Identification of Liver Patients Using Machine Learning Techniques

Matthew Higgins

5/23/2019

Contents

Introduction

Patients in hospitals commonly present with complex medical histories and significant comorbidities, making it challenging for healthcare professionals to determine whether any undiagnosed underlying conditions may be impacting their current presentation.

For example, when a patient presents with commonly seen symptoms, like pain, dizziness or fatigue, they may be examined and diagnosed with, for example, an underlying heart conditon. While this is likely to be the most significant conditon they currently have, it is possible that they have another condition which is aggravating their symptoms.

In recent years, there has been an increase in patients being diagnose with chronic liver disease, and a dataset including blood test results of patients likely to benefit from care from a hepatologist and some baseline results from a control group was easily available (UCI Machine Learning Repository, http://archive.ics.uci.edu/ml)

For this reason, some attempts will be made to find out whether there is a machine learning technique that identifies those patients who are similar to patients who receive care from a hepatologist.

This would not be intended to replace human diagnosis, but it could be run in the background against blood test results to tell whether patients who are not currently receiving hapatic care would benefit from further examination and potentially a referral for continuing care.

While some basic background to the biology involved is covered here, this is not the focus of the report, and understanding it is not necessary to appreciate the implications of the machine learning techniques.

Data Exploration

Downloading and Splitting the Data

Before being able to explore the dataset more thoroughly, the first step is to import it into a data frame, and define the names of the columns. In the following code, the comments after the column names are from the original data set's information page.

```
columnNames <- c('age', # Age of the patient
                 'sex', # Sex of the patient
                 'tb', # Total Bilirubin
                 'db', # Direct Bilirubin
                 'alkphos', # Alkaline Phosphotase
                 'sgpt', # Alamine Aminotransferase
                 'sgot', # Aspartate Aminotransferase
                 'tp', # Total Protiens
                 '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/00225/Indian%20Liver%
                       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
65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	Care
62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	Care
62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	Care
58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	Care
72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	Care
46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	Care

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

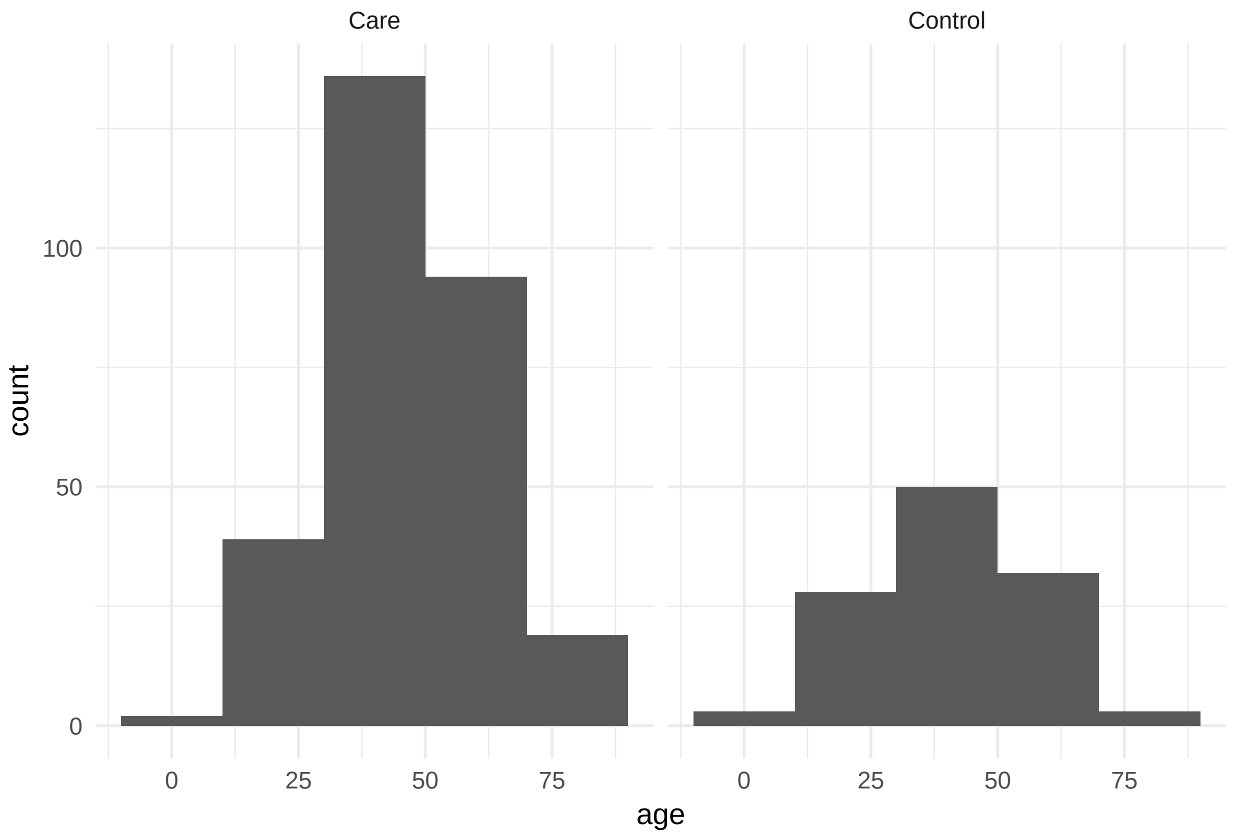
```
set.seed(1)
trainIndex <- createDataPartition(fullData$outcome, p=.7, list=FALSE)
train <- fullData[trainIndex,]
test <- fullData[-trainIndex,]
```

```
rm(trainIndex)
```

Column-by-column exploration

Age

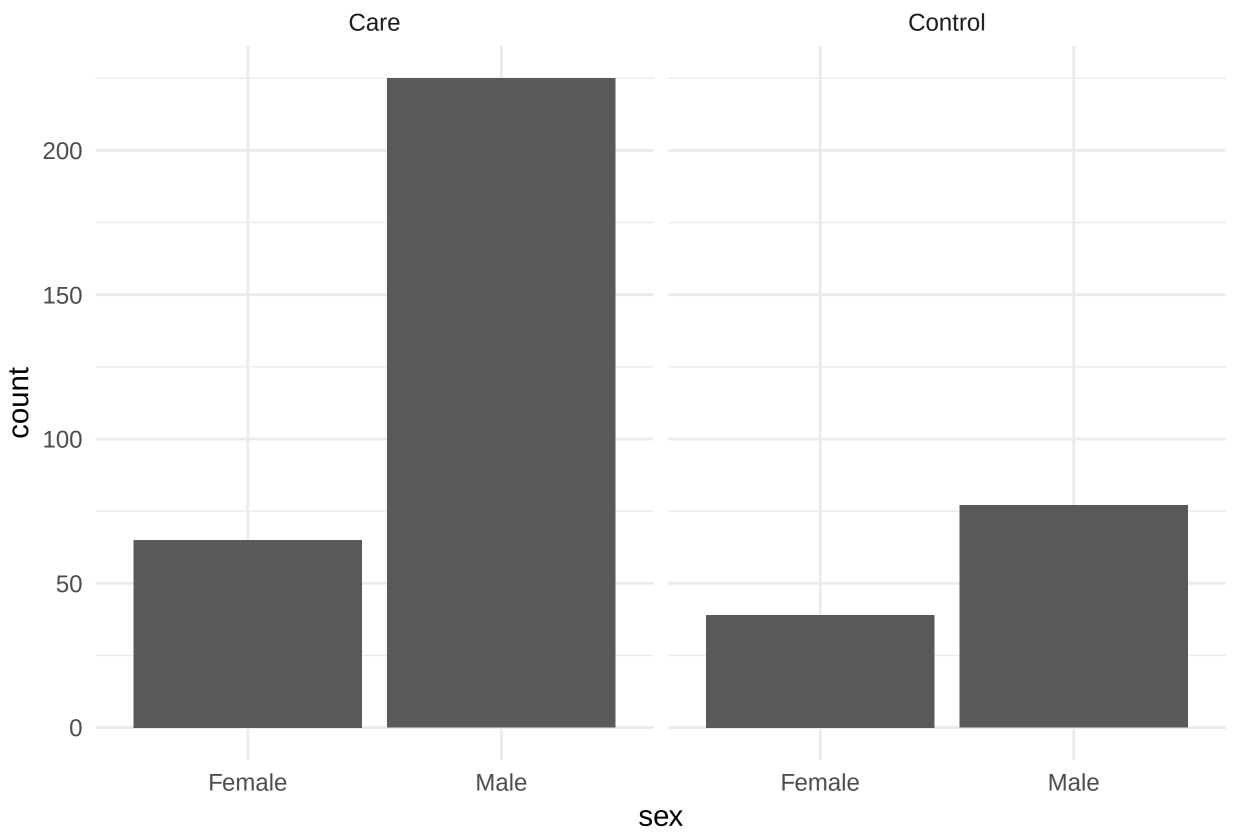
```
train %>%
  ggplot(aes(x = age)) +
    geom_histogram(binwidth = 20) +
    theme_minimal() +
    facet_grid(~ outcome)
```



With a coarse overview of the age distribution, both groups of patients appear to have similar shaped age distributions.

Sex

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



It seems that women appear less frequently than men in both the group with liver disease and the control group. The control group appears as though it may have been constructed to mimic the patients with liver disease.

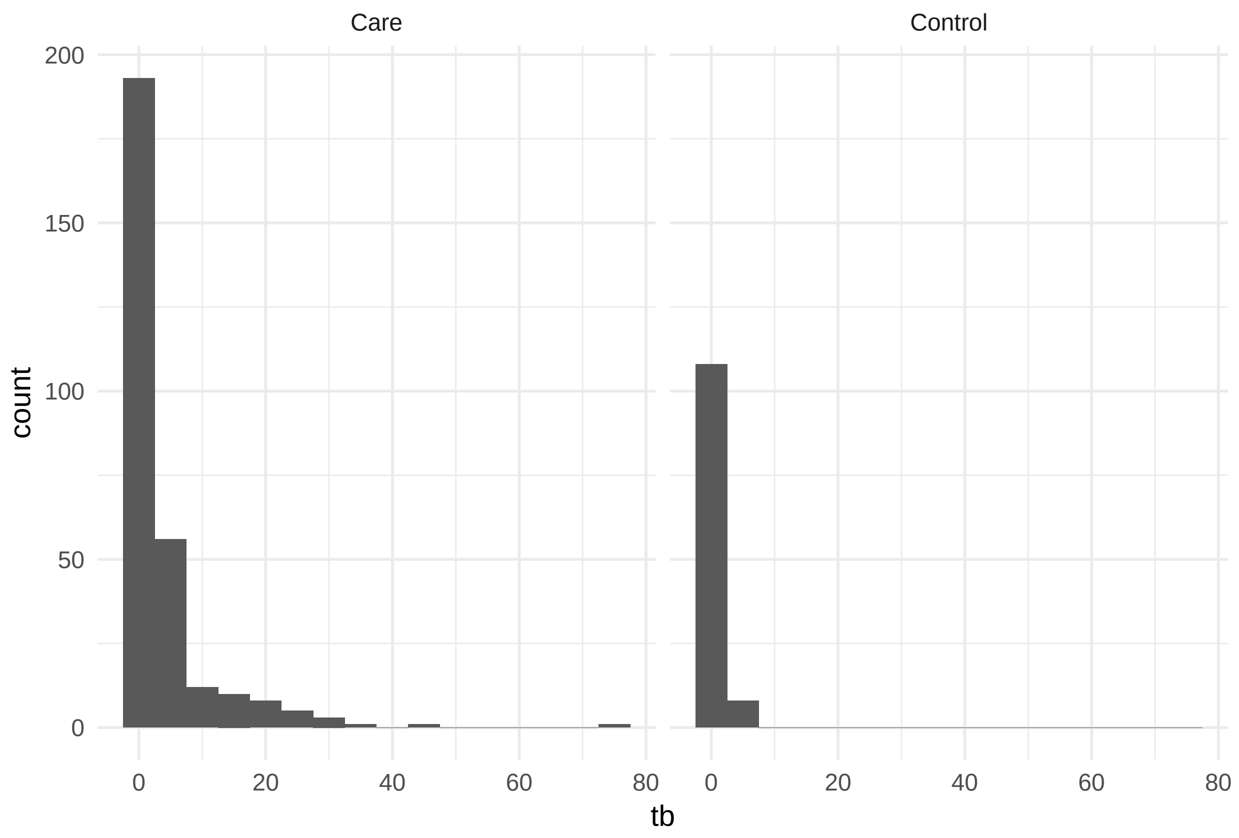
Bilirubin

Bilirubin is a compound that naturally occurs in the blood from the breakdown of heme.

Two values are included in the data;

- *Total bilirubin* which includes both conjugated and unconjugated bilirubin present in the sample.
- *Direct bilirubin* which refers to the water-soluble bilirubin present in the blood sample which is available to react with the reagents. Most of this will be conjugated bilirubin, but some may not be.

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

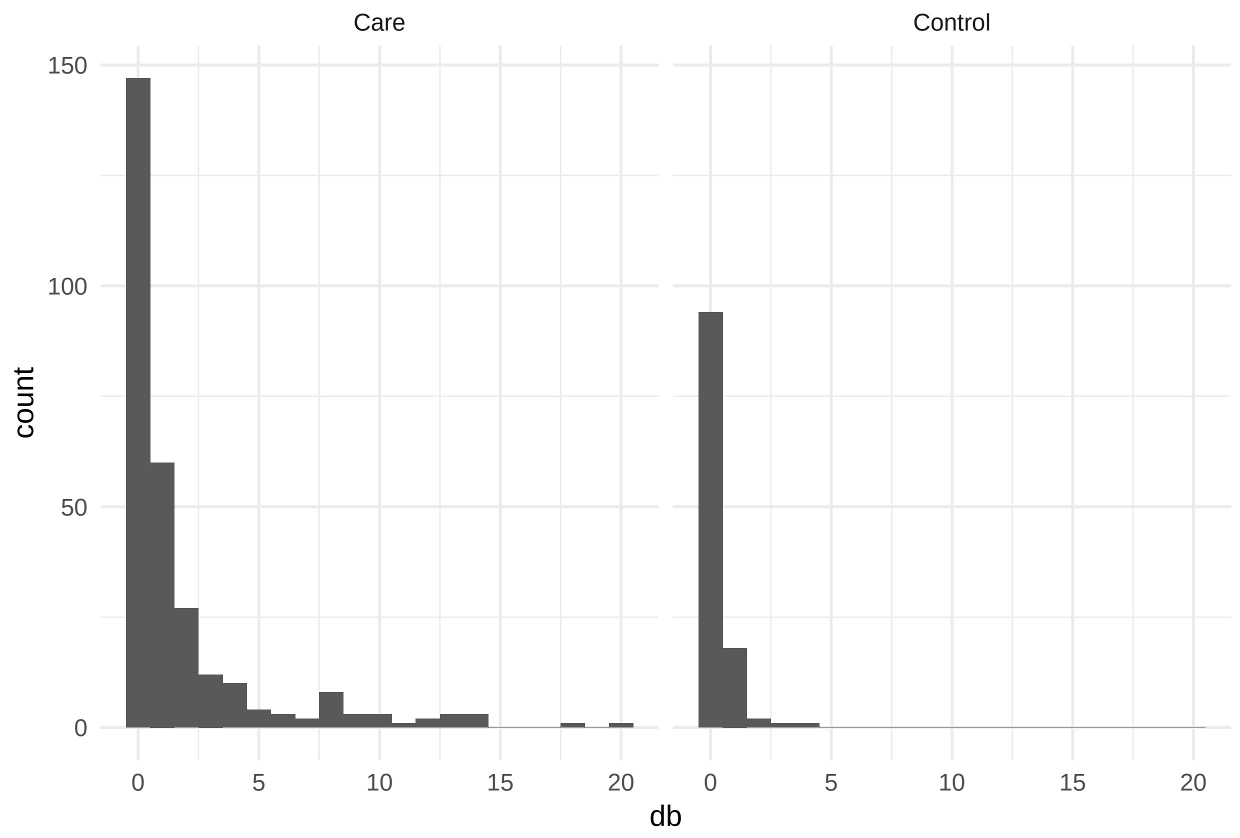


The histogram shows the total bilirubin levels of the two groups.

The striking point that can be seen immediately is the wider range of the patients with liver disease compared to the control group.

That said, it's clear that in isolation, this is not an indicator, as the most common values (the 0-5 column) is present in both groups.

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



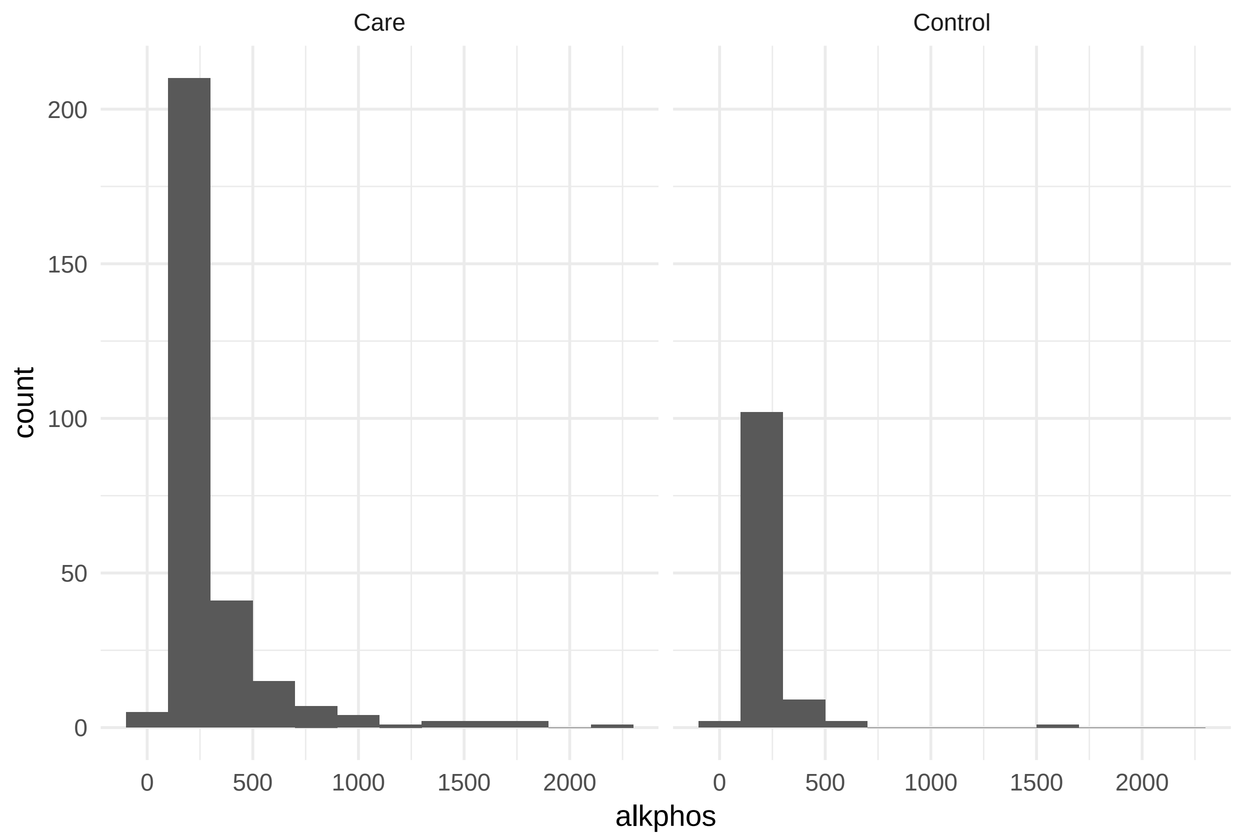
The same can be seen in this histogram of the direct bilirubin values.

Enzymes

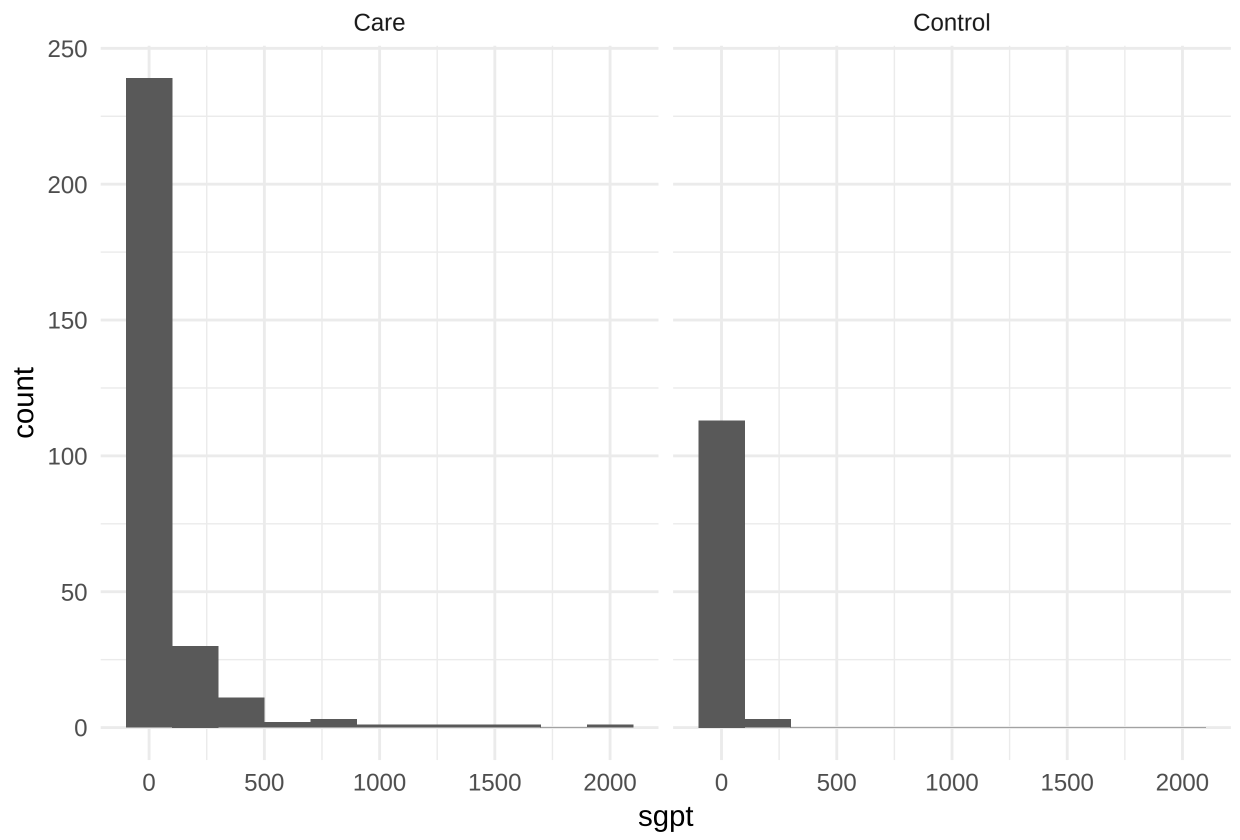
The next three columns refer to specific enzymes or groups of enzymes that may be present in the blood sample.

- *Alkaline Phosphotase* - which removes phosphate groups from organic compounds.
- *Alanine Aminotransferase* - affects the transfer of amino groups from L-alanine to ??-ketoglutarate
- *Aspartate Aminotransferase* - affects the transfer of amino groups from aspartic acid to alpha-ketoglutaric acid

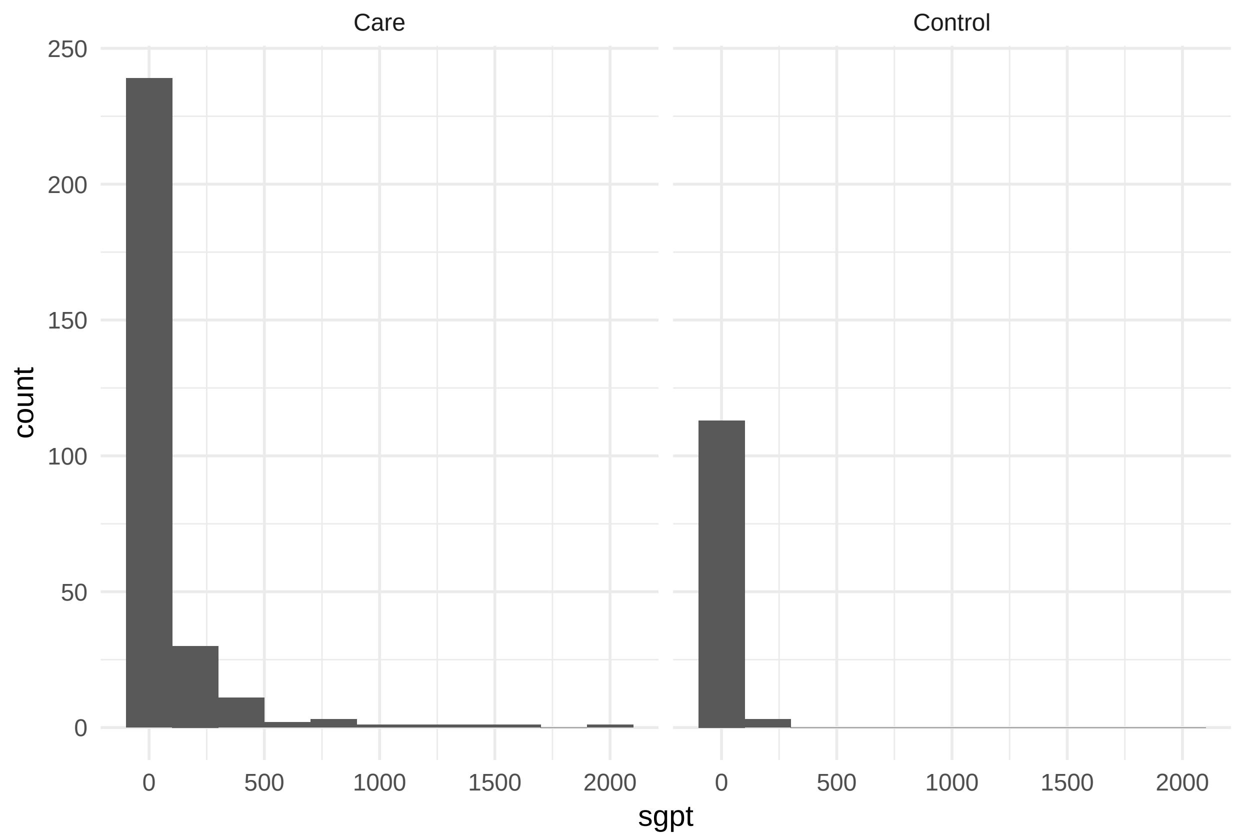
```r
train %>%
  ggplot(aes(x = alkphos)) +
    geom_histogram(binwidth = 200) +
    theme_minimal() +
    facet_grid(~ outcome)
```



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



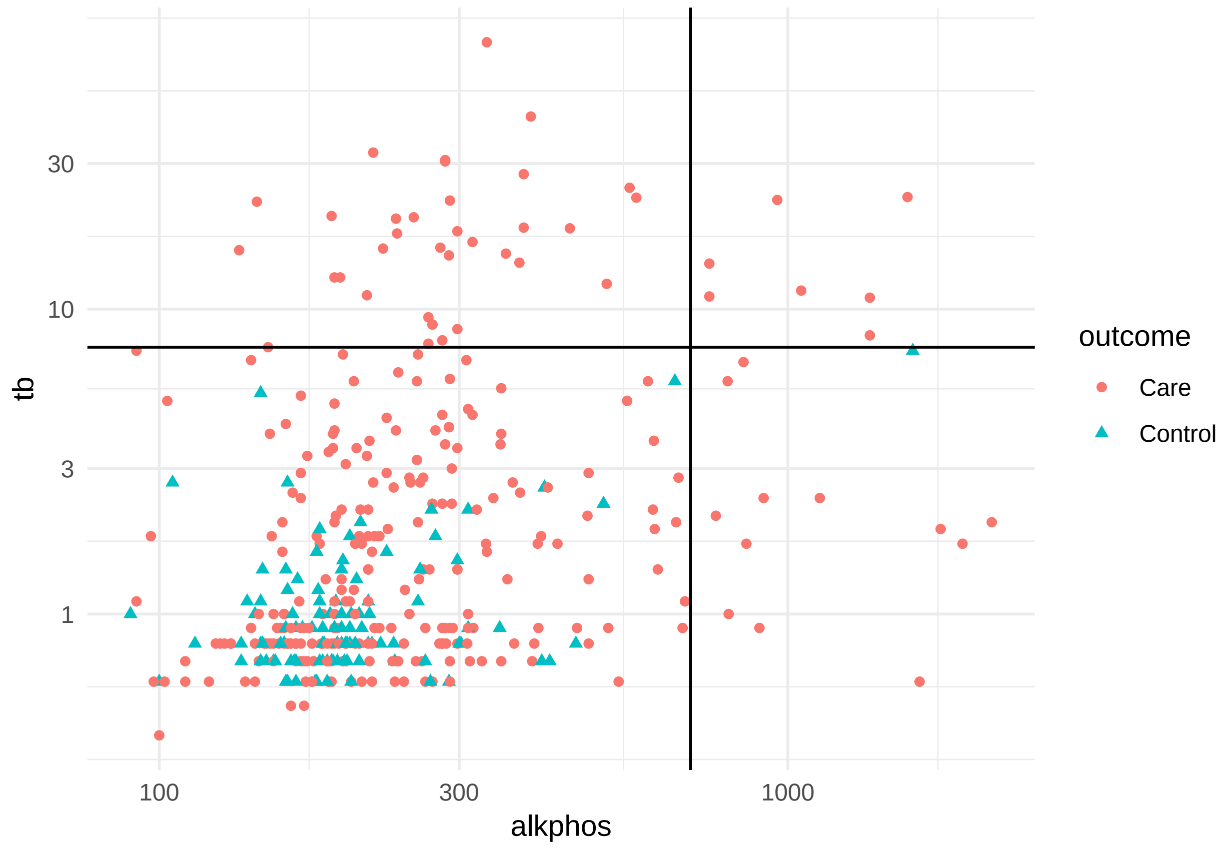
```
train %>%
  ggplot(aes(x = sgpt)) +
    geom_histogram(binwidth = 200) +
    theme_minimal() +
    facet_grid(~ outcome)
```



All of the enzyme levels show the same pattern as the bilirubin, where a wider range is seen in the group with liver disease, but many values are the same.

To see whether the outliers for one measure are also outliers for the other, the bilirubin level can be plotted against one of the enzymes.

```
train %>%
  ggplot(aes(x = alkphos, 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()
```



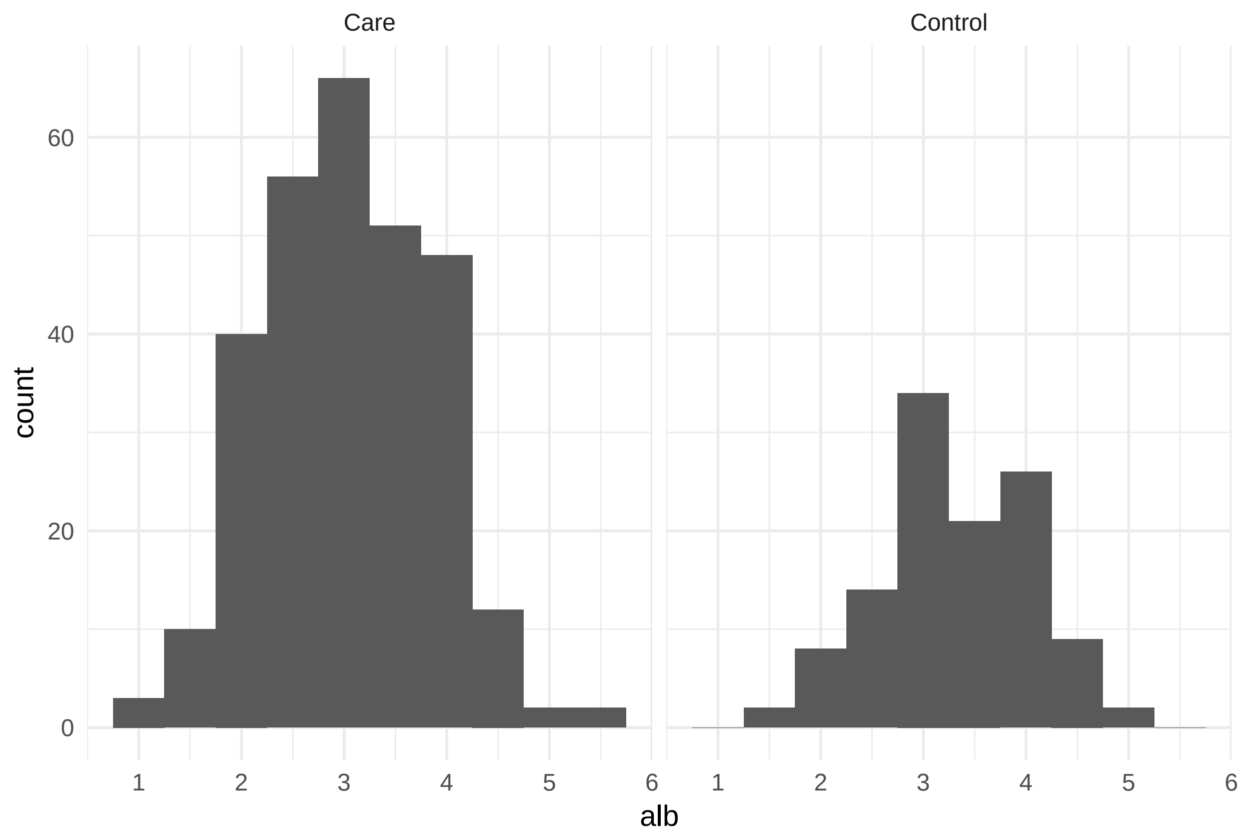
The blue lines (which are added manually using fixed values) seem to show that while practically all of the healthy patients are in the lower left quadrant, the patients with liver disease who are not most commonly have either a high `alkphos` with a normal `tb` (lower right) or a high `tb` with a normal `alkphose` (upper left).

The quadrant for patients with both elevated `tb` and `alkphos` (top right) contains comparatively quite few, but all of these are patients in the group receiving care.

Proteins

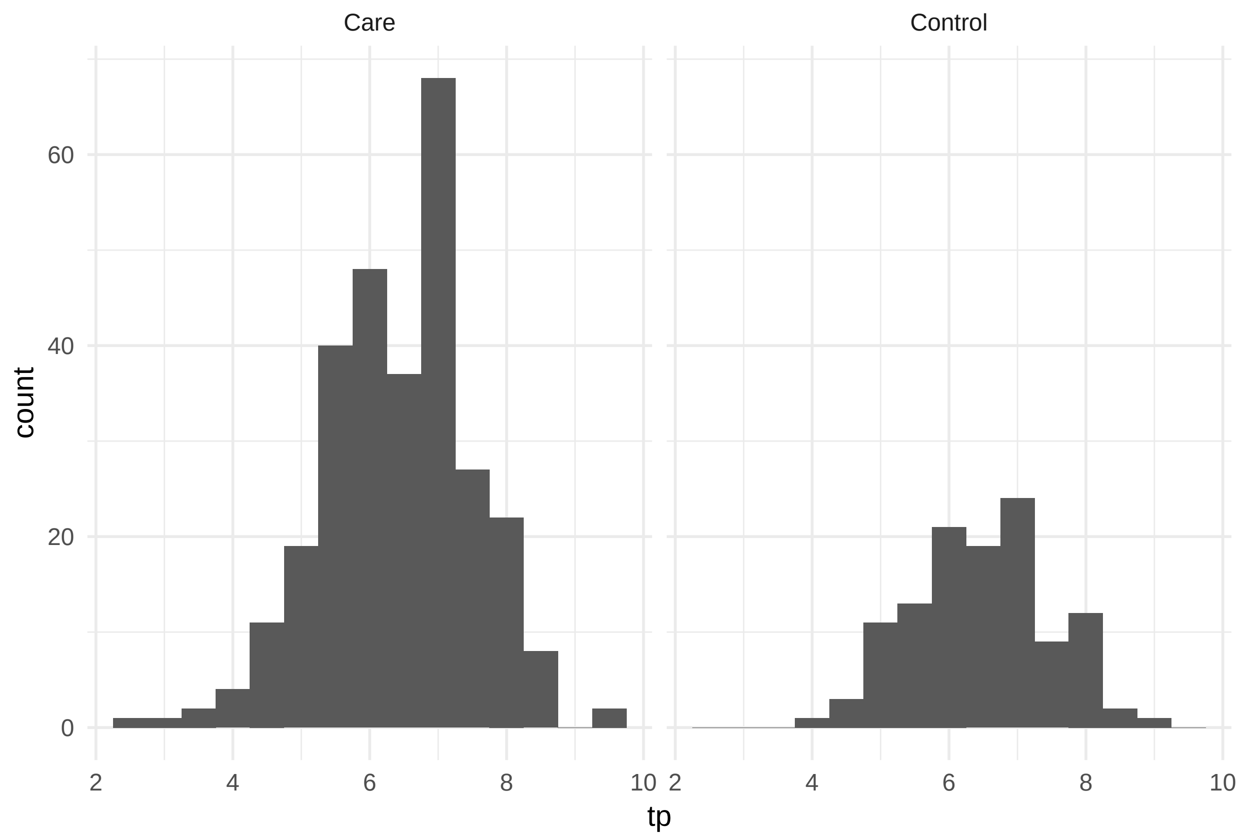
The next columns relate to the levels of Albumin (a group of water-soluble globular proteins) and the total amount of protein detected in the sample.

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



The albumin levels appear mostly similar.

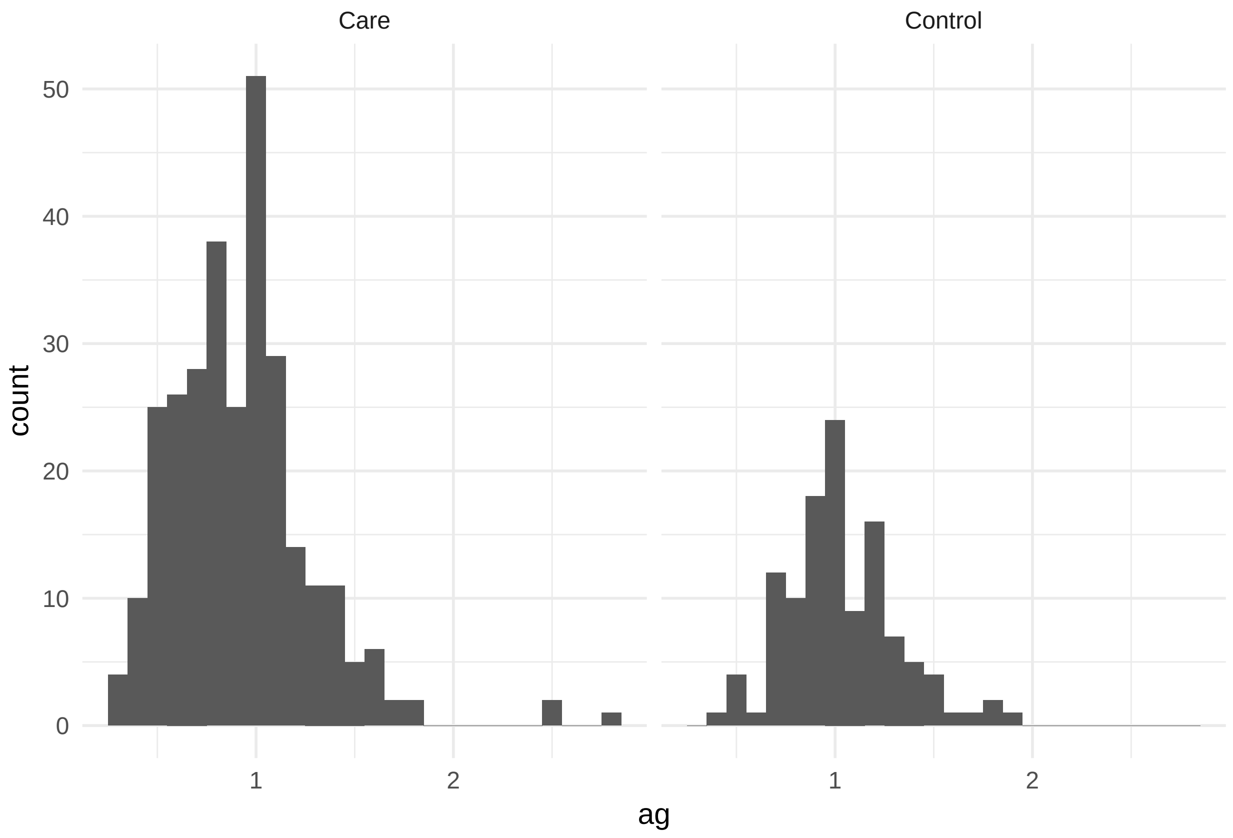
```r
train %>%
  ggplot(aes(x = tp)) +
    geom_histogram(binwidth = 0.5) +
    theme_minimal() +
    facet_grid(~ outcome)
```



The total protein amount also seems to be similar between the two groups.

The final column is ratio between albumin and globulin.

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



Here there are some clear outliers in the group of patients with liver disease, so again, the spread of the values appears to play a key part.

Data Preparation

Correlated Predictors

It is evident from the descriptions of the data that some predictors are most likely correlated; for example, total protein and each of the individual proteins that were meaured.

It is possible to quickly find the correlations as follows. Note that the categorical columns are excluded.

```r
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
```

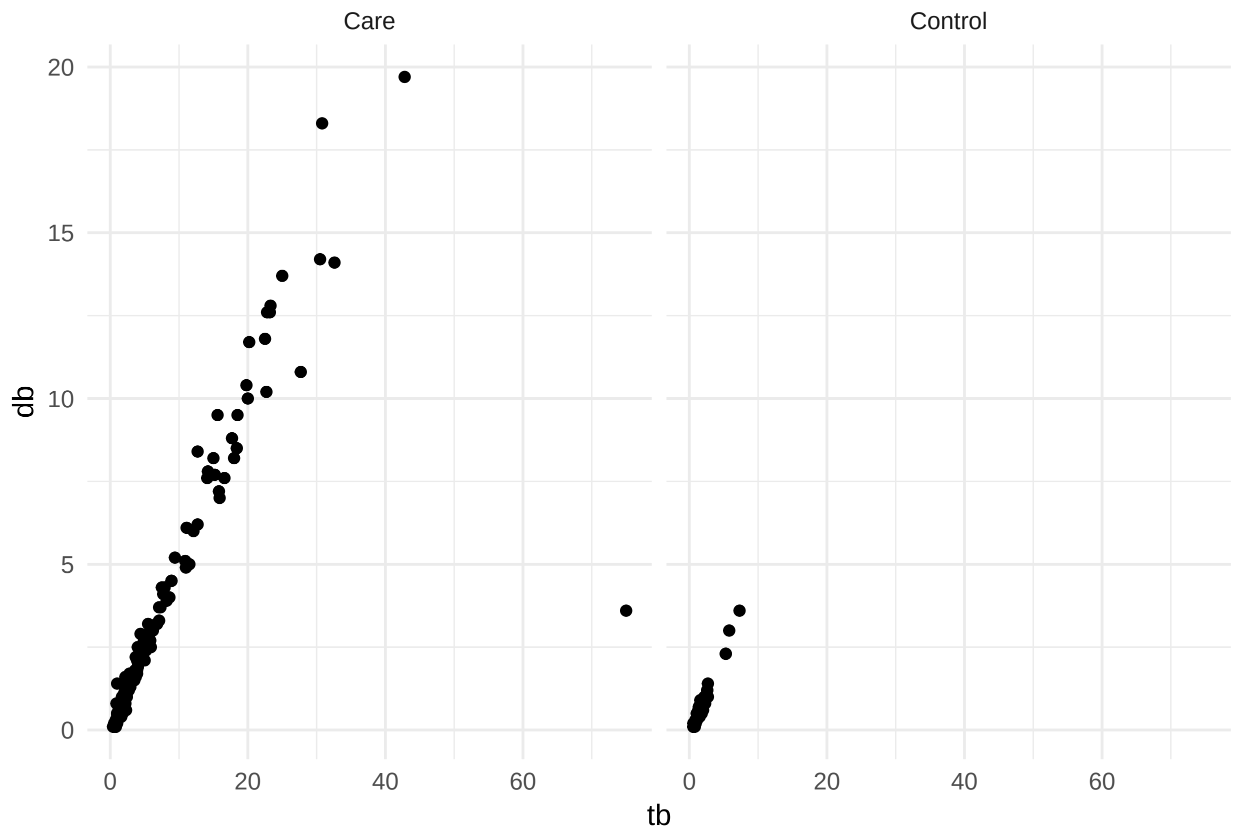
Using 0.7 as a cutoff to identify highly correlated predictors, those that warrant further investigation are;

- Direct bilirubin (`db`) and total bilirubin (`tb`)
- Aspartate Aminotransferase (`sgot`) and Alamine Aminotransferase (`sgpt`)
- 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.

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



Both groups have the same correlation, with outliers in a similar range.

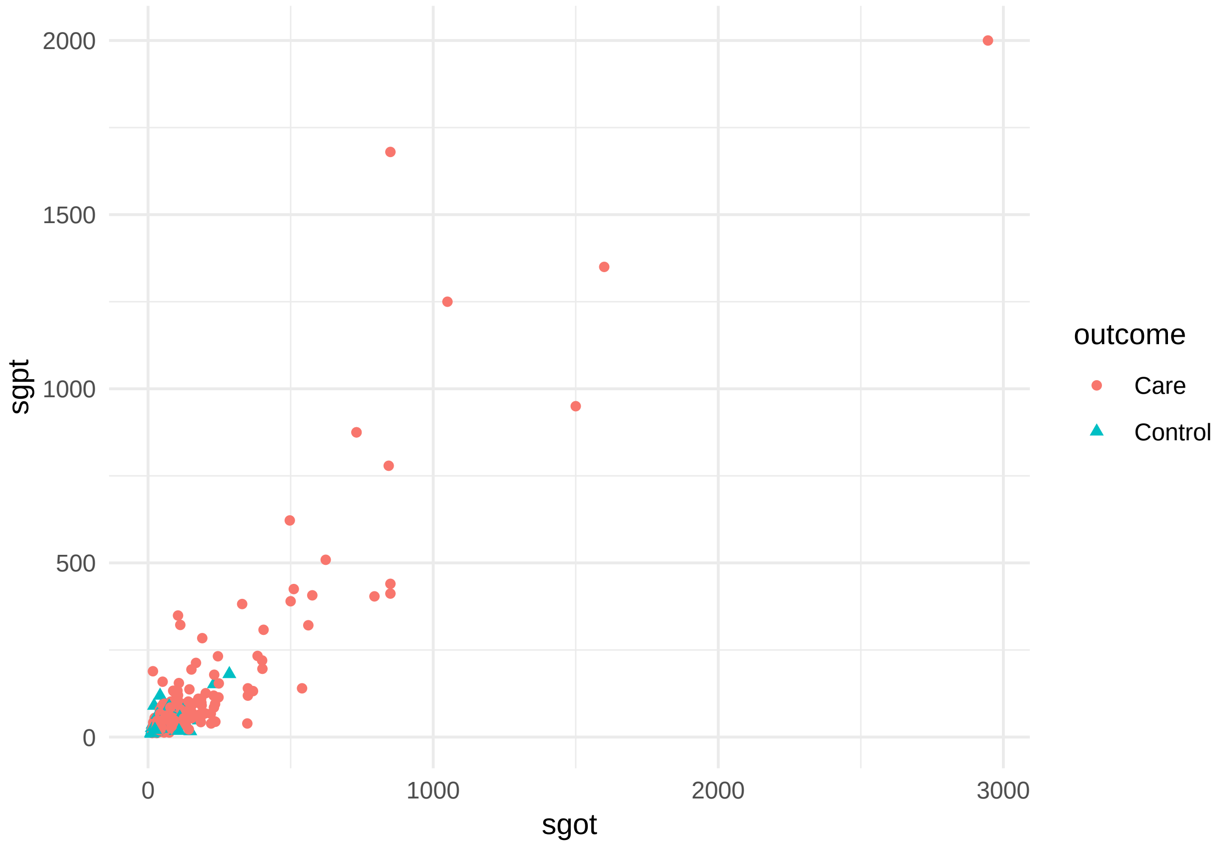
For this reason, it may make sense to drop one of these from the list of predictors.

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

Aminotransferases

The same approach can be used for the `sgot` and `sgpt` columns.

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



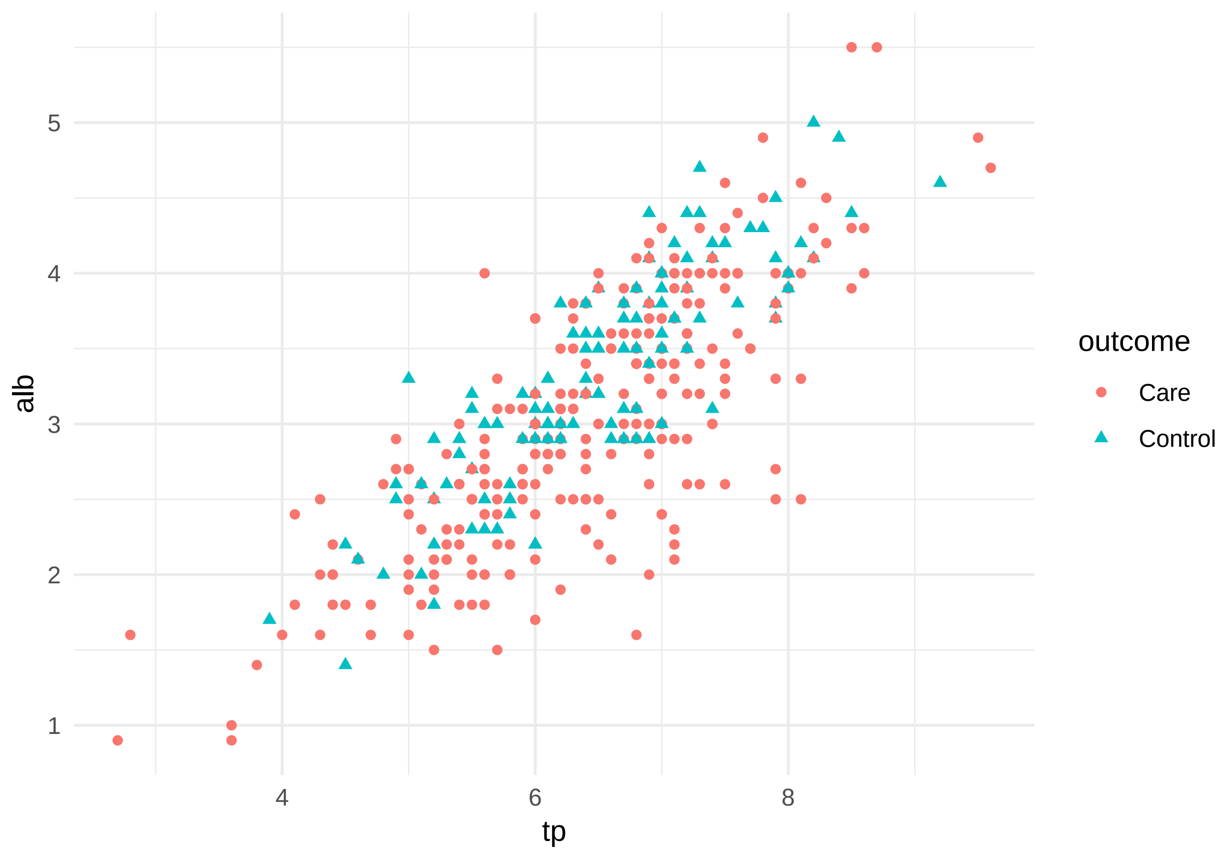
Again, with the same remaining fairly consistent, it makes sense to eliminate one of the columns. As it displays more variance, it makes sense to keep `sgot`.

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

Proteins

Finally, since total proteins (`tp`) includes Albumin (`alb`), these can be explored.

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



Again, these are highly correlated, with no visible significance to any outliers, so the total proteins value can be retained.

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

Machine Learning Methods

The data show some signs of correlations which might permit machine learning models to be developed. The stated aim of this project is to identify patients who may benefit from assessment for liver care, the natural metric to explore might be accuracy (the proportio of patients grouped correctly).

Other metrics to consider are the sensitivity (the proportion of patients who were in the disease group correctly identified) and specificity (the proportion of actual negatives, which are healthy patients, identified as such).

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

One of the most common classification models is the Naive Bayes model. This calculates a series of conditional probabilities, and the probabilities of each possible outcome.

```r
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      82      16
##    Control   42      33
##
##                Accuracy : 0.6647
##                  95% CI : (0.5891, 0.7346)
##     No Information Rate : 0.7168
##     P-Value [Acc > NIR] : 0.943649
##
##                   Kappa : 0.2885
##  Mcnemar's Test P-Value : 0.001028
##
##             Sensitivity : 0.6613
##             Specificity : 0.6735
##          Pos Pred Value : 0.8367
##          Neg Pred Value : 0.4400
##              Prevalence : 0.7168
##          Detection Rate : 0.4740
##    Detection Prevalence : 0.5665
##       Balanced Accuracy : 0.6674
##
##        'Positive' Class : Care
##
```

The accuracy of 0.6 seems low, but it's important to note that Naive Bayes assumes that the feeatures are independent, when they are clearly not.

Linear Classifier

Another popular option is a linear classifier. In this case, the "Boosted Generalized Linear Model".

```r
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'])
```

```r
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
##
```

Here the accuracy is clearly improved, but the confusion matrix shows a critical point - it predicted that nearly every patient needed care, despite 33 of these being incorrect.

Here, the sensitivity (the number of patients correctly identified) was close to 1 (which would otherwise be excellent), but the sensitivity (those without the disease categorised as such) was close to zero.

While the precise priorities are for healthcare systems (hospitals, primary care doctors, commissioners and insurers) to decide, it makes sense to avoid referring patients who are otherwise healthy.

The other problem is that in these data, the group with liver disease are larger than the control group. This gives the unusually large accuracy metric.

Glovally, liver disease is thought to occur in 4-8% of the population at some time in their lives. This means that with data representing a more typical population (rather than "liver patients"), the accuracy of this approach would be less than 0.1.

For that reason, it is now useful to repeat the previous two confusion matrices, this time calculating the values including the prevalence (which we can estimate as 6%).

```r
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       82      16
##    Control    42      33
##
##                 Accuracy : 0.6647
##                   95% CI : (0.5891, 0.7346)
##      No Information Rate : 0.7168
##      P-Value [Acc > NIR] : 0.943649
##
##                    Kappa : 0.2885
##   Mcnemar's Test P-Value : 0.001028
##
##              Sensitivity : 0.6613
##              Specificity : 0.6735
##           Pos Pred Value : 0.1145
##           Neg Pred Value : 0.9689
##               Prevalence : 0.0600
##           Detection Rate : 0.4740
##     Detection Prevalence : 0.5665
##        Balanced Accuracy : 0.6674
##
##         '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
```

```
##
```

If this is to be used as a medical screening tool, the key values here are the Positive Predictive Value and Negative Predictive Value.

In the case of the linear classifier, only 6% of those who the test flags up actually have liver disease, which, while it didn't miss any at-risk patients, seems unacceptably low, and means that 94% of patients who were referred for further investigation had no underlying liver disease.

Comparatively in the Naive Bayes model, while only 10% of the patients flagged needed further care, 96% of those who were identified as at less risk were not at risk.

Logistic Regression

Another very common machine learning approach is logistic regression. This time, using the Bayesian Generalized Linear Model, and including the actual prevalence in the confusion matrix.

```r
lr_model = train(outcome ~ ., data = train, method = "bayesglm")
predictions = predict(lr_model, newdata = test)
confusionMatrix <- confusionMatrix(predictions, test$outcome, prevalence = 0.06)
results[nrow(results) + 1, ] <- c(as.character('Logistic Regression (bayesglm)'),
                                  confusionMatrix$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
##
```

This shows a further improvement. The PPV of 0.07 is lower, but the NPV of 0.98 represents a far larger group of patients.

K-Nearest Neighbours

The next method to consider is the "Nearest Neighbours", here the K-nearest neighbours. This will find the k closest matching points from the training data. These will have their results averaged to find the predicted outcome.

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.

The value of K is a tuning parameter, and the following code will attempt to find the most appropriate value.

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

The value of 9 is settled on here. It's worth noting that in a more widespread deployment, where more pleniful training data are available, this could be different.

```r
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
##
```

Despite a more realistic range of predictions, there is no improvement to the accuracies, sensitivity or specificity.

Random Forest

The final tested approach is a random forest. This will create a "forest" of randomly chosen decision trees, which result in a classification.

The results of these trees will then be compared, and the best performing tree selected.

```r
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)
```

Results

The results of the comparison can be seen in the following table.

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

Model	Accuracy	Sensitivity	Specificity
Naive Bayes (nb)	0.664739884393064	0.661290322580645	0.673469387755102
K-nearest neighbours (knn)	0.676300578034682	0.82258064516129	0.306122448979592
Random Forest (rf)	0.716763005780347	0.895161290322581	0.26530612244898
Linear Classifier (glmboost)	0.722543352601156	1	0.0204081632653061
Logistic Regression (bayesglm)	0.728323699421965	0.935483870967742	0.204081632653061

At this point, it is worth considering the pros and cons of each combination of results.

Models that are highly sensitive, but less specific (the linear classifier, logistic regression and k-nearest neighbours) would mean patients are very likely to be flagged if they would benefit from specialist care, however many patients who would not benefit will also be flagged.

Models with high speficicity but low sensitivity (none of these models behaved this way) would be good for identifying patients who are very likely to need care. A 'hit' on a tool like this would indicate that the patient is almost certainly in the group requiring further care, but it is also likely to miss a large number of patients who would benefit from care.

Models (like Naive Bayes), which perform well in both sensitivity and specificty provide a compromise. Not all patients who would benefit from care will be spotted, but there will be fewer patients missed who may otherwise receive care.

Conclusion

After careful cnsideration, no model has a perfect combination that makes it clearly superior to the others.

Studying the documentation indicates that KNN would most likely improve in performance a little with a larger training data set.

Since the idea of this project was to aid analysis of blood results, but not replace the role of the physician in that process, a combination of high sensitivity and low specificity might be useful. It will lead to many patients being highlighted unnecessarily, but will allow a doctor to quickly note that the blood results (and wider patient presentation) need attention with regard to their liver.

Despite this, there is a risk that, especially in public-sector healthcare systems, this could be a huge increase in workload for clinicians, who would need to review significant numbers of patients who have no illness, no symptoms and no significant benefit from specialist care.

There is a risk of 'alert fatigue', where doctors get so used to seeing a warning marker that a patient may need specialist care that they begin ignoring the warnings altogether.

For this reason, the Naive Bayes approach, using the `nb` method of the `caret` package, appears to offer the best combination of accuracy, sensitivity and specificity.

This approach has been selected for creating the attached script.

References

1. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.