Liver Patient

Shuji Hachisu

20/02/2022

Executive Summary

This report explores liver patients in India. The liver patients will be analysed with respect to sex, age and various protein readings taken from the blood. Machine learning (ML) models to predict whether a given patient has a liver disease or not were constructed. The best performing model used the k-nearest neighbor algorithm on principal components and achieved: 62% precision, 47% sensitivity, 88% specificity and 61% balanced accuracy.

Method and Analysis

In this project, a disease prediction system will be created by applying ML principles to publicly available patient records. The full data set can be found here: https://www.kaggle.com/uciml/indian-liver-patient-records.

The data set has the following fields:

- Age of the patient (n.b. patient whose age is > 89 is listed as being 90)
- Gender of the patient
- Total Bilirubin
- Direct Bilirubin
- Alkaline Phosphotase
- Alamine Aminotransferase
- Aspartate Aminotransferase
- Total Protiens
- Albumin
- Albumin and Globulin Ratio
- Dataset: field used to split the data into two sets (patient with liver disease is assigned the value 2, and no disease is assigned the value 1)

Create training and validation sets

Liver patient data set will be downloaded from the web, and the files will be read and loaded into R as a data frame. The data frame will be given appropriate column names, and some field will be further processed (e.g. Dataset column is modified so no disease is represented by 0, and presence of disease is represented by 1). The full data set will be split into training and test sets to build the ML model and validate the ML model respectively.

```
##############################################################################
# Create training set, validation set (final hold-out test set)
##############################################################################


# Note: this process could take a couple of minutes

# install packages if needed

# setup libraries

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

# load libraries

library(tidyverse)
library(caret)
library(data.table)
library(corrplot)

# download data, create dataframe, name columns, create factors, and omit rows with blank values

dl <- tempfile()
download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/00225/Indian%20Liver%20Patient%

# read csv file and store in liverPatient

liverPatient <- read.csv(file = dl)

# set column names

colnames(liverPatient) <- c("Age", "Gender", "Total_Bilirubin", "Direct_Bilirubin",
                      "Alkaline_Phosphotase", "Alamine_Aminotransferase", "Aspartate_Aminotransferase",
                      "Total_Protiens", "Albumin", "Albumin_and_Globulin_Ratio", "Dataset")

# covert DataSet columns so that healthy patients are zero and sick patients are 1.
# Convert gendar into factor and drop rows with empty fields

liverPatient <-
  liverPatient %>%
  mutate(Dataset = factor(Dataset - 1),
         Gender = factor(Gender)) %>%
  na.omit()


# create both training set and validation set
# Validation set will be 10% of data set

set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = liverPatient$Dataset, times = 1, p = 0.1, list = FALSE)
```

```
training_set <- liverPatient[-test_index,]
test_set <- liverPatient[test_index,]

# remove unnecessary variable and data sets
rm(dl, test_index, liverPatient)
```

Explore and analyse the data set and its features

The data structure of the training set table, the first 6 rows, and their corresponding summary statistics are shown to gain insights into training sets.

```
# show first 6 rows of the training set to get insight into the table and data structure
head(training_set)
```

```
##   Age Gender Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 1  62   Male            10.9              5.5                  699
## 2  62   Male             7.3              4.1                  490
## 3  58   Male             1.0              0.4                  182
## 4  72   Male             3.9              2.0                  195
## 5  46   Male             1.8              0.7                  208
## 7  29 Female             0.9              0.3                  202
##   Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens Albumin
## 1                       64                        100            7.5     3.2
## 2                       60                         68            7.0     3.3
## 3                       14                         20            6.8     3.4
## 4                       27                         59            7.3     2.4
## 5                       19                         14            7.6     4.4
## 7                       14                         11            6.7     3.6
##   Albumin_and_Globulin_Ratio Dataset
## 1                       0.74       0
## 2                       0.89       0
## 3                       1.00       0
## 4                       0.40       0
## 5                       1.30       0
## 7                       1.10       0
```

```
# show data structure, column names and number of records
str(training_set)
```

```
## 'data.frame':    519 obs. of  11 variables:
##  $ Age                       : int  62 62 58 72 46 29 17 55 57 72 ...
##  $ Gender                    : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 1 2 2 2 2 ...
##  $ Total_Bilirubin           : num  10.9 7.3 1 3.9 1.8 0.9 0.9 0.7 0.6 2.7 ...
##  $ Direct_Bilirubin          : num  5.5 4.1 0.4 2 0.7 0.3 0.3 0.2 0.1 1.3 ...
##  $ Alkaline_Phosphotase      : int  699 490 182 195 208 202 202 290 210 260 ...
##  $ Alamine_Aminotransferase  : int  64 60 14 27 19 14 22 53 51 31 ...
##  $ Aspartate_Aminotransferase: int  100 68 20 59 14 11 19 58 59 56 ...
##  $ Total_Protiens            : num  7.5 7 6.8 7.3 7.6 6.7 7.4 6.8 5.9 7.4 ...
##  $ Albumin                   : num  3.2 3.3 3.4 2.4 4.4 3.6 4.1 3.4 2.7 3 ...
##  $ Albumin_and_Globulin_Ratio: num  0.74 0.89 1 0.4 1.3 1.1 1.2 1 0.8 0.6 ...
##  $ Dataset                   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...
```

```
##  - attr(*, "na.action")= 'omit' Named int [1:4] 209 241 253 312
##   ..- attr(*, "names")= chr [1:4] "209" "241" "253" "312"
```

```
# show summary statistics of training set
summary(training_set)
```

```
##       Age           Gender    Total_Bilirubin  Direct_Bilirubin
##  Min.   : 4.00   Female:123   Min.   : 0.500   Min.   : 0.100
##  1st Qu.:33.00   Male  :396   1st Qu.: 0.800   1st Qu.: 0.200
##  Median :45.00                Median : 1.000   Median : 0.300
##  Mean   :45.09                Mean   : 3.301   Mean   : 1.482
##  3rd Qu.:58.00                3rd Qu.: 2.600   3rd Qu.: 1.300
##  Max.   :90.00                Max.   :75.000   Max.   :18.300
##  Alkaline_Phosphotase Alamine_Aminotransferase Aspartate_Aminotransferase
##  Min.   :  63.0       Min.   :  10.00          Min.   :  10.0
##  1st Qu.: 178.0       1st Qu.:  23.00          1st Qu.:  26.0
##  Median : 209.0       Median :  35.00          Median :  42.0
##  Mean   : 295.8       Mean   :  76.64          Mean   : 106.8
##  3rd Qu.: 298.0       3rd Qu.:  60.00          3rd Qu.:  86.5
##  Max.   :2110.0       Max.   :2000.00          Max.   :4929.0
##  Total_Protiens     Albumin      Albumin_and_Globulin_Ratio Dataset
##  Min.   :2.700   Min.   :0.900   Min.   :0.3000             0:371
##  1st Qu.:5.800   1st Qu.:2.600   1st Qu.:0.7000             1:148
##  Median :6.500   Median :3.100   Median :0.9300
##  Mean   :6.481   Mean   :3.132   Mean   :0.9446
##  3rd Qu.:7.200   3rd Qu.:3.700   3rd Qu.:1.1000
##  Max.   :9.600   Max.   :5.500   Max.   :2.8000
```

```
# training set data set has 519 records and 11 fields
```

The data structure of the test set table, the first 6 rows, and their corresponding summary statistics are shown to gain insights into test sets.

```
# show first 6 rows of the test set to get insight into the table and data structure
head(test_set)
```

```
##    Age Gender Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 6   26 Female             0.9              0.2                  154
## 17  33   Male             1.6              0.5                  165
## 35  30   Male             1.3              0.4                  482
## 36  17 Female             0.7              0.2                  145
## 56  33   Male             0.8              0.2                  198
## 58  51   Male             0.8              0.2                  367
##    Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens Albumin
## 6                        16                          12            7.0     3.5
## 17                       15                          23            7.3     3.5
## 35                      102                          80            6.9     3.3
## 36                       18                          36            7.2     3.9
## 56                       26                          23            8.0     4.0
## 58                       42                          18            5.2     2.0
##    Albumin_and_Globulin_Ratio Dataset
## 6                        1.00       0
```

```
## 17                      0.92       1
## 35                      0.90       0
## 36                      1.18       1
## 56                      1.00       1
## 58                      0.60       0
```

```
# show data structure, column names and number of records
str(test_set)
```

```
## 'data.frame':    59 obs. of  11 variables:
##  $ Age                       : int  26 33 30 17 33 51 37 60 32 32 ...
##  $ Gender                    : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 2 2 2 2 ...
##  $ Total_Bilirubin           : num  0.9 1.6 1.3 0.7 0.8 0.8 1.8 5.2 15.9 18 ...
##  $ Direct_Bilirubin          : num  0.2 0.5 0.4 0.2 0.2 0.2 0.8 2.4 7 8.2 ...
##  $ Alkaline_Phosphotase      : int  154 165 482 145 198 367 215 168 280 298 ...
##  $ Alamine_Aminotransferase  : int  16 15 102 18 26 42 53 126 1350 1250 ...
##  $ Aspartate_Aminotransferase: int  12 23 80 36 23 18 58 202 1600 1050 ...
##  $ Total_Protiens            : num  7 7.3 6.9 7.2 8 5.2 6.4 6.8 5.6 5.4 ...
##  $ Albumin                   : num  3.5 3.5 3.3 3.9 4 2 3.8 2.9 2.8 2.6 ...
##  $ Albumin_and_Globulin_Ratio: num  1 0.92 0.9 1.18 1 0.6 1.4 0.7 1 0.9 ...
##  $ Dataset                   : Factor w/ 2 levels "0","1": 1 2 1 2 2 1 1 1 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:4] 209 241 253 312
##   ..- attr(*, "names")= chr [1:4] "209" "241" "253" "312"
```

```
# show summary statistics of test set
summary(test_set)
```

```
##       Age           Gender   Total_Bilirubin  Direct_Bilirubin
##  Min.   : 4.00   Female:16   Min.   : 0.400   Min.   : 0.10
##  1st Qu.:30.50   Male  :43   1st Qu.: 0.800   1st Qu.: 0.20
##  Median :42.00               Median : 0.900   Median : 0.20
##  Mean   :41.69               Mean   : 3.488   Mean   : 1.62
##  3rd Qu.:52.00               3rd Qu.: 2.650   3rd Qu.: 1.30
##  Max.   :75.00               Max.   :42.800   Max.   :19.70
##  Alkaline_Phosphotase Alamine_Aminotransferase Aspartate_Aminotransferase
##  Min.   : 100.0       Min.   :  10.0           Min.   :  12.0
##  1st Qu.: 162.0       1st Qu.:  24.0           1st Qu.:  23.0
##  Median : 208.0       Median :  40.0           Median :  36.0
##  Mean   : 254.4       Mean   : 121.7           Mean   : 143.8
##  3rd Qu.: 271.5       3rd Qu.:  80.0           3rd Qu.: 116.5
##  Max.   :1550.0       Max.   :1350.0           Max.   :1600.0
##  Total_Protiens     Albumin      Albumin_and_Globulin_Ratio Dataset
##  Min.   :3.800   Min.   :1.400   Min.   :0.3900             0:42
##  1st Qu.:5.600   1st Qu.:2.550   1st Qu.:0.8000             1:17
##  Median :6.700   Median :3.200   Median :1.0000
##  Mean   :6.486   Mean   :3.197   Mean   :0.9695
##  3rd Qu.:7.300   3rd Qu.:4.000   3rd Qu.:1.1000
##  Max.   :9.500   Max.   :4.900   Max.   :2.5000
```

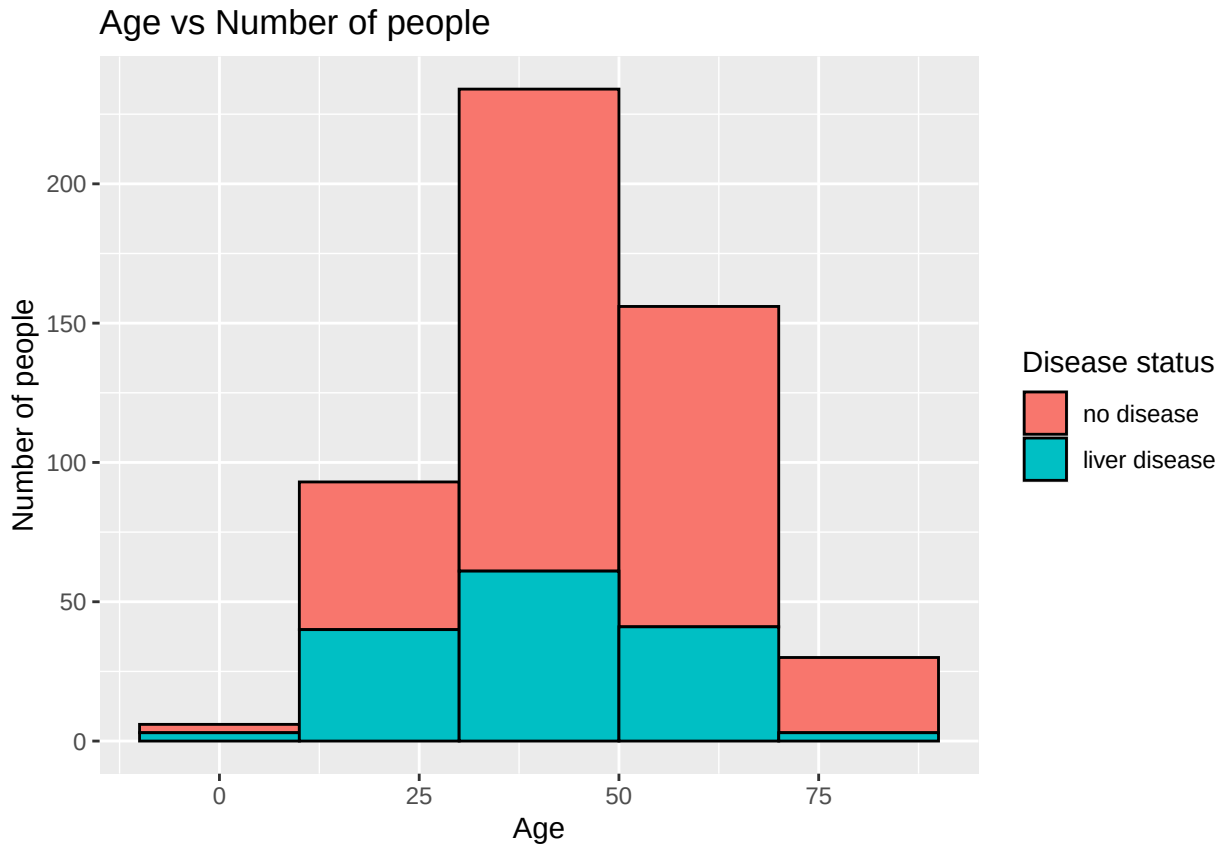
```
# Test set has 59 records and 11 fields.
# This will be used at the final stage to evaluated the model created using the training data set
```

Analyze age

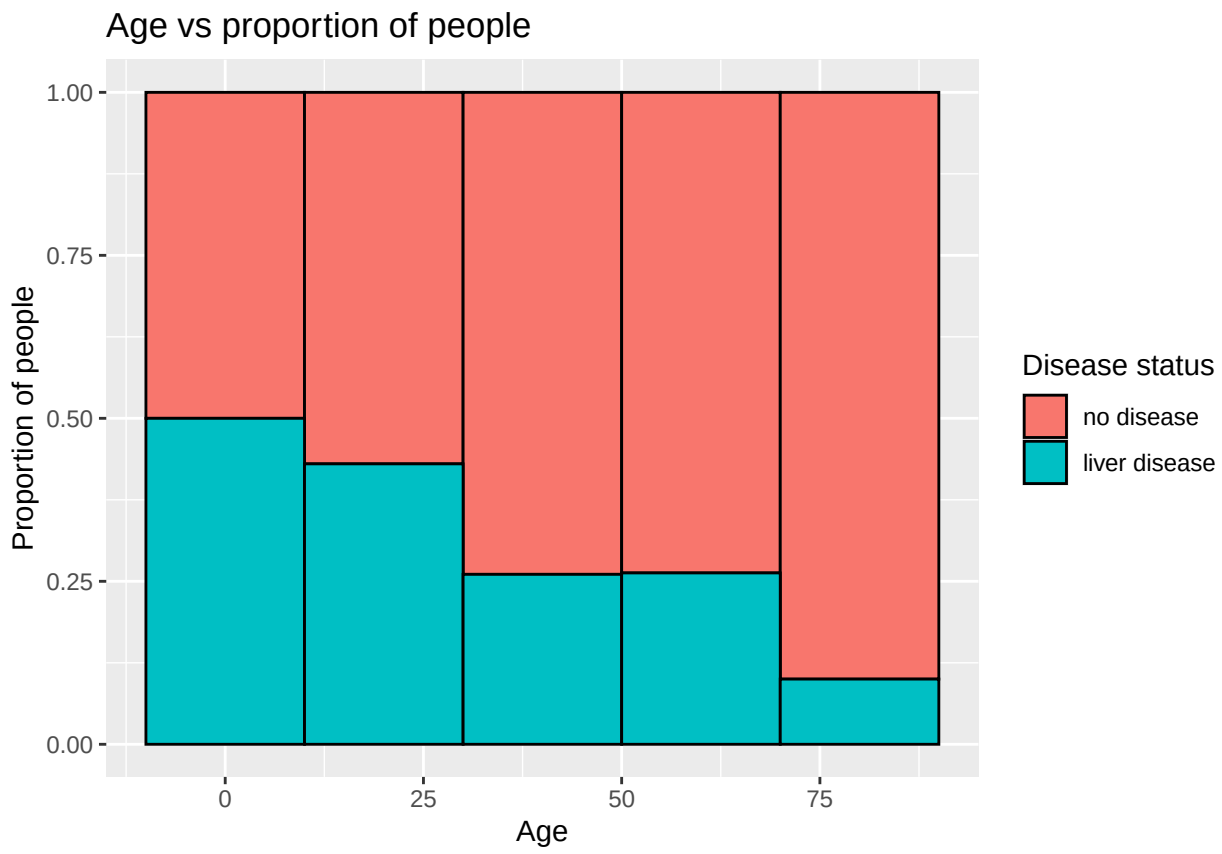
The first histogram shows the age profiles of the patient records.

The second normalized histogram shows that younger patients are more likely to have liver diseases than older patients. This could be because older patients are in hospital providing data for a variety of reasons, whereas youn patients have clear liver disease symptom to be there.

```
######################### age plots ##############################################

# plot a histogram by age for training set colored by Dataset
training_set %>%
  ggplot(aes(Age, fill = Dataset)) +
  geom_histogram(binwidth = 20, color = "black") +
  labs(title = "Age vs Number of people", y = "Number of people") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# plot a normalized histogram by age for training set colored by Dataset
training_set %>%
  ggplot(aes(Age, fill = Dataset)) +
  geom_histogram(binwidth = 20, color = "black", position = "fill") +
  labs(title = "Age vs proportion of people", y = "Proportion of people") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```

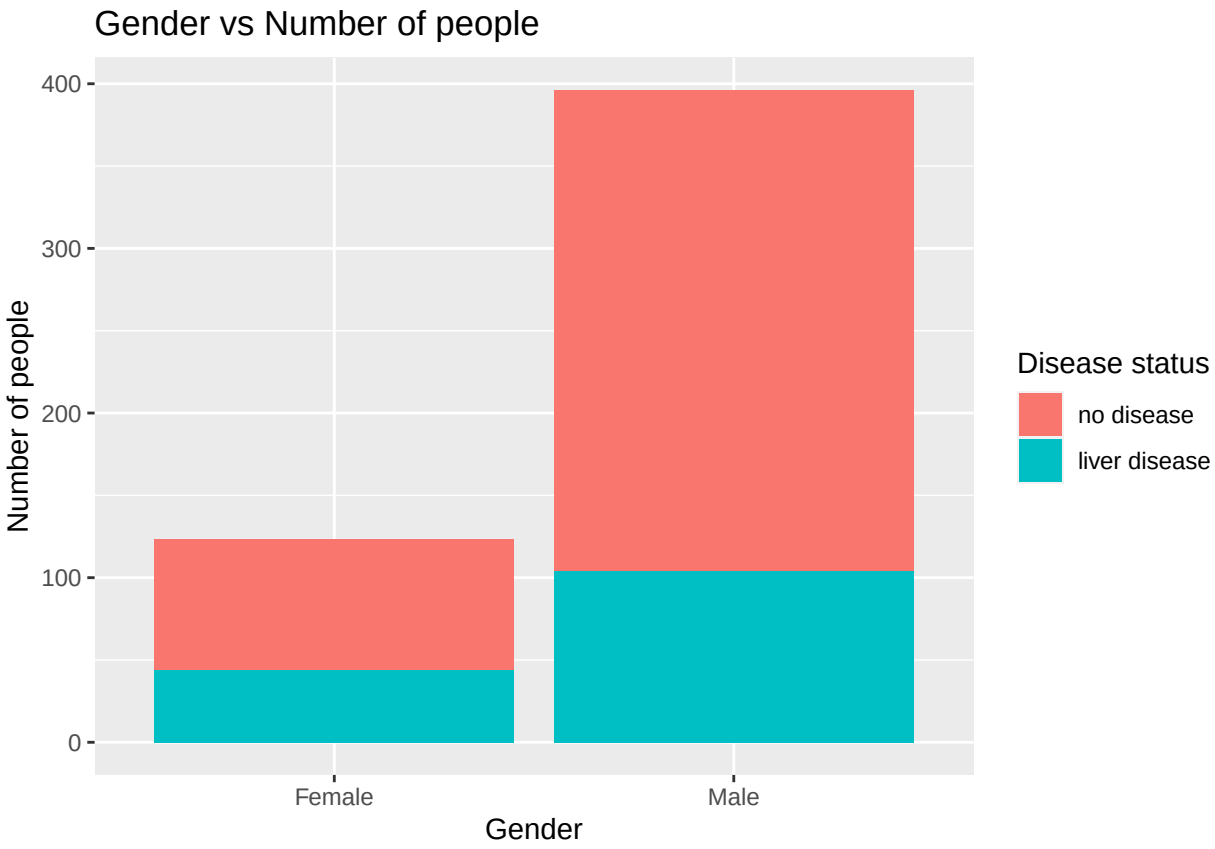


Analyze gender

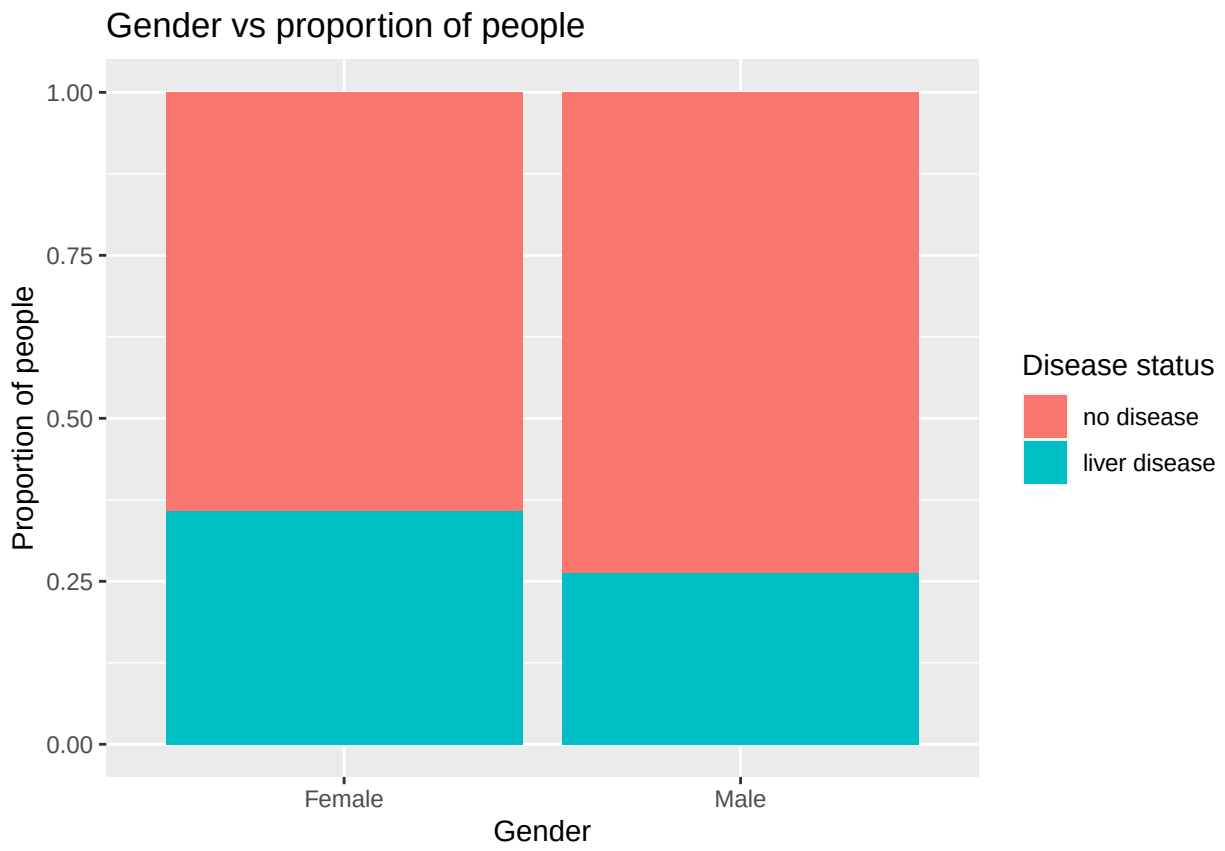
The first histogram shows that most patient records come from men.

The second normalized histogram shows that lady patient are more likely to have liver diseases than men.

```
######################### gender plots ###########################################

# plot a histogram by gender for training set with proportion of Dataset shown
# by color
training_set %>%
  group_by(Dataset, Gender) %>%
  summarize(count = n()) %>%
  ggplot(aes(Gender, count, fill = Dataset)) +
  geom_col() +
  labs(title = "Gender vs Number of people", y = "Number of people") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# plot a normalized histogram by gender for training set with proportion of
# Dataset shown by color
training_set %>%
  group_by(Dataset, Gender) %>%
  summarize(count = n()) %>%
  ggplot(aes(Gender, count, fill = Dataset)) +
  geom_col(position = "fill") +
  labs(title = "Gender vs proportion of people", y = "Proportion of people") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```

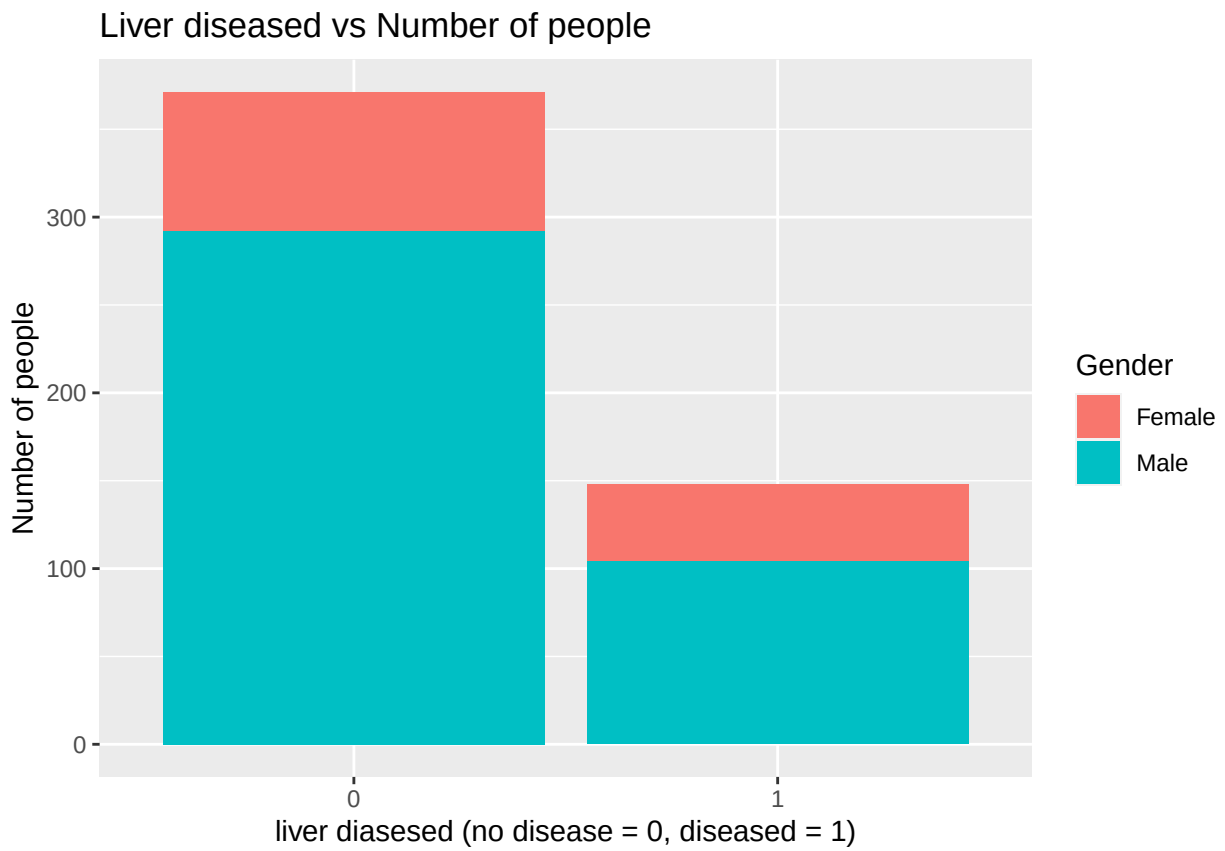


Analyze Dataset

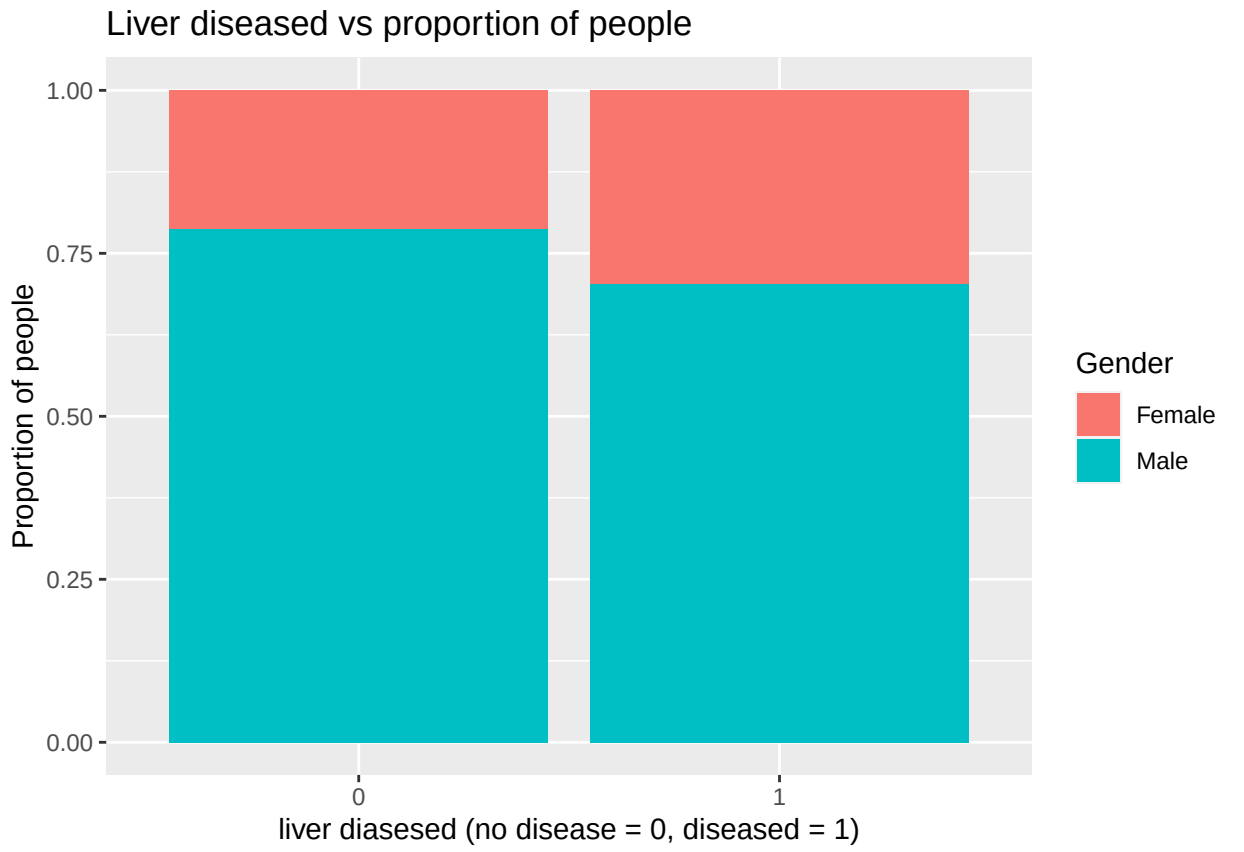
The first histogram shows that most patient do not have liver diseases.

The second normalized histogram shows that lady patient are more likely to have liver diseases than men similarly to the last section that analyzed by gender.

```
######################### Dataset plots ########################################

# plot a histogram by Dataset for training set colored by gender
training_set %>%
  group_by(Dataset, Gender) %>%
  summarize(count = n()) %>%
  ggplot(aes(Dataset, count, fill = Gender)) +
  geom_col() +
  labs(title = "Liver diseased vs Number of people",
       y = "Number of people",
       x = "liver diasesed (no disease = 0, diseased = 1)")
```




```
# plot a normalized histogram by Dataset for training set colored by gender
training_set %>%
  group_by(Dataset, Gender) %>%
  summarize(count = n()) %>%
  ggplot(aes(Dataset, count, fill = Gender)) +
  geom_col(position = "fill") +
  labs(title = "Liver diseased vs proportion of people",
       y = "Proportion of people",
       x = "liver diasesed (no disease = 0, diseased = 1)")
```

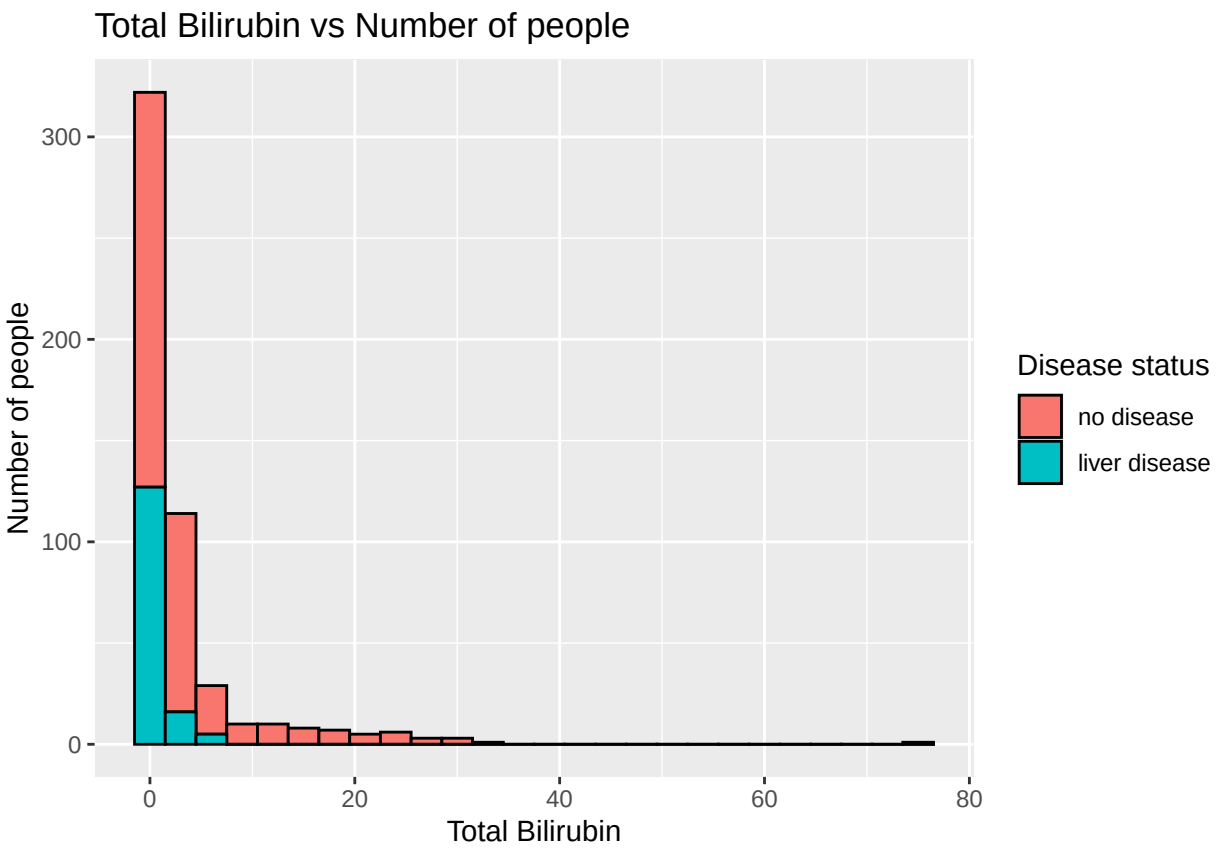


Analyze Total Bilirubin

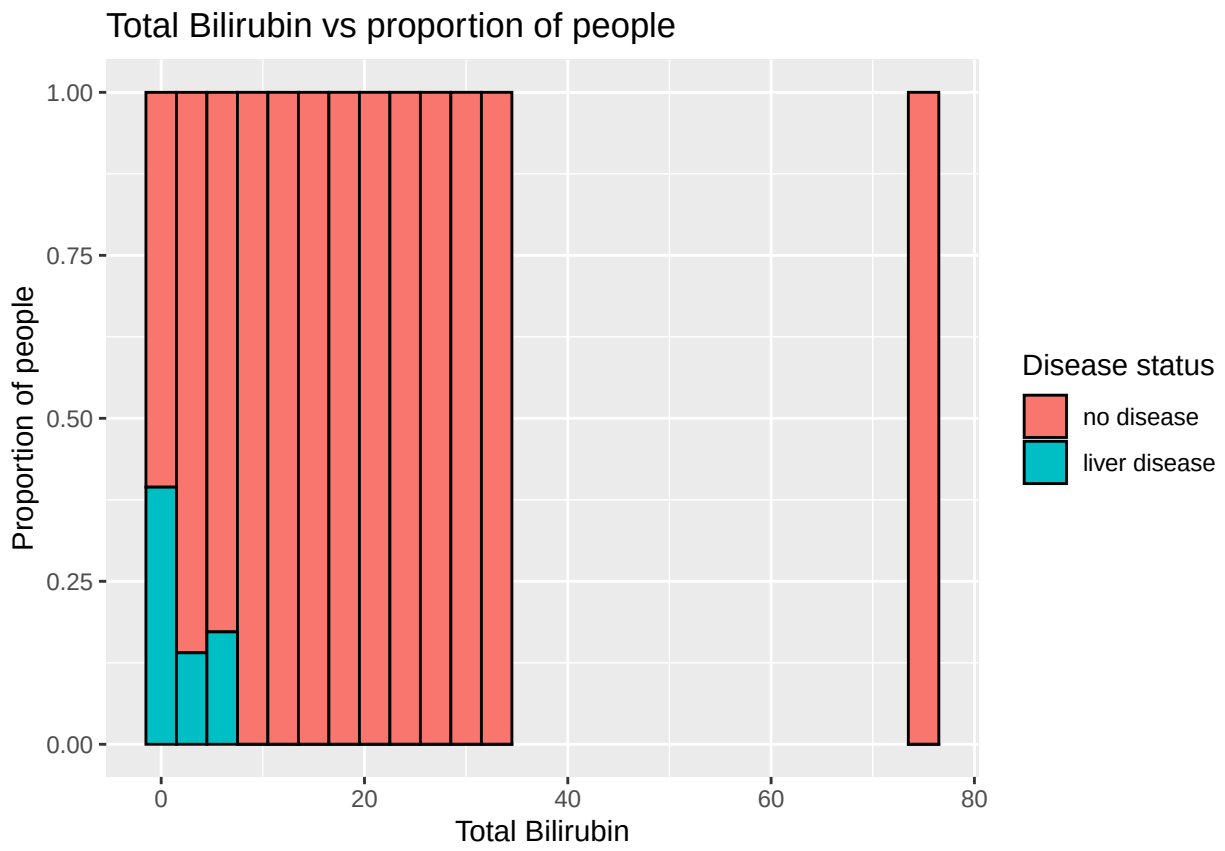
The first histogram shows that most patient have total bilirubin level less than 10 whether they have liver disease or not.

The second normalize histogram shows clearly that liver diseased patient always have total bilirubin level less than 10.

```
######################### Total Bilirubin plots ##########################################

# Plot a histogram by Total Bilirubin colored by Dataset
training_set %>%
  ggplot(aes(Total_Bilirubin, fill = Dataset)) +
  geom_histogram(binwidth = 3, color = "black") +
  labs(title = "Total Bilirubin vs Number of people",
       y = "Number of people",
       x = "Total Bilirubin") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# Plot a normalized histogram by Total Bilirubin colored by Dataset
training_set %>%
  ggplot(aes(Total_Bilirubin, fill = Dataset)) +
  geom_histogram(binwidth = 3, color = "black", position = "fill") +
  labs(title = "Total Bilirubin vs proportion of people",
       y = "Proportion of people",
       x = "Total Bilirubin") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```

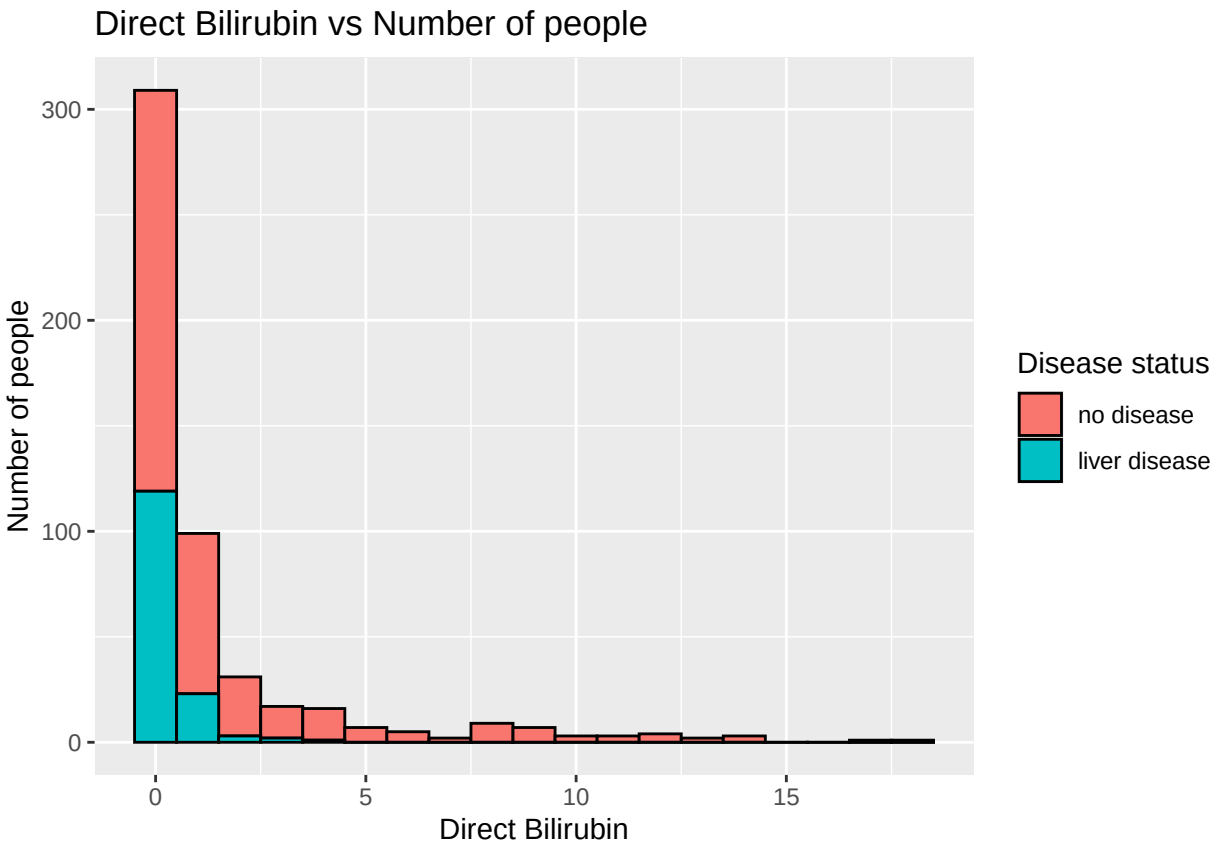


Analyze Direct Bilirubin

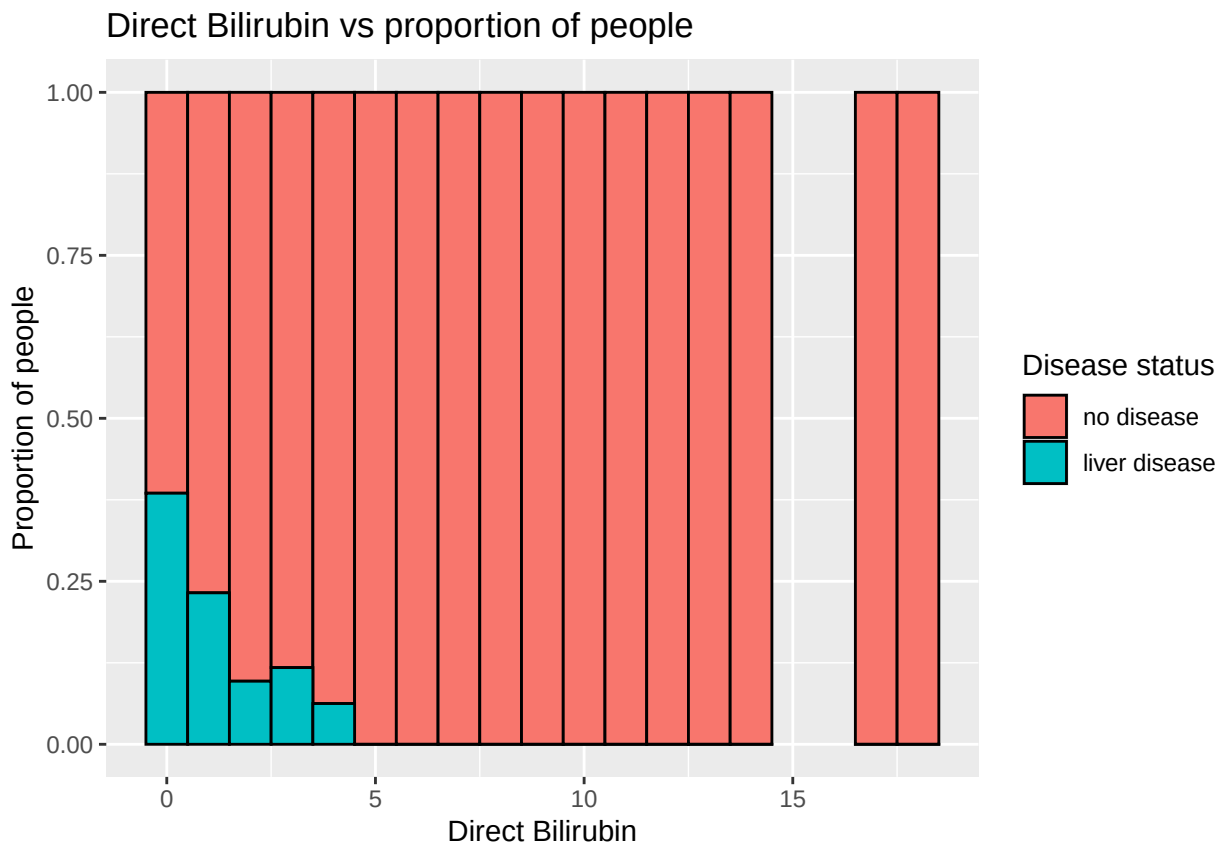
The first histogram shows that most patient have Direct bilirubin level less than 5 whether they have liver disease or not.

The second normalize histogram shows clearly that liver diseased patient always have Direct bilirubin level less than 5.

```
######################### Direct Bilirubin plots ################################

# Plot a histogram by Direct Bilirubin colored by Dataset
training_set %>%
  ggplot(aes(Direct_Bilirubin, fill = Dataset)) +
  geom_histogram(binwidth = 1, color = "black") +
  labs(title = "Direct Bilirubin vs Number of people",
       y = "Number of people",
       x = "Direct Bilirubin") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# Plot a normalized histogram by Direct Bilirubin colored by Dataset
training_set %>%
  ggplot(aes(Direct_Bilirubin, fill = Dataset)) +
  geom_histogram(binwidth = 1, color = "black", position = "fill") +
  labs(title = "Direct Bilirubin vs proportion of people",
       y = "Proportion of people",
       x = "Direct Bilirubin") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```

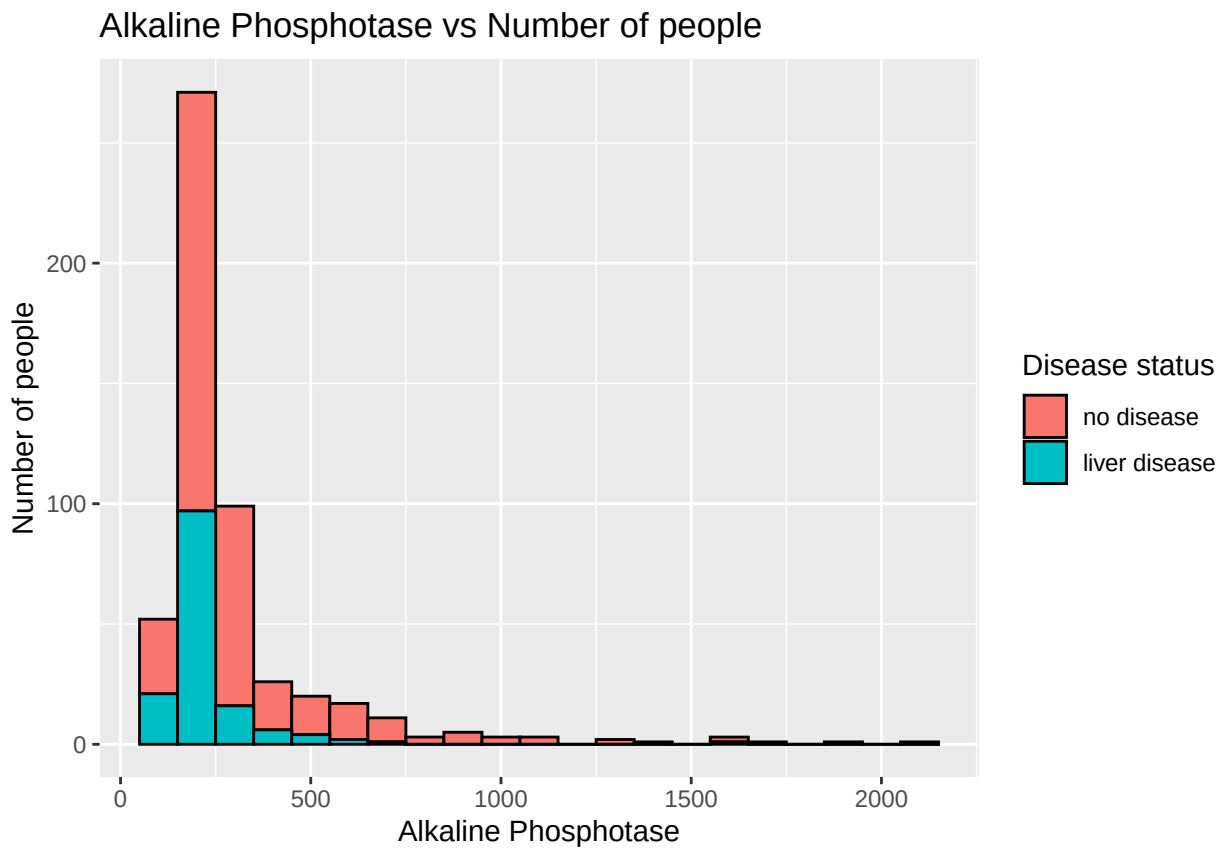


Analyze Alkaline Phosphotase

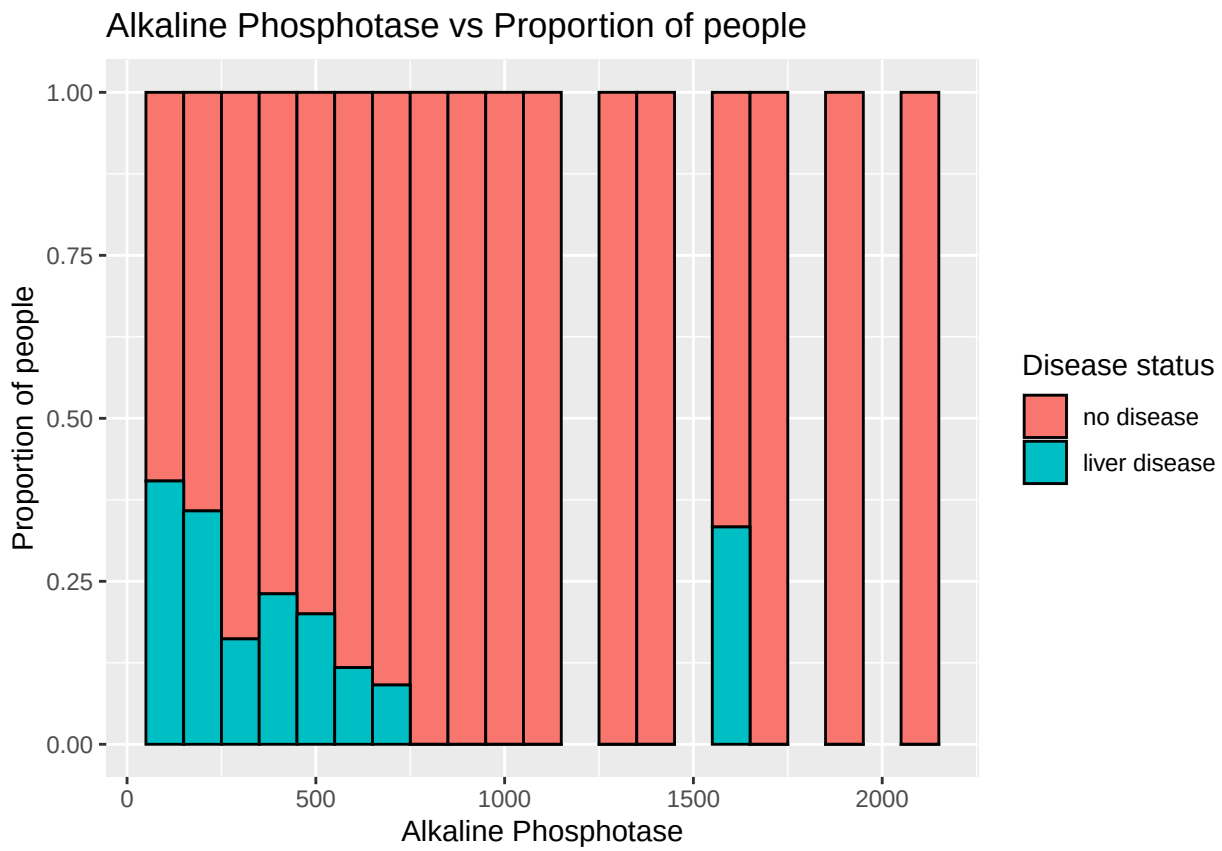
The first histogram shows that most patient have Alkaline Phosphotase level less than 500 whether they have liver disease or not.

The second normalize histogram shows clearly that majority of liver diseased patient have Alkaline Phosphotase level less than 750.

```
########################## Alkaline Phosphotase plots ############################

# Plot a histogram by Alkaline Phosphotase colored by Dataset
training_set %>%
  ggplot(aes(Alkaline_Phosphotase , fill = Dataset)) +
  geom_histogram(binwidth = 100, color = "black") +
  labs(title = "Alkaline Phosphotase vs Number of people",
       y = "Number of people",
       x = "Alkaline Phosphotase") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# Plot a normalized histogram by Alkaline Phosphotase colored by Dataset
training_set %>%
  ggplot(aes(Alkaline_Phosphotase , fill = Dataset)) +
  geom_histogram(binwidth = 100, color = "black", position = "fill") +
  labs(title = "Alkaline Phosphotase vs Proportion of people",
       y = "Proportion of people",
       x = "Alkaline Phosphotase") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```

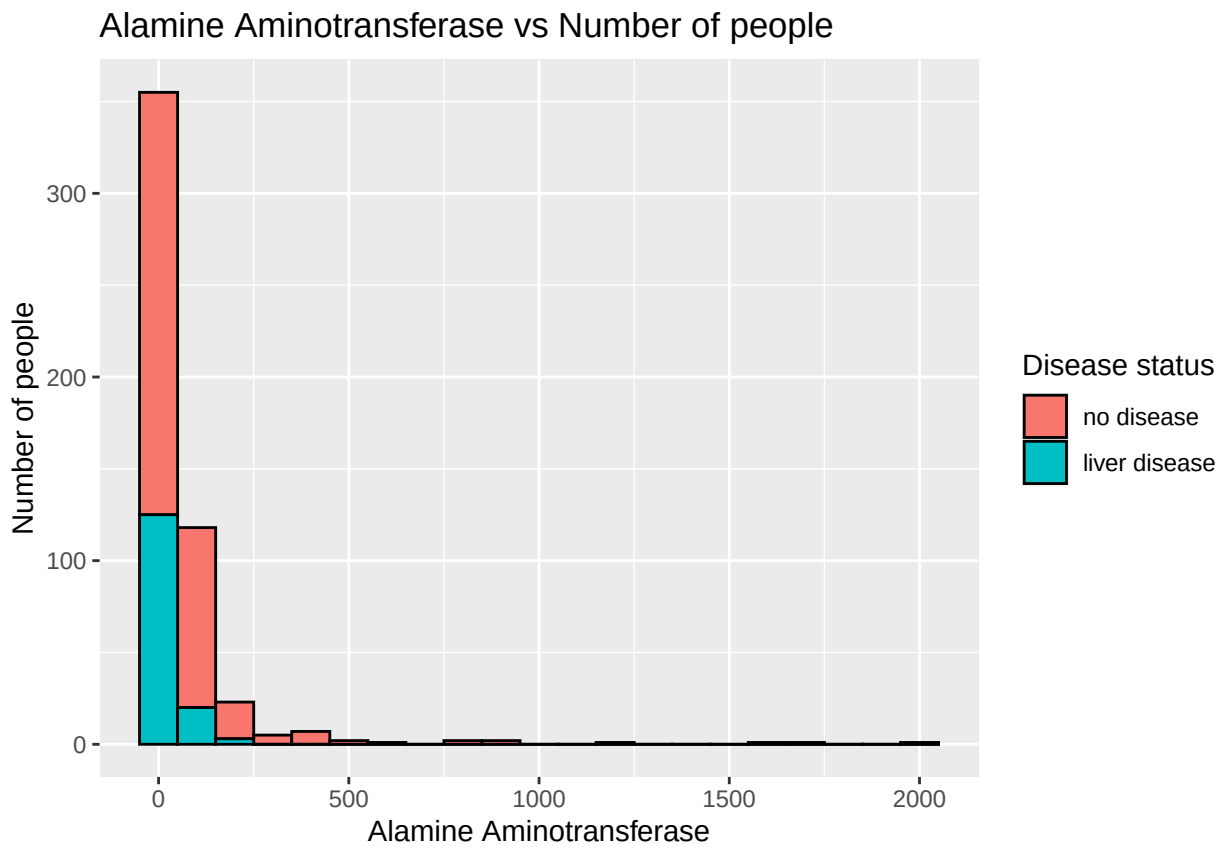


Analyze Alamine Aminotransferase

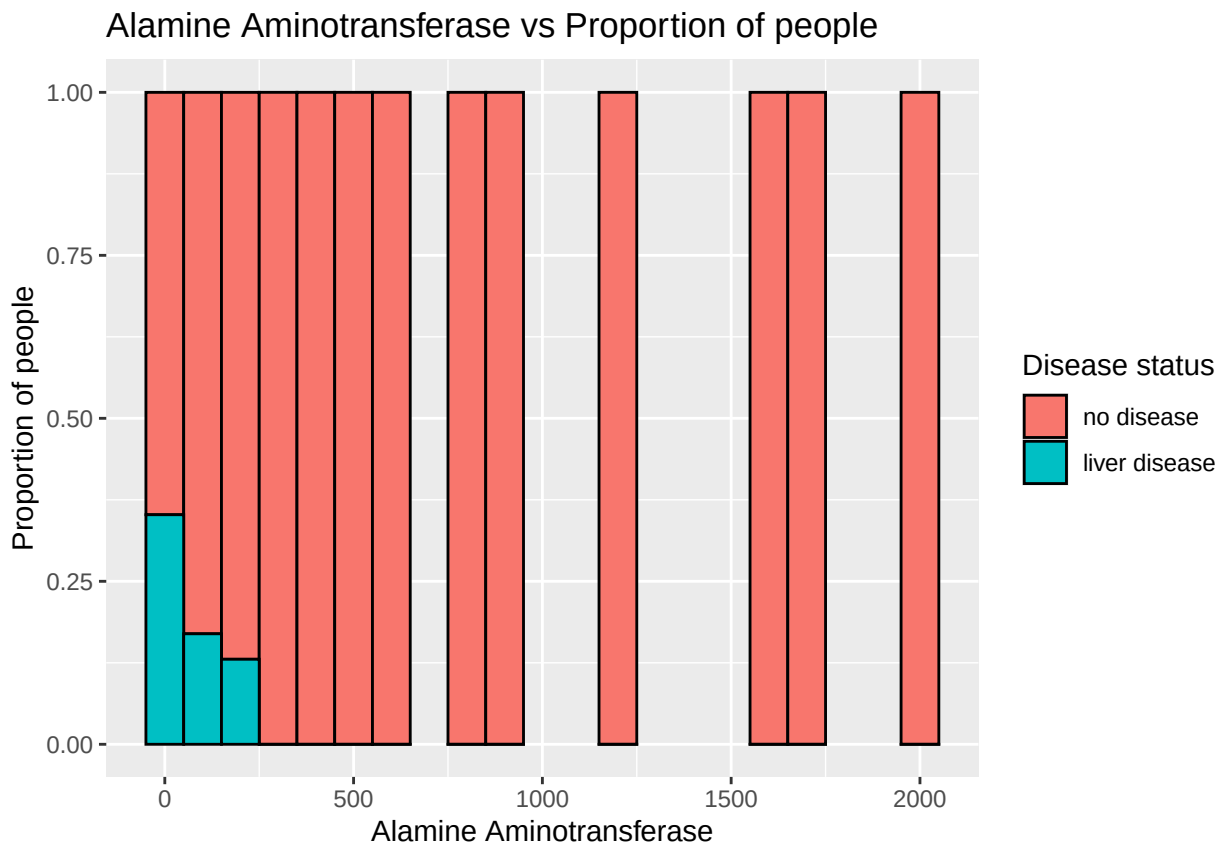
The first histogram shows that most patient have Alamine Aminotransferase level less than 250 whether they have liver disease or not.

The second normalize histogram shows clearly that liver diseased patient always have Alamine Aminotransferase level less than 250.

```
######################### Alamine Aminotransferase plots ########################

# Plot a histogram by Alamine Aminotransferase colored by Dataset
training_set %>%
  ggplot(aes(Alamine_Aminotransferase , fill = Dataset)) +
  geom_histogram(binwidth = 100, color = "black") +
  labs(title = "Alamine Aminotransferase vs Number of people",
       y = "Number of people",
       x = "Alamine Aminotransferase") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# Plot a normalized histogram by Alamine Aminotransferase colored by Dataset
training_set %>%
  ggplot(aes(Alamine_Aminotransferase , fill = Dataset)) +
  geom_histogram(binwidth = 100, color = "black", position = "fill") +
  labs(title = "Alamine Aminotransferase vs Proportion of people",
       y = "Proportion of people",
       x = "Alamine Aminotransferase") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```

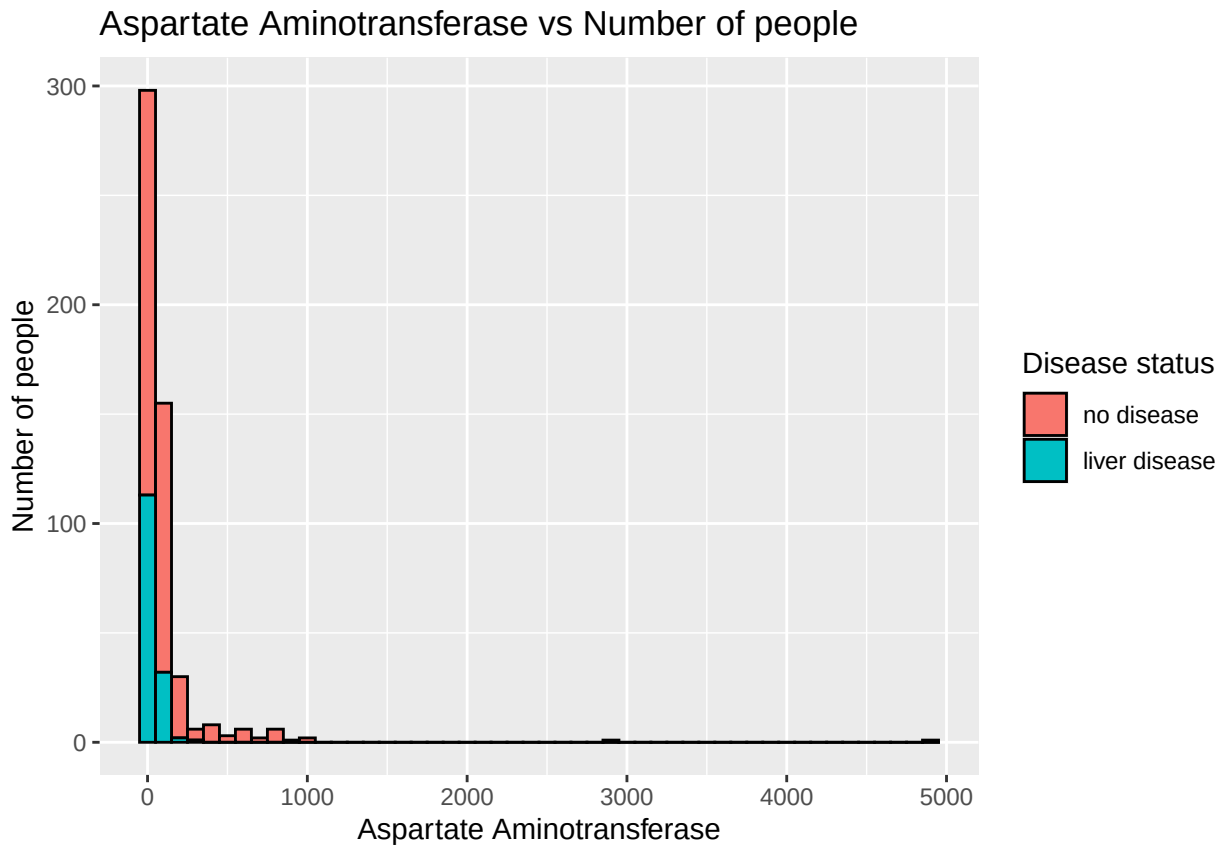


Analyze Aspartate Aminotransferase

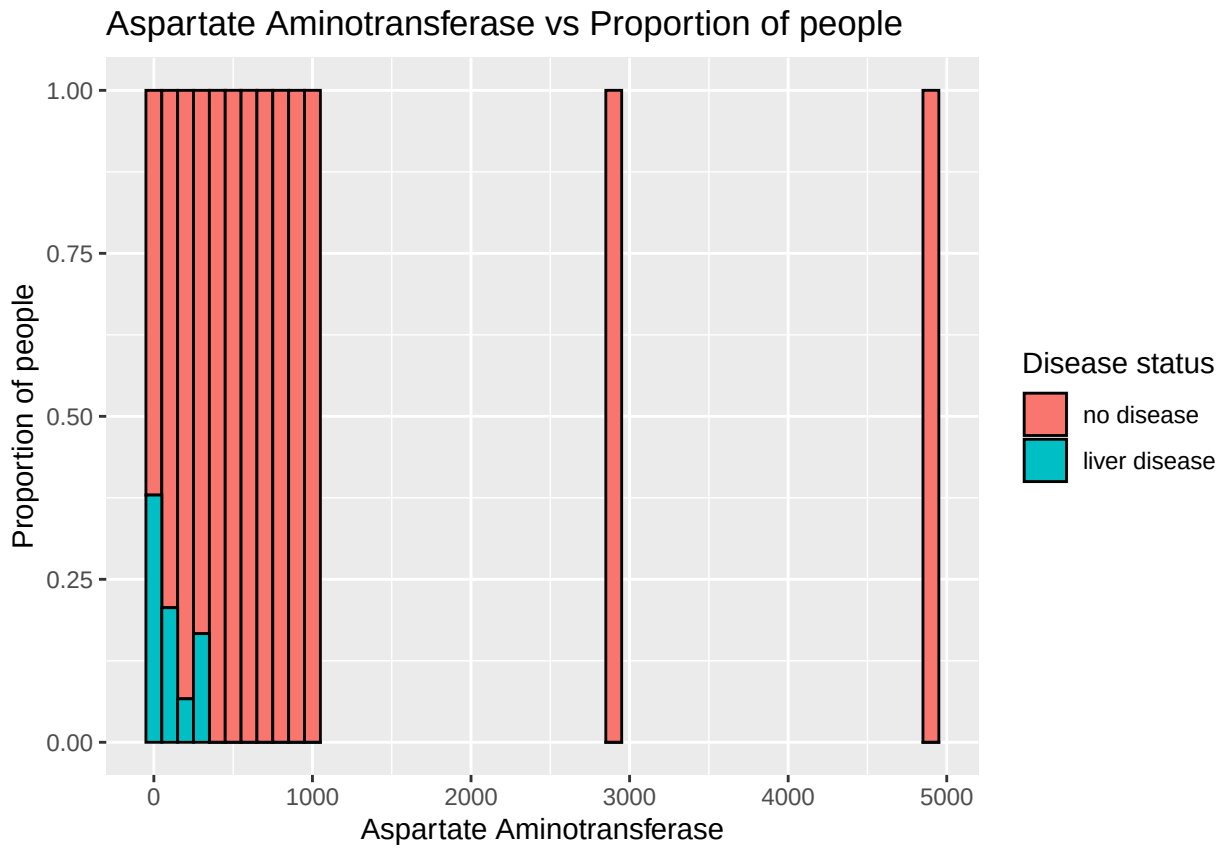
The first histogram shows that most patient have Aspartate Aminotransferase level less than 500 whether they have liver disease or not.

The second normalize histogram shows clearly that liver diseased patients always have Aspartate Aminotransferase level less than 500.

```
######################### Aspartate Aminotransferase plots ####################

# Plot a histogram by Aspartate Aminotransferase colored by Dataset
training_set %>%
  ggplot(aes(Aspartate_Aminotransferase , fill = Dataset)) +
  geom_histogram(binwidth = 100, color = "black") +
  labs(title = "Aspartate Aminotransferase vs Number of people",
       y = "Number of people",
       x = "Aspartate Aminotransferase") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# Plot a normalized histogram by Aspartate Aminotransferase colored by Dataset
training_set %>%
  ggplot(aes(Aspartate_Aminotransferase , fill = Dataset)) +
  geom_histogram(binwidth = 100, color = "black", position = "fill") +
  labs(title = "Aspartate Aminotransferase vs Proportion of people",
       y = "Proportion of people",
       x = "Aspartate Aminotransferase") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```

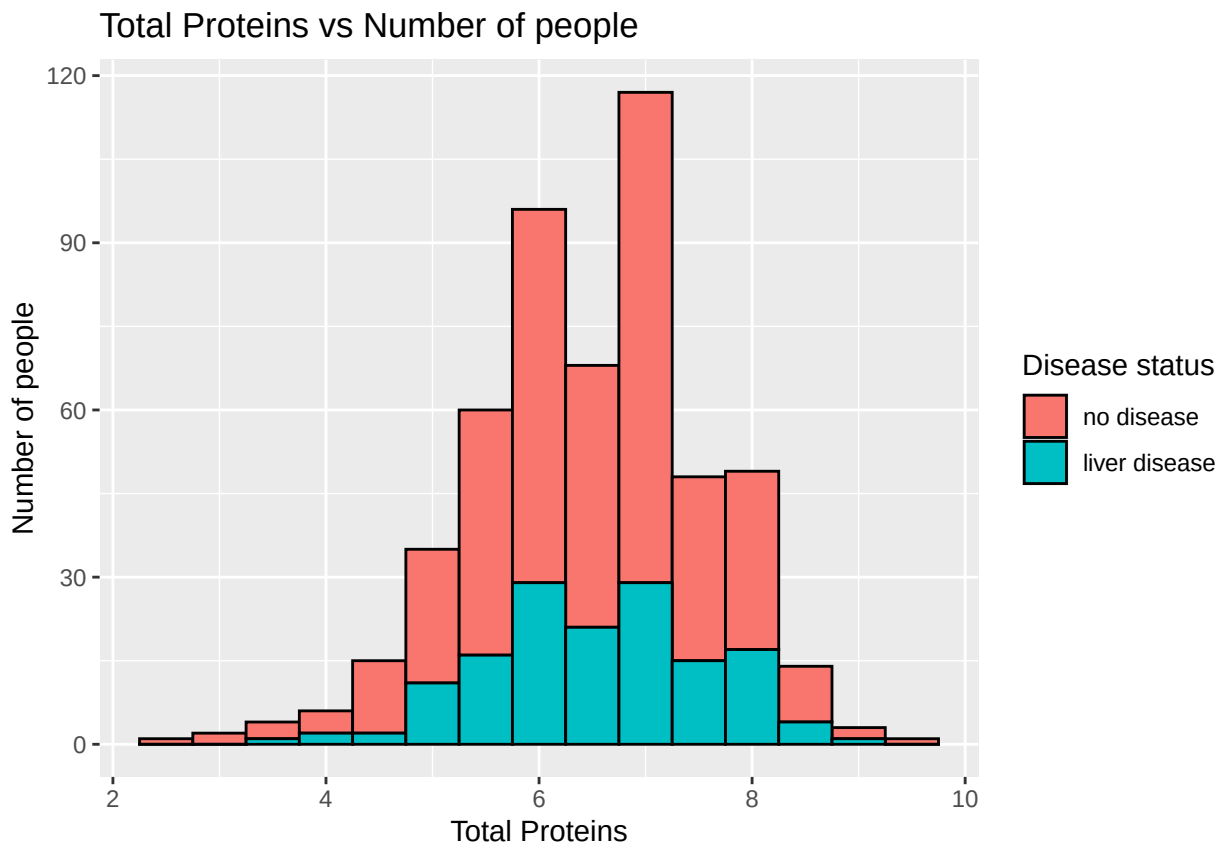


Analyze Total Protein

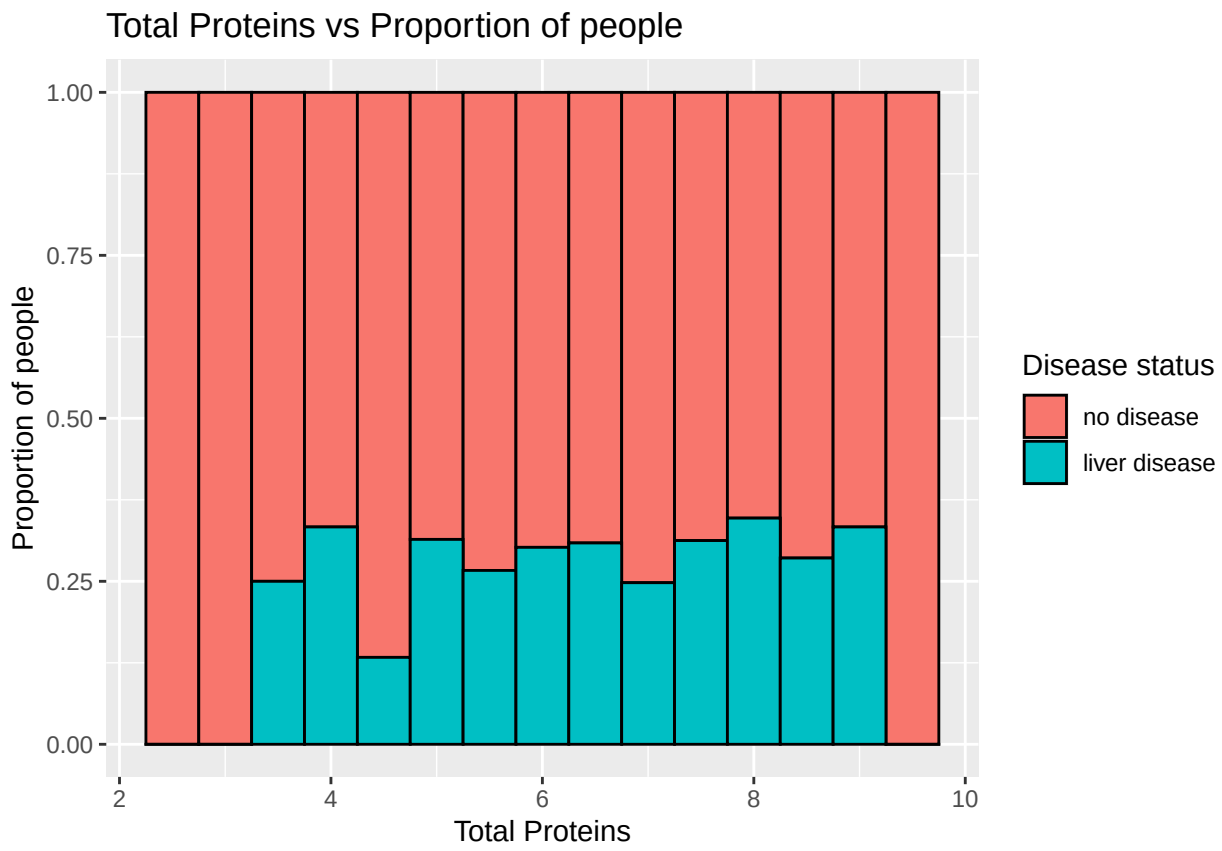
The first histogram shows that most patient have Total Protein level between 5 and 8 whether they have liver disease or not.

The second normalize histogram shows clearly that liver diseased patients have varying amount of Total Protein level.

```
######################### Total Protein plots ###################################

######################### Total Protein plots ###################################

# Plot a histogram by Total Proteins colored by Dataset
training_set %>%
  ggplot(aes(Total_Protiens, fill = Dataset)) +
  geom_histogram(binwidth = 0.5, color = "black") +
  labs(title = "Total Proteins vs Number of people",
       y = "Number of people",
       x = "Total Proteins") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# Plot a normalized histogram by Total Protiens colored by Dataset
training_set %>%
  ggplot(aes(Total_Protiens, fill = Dataset)) +
  geom_histogram(binwidth = 0.5, color = "black", position = "fill") +
  labs(title = "Total Proteins vs Proportion of people",
       y = "Proportion of people",
       x = "Total Proteins") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```

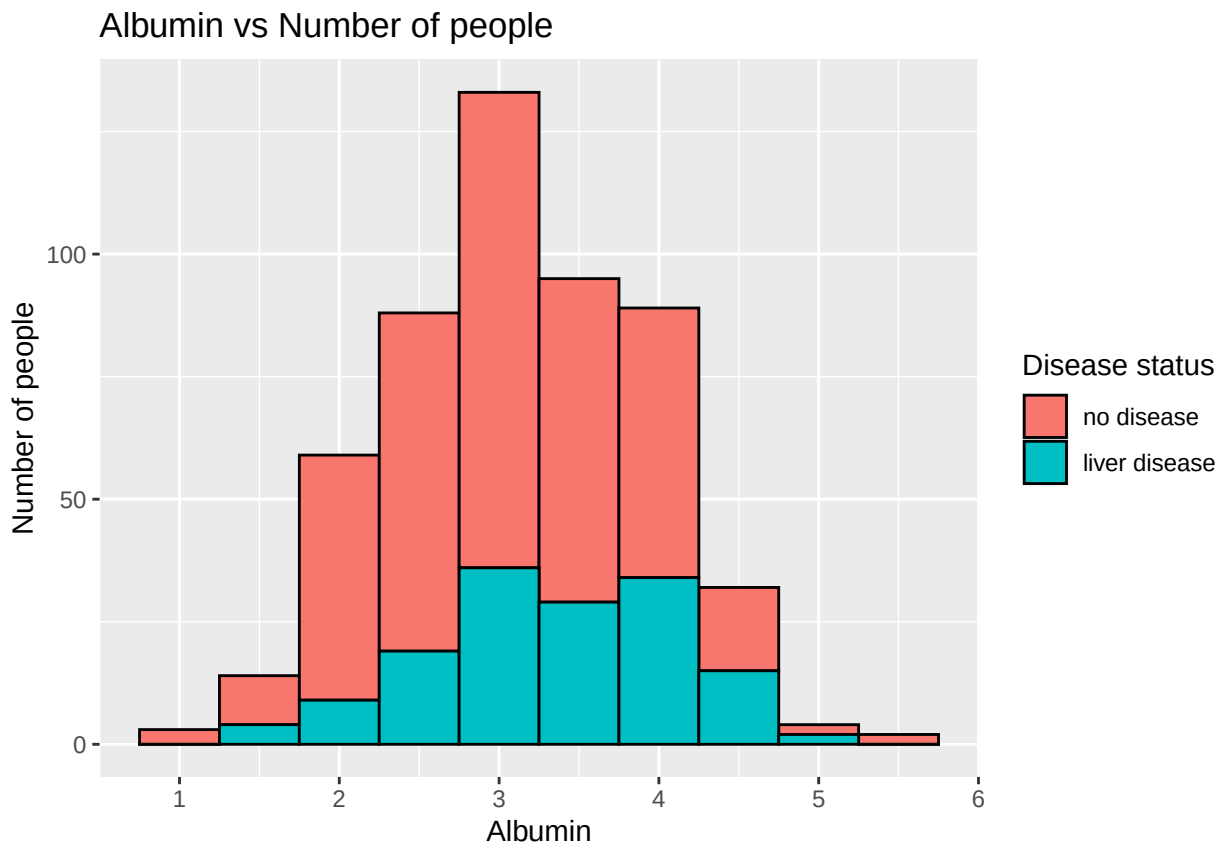


Analyze Albumin

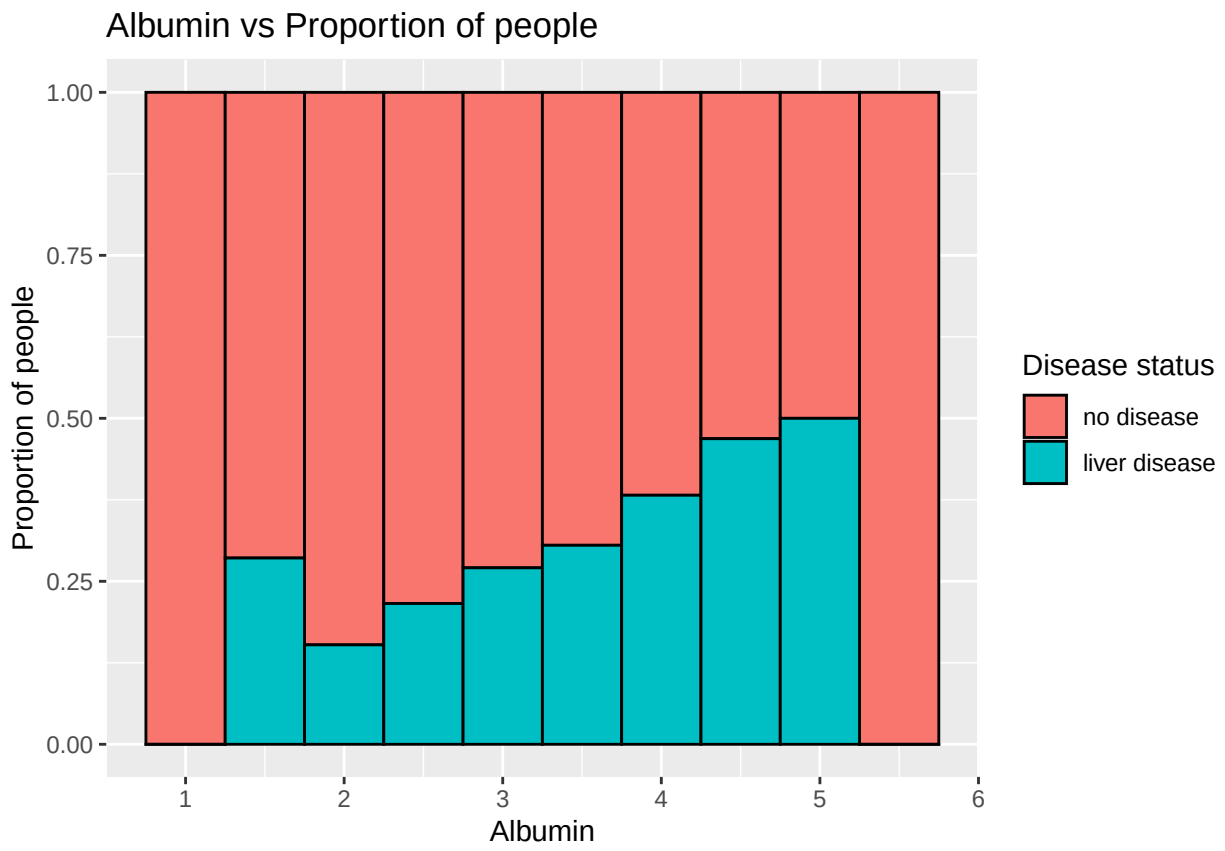
The first histogram shows that most patient have Albumin level between 2 and 4.5 whether they have liver disease or not.

The second normalize histogram shows clearly that liver diseased patient have varying amount of Albumin level.

```
######################### Albumin plots ##########################################

# Plot a histogram by Albumin colored by Dataset
training_set %>%
  ggplot(aes(Albumin, fill = Dataset)) +
  geom_histogram(binwidth = 0.5, color = "black") +
  labs(title = "Albumin vs Number of people",
       y = "Number of people",
       x = "Albumin") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# Plot a normalized histogram by Albumin colored by Dataset
training_set %>%
  ggplot(aes(Albumin, fill = Dataset)) +
  geom_histogram(binwidth = 0.5, color = "black", position = "fill") +
  labs(title = "Albumin vs Proportion of people",
       y = "Proportion of people",
       x = "Albumin") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```

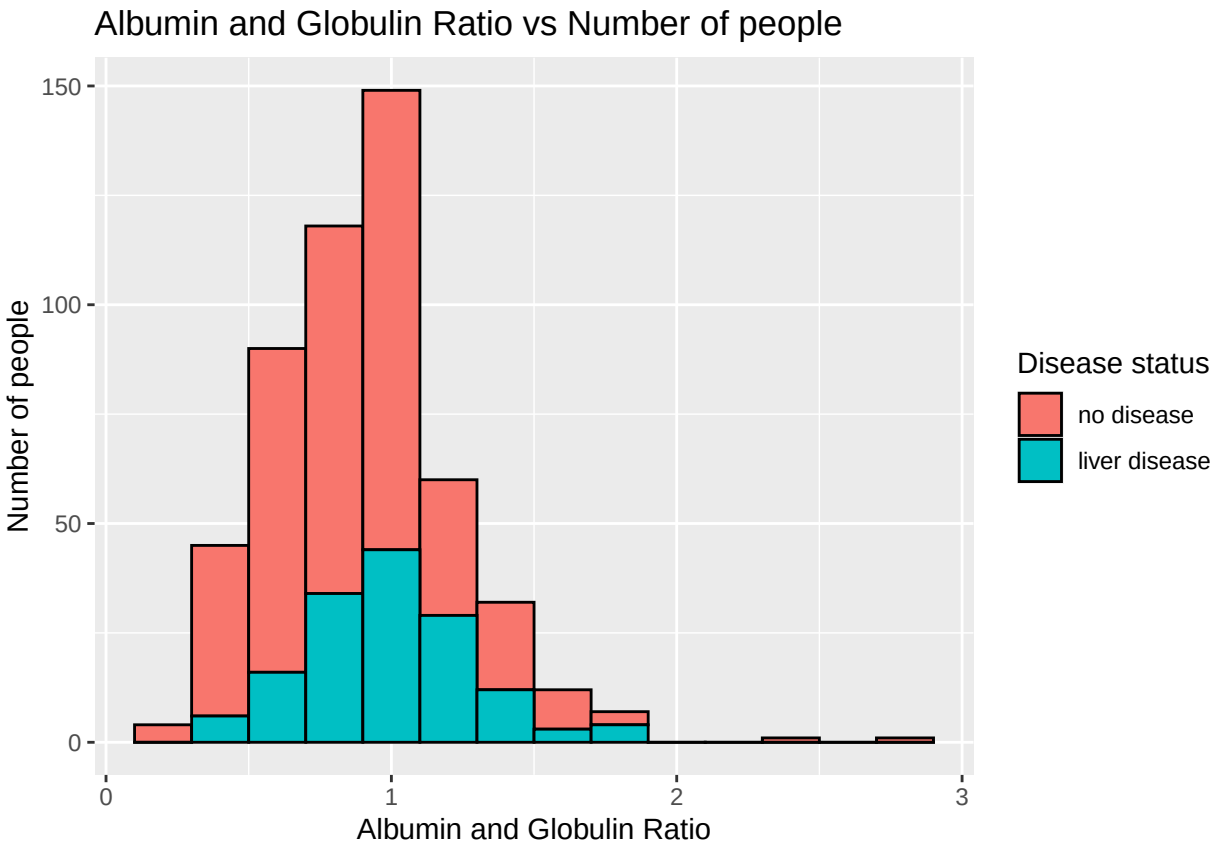


Analyze Albumin and Globulin Ratio

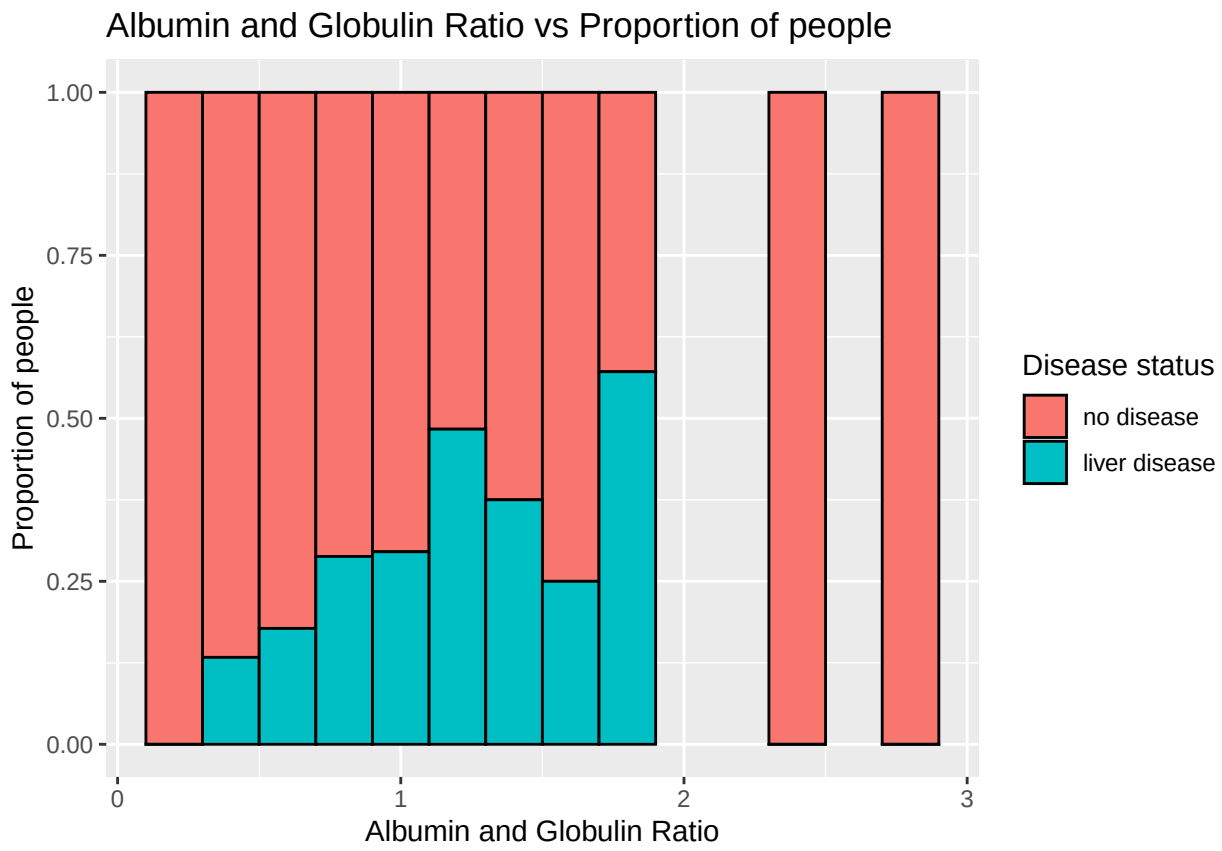
The first histogram shows that most patient have Albumin and Globulin Ratio level between 0.5 and 1.5 whether they have liver disease or not.

The second normalize histogram shows clearly that liver diseased patient have varying amount of Albumin and Globulin Ratio level.

```
######################### Albumin and Globulin Ratio plots ####################

# Plot a histogram by Albumin and Globulin Ratio colored by Dataset
training_set %>%
  ggplot(aes(Albumin_and_Globulin_Ratio, fill = Dataset)) +
  geom_histogram(binwidth = 0.2, color = "black") +
  labs(title = "Albumin and Globulin Ratio vs Number of people",
       y = "Number of people",
       x = "Albumin and Globulin Ratio") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```




```
# Plot a normalized histogram by Albumin and Globulin Ratio colored by Dataset
training_set %>%
  ggplot(aes(Albumin_and_Globulin_Ratio, fill = Dataset)) +
  geom_histogram(binwidth = 0.2, color = "black", position = "fill") +
  labs(title = "Albumin and Globulin Ratio vs Proportion of people",
       y = "Proportion of people",
       x = "Albumin and Globulin Ratio") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```



Principal Component Analysis (PCA)

The first 2 charts shows correlation between the 10 fields.

The following pairs of field have a particularly strong correlation:

- Total_Bilirubin & Direct_Bilirubin
- Alamine_Aminotransferase & Aspartage_Aminotransferase
- Total_Proteins & Albumin
- Albumin_and_Globulin_Raation & Albumin

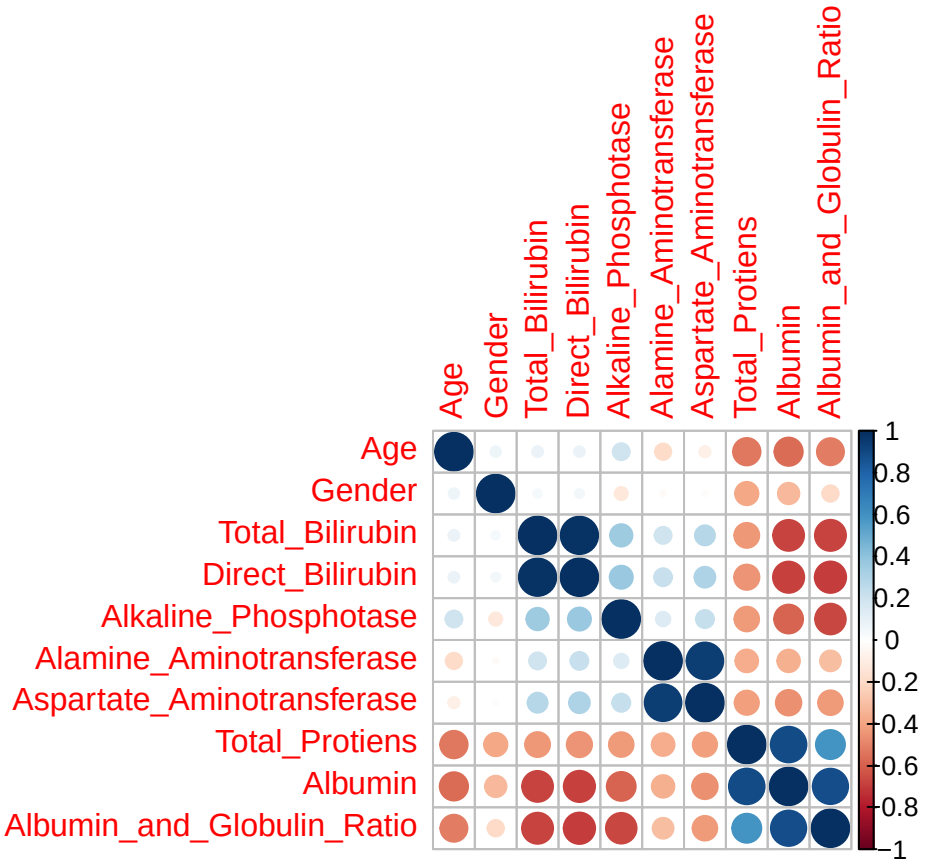
The following fields have very little correlation to any other variables:

- Age
- Gender
- Alkaline_Phosphotase

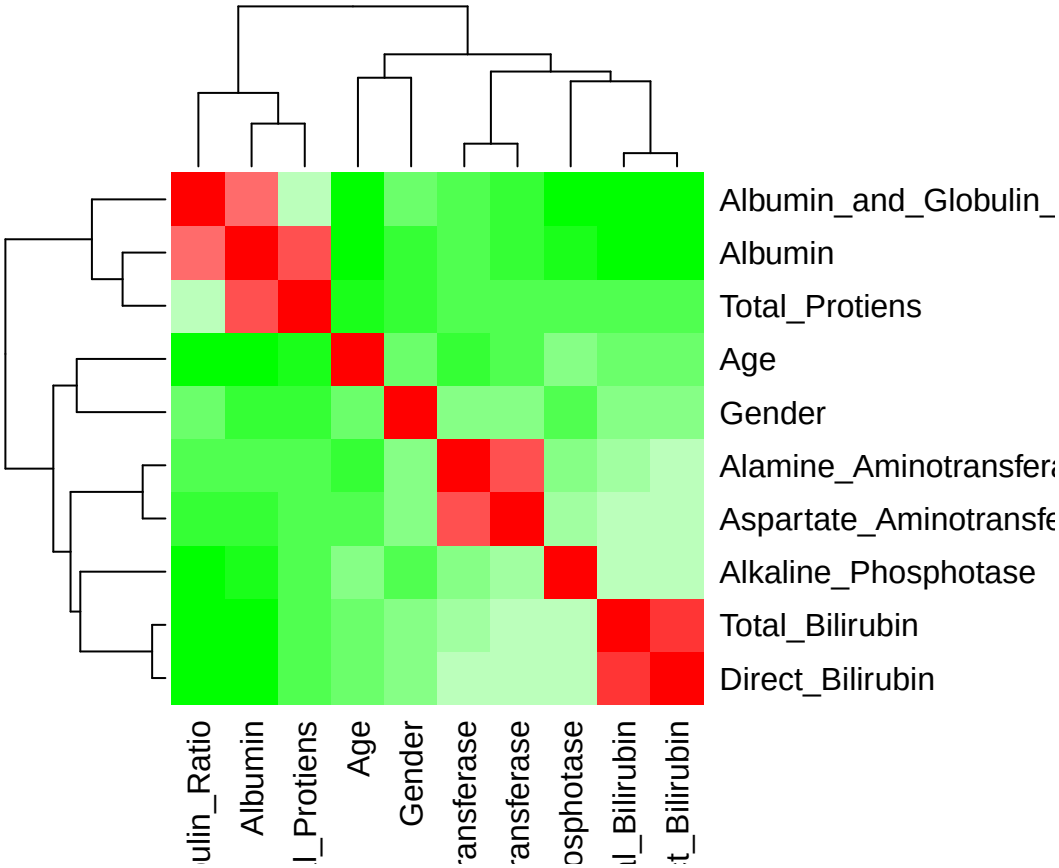
PCA has been carried out on the patient data as a dimensionality reduction technique. The tabular output of the R console shows that 10 PCs have been created, and that 5 PCs take into account 99.998% of the variance in the data.

The final graph of the section plots PC1 vs PC2, and shows that liver diseased patients (sick = 1 and blue) cluster in the left hand upper side of the graph. Unfortunately, liver diseased patients are overlapping significantly with non-liver diseased patients in the graph, which indicates that ML model may struggle to have a high specificity.

```
###################  Principal Component Analysis (PCA)  ########################

# convert test and training sets from data frames to matrices
test_set_matrix <-
  test_set[,1:10] %>%
  mutate(Gender = Gender == "Male") %>%
  as.matrix()

training_set_matrix <-
  training_set[,1:10] %>%
  mutate(Gender = Gender == "Male") %>%
  as.matrix()

# calculate correlation between fields in the training set
mydata_cor <- cor(training_set_matrix)

# plot correlation
corrplot(cor(mydata_cor))
```



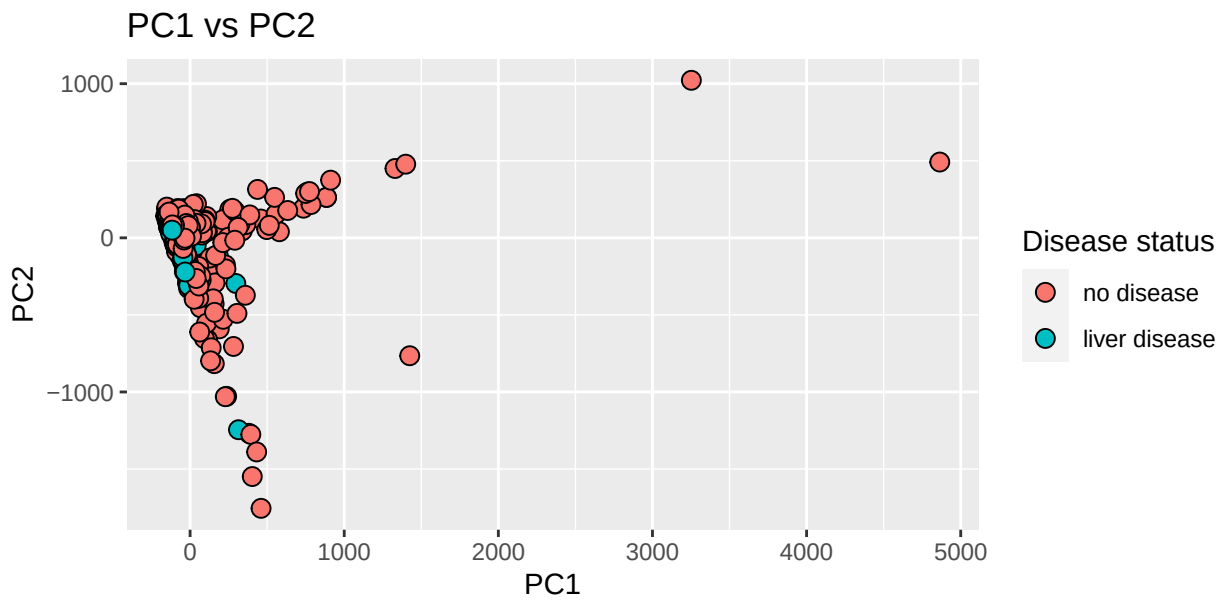

```
# plot correlation as heat map
palette = colorRampPalette(c("green", "white", "red")) (20)
heatmap(x = mydata_cor, col = palette, symm = TRUE)
```




```
# carry out PCA on training set
pca <- prcomp(training_set_matrix)

# show summary of PCA to gain insight on the PCs that account for most of the variance
summary(pca)
```

```
## Importance of components:
##                             PC1      PC2      PC3      PC4     PC5     PC6
## Standard deviation     325.7868 240.6435 98.58394 16.11399 6.36369 1.31188
## Proportion of Variance   0.6097   0.3327  0.05583  0.00149 0.00023 0.00001
## Cumulative Proportion    0.6097   0.9424  0.99825  0.99975 0.99998 0.99999
##                            PC7    PC8    PC9   PC10
## Standard deviation     1.23101 0.4429 0.4181 0.1428
## Proportion of Variance 0.00001 0.0000 0.0000 0.0000
## Cumulative Proportion  1.00000 1.0000 1.0000 1.0000
```

```
# plot PC1 vs PC2 and color point by whehter teh patient is sick or not
data.frame(pca$x[,1:2], Sick=training_set$Dataset) %>%
  ggplot(aes(PC1,PC2, fill = Sick))+
  geom_point(cex=3, pch=21) +
  coord_fixed(ratio = 1) +
  labs(title = "PC1 vs PC2") +
  scale_fill_discrete(name = "Disease status",
                      labels = c("no disease", "liver disease"))
```



Results

ML models to predict whether a given patient has a liver disease or not are now created. The following algorithms were explored: Generalized Liner Model, Boosted Logistic Regression, Random Forest and k-nearest neighbors. The models were created using both raw data from the liver patient data set and corresponding derived principal components.

With regards to measuring the performance of the models, the following metrics were used in priority order:

- precision
- sensitivity
- specificity
- balanced accuracy

Precision was considered most important as the prevalence of liver diseased patient is low and a liver diseased predictions need high rate of confirmation. Sensitivity was considered highly important as liver disease is a serious condition and a false negative can be particularly damaging for the prognosis of the patient without proper treatment.

create a function that returns the performance of a model in a data frame

A function was created that allows the efficient exploration of ML models through the calculation of a confusion matrix and the loading of the important performance metrics into a convenient data frame.

```r
# create a function that returns the performance of a model in a data frame

model_performance <-
  function (model, name_of_model) {
    # measure and record performance of model
    cm <- confusionMatrix(predict(model, test_set), test_set$Dataset, positive = '1')
    # create a data frame with overall precision, sensitivity, specificity and accuracy
    data_frame(Method = name_of_model,
               Precision = cm$byClass["Precision"],
               sensitivity = cm$byClass["Sensitivity"],
               Specificity = cm$byClass["Specificity"],
               Balanced_Accuracy = cm$byClass["Balanced Accuracy"])
  }
```

Model 1: Generalized Linear Model (GLM)

The first ML model is based on a GLM and the model is trained using both the raw liver patient data and 5 PCs that accounts for the vast majority of the variance in the data. GLM's peformance on the raw data was significantly better than that deployed the 5 PCs.

66% precision was achieved, but the 35% sensitivity is unacceptably low.

```r
##################### Generalized Linear Model (GLM) ############################

# create a linear model
train_glm <-
  train(Dataset ~ .,
        method = "glm",
        data = training_set)

# assess performance of results and store in a data frame
results <- model_performance(train_glm, "linear model")

# print out all the results for all the models explored above
results %>% knitr::kable()
```

Method	Precision	sensitivity	Specificity	Balanced_Accuracy
linear model	0.6666667	0.3529412	0.9285714	0.6407563

```r
# create a linear model with the 5 most significant PCs
train_glm_pca <-
  train(Dataset ~ .,
        method = "glm",
        data = training_set,
        preProcess = "pca",
        trControl = trainControl(preProcOptions = list(pcaComp = 5)))

# assess performance of results and add store as a new row in results
results <- bind_rows(results, model_performance(train_glm_pca, "linear model with 5 PCs"))

# print out all the results for all the models explored above
results %>% knitr::kable()
```

Method	Precision	sensitivity	Specificity	Balanced_Accuracy
linear model	0.6666667	0.3529412	0.9285714	0.6407563
linear model with 5 PCs	0.5555556	0.2941176	0.9047619	0.5994398

Model 2: Boosted Logistic Regression (LogitBoost)

The second ML model is based on a LogitBoost and the model is trained using both the raw liver patient data and 5 PCs that accounts for the vast majority of the variance in the data. LogitBoost's performance on the raw data was better than that deployed the 5 PCs.

60% precision was achieved, but the 35% sensitivity is unacceptably low when using the raw data.

```
####################### Boosted Logistic Regression (LogitBoost) ##############################

# create a LogitBoost model
train_LogitBoost <-
  train(Dataset ~ .,
        method = "LogitBoost",
        data = training_set)

# assess performance of results and add store as a new row in results
results <- bind_rows(results, model_performance(train_LogitBoost, "LogitBoost"))

# print out all the results for all the models explored above
results %>% knitr::kable()
```

Method	Precision	sensitivity	Specificity	Balanced_Accuracy
linear model	0.6666667	0.3529412	0.9285714	0.6407563
linear model with 5 PCs	0.5555556	0.2941176	0.9047619	0.5994398
LogitBoost	0.6000000	0.3529412	0.9047619	0.6288515

```
# create a KNN model with the 5 most significant PCs
train_LogitBoost_pca <-
  train(Dataset ~ .,
        method = "LogitBoost",
        data = training_set,
        preProcess = "pca",
        trControl = trainControl(preProcOptions = list(pcaComp = 5)))

# assess performance of results and add store as a new row in results
results <- bind_rows(results, model_performance(train_LogitBoost_pca, "LogitBoost with 5 PCs"))

# print out all the results for all the models explored above
results %>% knitr::kable()
```

Method	Precision	sensitivity	Specificity	Balanced_Accuracy
linear model	0.6666667	0.3529412	0.9285714	0.6407563
linear model with 5 PCs	0.5555556	0.2941176	0.9047619	0.5994398
LogitBoost	0.6000000	0.3529412	0.9047619	0.6288515

Method	Precision	sensitivity	Specificity	Balanced_Accuracy
LogitBoost with 5 PCs	0.3809524	0.4705882	0.6904762	0.5805322

Model 3: Random Forest (RF)

The third ML model is based on an RF and the model is trained using both the raw liver patient data and 5 PCs that accounts for the vast majority of the variance in the data. RF's performance on the raw data was worse than that deployed the 5 PCs.

57% precision was achieved through the use of PCs, and the 47% sensitivity is better than GLM and LogitBoost models above.

```
############################ Random Forest ########################################

# create a random forest model
train_Rborist <-
  train(Dataset ~ .,
        method = "Rborist",
        tuneGrid = data.frame(predFixed = 2, minNode = c(3, 50)),
        data = training_set)

# assess performance of results and add store as a new row in results
results <- bind_rows(results, model_performance(train_Rborist, "random forest"))

# print out all the results for all the models explored above
results %>% knitr::kable()
```

Method	Precision	sensitivity	Specificity	Balanced_Accuracy
linear model	0.6666667	0.3529412	0.9285714	0.6407563
linear model with 5 PCs	0.5555556	0.2941176	0.9047619	0.5994398
LogitBoost	0.6000000	0.3529412	0.9047619	0.6288515
LogitBoost with 5 PCs	0.3809524	0.4705882	0.6904762	0.5805322
random forest	0.5384615	0.4117647	0.8571429	0.6344538

```
# create a random forest with the 5 most significant PCs
train_Rborist_pca <-
  train(Dataset ~ .,
        method = "Rborist",
        tuneGrid = data.frame(predFixed = 2, minNode = c(3, 50)),
        data = training_set,
        preProcess = "pca",
        trControl = trainControl(preProcOptions = list(pcaComp = 5)))

# assess performance of results and add store as a new row in results
results <- bind_rows(results, model_performance(train_Rborist_pca, "random forest wiht with 5 PCs"))

# print out all the results for all the models explored above
results %>% knitr::kable()
```

Method	Precision	sensitivity	Specificity	Balanced_Accuracy
linear model	0.6666667	0.3529412	0.9285714	0.6407563
linear model with 5 PCs	0.5555556	0.2941176	0.9047619	0.5994398
LogitBoost	0.6000000	0.3529412	0.9047619	0.6288515
LogitBoost with 5 PCs	0.3809524	0.4705882	0.6904762	0.5805322
random forest	0.5384615	0.4117647	0.8571429	0.6344538
random forest wiht with 5 PCs	0.5714286	0.4705882	0.8571429	0.6638655

Model 4: k-nearest neighbors (KNN)

The fourth and last ML model is based on an KNN and the model is trained using both the raw liver patient data and 5 PCs that accounts for the vast majority of the variance in the data. KNN's performance on the raw data was worse than that deployed the 5 PCs.

66% precision was achieved through the use of PCs, and the 47% sensitivity is better than all previous models: GLM and LogitBoost and RF models.

The last table below shows that KNN modeling was carried out using 1 to 10 PCs to demonstrate that the use of 5 PCs leads to the best performance.

```
###################### k-nearest neighbors (KNN) ##############################

# create a KNN model
train_knn <-
  train(Dataset ~ .,
        method = "knn",
        data = training_set)

# assess performance of results and add store as a new row in results
results <- bind_rows(results, model_performance(train_knn, "KNN"))

# print out all the results for all the models explored above
results %>% knitr::kable()
```

Method	Precision	sensitivity	Specificity	Balanced_Accuracy
linear model	0.6666667	0.3529412	0.9285714	0.6407563
linear model with 5 PCs	0.5555556	0.2941176	0.9047619	0.5994398
LogitBoost	0.6000000	0.3529412	0.9047619	0.6288515
LogitBoost with 5 PCs	0.3809524	0.4705882	0.6904762	0.5805322
random forest	0.5384615	0.4117647	0.8571429	0.6344538
random forest wiht with 5 PCs	0.5714286	0.4705882	0.8571429	0.6638655
KNN	0.4666667	0.4117647	0.8095238	0.6106443

```
# create a KNN model with the 5 most significant PCs
train_knn_pca <-
  train(Dataset ~ .,
        method = "knn",
        data = training_set,
        preProcess = "pca",
        trControl = trainControl(preProcOptions = list(pcaComp = 5)))
```

```r
# assess performance of results and add store as a new row in results
results <- bind_rows(results, model_performance(train_knn_pca, "KNN with 5 PCs"))

# print out all the results for all the models explored above
results %>% knitr::kable()
```

Method	Precision	sensitivity	Specificity	Balanced_Accuracy
linear model	0.6666667	0.3529412	0.9285714	0.6407563
linear model with 5 PCs	0.5555556	0.2941176	0.9047619	0.5994398
LogitBoost	0.6000000	0.3529412	0.9047619	0.6288515
LogitBoost with 5 PCs	0.3809524	0.4705882	0.6904762	0.5805322
random forest	0.5384615	0.4117647	0.8571429	0.6344538
random forest wiht with 5 PCs	0.5714286	0.4705882	0.8571429	0.6638655
KNN	0.4666667	0.4117647	0.8095238	0.6106443
KNN with 5 PCs	0.6666667	0.4705882	0.9047619	0.6876751

```r
#explore performance of model using 1-10 PCs
knn_results <- sapply(seq(1,10), function(pc) {
  train_knn_pca <-
    train(Dataset ~ .,
          method = "knn",
          data = training_set,
          preProcess = "pca",
          trControl = trainControl(preProcOptions = list(pcaComp = pc)))
  cm <- confusionMatrix(predict(train_knn_pca, test_set), test_set$Dataset, positive = '1')
  data_frame(PC = pc,
             Precision = cm$byClass["Precision"],
             sensitivity = cm$byClass["Sensitivity"],
             Specificity = cm$byClass["Specificity"],
             Balanced_Accuracy = cm$byClass["Balanced Accuracy"])
})

# print out the performance of models using different number of PCs
knn_results
```

```
##                    [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## PC                 1         2         3         4         5         6
## Precision          0.4285714 0.375     0.4117647 0.3684211 0.5294118 0.5
## sensitivity        0.1764706 0.3529412 0.4117647 0.4117647 0.5294118 0.4705882
## Specificity        0.9047619 0.7619048 0.7619048 0.7142857 0.8095238 0.8095238
## Balanced_Accuracy  0.5406162 0.557423  0.5868347 0.5630252 0.6694678 0.640056
##                    [,7]      [,8]      [,9]      [,10]
## PC                 7         8         9         10
## Precision          0.6363636 0.6363636 0.6363636 0.375
## sensitivity        0.4117647 0.4117647 0.4117647 0.3529412
## Specificity        0.9047619 0.9047619 0.9047619 0.7619048
## Balanced_Accuracy  0.6582633 0.6582633 0.6582633 0.557423
```

Conclusion

The report analysed the liver patient data set and explored a variety of sophisticated ML model to predict the liver disease status of a given patient. The best performing model used the k-nearest neighbor algorithm on principal components and achieved: 62% precision, 47% sensitivity, 88% specificity and 68% balanced accuracy.