

Deploying Machine Learning Techniques to identify Liver Patients

Jamien Lim

12/3/2021

Contents

1	Introduction	1
2	Methodology	2
2.1	Data Exploration	2
2.1.1	Downloading of data	2
2.1.2	Splitting dataset into train and test sets	3
2.1.3	Exploration of data	3
2.1.4	Correlation of Predictors	12
2.2	Machine Learning Models	13
2.2.1	K-Nearest Neighbours	13
2.2.2	Linear Classifier	14
2.2.3	Logistic Regression	15
3	Results	16
4	Conclusion	17

1 Introduction

Liver disease one of the most common cause of premature death in UK and accounts for approximately 2 million deaths per year worldwide. One of the reason is due to late diagnosis of liver disease, when intervention becomes less effective. Moreover, the complexity of liver disease makes it challenging to be detected early. In recent years, contribution of early diagnosis to avoidable morbidity and mortality has been recognised and more focus has been garnered. Even though liver disease is now promoted as a priority by various healthcare groups, but the challenge in detecting it early and accurately still remains. Therefore, this is a motivation to develop a novel approach in aiding human diagnosis with regards to the detection of liver patients.

With that in mind, the aim of this project is to investigate if incorporating machine learning (ML) technique will aid the blood test results in identifying patients that are similar to patients receiving care from hepatologist. The hypothesis of this projects will be that the combination of both ML and blood test results will benefit healthcare groups in identifying liver patients efficiently and effectively.

In this project, the Indian Liver Patient Dataset from UCI Machine Learning Repository will be used to develop the model. The dataset includes blood test results from both liver patients and control group, thereby allowing us to compare and derive a model from it. Firstly, dataset of interest will be explored and formatted. Next, some ML methods are performed to develop the appropriate model for this project. Lastly, these results will then be compared to derive the prediction model for the aim of this project.

2 Methodology

2.1 Data Exploration

2.1.1 Downloading of data

The data is first downloaded from UCI Repository and formatted into a data frame with its column defined accordingly to the dataset information provided.

```
colnames <- c('age', # Age of the patient
              'sex', # Sex of the patient
              'tb', # Total Bilirubin
              'db', # Direct Bilirubin
              'alkphos', # Alkaline Phosphatase
              'sgpt', # Alanine Aminotransferase
              'sgot', # Aspartate Aminotransferase
              'tp', # Total Proteins
              'alb', # Albumin
              'ag', # Ratio Albumin and Globulin Ratio
              'outcome') # Selector field used to split the data into two sets

data <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/00225/Indian%20Liver%20Patient%20Data.csv",
                  sep=',',
                  header=FALSE,
                  col.names=colnames)

data <- subset(data, complete.cases(data))
data <- data %>%
  mutate(outcome = as.character(outcome)) %>%
  mutate(outcome = replace(outcome, outcome == '1', 'Disease')) %>%
  mutate(outcome = replace(outcome, outcome == '2', 'Normal')) %>%
  mutate(outcome = as.factor(outcome))

head(data)
```

##	age	sex	tb	db	alkphos	sgpt	sgot	tp	alb	ag	outcome
## 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

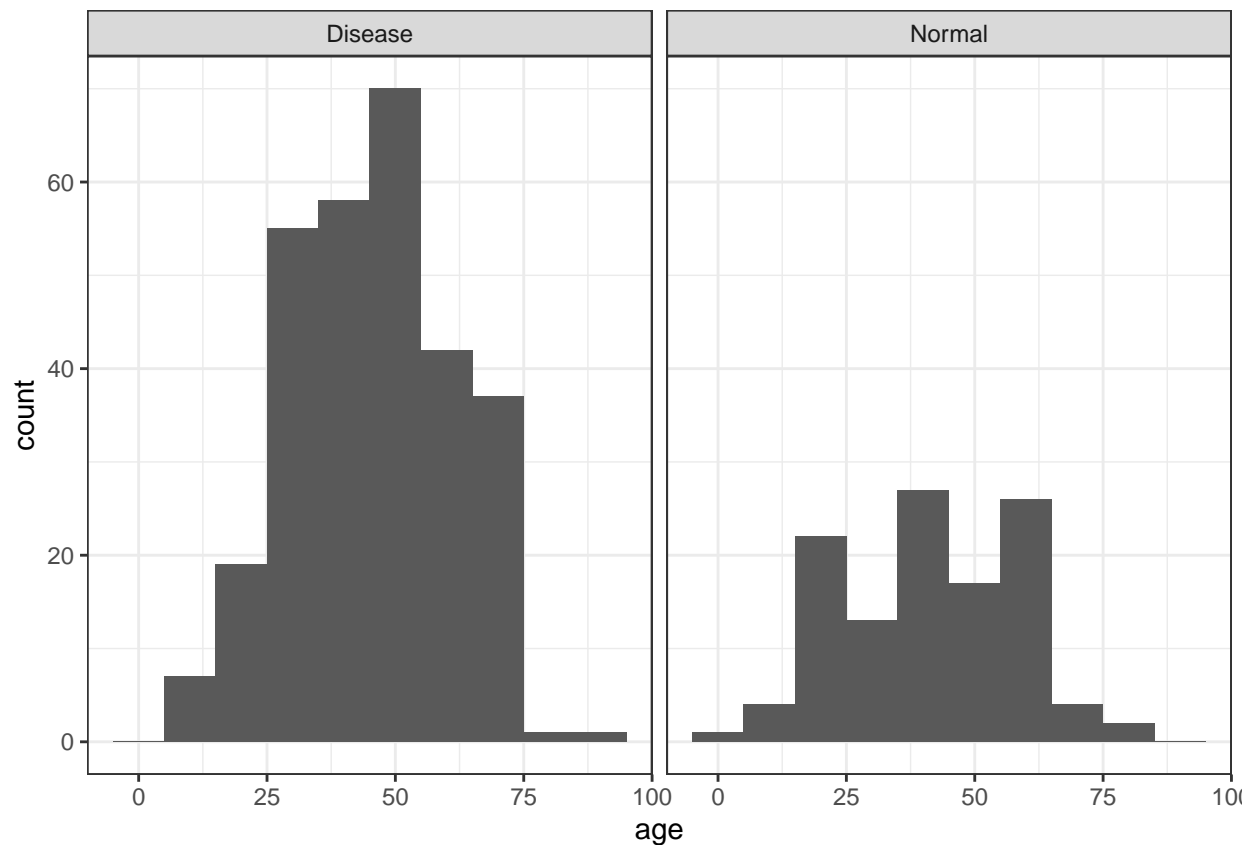
2.1.2 Splitting dataset into train and test sets

After importing the data into Rstudio, the data is splitted into training and test sets before proceeding further into data exploration. In addition, the approximate proportion of the outcome column is preserved.

```
set.seed(1)
train_index <- createDataPartition(data$outcome, p=.7, list=FALSE)
train <- data[train_index,]
test <- data[-train_index,]
```

2.1.3 Exploration of data

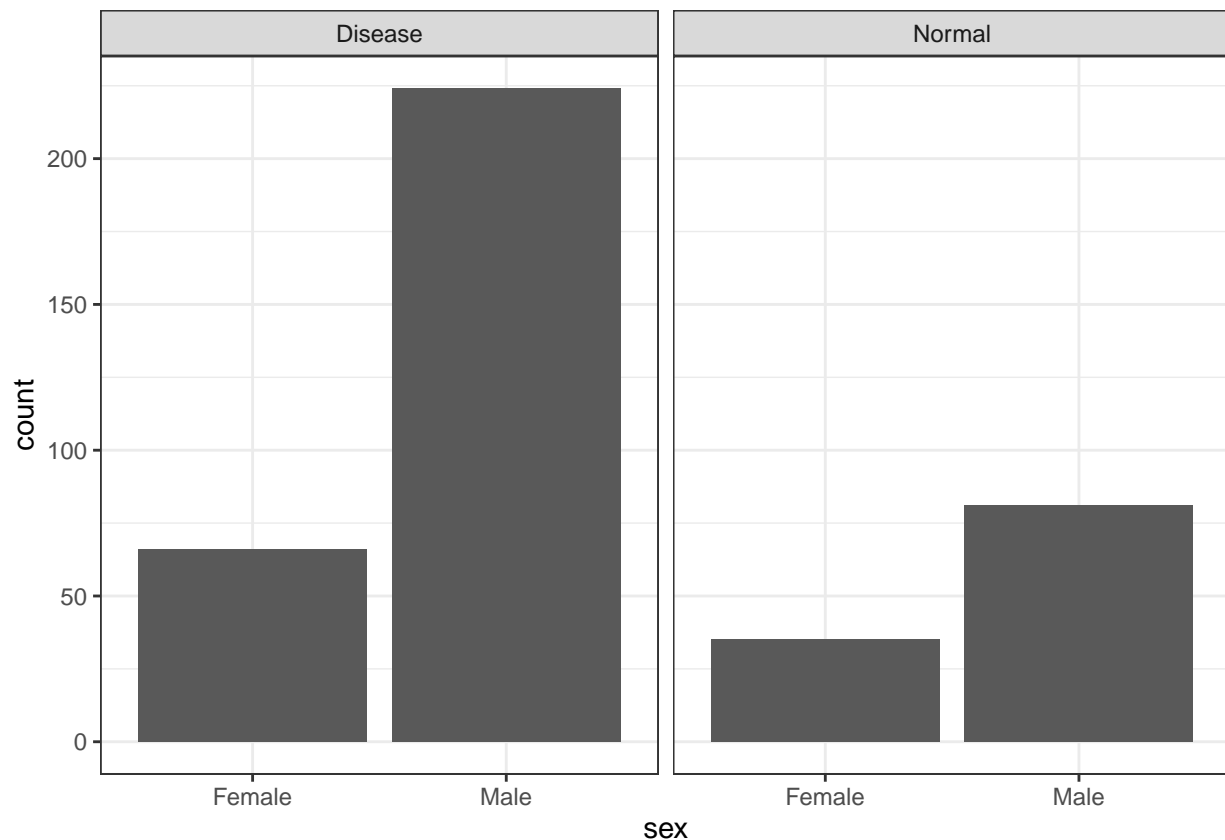
```
train %>%
  ggplot(aes(x = age)) +
    geom_histogram(binwidth = 10) +
    theme_bw() +
    facet_grid(~ outcome)
```



2.1.3.1 Age

From the graph plotted above, both groups are observed to have a similar distribution in terms of **age**.

```
train %>%
  ggplot(aes(x = sex)) +
  geom_bar() +
  theme_bw() +
  facet_grid(~ outcome)
```



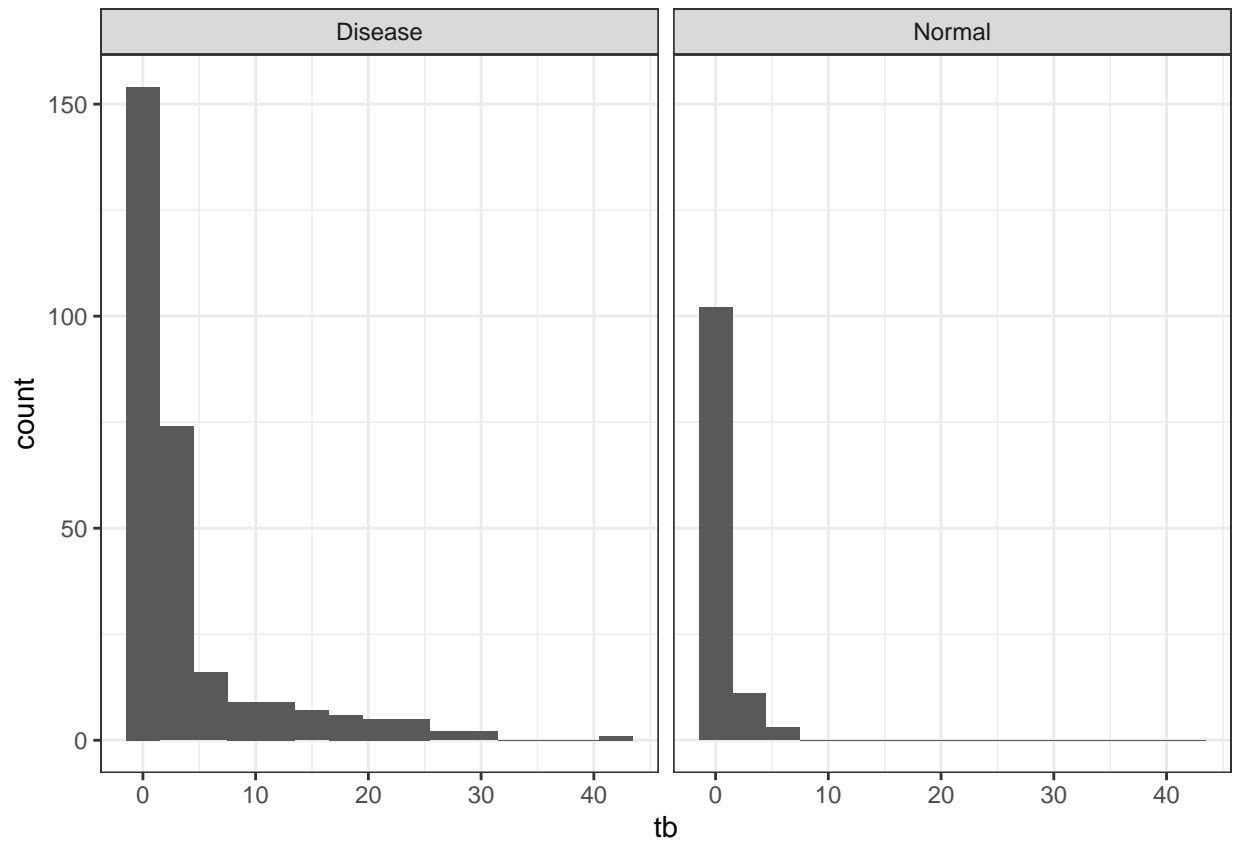
2.1.3.2 Sex

From the **sex** graph, female is observed to have a lower counts in both groups as compared to male, with a larger difference in the **Disease** group. This large difference could be due to the general lifestyle of male as compared to female, which causes the higher incidences in male. On the other hand, it seems that distribution of **Normal** group is constructed in the same way as **Disease** group.

2.1.3.3 Bilirubin Bilirubin is produced as a byproduct from the breakdown of heme, which occurs naturally in the blood. In the data, two types of bilirubin is reported:

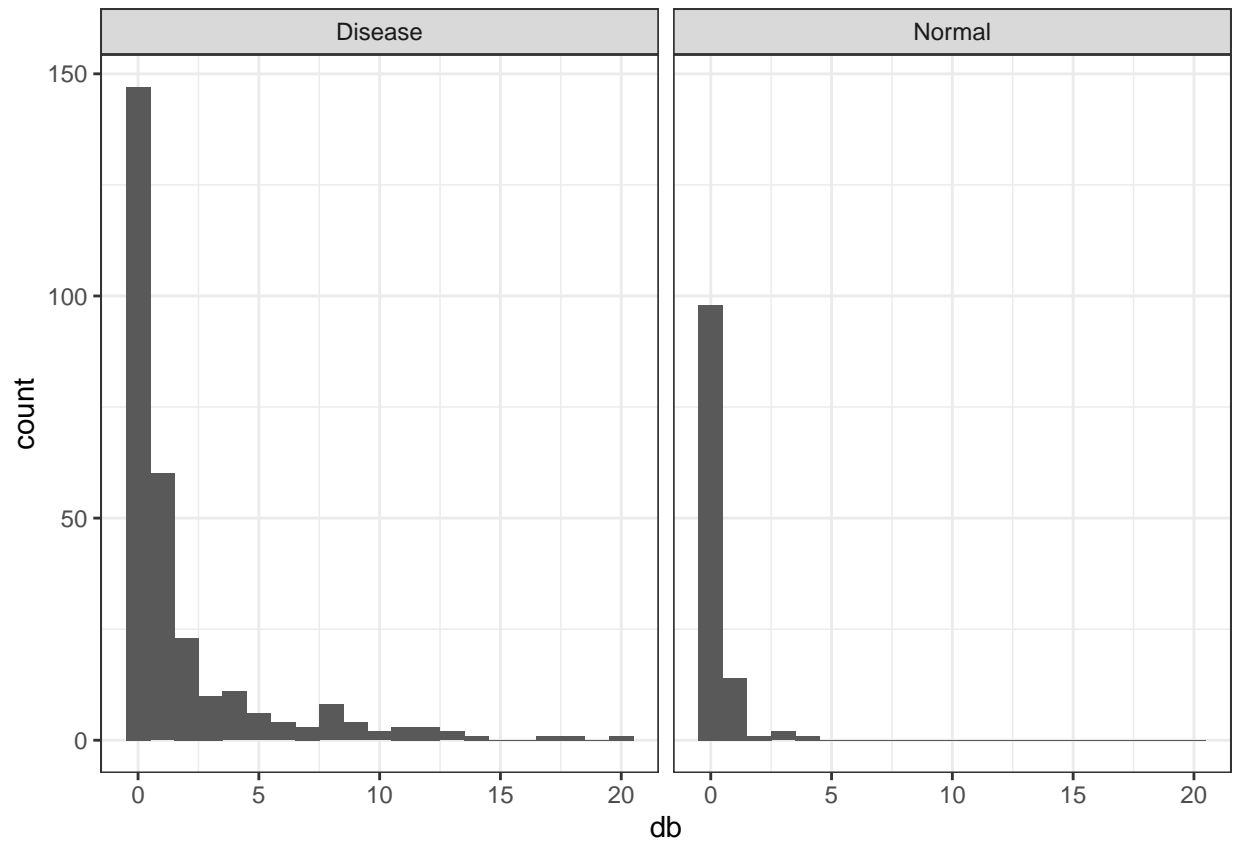
- **Total bilirubin (tb)** represents both conjugated and unconjugated bilirubin present in the sample.
- **Direct bilirubin (db)** represents the water-soluble bilirubin that is present in the blood sample which reacts with the reagents.

```
train %>%
  ggplot(aes(x = tb)) +
  geom_histogram(binwidth = 3) +
  theme_bw() +
  facet_grid(~ outcome)
```



The `tb` graph above shows the distribution of total bilirubin levels in both groups. It is important to note the wider distribution observed in **Disease** as compared to the **Normal**. However, this is not a good indicator alone as majority of the counts are generally in the 0 to 10 `tb` level range.

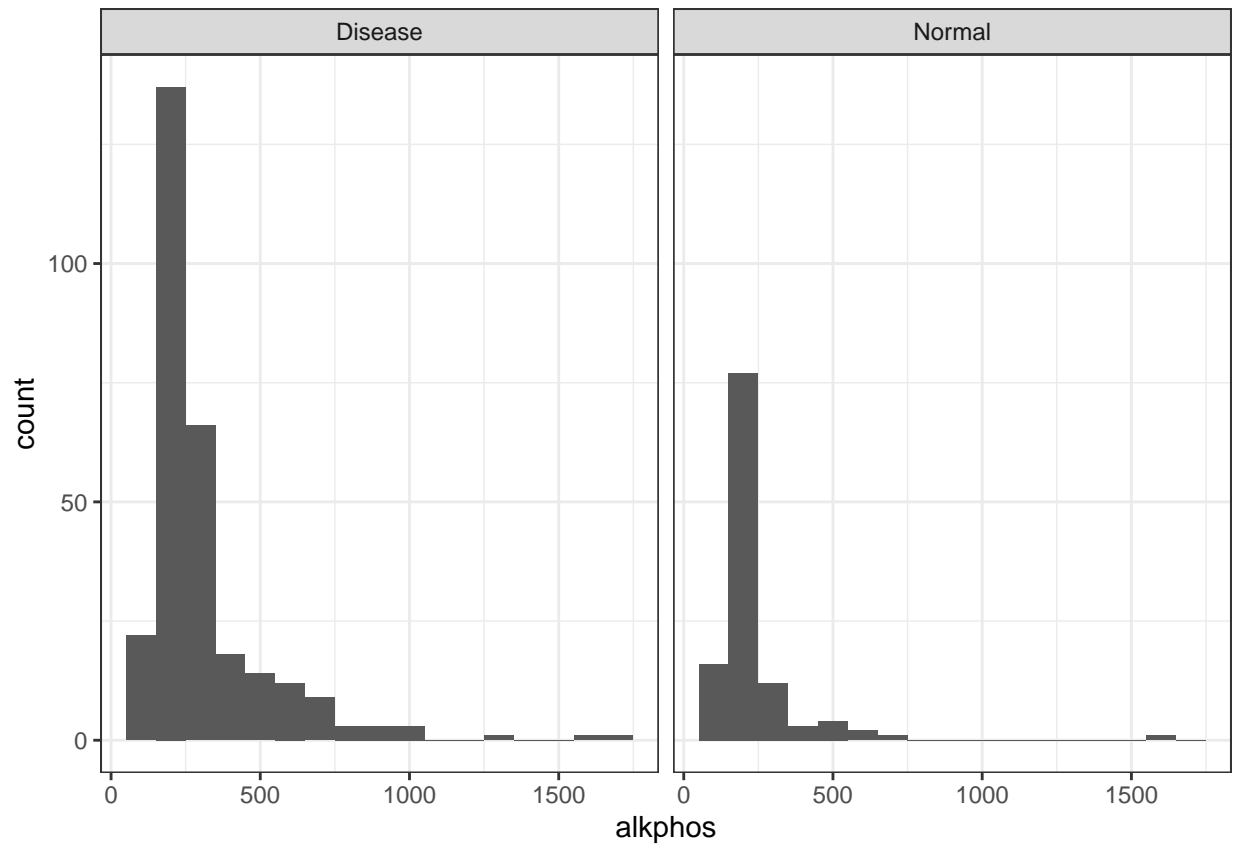
```
train %>%  
  ggplot(aes(x = db)) +  
    geom_histogram(binwidth = 1) +  
    theme_bw() +  
    facet_grid(~ outcome)
```



Similarly, it is observed that `db` has the same distribution pattern as `tb`, thus indicating that it is not a good indicator alone as well.

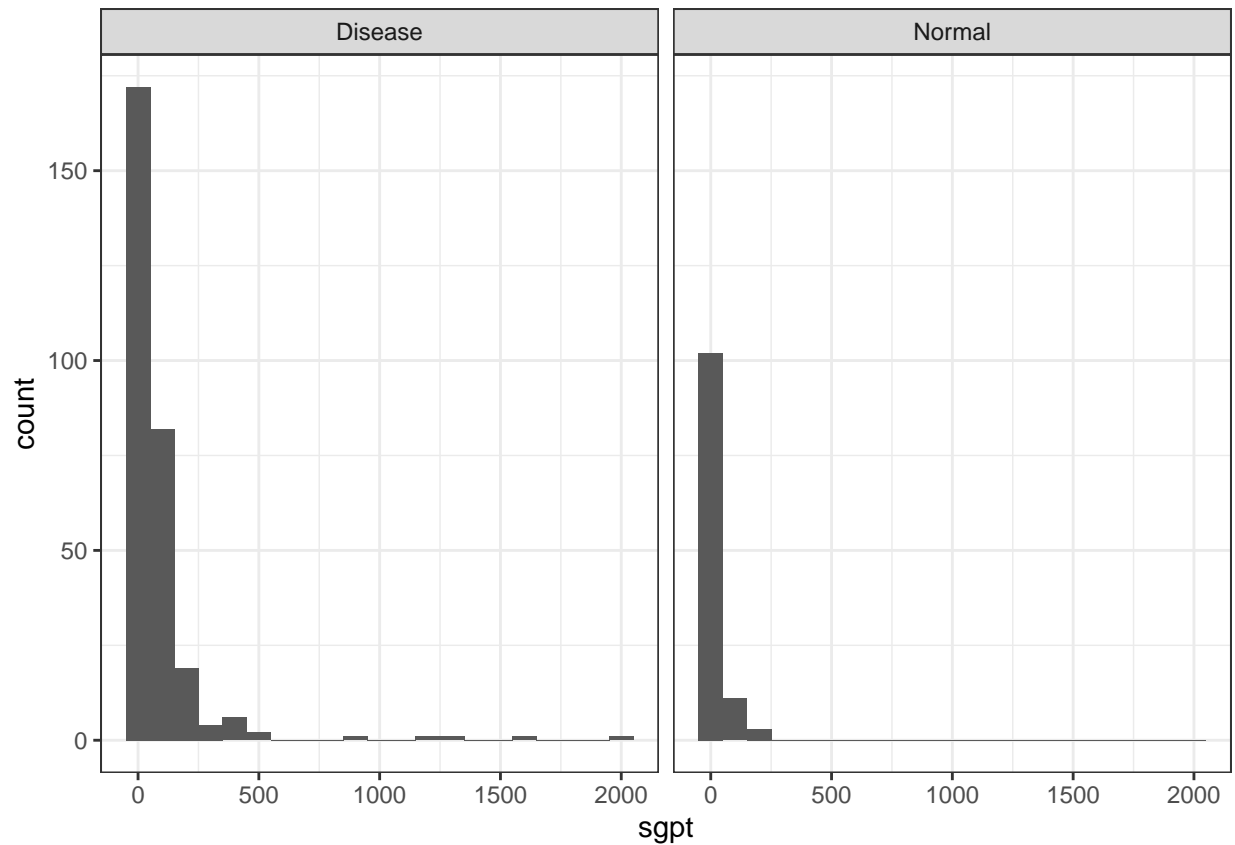
2.1.3.4 Alkaline Phosphatase Alkaline Phosphatase (`alkphos`) is an enzyme that removes phosphate groups from organic compounds.

```
train %>%
  ggplot(aes(x = alkphos)) +
  geom_histogram(binwidth = 100) +
  theme_bw() +
  facet_grid(~ outcome)
```



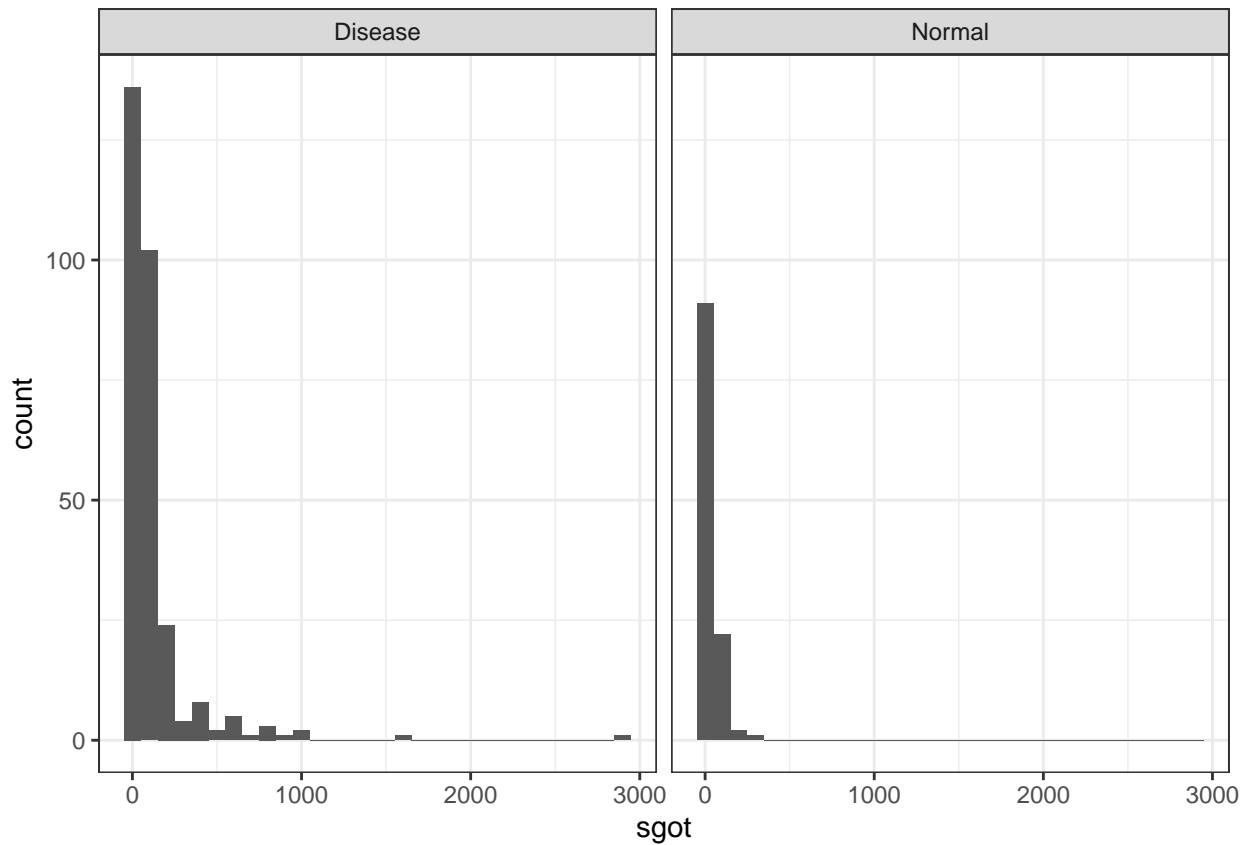
2.1.3.5 Alanine Aminotransferase Alanine Aminotransferase (sgpt) is the enzyme responsible for the transfer of amino groups from L-alanine to alpha-ketoglutarate.

```
train %>%  
  ggplot(aes(x = sgpt)) +  
    geom_histogram(binwidth = 100) +  
    theme_bw() +  
    facet_grid(~ outcome)
```



2.1.3.6 Aspartate Aminotransferase Aspartate Aminotransferase (sgot) is the enzyme that is involved in the transfer of amino groups from aspartic acid to alpha-ketoglutaric acid.

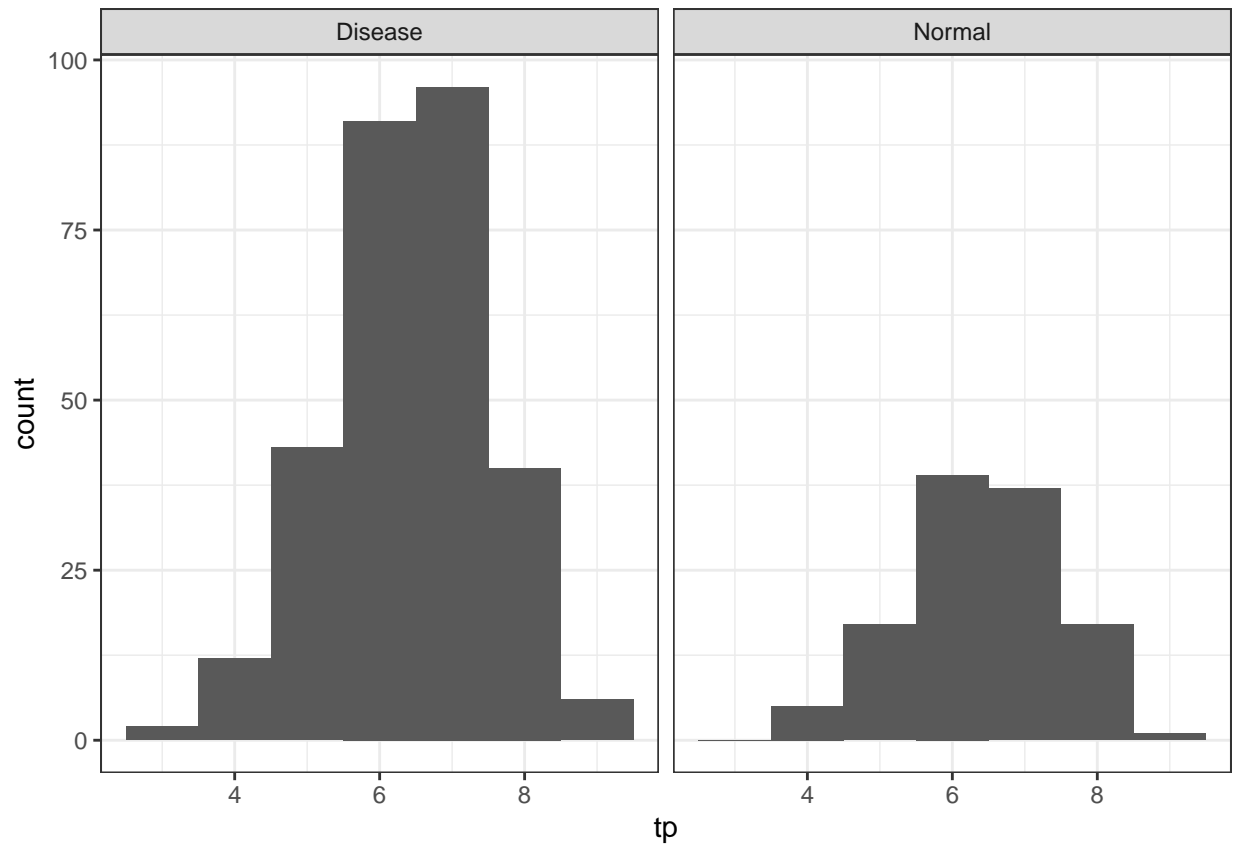
```
train %>%  
  ggplot(aes(x = sgot)) +  
    geom_histogram(binwidth = 100) +  
    theme_bw() +  
    facet_grid(~ outcome)
```



It is observed that the levels of all the enzymes displayed a similar distribution pattern as bilirubin, where wider range is observed in **Disease** as compared to the control. Similarly, majority of the counts still reside in the narrow range of values.

2.1.3.7 Total Protein The Total Protein (tp) level is measured in both groups as well in search for a correlation to the liver disease.

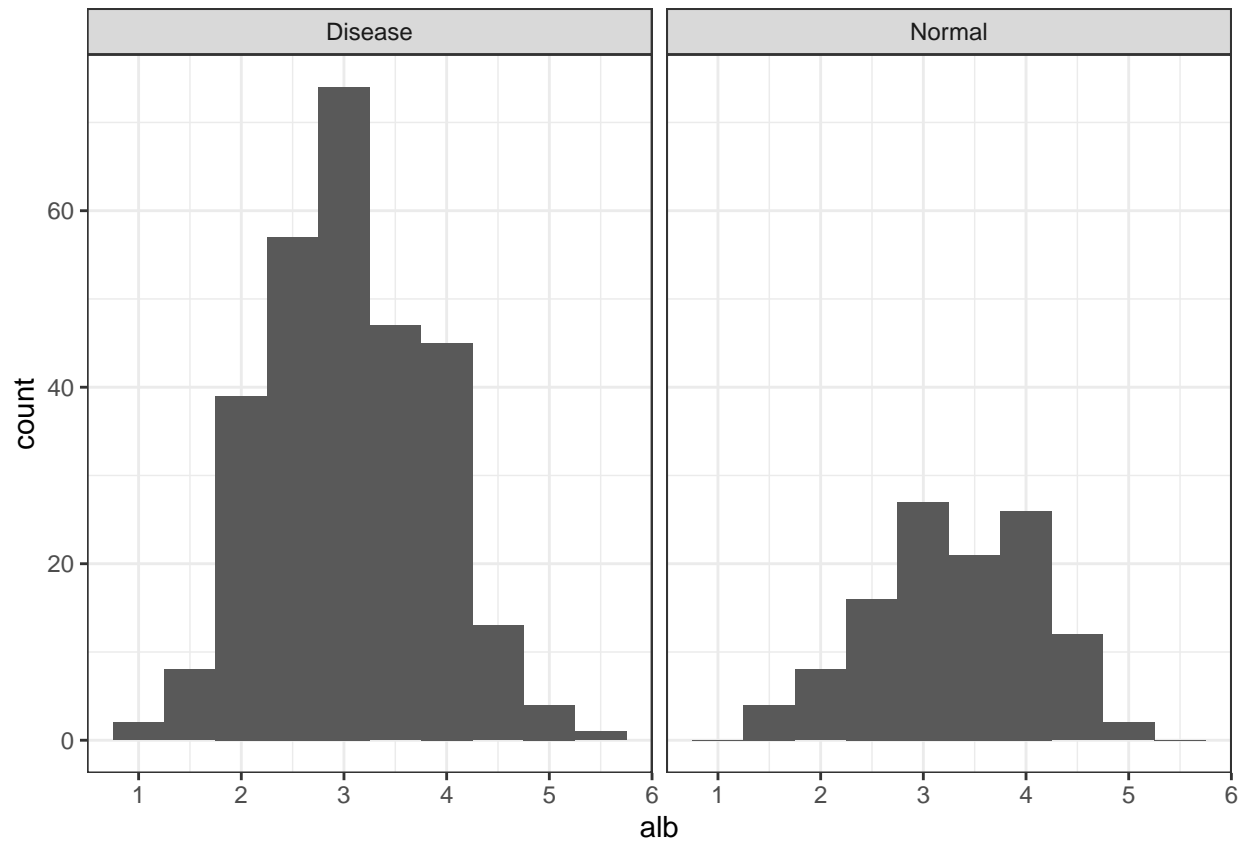
```
train %>%
  ggplot(aes(x = tp)) +
    geom_histogram(binwidth = 1) +
    theme_bw() +
    facet_grid(~ outcome)
```



The total protein level is observed to be similar between the two groups.

2.1.3.8 Albumin Besides total protein, Albumin (alb) level is also measured for both groups.

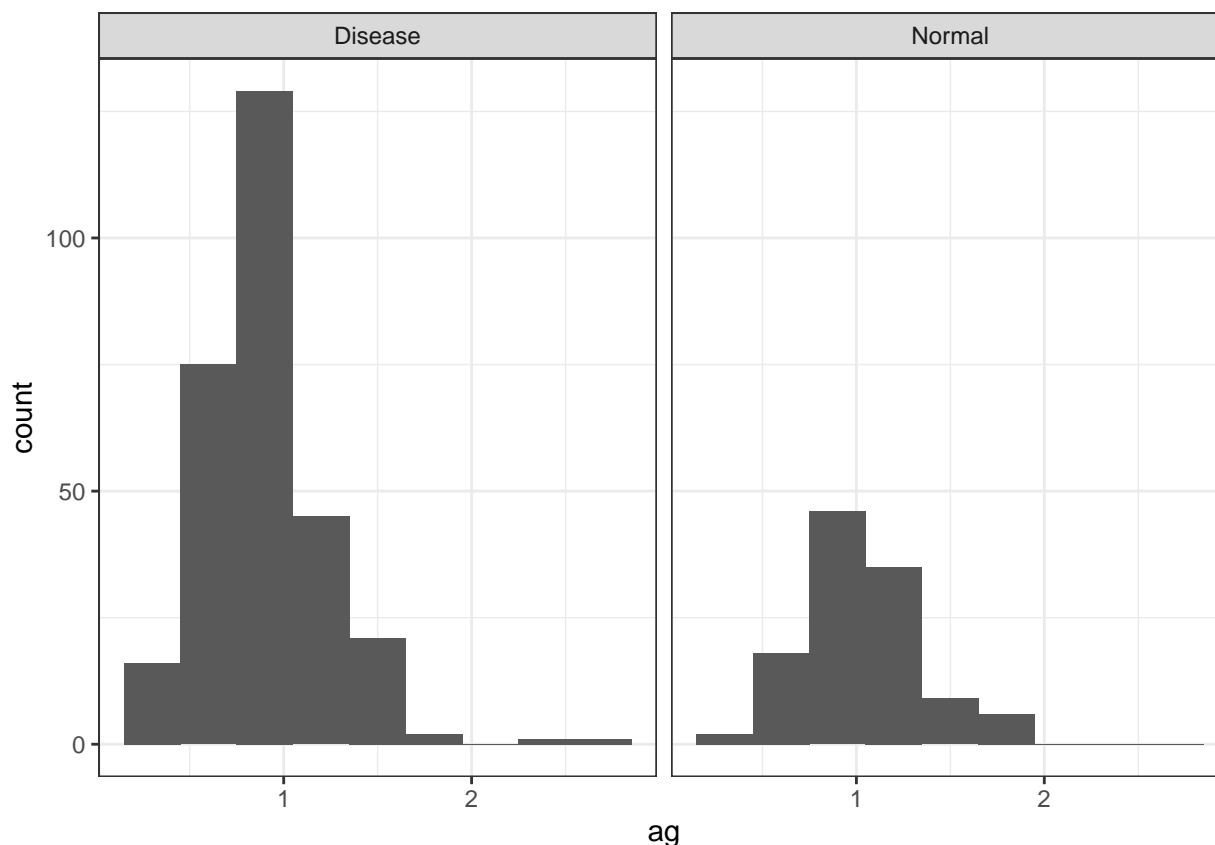
```
train %>%
  ggplot(aes(x = alb)) +
  geom_histogram(binwidth = 0.5) +
  theme_bw() +
  facet_grid(~ outcome)
```



Once again, albumin levels appear mostly similar for both groups.

2.1.3.9 Albumin/ Globulin Ratio The final data to explore into will be the ratio between albumin and globulin.

```
train %>%  
  ggplot(aes(x = ag)) +  
    geom_histogram(binwidth = 0.3) +  
    theme_bw() +  
    facet_grid(~ outcome)
```



In the graph above, similar distribution is observed for both groups, with a wider range of distribution observed for Disease.

2.1.4 Correlation of Predictors

Next, it is evident that some predictors may be correlated which suggest a possible relationship between the predictors and may aid in the development of the model. Therefore it is important to look into the correlation of predictors as shown below.

```
cor(subset(train, select = -c(sex, outcome)))
```

```
##          age          tb          db      alkphos      sgpt      sgot
## age      1.00000000  0.02745113  0.02713288  0.10393973 -0.08820424 -0.06746583
## tb       0.02745113  1.00000000  0.98583307  0.22755545  0.22158387  0.30717453
## db       0.02713288  0.98583307  1.00000000  0.23349181  0.20857299  0.28429813
## alkphos  0.10393973  0.22755545  0.23349181  1.00000000  0.05864885  0.05186084
## sgpt     -0.08820424  0.22158387  0.20857299  0.05864885  1.00000000  0.88218405
## sgot     -0.06746583  0.30717453  0.28429813  0.05186084  0.88218405  1.00000000
## tp       -0.15624022  0.03718640  0.04208275 -0.04221924 -0.07276461 -0.05943760
## alb      -0.23205958 -0.25244127 -0.24069175 -0.18292997 -0.03301984 -0.09210596
## ag       -0.21016352 -0.23405606 -0.21836664 -0.23368595  0.02158976 -0.04228052
##          tp          alb          ag
## age      -0.15624022 -0.23205958 -0.21016352
## tb       0.03718640 -0.25244127 -0.23405606
## db       0.04208275 -0.24069175 -0.21836664
```

```
## alkphos -0.04221924 -0.18292997 -0.23368595
## sgpt    -0.07276461 -0.03301984  0.02158976
## sgot    -0.05943760 -0.09210596 -0.04228052
## tp      1.00000000  0.76914682  0.23047803
## alb      0.76914682  1.00000000  0.68867867
## ag      0.23047803  0.68867867  1.00000000
```

Highly correlated predictors were identified during this process, in preparation for the development of prediction models. Moreover, threshold of >0.75 is applied for the list of predictors to avoid ‘closely related’ variables, thus removing three variables as shown in the following code.

```
train <- train %>% subset(select = -c(db, sgpt, alb))
test  <- test  %>% subset(select = -c(db, sgpt, alb))
```

2.2 Machine Learning Models

Correlations were found for some predictors, enabling machine learning models to be developed. Following code is set up to collate the results for better representation in this report. Subsequently, a few models were constructed to determine the most appropriate prediction model for this project.

```
results <- data.frame(Model = character(),
                      Accuracy = double(),
                      stringsAsFactors = FALSE)
```

2.2.1 K-Nearest Neighbours

One of the most common classification model is the K-Nearest Neighbours model. The k-nearest matching points from will be evaluated the training data, where the predicted outcome is derived from the average evaluated result. The patients with similar medical profiles will result in them producing the same predicted outcome.

```
knn = train(outcome ~ ., data = train, method = "knn", preProcess=c('knnImpute'))
knn
```

```
## k-Nearest Neighbors
##
## 406 samples
## 7 predictor
## 2 classes: 'Disease', 'Normal'
##
## Pre-processing: nearest neighbor imputation (7), centered (7), scaled (7)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 406, 406, 406, 406, 406, ...
## Resampling results across tuning parameters:
##
## k Accuracy Kappa
## 5 0.6662388 0.1635275
## 7 0.6589695 0.1367210
## 9 0.6750102 0.1558779
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

The value of K is first evaluated to determine the optimal input value to produce the most appropriate prediction model.

```
pred = predict(knn, newdata = test)
confusionMatrix <- confusionMatrix(pred, test$outcome, prevalence = 0.06)
results[nrow(results) + 1, ] <- c(as.character('K-nearest neighbours (knn)'),
                                confusionMatrix$overall['Accuracy'])
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction Disease Normal
##      Disease      105      39
##      Normal       19      10
##
##              Accuracy : 0.6647
##              95% CI : (0.5891, 0.7346)
##      No Information Rate : 0.7168
##      P-Value [Acc > NIR] : 0.9436
##
##              Kappa : 0.058
##
##      Mcnemar's Test P-Value : 0.0126
##
##              Sensitivity : 0.84677
##              Specificity : 0.20408
##              Pos Pred Value : 0.06359
##              Neg Pred Value : 0.95427
##              Prevalence : 0.06000
##              Detection Rate : 0.60694
##      Detection Prevalence : 0.83237
##              Balanced Accuracy : 0.52543
##
##              'Positive' Class : Disease
##
```

```
results %>% knitr::kable()
```

Model	Accuracy
K-nearest neighbours (knn)	0.664739884393064

As shown from the code above, the accuracy of this model is too low to be used as a prediction model for the aim of this project.

2.2.2 Linear Classifier

The next model to investigate is linear classifier. `glmboost` is applied in this case to achieve a better fit.

```
lc = train(outcome ~ ., data = train, method = "glmboost")
pred = predict(lc, newdata = test)
confusionMatrix <- confusionMatrix(pred, test$outcome)
results[nrow(results) + 1, ] <- c(as.character('Linear Classifier (glmboost)'),
                                confusionMatrix$overall['Accuracy'])
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Disease Normal
##   Disease      124      49
##   Normal         0       0
##
##           Accuracy : 0.7168
##           95% CI : (0.6434, 0.7825)
##   No Information Rate : 0.7168
##   P-Value [Acc > NIR] : 0.5384
##
##           Kappa : 0
##
## Mcnemar's Test P-Value : 7.025e-12
##
##           Sensitivity : 1.0000
##           Specificity : 0.0000
##           Pos Pred Value : 0.7168
##           Neg Pred Value :      NaN
##           Prevalence : 0.7168
##           Detection Rate : 0.7168
##   Detection Prevalence : 1.0000
##           Balanced Accuracy : 0.5000
##
##           'Positive' Class : Disease
##
```

```
results %>% knitr::kable()
```

Model	Accuracy
K-nearest neighbours (knn)	0.664739884393064
Linear Classifier (glmboost)	0.716763005780347

The accuracy of prediction is improved with this model. However, almost the whole **Normal** group is considered positive in this model, thus making this model unsuitable as a prediction model.

2.2.3 Logistic Regression

Another common model to investigate will be logistic regression. **bayesglm** is applied in this model for better fit.

```

lr = train(outcome ~ ., data = train, method = "bayesglm")
pred = predict(lr, newdata = test)
confusionMatrix <- confusionMatrix(pred, test$outcome)
results[nrow(results) + 1, ] <- c(as.character('Logistic Regression (bayesglm)'),
                                confusionMatrix$overall['Accuracy'])
confusionMatrix

```

```

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Disease Normal
##   Disease      118      41
##   Normal         6       8
##
##              Accuracy : 0.7283
##              95% CI : (0.6556, 0.7931)
##   No Information Rate : 0.7168
##   P-Value [Acc > NIR] : 0.4046
##
##              Kappa : 0.1465
##
##  Mcnemar's Test P-Value : 7.071e-07
##
##              Sensitivity : 0.9516
##              Specificity : 0.1633
##              Pos Pred Value : 0.7421
##              Neg Pred Value : 0.5714
##              Prevalence : 0.7168
##              Detection Rate : 0.6821
##   Detection Prevalence : 0.9191
##   Balanced Accuracy : 0.5574
##
##   'Positive' Class : Disease
##

```

```

results %>% knitr::kable()

```

Model	Accuracy
K-nearest neighbours (knn)	0.664739884393064
Linear Classifier (glmboost)	0.716763005780347
Logistic Regression (bayesglm)	0.728323699421965

Further improvement is seen through this model, however, the accuracy of the model is still not robust enough to be used as an appropriate prediction model for this project.

3 Results

The results from all the models are compiled in the following code.

```
results %>% arrange(Accuracy) %>% knitr::kable()
```

Model	Accuracy
K-nearest neighbours (knn)	0.664739884393064
Linear Classifier (glmboost)	0.716763005780347
Logistic Regression (bayesglm)	0.728323699421965

A few models were performed in this project and it is important to note that even though models like Linear Classifier have higher accuracy, but the high sensitivity and low specificity of the model caused it to be an inappropriate model. On the other hand, models like K-Nearest Neighbours model have a better sensitivity and specificity as compared to Linear Classifier. However, the accuracy of the model is a concern, thus unsuitable to be used as a prediction model.

4 Conclusion

In conclusion, the models that are investigated in this project are not appropriate to be utilized as a prediction model for the aim of this project. However, should there be a larger dataset, K-Nearest Neighbours model's accuracy may be further improved, which may establish it to be an appropriate prediction model. With that in mind, more data has to be acquired before establishing a prediction model for the aim of this project.