HarvardX: PH125.9x Data Science: Choose Your Own Project Submission

Chang Soo Yen

28 March 2020

1 Introduction

This section introduces the dataset and variables to be used in this project, project goals and key steps taken to achieve the project goals.

1.1 Dataset

The dataset used in this project is obtained from kaggle's ML-friendly Public Datasets: **Indian Liver Patient Records** (https://www.kaggle.com/uciml/indian-liver-patient-records). We will be downloading the relevant packages required in this project as well as preparing the dataset for exploratory purposes.

The coding for this project will be done in R, and the starting code to create the data frame is as follows:

```
# Installation of packages required in this project
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")

# Indian Liver Patient Dataset:
# Either from https://archive.ics.uci.edu/ml/machine-learning-databases/00225/
# or https://www.kaggle.com/uciml/indian-liver-patient-records

# Creation of column names based on the Indian Liver Patient Dataset
# Abbreviations for Medical Terms Used for Better Clarity
colNames <- c("Age", "Sex", "TB", "DB", "ALP", "ALT", "AST", "TP", "AB", "AGR", "Current")

# Assembling dataset together
# Separation of elements in each column
# Renaming of column names
downloadData <- read.table(
"https://archive.ics.uci.edu/ml/machine-learning-databases/00225/Indian%20Liver%20Patient%20Dataset%20(
sep=",", col.names=colNames, header=FALSE)

# Create a new column called 'Status', 1 in 'Current': Disease, 2 in 'Current': No Disease
# Remove 'Current' column
liverData <- downloadData %>%
  mutate(Status = ifelse(Current=="1", "Disease", "No Disease")) %>%
  mutate(Status = as.factor(Status)) %>%
  select("Age", "Sex", "TB", "DB", "ALP", "ALT", "AST", "TP", "AB", "AGR", "Status")
```

Now, we will proceed to explore the various variables in the **liverData** dataset to gain a better understanding of it.

This dataset contains liver disease patient records and non liver disease patient records collected from North East of Andhra Pradesh, India.

= =

Below are the variables in the **liverData** dataset:

Age: Age of the patient

Sex: Gender of the patient

TB: Total Bilirubin (mg/dL), where **elevated** levels may indicate liver damage or disease.

DB: Direct Bilirubin (mg/dL), where **elevated** levels may indicate liver damage or disease.

ALP: Alkaline Phosphatase, where **elevated** levels may indicate liver damage or disease.

ALT: Alanine Aminotransferase, where **elevated** levels may indicate liver damage or disease.

AST: Aspartate Aminotransferase, where **elevated** levels may indicate liver damage or disease.

TP: Total Protein, where **elevated** levels may indicate liver damage or disease.

AB: Albumin, where **low** levels may indicate liver damage or disease.

AGR: Albumin Globulin Ratio, where **low** levels may indicate liver damage or disease.

Status: **Liver Disease** Patient or **non Liver Disease** Patient

= =

1.2 Project Goals

The theme of this project is on health because personally, using Data Science to improve living conditions for others in the future is my goal.

Prevalence of Liver Disease in India is 5-28% and it is a worrying number. Globally, 1.5 million people had Chronic Liver Disease in 2017.This is a pressing concern.

Thus, this project's goals would be to perform some analysis on the **liverData** dataset and adopt some modeling approaches in hopes to predict the possibility of a healthy patient contracting the liver disease in the future (and hence preventing it or seeking early treatment!).

But mainly, a personal goal would be for me to consolidate all the knowledge I have learnt in the Harvardx Data Science course and put them into use here - especially with this dataset relating to health which I am concerned about. It is definite that I will not be able to make a major discovery just with the knowledge I currently have, but I believe in tiny steps! (I will get there one day!)

1.3 Key Steps

In this project, we will be doing data exploration of the **liverData** dataset on a macro and micro level, followed by conducting some modeling to test the **Accuracy, Sensitivity and Specificity** of the approaches. Ultimately, we would analyse them and decide on the most ideal model.

In the context of this project, here are the following definitions for the **Accuracy, Sensitivity and Specificity**:

Accuracy: the overall proportion that is predicted correctly -> the overall proportion of patients predicted to be in the correct status group

Sensitivity: the ability to predict a positive outcome when the actual outcome is positive -> the ability to predict a patient has Liver Disease when they actually have it

Specificity: the ability to not predict a positive outcome when the actual outcome is not positive -> the ability to predict a patient does not have Liver Disease when they actually do not have it

These components are essential in this project as we will get to weigh each component and eventually decide which one is the most important, therefore making a more informed decision instead of simply looking at Accuracy only.

2 Data Exploration and Visualization

This section explores and analyses the **liverData** dataset in a macro then micro level. We will look at the structures of the dataset first, then branch into analysing the individual variables that make up the dataset. From there, observations and relationships will be made to aid in the next section.

2.1 Exploration of Entire Dataset

We will now explore the dataset as a whole.

To begin off, a familiarisation with the **liverData** dataset would be useful in understanding it:

```
head(liverData)
```

```
##   Age    Sex   TB  DB ALP ALT AST  TP  AB  AGR  Status
## 1  65 Female  0.7 0.1 187  16  18 6.8 3.3 0.90 Disease
## 2  62   Male 10.9 5.5 699  64 100 7.5 3.2 0.74 Disease
## 3  62   Male  7.3 4.1 490  60  68 7.0 3.3 0.89 Disease
## 4  58   Male  1.0 0.4 182  14  20 6.8 3.4 1.00 Disease
## 5  72   Male  3.9 2.0 195  27  59 7.3 2.4 0.40 Disease
## 6  46   Male  1.8 0.7 208  19  14 7.6 4.4 1.30 Disease
```

It can be seen that the **liverData** dataset is in tidy format.

The structure of the **liverData** dataset can be viewed with the following code:

```
str(liverData)
```

```
## 'data.frame':    583 obs. of  11 variables:
##  $ Age   : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Sex   : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 2 1 1 2 2 ...
##  $ TB    : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7 ...
##  $ DB    : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2 ...
##  $ ALP   : int  187 699 490 182 195 208 154 202 202 290 ...
##  $ ALT   : int  16 64 60 14 27 19 16 14 22 53 ...
##  $ AST   : int  18 100 68 20 59 14 12 11 19 58 ...
##  $ TP    : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 ...
##  $ AB    : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.4 ...
##  $ AGR   : num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 ...
##  $ Status: Factor w/ 2 levels "Disease","No Disease": 1 1 1 1 1 1 1 1 2 1 ...
```

We see that there are 583 rows and 11 columns. It can also be observed that each row represents the records of one patient.

Before exploring the **liverData** dataset further, we need to check if the dataset contains any "NA" values. If there are, we will have to remove them to prevent erroneous analysis later.

```
any(is.na(liverData))
```

```
## [1] TRUE
```

It seems like there are indeed "NA" values. In that case, we will have to remove them with the following code before proceeding further:

```
liverData <- liverData[complete.cases(liverData),]
```

We have successfully removed rows with some NA values. We will now check the structure of the **liverData** dataset again.

```
str(liverData)
```

```
## 'data.frame':    579 obs. of  11 variables:
##  $ Age   : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Sex   : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 2 1 1 2 2 ...
##  $ TB    : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7 ...
##  $ DB    : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2 ...
##  $ ALP   : int  187 699 490 182 195 208 154 202 202 290 ...
##  $ ALT   : int  16 64 60 14 27 19 16 14 22 53 ...
##  $ AST   : int  18 100 68 20 59 14 12 11 19 58 ...
##  $ TP    : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 ...
##  $ AB    : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.4 ...
##  $ AGR   : num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 ...
##  $ Status: Factor w/ 2 levels "Disease","No Disease": 1 1 1 1 1 1 1 1 2 1 ...
```

Here, we now see 579 rows and 11 columns to work with.

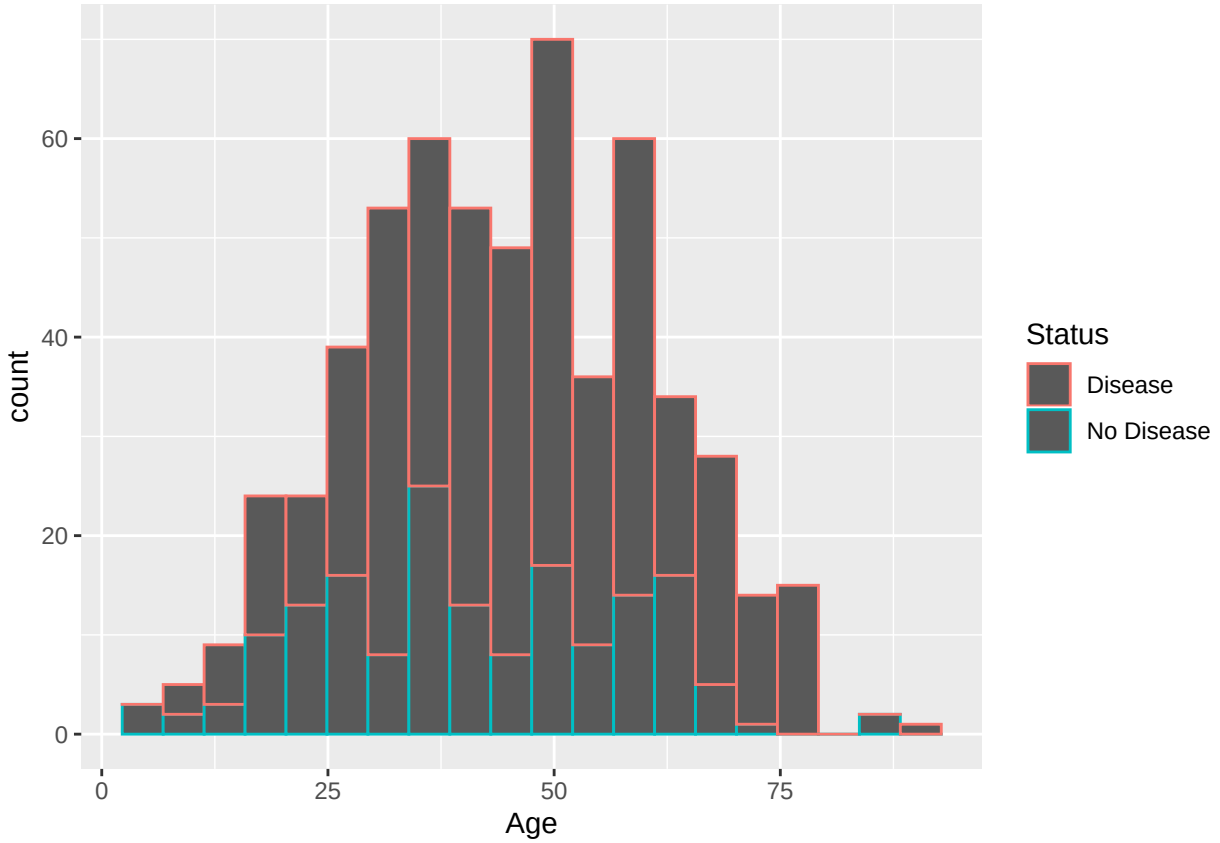
2.2 Exploration of Individual Variables

We will proceed to explore the individual variables one by one. In each variable, we will split observations up based on the Status (Disease / No Disease) of the patient for a clearer view. It will be easier to spot similarities and differences in this way.

2.2.1 Age: Age of the patient

The variable "Age" essentially refers to the respectives ages of the patients. Let's view the proportion of patients in the same age group that has or does not have Liver Disease with the following code:

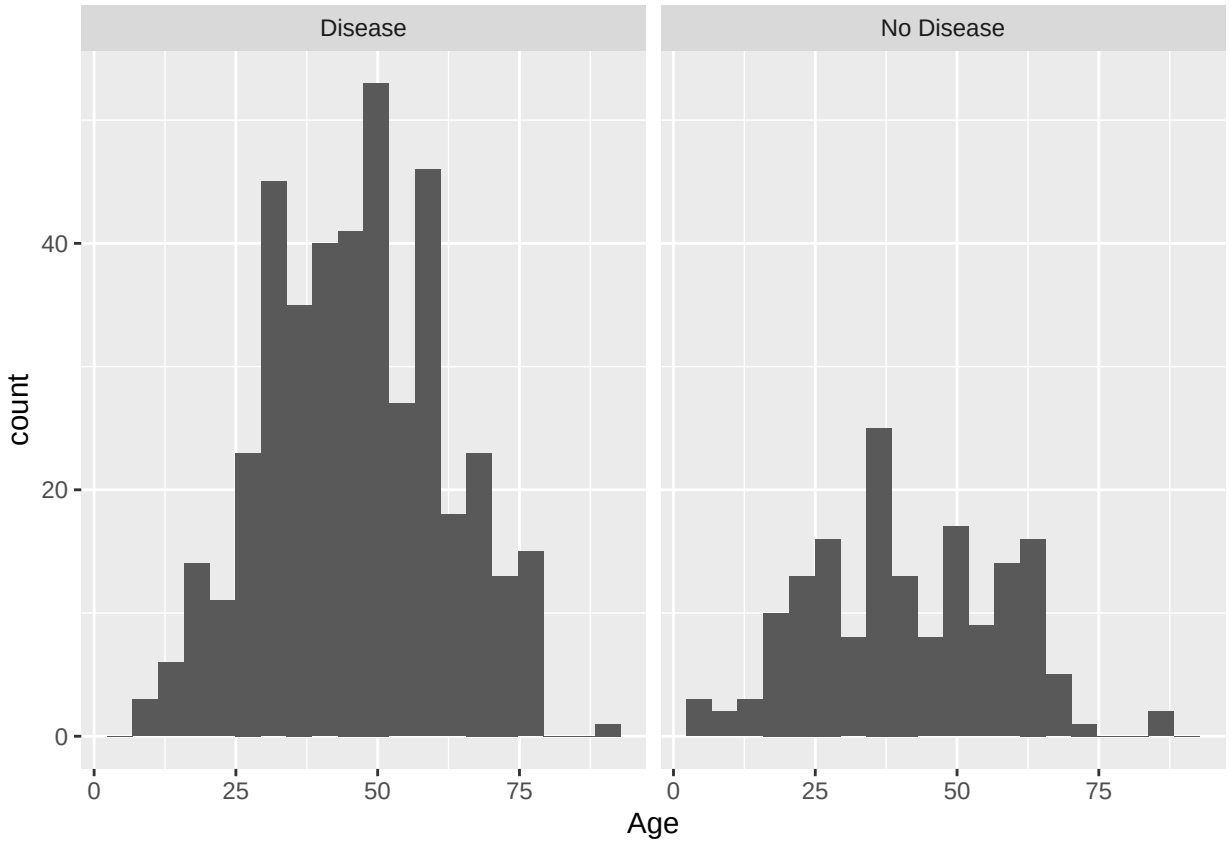
```
liverData %>% ggplot(aes(Age, col=Status)) + geom_histogram(bins=20)
```



There does not seem to be anything too odd, the dataset seems to have a reasonable spread of each age and there are data for those without Liver Disease for almost all age groups.

We will now proceed to divide the spread of patients by age with the same Status variable, and see if we can find anything from there.

```
liverData %>% ggplot(aes(Age)) + geom_histogram(bins=20) + facet_grid(~Status)
```



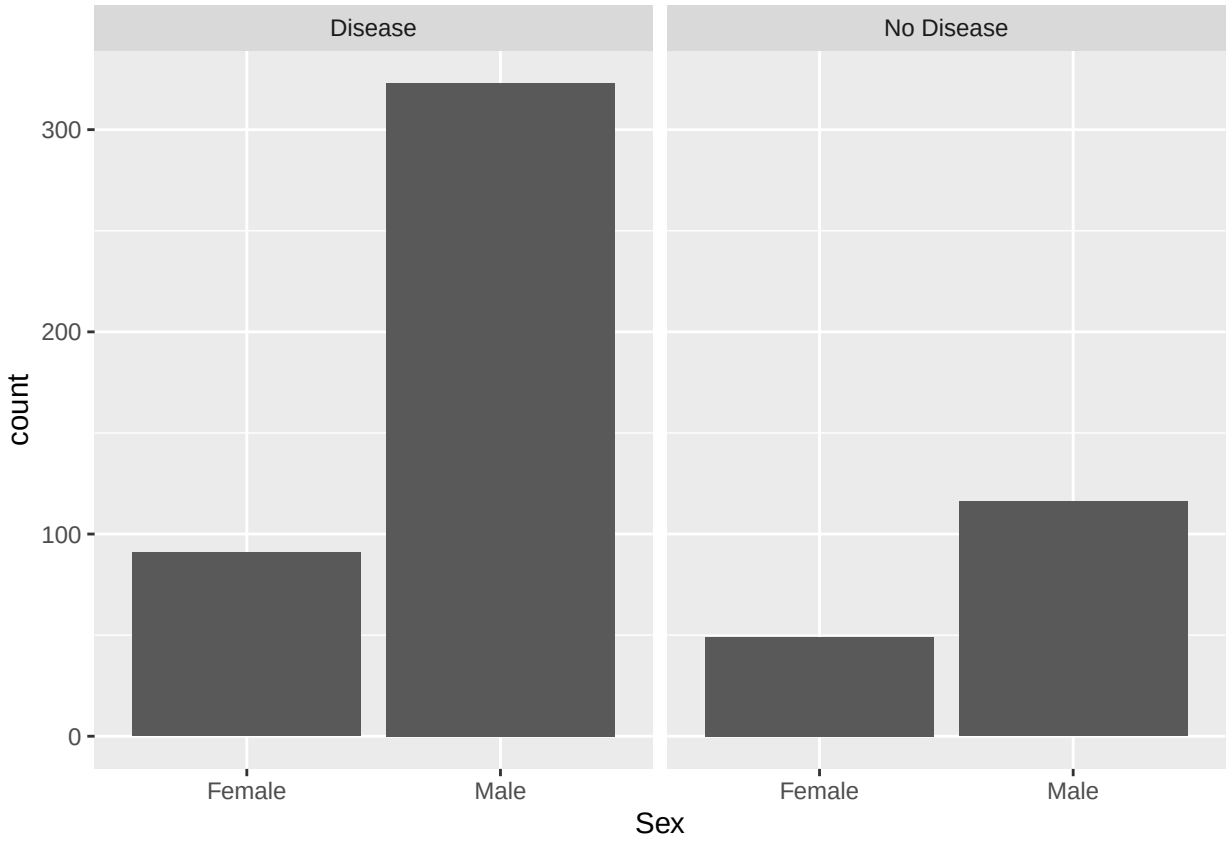
From these two histograms, we can see that both Status groups have relatively similar distributions.

This probably means that for each age group, there is a fair share of people contracting the Liver Disease with the highest amount coming from those around the age of 50 years old.

2.2.2 Sex: Gender of the patient

The variable "Sex" essentially refers to the respectives genders of the patients. Let's view the spread of patients by gender with the Status variable, and see if we can find anything from there.

```
liverData %>% ggplot(aes(Sex)) + geom_bar() + facet_grid(~Status)
```



From these two bar graphs, we can see that both Status groups have relatively similar distributions yet again.

However, we can observe that Males seem to contract the Liver Disease more than Females.

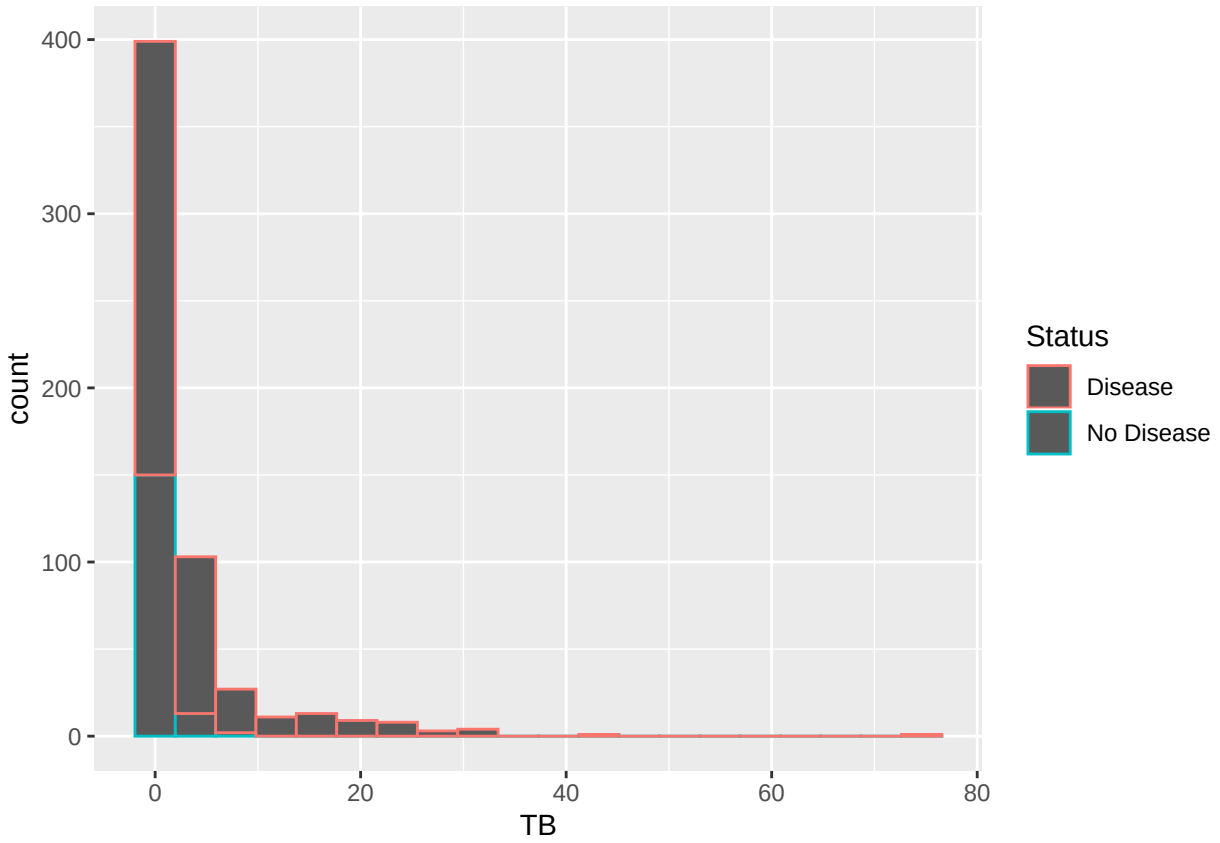
2.2.3 TB: Total Bilirubin (mg/dL)

The variable "TB" essentially refers to the Total Bilirubin levels of the patients. As from what Google describes it to be, Bilirubin is a yellow compound that occurs in the normal catabolic pathway that breaks down heme in vertebrates. This catabolism is a necessary process in the body's clearance of waste products that arise from the destruction of aged or abnormal red blood cells.

Hence, **elevated** levels of TB may indicate liver damage or disease.

Let's view the proportion of patients in the same TB range that has or does not have Liver Disease with the following code:

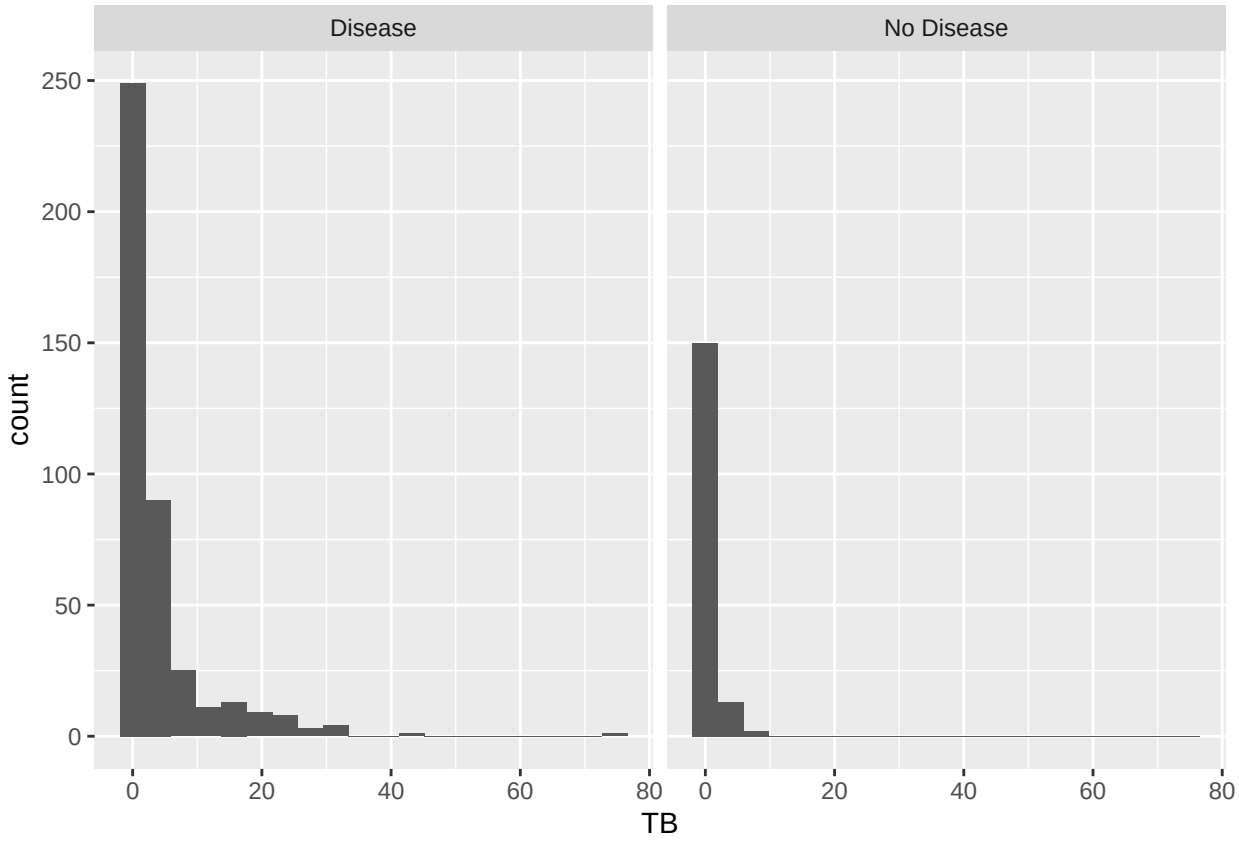
```
liverData %>% ggplot(aes(TB, col=Status)) + geom_histogram(bins=20)
```



What is seen from the above graph is pretty much what I expected to see - some Liver Disease patients having high levels of TB. What I did not think we would see would be the fact that there is a bulk of Liver Disease patients who have low TB levels. This could be due to TB just being an indicator, it is not definite that a Liver Disease patient would have a high TB level as there might be other causes of Liver Disease.

We will now proceed to divide the spread of patients by TB levels with the same Status variable, and see if we can find anything else from there.

```
liverData %>% ggplot(aes(TB)) + geom_histogram(bins=20) + facet_grid(~Status)
```



True enough, this is a clearer view of the previous diagram. It is noticeable that yet again, both Status groups have relatively similar distributions. However, those patients without Liver Disease do not have TB levels crossing 7.3mg/dL, whereas for those patients with Liver Disease, the highest TB level is 75mg/dL.

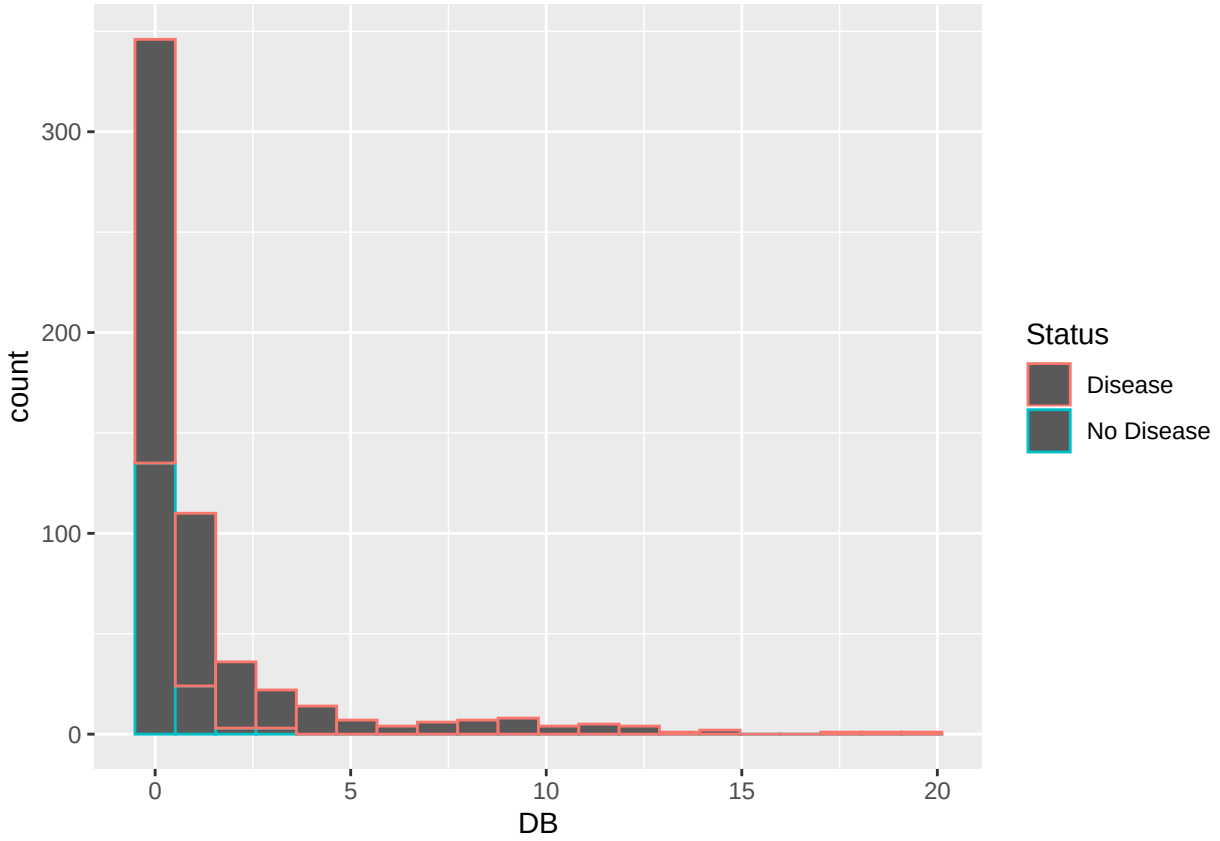
2.2.4 DB: Direct Bilirubin (mg/dL)

The variable "DB" essentially refers to the Direct Bilirubin levels of the patients. As from what Google describes it to be, Direct Bilirubin is a portion of Total Bilirubin and it makes up the water-soluble segments.

Similarly, **elevated** levels of DB may indicate liver damage or disease.

Let's view the proportion of patients in the same DB range that has or does not have Liver Disease with the following code:

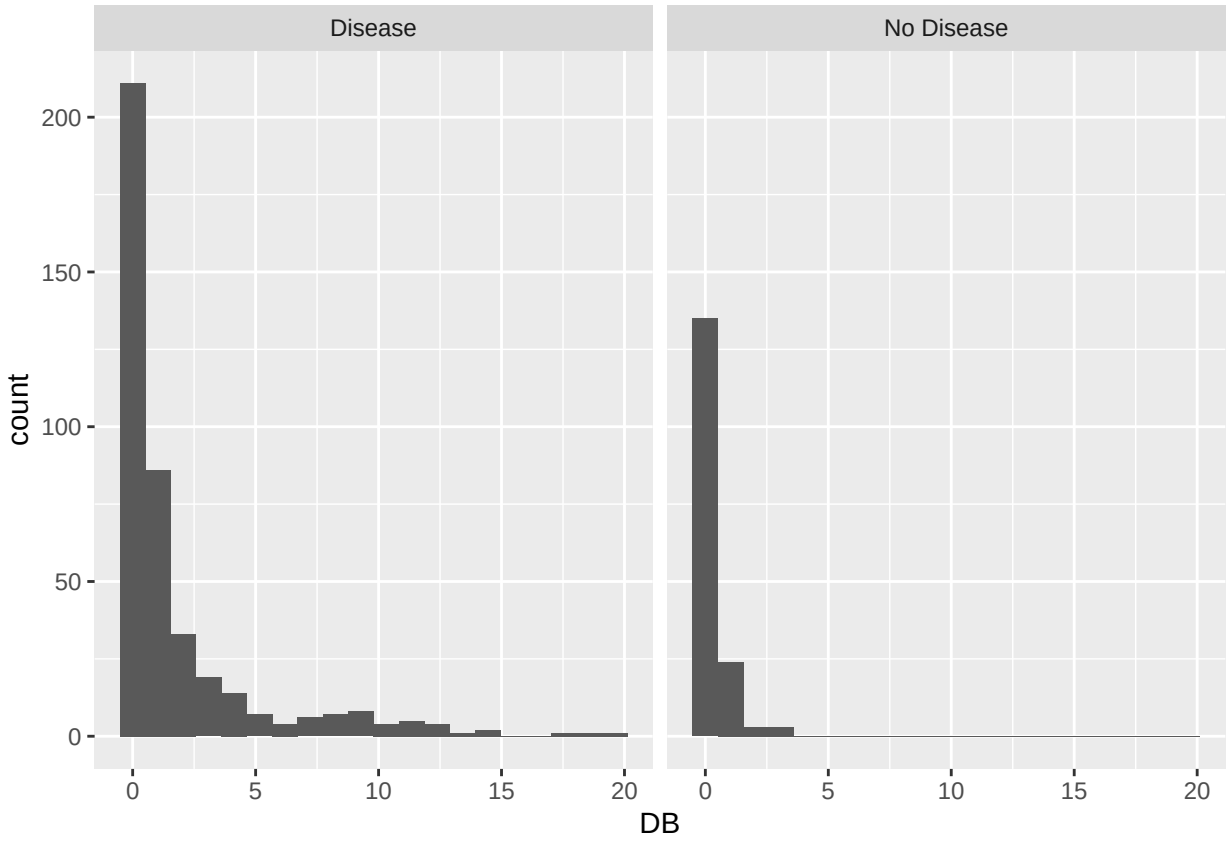
```
liverData %>% ggplot(aes(DB, col=Status)) + geom_histogram(bins=20)
```



What is seen from the above graph is pretty much a similar distribution to the one for TB, which makes sense since they are from the same category and DB is a subset.

We will now proceed to divide the spread of patients by DB levels with the same Status variable, and see if we can find anything else from there.

```
liverData %>% ggplot(aes(DB)) + geom_histogram(bins=20) + facet_grid(~Status)
```



And again, both Status groups have relatively similar distributions. However, those patients without Liver Disease do not have DB levels crossing 3.6mg/dL, whereas for those patients with Liver Disease, the highest DB level is 19.7mg/dL. These observations are similar to that for TB earlier.

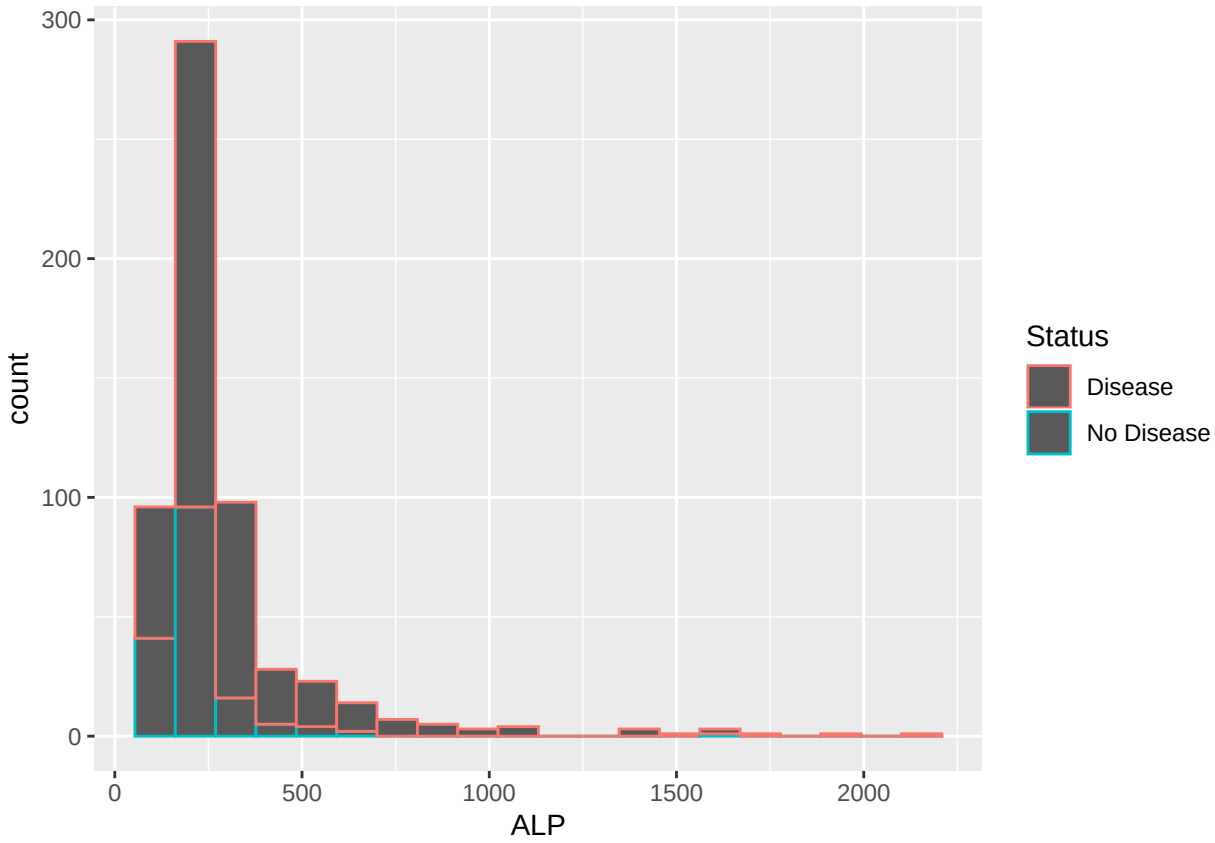
2.2.5 ALP: Alkaline Phosphatase

The variable "ALP" essentially refers to the Alkaline Phosphatase levels of the patients. As from what Google describes it to be, ALP is an enzyme found in several tissues throughout the body. The highest concentrations of ALP are present in the cells that comprise bone and the liver.

Elevated levels of ALP in the blood are most commonly caused by liver disease or bone disorders.

Let's view the proportion of patients in the same ALP range that has or does not have Liver Disease with the following code:

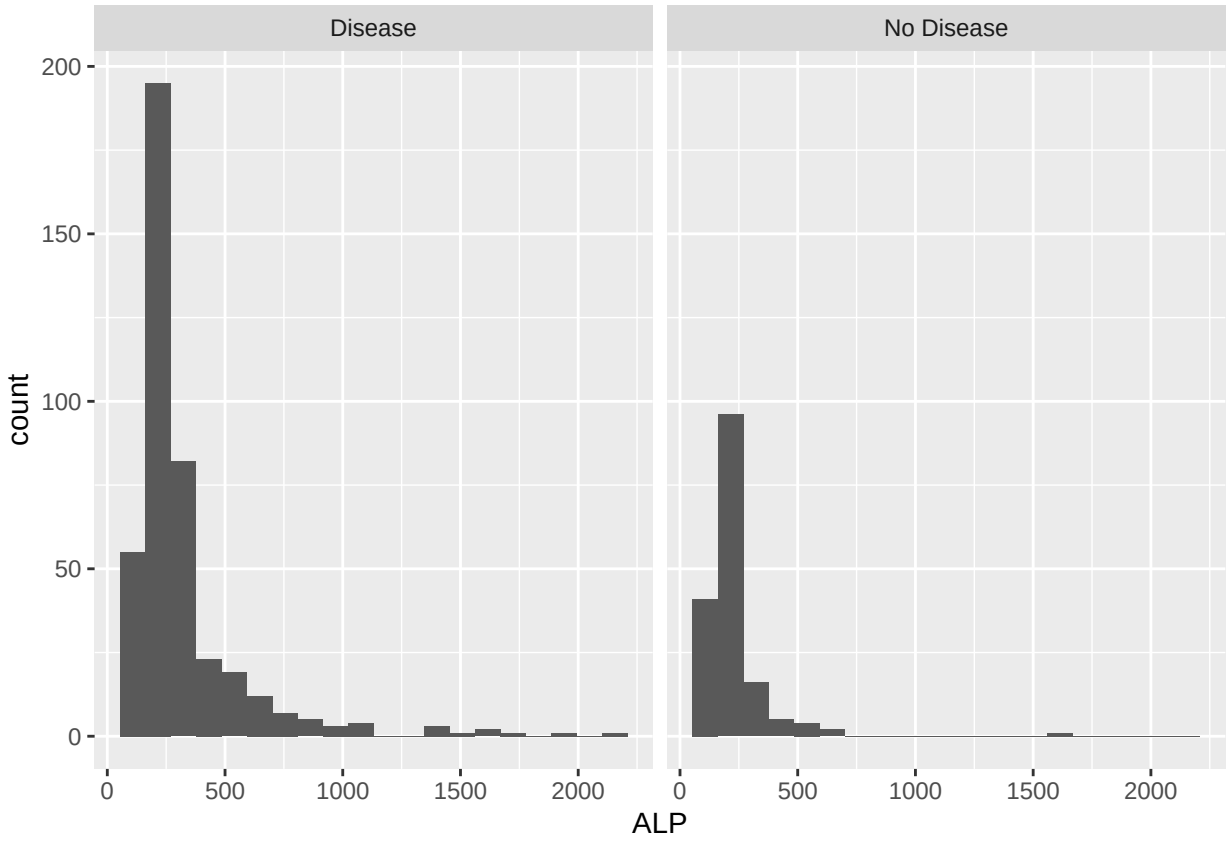
```
liverData %>% ggplot(aes(ALP, col=Status)) + geom_histogram(bins=20)
```



The above graph displays that a bulk of the high ALP levels are all from Liver Disease patients. This is probably expected since elevated levels of ALP do signify a possibility of Liver Disease.

We will now proceed to divide the spread of patients by ALP levels with the same Status variable, and see if we can find anything else from there.

```
liverData %>% ggplot(aes(ALP)) + geom_histogram(bins=20) + facet_grid(~Status)
```



It can be seen that both Status groups have relatively similar distributions. However, those patients without Liver Disease mostly have ALP levels before 750, whereas for those patients with Liver Disease, the highest ALP level is 2110.

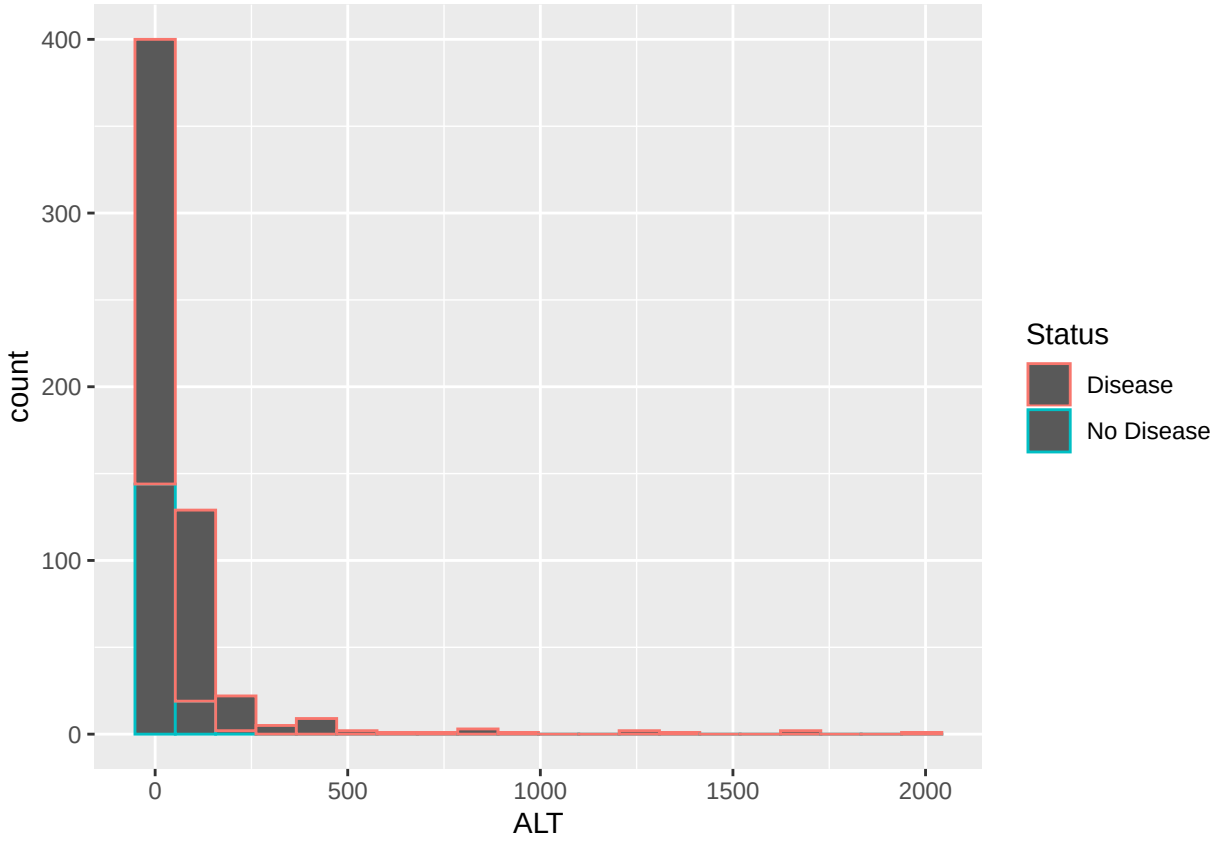
2.2.6 ALT: Alanine Aminotransferase

The variable "ALT" essentially refers to the Alanine Aminotransferase levels of the patients. As from what Google describes it to be, ALT is an enzyme found primarily in the liver and kidney. It was originally referred to as serum glutamic pyruvic transaminase (SGPT). Normally, a low level of ALT exists in the serum.

ALT is **increased** with liver damage and is used to screen for and/or monitor liver disease.

Let's view the proportion of patients in the same ALT range that has or does not have Liver Disease with the following code:

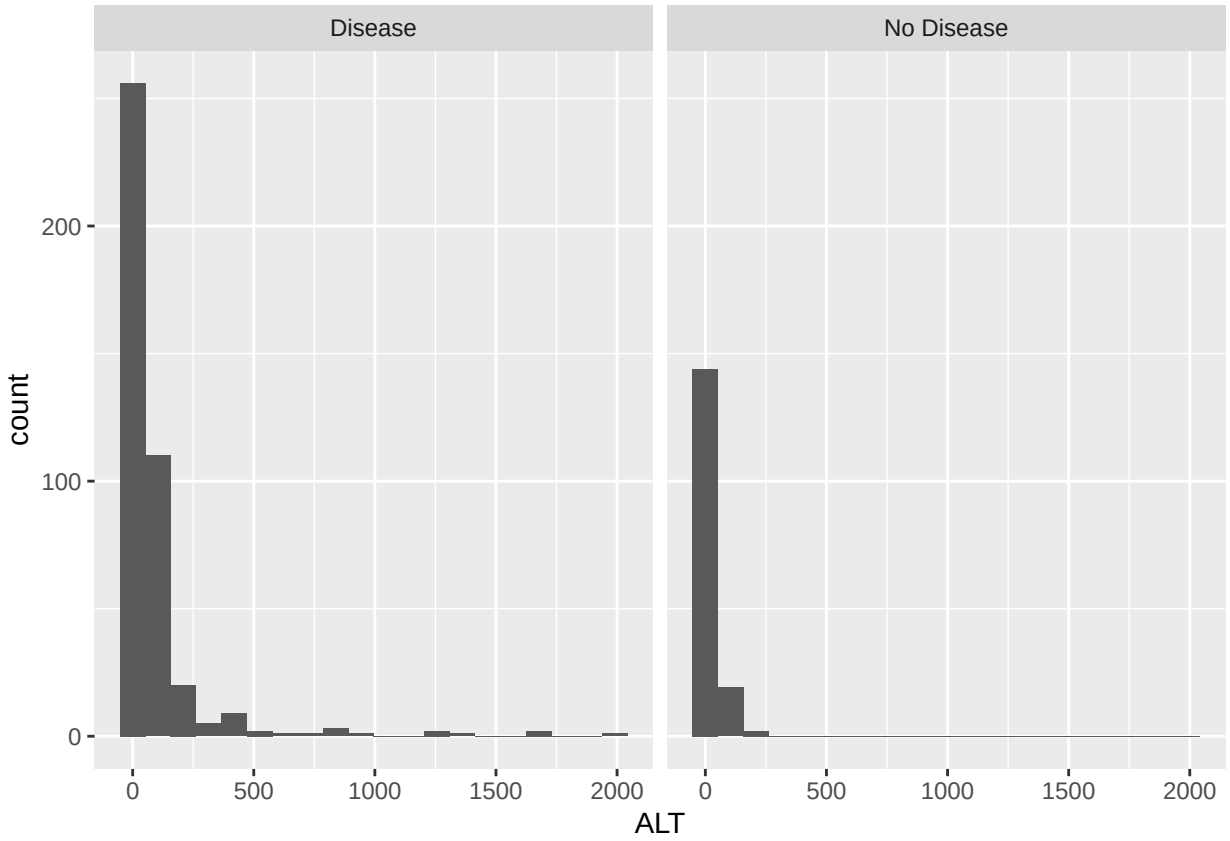
```
liverData %>% ggplot(aes(ALT, col=Status)) + geom_histogram(bins=20)
```



From here, we can roughly see that those patients without Liver Disease do not branch out further along the ALT level like how patients with Liver Disease do. But it is not very clear yet.

Hence, we will now proceed to divide the spread of patients by ALT levels with the same Status variable, and see if we can find anything else from there.

```
liverData %>% ggplot(aes(ALT)) + geom_histogram(bins=20) + facet_grid(~Status)
```



It can be seen that both Status groups have relatively similar distributions yet again, where a bulk have low ALT levels. However, it can be observed that patients without Liver Disease have maximum ALT levels of 181, whereas for those patients with Liver Disease, the highest ALT level is 2000. This is a very big difference, probably why ALT is clinically used to screen and monitor Liver Disease because it is a good indicator.

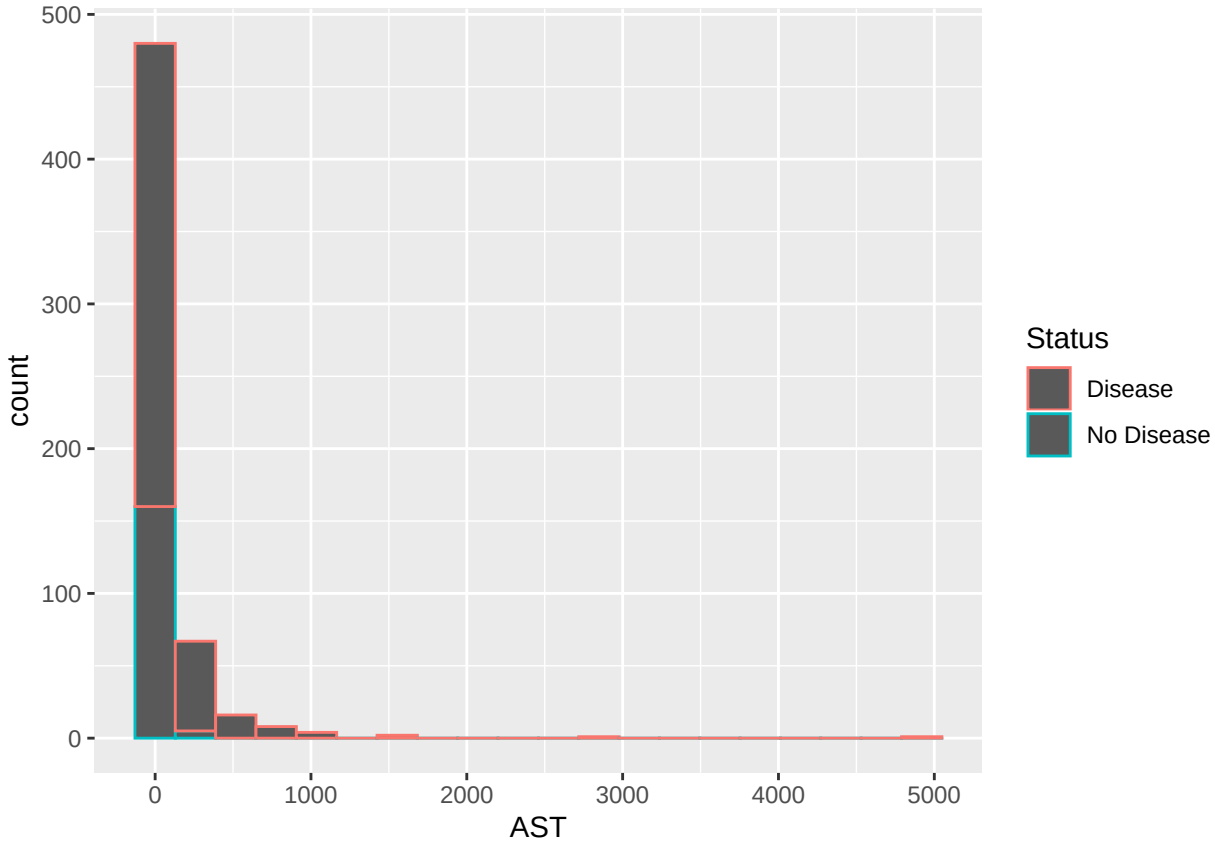
2.2.7 AST: Aspartate Aminotransferase

The variable "AST" essentially refers to the Aspartate Aminotransferase levels of the patients. As from what Google describes it to be, AST is an enzyme found in cells throughout the body but mostly in the heart and liver and, to a lesser extent, in the kidneys and muscles. In healthy individuals, levels of AST in the blood are low. When liver or muscle cells are injured, they release AST into the blood.

Hence, **elevated** levels of AST in the blood could be caused by liver disease.

Let's view the proportion of patients in the same AST range that has or does not have Liver Disease with the following code:

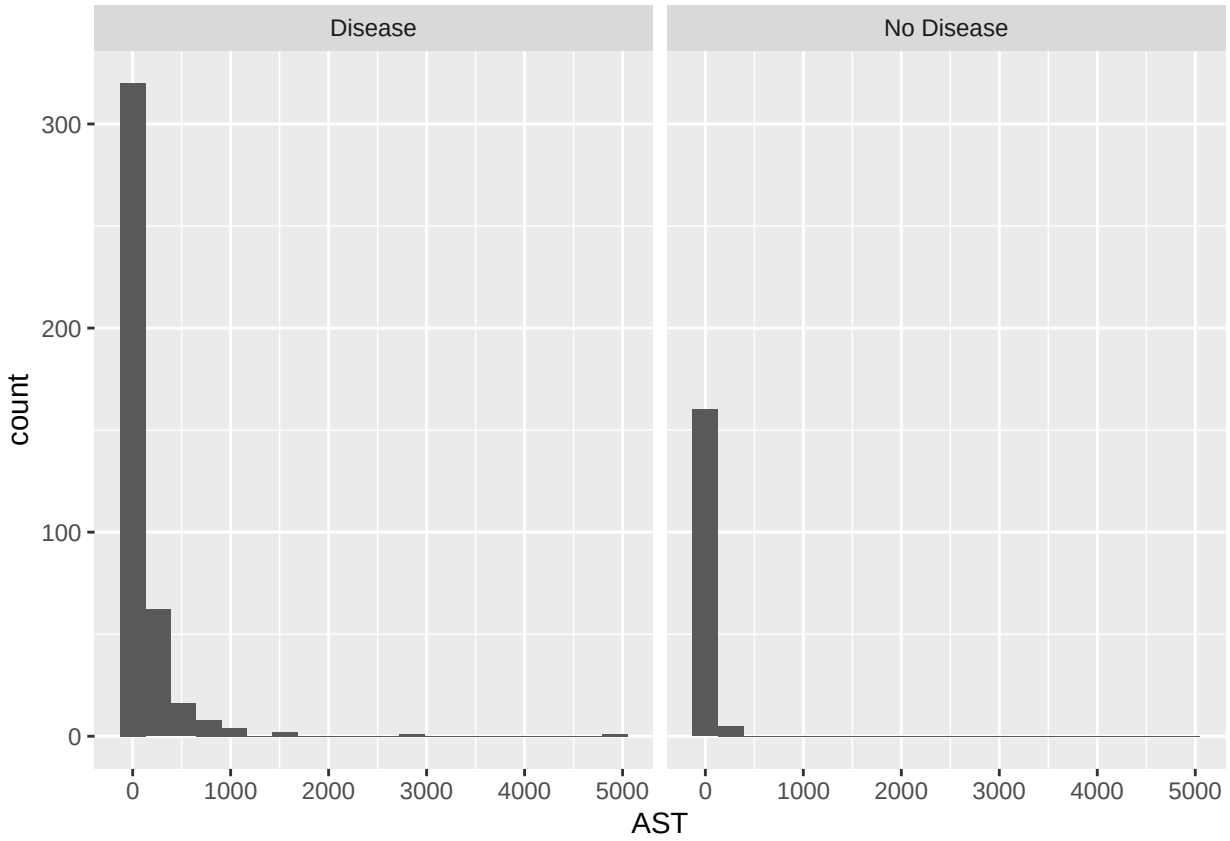
```
liverData %>% ggplot(aes(AST, col=Status)) + geom_histogram(bins=20)
```



Similar to what was seen earlier in the ALT diagram, those patients without Liver Disease do not branch out further along the AST level like how patients with Liver Disease do.

We will now proceed to divide the spread of patients by AST levels with the same Status variable, and see if we can find anything else from there.

```
liverData %>% ggplot(aes(AST)) + geom_histogram(bins=20) + facet_grid(~Status)
```



Also similar to what was seen earlier in the ALT diagram, it can be seen that both Status groups have relatively similar distributions yet again, where a bulk have low AST levels. However, it can be observed that patients without Liver Disease have maximum AST levels of 285, whereas for those patients with Liver Disease, the highest AST level is 4929. This is a relatively big difference, bigger than the one seen for ALT. However, for ALT: more Liver Disease patients branch out to the high levels whereas for AST: less Liver Disease patients branch out but further.

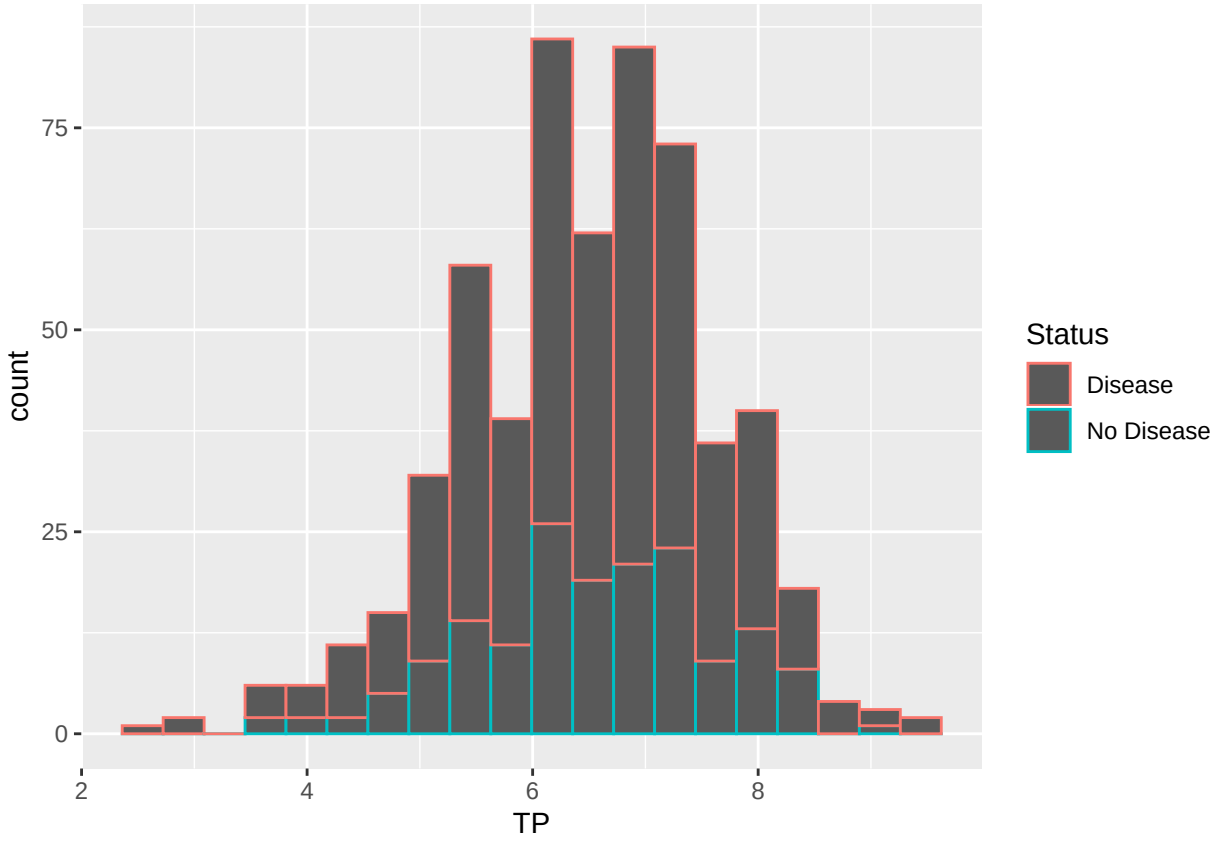
2.2.8 TP: Total Protein

The variable "TP" essentially refers to the Total Protein levels of the patients. As from what Google describes it to be, TP measures the total amount albumin and globulin in your body. High levels of total protein can mean that either albumin and globulin are high. High globulin levels can be from blood diseases such as multiple myeloma or autoimmune diseases such as lupus, kidney disease, or liver disease.

Hence, **elevated** levels of TP could be caused by liver disease.

Let's view the proportion of patients in the same TP range that has or does not have Liver Disease with the following code:

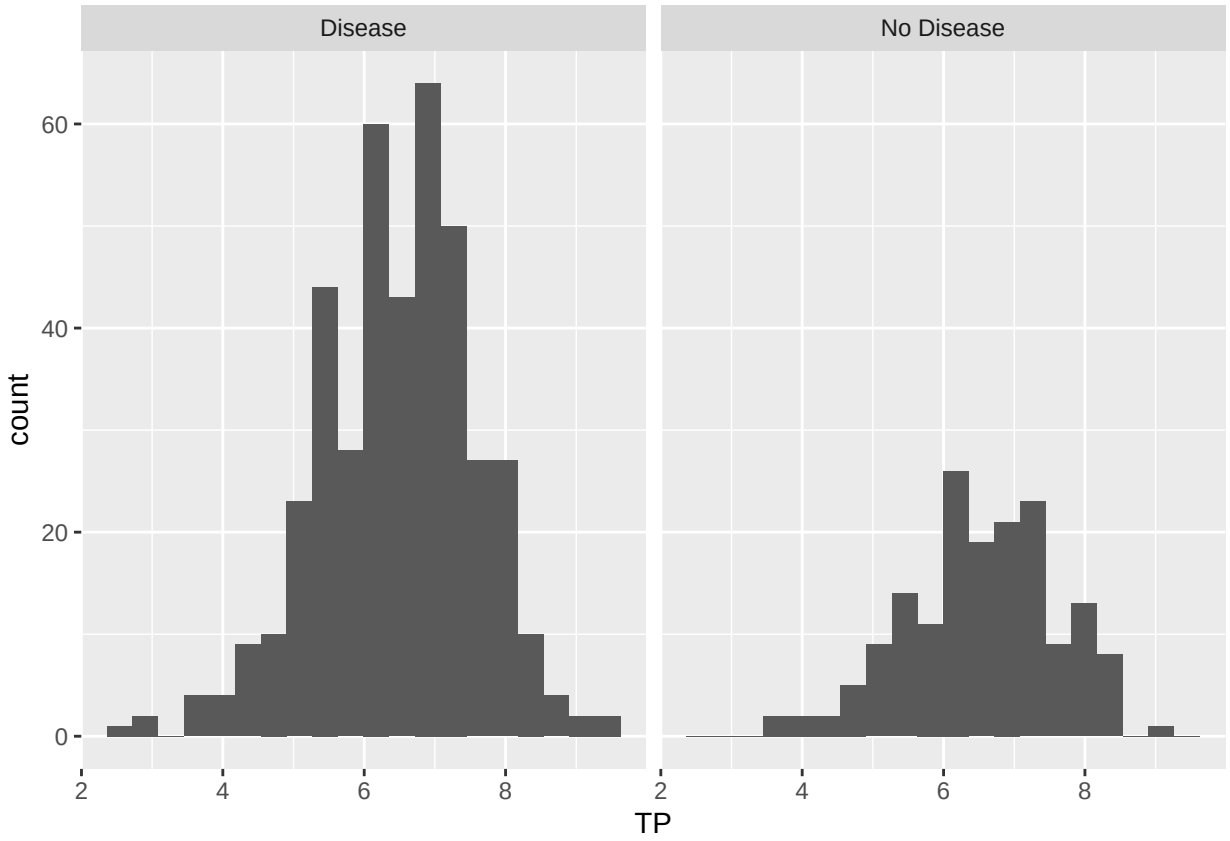
```
liverData %>% ggplot(aes(TP, col=Status)) + geom_histogram(bins=20)
```



From this diagram, we do not see anything particularly useful, apart from the fact that maybe, both distributions for the Red and the Green look similar. Else, TP levels seem to spread out consistently across a range of values.

We will now proceed to divide the spread of patients by TP levels with the same Status variable, and see if we can find anything else from there.

```
liverData %>% ggplot(aes(TP)) + geom_histogram(bins=20) + facet_grid(~Status)
```



This diagram proves the above guesses I made, where TP levels are spread out pretty evenly. However, you can see a bigger range of TP levels in the Liver Disease patients graph compared to those without. Also, the distributions look fairly similar again.

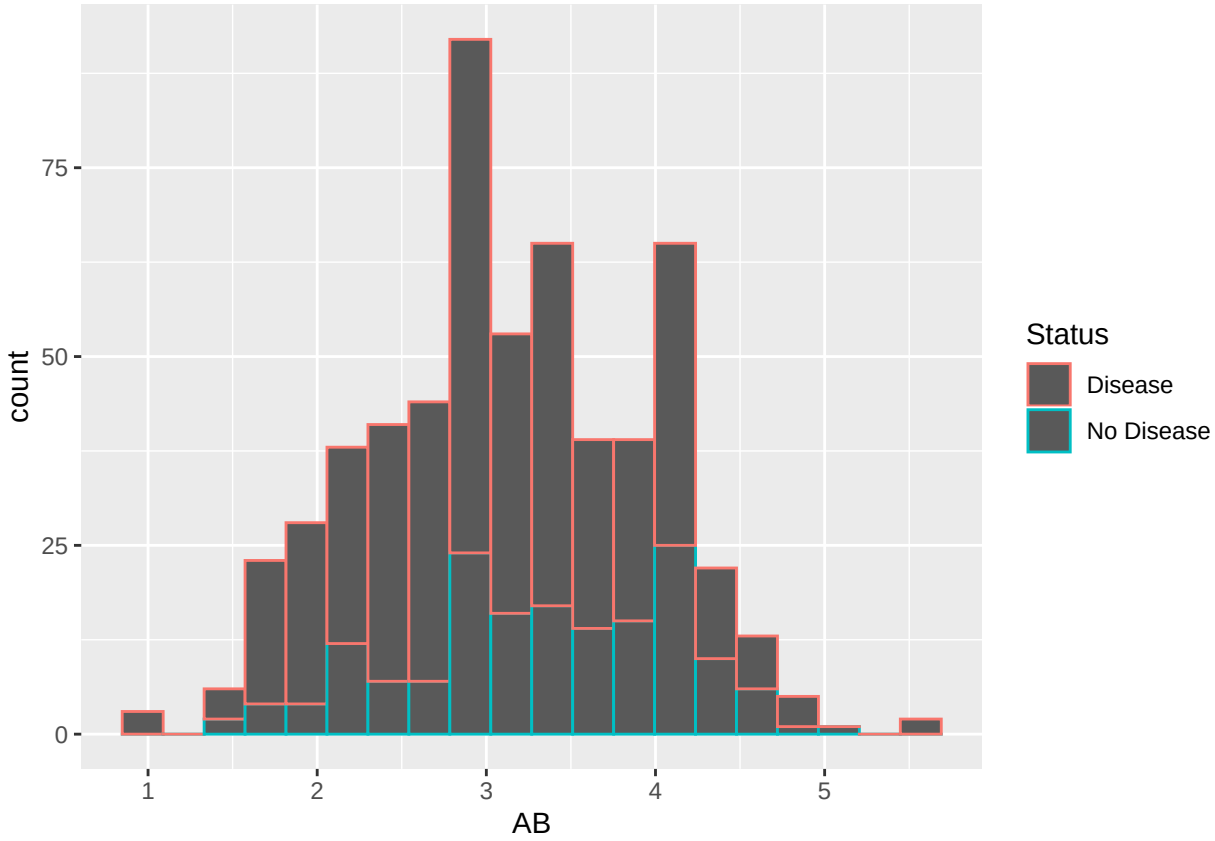
2.2.9 AB: Albumin

The variable "AB" essentially refers to the Albumin levels of the patients. As from what Google describes it to be, AB is a protein made by your liver. Albumin helps keep fluid in your bloodstream so it doesn't leak into other tissues. It also carries various substances throughout your body, including hormones, vitamins, and enzymes.

Low AB levels can indicate a problem with your liver or kidneys.

Let's view the proportion of patients in the same AB range that has or does not have Liver Disease with the following code:

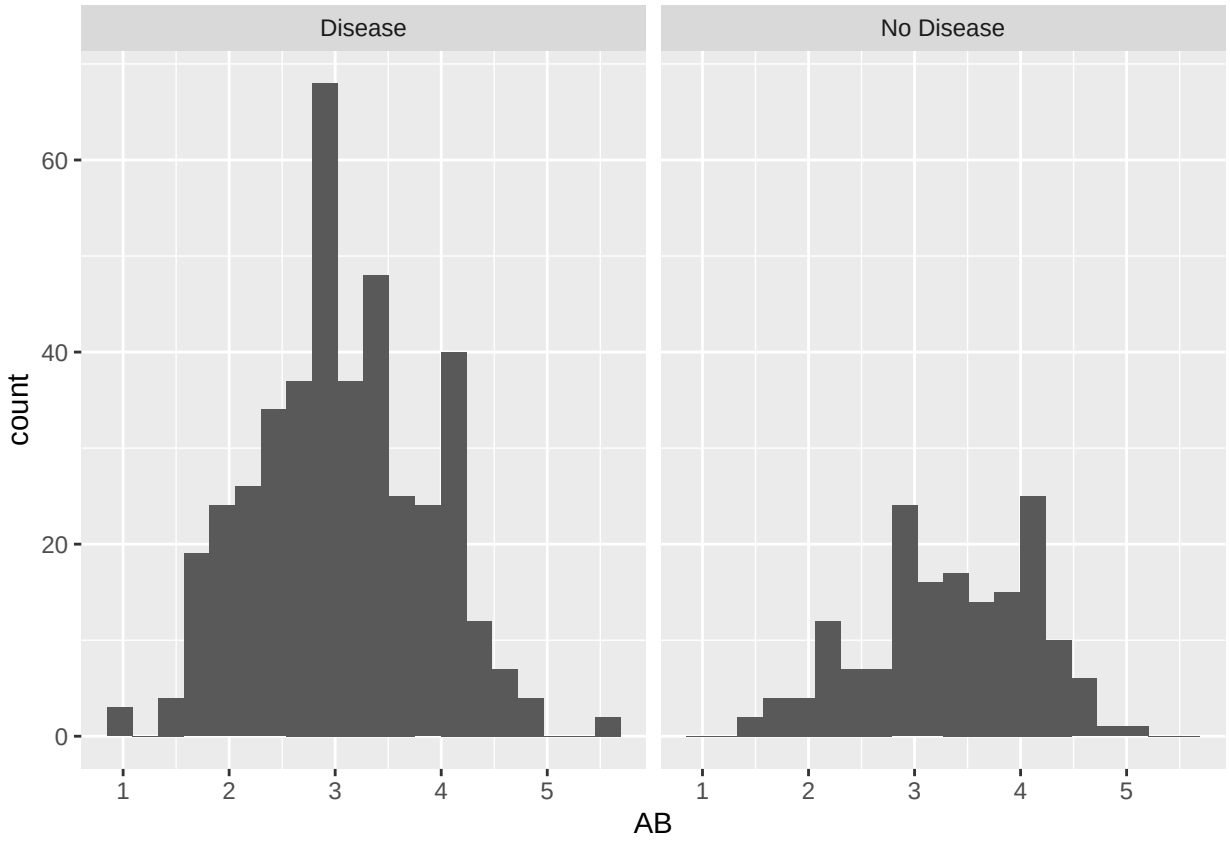
```
liverData %>% ggplot(aes(AB, col=Status)) + geom_histogram(bins=20)
```



Just like from the TP diagram earlier, we do not see anything particularly useful, apart from the fact that maybe, both distributions for the Red and the Green look similar. Else, AB levels seem to spread out consistently across a range of values.

We will now proceed to divide the spread of patients by AB levels with the same Status variable, and see if we can find anything else from there.

```
liverData %>% ggplot(aes(AB)) + geom_histogram(bins=20) + facet_grid(~Status)
```



This diagram proves the above guesses I made, where AB levels are spread out pretty evenly. Also, the distributions look fairly similar again. However, it seems that indeed there is a portion of those with Liver Disease that have relatively low AB levels.

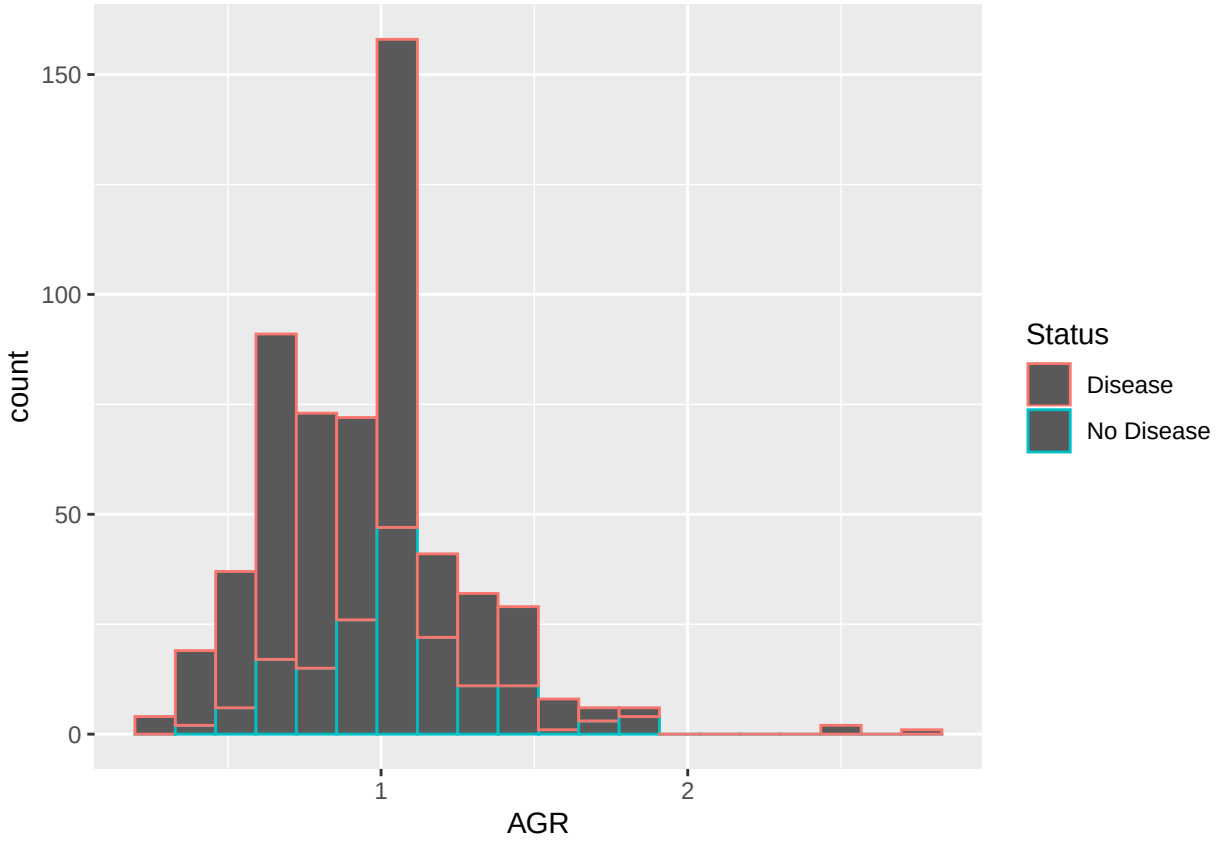
2.2.10 AGR: Albumin Globulin Ratio

The variable "AGR" essentially refers to the Albumin Globulin Ratio of the patients. As from what Google describes it to be, high globulin levels can be from blood diseases such as multiple myeloma or autoimmune diseases such as lupus, kidney disease, or liver disease.

Hence, **low** AGR in the blood could be caused by liver disease.

Let's view the proportion of patients in the same AGR range that has or does not have Liver Disease with the following code:

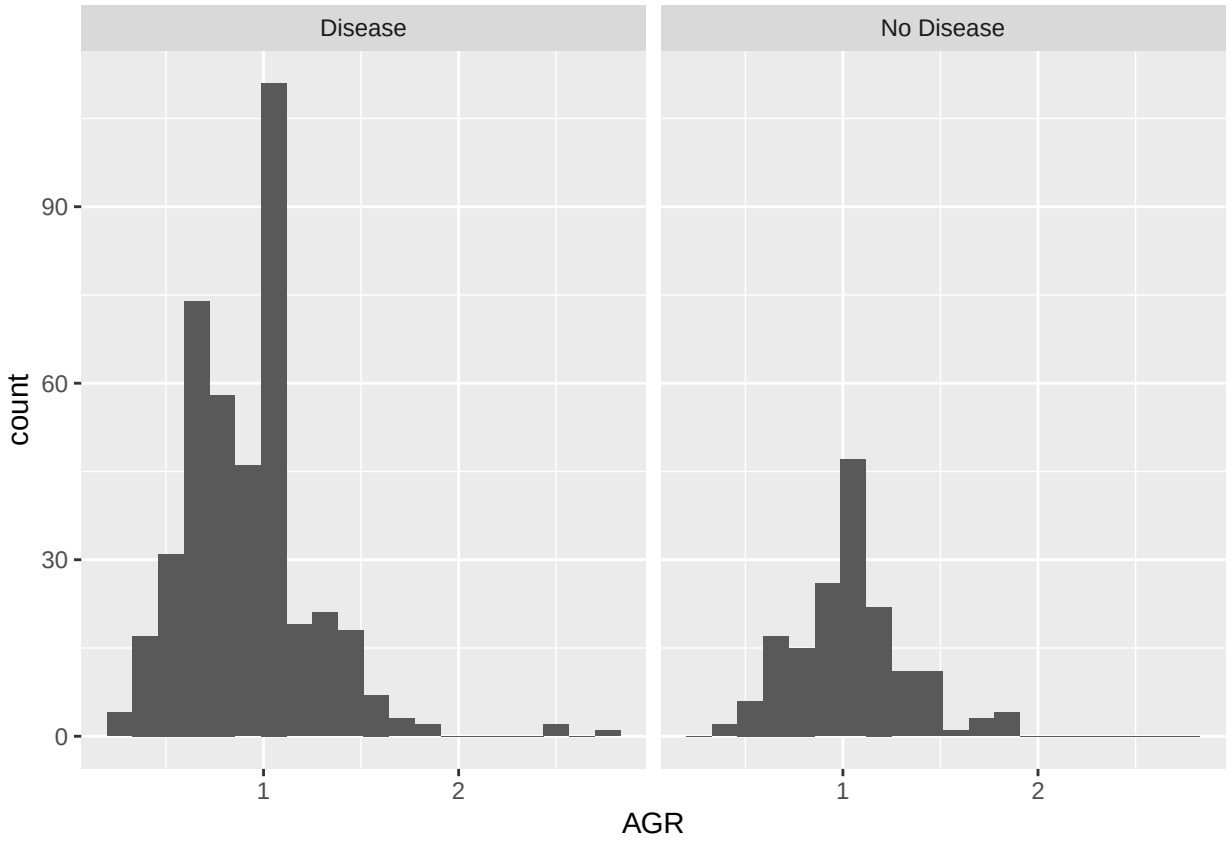
```
liverData %>% ggplot(aes(AGR, col=Status)) + geom_histogram(bins=20)
```



Based on the diagram, it seems similar to the AB diagram earlier.

We will now proceed to divide the spread of patients by AGR levels with the same Status variable, and see if we can find anything else from there.

```
liverData %>% ggplot(aes(AGR)) + geom_histogram(bins=20) + facet_grid(~Status)
```



The 2 histograms seem to share a similar distribution, but it also seems that there are some outliers in the Liver Disease patients' histogram. This could be due to the fact that AGR might not be such a strong indicator after all.

3 Modeling Approaches

This section starts with establishing training and test sets for the upcoming modeling approaches. We then attempt various modeling approaches and in the process of machine learning, we will look out for the readings of the three factors - Accuracy, Sensitivity and Specificity.

3.1 Establishing Training and Test Sets

We start off by establishing the training and test sets for our modeling. We will split the dataset into training and test sets first, followed by a simple analysis from Section 2's findings. We will then refine the training and test sets after considering the findings.

3.1.1 Separating the liverData Dataset

We will separate the **liverData** dataset into one for training and one for testing. In the training set, we will have **train** as well as **temptest**. This is because the final test set should only be used on the select model after modeling. Hence, we will have **train** dataset, **temptest** dataset and lastly **test** dataset.

It will be done with the following code:

```
# test dataset will be 10% of liverData dataset
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y=liverData$Status, times=1, p=0.1, list=FALSE)

# Creation of temptrain (90%) and test (10%) datasets from liverData dataset
temptrain <- liverData[-test_index,]
test <- liverData[test_index,]

# temptest dataset will be 10% of temptrain dataset
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y=temptrain$Status, times=1, p=0.1, list=FALSE)

# Creation of train (90%) and temptest (10%) datasets from temptrain dataset
train <- temptrain[-test_index,]
temptest <- temptrain[test_index,]
```

Now that the separation is complete, we will view the structures of the **train**, **temptest** and **test** datasets.

```
str(train)
```

```
## 'data.frame':    467 obs. of  11 variables:
##  $ Age   : int  65 62 62 58 26 29 17 55 57 72 ...
##  $ Sex   : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 1 2 2 2 2 ...
##  $ TB    : num  0.7 10.9 7.3 1 0.9 0.9 0.9 0.7 0.6 2.7 ...
##  $ DB    : num  0.1 5.5 4.1 0.4 0.2 0.3 0.3 0.2 0.1 1.3 ...
##  $ ALP   : int  187 699 490 182 154 202 202 290 210 260 ...
##  $ ALT   : int  16 64 60 14 16 14 22 53 51 31 ...
##  $ AST   : int  18 100 68 20 12 11 19 58 59 56 ...
##  $ TP    : num  6.8 7.5 7 6.8 7 6.7 7.4 6.8 5.9 7.4 ...
##  $ AB    : num  3.3 3.2 3.3 3.4 3.5 3.6 4.1 3.4 2.7 3 ...
##  $ AGR   : num  0.9 0.74 0.89 1 1 1.1 1.2 1 0.8 0.6 ...
##  $ Status: Factor w/ 2 levels "Disease","No Disease": 1 1 1 1 1 1 2 1 1 1 ...
```

```
str(temptest)
```

```
## 'data.frame':    53 obs. of  11 variables:
##  $ Age   : int  72 63 57 38 57 60 74 66 60 72 ...
##  $ Sex   : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 1 2 2 ...
##  $ TB    : num  3.9 0.9 4 1.1 1.3 5.7 0.6 4.2 0.8 1.7 ...
##  $ DB    : num  2 0.2 1.9 0.3 0.4 2.8 0.1 2.1 0.2 0.8 ...
##  $ ALP   : int  195 194 190 198 259 214 272 159 286 200 ...
##  $ ALT   : int  27 52 45 86 40 412 24 15 21 28 ...
##  $ AST   : int  59 45 111 150 86 850 98 30 27 37 ...
##  $ TP    : num  7.3 6 5.2 6.3 6.5 7.3 5 7.1 7.1 6.2 ...
##  $ AB    : num  2.4 3.9 1.5 3.5 2.5 3.2 2 2.2 4 3 ...
##  $ AGR   : num  0.4 1.85 0.4 1.2 0.6 0.78 0.6 0.4 1.2 0.93 ...
##  $ Status: Factor w/ 2 levels "Disease","No Disease": 1 2 1 1 1 1 1 1 1 1 ...
```

```
str(test)
```

```
## 'data.frame':    59 obs. of  11 variables:
```

```
##  $ Age   : int  46 33 30 17 42 33 37 60 60 32 ...
##  $ Sex   : Factor w/ 2 levels "Female","Male": 2 2 2 1 2 2 2 2 2 2 ...
##  $ TB    : num  1.8 1.6 1.3 0.7 8.9 0.8 1.8 5.8 5.2 12.7 ...
##  $ DB    : num  0.7 0.5 0.4 0.2 4.5 0.2 0.8 2.7 2.4 6.2 ...
##  $ ALP   : int  208 165 482 145 272 198 215 204 168 194 ...
##  $ ALT   : int  19 15 102 18 31 26 53 220 126 2000 ...
##  $ AST   : int  14 23 80 36 61 23 58 400 202 2946 ...
##  $ TP    : num  7.6 7.3 6.9 7.2 5.8 8 6.4 7 6.8 5.7 ...
##  $ AB    : num  4.4 3.5 3.3 3.9 2 4 3.8 3 2.9 3.3 ...
##  $ AGR   : num  1.3 0.92 0.9 1.18 0.5 1 1.4 0.7 0.7 1.3 ...
##  $ Status: Factor w/ 2 levels "Disease","No Disease": 1 2 1 2 1 2 1 1 1 1 ...
```

We can see that the **liverData** dataset has been divided across the 3 new data frames! On to the next step!

3.1.2 Key Findings from Data Exploration

After exploring every single variable in the **liverData** dataset, it is important to note that there are way too many variables to consider. Hence, based on the observations earlier and with further investigation, we will be removing a few from the **train**, **temptest** and **test** datasets before proceeding to create some models with it.

For the purposes of further investigation, we will be making use of the **temptrain** dataset which comprises of both **train** and **temptest** data. This ensures a slightly bigger dataset for investigation, but does not cover the validation **test** dataset also.

3.1.3 The Variables to Keep

Age: The age of patients is an independent variable and crucial in accounting for the demographics of patients. It has no dependent variable and is a common variable used medically to divide patients into select groups (where how long you are alive can be a gauge for various bodily functionalities). Hence, this variable should be kept.

Sex: The sex of patients is also an independent variable and vital in accounting for the demographics of patients. It has no dependent variable and is a common variable used medically to divide patients into select groups (where different genders tend to have different characteristics, habits). Hence, this variable should be kept.

TB: The Total Bilirubin levels of patients is a comprehensive variable that covers the entire Bilirubin level in the body. It is also observed that elevated levels of TB is an indicator for Liver Disease. Hence, this variable should be kept.

ALP: The Alkaline Phosphatase levels of patients are mainly from the human bone and liver. With a high probability that high levels of ALP in the blood are most commonly caused by Liver Disease, this is a variable that should be kept.

ALT: The Alanine Aminotransferase levels of patients are clinically used to screen for and monitor Liver Disease. ALT is increased with liver damage, hence a good variable that should be kept.

AB: The Albumin levels of patients is determined by the amount of protein made by one's liver. AB is directly related to the liver and low AB levels indicate a problem with the liver. Hence, this variable should be kept.

AGR: The Albumin Globulin Ratio of patients is determined by both Albumin and Globulin levels. If Globulin levels are high, low AGR is likely caused by Liver Disease. If Albumin levels are low, low AGR is likely caused by Liver Disease too. Hence, this variable should be kept.

3.1.4 The Variables to Remove

DB: Though the Direct Bilirubin levels of patients when high is also an indicator for Liver Disease, the TB levels is sufficient. To further affirm this decision, we can check if TB and DB are strongly correlated so that the removal of DB would not affect modeling.

We do so with the following code, and will investigate with the **temptrain** dataset, which is 90% of the **liverData** dataset:

```
cor_tb_db <- cor(temptrain$TB, temptrain$DB)
cor_tb_db
```

```
## [1] 0.8537099
```

Since the correlation is 0.854, TB and DB levels have a strong positive correlation. Removing DB levels is hence justified.

AST: Though the Aspartate Aminotransferase levels of patients increase when liver or muscle cells are injured, it is mostly found in the heart, liver, kidney and muscles. For the purposes of this Liver Disease project, it would only be better to remove this variable as it might not be a firm indicator.

Moreover, ALT and AST are both aminotransferase. We will check if ALT and AST are strongly correlated so that the removal of AST would be more justified.

We do so with the following code, and will investigate with the **temptrain** dataset, which is 90% of the **liverData** dataset:

```
cor_alt_ast <- cor(temptrain$ALT, temptrain$AST)
cor_alt_ast
```

```
## [1] 0.7103083
```

Since the correlation is 0.710, ALT and AST levels have a relatively strong positive correlation. Removing AST levels is hence justified.

TP: Through high Total Protein levels might refer to high Golbulin levels which is from Liver Disease, TP levels include both Albumin and Golbulin levels. With AB levels already kept in the dataset, this variable does not need to be kept as TP levels do not directly determine whether a patient has Liver Disease or not.

3.1.5 Final Dataset - Training Set, Test Set

We will now remove the 3 variables - DB, AST and TP from the 3 datasets - **train**, **temptest** and **test**.

```
train <- train %>% select(-c(DB,AST,TP))
temptest <- temptest %>% select(-c(DB,AST,TP))
test <- test %>% select(-c(DB,AST,TP))
```

We can now view the structures of the 3 datasets to confirm that they have been removed.

```
str(train)
```

```
## 'data.frame':    467 obs. of  8 variables:
##  $ Age   : int  65 62 62 58 26 29 17 55 57 72 ...
##  $ Sex   : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 1 2 2 2 2 ...
##  $ TB    : num  0.7 10.9 7.3 1 0.9 0.9 0.9 0.7 0.6 2.7 ...
##  $ ALP   : int  187 699 490 182 154 202 202 290 210 260 ...
##  $ ALT   : int  16 64 60 14 16 14 22 53 51 31 ...
##  $ AB    : num  3.3 3.2 3.3 3.4 3.5 3.6 4.1 3.4 2.7 3 ...
##  $ AGR   : num  0.9 0.74 0.89 1 1 1.1 1.2 1 0.8 0.6 ...
##  $ Status: Factor w/ 2 levels "Disease","No Disease": 1 1 1 1 1 1 2 1 1 1 ...
```

```
str(temptest)
```

```
## 'data.frame':    53 obs. of  8 variables:
##  $ Age   : int  72 63 57 38 57 60 74 66 60 72 ...
##  $ Sex   : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 1 2 2 ...
##  $ TB    : num  3.9 0.9 4 1.1 1.3 5.7 0.6 4.2 0.8 1.7 ...
##  $ ALP   : int  195 194 190 198 259 214 272 159 286 200 ...
##  $ ALT   : int  27 52 45 86 40 412 24 15 21 28 ...
##  $ AB    : num  2.4 3.9 1.5 3.5 2.5 3.2 2 2.2 4 3 ...
##  $ AGR   : num  0.4 1.85 0.4 1.2 0.6 0.78 0.6 0.4 1.2 0.93 ...
##  $ Status: Factor w/ 2 levels "Disease","No Disease": 1 2 1 1 1 1 1 1 1 1 ...
```

```
str(test)
```

```
## 'data.frame':    59 obs. of  8 variables:
##  $ Age   : int  46 33 30 17 42 33 37 60 60 32 ...
##  $ Sex   : Factor w/ 2 levels "Female","Male": 2 2 2 1 2 2 2 2 2 2 ...
##  $ TB    : num  1.8 1.6 1.3 0.7 8.9 0.8 1.8 5.8 5.2 12.7 ...
##  $ ALP   : int  208 165 482 145 272 198 215 204 168 194 ...
##  $ ALT   : int  19 15 102 18 31 26 53 220 126 2000 ...
##  $ AB    : num  4.4 3.5 3.3 3.9 2 4 3.8 3 2.9 3.3 ...
##  $ AGR   : num  1.3 0.92 0.9 1.18 0.5 1 1.4 0.7 0.7 1.3 ...
##  $ Status: Factor w/ 2 levels "Disease","No Disease": 1 2 1 2 1 2 1 1 1 1 ...
```

The 3 variables have been successfully removed from the datasets, so we may now proceed with our modeling approaches.

3.2 Selecting Modeling Methods from Caret Package

The caret package (Classification And Regression Training) is a set of functions that attempt to streamline the process for creating predictive models. This makes modeling way easier as this package has them ready for direct usage.

For this project, we will select 4 models to try on for our dataset.

Naive Bayes: method = 'naive_bayes' Generalized Linear Model: method = 'glm' K-Nearest Neighbours: method = "knn" Random Forest: method = 'rf'

We will elaborate on these chosen models in the upcoming sections. We will also be adopting India's Liver Disease prevalence of 5-28%, and set it at 10% for our investigation.

3.3 Modeling with Naive Bayes

The Naive Bayes is 'naive' because it makes the assumption that features of a measurement are independent of each other. This is naive because it is (almost) never true. However, in a health situation like this, maybe such a 'naive' model will be able to help prevent or predict Liver Disease patients because of the assumption. In this case, "worst case scenarios" might be considered, since the presence of a particular variable is unrelated to the presence of any other variable

We form the Naive Bayes Model with the following code:

```
naives_bayes=train(Status ~ ., data=train, method="naive_bayes")
predicted_status=predict(naives_bayes,newdata=temptest)
results <- confusionMatrix(predicted_status, temptest$Status, prevalence=0.1)
```

Now, we will check the respective Accuracy, Sensitivity and Specificity of the Naives Bayes Model.

```
nb_accuracy <- results$overall["Accuracy"]
nb_accuracy
```

```
##  Accuracy
## 0.7169811
```

```
nb_sensitivity <- results$byClass["Sensitivity"]
nb_sensitivity
```

```
## Sensitivity
##   0.6842105
```

```
nb_specificity <- results$byClass["Specificity"]
nb_specificity
```

```
## Specificity
##         0.8
```

The accuracy of 0.72, sensitivity of 0.68 and specificity of 0.8 are the respective results from the Naive Bayes Model.

We will put the following results in a table for comparison later.

```
options(pillar.sigfig=5)
modelresults <- tibble(method= "Naive Bayes Model", Accuracy=nb_accuracy,
            Sensitivity=nb_sensitivity, Specificity=nb_specificity)
modelresults
```

```
## # A tibble: 1 x 4
##   method            Accuracy Sensitivity Specificity
##   <chr>                <dbl>       <dbl>       <dbl>
## 1 Naive Bayes Model  0.71698     0.68421         0.8
```

3.4 Modeling with Generalized Linear Model

The Generalized Linear Model generalizes linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value. It allows for predictions to be made on the raw cost scale, hence no retransformation is required. This is particularly useful for the healthcare arena.

We form the Generalized Linear Model with the following code:

```
glm=train(Status ~ ., data=train, method="glm")
predicted_status=predict(glm,newdata=temptest)
results <- confusionMatrix(predicted_status, temptest$Status, prevalence=0.1)
```

Now, we will check the respective Accuracy, Sensitivity and Specificity of the Generalized Linear Model.

```
glm_accuracy <- results$overall["Accuracy"]
glm_accuracy
```

```
## Accuracy
## 0.754717
```

```
glm_sensitivity <- results$byClass["Sensitivity"]
glm_sensitivity
```

```
## Sensitivity
##           1
```

```
glm_specificity <- results$byClass["Specificity"]
glm_specificity
```

```
## Specificity
##   0.1333333
```

The accuracy of 0.75, sensitivity of 1 and specificity of 0.13 are the respective results from the Generalized Linear Model.

We will put the following results in a table for comparison later.

```
options(pillar.sigfig=5)
modelresults <- bind_rows(modelresults, tibble(method= "Generalized Linear Model",
  Accuracy=glm_accuracy, Sensitivity=glm_sensitivity, Specificity=glm_specificity))
modelresults
```

```
## # A tibble: 2 x 4
##   method                    Accuracy Sensitivity Specificity
##   <chr>                        <dbl>       <dbl>       <dbl>
## 1 Naive Bayes Model          0.71698     0.68421     0.8
## 2 Generalized Linear Model   0.75472     1           0.13333
```

3.5 Modeling with K-Nearest Neighbours

The K-Nearest Neighbours Model is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It assumes that similar things exist in close proximity. In other words, similar things are near to each other. Hopefully, KNN will be useful in predicting Liver Disease patients.

We form the K-Nearest Neighbours Model with the following code:

```
knn=train(Status ~ ., data=train, method="knn")
predicted_status=predict(knn,newdata=temptest)
results <- confusionMatrix(predicted_status, temptest$Status, prevalence=0.1)
```

Now, we will check the respective Accuracy, Sensitivity and Specificity of the K-Nearest Neighbours Model.

```
knn_accuracy <- results$overall["Accuracy"]
knn_accuracy
```

```
##  Accuracy
## 0.6603774
```

```
knn_sensitivity <- results$byClass["Sensitivity"]
knn_sensitivity
```

```
## Sensitivity
##   0.8947368
```

```
knn_specificity <- results$byClass["Specificity"]
knn_specificity
```

```
## Specificity
##  0.06666667
```

The accuracy of 0.66, sensitivity of 0.89 and specificity of 0.07 are the respective results from the K-Nearest Neighbours Model.

We will put the following results in a table for comparison later.

```
options(pillar.sigfig=5)
modelresults <- bind_rows(modelresults, tibble(method= "K-Nearest Neighbours Model",
  Accuracy=knn_accuracy, Sensitivity=knn_sensitivity, Specificity=knn_specificity))
modelresults
```

```
## # A tibble: 3 x 4
##   method                     Accuracy Sensitivity Specificity
##   <chr>                         <dbl>       <dbl>       <dbl>
## 1 Naive Bayes Model           0.71698     0.68421    0.8
## 2 Generalized Linear Model    0.75472     1          0.13333
## 3 K-Nearest Neighbours Model  0.66038     0.89474    0.066667
```

3.6 Modeling with Random Forest

The Random Forest Model is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

We form the Random Forest Model with the following code:

```
rf=train(Status ~ ., data=train, method="rf")
predicted_status=predict(rf,newdata=temptest)
results <- confusionMatrix(predicted_status, temptest$Status, prevalence=0.1)
```

Now, we will check the respective Accuracy, Sensitivity and Specificity of the Random Forest Model.

```
rf_accuracy <- results$overall["Accuracy"]
rf_accuracy
```

```
##  Accuracy
## 0.7924528
```

```
rf_sensitivity <- results$byClass["Sensitivity"]
rf_sensitivity
```

```
## Sensitivity
##   0.9473684
```

```
rf_specificity <- results$byClass["Specificity"]
rf_specificity
```

```
## Specificity
##         0.4
```

The accuracy of 0.79, sensitivity of 0.95 and specificity of 0.4 are the respective results from the Random Forest Model.

We will put the following results in a table for comparison later.

```
options(pillar.sigfig=5)
modelresults <- bind_rows(modelresults, tibble(method= "Random Forest Model",
  Accuracy=rf_accuracy, Sensitivity=rf_sensitivity, Specificity=rf_specificity))
modelresults
```

```
## # A tibble: 4 x 4
##   method                     Accuracy Sensitivity Specificity
##   <chr>                         <dbl>       <dbl>       <dbl>
## 1 Naive Bayes Model           0.71698     0.68421    0.8
## 2 Generalized Linear Model    0.75472     1          0.13333
## 3 K-Nearest Neighbours Model  0.66038     0.89474    0.066667
## 4 Random Forest Model         0.79245     0.94737    0.4
```

4 Results

This section summarizes the various modeling approaches adopted in the previous section, does comparisons across the various approaches then concludes with the 'best' model to be used for the **liverData** dataset.

4.1 Modeling Results and Performance

We will begin by reviewing the Accuracy, Sensitivity and Specificity of individual models as well as make comparisons across models.

We can view the results with the following code:

```
modelresults
```

```
## # A tibble: 4 x 4
##   method                      Accuracy Sensitivity Specificity
##   <chr>                          <dbl>       <dbl>       <dbl>
## 1 Naive Bayes Model            0.71698     0.68421    0.8
## 2 Generalized Linear Model     0.75472     1          0.13333
## 3 K-Nearest Neighbours Model   0.66038     0.89474    0.066667
## 4 Random Forest Model          0.79245     0.94737    0.4
```

To rank the 3 components in each model,

Naive Bayes Model: Specificity (0.8), Accuracy (0.72), Sensitivity (0.68) Mostly high for everything.

Generalized Linear Model: Sensitivity (1), Accuracy (0.75), Specificity (0.13) Exceptionally high Sensitivity, high Accuracy but very low Specificity.

K-Nearest Neighbours Model: Sensitivity (0.89), Accuracy (0.66), Specificity (0.07) Mostly high except exceptionally low Specificity.

Random Forest Model: Sensitivity (0.95), Accuracy (0.79), Specificity (0.4) Mostly high except low Specificity.

We can see that **only** the Naive Bayes Model has Specificity as its highest component. The Random Forest Model has the **highest Accuracy** across all 4 models, while the Naive Bayes Model has the **highest Specificity** and the Generalized Linear Model has the **highest Sensitivity**. Overall, the Model with the **highest Mean** would be the Naive Bayes model followed swiftly by the Random Forest Model.

4.2 'Best' Model

Let's have a recap on the definitions defined earlier:

= Accuracy: the overall proportion that is predicted correctly -> **the overall proportion of patients predicted to be in the correct status group**

Sensitivity: the ability to predict a positive outcome when the actual outcome is positive -> **the ability to predict a patient has Liver Disease when they actually have it**

Specificity: the ability to not predict a positive outcome when the actual outcome is not positive -> **the ability to predict a patient does not have Liver Disease when they actually do not have it** =

Accuracy plays a part in weighing the effectiveness of overall decision making.

Sensitivity on the other hand, helps to ensure that Liver Disease patients get the help they need.

Specificity then helps to ensure treatement is not provided to non Liver Disease patients and instead saved for those who need it.

If we chose the Generalized Linear Model which has a perfect score for Sensitivity, it would be compromising Specificity. This also means that the K-Nearest Neighbours Model would be out as it has the same results as the Generalized Linear Model but even lower Specificity.

It is difficult to decide if the Random Forest Model or the Naive Bayes Model should be selected. The Naive Bayes Model has more promising number with the **train** dataset, but in the bigger picture, hospitals with large amounts of data would probably benefit more from the Random Forest Model which suits for massive data.

Hence, the 'best' model for this project would be concluded to be the Random Forest Model.

We will now use it to test the final Accuracy, Sensitivity and Specificity using the **test** dataset instead of the **temptest** dataset.

```
rf=train(Status ~ ., data=train, method="rf")
predicted_status=predict(rf,newdata=test)
results <- confusionMatrix(predicted_status, test$Status, prevalence=0.1)

rf_accuracy <- results$overall["Accuracy"]
rf_accuracy
```

```
##  Accuracy
## 0.7627119
```

```
rf_sensitivity <- results$byClass["Sensitivity"]
rf_sensitivity
```

```
## Sensitivity
##   0.9047619
```

```
rf_specificity <- results$byClass["Specificity"]
rf_specificity
```

```
## Specificity
##   0.4117647
```

Our final results give us an Accuracy of 0.763, Sensitivity of 0.905, and Specificity of 0.412.

5 Conclusion

This section concludes what has been done in the entire project, as well as discusses the limitations of the project and future work ideas.

5.1 Summary

In this project, we analysed the **Indian Liver Patient Records** and eventually selected the Random Forest Model as the 'best' Model for the purpose of investigating blood records.

Through data visualisation, data cleaning and data exploration, we figured that some variables could be removed from the dataset. We then proceeded to try out different modeling approaches, looking at the Accuracy, Sensitivity and Specificity as methods of reasoning.

5.2 Limitations and Future Work

More could definitely be done, this is just the beginning.

Limitations of this project such as the sample size certainly limited the reliability of the analysis. Additionally, tastes and preferences of patients as well as past health records could also contribute to more detailed analysis. This is because Liver Disease can be induced by one's lifestyle, could be genetic and more.

There are many factors to consider in the healthcare arena, hence this project is not the end - it was only a first step I took. In the future, with more knowledge after university, perhaps I would look back at this dataset again, or similar but bigger datasets. The future is uncertain, but with determination nothing is impossible.

6 Operating System

Following is the operating system utilised for this project:

```
version
```

```
##                _
## platform       x86_64-apple-darwin15.6.0
## arch           x86_64
## os             darwin15.6.0
## system         x86_64, darwin15.6.0
## status
## major          3
## minor          6.1
## year           2019
## month          07
## day            05
## svn rev        76782
## language       R
## version.string R version 3.6.1 (2019-07-05)
## nickname       Action of the Toes
```