Data Science CapStone Project - Indian Liver Patient Records

Shweta Kothadiya

3/20/2021

Introduction/ Overview/Executive Summary:

The name of the project that I have chosen is "Indian Liver Patient Records". The dataset was downloaded from the UCI ML Repository (under kaggle website) Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.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.

Goal of the Project: The goal of this project is to calculate the highest accuracy by testing multiple models and to find out the possible variables that could have the highest impact on the liver disease using the results from the tested models. The key steps that would be performed during this project are - installing required packages/ libraries and loading the data, data analysis, data cleaning, data wrangling, then data visualization via plotting the data on maps. After this, data will be partitioned into train and test sets. Then I would be running few models against a variable / combination of variables to calculate the accuracy on each model. At the end, the results will be displayed to show the model that gives the highest accuracy followed by conclusion to summarize overall work and findings.

Method & Analysis Section:

Step 1 - Installing packages and Loading libraries:

```
if(!require(tidyverse)) install.packages("tidyverse", repos =
"http://cran.us.r-project.org")

## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse
1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart
```

```r
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(rpart)
library(randomForest)
```

For this project, "indian_liver_patient.csv" was downloaded from kaggle site to my machine. Function "read_csv" is used to import data into R as a dataframe. This same file can be found under this website - https://www.kaggle.com/uciml/indian-liver-patient-records

```
indianLiverPatient <-
read.csv("C:\\Users\\Sanchit\\Documents\\Shweta\\edX\\Data Science
Professional Certificate\\indian_liver_patient.csv", stringsAsFactors =
FALSE)
```

Step 2 - Data Analysis:

Let's check the first 6 lines of "indianLiverPatient" dataset.

```
head(indianLiverPatient)

##   Age Gender Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 1  65 Female             0.7              0.1                  187
## 2  62   Male            10.9              5.5                  699
## 3  62   Male             7.3              4.1                  490
## 4  58   Male             1.0              0.4                  182
## 5  72   Male             3.9              2.0                  195
## 6  46   Male             1.8              0.7                  208
##   Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
Albumin
## 1                       16                         18            6.8
3.3
## 2                       64                        100            7.5
3.2
## 3                       60                         68            7.0
3.3
## 4                       14                         20            6.8
3.4
## 5                       27                         59            7.3
2.4
## 6                       19                         14            7.6
4.4
##   Albumin_and_Globulin_Ratio Dataset
## 1                       0.90       1
## 2                       0.74       1
## 3                       0.89       1
```

```
## 4                       1.00       1
## 5                       0.40       1
## 6                       1.30       1
```

Let's analyze what are the dimensions of data frame and class of each column.

```
dim(indianLiverPatient)
```

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

```
class(indianLiverPatient)
```

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

```
class(indianLiverPatient$Age)
```

```
## [1] "integer"
```

```
class(indianLiverPatient$Gender)
```

```
## [1] "character"
```

```
class(indianLiverPatient$Total_Bilirubin)
```

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

```
class(indianLiverPatient$Direct_Bilirubin)
```

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

```
class(indianLiverPatient$Alkaline_Phosphotase)
```

```
## [1] "integer"
```

```
class(indianLiverPatient$Alamine_Aminotransferase)
```

```
## [1] "integer"
```

```
class(indianLiverPatient$Aspartate_Aminotransferase)
```

```
## [1] "integer"
```

```
class(indianLiverPatient$Total_Protiens)
```

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

```
class(indianLiverPatient$Albumin)
```

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

```
class(indianLiverPatient$Albumin_and_Globulin_Ratio)
```

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

```
class(indianLiverPatient$Dataset)
```

```
## [1] "integer"
```

Here is the count of records for each Gender in this data frame.

```
indianLiverPatient %>% filter(Gender=="Male") %>% nrow()

## [1] 441

indianLiverPatient %>% filter(Gender=="Female") %>% nrow()

## [1] 142
```

Let's find out how many records are there for Patient with Liver disease and without liver disease.

```
indianLiverPatient %>% filter(Dataset=="1") %>% nrow()

## [1] 416

indianLiverPatient %>% filter(Dataset=="2") %>% nrow()

## [1] 167
```

Here is the summary of the data after the analysis was performed as above.

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 are as below: • Age of the patient • Gender of the patient • Total Bilirubin • Direct Bilirubin • Alkaline Phosphatase • Alanine Aminotransferase • Aspartate Aminotransferase • Total Proteins • Albumin • Albumin and Globulin Ratio • Dataset: field used to split the data into two sets (patient with liver disease, or no disease)

Step 3 - Data Cleaning:

The dataset was checked for any NA values in any columns using the below code. I have found out that the column "Albumin_and_Globulin_Ratio" had NA values for 4 records. Those NA values were replaced with the average value for that column. Then the class of Gender column was set as factor and assigned values for Females and Males as 0 and 1 respectively.

```
indianLiverPatient[rowSums(is.na(indianLiverPatient))>0,]

##     Age Gender Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 210  45 Female             0.9              0.3                   189
## 242  51   Male             0.8              0.2                   230
## 254  35 Female             0.6              0.2                   180
## 313  27   Male             1.3              0.6                   106
##     Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
Albumin
```

```
## 210                         23                          33            6.6
3.9
## 242                         24                          46            6.5
3.1
## 254                         12                          15            5.2
2.7
## 313                         25                          54            8.5
4.8
##     Albumin_and_Globulin_Ratio Dataset
## 210                         NA       1
## 242                         NA       1
## 254                         NA       2
## 313                         NA       2
```

```
indianLiverPatient$Albumin_and_Globulin_Ratio=ifelse(is.na(indianLiverPatient
$Albumin_and_Globulin_Ratio),ave(indianLiverPatient$Albumin_and_Globulin_Rati
o,FUN = function(x)mean(x,na.rm =
TRUE)),indianLiverPatient$Albumin_and_Globulin_Ratio)

indianLiverPatient[rowSums(is.na(indianLiverPatient))>0,]
```

```
##  [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] Dataset
## <0 rows> (or 0-length row.names)
```

Step 4 - Data Wrangling:

Here we are setting the "Dataset" and "Gender" columns as factor. Now "Female" will be shown as "0" and "Male" will be shown as "1". The same way, the "Dataset" column was set as factor too.

```
ind_liv_patient_clean <- indianLiverPatient %>%
  mutate(Dataset = factor(indianLiverPatient$Dataset),
              Gender = factor(x=indianLiverPatient$Gender,levels =
c('Female','Male'),labels=c(0,1))) %>%
  select(Age, Gender, Total_Bilirubin, Direct_Bilirubin,
Alkaline_Phosphotase, Alamine_Aminotransferase, Aspartate_Aminotransferase,
Total_Protiens, Albumin, Albumin_and_Globulin_Ratio, Dataset)
```

We can check the first 6 records of the cleaned data along with the "Gender" and "Dataset" columns which we had set them as "factor".

```
head(ind_liv_patient_clean)
```

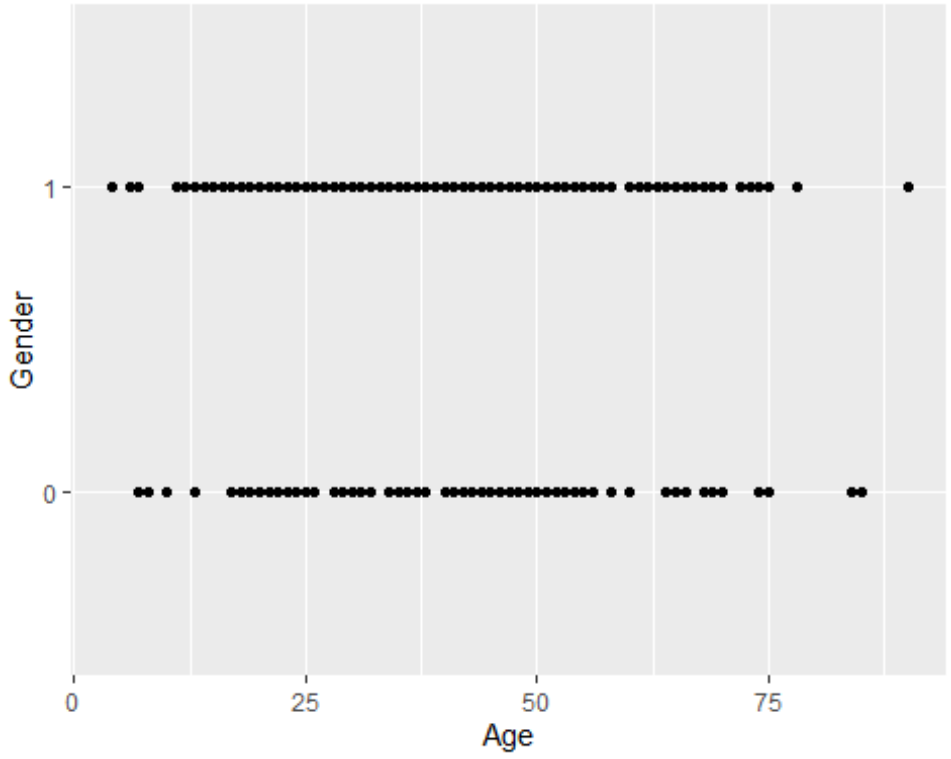
```
##   Age Gender Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 1  65      0             0.7              0.1                   187
## 2  62      1            10.9              5.5                   699
```

```
## 3  62      1             7.3              4.1                   490
## 4  58      1             1.0              0.4                   182
## 5  72      1             3.9              2.0                   195
## 6  46      1             1.8              0.7                   208
##   Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens
Albumin
## 1                       16                         18            6.8
3.3
## 2                       64                        100            7.5
3.2
## 3                       60                         68            7.0
3.3
## 4                       14                         20            6.8
3.4
## 5                       27                         59            7.3
2.4
## 6                       19                         14            7.6
4.4
##   Albumin_and_Globulin_Ratio Dataset
## 1                       0.90       1
## 2                       0.74       1
## 3                       0.89       1
## 4                       1.00       1
## 5                       0.40       1
## 6                       1.30       1

class(ind_liv_patient_clean$Gender)

## [1] "factor"

class(ind_liv_patient_clean$Dataset)

## [1] "factor"
```

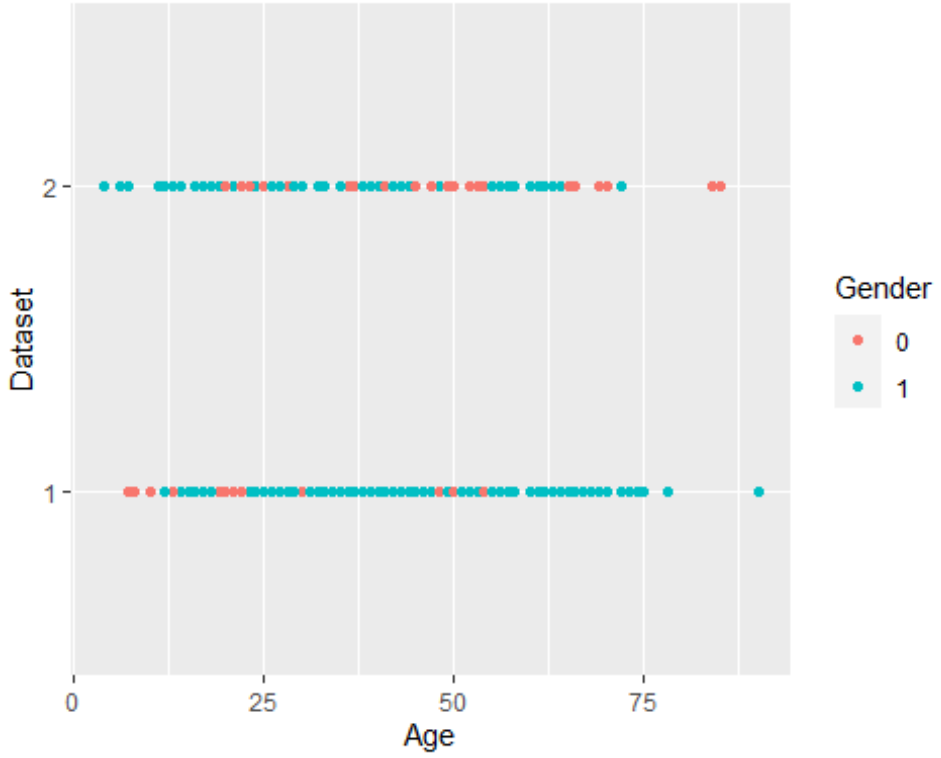
Step 5 - Data Visualization:

Let's visualize the data using plots.

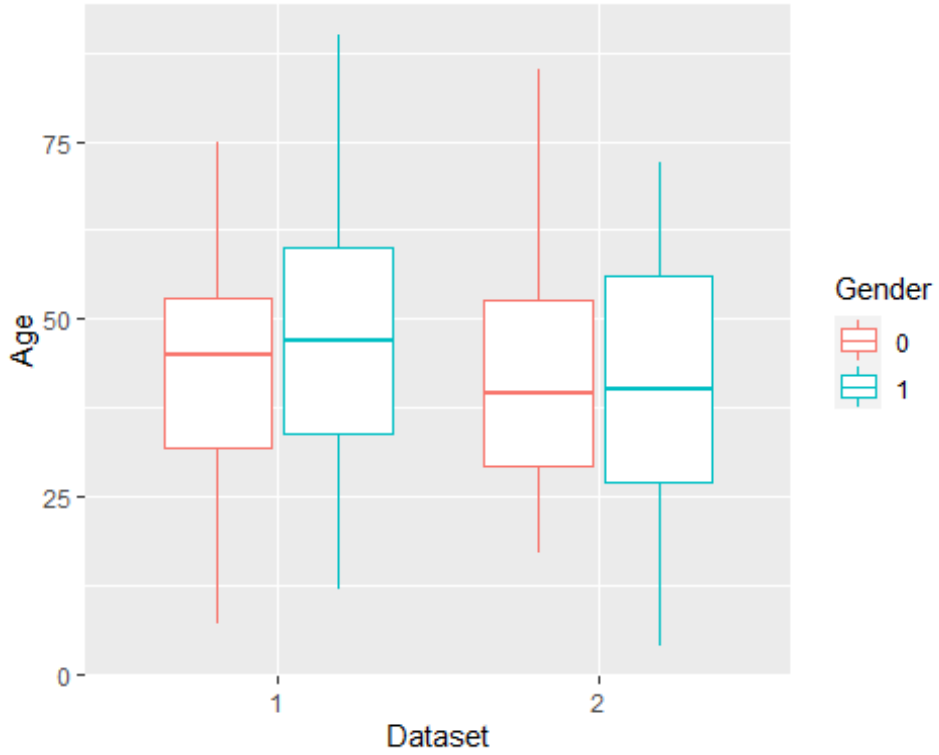
```
#Plot of Age vs. Gender
qplot(x=Age, y=Gender, data=ind_liv_patient_clean, geom="point")
```



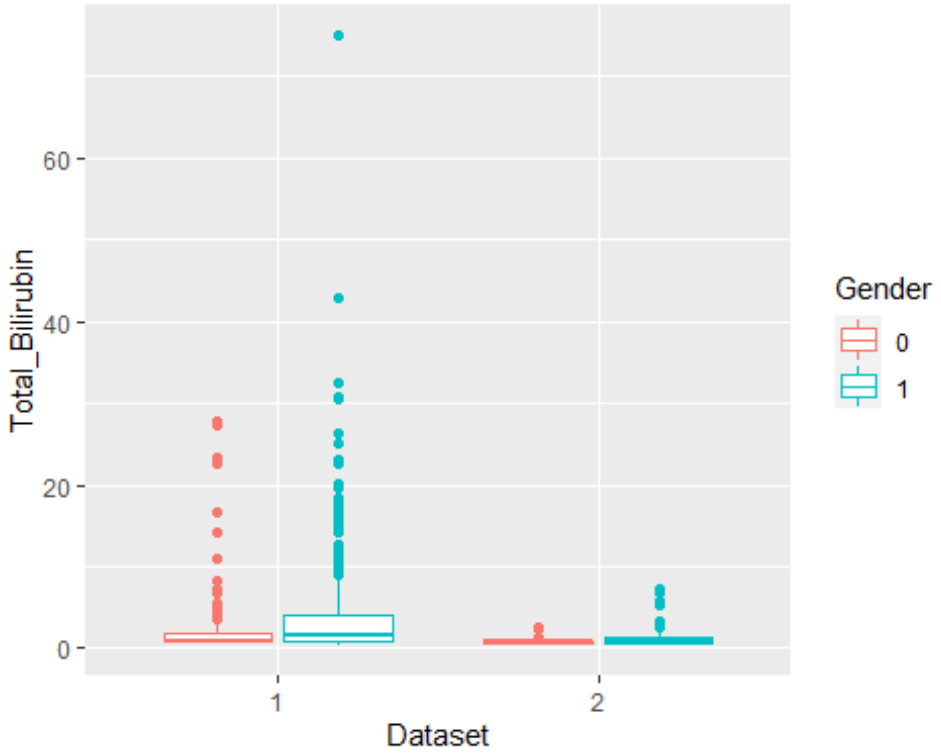

```
#Plot of Age vs. Dataset, grouped by Gender and colored by Gender.
ind_liv_patient_clean %>% group_by(Gender) %>% ggplot(aes(x=Age, y =Dataset,
color=Gender)) + geom_point()
```



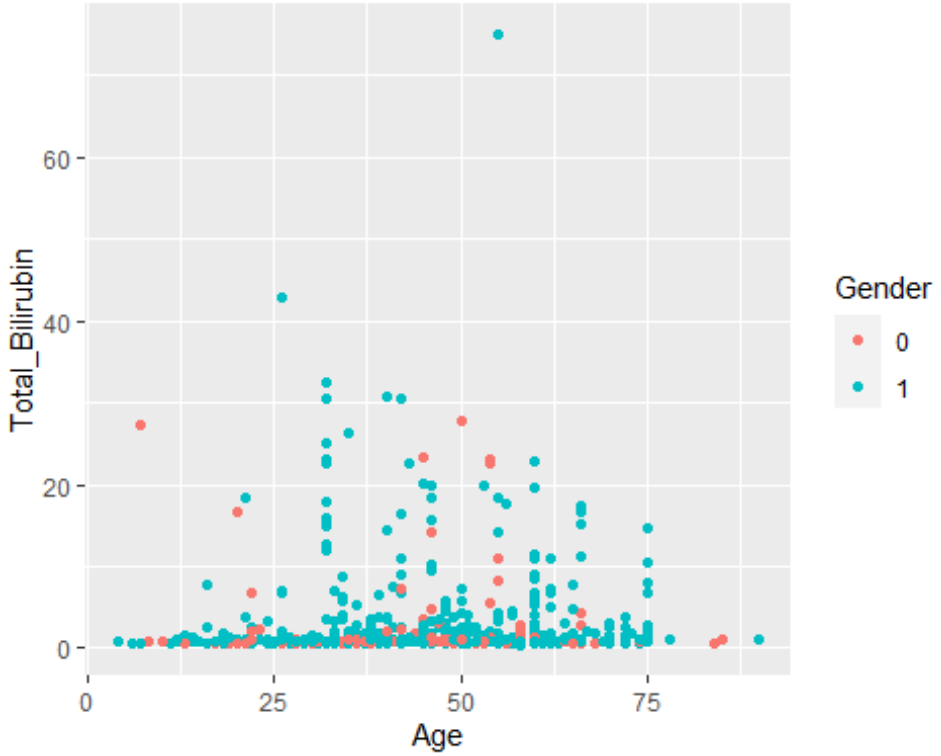
```
#BoxPlot of Dataset vs. Age,grouped by Gender and colored by Gender.
ind_liv_patient_clean %>% group_by(Gender) %>% ggplot(aes(x=Dataset, y =Age,
color=Gender)) + geom_boxplot()
```



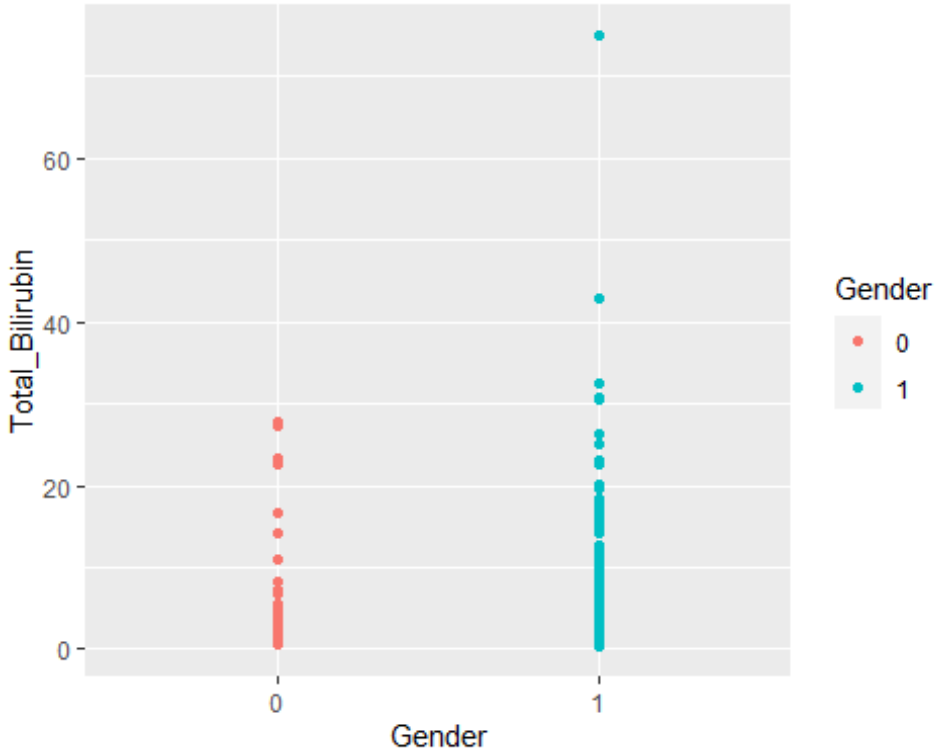

```
#Boxplot of Dataset vs. Total Bilirubin and grouped and colored by Gender
ind_liv_patient_clean %>% group_by(Gender) %>% ggplot(aes(x=Dataset, y
=Total_Bilirubin, color=Gender)) + geom_boxplot()
```



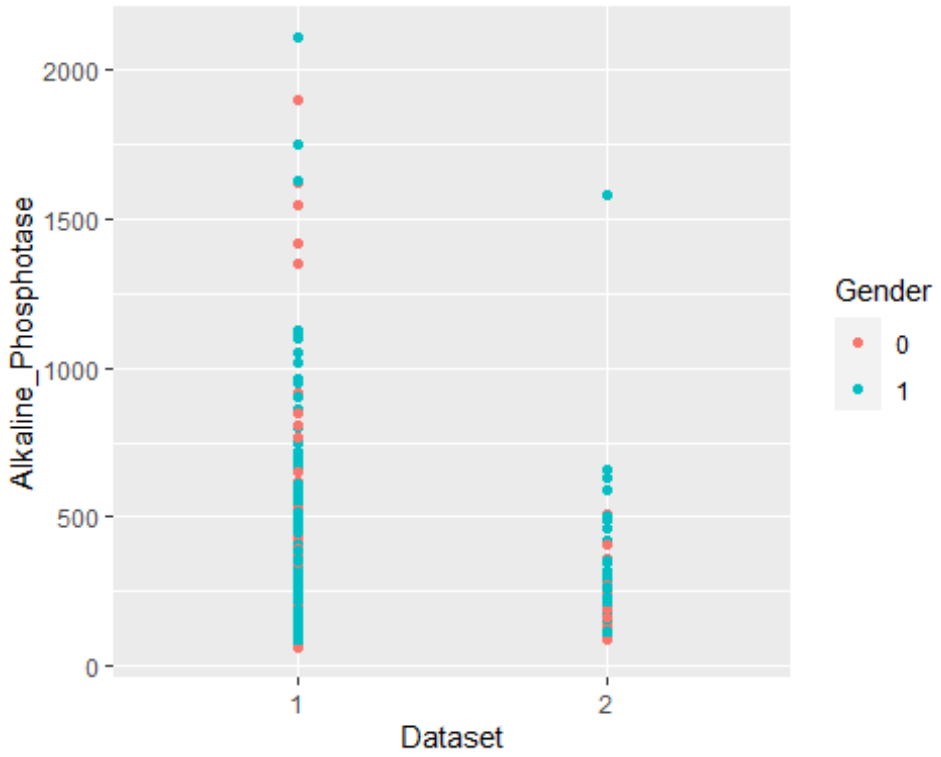

```
#ggplot of Age vs. Total Bilirubin and colored by Gender
ind_liv_patient_clean %>% ggplot(aes(Age,Total_Bilirubin, color=Gender)) +
geom_point()
```



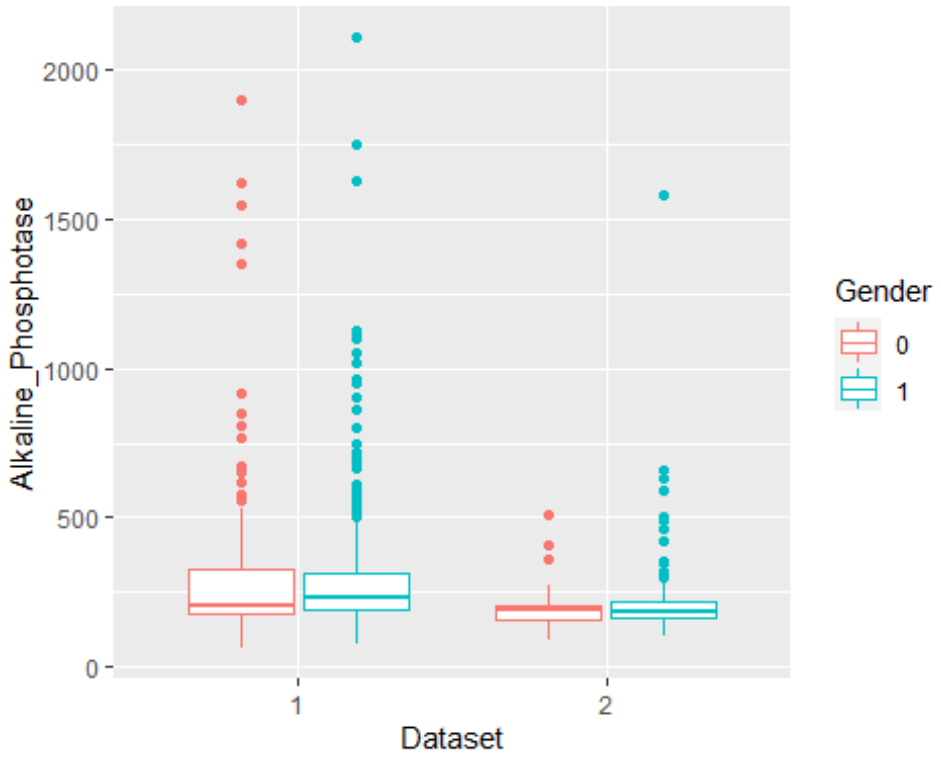
```r
#ggplot of Gender and Total Bilirubin colored by Gender
ind_liv_patient_clean %>% ggplot(aes(Gender,Total_Bilirubin, color=Gender)) +
geom_point()
```




```r
#ggplot of Dataset and Alkaline Phosphotase colored by Gender
ind_liv_patient_clean %>% ggplot(aes(x=Dataset, y=Alkaline_Phosphotase,
color=Gender)) + geom_point()
```




```
#Boxplot of Dataset vs. Alkaline Phosphotase grouped and colored by Gender
ind_liv_patient_clean %>% group_by(Gender) %>% ggplot(aes(x=Dataset, y
=Alkaline_Phosphotase, color=Gender)) + geom_boxplot()
```



Step 6 - Data Partitioning:

After the data visualization, let's start working towards splitting the data into train and test sets so we can start training the models to calculate the accuracy.

First I will set the seed to 1 with "sample.kind as Rounding" as I am using R version 4.0.3. Then I will create partition and will send 20 percent data into test set and rest 80 percent under training. Then we will check the dimentions of test_set and train_set.

```
set.seed(1, sample.kind = "Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
sampler
## used

test_index <- createDataPartition(ind_liv_patient_clean$Dataset, times = 1, p
= 0.2, list = FALSE)
test_set <- ind_liv_patient_clean[test_index,]
train_set <- ind_liv_patient_clean[-test_index,]

dim(test_set)

## [1] 118  11

dim(train_set)

## [1] 465  11
```

Step 7 - Models and analysis to calculate the accuracy:

The models that I have used in this project to predict the disease and to calculate the accuracy are Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Generalized Linear Model (GLM), Classification tree, and Random Forest model.

1. Training LDA (Linear Discriminant Analysis) model on train_set to check if liver disease is affected by Total Bilirubin. Then I will be using the train_lda model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_lda <- train(Dataset ~ Total_Bilirubin, method = "lda", data =
train_set)
lda_preds <- predict(train_lda, test_set)
mean(lda_preds == test_set$Dataset)

## [1] 0.7118644
```

2. Training QDA (Quadratic Discriminant Analysis) model on train_set to check if liver disease is affected by Total Bilirubin. Then I will be using the train_qda model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_qda <- train(Dataset ~ Total_Bilirubin, method = "qda", data =
train_set)
```

```
qda_preds <- predict(train_qda, test_set)
mean(qda_preds == test_set$Dataset)

## [1] 0.4915254
```

3. Training LDA (Linear Discriminant Analysis) model on train_set to check if liver disease is affected by Total Bilirubin & Age combined. Then I will be using the train_lda_TB_Age model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_lda_TB_Age <- train(Dataset ~ Total_Bilirubin + Age, method = "lda",
data = train_set)
lda_preds_TB_Age <- predict(train_lda_TB_Age, test_set)
mean(lda_preds_TB_Age == test_set$Dataset)

## [1] 0.7033898
```

4. Training QDA (Quadratic Discriminant Analysis) model on train_set to check if liver disease is affected by Total Bilirubin & Age combined. Then I will be using the train_qda_TB_Age model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_qda_TB_Age <- train(Dataset ~ Total_Bilirubin + Age, method = "qda",
data = train_set)
qda_preds_TB_Age <- predict(train_qda_TB_Age, test_set)
mean(qda_preds_TB_Age == test_set$Dataset)

## [1] 0.5084746
```

5. Training LDA (Linear Discriminant Analysis) model on train_set to check if liver disease is affected by Total Bilirubin & Gender combined. Then I will be using the train_lda_TB_gender model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_lda_TB_gender <- train(Dataset ~ Total_Bilirubin + Gender, method =
"lda", data = train_set)
lda_preds_TB_gender <- predict(train_lda_TB_gender, test_set)
mean(lda_preds_TB_gender == test_set$Dataset)

## [1] 0.7118644
```

6. Training QDA (Quadratic Discriminant Analysis) model on train_set to check if liver disease is affected by Total Bilirubin & Gender combined. Then I will be using the train_qda_TB_gender model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_qda_TB_gender <- train(Dataset ~ Total_Bilirubin + Gender, method =
"qda", data = train_set)
qda_preds_TB_gender <- predict(train_qda_TB_gender, test_set)
mean(qda_preds_TB_gender == test_set$Dataset)

## [1] 0.4915254
```

7. Training LDA (Linear Discriminant Analysis) model on train_set to check if liver disease is affected by Age & Gender combined. Then I will be using the train_lda_Age_gender model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_lda_Age_gender <- train(Dataset ~ Age + Gender, method = "lda", data =
train_set)
lda_preds_Age_gender <- predict(train_lda_Age_gender, test_set)
mean(lda_preds_Age_gender == test_set$Dataset)

## [1] 0.6949153
```

8. Training QDA (Quadratic Discriminant Analysis) model on train_set to check if liver disease is affected by Age & Gender combined. Then I will be using the train_qda_Age_gender model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_qda_Age_gender <- train(Dataset ~ Age + Gender, method = "qda", data =
train_set)
qda_preds_Age_gender <- predict(train_qda_Age_gender, test_set)
mean(qda_preds_Age_gender == test_set$Dataset)

## [1] 0.7033898
```

9. Training LDA (Linear Discriminant Analysis) model on train_set to check if liver disease is affected by Age. Then I will be using the train_lda_age model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_lda_age <- train(Dataset ~ Age, method = "lda", data = train_set)
lda_preds_age <- predict(train_lda_age, test_set)
mean(lda_preds_age == test_set$Dataset)

## [1] 0.720339
```

10. Training QDA (Quadratic Discriminant Analysis) model on train_set to check if liver disease is affected by Age. Then I will be using the train_qda_age model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_qda_age <- train(Dataset ~ Age, method = "qda", data = train_set)
qda_preds_age <- predict(train_qda_age, test_set)
mean(qda_preds_age == test_set$Dataset)

## [1] 0.7033898
```

11. Training GLM (Generalized Linear Model) model on train_set to check if liver disease is affected by Age. Then I will be using the train_glm_age model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_glm_age <- train(Dataset ~ Age, method = "glm", data = train_set)
glm_preds_age <- predict(train_glm_age, test_set)
mean(glm_preds_age == test_set$Dataset)
```

```
## [1] 0.720339
```

12. Training GLM (Generalized Linear Model) model on train_set to check if liver disease is affected by Age, Gender, Direct Bilirubin combined. Then I will be using the train_glm_age_gender_db model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_glm_age_gender_db <- train(Dataset ~ Age + Gender + Direct_Bilirubin,
method = "glm", data = train_set)
glm_preds_age_gender_db <- predict(train_glm_age_gender_db, test_set)
mean(glm_preds_age_gender_db == test_set$Dataset)

## [1] 0.6949153
```

13. Training GLM (Generalized Linear Model) model on train_set to check if liver disease is affected by all predictors combined. Then I will be using the train_glm_all model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_glm_all <- train(Dataset ~ ., method = "glm", data = train_set)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
glm_preds_all <- predict(train_glm_all, test_set)
mean(glm_preds_all == test_set$Dataset)
```

```
## [1] 0.720339
```

14. Training LDA (Linear Discriminant Analysis) model on train_set to check if liver disease is affected by Alkaline_Phosphotase. Then I will be using the train_lda_AlkPho model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_lda_AlkPho <- train(Dataset ~ Alkaline_Phosphotase, method = "lda",
data = train_set)
lda_preds_AlkPho <- predict(train_lda_AlkPho, test_set)
mean(lda_preds_AlkPho == test_set$Dataset)
```

```
## [1] 0.7118644
```

15. Training QDA (Quadratic Discriminant Analysis) model on train_set to check if liver disease is affected by Alkaline_Phosphotase. Then I will be using the train_qda_AlkPho model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_qda_AlkPho <- train(Dataset ~ Alkaline_Phosphotase, method = "qda",
data = train_set)
qda_preds_AlkPho <- predict(train_qda_AlkPho, test_set)
mean(qda_preds_AlkPho == test_set$Dataset)
```

```
## [1] 0.7118644
```

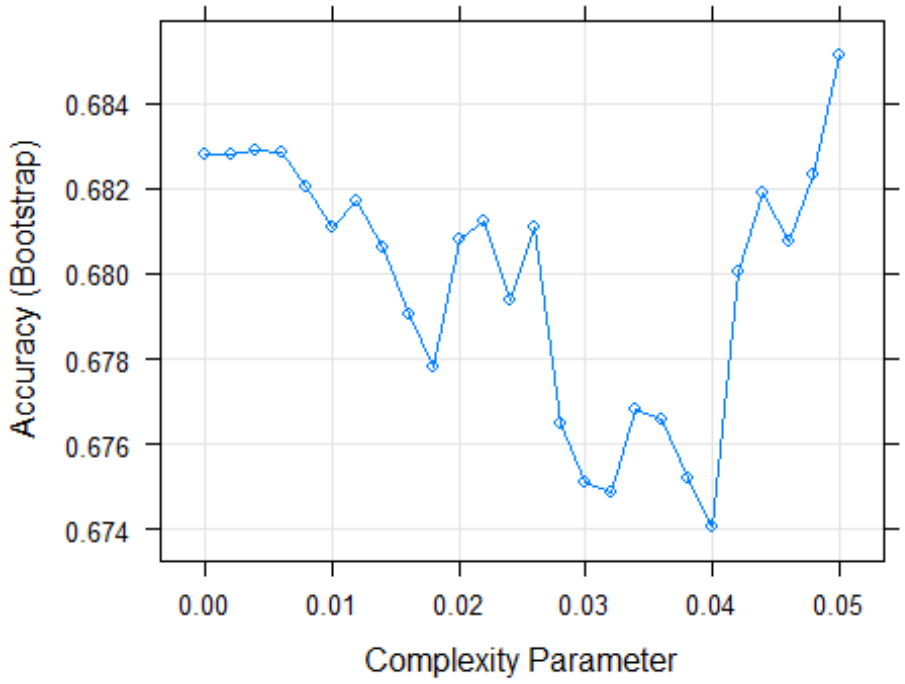
16. Training GLM (Generalized Linear Model) model on train_set to check if liver disease is affected by Alkaline_Phosphotase. Then I will be using the train_glm_AlkPho model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function.

```
train_glm_AlkPho <- train(Dataset ~ Alkaline_Phosphotase, method = "glm",
data = train_set)
glm_preds_AlkPho <- predict(train_glm_AlkPho, test_set)
mean(glm_preds_AlkPho == test_set$Dataset)
```

```
## [1] 0.7118644
```

17. Training Classification Tree model on train_set to check if liver disease is affected by all the predictors combined. Then I will be using the train_rpart model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function. After this we can find what is the best tune and final model that this "rpart" method gives us.

```
train_rpart <- train(Dataset ~ ., method = "rpart", tuneGrid = data.frame(cp
= seq(0, 0.05, 0.002)), data = train_set)
plot(train_rpart)
```




```
rpart_preds <- predict(train_rpart, test_set)
mean(rpart_preds == test_set$Dataset)

## [1] 0.7118644

train_rpart$bestTune

##      cp
## 26 0.05

train_rpart$finalModel

## n= 465
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 465 133 1 (0.7139785 0.2860215) *
```

```
train_rpart
```

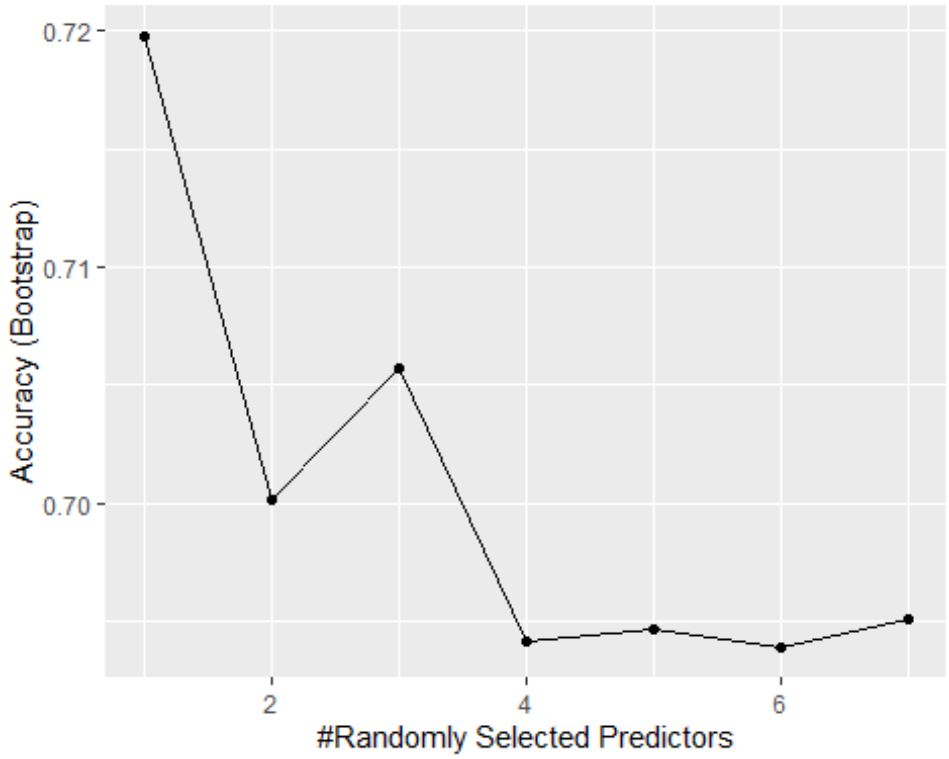
```
## CART
##
## 465 samples
##  10 predictor
##   2 classes: '1', '2'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 465, 465, 465, 465, 465, 465, ...
## Resampling results across tuning parameters:
##
##   cp     Accuracy   Kappa
##   0.000  0.6828167  0.17901588
##   0.002  0.6828167  0.17901588
##   0.004  0.6828764  0.17490095
##   0.006  0.6828422  0.17298948
##   0.008  0.6820628  0.17579942
##   0.010  0.6810747  0.17331229
##   0.012  0.6817071  0.16860843
##   0.014  0.6806230  0.17216176
##   0.016  0.6790438  0.16379348
##   0.018  0.6778331  0.16003718
##   0.020  0.6808265  0.16278565
##   0.022  0.6812384  0.16068544
##   0.024  0.6793787  0.14498635
##   0.026  0.6810867  0.14182295
##   0.028  0.6764698  0.12895535
##   0.030  0.6751121  0.12430545
##   0.032  0.6748517  0.11498631
##   0.034  0.6768270  0.11403849
##   0.036  0.6765777  0.11386902
##   0.038  0.6751884  0.10682612
##   0.040  0.6740388  0.09986735
##   0.042  0.6800579  0.08240076
##   0.044  0.6819226  0.07292559
##   0.046  0.6807464  0.04880327
##   0.048  0.6823283  0.04546101
##   0.050  0.6851509  0.05579870
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.05.
```

18. Training RF (Random Forest Model) model on train_set to check if liver disease is affected by all the predictors combined. Then I will be using the train_rf model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function. Then we can plot the model. After that we can use "varimp" function (Variable Importance) to find the most important variable from this model.

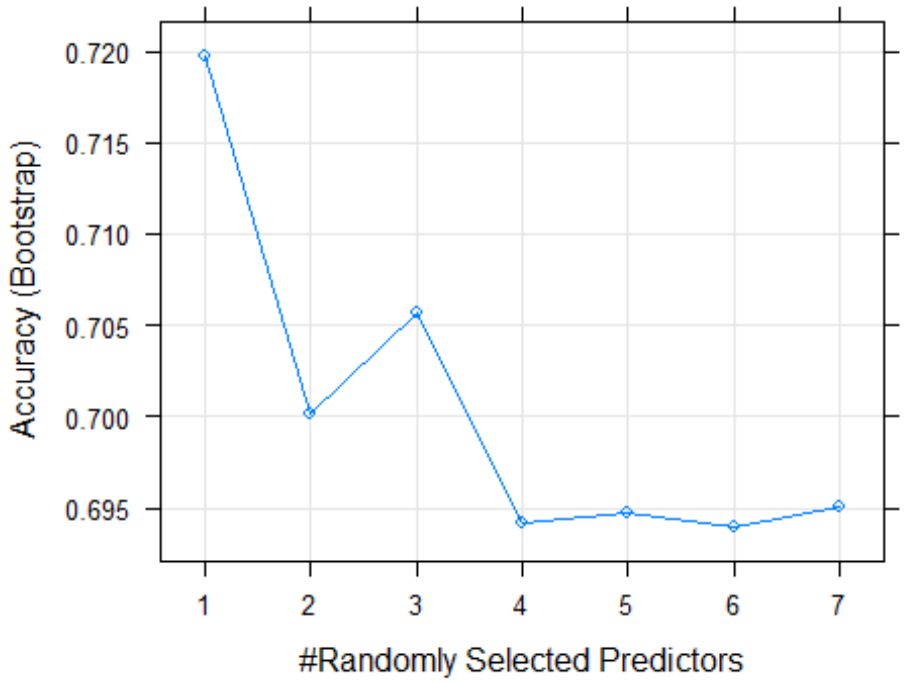
```r
train_rf <- train(Dataset ~ ., data = train_set, method = "rf", ntree = 100,
tuneGrid = data.frame(mtry = seq(1:7)))
rf_preds <- predict(train_rf, test_set)
mean(rf_preds == test_set$Dataset)
```

```
## [1] 0.7372881
```

```r
ggplot(train_rf)
```



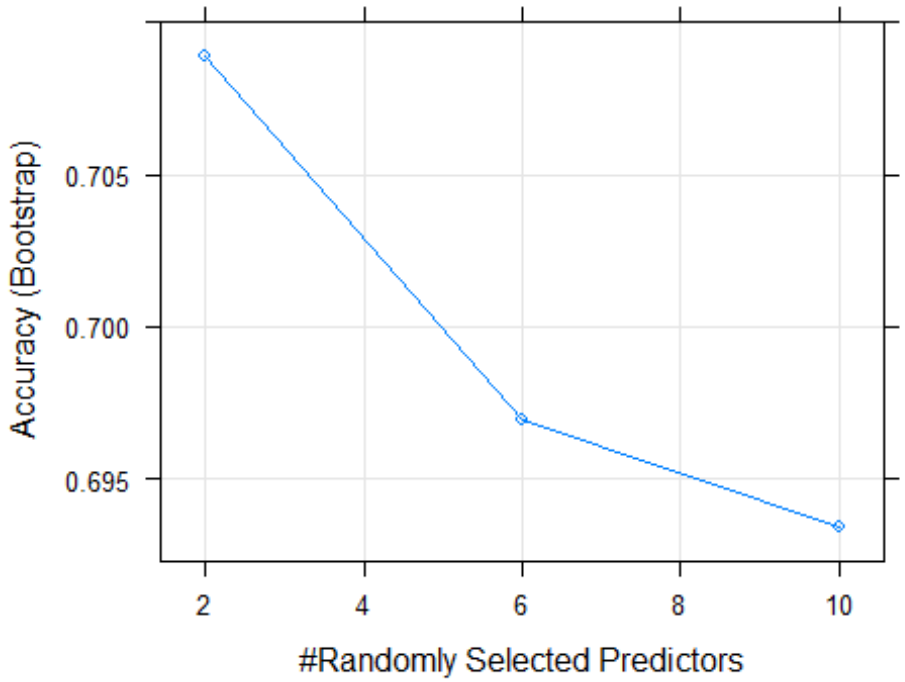

```r
plot(train_rf)
```




```
varImp(train_rf)

## rf variable importance
##
##                             Overall
## Alamine_Aminotransferase     100.00
## Alkaline_Phosphotase          95.44
## Aspartate_Aminotransferase    94.10
## Age                           84.52
## Total_Bilirubin               81.02
## Albumin                       72.69
## Total_Protiens                69.12
## Direct_Bilirubin              63.99
## Albumin_and_Globulin_Ratio    61.56
## Gender1                        0.00
```

19. Random Forest model with nodesize as 50 and maxnodes as 25. Then we can plot train_rf_n model. Then we can use the train_rf_n model against test_set. Then the accuracy will be checked between model's prediction and actual data in test_set using mean function. After that we can use "varimp" function (Variable Importance) to find the most important variable from this model.

```
train_rf_n <- train(Dataset ~ ., data = train_set, method = "rf", nodesize =
50, maxnodes = 25)
plot(train_rf_n)
```



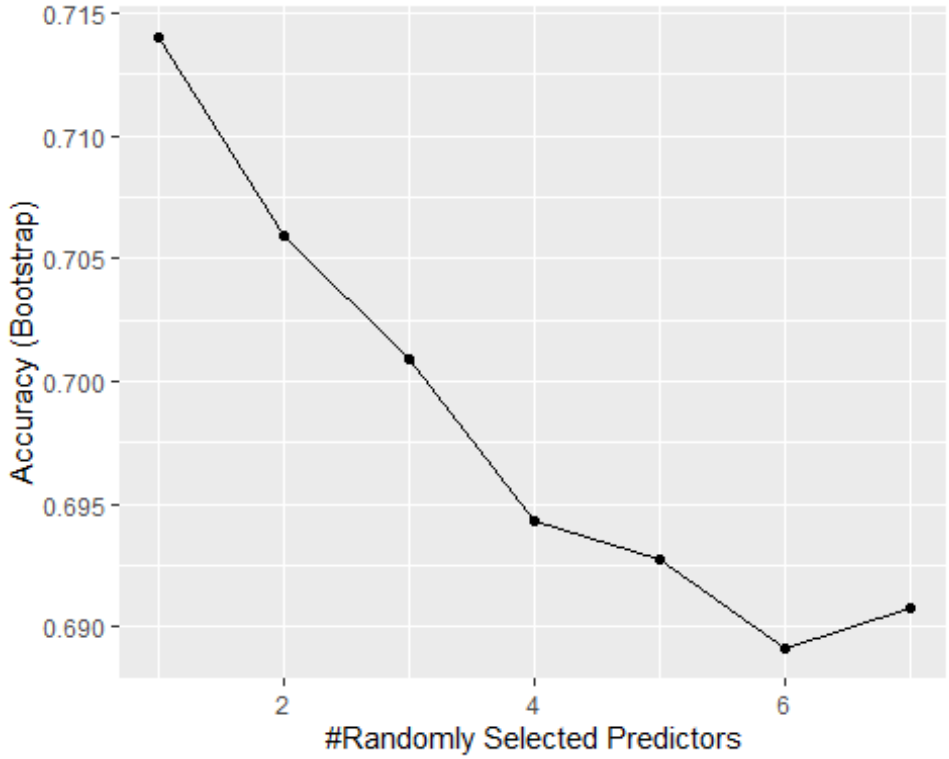

```
rf_preds_n <- predict(train_rf_n, test_set)
mean(rf_preds_n == test_set$Dataset)

## [1] 0.6949153

varImp(train_rf_n)

## rf variable importance
##
##                             Overall
## Alkaline_Phosphotase         100.00
## Aspartate_Aminotransferase    87.30
## Alamine_Aminotransferase      84.70
## Total_Bilirubin               80.62
## Direct_Bilirubin              78.10
## Age                           76.93
## Albumin                       47.71
## Albumin_and_Globulin_Ratio    46.42
## Total_Protiens                34.72
## Gender1                        0.00
```

Results:

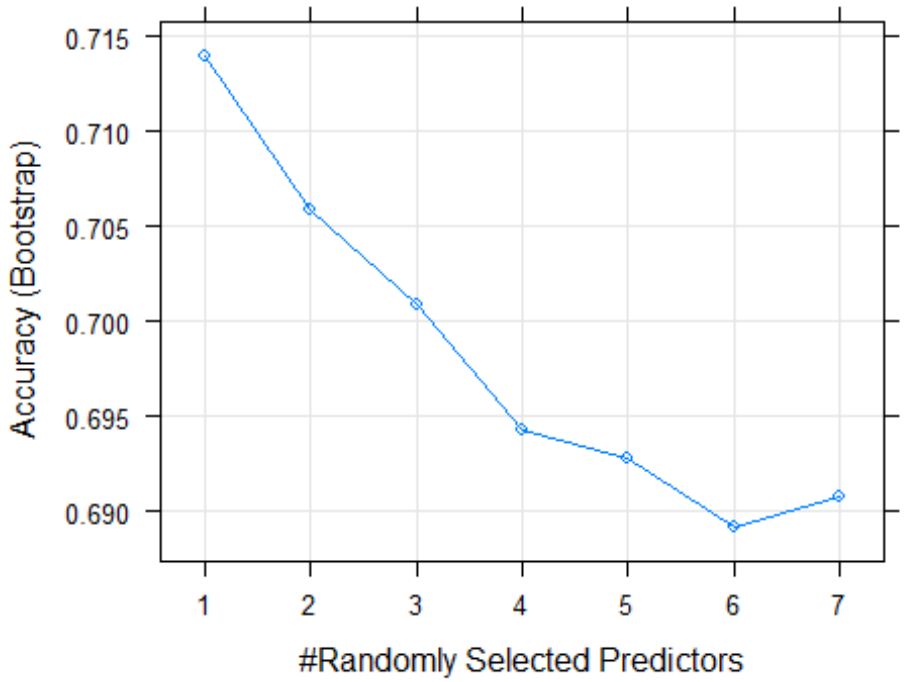
Model# 18 “Random Forest Model “is giving the best and highest accuracy among all the models that we have tested so far. The accuracy that we received using this model is 0.7372881 for the first time and 0.7288136 for the 2nd time. The code of this model is as below.

Comments on the code: Training RF (Random Forest Model) model on train_set to check if liver disease is affected by all the predictors combined. Then we used the train_rf model against test_set. Then the accuracy was checked between model's prediction and actual data in test_set using mean function. The plot on this model has been shown and the variable importance has been displayed here as well to find the most important variable from this model. As per the results, the most important variable that affects the liver disease was "Alamine_Aminotransferase" followed by "Alkaline_Phosphotase".

```
train_rf <- train(Dataset ~ ., data = train_set, method = "rf", ntree = 100,
tuneGrid = data.frame(mtry = seq(1:7)))
rf_preds <- predict(train_rf, test_set)
mean(rf_preds == test_set$Dataset)

## [1] 0.7288136

ggplot(train_rf)
```




```
plot(train_rf)
```



Conclusion:

The best model that gave the highest accuracy was “train_rf” using the “Random Forest method”. We got the accuracy as 0.7372881 for the first time and 0.7288136 for the 2nd time. This model was tested against all the variables. According to this “Random Forest model”, the most important variable that causes the liver disease is “Alamine_Aminotransferase“ followed by “Alkaline_Phosphotase“, “Aspartate_Aminotransferase“, “Age”, and “Total_Bilirubin“ in the respective order.

After this “Random Forest model”, there were 3 other models that gave the 2nd highest accuracy which is 0.720339. Those models are - model# 9 “train_lda_age “ with the use of “Linear Discriminant Analysis” method, model# 11 “train_glm_age “ with the use of “Generalized Linear Method“ and model# 13 “train_glm_all “ with the use of “Generalized Linear Method“. Out of these three models, two models were tested against only one variable and that was “Age”. So to some extent, “Age” factor also is a big contributor that causes the liver disease. According to our “Random Forest model”, Age variable came as number 4 in the rank that can cause the liver disease.

Another thing that I noticed was if we to compare “Linear Discriminant Analysis” (LDA) and “Quadratic Discriminant Analysis” (QDA) models in general, then “Linear Discriminant Analysis” (LDA) model seemed better than using “Quadratic Discriminant Analysis” (QDA) model. The reason being LDA gave the higher accuracy that we were looking for when it was tested against multiple different variables like Total Bilirubin, Age, Gender, and combinations of these variables.

Overall I would recommend using “Random Forest” model. This model would be beneficial to predict the contributing factors in the order of importance that affect the liver disease. The data from this model can be used by medical professionals to identify the patients that could be at high risk of having a liver disease in order to monitor on the regular basis.