Liver Disease Classification

Nabeel Khan

27-May-2020

Contents

1 Introduction

This report is part of the 'HarvardX: PH125.9x Data Science: Capstone' course. In this report, we chose a dataset of our choice and develop machine learning models to perform binary classification to diagnose liver disease.

1.1 Background

The liver plays a vital role in keeping us healthy. The liver's main job is to filter the blood coming from the digestive tract before passing it to the rest of the body. The liver also turns nutrients into chemicals our body needs, converts food into energy, and filters out poisons. The malfunctioning of the liver affects the whole body.

The problems with liver patients are not easily discovered in an early stage. Early diagnosis of liver disease increases the survival rate of patients. The liver disease can be detected by analyzing the levels of enzymes in the human blood [2, 3]. Therefore, a classification algorithm capable of automatically detecting the liver disease can assist the doctors in diagnosis. The classification techniques are commonly employed in various automatic medical diagnoses tools[1].

1.2 Aim of Project

The patients with liver disease are on the rise because of excessive consumption of alcohol, inhale of harmful gases, or intake of contaminated food. This project aims to develop a binary classifier, which can use blood enzymes information to diagnose liver disease.

2 Dataset and Evaluation Metrics

We use the liver patient records, which are collected from North East of Andhra Pradesh, India. The data set contains:

1. 416 liver patient records and 167 non-liver patient records.

2.1 Download Data

The dataset is publically available online both at Kaggle and UCI repository. We download data from the website. Then, we split data into training and validation sets.

- 10% of the data is used for validation, and 90

```
################################
#  Install packages (if not installed)
################################
# Note: this process could take a couple of minutes
repos_path<- "http://cran.us.r-project.org"
if(!require(tidyverse)) install.packages("tidyverse", repos =repos_path)
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

```
if(!require(caret)) install.packages("caret", repos = repos_path)
```

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

##
## Attaching package: 'caret'

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

```
if(!require(data.table)) install.packages("data.table", repos =repos_path)
```

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

## Warning: package 'data.table' was built under R version 3.6.3

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

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

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

```
if(!require(lubridate)) install.packages("lubridate", repos = repos_path)
```

```
## Loading required package: lubridate

##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year

## The following object is masked from 'package:base':
##
##     date
```

```
if(!require(dplyr)) install.packages("dplyr", repos = repos_path)
if(!require(sjmisc)) install.packages("dplyr", repos = repos_path)
```

```
## Loading required package: sjmisc

## Warning: package 'sjmisc' was built under R version 3.6.3

##
## Attaching package: 'sjmisc'

## The following object is masked from 'package:purrr':
##
##     is_empty

## The following object is masked from 'package:tidyr':
##
##     replace_na

## The following object is masked from 'package:tibble':
##
##     add_case
```

```
if(!require(scales)) install.packages("scales", repos = repos_path)
```

```
## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
if(!require(caret)) install.packages("caret", repos = repos_path)
if(!require(caretEnsemble)) install.packages("caretEnsemble", repos = repos_path)
```

```
## Loading required package: caretEnsemble
```

```
## Warning: package 'caretEnsemble' was built under R version 3.6.3
```

```
##
## Attaching package: 'caretEnsemble'
```

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

```
################################
# Load libraries
################################
library(lubridate)
library(tidyverse)
library(dplyr)
library(lubridate)
library(sjmisc)
library(scales)
library(caret)
library(caretEnsemble)
```

```
################################
# Downloading data
################################
# Indian Live Patient Records :
 # https://www.kaggle.com/uciml/indian-liver-patient-records/
 # https://archive.ics.uci.edu/ml/machine-learning-databases/00225/Indian Liver Patient Dataset (ILPD).

url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00225/Indian Liver Patient Dataset (I

# Download csv
liverData <- read.csv(url)

# Rename columns of csv
colnames(liverData)<- c("Age","Gender","Total_Bilirubin","Direct_Bilirubin", "Alkaline_Phosphotase","Ala

################################
# Creating training and validation sets
################################

# Validation set will be 10% of whole data
set.seed(1, sample.kind = "Rounding")
```

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

```
test_index <- createDataPartition(y = liverData$Dataset, times = 1, p = 0.1, list = FALSE)

training <- liverData[-test_index,]
validation <- liverData[test_index,]
```

```
 # Removing the objects from environment as no longer required
rm(liverData)
```

2.2 Metrics

To evaluate the performance of classifiers, we will use the following metrics:

1. **Accuracy** It is the ratio of the number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Truepositives + Truenegatives}{TotalPredictions} \tag{1}$$

2. **Sensitivity** It is also referred as true positive rate or recall. It is the proportion of true positives that are correctly identified.

$$Sensitivity = \frac{Numberoftruepositives}{Numberoftruepositives + Numberoffalsenegatives} \tag{2}$$

3. **Precision** It is defined as the proportion of the true positives against all the positive results.

$$Precision = \frac{Numberoftruepositives}{Numberoftruepositives + Numberoffalsepositives} \tag{3}$$

4. **Specificity** It is the true negative rate. It is the proportion of true negatives that are correctly identified.

$$Specificity = \frac{Numberoftruenegatives}{Numberoftruenegatives + Numberoffalsepositives} \tag{4}$$

5. **F1 Score** Specificity and sensitivity are rates. Therefore, it is more appropriate to compute the harmonic average of precision and recall. The metric is referred to as F1 Score:

$$F1Score = 2 * \frac{Precision - Recall}{Precision + Recall} \tag{5}$$

3 Data Exploration

The dataset contains 11 variables, namely, “Age”, "Gender‘‘, “Total_Bilirubin”, or “Alkaline_Phosphotase”. The ’Dataset’ variable indicates if the liver has a disease or not. For instance, a value of 1 means that the liver is damaged, while a value of 2 means that the liver is healthy.

All other variables except “Age”, “Gender”, and “Dataset” represent the amount of enzymes or proteins in the blood. These variables (or a subset) will be used to train our machine learning models to make diagnoses.

```
head(training)
```

Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Ar
62	Male	10.9	5.5	699	64	
62	Male	7.3	4.1	490	60	
58	Male	1.0	0.4	182	14	
72	Male	3.9	2.0	195	27	
46	Male	1.8	0.7	208	19	
26	Female	0.9	0.2	154	16	

The training dataset has 523 records. We can see that the "Albumin_and_Globulin_Ratio" variable has 4 null values. The remaining variables do not contain any null values.

- The validation data has no null values (confirmed via summary).

```
sprintf("Rows of training dataset = %d", nrow(training))
```

```
## [1] "Rows of training dataset = 523"
```

```
print("==========================")
```

```
## [1] "=========================="
```

```
summary(training)
```

```
##       Age           Gender    Total_Bilirubin Direct_Bilirubin
##  Min.   : 4.00   Female:125   Min.   : 0.40   Min.   : 0.100
##  1st Qu.:33.00   Male  :398   1st Qu.: 0.80   1st Qu.: 0.200
##  Median :45.00                Median : 1.00   Median : 0.300
##  Mean   :45.33                Mean   : 3.22   Mean   : 1.446
##  3rd Qu.:58.00                3rd Qu.: 2.60   3rd Qu.: 1.300
##  Max.   :90.00                Max.   :75.00   Max.   :19.700
##
##  Alkaline_Phosphotase Alamine_Aminotransferase Aspartate_Aminotransferase
##  Min.   :  63.0       Min.   :  10.00          Min.   :  10.0
##  1st Qu.: 176.0       1st Qu.:  24.00          1st Qu.:  25.0
##  Median : 208.0       Median :  35.00          Median :  41.0
##  Mean   : 289.9       Mean   :  76.34          Mean   : 105.0
##  3rd Qu.: 298.0       3rd Qu.:  60.00          3rd Qu.:  86.5
##  Max.   :1896.0       Max.   :1680.00          Max.   :4929.0
##
##  Total_Protiens    Albumin      Albumin_and_Globulin_Ratio    Dataset
##  Min.   :2.70   Min.   :0.900   Min.   :0.3000             Min.   :1.000
##  1st Qu.:5.80   1st Qu.:2.600   1st Qu.:0.7000             1st Qu.:1.000
##  Median :6.60   Median :3.100   Median :0.9300             Median :1.000
##  Mean   :6.49   Mean   :3.147   Mean   :0.9458             Mean   :1.281
##  3rd Qu.:7.20   3rd Qu.:3.800   3rd Qu.:1.1000             3rd Qu.:2.000
##  Max.   :9.50   Max.   :5.500   Max.   :2.8000             Max.   :2.000
##                                 NA's   :4
```

```
summary(validation)
```

```
##       Age          Gender   Total_Bilirubin  Direct_Bilirubin
##  Min.   : 8.0   Female:16   Min.   : 0.600   Min.   : 0.100
##  1st Qu.:27.5   Male  :43   1st Qu.: 0.800   1st Qu.: 0.200
##  Median :38.0               Median : 1.100   Median : 0.400
##  Mean   :39.2               Mean   : 4.037   Mean   : 1.863
##  3rd Qu.:49.5               3rd Qu.: 2.300   3rd Qu.: 1.300
##  Max.   :75.0               Max.   :26.300   Max.   :12.100
##  Alkaline_Phosphotase Alamine_Aminotransferase Aspartate_Aminotransferase
##  Min.   :  92.0       Min.   :  10.0           Min.   :  15.0
```

```
##  1st Qu.: 160.5       1st Qu.:  22.0           1st Qu.:  28.5
##  Median : 215.0       Median :  36.0           Median :  43.0
##  Mean   : 298.4       Mean   : 120.6           Mean   : 155.0
##  3rd Qu.: 305.0       3rd Qu.:  63.0           3rd Qu.:  90.0
##  Max.   :2110.0       Max.   :2000.0           Max.   :2946.0
##  Total_Protiens     Albumin      Albumin_and_Globulin_Ratio
##  Min.   :3.600   Min.   :0.900   Min.   :0.3000
##  1st Qu.:5.700   1st Qu.:2.500   1st Qu.:0.8000
##  Median :6.300   Median :3.200   Median :1.0000
##  Mean   :6.419   Mean   :3.095   Mean   :0.9593
##  3rd Qu.:7.200   3rd Qu.:3.550   3rd Qu.:1.1900
##  Max.   :9.600   Max.   :4.700   Max.   :1.9000
##     Dataset
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.339
##  3rd Qu.:2.000
##  Max.   :2.000
```

3.1 Data Wrangling

3.1.1 Remove null values

The variable "Albumin_and_Globulin_Ratio" has four null values. We replace null values with the mean of the variable as done commonly in data science.

```
# Replace null values with the mean
training$Albumin_and_Globulin_Ratio[is.na(training$Albumin_and_Globulin_Ratio)] <- mean(training$Albumi
```

3.1.2 Create Diagnosis Variable

To improve readability, we create a new column, namely, "LiverDisease", which can have one of the following values:

1. Malignant (M) indicating that the patient has liver disease.
2. Benign (B) indicating that the patient has no liver disease.

We further delete the "Dataset" variable as it is no longer needed. We apply these operations to both training and validation datasets.

```
# Adding a new column, which will contain the disease information
training <- transform(training, LiverDisease= ifelse(Dataset==1, "M","B"))
validation <- transform(validation, LiverDisease= ifelse(Dataset==1, "M","B"))

# Deleting the column 'Dataset' as no longer required
training<-within(training, rm(Dataset))
validation<-within(validation, rm(Dataset))

# Displaying the first six rows
head(training)
```

Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Ar
62	Male	10.9	5.5	699	64	
62	Male	7.3	4.1	490	60	
58	Male	1.0	0.4	182	14	
72	Male	3.9	2.0	195	27	
46	Male	1.8	0.7	208	19	
26	Female	0.9	0.2	154	16	

4 Data Analysis

In this section, we extract insights from all variables to get in-depth understanding.

4.1 Age

The dataset consists of patients with varying ages ranging from 4 to 90. The distribution of ages shows a nice spread and indicates that the dataset is unbiased towards a specific age group.

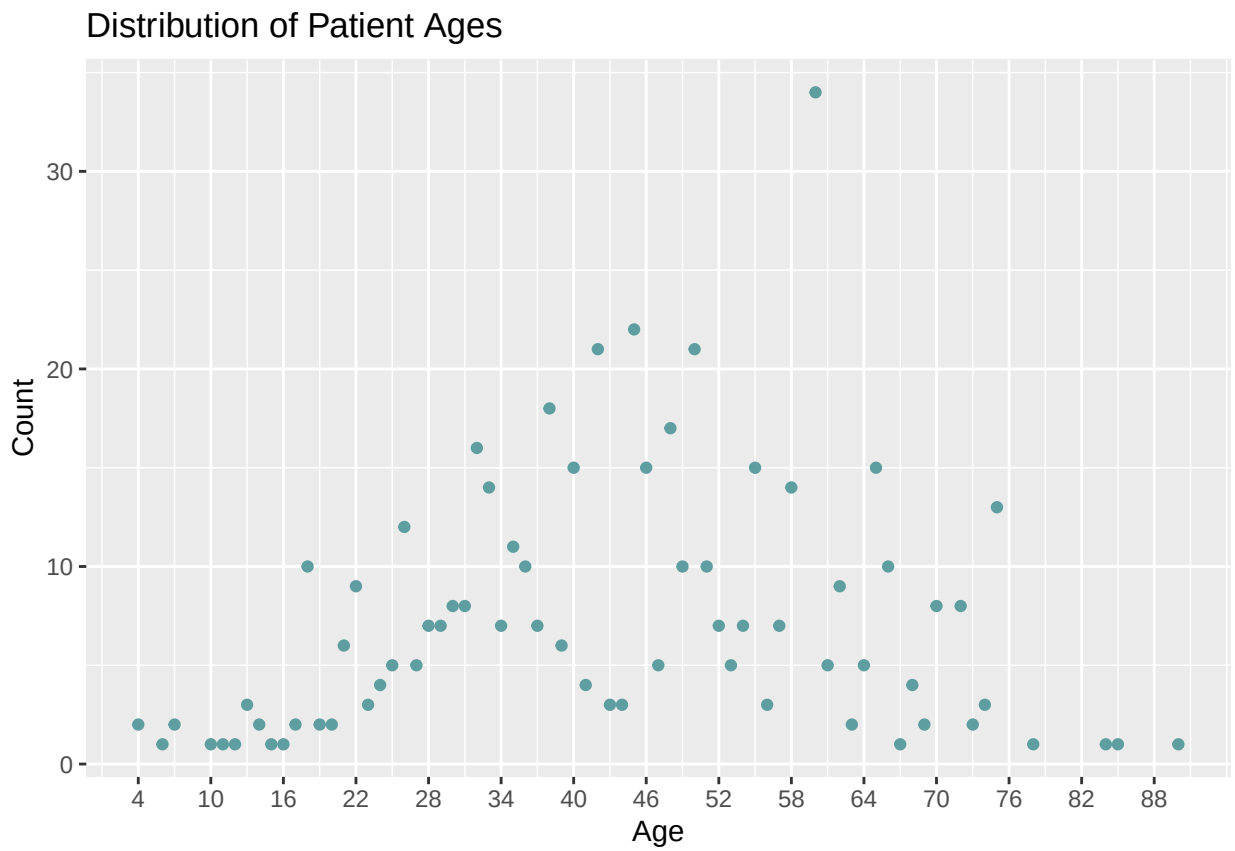
```
sprintf("Minimum age = %d",min(training$Age))
```

```
## [1] "Minimum age = 4"
```

```
sprintf("Maximum age = %d",max(training$Age))
```

```
## [1] "Maximum age = 90"
```

```
# Extracting frequency of patient ages
age_stats <-as.data.frame(table(training$Age))
names(age_stats)<- c("Age","Count")

# Remvoing the factor
age_stats$Age<-as.numeric(levels(age_stats$Age))

# Plotting distribution of ages
age_stats %>% ggplot(aes(Age, Count)) +
  geom_point(color="cadetblue") +
  scale_x_continuous(breaks = round(seq(min(age_stats$Age),
                                        max(age_stats$Age), by = 6),1)) +
  ggtitle("Distribution of Patient Ages")
```



We breakdown the distribution of ages to the presence or absence of liver disease. Again, we notice a good spread of age group for both scenarios.

```
# Plotting distributions of ages based on liver diseases
training %>%
  ggplot(aes(as.numeric(row.names(training)),Age, color=LiverDisease)) +
  geom_point() +
  labs(y="Age", x = "Number of patients")+
  facet_wrap( ~ LiverDisease) +
  ggtitle("Distribution of ages based on liver disease")
```

4.2 Gender

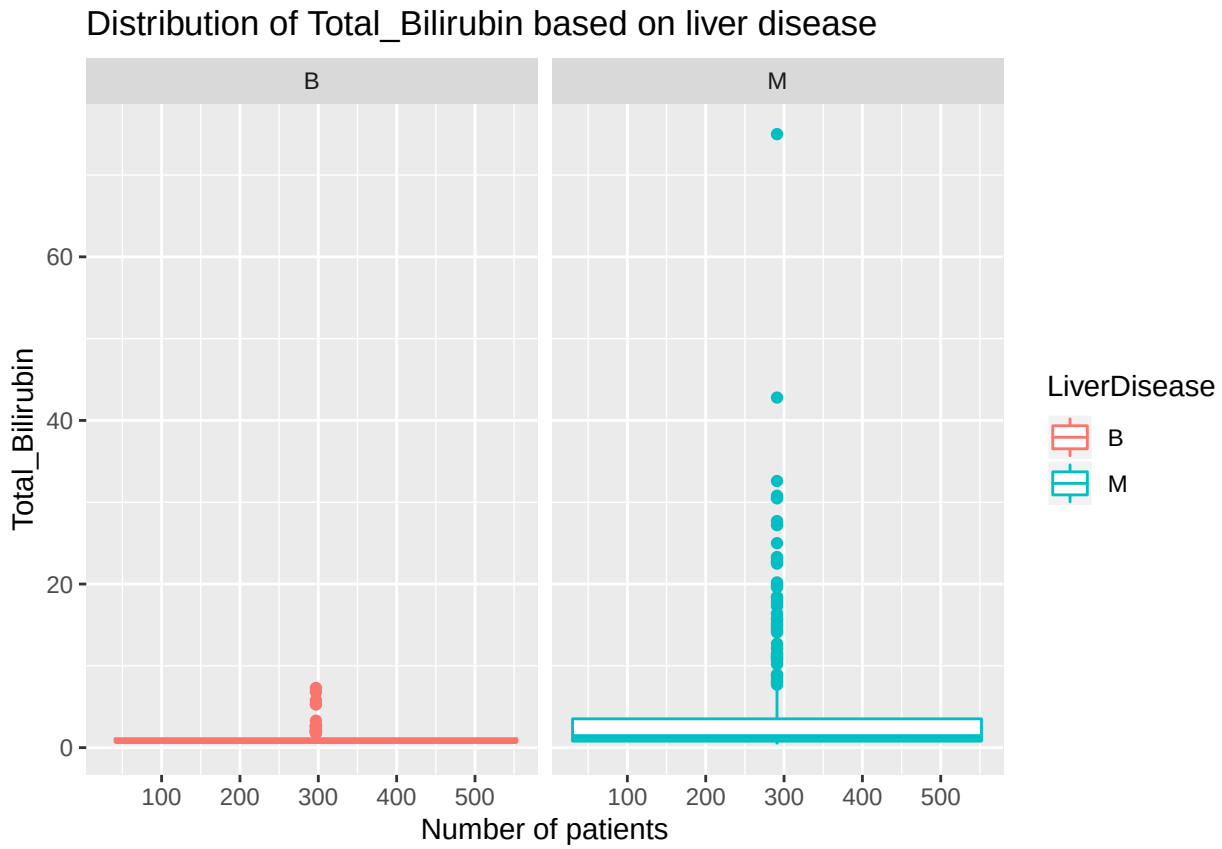
76% of the patient records are of males. It would be good to have a more and less equal distribution of records for both genders, although we do not expect it to make any difference in our models.

```
# Getting summary of genders
summary(training$Gender)
```

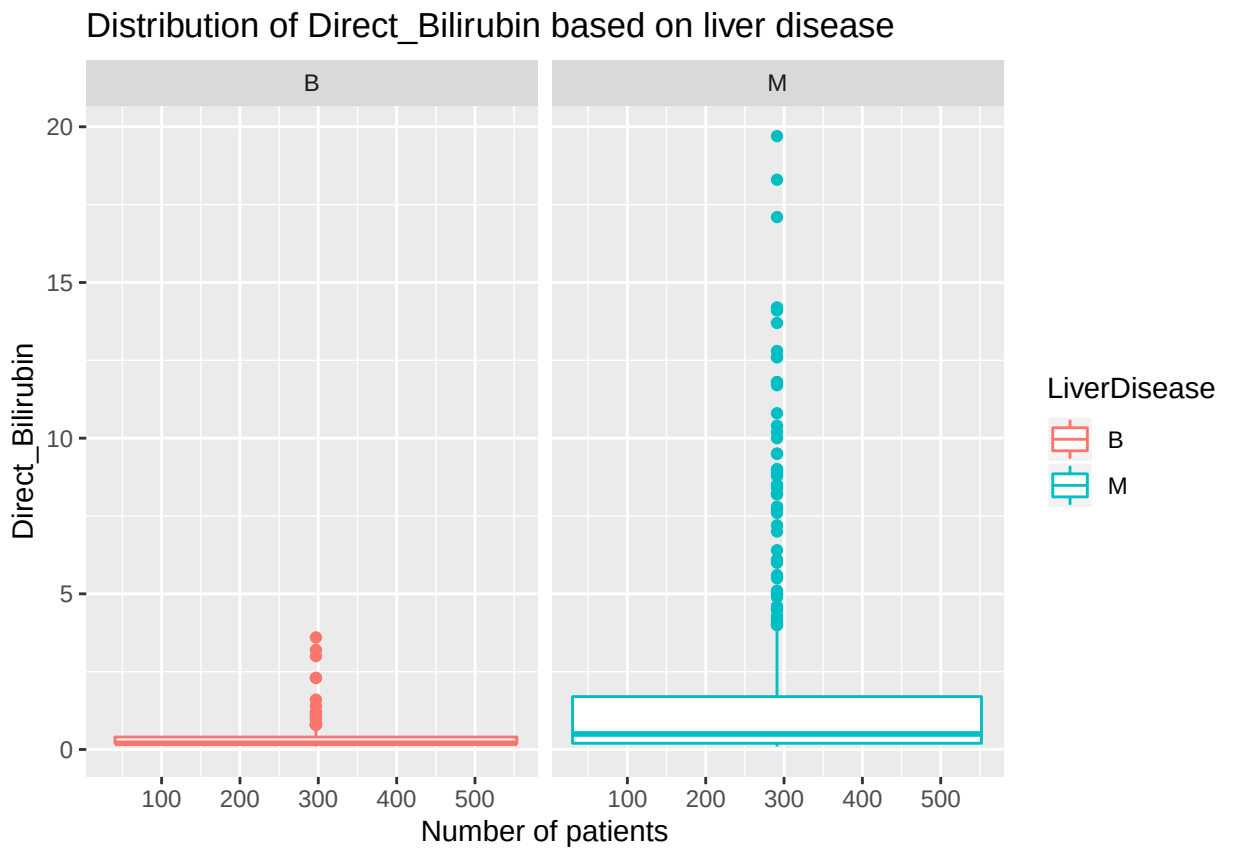
```
## Female   Male
##    125    398
```

\subsection{Total_Bilirubin and Direct_Bilirubin} Bilirubin refers to any form of a yellowish pigment made in the liver when red blood cells are broken down. The elevated levels indicate that the liver is damaged. We find a similar trend with the variable that Bilirubin levels are high for patients with liver diseases.

```
# Plotting distributions of Total_Bilirubin based on liver diseases
training %>%
  ggplot(aes(as.numeric(row.names(training)),Total_Bilirubin, color=LiverDisease)) +
  geom_boxplot() +
  labs(y="Total_Bilirubin", x = "Number of patients")+
  facet_wrap( ~ LiverDisease) +
  ggtitle("Distribution of Total_Bilirubin based on liver disease")
```




```
# Plotting distributions of Direct_Bilirubin based on liver disease
training %>%
  ggplot(aes(as.numeric(row.names(training)),Direct_Bilirubin, color=LiverDisease)) +
  geom_boxplot() +
  labs(y="Direct_Bilirubin", x = "Number of patients")+
  facet_wrap( ~ LiverDisease) +
  ggtitle("Distribution of Direct_Bilirubin based on liver disease")
```



The correlations show that both bilirubins are weakly correlated with liver disease. However, both bilirubins are highly correlated with each other.

```
# Making a subset of data
subset_train <- training[c("Total_Bilirubin","Direct_Bilirubin","LiverDisease")]
# Converting disease variable to numeric format
subset_train <- transform(subset_train, LiverDisease= ifelse(subset_train$LiverDisease=="M", 1,0))
# Looking at the coorelations
cor(subset_train)
```

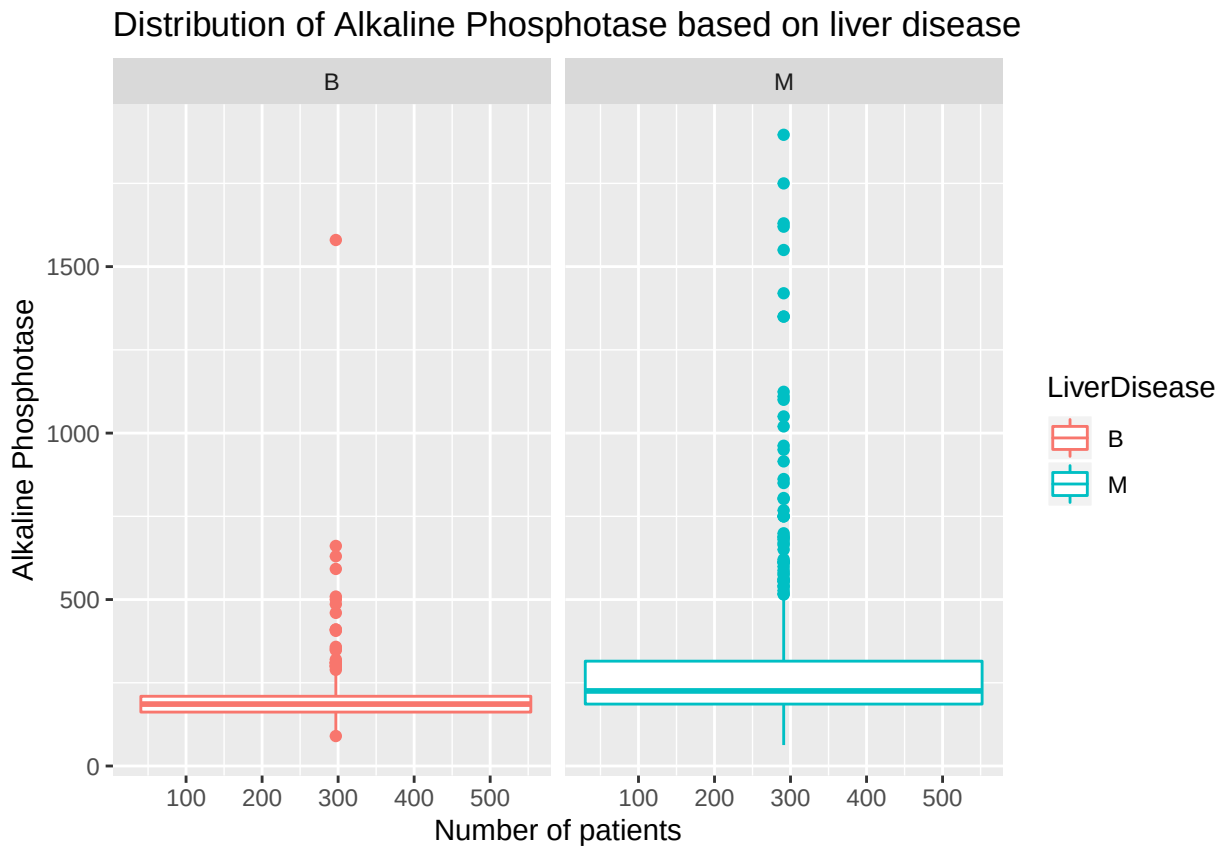
```
##                 Total_Bilirubin Direct_Bilirubin LiverDisease
## Total_Bilirubin       1.0000000        0.8584292    0.2065553
## Direct_Bilirubin      0.8584292        1.0000000    0.2347388
## LiverDisease          0.2065553        0.2347388    1.0000000
```

4.3 Alkaline Phosphotase

Alkaline phosphatase (ALP) is an enzyme in a person's blood that helps break down proteins. We notice that levels of ALP are comparatively high for patients with liver diseases.

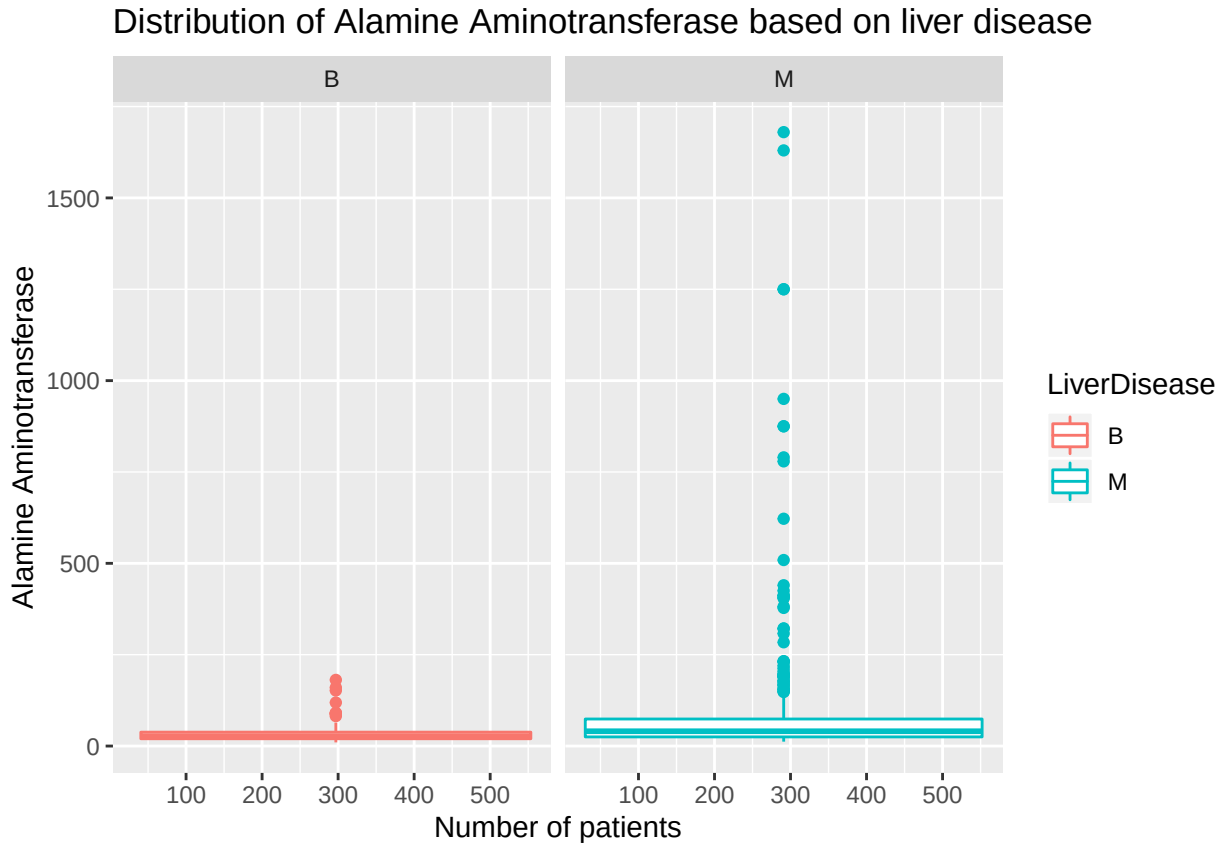
```
# Plotting distributions of Alkaline Phosphotase based on liver diseases
training %>%
  ggplot(aes(as.numeric(row.names(training)),Alkaline_Phosphotase, color=LiverDisease)) +
  geom_boxplot() +
  labs(y="Alkaline Phosphotase", x = "Number of patients")+
```

```
  facet_wrap( ~ LiverDisease) +
  ggtitle("Distribution of Alkaline Phosphotase based on liver disease")
```

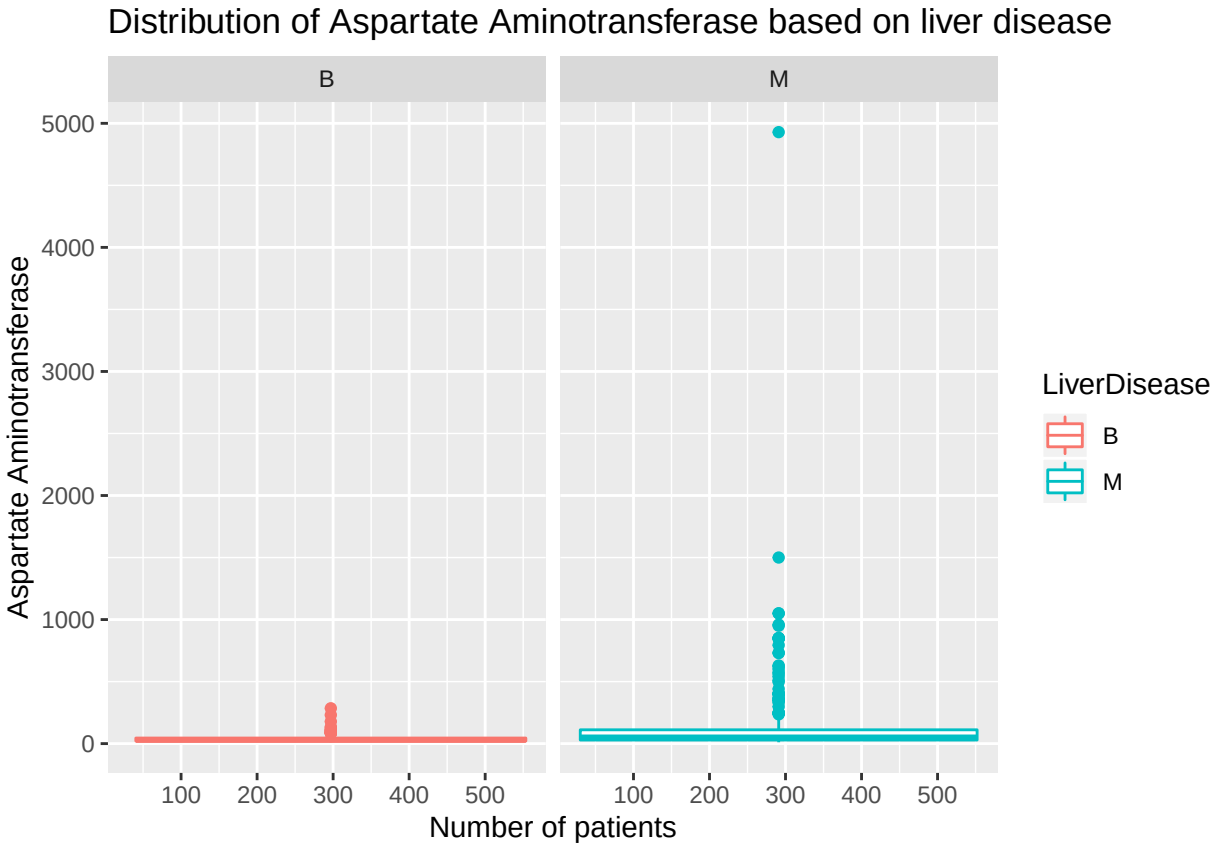


\subsection{Alamine_Aminotransferase and Aspartate_Aminotransferase} Aminotransferases are enzymes that are important in the synthesis of amino acids, which form proteins. Alanine aminotransferase (ALT) and Aspartate aminotransferase (AST) are found primarily in the liver and kidney. High levels of ALT and AST are expected for patients with liver diseases. We also observe the slightly elevated levels of these enzymes for patients with liver diseases.

```
# Plotting distributions of Alamine Aminotransferase based on liver diseases
training %>%
  ggplot(aes(as.numeric(row.names(training)),Alamine_Aminotransferase, color=LiverDisease)) +
  geom_boxplot() +
  labs(y="Alamine Aminotransferase", x = "Number of patients")+
  facet_wrap( ~ LiverDisease) +
  ggtitle("Distribution of Alamine Aminotransferase based on liver disease")
```




```
# Plotting distributions of Aspartate_Aminotransferase based on liver diseases
training %>%
  ggplot(aes(as.numeric(row.names(training)),Aspartate_Aminotransferase, color=LiverDisease)) +
  geom_boxplot() +
  labs(y="Aspartate Aminotransferase", x = "Number of patients")+
  facet_wrap( ~ LiverDisease) +
  ggtitle("Distribution of Aspartate Aminotransferase based on liver disease")
```



Contrary to bilirubins, there exists a weak correlation between both aminotransferases.

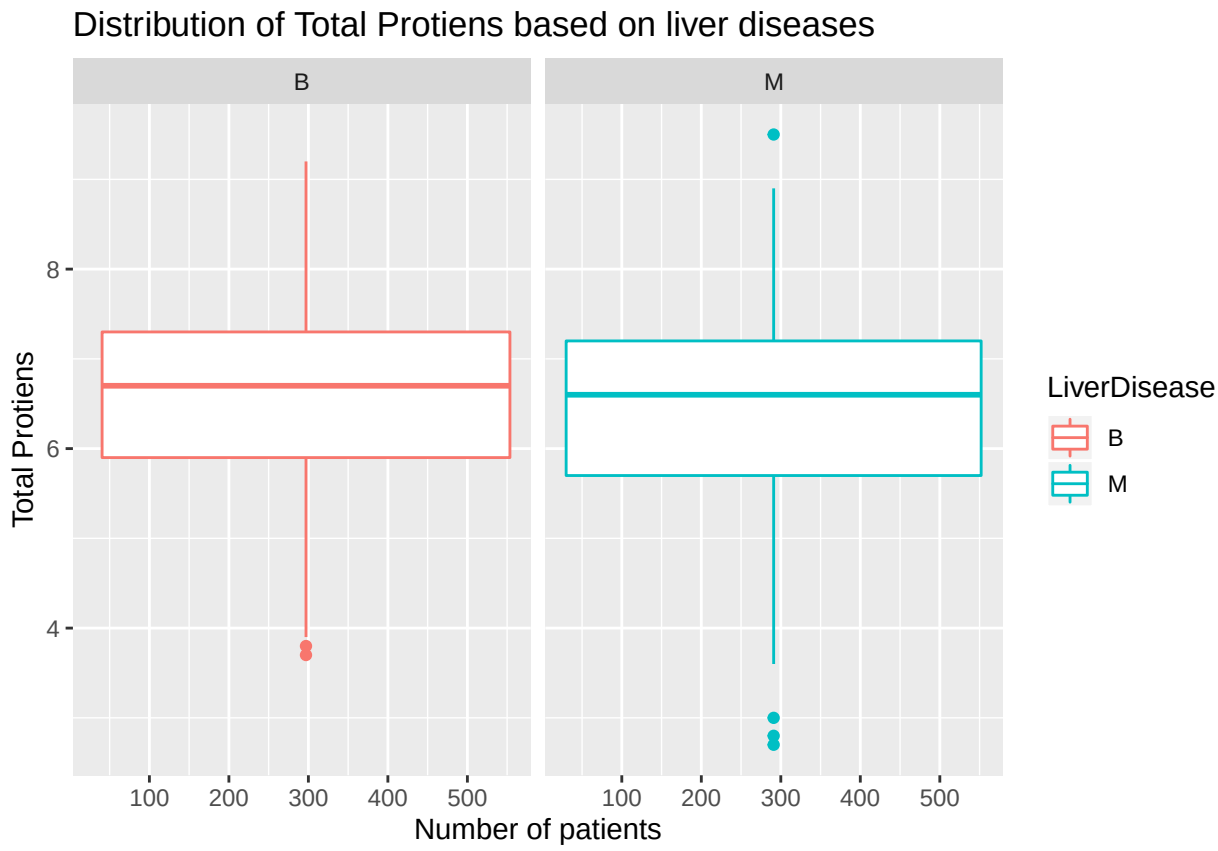
```
# Making a subset of data
subset_train <- training[c("Alkaline_Phosphotase","Aspartate_Aminotransferase","LiverDisease")]

# Converting disease variable to numeric format
subset_train <- transform(subset_train, LiverDisease= ifelse(subset_train$LiverDisease=="M", 1,0))

# Looking at the coorelations
cor(subset_train)
```

```
##                           Alkaline_Phosphotase Aspartate_Aminotransferase
## Alkaline_Phosphotase                  1.000000                  0.2156400
## Aspartate_Aminotransferase            0.215640                  1.0000000
## LiverDisease                          0.178212                  0.1488358
##                           LiverDisease
## Alkaline_Phosphotase         0.1782120
## Aspartate_Aminotransferase   0.1488358
## LiverDisease                 1.0000000
```

4.4 Total Protiens

The total protein test measures the total amount of protein in your body. The distributions indicate that we can diagnose liver disease reliably using this variable.

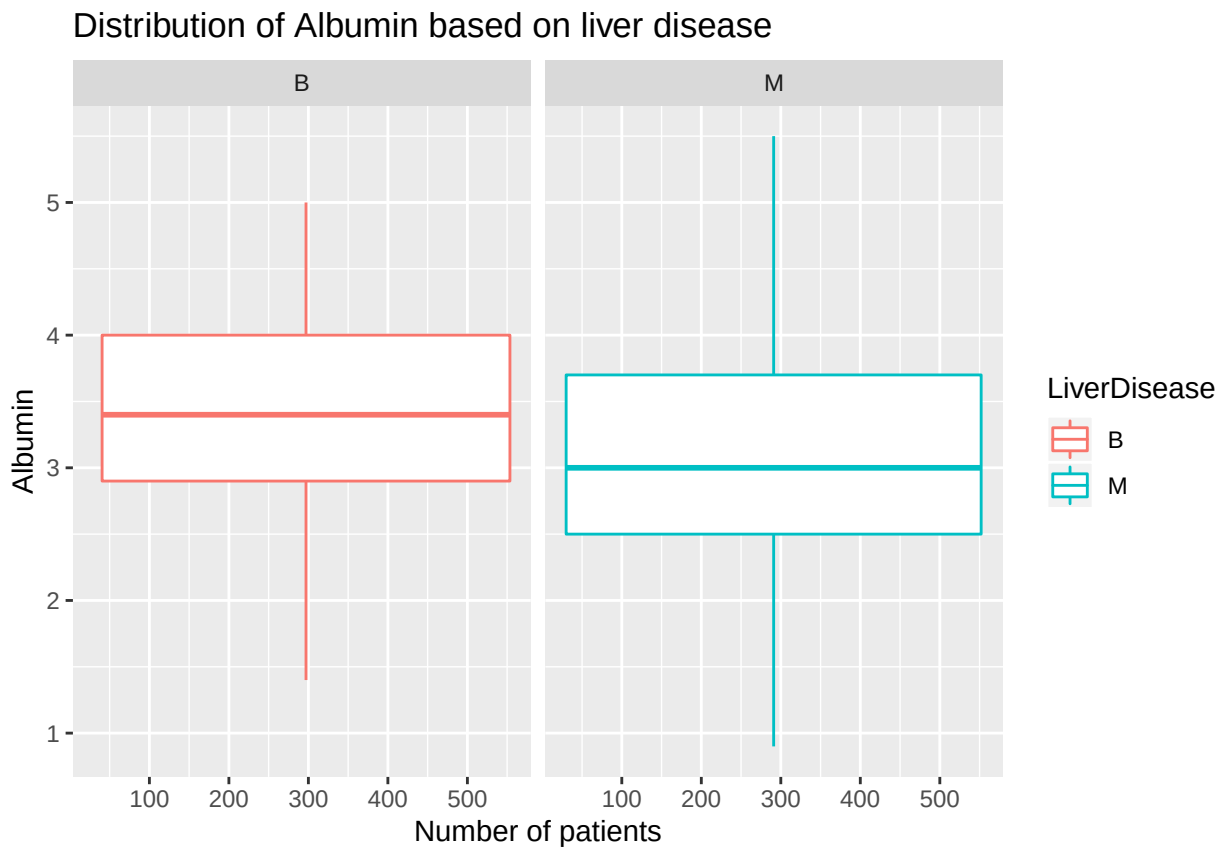
```r
# Plotting distributions of Total Protiens based on liver diseases
training %>%
  ggplot(aes(as.numeric(row.names(training)),Total_Protiens, color=LiverDisease)) +
  geom_boxplot() +
  labs(y="Total Protiens", x = "Number of patients")+
  facet_wrap( ~ LiverDisease) +
  ggtitle("Distribution of Total Protiens based on liver diseases")
```



4.5 Albumin

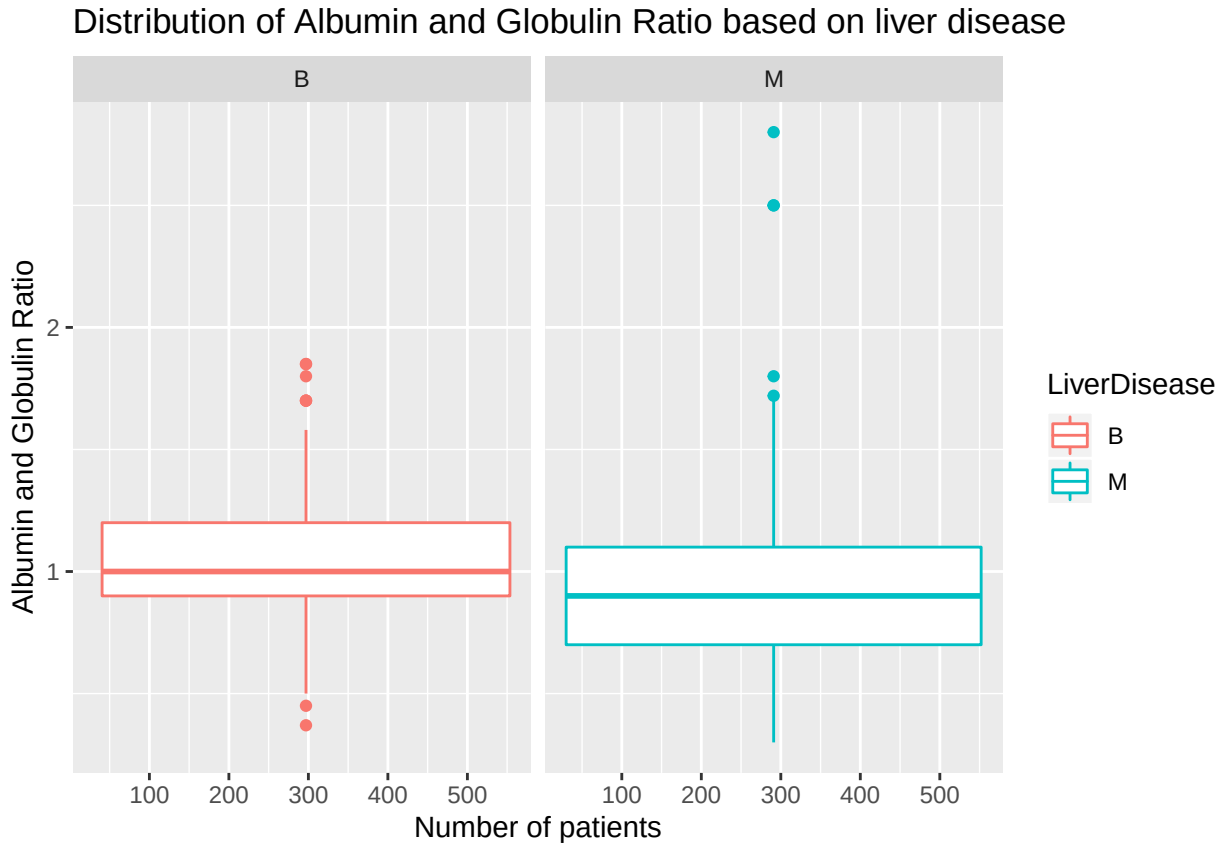
Albumin is a protein made by the liver to keep fluid in the bloodstream. The low levels of albumin often indicate a problem with the liver, and we notice a similar trend with this variable.

```r
# Plotting distributions of Albumin based on liver diseases
training %>%
  ggplot(aes(as.numeric(row.names(training)),Albumin, color=LiverDisease)) +
  geom_boxplot() +
  labs(y="Albumin", x = "Number of patients")+
  facet_wrap( ~ LiverDisease) +
  ggtitle("Distribution of Albumin based on liver disease")
```



\subsection{Albumin_and_Globulin_Ratio} These proteins are crucial for body growth, development, and health. They form the structural part of most organs and makeup enzymes and hormones that regulate body functions. The low ratios refer to liver issues, and we can notice the same from distributions graphs.

```
# Plotting distributions of Albumin based on liver diseases
training %>%
  ggplot(aes(as.numeric(row.names(training)),Albumin_and_Globulin_Ratio, color=LiverDisease)) +
  geom_boxplot() +
  labs(y="Albumin and Globulin Ratio", x = "Number of patients")+
  facet_wrap( ~ LiverDisease) +
  ggtitle("Distribution of Albumin and Globulin Ratio based on liver disease")
```



5 Methods

Based on the discussion in Section 4, we will not use "Age" and "Total Protein" variables to train the machine learning models. We remove these variables from both training and validation datasets.

```
# Deleting the column 'Dataset' as no longer required
training<-within(training, rm(Age,Total_Protiens))
validation<-within(validation, rm(Age,Total_Protiens))
```

5.1 Logistic Regression

We use logistic regression with cross-validation of 10 folds to train the model.

```
# Defining a cross-validation (10 K folds )
control <- trainControl(method = "cv", number =10)

# Train logistic regression model
train_glm <- train(LiverDisease ~.,
                   method = "glm",
                   data = training,
                   trControl=control)
```

```
## 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
```

```
sprintf("The accuracy of GLM = %f",train_glm$results$Accuracy)
```

```
## [1] "The accuracy of GLM = 0.703512"
```

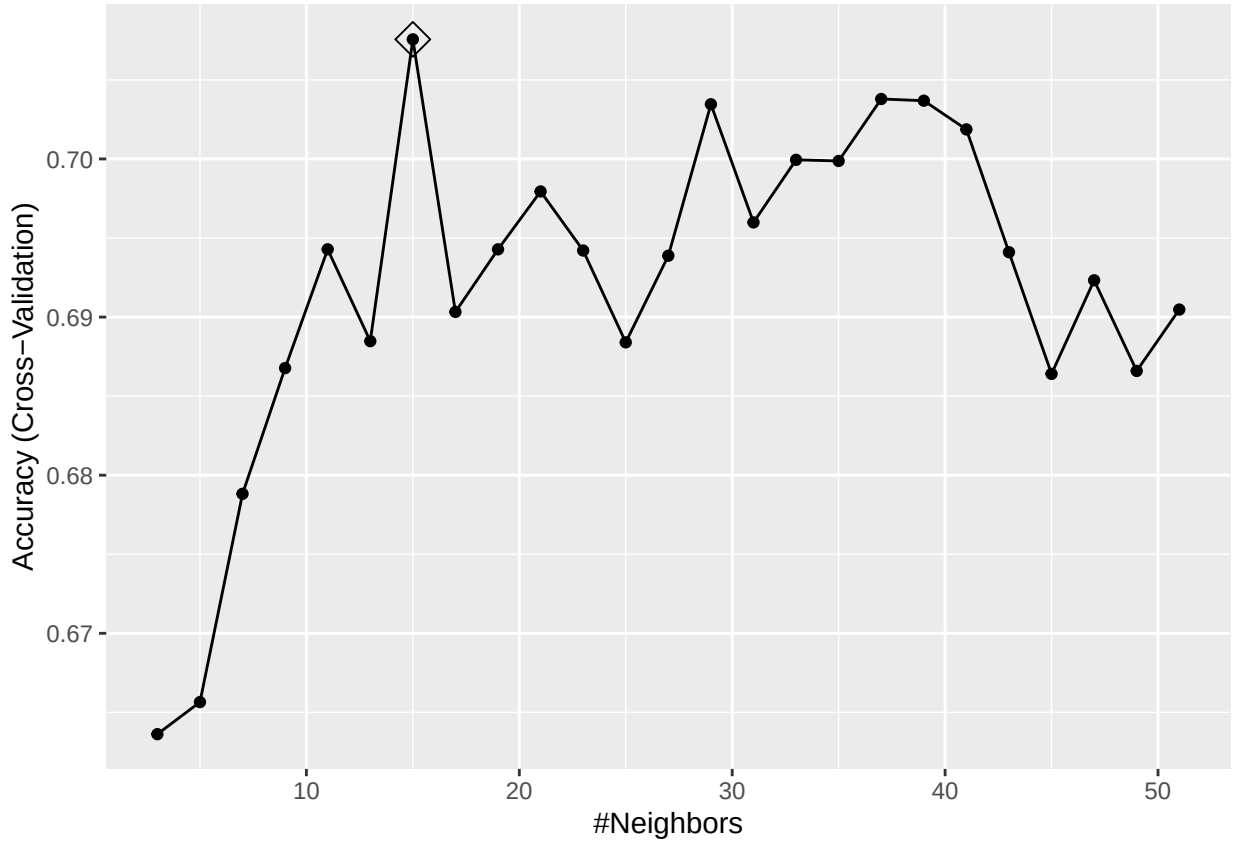
```
model_results <- data_frame(method = "glm", Accuracy = train_glm$results$Accuracy)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

5.2 K-nearest neigbors (knn)

We use knn with cross-validation of 10 folds to train the model.

```
# Defining a cross validation (10 K folds )
control <- trainControl(method = "cv", number = 10)
# Train knn model
train_knn <- train(LiverDisease~ .,
                   method = "knn",
                   data = training,
                   tuneGrid = data.frame(k = seq(3, 51, 2)),
                   trControl = control)

ggplot(train_knn, highlight = TRUE)
```




```
sprintf("The best accuracy = %f",max(train_knn$results$Accuracy))
```

```
## [1] "The best accuracy = 0.707565"
```

```
# Stroing the results
model_results <- bind_rows(model_results,data_frame(method="knn",
                                      RMSE = max(train_knn$results$Accuracy) ))
```

5.3 Partial Least Squares (PLS)

We use PLS with cross-validation of 10 folds to train the model.

```
set.seed(10)
# Defining a cross validation (10 K folds )
control <- trainControl(method = "cv", number = 10)
# Train the model
train_pls <- train(LiverDisease~.,
                   method = "pls",
                   data = training,
                   preProc = c("center", "scale"),
                   tuneLength = 15,
                   trControl = control)

sprintf("The accuracy of PLS  = %f",max(train_pls$results$Accuracy))
```

```
## [1] "The accuracy of PLS  = 0.720860"
```

```
# Stroing the results
model_results <- bind_rows(model_results,data_frame(method="pls",
                                     RMSE = max(train_pls$results$Accuracy) ))
```

5.4 Linear Discriminant Analysis (LDA)

The LDA is a statistical classifier, and we use it with a cross-validation of 10 folds for training.

```
# Defining a cross validation (10 K folds )
control <- trainControl(method = "cv", number=10)
# Train the model
train_lda <- train(LiverDisease~.,
                   method = "lda",
                   data = training,
                   trControl = control)

sprintf("The accuracy of lda = %f",max(train_lda$results$Accuracy))
```

```
## [1] "The accuracy of lda = 0.717124"
```

```
# Stroing the results
model_results <- bind_rows(model_results,data_frame(method="lda",
                                     RMSE = max(train_lda$results$Accuracy) ))
```

5.5 Quadratic Discriminant Analysis (QDA)

The QDA is a statistical classifier, and we use it with a cross-validation of 10 folds for training.

```
# Defining a cross validation (10 K folds )
control <- trainControl(method = "cv", number=10)
# Train the model
train_qda <- train(LiverDisease~.,
                   method = "qda",
                   data = training,
                   trControl = control)

sprintf("The accuracy of qda = %f",max(train_qda$results$Accuracy))
```

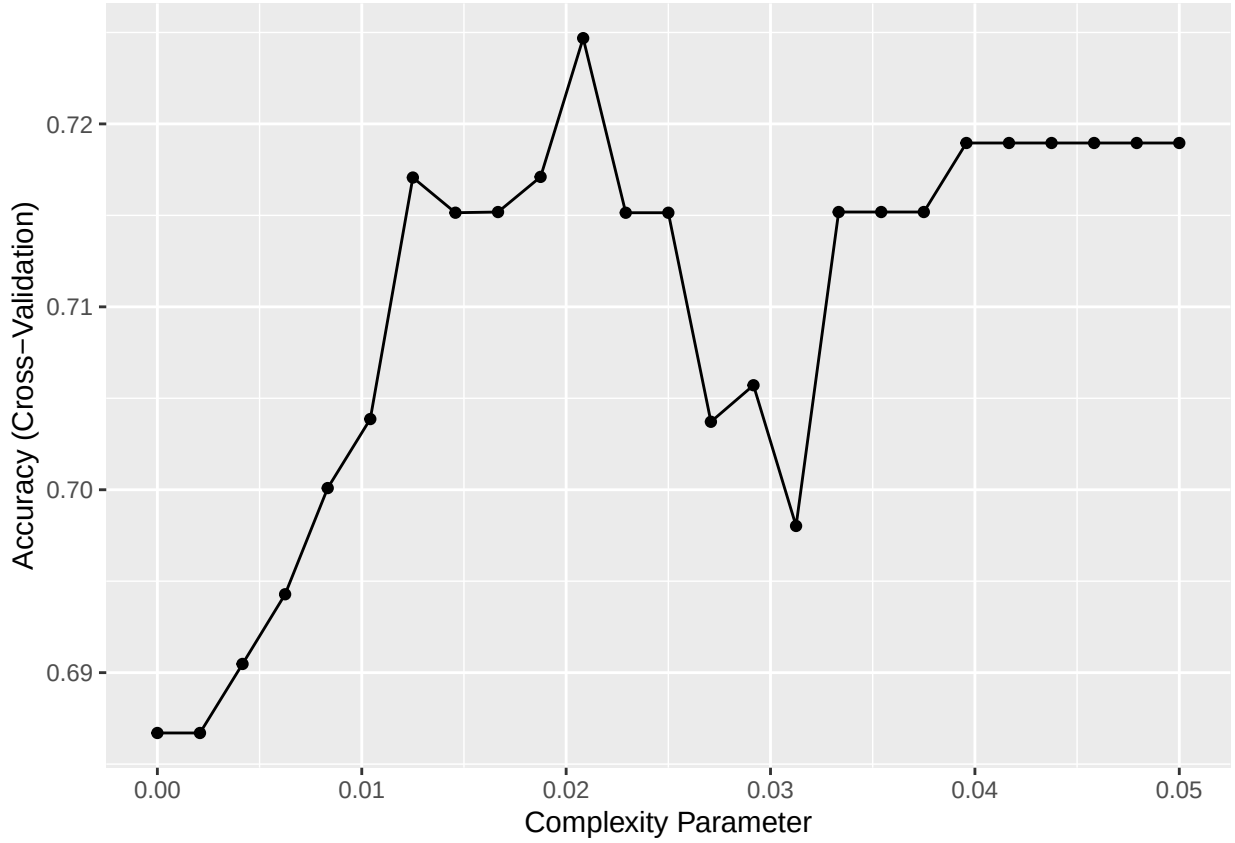
```
## [1] "The accuracy of qda = 0.539227"
```

```
# Stroing the results
model_results <- bind_rows(model_results,data_frame(method="qda",
                                     RMSE = max(train_qda$results$Accuracy) ))
```

5.6 Decision Tress

We use decision trees with a cross-validation of 10 folds for training.

```
# Defining a cross validation (10 K folds )
control <- trainControl(method = "cv", number = 10)
# Train the model
train_rpart <- train(LiverDisease~.,
                     method = "rpart",
                     data = training,
                     trControl = control,
                     tuneGrid = data.frame(cp = seq(0, 0.05, len = 25)))
ggplot(train_rpart)
```




```
sprintf("The accuracy of decision tree = %f",max(train_rpart$results$Accuracy))
```

```
## [1] "The accuracy of decision tree = 0.724688"
```

```
# Stroing the results
model_results <- bind_rows(model_results,data_frame(method="rpart",
                                    RMSE = max(train_rpart$results$Accuracy) ))
```

5.7 Random Forests

We use random forests with a cross-validation of 10 folds for training.

```r
# Defining a cross validation (10 K folds )
control <- trainControl(method = "cv", number = 10)
# Train the model
train_rf <- train(LiverDisease~.,
                      method = "rf",
                      data = training,
                      trControl = control,
                      tuneGrid = data.frame(mtry=seq(1,7)),
                      ntree=100)
sprintf("The accuracy of forest tree = %f",max(train_rf$results$Accuracy))
```

```
## [1] "The accuracy of forest tree = 0.728628"
```

```r
# Stroing the results
model_results <- bind_rows(model_results,data_frame(method="rf",
                                      RMSE = max(train_rf$results$Accuracy) ))
```

5.8 Support Vector Machine

We use support vector machine with a cross-validation of 10 folds for training.

```r
# Defining a cross validation (10 K folds )
control <- trainControl(method = "cv", number = 10)
# Train the model
train_svm <- train(LiverDisease~.,
                     data = training,
                     method = "svmLinear",
                     trControl = control)

sprintf("The accuracy of svm = %f",max(train_svm$results$Accuracy))
```

```
## [1] "The accuracy of svm = 0.718973"
```

```r
# Stroing the results
model_results <- bind_rows(model_results,data_frame(method="svm",
                                      RMSE = max(train_svm$results$Accuracy) ))
```

5.9 Adaptive Boosting (Adaboost)

AdaBoost is a machine learning meta-algorithm for classification. We train the model with a cross-validation of 10 folds.

```r
# Defining a cross validation (10 K folds )
control <- trainControl(method = "cv", number = 10)
# Train the model
train_ada <- train(LiverDisease~.,
                     data = training,
                     method = "ada",
                     trControl = control)

sprintf("The accuracy of adaboost = %f",max(train_ada$results$Accuracy))
```

```
## [1] "The accuracy of adaboost = 0.724673"
```

```
# Stroing the results
model_results <- bind_rows(model_results,data_frame(method="ada",
                                     RMSE = max(train_ada$results$Accuracy) ))
```

The reported accuracies for all models across training dataset are shown in the following table. The results show that lda model performs the worse. All other models provide an accuracy of around 0.70.The random forest seems to perform the best.

```
model_results
```

method	Accuracy	RMSE
glm	0.7035118	NA
knn	NA	0.7075650
pls	NA	0.7208600
lda	NA	0.7171241
qda	NA	0.5392271
rpart	NA	0.7246884
rf	NA	0.7286284
svm	NA	0.7189732
ada	NA	0.7246734

6 Results

References

[1] Ethan Du-Crowa, Lucy Warrenb, Susan M Astleya and Johan Hullemanc,"Is there a safety-net effect with Computer-Aided Detection (CAD)?", Medical Imaging 2019.

[2] Eugene, R., Sorrell, Michael F.; Maddrey, Willis C., "Schiff's Diseases of the Liver", 10th Edition, Lippincott Williams & Wilkins by Schiff.

[3] Bendi, Venkata . R, M. S. Prasad Babu, and N. B. Venkateswarlu, "Critical Comparative Study of Liver Patients from USA and INDIA: An Exploratory Analysis", International Journal of Computer Science Issues, May 2012.