

```
# Append to accuracy_results table
accuracy_results <- bind_rows(accuracy_results,
  data_frame(Model = "Model#6: Default + MTRY + MaxNodes + N-Trees Predictions",
             Accuracy = maxtree_acc
            ))
toc()
```

```
## Best ntrees Model...: 27.044 sec elapsed
```

ROC Curve for ntrees

```
maxtree_pred_prob <- as.data.frame(predict(rf_maxtrees, test, type = "prob"))
plot(roc(test$Diseased, maxtree_pred_prob$`0`), main = "MaxTrees ROC Curve")
```



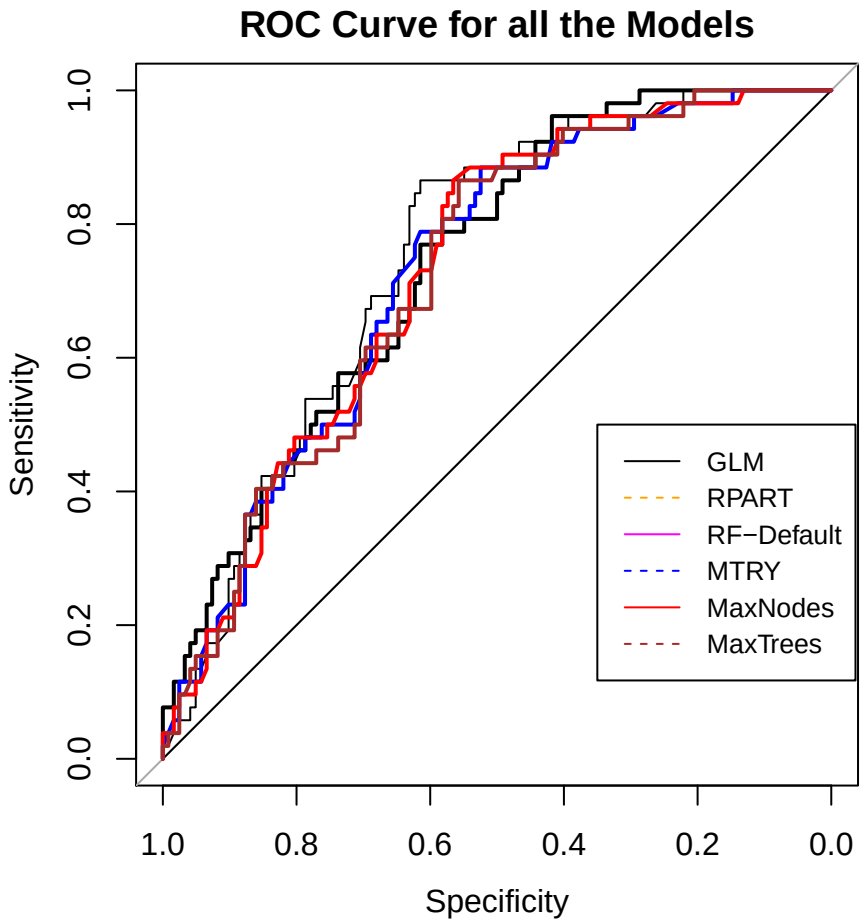

```
print(roc(test$Diseased, maxtree_pred_prob$`0`))
```

```
##
## Call:
## roc.default(response = test$Diseased, predictor = maxtree_pred_prob$`0`)
##
## Data: maxtree_pred_prob$`0` in 122 controls (test$Diseased 0) > 52 cases (test$Diseased 1).
## Area under the curve: 0.7251
```

ROC Curve for all the models

```
roc_glm = suppressWarnings(roc(test$Diseased, glm_pred_prob$`predict(fit_glm, test, type = "response")`)
roc_rpart = roc(test$Diseased, rpart_pred_prob$`0`)
roc_rf_default = roc(test$Diseased, default_pred_prob$`0`)
roc_mtry = roc(test$Diseased, mtry_pred_prob$`0`)
roc_maxnode = roc(test$Diseased, maxnode_pred_prob$`0`)
roc_maxtree = roc(test$Diseased, maxtree_pred_prob$`0`)

par(pty="s")
plot(roc_glm, main = "ROC Curve for all the Models")
lines(roc_rpart, 'orange')
lines(roc_rf_default, 'magenta')
lines(roc_mtry, col = 'blue')
lines(roc_maxnode, col = 'red')
lines(roc_maxtree, col = 'brown')
legend(0.35, 0.5, legend=c("GLM", "RPART", "RF-Default", "MTRY", "MaxNodes", "MaxTrees"),
       col=c("black", "orange", "magenta", "blue", "red", "brown"), lty=1:2, cex=0.8)
```



3 Final Results

Below is the list of all Model's Accuracy and we can see that the Model#3 and Model#6 has the better accuracy of 0.7241379

```
# Print accuracy that we btained from all the Models above
accuracy_results %>% knitr::kable()
```

Model	Accuracy
Model#1: GLM(Generalized Liner Model) Predictions	0.7011494
Model#2: RPART Predictions	0.7011494
Model#3: Default Predictions	0.7241379
Model#4: Default + MTRY Predictions	0.7011494
Model#5: Default + MTRY + MaxNodes Predictions	0.7126437
Model#6: Default + MTRY + MaxNodes + N-Trees Predictions	0.7241379

4 Conclusion

The Random Forest default model **and** Random Forest with n-trees seems to get us the maximum accuracy. There is still a room to improve the accuracy by performing different approaches such as by removing redundant feature method with highly correlated attributes or using Rank Features by Importance method.

5 Envrionment Used for this Project

```
# Show the environment used for this project
print("Envrionment Information:")
```

```
## [1] "Envrionment Information:"
```

```
version
```

```
##                _
## platform       x86_64-apple-darwin15.6.0
## arch           x86_64
## os             darwin15.6.0
## system         x86_64, darwin15.6.0
## status
## major          3
## minor          6.0
## year           2019
## month          04
## day            26
## svn rev        76424
## language       R
## version.string R version 3.6.0 (2019-04-26)
## nickname       Planting of a Tree
```