Role of Liver Function Test in Diagnosis of Liver Diseases

Zenasimlab

1/8/2020

```
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------- tidyverse 1.2.
1 --
```

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

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

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.2
```

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

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

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

Introduction

Chronic liver disease(cirrhosis) is responsible for more than 1 million deaths annually and the majority of these deaths are preventable. They usually present with non-specific symptoms such as weakness, fatigue, intermittent nausea and abdominal discomfort. Moreover it can be further complicated by the underlying chronic diseases hence increases the challenges of diagnosis clinically.

No serologic test can diagnose cirrhosis accurately.(Reference 1) As I'm a practicing clinician in Malaysia, I'm interested to explore whether machine learning technique can be applied to the blood parameters dataset to identify the patients with liver diseases. Potentially the outcome of the project can complement the current practice of clinican to achieve more accurate diagnosis of liver diseases.

In this project,the dataset including blood results of liver function test in patients receiving care from hepatologist and baseline results from a control group was used for the analysis. It is freely availble through UCI Machine Learning Repository.(Reference 2)

Data ingestion

Download and splitting the data

First we import the dataset into a data frame and then define the column names.

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

fullData <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/0022
5/Indian%20Liver%20Patient%20Dataset%20(ILPD).csv",
                 sep=',',
                 header=FALSE,
                 col.names=columnNames)
```

```
fullData <- subset(fullData, complete.cases(fullData))

fullData <- fullData%>%
  mutate(outcome = as.character(outcome))%>%
  mutate(outcome = replace(outcome, outcome == '1', 'Care'))%>%
  mutate(outcome = replace(outcome, outcome == '2', 'Control'))%>%
  mutate(outcome = as.factor(outcome))

rm(columnNames)
head(fullData)
```

```
##   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    Care
## 2  62   Male 10.9 5.5     699   64  100 7.5 3.2 0.74    Care
## 3  62   Male  7.3 4.1     490   60   68 7.0 3.3 0.89    Care
## 4  58   Male  1.0 0.4     182   14   20 6.8 3.4 1.00    Care
## 5  72   Male  3.9 2.0     195   27   59 7.3 2.4 0.40    Care
## 6  46   Male  1.8 0.7     208   19   14 7.6 4.4 1.30    Care
```

Normal range of liver function test(Reference 2):

Total bilirubin:2 to 21μmol/L Direct bilirubin:< 8μmol/L Alkaline Phosphatase: 41 to 133U/L Alanine Aminotransferase: 7–56 U/ L Aspartate Aminotransferase: 0 to 35U/L Total Protein: 6 to 8 g/dl Albumin: 3.5 to 5.0 g/dl

Data exploration

Creating training and test sets

Next generate the training and test sets by splitting the data, preserving the approximate proportions in the outcome column.

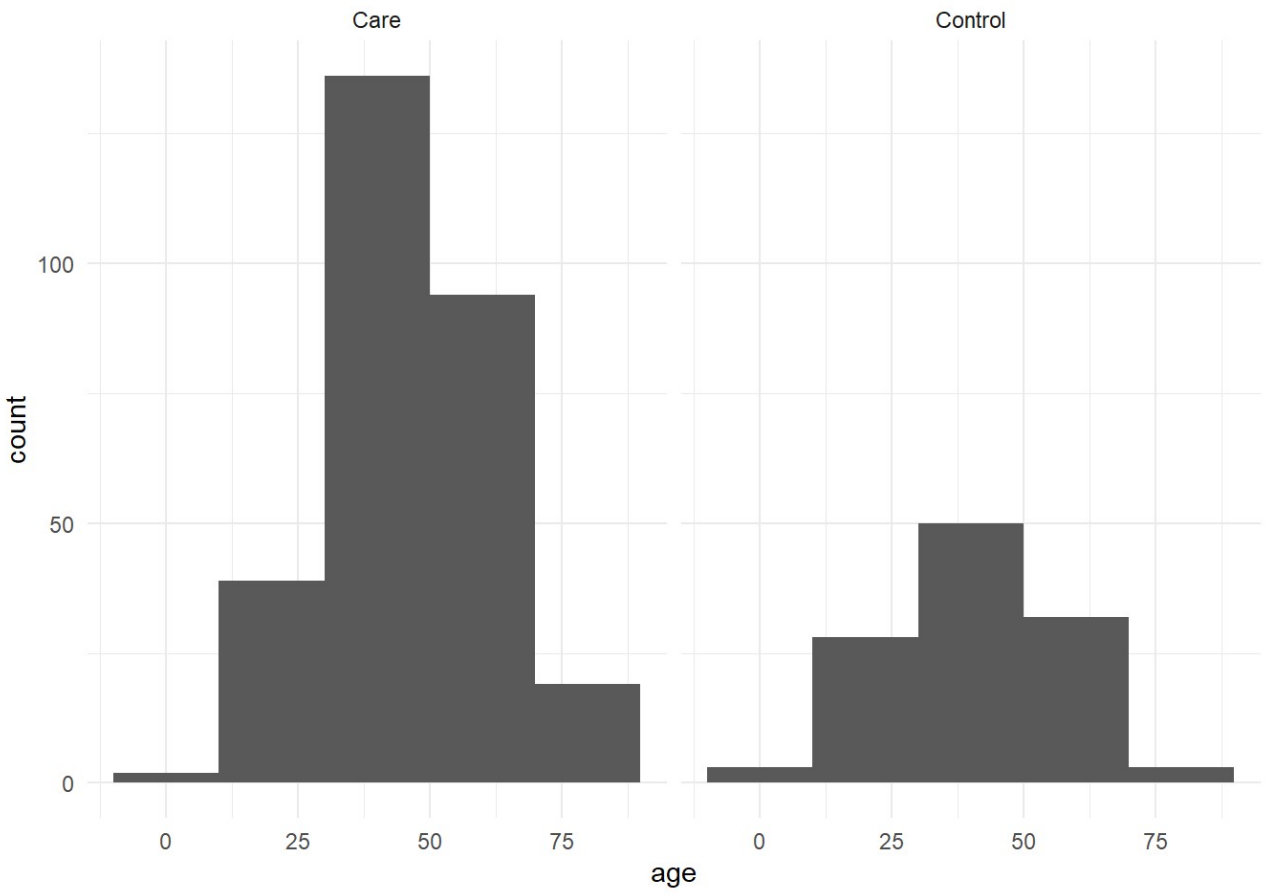
```
set.seed(1, sample.kind = "Rounding")
```

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

```
trainIndex <- createDataPartition(fullData$outcome, times=1, p=0.7, list=FALSE)
train <- fullData[trainIndex,]
test <- fullData[-trainIndex,]
rm(trainIndex)
```

Age

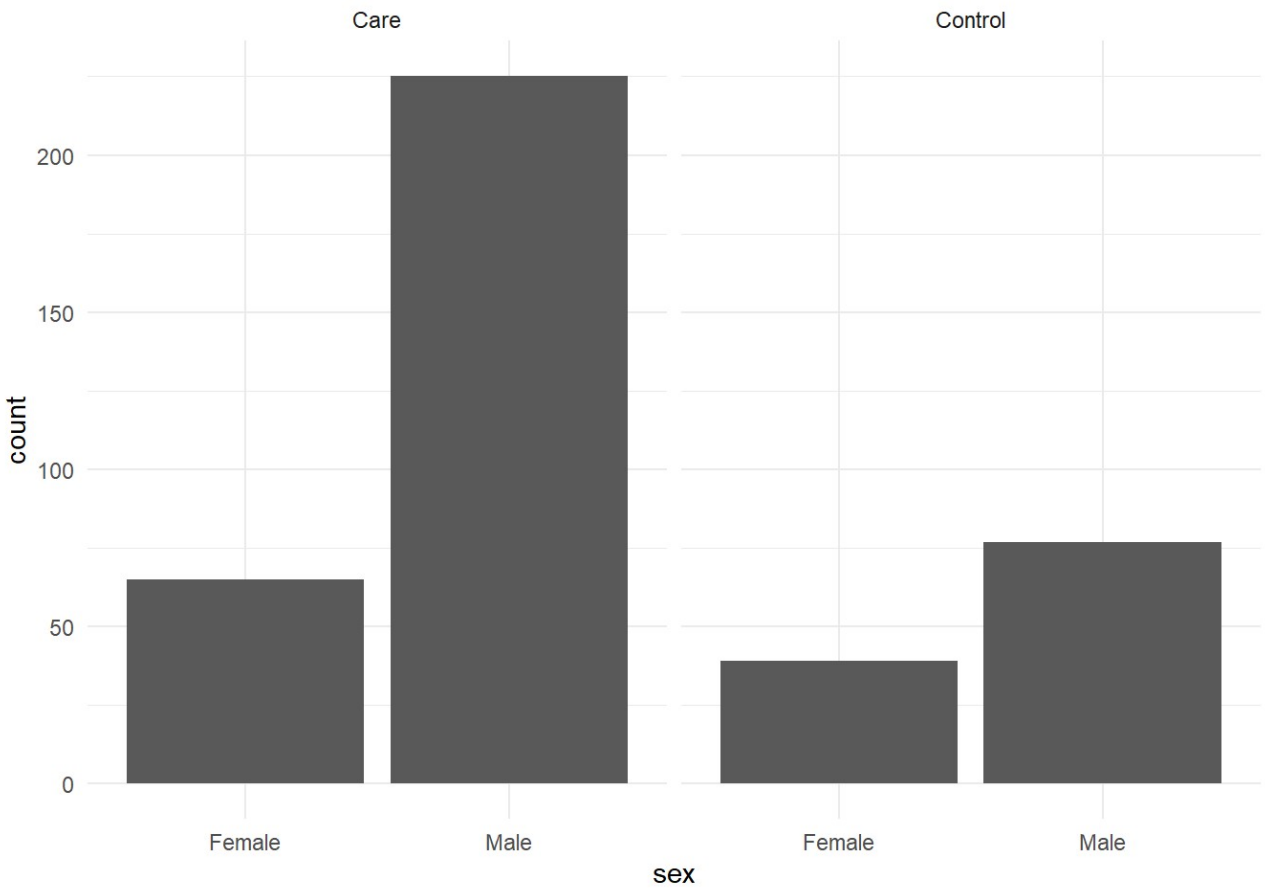
```
train%>%
  ggplot(aes(x = age)) +
    geom_histogram(binwidth = 20) +
    theme_minimal() +
    facet_grid(~ outcome)
```



The age distribution in both care and control group appear to be in similar pattern, both peaks at around 30-50 years old.

Sex

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



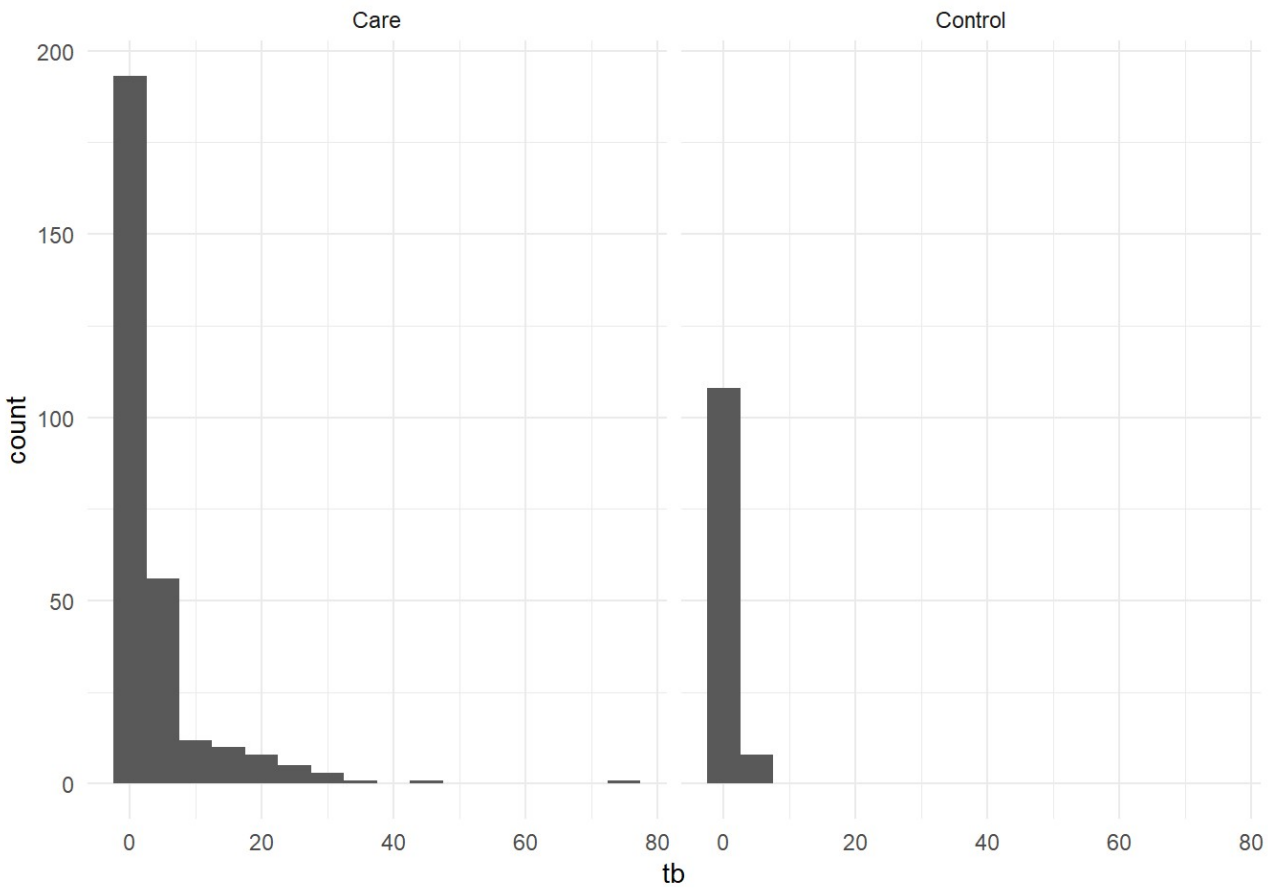
This bar graph depicts the gender distribution in both care and control groups whereby there are predominantly male.

Bilirubin

Bilirubin is formed by the breakdown of red blood cells in the body. It can be divided into: a) Total bilirubin - encompass both conjugated and unconjugated bilirubin b) Direct bilirubin - this is also known as conjugated bilirubin. It is water-soluble bilirubin which travels from the liver into the small intestine. A very small amount passes into the kidneys and is excreted in urine.

Total bilirubin

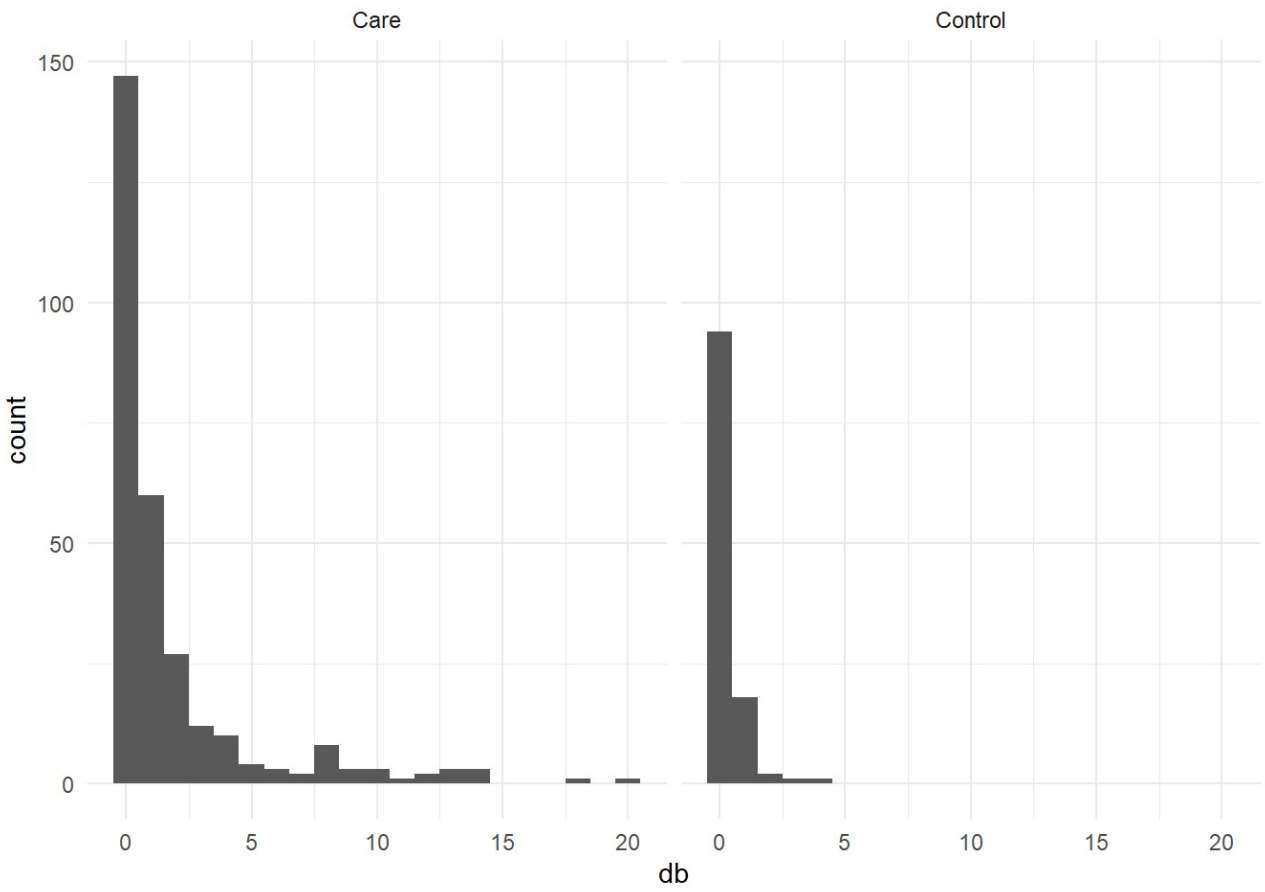
```
train %>%
  ggplot(aes(x = tb)) +
    geom_histogram(binwidth = 5) +
    theme_minimal() +
    facet_grid(~ outcome)
```



Most values in histogram lie between “0-”5" column with the data in care group spread across more columns.

Direct bilirubin

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



Similar pattern was observed in the histograms for direct bilirubin.

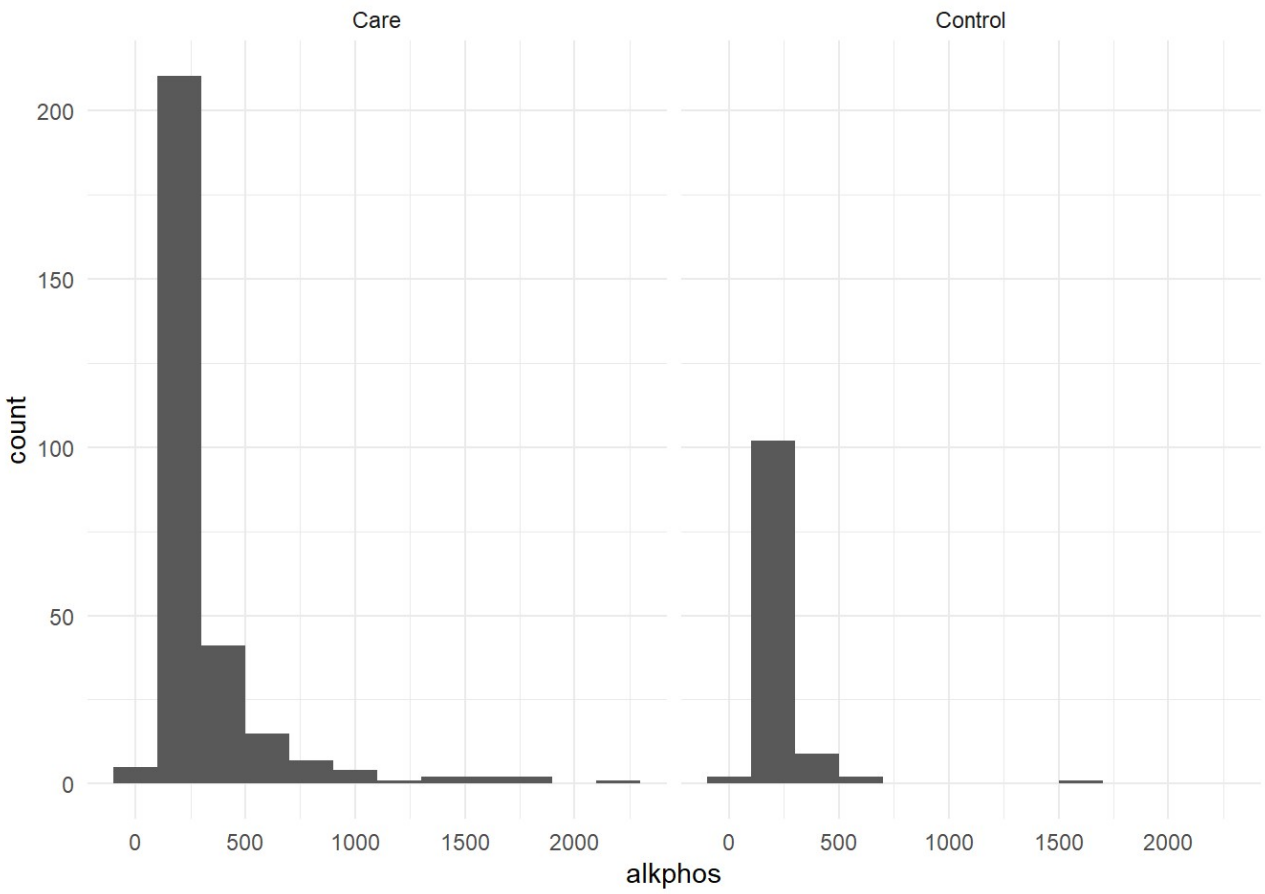
Enzymes

There are 3 specific liver enzymes present in our blood sample, namely:

a. Alkaline Phosphatase- mostly found in the liver, bones, kidneys, and digestive system
b. Alanine Aminotransferase - primarily in the liver and kidney
c. Aspartate Aminotransferase - found in the highest concentrations in liver, muscles, heart, kidney, brain and red blood cells

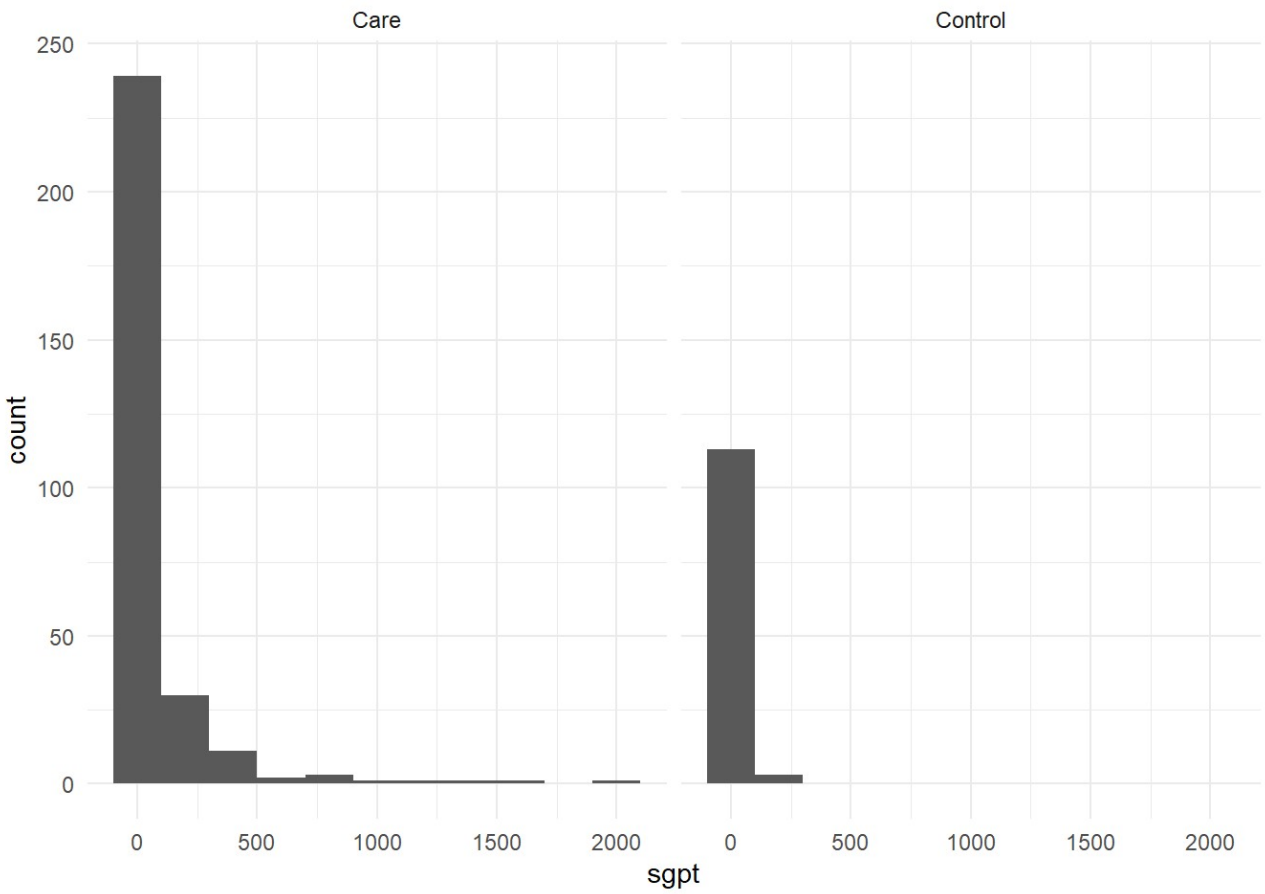
Alkaline Phosphatase

```
train %>%
  ggplot(aes(x = alkphos)) +
    geom_histogram(binwidth = 200) +
    theme_minimal() +
    facet_grid(~ outcome)
```



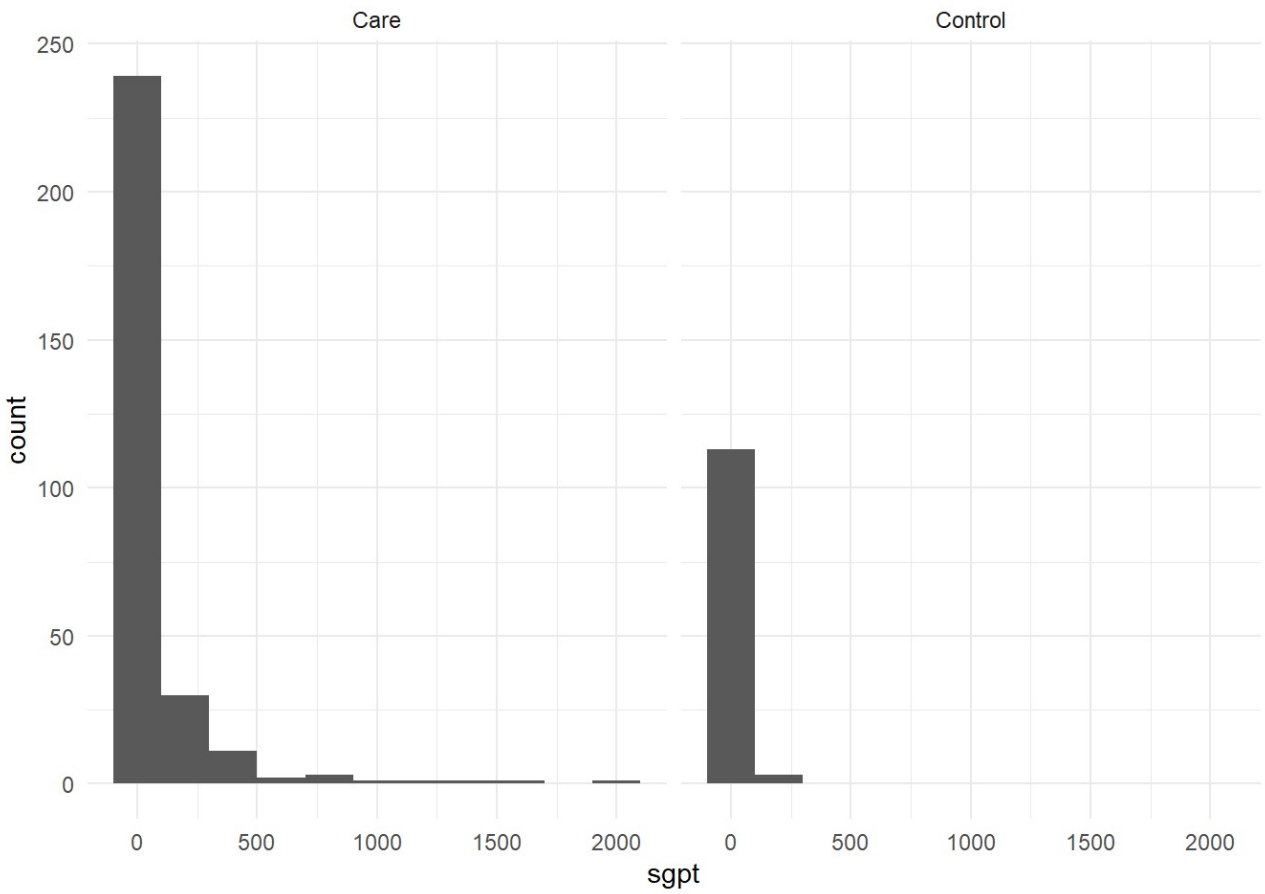
Alanine Aminotransferase

```
train %>%
  ggplot(aes(x = sgpt)) +
    geom_histogram(binwidth = 200) +
    theme_minimal() +
    facet_grid(~ outcome)
```



Aspartate Aminotransferase

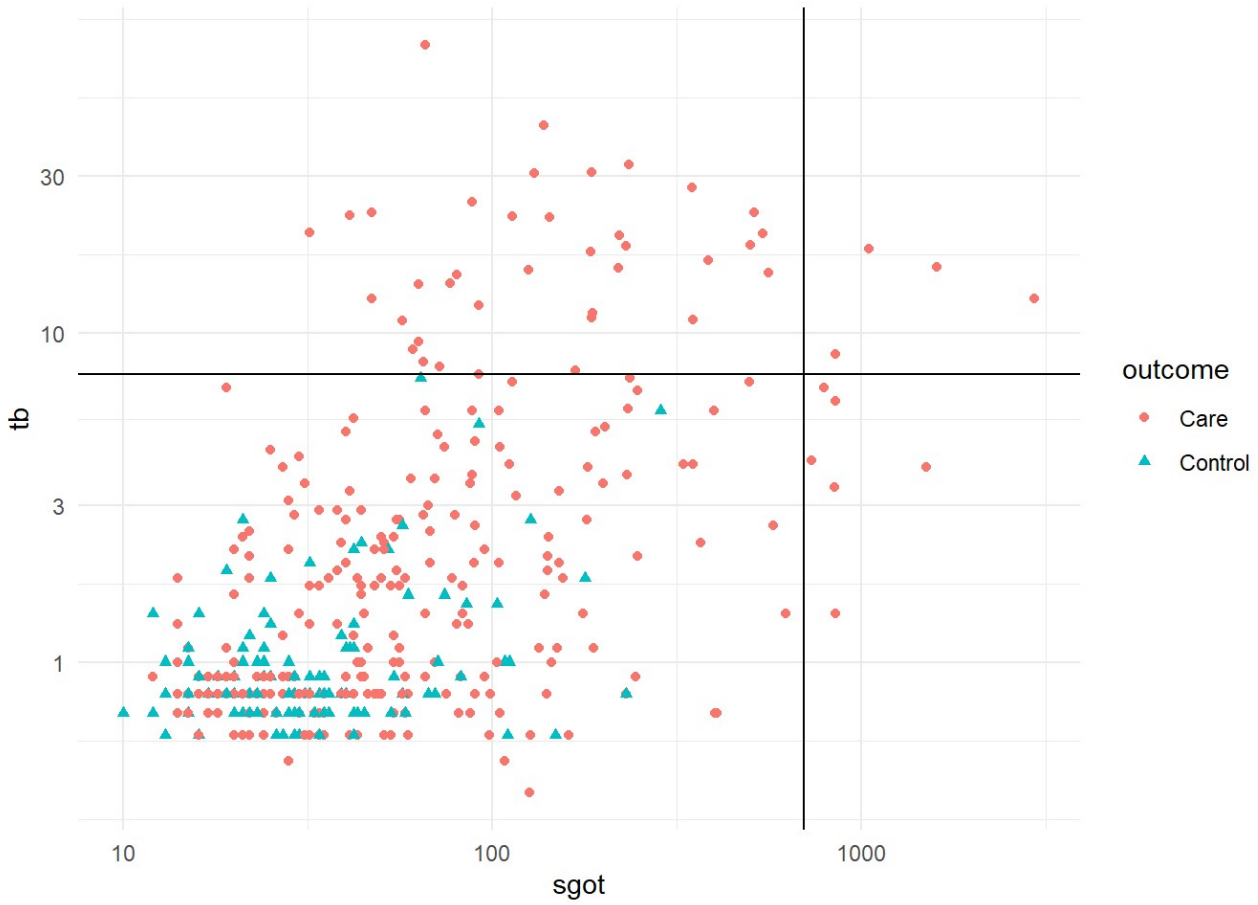
```
train %>%
  ggplot(aes(x = sgpt)) +
    geom_histogram(binwidth = 200) +
    theme_minimal() +
    facet_grid(~ outcome)
```



Similar patterns were observed in all 3 enzymes in both care and control group.In order to compare the outliers of one parameter to another, the total bilirubin level is plotted against one of the enzymes.

Total bilirubin level Against Aspartate Aminotransferase

```
train %>%
  ggplot(aes(x = sgot, y = tb, shape = outcome, color = outcome)) +
    geom_point() +
    scale_y_log10() +
    scale_x_log10() +
    geom_vline(xintercept = 700) +
    geom_hline(yintercept = 7.5) +
    theme_minimal()
```



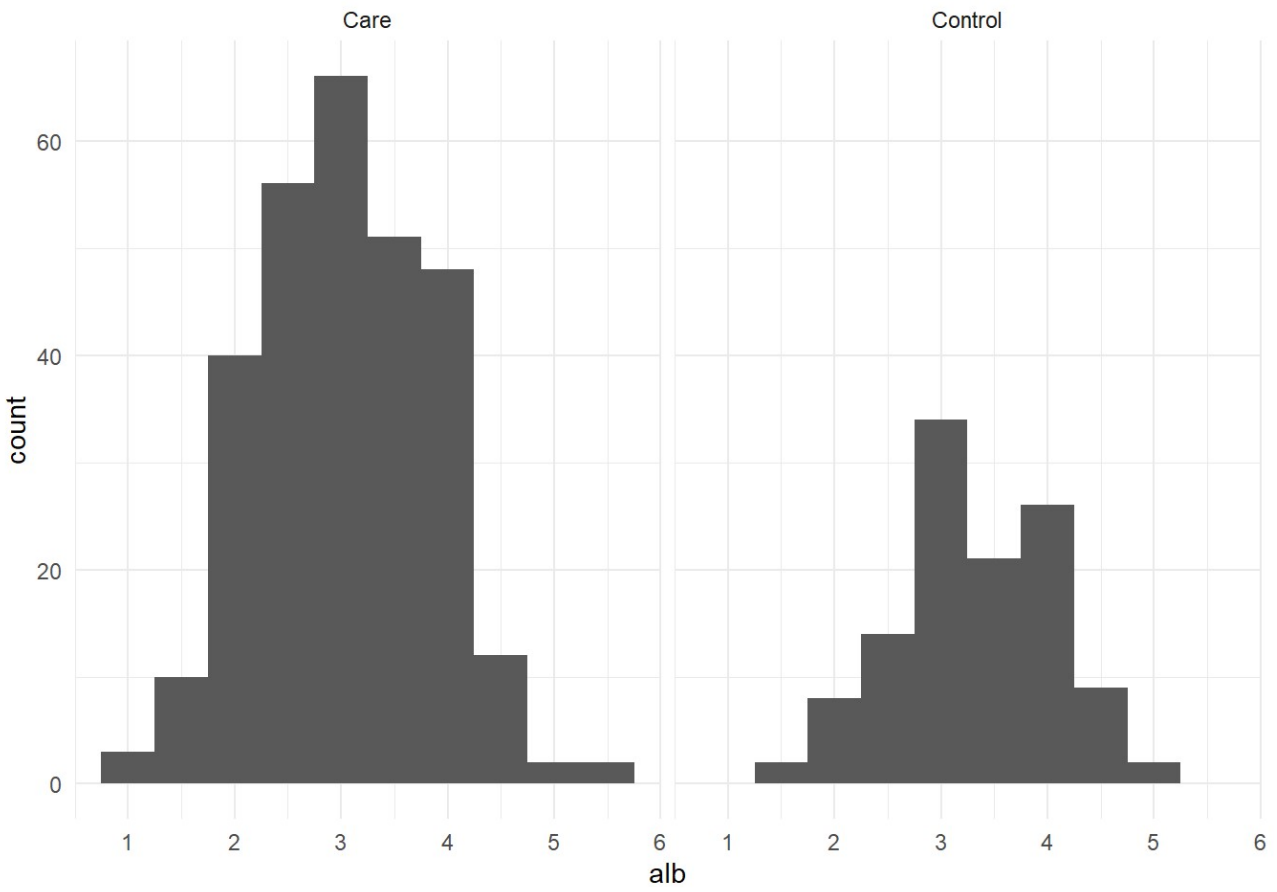
Practically all healthy subjects in control group fall under the left lower quadrant(low total bilirubin and low aspartate aminotransferase).Most of the patients group receiving hepatologist care fall under left upper quadrant (high total bilirubin and low asparate aminotransferase) with the rest in right upper and lower quadrant.

Proteins

Under protein category, it can be further divided into: a) Albumin - a group of water-soluble globular proteins b) Total protein

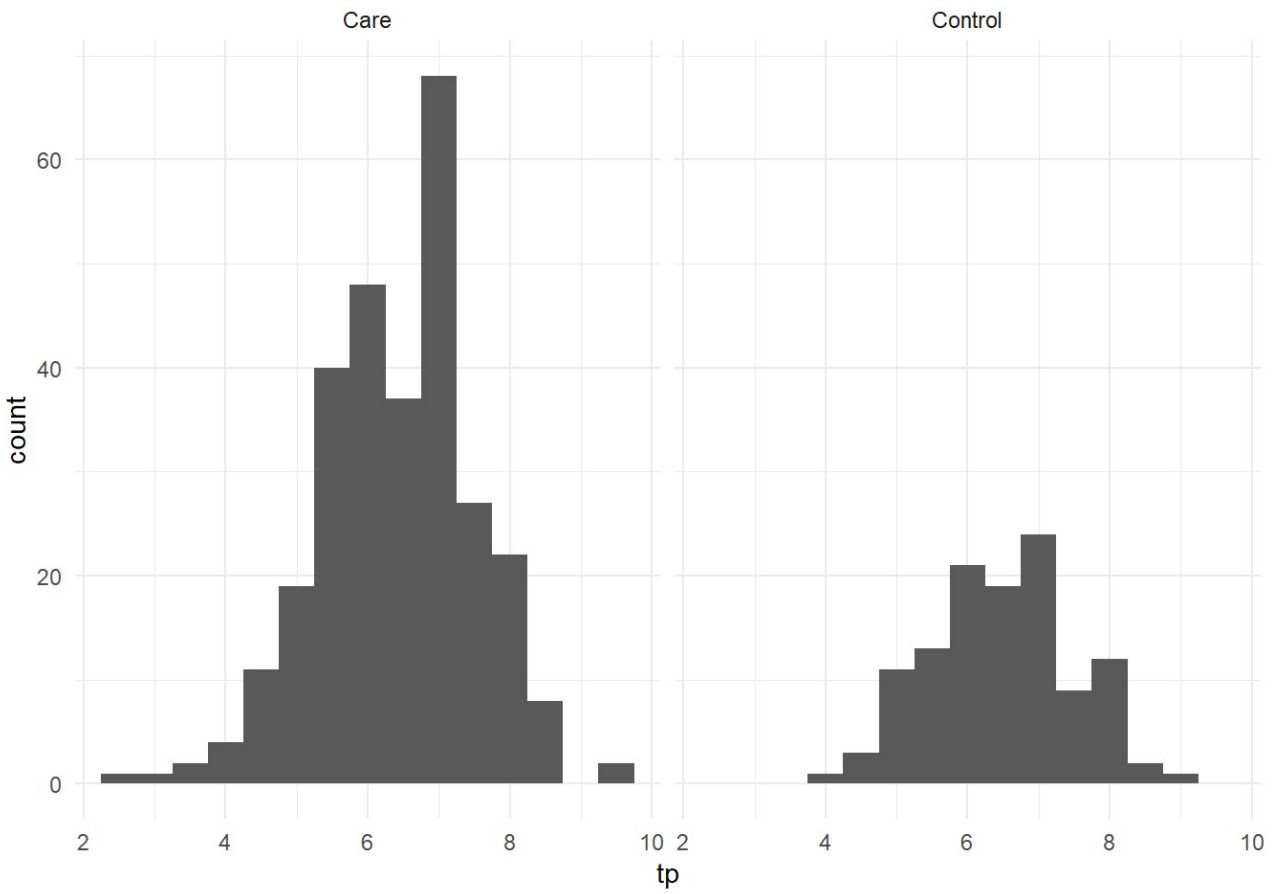
Albumin

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



Total protein

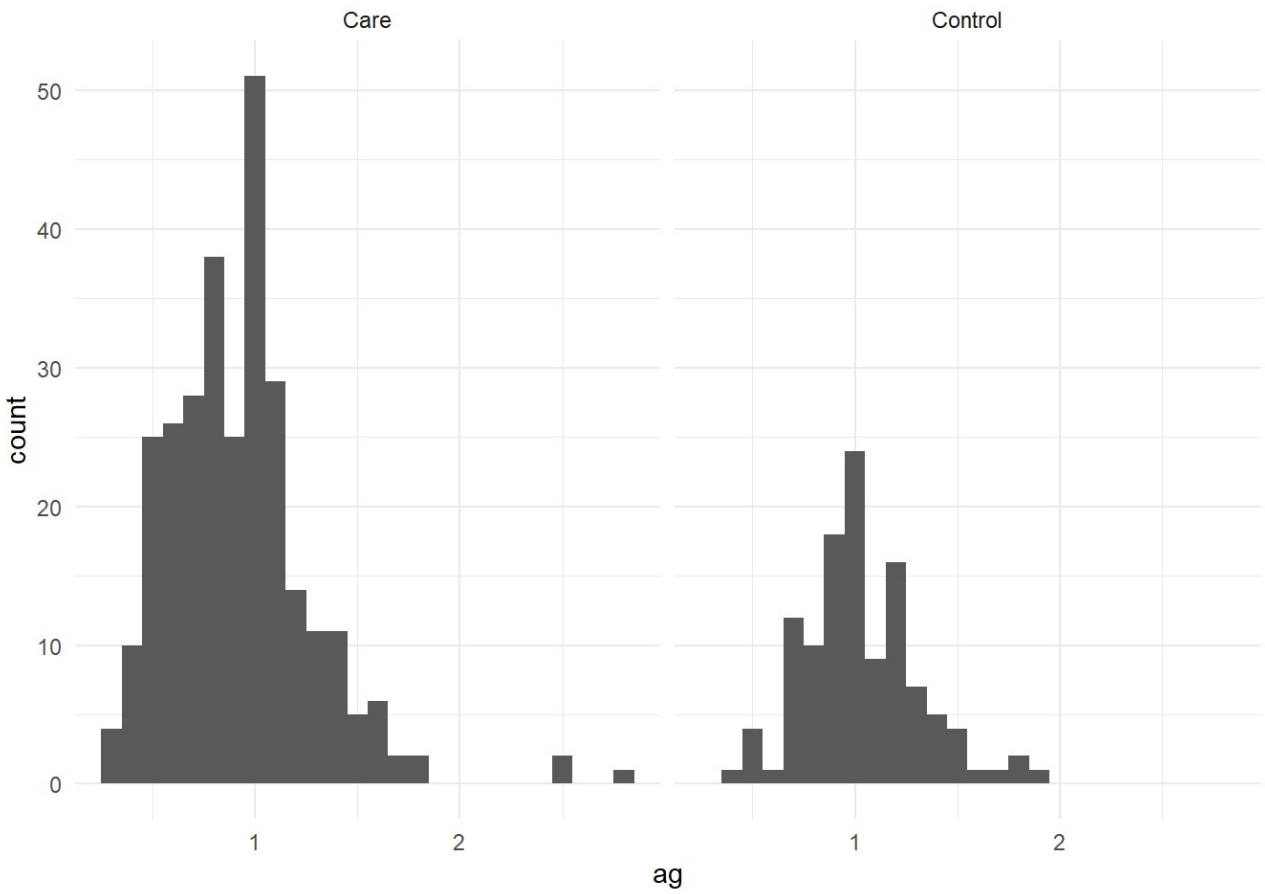
```
train %>%
  ggplot(aes(x = tp)) +
    geom_histogram(binwidth = 0.5) +
    theme_minimal() +
    facet_grid(~ outcome)
```



The distribution of albumin and total protein is almost identical in both care and control groups.

Ratio between albumin and globulin.

```
subset(train, !is.na(ag)) %>%
  ggplot(aes(x = ag)) +
    geom_histogram(binwidth = 0.1) +
    theme_minimal() +
    facet_grid(~ outcome)
```



It is obvious that there are some outliers in the patients group who receive hepatologist care.

Correlated Predictors

From the previous graphs, we learnt that some of the dataset might be correlated with each other. It can be further confirmed by correlation function and graphical comparison of both parameters. For both predictors which are correlated, we can remove one of the categorical columns from the dataset.

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

```
##                 age           tb           db     alkphos        sgpt
## age      1.00000000 -0.010064846 -0.028276740  0.07025503 -0.12696088
## tb      -0.01006485  1.000000000  0.851408700  0.17729799  0.20094208
## db      -0.02827674  0.851408700  1.000000000  0.21226099  0.23494336
## alkphos  0.07025503  0.177297994  0.212260992  1.00000000  0.06226685
## sgpt    -0.12696088  0.200942076  0.234943358  0.06226685  1.00000000
## sgot    -0.08389212  0.258589373  0.298457164  0.06753361  0.90614301
## tp      -0.20178775 -0.007717427  0.002892306 -0.03101599 -0.03820811
## alb     -0.25850218 -0.192718816 -0.202003023 -0.15316481  0.00256560
## ag      -0.19364814 -0.151605936 -0.134533092 -0.20432112  0.02027530
##                sgot           tp         alb          ag
## age     -0.08389212 -0.201787747 -0.25850218 -0.19364814
## tb       0.25858937 -0.007717427 -0.19271882 -0.15160594
## db       0.29845716  0.002892306 -0.20200302 -0.13453309
## alkphos  0.06753361 -0.031015987 -0.15316481 -0.20432112
## sgpt     0.90614301 -0.038208108  0.00256560  0.02027530
## sgot     1.00000000 -0.052080337 -0.05063058 -0.01518947
## tp      -0.05208034  1.000000000  0.79248223  0.24914789
## alb     -0.05063058  0.792482228  1.00000000  0.67299646
## ag      -0.01518947  0.249147887  0.67299646  1.00000000
```

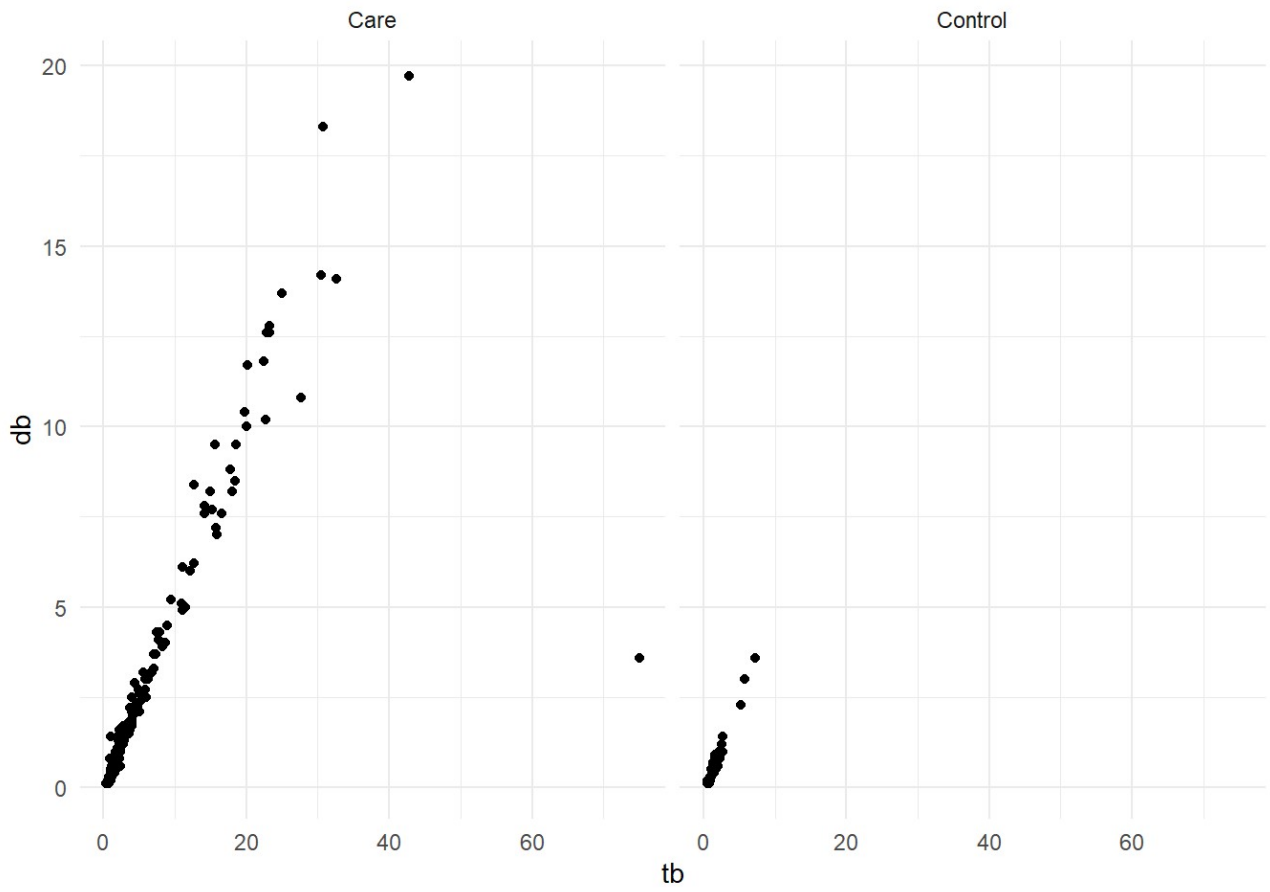
0.7 was used as a cutoff value to identify highly correlated predictors:

a. Direct bilirubin (`db`) and total bilirubin (`tb`)
b. Aspartate Aminotransferase (`sgot`) and Alamine Aminotransferase (`sgpt`)
c. Albumin (`alb`) and Total protein (`tp`)

Bilirubin

Graphing the distribution of the points in both groups will make it clear whether the correlation affects both patient groups equally.

```
train %>%
  ggplot(aes(x = tb, y = db)) +
    geom_point() +
    theme_minimal() +
    facet_grid(~ outcome)
```



Both groups have the same distribution of the points.

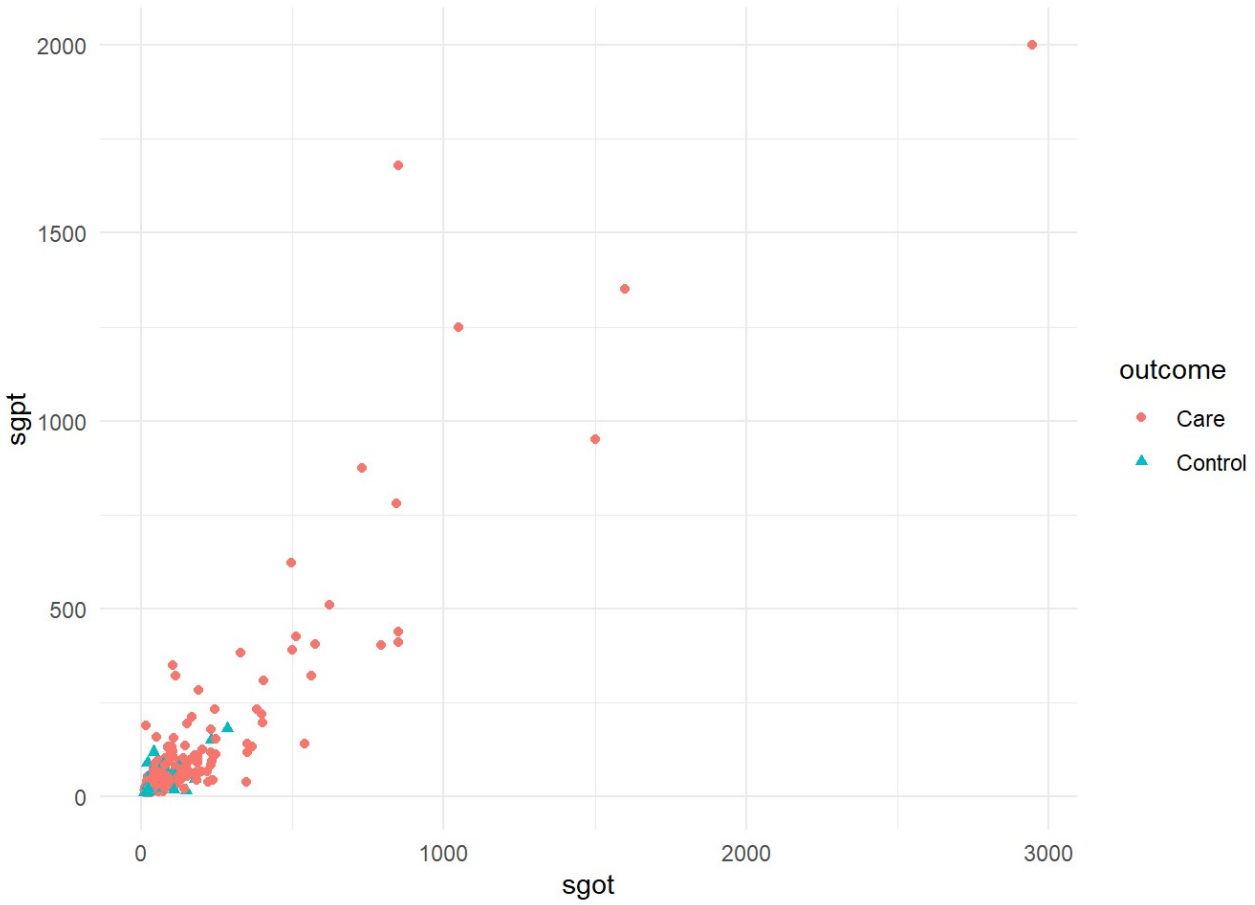
Hence it is sensible to drop one of these from the list of predictors.

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

Aminotransferases

The same approach can be used for the Aspartate aminotransferase and Alanine aminotransferase columns.

```
train %>%
  ggplot(aes(x = sgot, y = sgpt, color = outcome, shape = outcome)) +
    geom_point() +
    theme_minimal()
```



The distribution is fairly constant thus `sgpt` is removed from the list. `sgot` was kept as it displays more variance.

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

Proteins

The third pair of predictors to be investigated are total protein and albumin.

```
train %>%
  ggplot(aes(x = tp, y = alb, color = outcome, shape = outcome)) +
    geom_point() +
    theme_minimal()
```

As there are no significant outliers and both distribution are in similar pattern, we can eliminate albumin from the list of predictors.

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

Liver enzymes are important parameters in liver function test, although Aspartate aminotransferase and Alanine aminotransferase has correlation of more than 0.7, all enzymes parameters will be kept in the dataset.

Machine Learning Methods

Machine learning refers to a variety of techniques dealing with pattern recognition based on models for classification and the prediction of new data. As this dataset demonstrated some signs of correlation, machine learning is used to identify patients who may benefit from receiving hepatologist care.

A few metrics to explore in this project: a) Accuracy - the fraction of predictions our model got right b) Sensitivity -also called the true positive rate, measures the proportion of actual positives that are correctly identified as such c) Specificity - also called true negative rate, relates to the classifier's ability to identify negative results

The results can be saved after each model, to allow for easier comparison.

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

Naive Bayes

Naive Bayes is a simple, yet effective and commonly-used, machine learning classifier. This calculates a series of conditional probabilities, and the probabilities of each possible outcome.

```
nb_model = train(outcome ~ ., data = train, method = "nb")
predictions = predict(nb_model, newdata = test)
confusionMatrix <- confusionMatrix(predictions, test$outcome)
results[nrow(results) + 1, ] <- c(as.character('Naive Bayes (nb)'),
                                  confusionMatrix$overall['Accuracy'],
                                  confusionMatrix$byClass['Sensitivity'],
                                  confusionMatrix$byClass['Specificity'])
rm(nb_model, predictions)
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      83      16
##    Control   41      33
##
##                Accuracy : 0.6705
##                  95% CI : (0.5951, 0.74)
##     No Information Rate : 0.7168
##     P-Value [Acc > NIR] : 0.922614
##
##                   Kappa : 0.297
##
##  Mcnemar's Test P-Value : 0.001478
##
##             Sensitivity : 0.6694
##             Specificity : 0.6735
##          Pos Pred Value : 0.8384
##          Neg Pred Value : 0.4459
##              Prevalence : 0.7168
##          Detection Rate : 0.4798
##    Detection Prevalence : 0.5723
##       Balanced Accuracy : 0.6714
##
##        'Positive' Class : Care
##
```

The model provides the accuracy of 0.67 which is unsatisfactory. The sensitivity is of 0.62 is also quite low as compared to the specificity of 0.796. The main imitation of Naive Bayes is the assumption of independent predictors. However, it is almost impossible that we get a set of predictors which are completely independent in real life.

Linear Classifier

A linear classifier makes a classification decision based on the value of a linear combination of the characteristics. In this case, the “Boosted Generalized Linear Model” is applied.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care     124      48
##    Control    0       1
##
##                Accuracy : 0.7225
##                  95% CI : (0.6495, 0.7878)
##     No Information Rate : 0.7168
##     P-Value [Acc > NIR] : 0.4712
##
##                   Kappa : 0.029
##
##  Mcnemar's Test P-Value : 1.17e-11
##
##             Sensitivity : 1.00000
##             Specificity : 0.02041
##          Pos Pred Value : 0.72093
##          Neg Pred Value : 1.00000
##              Prevalence : 0.71676
##          Detection Rate : 0.71676
##    Detection Prevalence : 0.99422
##       Balanced Accuracy : 0.51020
##
##        'Positive' Class : Care
##
```

Although there is improvement of accuracy (0.72), the result of confusion matrix predicted that nearly every patient needed liver care. The sensitivity of the number of patients identified correctly is 1 which indicates that all patients with liver disease receiving hepatic care have flagged liver function test. The specificity (true negative rate) is incredibly low (0.02). Besides, another shortcoming of this approach is that the significantly higher number in the group with liver disease as compared with control group which can contribute to the higher accuracy metric.

Global prevalence of chronic liver disease is around 6%.It is worthwhile to include prevalence into the prediction of accuracy and repeat 2 previous confusion matrices by using the estimation of 6%.

```
nb_model = train(outcome ~ ., data = train, method = "nb")
predictions = predict(nb_model, newdata = test)
confusionMatrix <- confusionMatrix(predictions, test$outcome, prevalence = 0.06)
rm(nb_model, predictions)
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      83      16
##    Control   41      33
##
##                Accuracy : 0.6705
##                  95% CI : (0.5951, 0.74)
##     No Information Rate : 0.7168
##     P-Value [Acc > NIR] : 0.922614
##
##                   Kappa : 0.297
##
##  Mcnemar's Test P-Value : 0.001478
##
##             Sensitivity : 0.6694
##             Specificity : 0.6735
##          Pos Pred Value : 0.1157
##          Neg Pred Value : 0.9696
##              Prevalence : 0.0600
##          Detection Rate : 0.4798
##    Detection Prevalence : 0.5723
##       Balanced Accuracy : 0.6714
##
##        'Positive' Class : Care
##
```

```
lc_model = train(outcome ~ ., data = train, method = "glmboost")
predictions = predict(lc_model, newdata = test)
confusionMatrix <- confusionMatrix(predictions, test$outcome, prevalence = 0.06)
rm(lc_model, predictions)
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care     124      48
##    Control    0       1
##
##                Accuracy : 0.7225
##                  95% CI : (0.6495, 0.7878)
##     No Information Rate : 0.7168
##     P-Value [Acc > NIR] : 0.4712
##
##                   Kappa : 0.029
##
##  Mcnemar's Test P-Value : 1.17e-11
##
##             Sensitivity : 1.00000
##             Specificity : 0.02041
##          Pos Pred Value : 0.06117
##          Neg Pred Value : 1.00000
##              Prevalence : 0.06000
##          Detection Rate : 0.71676
##    Detection Prevalence : 0.99422
##       Balanced Accuracy : 0.51020
##
##        'Positive' Class : Care
##
```

Positive Predictive Value and Negative Predictive Value are 2 important indicators to be used for medical screening tool.

When Naive Bayes model is in place, 62% of patients at risk were successfully picked up with flagged test results, out of which 16% of them require further care, and 97% of them who are initially identified as not at risk were actually healthy.

Under linear classifier model, it is able to to detect almost all of the at-risk patients (99%), however only 6% of those with abnormal liver blood test results are actually having liver disease. 97% of patients who have normal results are free from liver diseases.

Logistic Regression

Linear Regression is a machine learning algorithm targeting prediction value based on independent variables.The prediction is run by using the Bayesian Generalized Linear Model and actual prevalence in confusion matrix.

```
lr_model = train(outcome ~ ., data = train, method = "bayesglm")
predictions = predict(lr_model, newdata = test)
confusionMatrix <- confusionMatrix(predictions, test$outcome, prevalence = 0.06)
results[nrow(results) + 1, ] <- c(as.character('Logistic Regression (bayesgl
m)'),                                    confusionMatrix$overall['Accuracy'],
                                  confusionMatrix$byClass['Sensitivity'],
                                  confusionMatrix$byClass['Specificity'])
rm(lr_model, predictions)
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care     116      39
##    Control    8      10
##
##                Accuracy : 0.7283
##                  95% CI : (0.6556, 0.7931)
##     No Information Rate : 0.7168
##     P-Value [Acc > NIR] : 0.4046
##
##                   Kappa : 0.1726
##
##  Mcnemar's Test P-Value : 1.209e-05
##
##             Sensitivity : 0.93548
##             Specificity : 0.20408
##          Pos Pred Value : 0.06979
##          Neg Pred Value : 0.98022
##              Prevalence : 0.06000
##          Detection Rate : 0.67052
##    Detection Prevalence : 0.89595
##       Balanced Accuracy : 0.56978
##
##        'Positive' Class : Care
##
```

Accurary remains the same as compared to the previous model, 0.72.The positive predictive value is not still low (7%) but the negative predictive value has a slight increase to 98%.

K-Nearest Neighbours

K-nearest neighbours is is a simple algorithm that and classifies new cases based on a similarity measure by finding the k closest matching points from training data (the nearest neighbours to any patient will be patients with similar characteristics (age, sex or blood results). It also better adapts to the non-linear shape of data pattern.

This is a powerful idea, as the nearest neighbours to any patient will be patients with similar profiles (demographics and blood results), so may be more likely to have the same outcome.

Using k value as tuning parameter:

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

```
## k-Nearest Neighbors
##
## 406 samples
##   7 predictor
##   2 classes: 'Care', 'Control'
##
## Pre-processing: nearest neighbor imputation (7), centered (7), scaled (7)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 406, 406, 406, 406, 406, 406, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6445217  0.1222507
##   7  0.6486127  0.1075331
##   9  0.6597048  0.1258183
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

As we can see from the above, the k value of 9 has the higher accuracy as compared to the others since we are using relatively smaller sample. Larger k value result in smoother estimates, while smaller k value lead to more flexible and more wiggly estimates.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care     102      34
##    Control   22      15
##
##                Accuracy : 0.6763
##                  95% CI : (0.6011, 0.7453)
##     No Information Rate : 0.7168
##     P-Value [Acc > NIR] : 0.8960
##
##                   Kappa : 0.139
##
##  Mcnemar's Test P-Value : 0.1416
##
##             Sensitivity : 0.82258
##             Specificity : 0.30612
##          Pos Pred Value : 0.07035
##          Neg Pred Value : 0.96433
##              Prevalence : 0.06000
##          Detection Rate : 0.58960
##    Detection Prevalence : 0.78613
##       Balanced Accuracy : 0.56435
##
##        'Positive' Class : Care
##
```

There is not much improvement on sensitivity or specificiy nor positive predictive or negative predictive value.

Random Forest

Random forest is the last algorithm that we are using here for machine learning analysis. It applies bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

```
rf_model = train(outcome ~ ., data = train, method = "rf")
predictions = predict(rf_model, newdata = test)
confusionMatrix <- confusionMatrix(predictions, test$outcome, prevalence = 0.06)
results[nrow(results) + 1, ] <- c(as.character('Random Forest (rf)'),
                                  confusionMatrix$overall['Accuracy'],
                                  confusionMatrix$byClass['Sensitivity'],
                                  confusionMatrix$byClass['Specificity'])
rm(rf_model, predictions)
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care     111      36
##    Control   13      13
##
##                Accuracy : 0.7168
##                  95% CI : (0.6434, 0.7825)
##     No Information Rate : 0.7168
##     P-Value [Acc > NIR] : 0.538423
##
##                   Kappa : 0.187
##
##  Mcnemar's Test P-Value : 0.001673
##
##             Sensitivity : 0.89516
##             Specificity : 0.26531
##          Pos Pred Value : 0.07216
##          Neg Pred Value : 0.97540
##              Prevalence : 0.06000
##          Detection Rate : 0.64162
##    Detection Prevalence : 0.84971
##       Balanced Accuracy : 0.58023
##
##        'Positive' Class : Care
##
```

```
rm(confusionMatrix)
```

Again the result is not more superior than the outcome obtained from prevous models.

Filter the dataset with liver enzymes higher than 2 times of upper limit of normal range

According to Malaysia Liver Guideline, the indication to refer for hepatologist is when the liver enzymes raised higher than 2 times of upper limit of normal range. Therefore, I'm going to filter the dataset to check whether there is any improvement of the matrices.

Creating another sets of training and test sets

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

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

```
fullData2<-fullData%>%filter(alkphos>266&sgpt>112&sgot>70)
trainIndex2 <- createDataPartition(fullData2$outcome,times=1,p=0.7, list=FALSE)
train2 <- fullData2[trainIndex2,]
test2 <- fullData2[-trainIndex2,]
rm(trainIndex2)

head(fullData2)
```

```
##   age    sex  tb  db alkphos sgpt sgot  tp alb  ag outcome
## 1  38   Male 1.8 0.8     342  168  441 7.6 4.4 1.3    Care
## 2  40 Female 0.9 0.3     293  232  245 6.8 3.1 0.8    Care
## 3  40 Female 0.9 0.3     293  232  245 6.8 3.1 0.8    Care
## 4  34   Male 4.1 2.0     289  875  731 5.0 2.7 1.1    Care
## 5  34   Male 4.1 2.0     289  875  731 5.0 2.7 1.1    Care
## 6  47   Male 2.7 1.3     275  123   73 6.2 3.3 1.1    Care
```

```
nrow(fullData2)
```

```
## [1] 47
```

Machine Learning Methods

Naive Bayes

```
nb_model2 = train(outcome ~ ., data = train2, method = "nb")
predictions2 = predict(nb_model2, newdata = test2)
confusionMatrix2 <- confusionMatrix(predictions2, test2$outcome)
rm(nb_model2, predictions2)
confusionMatrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      10       0
##    Control    3       0
##
##                Accuracy : 0.7692
##                  95% CI : (0.4619, 0.9496)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 0.7692
##             Specificity :     NA
##          Pos Pred Value :     NA
##          Neg Pred Value :     NA
##              Prevalence : 1.0000
##          Detection Rate : 0.7692
##    Detection Prevalence : 0.7692
##       Balanced Accuracy :     NA
##
##        'Positive' Class : Care
##
```

Sensitivity reported is 0.77 and specificity is not availbale due to small size of sample obtained.

Linear Classifier

```
lc_model2 = train(outcome ~ ., data = train2, method = "glmboost")
predictions2 = predict(lc_model2, newdata = test2)
confusionMatrix2 <- confusionMatrix(predictions2, test2$outcome)
rm(lc_model2, predictions2)
confusionMatrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      13       0
##    Control    0       0
##
##                Accuracy : 1
##                  95% CI : (0.7529, 1)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : NaN
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity :  1
##             Specificity : NA
##          Pos Pred Value : NA
##          Neg Pred Value : NA
##              Prevalence :  1
##          Detection Rate :  1
##    Detection Prevalence :  1
##       Balanced Accuracy : NA
##
##        'Positive' Class : Care
##
```

Sensitivity reported is 1 and specificity is not availbale due to small size of sample obtained.

Logistic regression

```
lr_model2 = train(outcome ~ ., data = train2, method = "bayesglm")
predictions2 = predict(lr_model2, newdata = test2)
confusionMatrix2 <- confusionMatrix(predictions2, test2$outcome, prevalence = 0.06)
rm(lr_model2, predictions2)
confusionMatrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      13       0
##    Control    0       0
##
##                Accuracy : 1
##                  95% CI : (0.7529, 1)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : NaN
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.00
##             Specificity :   NA
##          Pos Pred Value :   NA
##          Neg Pred Value :   NA
##              Prevalence : 0.06
##          Detection Rate : 1.00
##    Detection Prevalence : 1.00
##       Balanced Accuracy :   NA
##
##        'Positive' Class : Care
##
```

K-Nearest Neighbours

```
knn_model2 = train(outcome ~ ., data = train2, method = "knn", preProcess=c('knnImput
e'))
knn_model2
```

```
## k-Nearest Neighbors
##
## 34 samples
## 10 predictors
##  2 classes: 'Care', 'Control'
##
## Pre-processing: nearest neighbor imputation (10), centered (10),
##  scaled (10)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 34, 34, 34, 34, 34, 34, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.8927703  -0.014776266
##   7  0.9250347  -0.004784689
##   9  0.9224340   0.000000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

```
predictions2 = predict(knn_model2, newdata = test2)
confusionMatrix2 <- confusionMatrix(predictions2, test2$outcome, prevalence = 0.06)
rm(knn_model2, predictions2)
confusionMatrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      13       0
##    Control    0       0
##
##                Accuracy : 1
##                  95% CI : (0.7529, 1)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : NaN
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.00
##             Specificity :   NA
##          Pos Pred Value :   NA
##          Neg Pred Value :   NA
##              Prevalence : 0.06
##          Detection Rate : 1.00
##    Detection Prevalence : 1.00
##       Balanced Accuracy :   NA
##
##        'Positive' Class : Care
##
```

Random Forest

```
rf_model2 = train(outcome ~ ., data = train2, method = "rf")
predictions2 = predict(rf_model2, newdata = test2)
confusionMatrix2 <- confusionMatrix(predictions2, test2$outcome, prevalence = 0.06)
rm(rf_model2, predictions2)
confusionMatrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      13       0
##    Control    0       0
##
##                Accuracy : 1
##                  95% CI : (0.7529, 1)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : NaN
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.00
##             Specificity :   NA
##          Pos Pred Value :   NA
##          Neg Pred Value :   NA
##              Prevalence : 0.06
##          Detection Rate : 1.00
##    Detection Prevalence : 1.00
##       Balanced Accuracy :   NA
##
##        'Positive' Class : Care
##
```

In summary, from the 5 models that we had tested, besides NB model reported sensitivity of 0.77, the rest of the models demonstrated sensitivity of 1. It indicates that if all liver enzymes are at least 2 times higher than the upper limit of normal range, it is almost certain that the patients truely have the liver diseases. However, as the sample size is small, the specificiy is not able to be generated.

Filter the dataset with liver enzymes raised but less than 2 times of upper limit of normal range

Now we filter the dataset with any of the liver enzymes raised more than normal but lower than the 2 times of the upper limit of normal range.

Creating another sets of training and test sets

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

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

```
fullData3<-fullData%>%filter(alkphos%in%(133:266)|sgpt%in%(56:112)|sgot%in%(35:70))
trainIndex3 <- createDataPartition(fullData3$outcome, times=1, p=0.7, list=FALSE)
train3 <- fullData3[trainIndex3,]
test3 <- fullData3[-trainIndex3,]
rm(trainIndex3)

head(fullData3)
```

```
##   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    Care
## 2  62   Male 10.9 5.5     699   64  100 7.5 3.2 0.74    Care
## 3  62   Male  7.3 4.1     490   60   68 7.0 3.3 0.89    Care
## 4  58   Male  1.0 0.4     182   14   20 6.8 3.4 1.00    Care
## 5  72   Male  3.9 2.0     195   27   59 7.3 2.4 0.40    Care
## 6  46   Male  1.8 0.7     208   19   14 7.6 4.4 1.30    Care
```

```
nrow(fullData3)
```

```
## [1] 461
```

Machine Learning Methods

Naive Bayes

```
nb_model3 = train(outcome ~ ., data = train3, method = "nb")
predictions3 = predict(nb_model3, newdata = test3)
confusionMatrix3 <- confusionMatrix(predictions3, test3$outcome)
rm(nb_model3, predictions3)
confusionMatrix3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      49       5
##    Control   44      39
##
##                Accuracy : 0.6423
##                  95% CI : (0.556, 0.7223)
##     No Information Rate : 0.6788
##     P-Value [Acc > NIR] : 0.8429
##
##                   Kappa : 0.335
##
##  Mcnemar's Test P-Value : 5.681e-08
##
##             Sensitivity : 0.5269
##             Specificity : 0.8864
##          Pos Pred Value : 0.9074
##          Neg Pred Value : 0.4699
##              Prevalence : 0.6788
##          Detection Rate : 0.3577
##    Detection Prevalence : 0.3942
##       Balanced Accuracy : 0.7066
##
##        'Positive' Class : Care
##
```

Sensitivity is 0.52 and specificity is 0.88.

Linear Classifier

```
lc_model3 = train(outcome ~ ., data = train3, method = "glmboost")
predictions3 = predict(lc_model3, newdata = test3)
confusionMatrix3 <- confusionMatrix(predictions3, test3$outcome)
rm(lc_model3, predictions3)
confusionMatrix3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      91      40
##    Control    2       4
##
##                Accuracy : 0.6934
##                  95% CI : (0.609, 0.7693)
##     No Information Rate : 0.6788
##     P-Value [Acc > NIR] : 0.3958
##
##                   Kappa : 0.0898
##
##  Mcnemar's Test P-Value : 1.135e-08
##
##             Sensitivity : 0.97849
##             Specificity : 0.09091
##          Pos Pred Value : 0.69466
##          Neg Pred Value : 0.66667
##              Prevalence : 0.67883
##          Detection Rate : 0.66423
##    Detection Prevalence : 0.95620
##       Balanced Accuracy : 0.53470
##
##        'Positive' Class : Care
##
```

Sensitivity is 0.99 and specificy is 0.02.

Logistic regression

```
lr_model3 = train(outcome ~ ., data = train3, method = "bayesglm")
predictions3 = predict(lr_model3, newdata = test3)
confusionMatrix3 <- confusionMatrix(predictions3, test3$outcome, prevalence = 0.06)
rm(lr_model3, predictions3)
confusionMatrix3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      86      25
##    Control    7      19
##
##                Accuracy : 0.7664
##                  95% CI : (0.6866, 0.8344)
##     No Information Rate : 0.6788
##     P-Value [Acc > NIR] : 0.015795
##
##                   Kappa : 0.3996
##
##  Mcnemar's Test P-Value : 0.002654
##
##             Sensitivity : 0.92473
##             Specificity : 0.43182
##          Pos Pred Value : 0.09411
##          Neg Pred Value : 0.98900
##              Prevalence : 0.06000
##          Detection Rate : 0.62774
##    Detection Prevalence : 0.81022
##       Balanced Accuracy : 0.67827
##
##        'Positive' Class : Care
##
```

Sensitivity is 0.92 and specificiy is 0.43.

K-Nearest Neighbours

```
knn_model3 = train(outcome ~ ., data = train3, method = "knn", preProcess=c('knnImput
e'))
knn_model3
```

```
## k-Nearest Neighbors
##
## 324 samples
##  10 predictor
##   2 classes: 'Care', 'Control'
##
## Pre-processing: nearest neighbor imputation (10), centered (10),
##  scaled (10)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 324, 324, 324, 324, 324, 324, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6198267  0.1164703
##   7  0.6305956  0.1244723
##   9  0.6344841  0.1253542
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
predictions3 = predict(knn_model3, newdata = test3)
confusionMatrix3 <- confusionMatrix(predictions3, test3$outcome, prevalence = 0.06)
rm(knn_model3, predictions3)
confusionMatrix3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      71      37
##    Control   22       7
##
##                Accuracy : 0.5693
##                  95% CI : (0.482, 0.6536)
##     No Information Rate : 0.6788
##     P-Value [Acc > NIR] : 0.99724
##
##                   Kappa : -0.0851
##
##  Mcnemar's Test P-Value : 0.06836
##
##             Sensitivity : 0.76344
##             Specificity : 0.15909
##          Pos Pred Value : 0.05478
##          Neg Pred Value : 0.91332
##              Prevalence : 0.06000
##          Detection Rate : 0.51825
##    Detection Prevalence : 0.78832
##       Balanced Accuracy : 0.46127
##
##        'Positive' Class : Care
##
```

Sensitivity is0.76 and specificity is 0.16.

Random Forest

```
rf_model3 = train(outcome ~ ., data = train3, method = "rf")
predictions3 = predict(rf_model3, newdata = test3)
confusionMatrix3 <- confusionMatrix(predictions3, test3$outcome, prevalence = 0.06)
rm(rf_model3, predictions3)
confusionMatrix3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care      77      27
##    Control   16      17
##
##                Accuracy : 0.6861
##                  95% CI : (0.6013, 0.7627)
##     No Information Rate : 0.6788
##     P-Value [Acc > NIR] : 0.4679
##
##                   Kappa : 0.2294
##
##  Mcnemar's Test P-Value : 0.1273
##
##             Sensitivity : 0.82796
##             Specificity : 0.38636
##          Pos Pred Value : 0.07929
##          Neg Pred Value : 0.97236
##              Prevalence : 0.06000
##          Detection Rate : 0.56204
##    Detection Prevalence : 0.75912
##       Balanced Accuracy : 0.60716
##
##        'Positive' Class : Care
##
```

Sensitivity obtained by Random Forest model is 0.82 and specificity is 0.39. Generally the sensivity decreased compared to the previous dataset above, the sensitivity carry a wider range of 0.52-0.99 and likewise to specificity which account from 0.02-0.88. Overall the senstivity for this range of liver enzymes are lower-ability to pick up true liver diseases had dropped if compared to the liver enzyme range fo more than 2 times of upper limit of normal range.

Filter the dataset with liver enzymes which are within normal range

Finally we use the filter if all liver enzymes are within the normal value.

Creating another sets of training and test sets

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

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

```
fullData4<-fullData%>%filter(alkphos%in%(41:133)&sgpt%in%(7:56)&sgot%in%(0:35))
trainIndex4 <- createDataPartition(fullData4$outcome,times=1,p=0.7, list=FALSE)
train4 <- fullData4[trainIndex4,]
test4 <- fullData4[-trainIndex4,]
rm(trainIndex4)

head(fullData4)
```

```
##   age    sex  tb  db alkphos sgpt sgot  tp alb   ag outcome
## 1  20   Male 1.1 0.5     128   20   30 3.9 1.9 0.95 Control
## 2  29   Male 1.0 0.3      75   25   26 5.1 2.9 1.30    Care
## 3  55   Male 0.9 0.2     116   36   16 6.2 3.2 1.00 Control
## 4  26   Male 0.6 0.1     110   15   20 2.8 1.6 1.30    Care
## 5  30 Female 0.7 0.2      63   31   27 5.8 3.4 1.40    Care
## 6  46   Male 0.6 0.2     115   14   11 6.9 3.4 0.90    Care
```

```
nrow(fullData4)
```

```
## [1] 13
```

Naive Bayes

```
nb_model4 = train(outcome ~ ., data = train4, method = "nb")
predictions4 = predict(nb_model4, newdata = test4)
confusionMatrix4 <- confusionMatrix(predictions4, test4$outcome)
rm(nb_model4, predictions4)
confusionMatrix4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care       2       1
##    Control    0       0
##
##                Accuracy : 0.6667
##                  95% CI : (0.0943, 0.9916)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.7407
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.6667
##          Neg Pred Value :    NaN
##              Prevalence : 0.6667
##          Detection Rate : 0.6667
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Care
##
```

Linear regression

```
lc_model4 = train(outcome ~ ., data = train4, method = "glmboost")
predictions4 = predict(lc_model4, newdata = test4)
confusionMatrix3 <- confusionMatrix(predictions4, test4$outcome)
rm(lc_model4, predictions4)
confusionMatrix4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care       2       1
##    Control    0       0
##
##                Accuracy : 0.6667
##                  95% CI : (0.0943, 0.9916)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.7407
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.6667
##          Neg Pred Value :    NaN
##              Prevalence : 0.6667
##          Detection Rate : 0.6667
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Care
##
```

Logistic regression

```
lr_model4 = train(outcome ~ ., data = train4, method = "bayesglm")
predictions4 = predict(lr_model4, newdata = test4)
confusionMatrix4 <- confusionMatrix(predictions4, test4$outcome, prevalence = 0.06)
rm(lr_model4, predictions4)
confusionMatrix4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care       1       1
##    Control    1       0
##
##                Accuracy : 0.3333
##                  95% CI : (0.0084, 0.9057)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.963
##
##                   Kappa : -0.5
##
##  Mcnemar's Test P-Value : 1.000
##
##             Sensitivity : 0.50000
##             Specificity : 0.00000
##          Pos Pred Value : 0.03093
##          Neg Pred Value : 0.00000
##              Prevalence : 0.06000
##          Detection Rate : 0.33333
##    Detection Prevalence : 0.66667
##       Balanced Accuracy : 0.25000
##
##        'Positive' Class : Care
##
```

K-Nearest Neighbours

```
knn_model4 = train(outcome ~ ., data = train4, method = "knn", preProcess=c('knnImput
e'))
knn_model4
```

```
## k-Nearest Neighbors
##
## 10 samples
## 10 predictors
##  2 classes: 'Care', 'Control'
##
## Pre-processing: nearest neighbor imputation (10), centered (10),
##  scaled (10)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 10, 10, 10, 10, 10, 10, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6473333  0.09682540
##   7  0.7206667  0.03703704
##   9  0.7206667  0.00000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
predictions4 = predict(knn_model4, newdata = test4)
confusionMatrix4 <- confusionMatrix(predictions4, test4$outcome)
rm(knn_model4, predictions4)
confusionMatrix4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care       2       1
##    Control    0       0
##
##                Accuracy : 0.6667
##                  95% CI : (0.0943, 0.9916)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.7407
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.6667
##          Neg Pred Value :    NaN
##              Prevalence : 0.6667
##          Detection Rate : 0.6667
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Care
##
```

Random Forest

```
rf_model4 = train(outcome ~ ., data = train4, method = "rf")
predictions4 = predict(rf_model4, newdata = test4)
confusionMatrix4 <- confusionMatrix(predictions4, test4$outcome, prevalence = 0.06)
rm(rf_model4, predictions4)
confusionMatrix4
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Care Control
##    Care       1       1
##    Control    1       0
##
##                Accuracy : 0.3333
##                  95% CI : (0.0084, 0.9057)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.963
##
##                   Kappa : -0.5
##
##  Mcnemar's Test P-Value : 1.000
##
##             Sensitivity : 0.50000
##             Specificity : 0.00000
##          Pos Pred Value : 0.03093
##          Neg Pred Value : 0.00000
##              Prevalence : 0.06000
##          Detection Rate : 0.33333
##    Detection Prevalence : 0.66667
##       Balanced Accuracy : 0.25000
##
##        'Positive' Class : Care
##
```

Over the 5 models that we had run, Logistic regression and Random Forest had generated sensitivity of 0.5 and the rest with sensitivity of 0. The result is not so reliable as the sample size is very small (all 3 liver enzymes are normal). It also gives an impression that abnormality in any one of the liver enzymes are not uncommon even in healthy population.

Results

Comparison of the results for accuracy,sensitivity and specificity of the dataset before applying filter:

```
results %>% arrange(Accuracy)
```

```
##                            Model          Accuracy       Sensitivity
## 1               Naive Bayes (nb) 0.670520231213873 0.669354838709677
## 2     K-nearest neighbours (knn) 0.676300578034682  0.82258064516129
## 3             Random Forest (rf) 0.716763005780347 0.895161290322581
## 4   Linear Classifier (glmboost) 0.722543352601156                 1
## 5 Logistic Regression (bayesglm) 0.728323699421965 0.935483870967742
##          Specificity
## 1  0.673469387755102
## 2  0.306122448979592
## 3   0.26530612244898
## 4 0.0204081632653061
## 5  0.204081632653061
```

Linear classifier accounts for the highest sensitivity of 100% among all models but the speficity of 20% is quite low. High sensitivity indicates that among all patients flagged with abnormal liver function tests only small portion of them truly need futher work out and treatment from hepatologists. Other similar models with high sensitivity and relatively low specificity are knn model,logistic regression and random forest.

On the other hand, Naive Bayes model yields the same specificity (67%) as sensitivity(67%). Higher specificity indicates that there are fewer false positive results.

After using the filter: a) All 3 liver enzymes raised more than 2 times of the upper limit of normal range: The overall sensitivy increased, however due to small sample size,there is error/NA result for specificity.

b. Any one of the liver enzymes raised but not more than 2 times of the upper limit of normal range: The overall sensitivity became lower, and specificy ranged from 0.02-0.88(wide range).

c. All liver enzymes within normal range: The sample size is limited thus the sensitivity and specificity became not reliable.

Conclusion

After the data analysis, it can be concluded that no model has a perfect combination which makes it distinctly superior to the others.

If we are using the full dataset without filter, most of the models demonstrated high sensitivy and lower specificity. Flagged liver function tests can easily detect patients at risk but only few patients at risk are truly suffer from liver diseases. As a result, this can cause unccessary referral to hepatologists. If liver function test is in normal range, it indicates that the patient is likely to be free from liver disease since the negative predictive value is generally high from all models.

The Naive Bayes (“nb” method) approach seems to provide the highest specificity but lower sensitivity can lead to underdetection of patients at risk of liver diseases.

On the other hand,if we rely on liver enzymes for the diganosis, it is likely that we can improve the existing results by absorbing larger sample size in dataset:

When liver enzymes are raised but less than 2 times of upper limit of normal range: -There can be rise in single blood parameters eg: only raise in alkaline phosphatase or aspartate aminotransferase – this can be due to causes other than liver since these 2 enzyems are found in other organs.Eg:single rise in alkaline phosphatase can be due to bone pathology. -If liver enzymes are raised but less than 2 times of upper limit of normal range, it is worthwhile to repeat another test and not to refer the patient to liver specialist first since the sensitivity is generally lower and the specificity can be as low as 0.02.

If all liver enzymes are raised at least 2 times higher than the upper limit of normal range, it is indication to refer to hepatologists for further work out as it demonstrated that this pattern of result is very sensitive to pick up liver disease and specificity could be further determined by using larger size of dataset.

Discussion

Liver function test is sensitive but not specific. The positive predictive value is low as many patients who had abnormal liver function test are likely to be healthy and do not require any liver care ultimately. On the other hand, the negative predictive value is high which signifies that if the test is negative, it is very likely that the patient is free from liver diseases.

According to Malaysia Liver Guideline, clinicians should refer patients if the liver enzymes are at least 2-3 times higher than the upper limit of the normal range. The results from this project had shown that the current liver guideline in Malaysia is appropriate and in tandem with the results of the project.

As clinician, when we encounter the case whereby the liver enzymes are raised but not more than 2 times higher than the normal range,it is important to take a thorough history and physical examination to rule out any other possible causes which may induce the liver enzymes alteration such as alcohol intake and consumption of liver toxic drugs.

The limitation of the project included small sample size (579 patients). Besides, the dataset is from unmatched case-control study so the proportion of the patient receiving care are not in equal proportion with control group (414 vs 165) and case and control groups are not matched with same characterstic. Following this project, I plan to gather the dataset in Malaysia context with larger sample size and run the same project to prove that the methodology and result of this project is not population-specific and can be generalized to global population.

My “Liver_function_test Github repository” is **in this link (https://github.com/zenasimlab/Capsone_Liver_function_test)**

References

1. Yee HF, Lidofsky SD. Acute liver failure. In: Feldman M, Friedman LS, Sleisenger MH, eds. Sleisenger and Fordtran’s Gastrointestinal and Liver Disease: Pathophysiology, Diagnosis, Management. 7th ed. Philadelphia, Pa.: Saunders, 2002:1567–74
2. http://archive.ics.uci.edu/ml (http://archive.ics.uci.edu/ml)