Liver desease prediction

Ana Radić

18.06.2020.

INTRODUCTION

This is the first data science projects on which I want to apply the knowledge base and skills in R data analysis that I have gained throughout the series HarvardX Professional Certificate in Data Science. The dataset I will be working on is about Indian liver pacients and was downloaded from Kaggle (a subsidiary of Google LLC, an online community of data scientists and machine learning practitioners).

Patients with liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. This dataset, called shorter `liver`, was used to evaluate prediction algorithms in an effort to reduce burden on doctors. Based on chemical compounds (bilrubin,albumin,protiens,alkaline phosphatase...) present in human body and tests like SGOT (measures a liver enzyme called serum glutamic-oxaloacetic transaminase.) we are going to determine which patients have liver disease, and which does not.

Liver dataset contains 11 columns and 583 rows which represent 416 liver patient records and 167 non liver patient records. Data was collected from North East of Andhra Pradesh, India, and will be used to train different predicting models. For that purpose will be divided into train and test set. Decision on a winner model we based on the *accuracy*, *sensitivity*, *specificity*, *f_score* and *AUC* obtain from the test set. One column is a class label used to divide groups into liver patient (liver disease) or not (no disease). The key steps in this project are:

- Introduction
- Data Wrangling
- Data Visualization
  - Univariate analysis
    - Patient
    - Chemical compounds present in blood
    - Gender
    - Age
  - Bivariate analysis
- Data Transformation
- Modeling
  - Logistic Regression
  - K-Nearest Neighbour
  - Classification tree
  - Random Forest

- Conclusion

DATA WRANGLING

Let's look at some of the general properties of the data for better understanding.

variable	classe	first_values
Age	integer	65, 62, 62, 58, 72, 46
Gender	integer	Female, Male, Male, Male, Male, Male
Total_Bilirubin	double	0.7, 10.9, 7.3, 1, 3.9, 1.8
Direct_Bilirubin	double	0.1, 5.5, 4.1, 0.4, 2, 0.7
Alkaline_Phosphotase	integer	187, 699, 490, 182, 195, 208
Alamine_Aminotransferase	integer	16, 64, 60, 14, 27, 19
Aspartate_Aminotransferase	integer	18, 100, 68, 20, 59, 14
Total_Protiens	double	6.8, 7.5, 7, 6.8, 7.3, 7.6
Albumin	double	3.3, 3.2, 3.3, 3.4, 2.4, 4.4
Albumin_and_Globulin_Ratio	double	0.9, 0.74, 0.89, 1, 0.4, 1.3
Dataset	integer	1, 1, 1, 1, 1, 1

Each row represents the results of a blood test of certain chemical components for one pearson. Features given in the columns are:

- **Age**

  Age of the patient. Any patient whose age exceeded 89 is listed as being of age "90"

- **Gender**

  Gender of the patient denote as factor with 2 levels: "Female" and "Male"

- **Total_Bilirubin**

  Total serum bilirubin (TSB) measures the amount of a substance called bilirubin. This test is used to find out how well liver is working. It is often given as part of a panel of tests that measure liver function. A small amount of bilirubin in blood is normal, but a high level may be a sign of liver disease.

- **Direct_Bilirubin**

  Conjugated bilirubin (DSB), or direct, bilirubin travels from the liver into the small intestine. A very small amount passes into kidneys and is excreted in urine. This bilirubin also gives urine its distinctive yellow color. Bilirubin that is bound to a certain protein (albumin) in the blood is called unconjugated, or indirect, bilirubin.

- **Alkaline_Phosphotase**

  The alkaline phosphatase (ALP) is an enzyme found throughout body, mainly in liver, bone, kidney, and digestive tract.The ALP test may be done as part of a routine liver panel, a group of blood tests that looks at how well liver is working. If ALP levels are too high, there is need test to find out what type of ALP is elevated in blood.

- **Alamine_Aminotransferase**

  Alanine aminotransferase (ALT) is mostly found in liver cells. When liver cells are injured, they release this enzyme into blood. High levels are a sign of liver damage.

- **Aspartate_Aminotransferase**

  Aspartate transaminase (AST) is an enzyme that is released when liver or muscles are damaged. Although AST is found mainly in liver and heart, AST can also be found in small amounts in other muscles. This test can also be used to monitor liver disease.

- **Total_Protiens**

  The total protein measures the total amount albumin and globulin in body. It's used as part of routine health checkup. It may also be used if you have unexpected weight loss, fatigue, or the symptoms of a kidney or liver disease.

- **Albumin**

  Albumin is the most abundant protein in blood plasma, making up about 40-60%. Albumin is produced in the liver and plays a role in the transport of various substances through the blood plasma. Low albumin levels are the result of a liver or kidney disorder.

- **Albumin_and_Globulin_Ratio**

  The Albumin to Globulin ratio (A:G) is the ratio of albumin present in serum in relation to the amount of globulin. The ratio can be interpreted only in light of the total protein concentration. Healthy people have a little more albumin than globulin, but if you're sick, this won't be the case.

- **Dataset**

  field used to split the data into two sets (patient with liver disease, or no disease).

For practical reasons, we will replace the column names with abbreviations used in the professional literature.

```
colnames(liver) <- c("age", "gender", "TSB", "DSB", "ALP", "ALT",
                     "AST", "TP", "albumin", "A_G_ratio", "patient")
```

Elements of the patient column are twos and ones. Let's see what number denotes the liver patient. Knowing that the high value of some features in the table indicates the disease, we will check which label have those patients who have a high level of DSB, ALT and AST.

```
liver %>% group_by(patient) %>% summarise(TSB = round(mean(TSB),2),
                                          ALT = round(mean(ALT),2),
                                          AST = round(mean(AST),2)) %>% kable()
```

patient	TSB	ALT	AST
1	4.16	99.61	137.70
2	1.14	33.65	40.69

We conclude that 1 indicates a liver patient and 2 healthy persons. Let's denote a healthy person by "not sick" and a sick person by "sick" for visualization purpose, and convert that column into factor class.

```
liver$patient <- as.character(liver$patient)
liver$patient[liver$patient == 2] <- "not sick"
liver$patient[liver$patient == 1] <- "sick"
liver$patient <- as.factor(liver$patient)
```

Now we can see the first few lines of `liver` data set:

age	gender	TSB	DSB	ALP	ALT	AST	TP	albumin	A_G_ratio	patient
65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	sick
62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	sick
62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	sick
58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	sick
72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	sick
46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	sick

Through briefly summary of features, we discover what the feature distribution is like, is there any missing value, about the quantile...

age	gender	TSB	DSB	ALP	ALT	AST	TP	albumin	A_G_ratio
Min. : 4.00	Female:142	Min. : 0.400	Min. : 0.100	Min. : 63.0	Min. : 10.00	Min. : 10.0	Min. :2.700	Min. :0.900	Min. :0.3000
1st Qu.:33.00	Male :441	1st Qu.: 0.800	1st Qu.: 0.200	1st Qu.: 175.5	1st Qu.: 23.00	1st Qu.: 25.0	1st Qu.:5.800	1st Qu.:2.600	1st Qu.:0.7000
Median :45.00	NA	Median : 1.000	Median : 0.300	Median : 208.0	Median : 35.00	Median : 42.0	Median :6.600	Median :3.100	Median :0.9300
Mean :44.75	NA	Mean : 3.299	Mean : 1.486	Mean : 290.6	Mean : 80.71	Mean : 109.9	Mean :6.483	Mean :3.142	Mean :0.9471
3rd Qu.:58.00	NA	3rd Qu.: 2.600	3rd Qu.: 1.300	3rd Qu.: 298.0	3rd Qu.: 60.50	3rd Qu.: 87.0	3rd Qu.:7.200	3rd Qu.:3.800	3rd Qu.:1.1000
Max. :90.00	NA	Max. :75.000	Max. :19.700	Max. :2110.0	Max. :2000.00	Max. :4929.0	Max. :9.600	Max. :5.500	Max. :2.8000
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA's :4

From the previous table we see that one column has missing values.

```
liver %>% summarise_all(~ sum(is.na(.))) %>% select_if(.!=0)
```

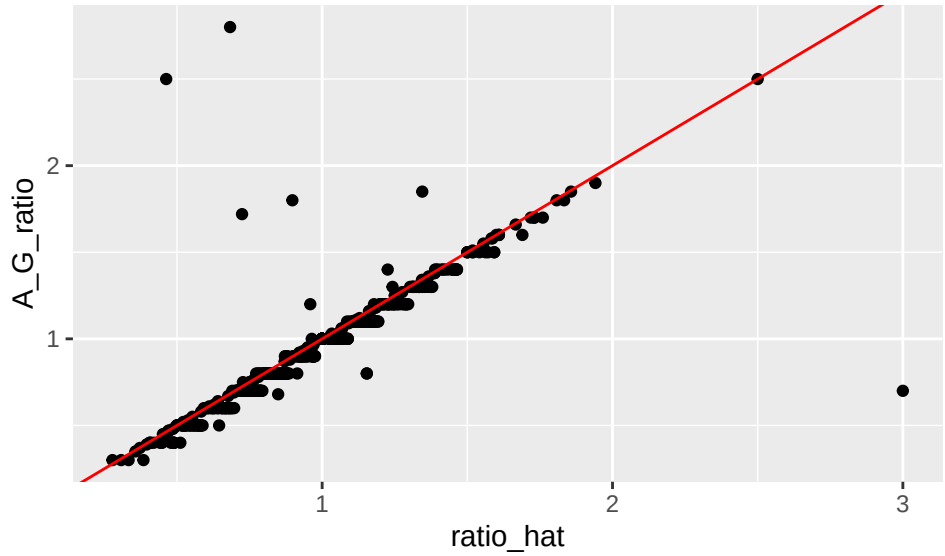
```
##   A_G_ratio
## 1         4
```

In the professional literature we can find formula $A\_G\_ratio = \frac{albumin}{globulin}$, where globulin is another protein (about 40% of total protein). Most of the total proteins are albumin and globulin so we can look at TP as approximately equal to albumin + globulin. Then A_G_ratio is:

$$A\_G\_ratio = \frac{albumin}{TP - albumin}$$

Let's see if our data supports this.

```
liver %>% mutate(ratio_hat = albumin/(TP-albumin)) %>% ggplot(aes(ratio_hat, A_G_ratio)) +
  geom_point(na.rm=TRUE)+
  geom_abline(col="red")
```



We conclude that the missing values can be filled with the previous formula.

```
liver <- liver %>%
  mutate(A_G_ratio = ifelse(is.na(A_G_ratio), round(albumin/(TP-albumin),digits= 2), A_G_ratio))
```

Now we no longer have the missing values and the data is ready for visualization.

DATA VISUALIZATION

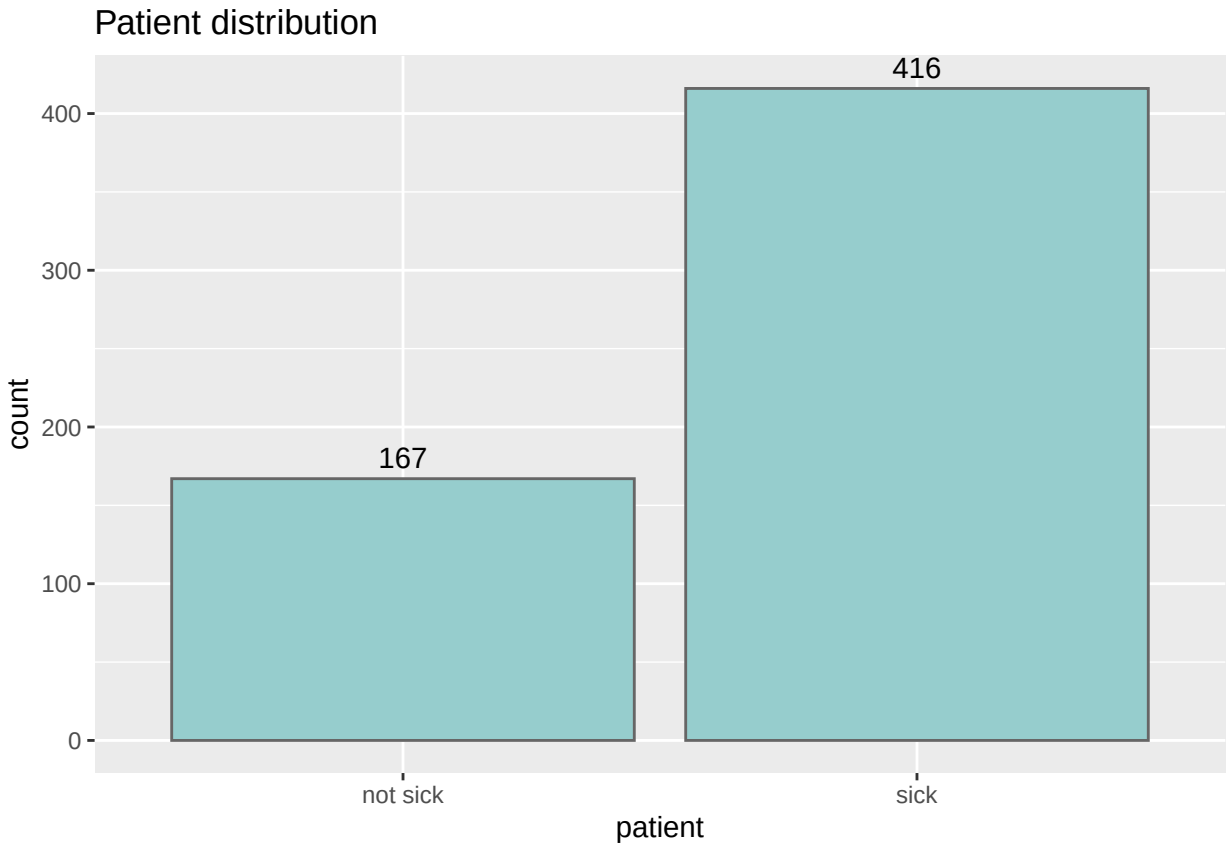
With data visualization tools we are goin to see and understand trends, outliers and patterns in data. First we will explore distribution of every colomn.

Univariate analysis

Patient

We start with the target column ´patient´.

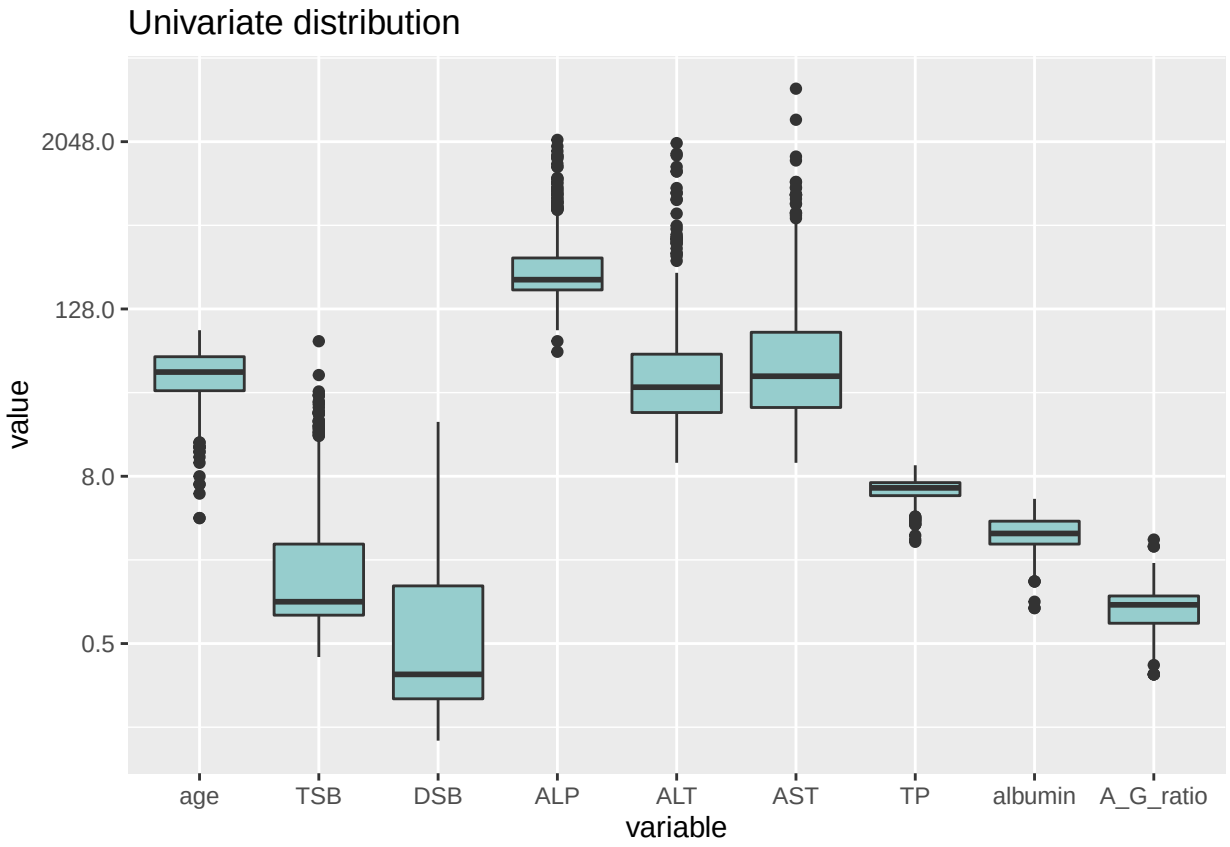
```
liver %>% ggplot(aes(patient)) +
          geom_bar( fill = "paleturquoise3", color = "gray40") +
          geom_text(aes(label=..count..), stat="count", vjust=-0.5) +
          ggtitle("Patient distribution")
```



In the bar plot above we see that our data is highly unbalanced. There are many more sick person than healthy.

The variables have a very different scale of values. Some variables have a large range of values, so the graph with the boxplot is not clearly unless we scale the y axis with, e.g. function `log2`. This gives a clearer picture of the diversity of variable distributions.

```
liver %>% select(-gender, -patient) %>% reshape2::melt() %>%
       ggplot(aes(x = variable, y = value)) +
       geom_boxplot(na.rm = TRUE, fill = "paleturquoise3") +
       scale_y_continuous(trans = "log2") +
       ggtitle("Univariate distribution")
```

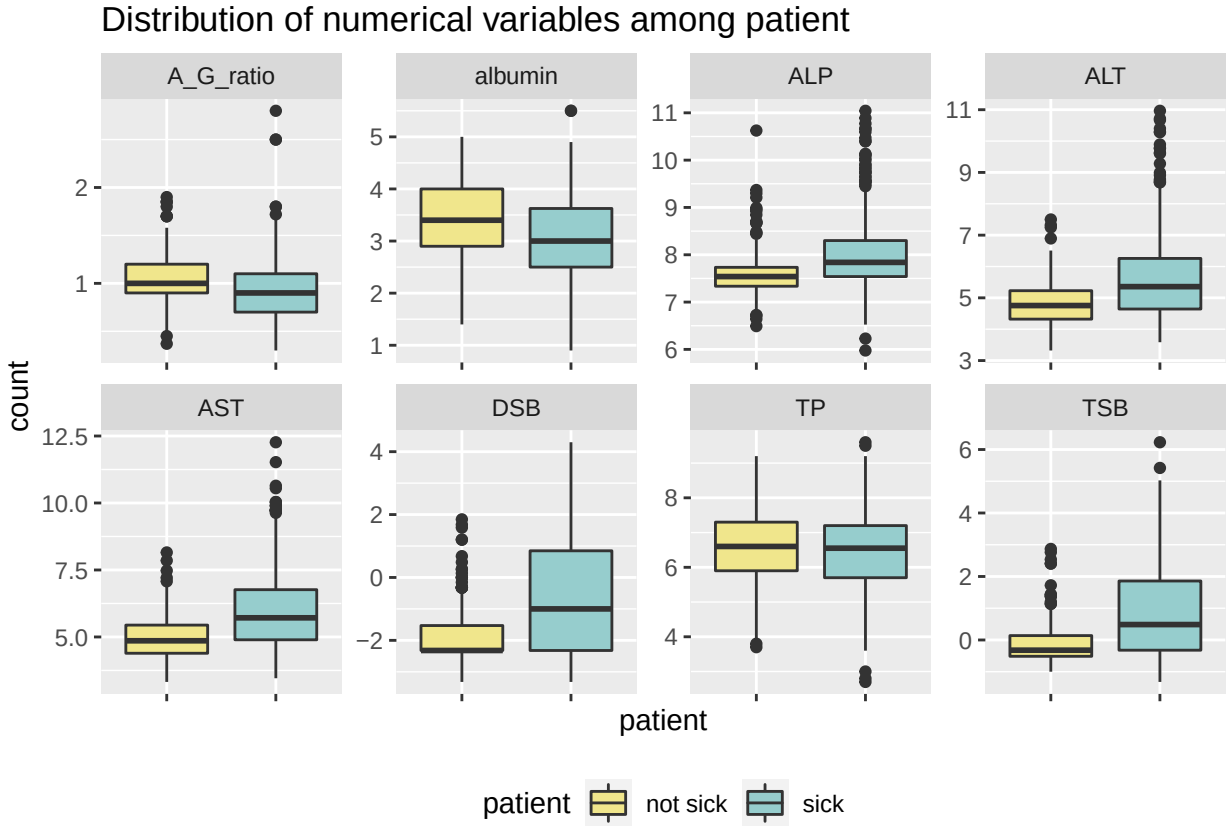


ALP, ALT and AST variables have a very wide range, so we will keep this in mind during modeling. Also, these variables stand out by a large number of outliers, although almost every variable has them.

Chemical compounds present in blood

We are interested in how each of the columns affects the target patient column. For this purpose, we need a graph of the distribution of each column (with numerical results of blood analysis) among patient column.

```
liver %>% mutate(ALP=log2(ALP), ALT=log2(ALT), AST=log2(AST), DSB=log2(DSB), TSB=log2(TSB)) %>%
        gather(liver_column, count, -patient, -age, -gender) %>%
        ggplot(aes(patient, count, fill = patient)) +
        geom_boxplot() +
        facet_wrap(~liver_column, scales = "free", ncol = 4) +
        scale_fill_manual(values=c("khaki","paleturquoise3")) +
        theme(axis.text.x = element_blank(), legend.position="bottom") +
        ggtitle("Distribution of numerical variables among patient")
```

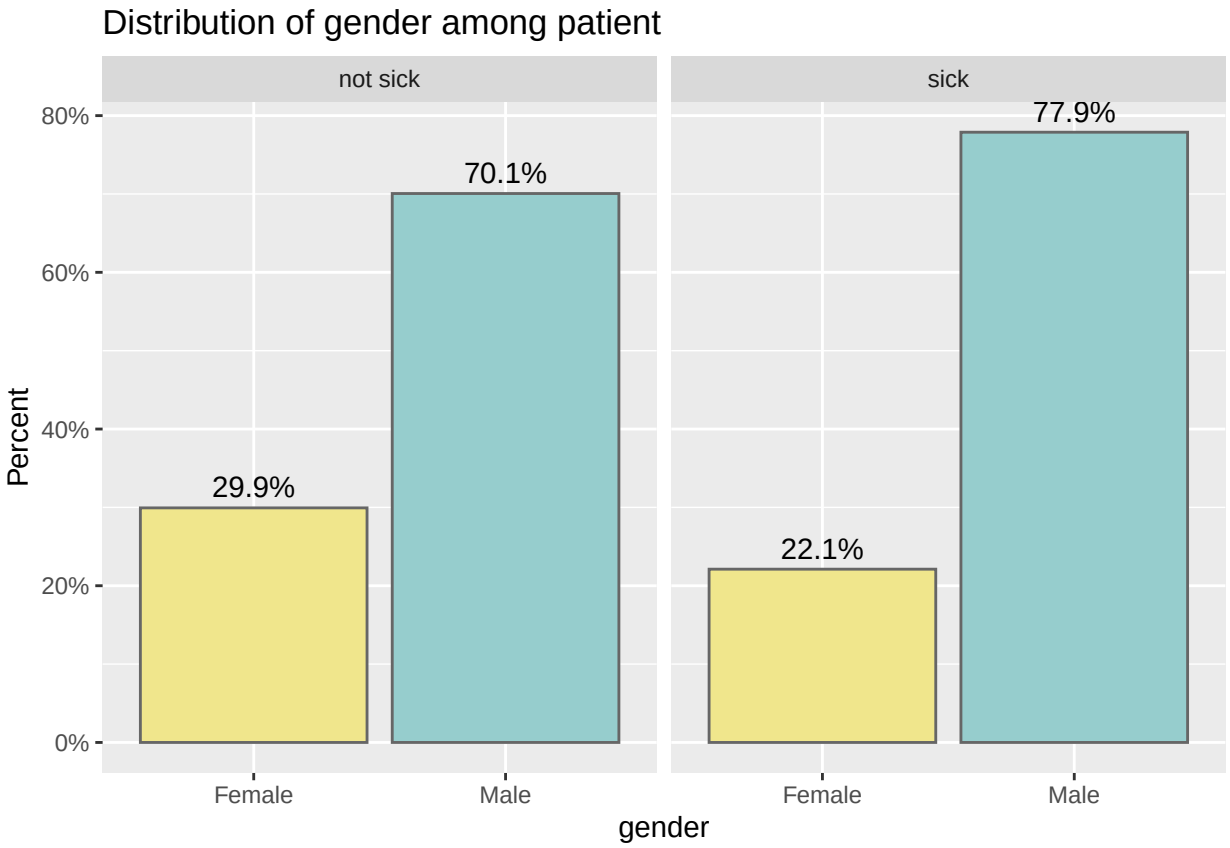


ALP, ALT and AST variables the best separates the sick from healthy, while TP variable has very small predictive power.

Gender

Gender is the only categorical variable among predictors. There are more mens in our observations, 24.4% of women and 75.6% of men. What is the gender distribution among patient and whether gender affects the disease are the following questions whose answers are offered in the following graph.

```
liver %>% ggplot(aes(gender, group = patient)) +
  geom_bar(aes(y = ..prop.., fill= factor(..x..)), stat = "count", color= "gray40") +
  geom_text(aes(label = percent(..prop..), y = ..prop..), stat = "count", vjust = -0.5) +
  labs(y= "Percent") +
  facet_grid(~patient) +
  scale_fill_manual(values=c("khaki","paleturquoise3")) +
  theme(legend.position = "none") +
  scale_y_continuous(labels = percent) +
  ggtitle("Distribution of gender among patient")
```

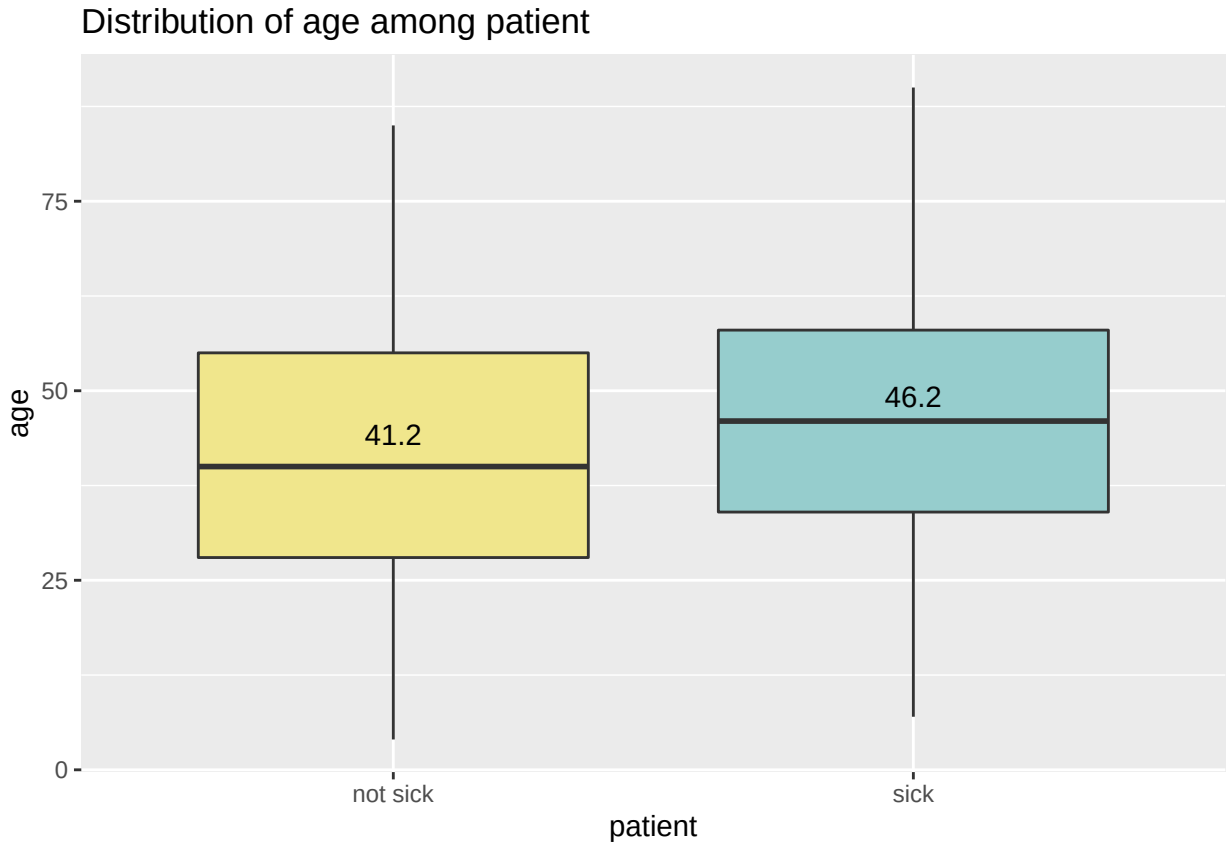


Gender has little effect on the disease, the graphs show that men get liver disease slightly more often than women.

Age

Age is only numerical column which is not relatet with blood test results.

```
means_age <- liver %>% group_by(patient) %>%
     summarise(age=round(mean(age),1))

   liver %>% ggplot(aes(patient, age, fill=patient)) +
     geom_boxplot() +
     theme(legend.position = "none") +
     scale_fill_manual(values=c("khaki","paleturquoise3")) +
     geom_text(data = means_age, aes(label = age, y = age +3)) +
     ggtitle("Distribution of age among patient")
```

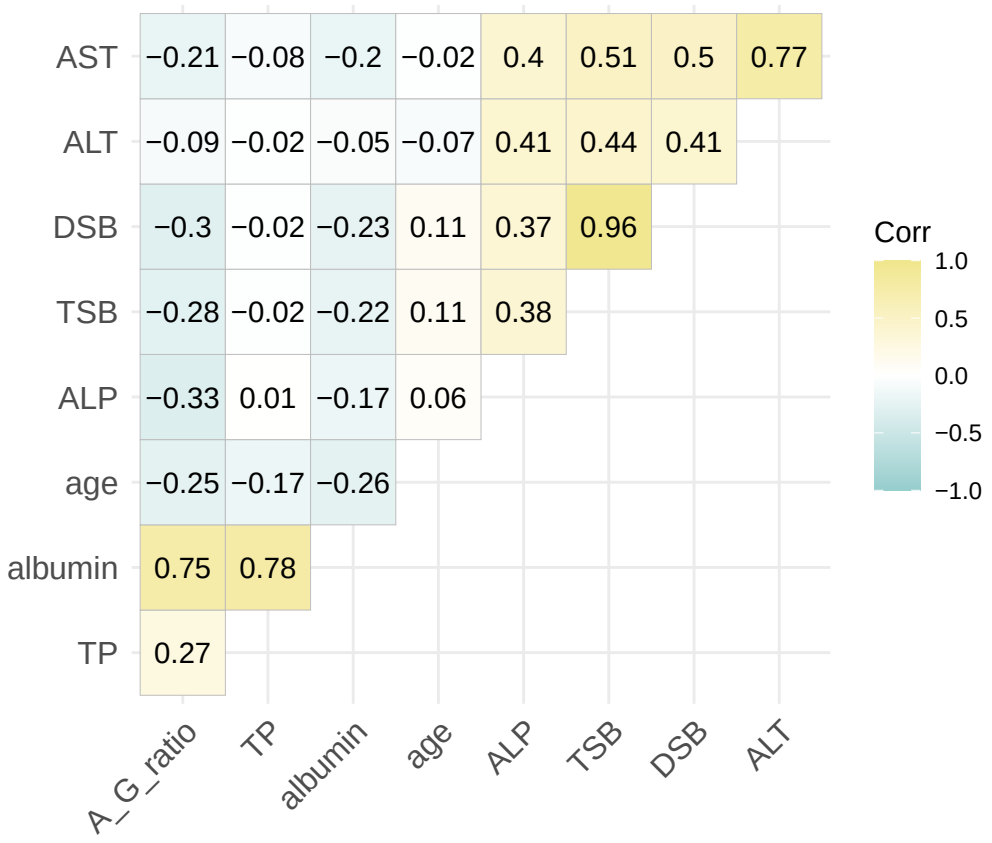


Sick people are on average older than healthy.

Bivariate analysis

Correlation test is used to evaluate the association between two or more variables.
Spearman correlation methods is non-parametric rank-based correlation test, that evaluates the monotonic relationship between two continuous or ordinal variables. In a monotonic relationship, the variables tend to change together, but not necessarily at a constant rate. Spearman correlation method is used due to outliers, instead of Pearson's, which is the default.

```
round(liver %>% select(-patient, -gender) %>% cor(method = "spearman"),2) %>%
   ggcorrplot(hc.order = TRUE,
              type = "upper",
              colors = c("paleturquoise3", "white", "khaki"),
              lab = TRUE)
```

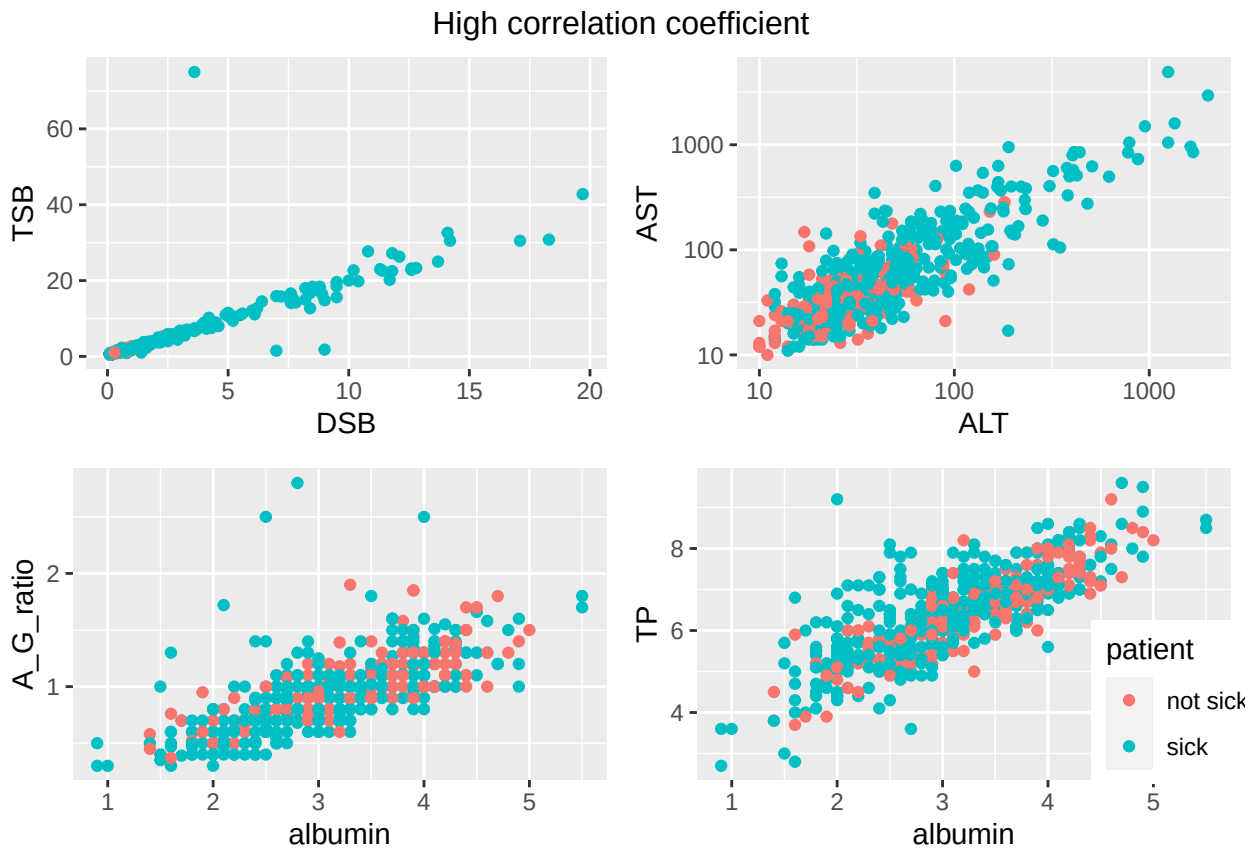


The table shows that there is a strong positive relationship between:

- DSB and TSB (correlation coef. 0.96)
- albumin and TP (correlation coef. 0.78)
- albumin and A_G_ratio (correlation coef. 0.75)
- AST and ALT (correlation coef. 0.77).

Confirmation of this correlation can be seen in the following scatterplots graphs.

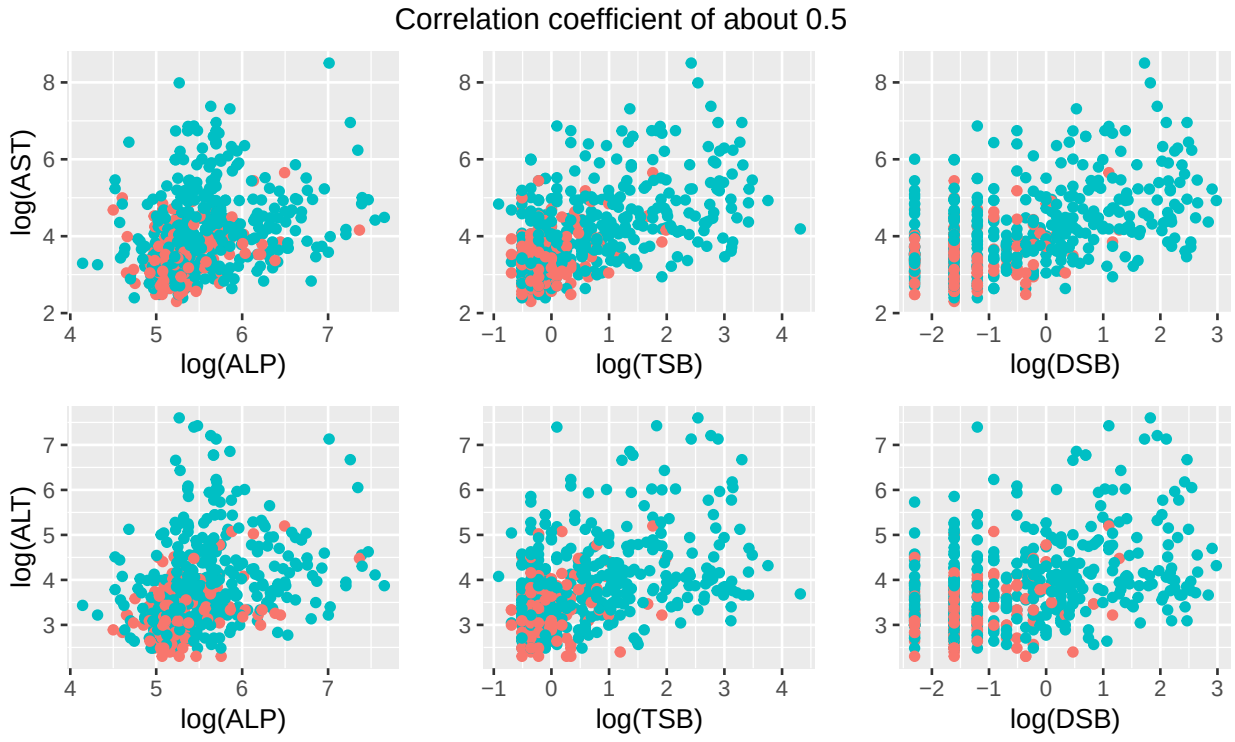
```
    q1 <- liver %>% ggplot(aes(DSB,TSB, col = patient)) +
              geom_point()+
              theme(legend.position ="none")

    q2 <- liver %>% ggplot(aes(ALT,AST, col=patient)) +
              geom_point() +
              scale_x_continuous(trans = "log10") +
              scale_y_continuous(trans = "log10") +
              theme(legend.position = "none")

    q3 <- liver %>% ggplot(aes(albumin,A_G_ratio, col=patient)) +
              geom_point() +
              theme(legend.position = "none")

    q4 <- liver %>% ggplot(aes(albumin,TP, col=patient)) +
              geom_point() +
              theme(legend.position = c(0.9,0.25))

  grid.arrange(q1,q2,q3,q4,ncol=2, nrow=2, top = "High correlation coefficient")
```



All graphs show a positive correlation between the variables, although the strongest linear relationship is between TSB and DSB, as shown by the correlation coefficient.

The correlation table shows that some pairs of variables have a correlation coefficient of about 0.5. The following 6 graphs show what their joint distribution looks like.

```
g1 <- liver %>% ggplot(aes(log(ALP),log(AST), col = patient)) +
     geom_point() + theme(legend.position = "none")

g2 <- liver %>% ggplot(aes(log(TSB),log(AST), col = patient)) +
    geom_point() +
   theme(legend.position = "none") + ylab("")

g3 <- liver %>% ggplot(aes(log(DSB),log(AST), col = patient)) +
    geom_point() +
    theme(legend.position = "none") + ylab("")

p1 <- liver %>% ggplot(aes(log(ALP),log(ALT), col = patient)) +
    geom_point()+ theme(legend.position = "none")

p2 <- liver %>% ggplot(aes(log(TSB), log(ALT),  col = patient)) +
    geom_point()+
    theme(legend.position = "none") + ylab("")

p3 <- liver %>% ggplot(aes(log(DSB), log(ALT), col = patient)) +
    geom_point()+
    theme(legend.position = "none") + ylab("")

grid.arrange(g1, g2, g3, p1, p2, p3, ncol=3, nrow=2,
             top = "Correlation coefficient of about 0.5")
```



It can be concluded that there is no pattern by which the data appear.

DATA TRANSFORMATION

For the purpose of modeling, we change the levels in the variable patient to 0 and 1, and transform gender variable into numeric.

```
liver$patient <- plyr::revalue(liver$patient, c("sick"= 1, "not sick" = 0))
liver$gender <- ifelse(liver$gender == "Male", 1, 0) %>% as.factor()
```

Data appear at different scales and some variables have a very wide range of values.
We will transform the liver data set into `liver_norm` using normalization for better performance of some models. The variables AST, ALT and ALP stand out as those with the highest value range, so our second option of data standardization will be to apply the log function to these three columns and we named new set as `liver_log`.

```
liver_norm <- liver %>% select(-patient, -gender) %>%
                sapply(function(x) {(x-min(x))/(max(x)-min(x))}) %>%
                data.frame(patient = liver$patient,
                           gender = liver$gender)


liver_log <- liver %>% mutate(ALT=log(ALT),
                              ALP= log(ALP),
                              AST=log(AST))
```

MODELING

The problem of predicting outcome of the patient column is a binary classification problem. Several prediction models will be used: **logistic regression (glm), k-nearest neighbors (knn), decision tree (tree)** and **random forest (rf)**. The following metrics will be used to compare model performance: *accuracy*, *sensitivity*, *specificity*, *F1 score* and *AUC*.

The data are unbalanced so it isn't enough to observe accuracy alone. Since there are 71.4% of sick people in data set, if we predict that everyone is sick, we have achieved accurasy of 71.4%. That is why we are interested in both sensitivity(recall) and specificity. F1 score is the weighted average of Precision and Recall. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. AUC(Area under the ROC Curve) represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at distinguishing between patients with disease and no disease.

We will define two functions whose outcomes are metrics of interest. Function `metrics` for input has predicted values, true values, auc (that is output of AUC function) and name of model. AUC function has input object of class train.formula and data set to which we apply the formula.

```
AUC <- function(model, test) {
   probs  <- predict(model, test, type = "prob")[,2]
   p <- prediction(probs,test$patient)
   per <- performance(p,"tpr", "fpr")
   perf_auc <- performance(p,"auc")
   perf_auc@y.values[[1]]}

metrics <- function(y_hat, y, auc, model){
    tibble(model = model,
          accuracy = confusionMatrix(y_hat, y)$overall["Accuracy"],
          sensitivity = confusionMatrix(y_hat, y)$byClass["Sensitivity"],
          specificity = confusionMatrix(y_hat, y)$byClass["Specificity"],
          f_score = confusionMatrix(y_hat, y)$byClass["F1"],
          AUC = auc)}
```

Data partition into train(80%) and test set(20%) allows us to check the performance of the model. The function `set.seed` will be used to make the results interpretable. For the purpose of comparing the performance of the model, we will use three data sets, liver, liver_log and liver_norm.

```
set.seed(14, sample.kind = "Rounding")
ind <- createDataPartition(liver$patient, time = 1, p = 0.2, list = FALSE)

train <- liver %>% slice(-ind)
test <- liver %>% slice(ind)

trainl <- liver_log %>% slice(-ind)
testl <- liver_log %>% slice(ind)

trainn <- liver_norm %>% slice(-ind)
testn <- liver_norm %>% slice(ind)
```

Logistic regresion

Logistic regresion (glm) will be perform on `liver_log` data set.

```
fit_glml_all <- train(patient ~ .,
                      method = "glm",
                      data = trainl)
y_hat <- predict(fit_glml_all, testl)
tbl <- metrics(y_hat, testl$patient, AUC(fit_glml_all, testl), "GLM (all predictors)")
tbl %>% kable()
```

model	accuracy	sensitivity	specificity	f_score	AUC
GLM (all predictors)	0.7119	0.2059	0.9167	0.2917	0.7384

This table show us first results. Based on variable importance we will reject some feature and in that way we will try to improve the results.

```
varImp(fit_glml_all)
```

```
## glm variable importance
##
##           Overall
## TP          100.0
## age          95.8
## albumin      81.2
## ALT          79.9
## ALP          75.8
## DSB          75.0
## A_G_ratio    65.1
## AST          44.1
## gender1      19.0
## TSB           0.0
```

According to the previous table, the columns TSB, gender, AST and A_G_ratio had the least importance for the algorithm. Due to that and the correlation coefficients(TSB-DSB, AST-ALT, A_G_ratio-albumin are correlated), we will throw out those features and train the **glm** algorithm again.

```
fit_glm6 <- train(patient ~ age + TP  + ALT + DSB + ALP +albumin,
                  method = "glm",
                  data = trainl)
y_hat <- predict(fit_glm6, testl)
tbl <- bind_rows(tbl,metrics(y_hat, testl$patient, AUC(fit_glm6, testl), "GLM (6 predictors)"))
tbl %>% kable()
```

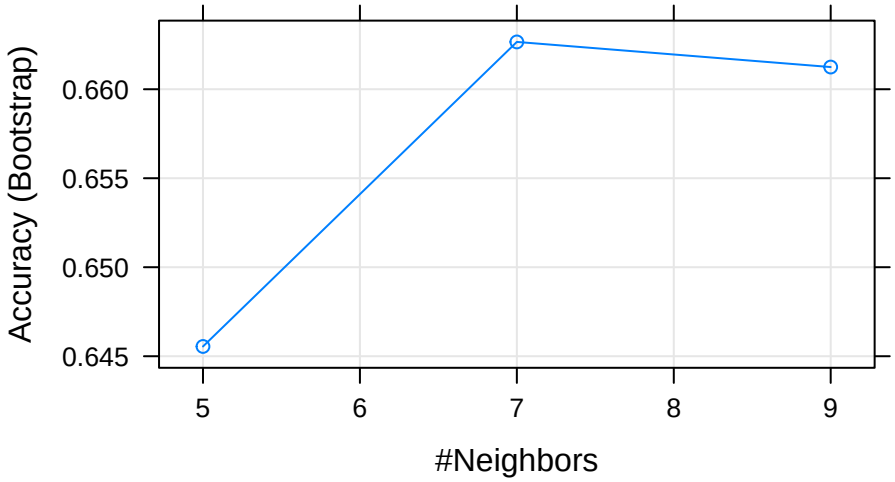
model	accuracy	sensitivity	specificity	f_score	AUC
GLM (all predictors)	0.7119	0.2059	0.9167	0.2917	0.7384
GLM (6 predictors)	0.7542	0.2941	0.9405	0.4082	0.7496

Improvements in model performance are noticeable.

K-Nearest Neighbour

Next, will be perform K-Nearest Neighbour (knn) algoritam on three data set: liver, liver_norm and liver_log and we will compare the results.

```
set.seed(14, sample.kind = "Rounding")
fit_knn <- train(patient ~ .,
                 method = "knn",
                 data = train)

plot(fit_knn)
```



The `train` function applies cross validation(bootstrap version) by default to select the best parameter k between 5, 7 and 10. The best accuracy is obtain for k=7.

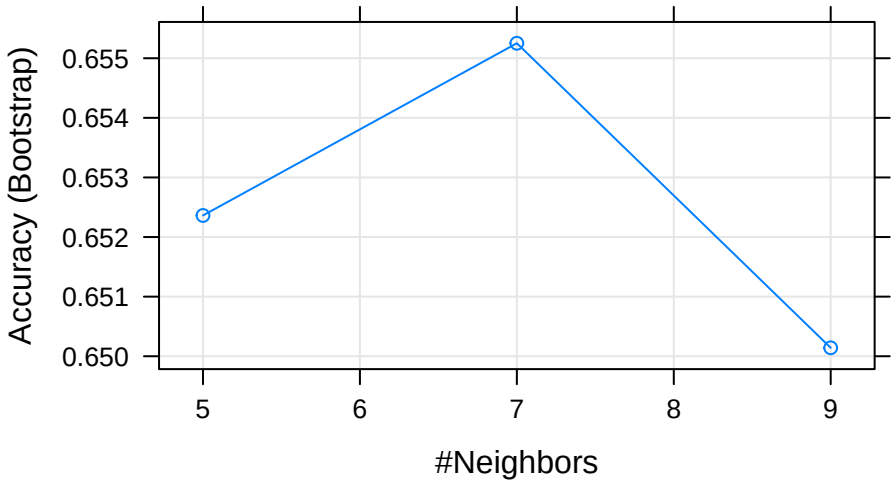
```
y_hat <- predict(fit_knn,test)
tbl <- bind_rows(tbl,metrics(y_hat,
                             test$patient,
                             AUC(fit_knn, test),
                             "KNN on liver"))
tbl %>% kable()
```

model	accuracy	sensitivity	specificity	f_score	AUC
GLM (all predictors)	0.7119	0.2059	0.9167	0.2917	0.7384
GLM (6 predictors)	0.7542	0.2941	0.9405	0.4082	0.7496
KNN on liver	0.7458	0.4118	0.8810	0.4828	0.7518

A better result was achieved compared to glm algoritam.

Then, will be train knn model on `liver_norm` data set:

```
set.seed(14, sample.kind = "Rounding")
fit_norm_knn <- train(patient ~ .,
                      method = "knn",
                      data = trainn)
plot(fit_norm_knn)
```



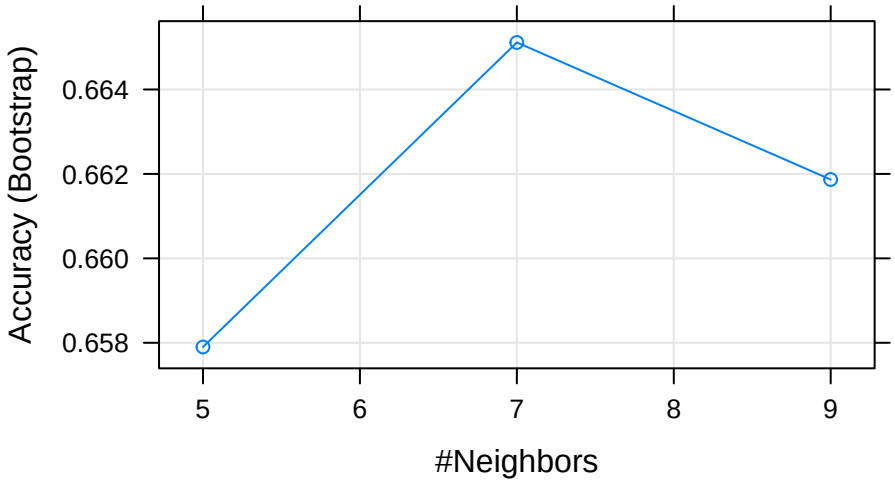

```
y_hat <- predict(fit_norm_knn, testn)
tbl <- bind_rows(tbl,metrics(y_hat,
                             testn$patient,
                             AUC(fit_norm_knn, testn),
                             "KNN on liver(norm)"))
tbl %>% kable()
```

model	accuracy	sensitivity	specificity	f_score	AUC
GLM (all predictors)	0.7119	0.2059	0.9167	0.2917	0.7384
GLM (6 predictors)	0.7542	0.2941	0.9405	0.4082	0.7496
KNN on liver	0.7458	0.4118	0.8810	0.4828	0.7518
KNN on liver(norm)	0.6356	0.2941	0.7738	0.3175	0.6066

The best coefficient k is again 7 but the performance of the model is much worse. The accuracy for this model on trainn set is 0.76 , therefore we can conclude that overfitting has occurred.

Let's see what the result will be when knn is applied to the liver_log table.

```
set.seed(14, sample.kind = "Rounding")
fit_log_knn <- train(patient ~ .,
                     method = "knn",
                     data = trainl)
plot(fit_log_knn)
```




```
y_hat <- predict(fit_log_knn, testl)
tbl <- bind_rows(tbl,metrics(y_hat,
                             testl$patient,
                             AUC(fit_log_knn, testl),
                             "KNN on liver(log)"))
tbl %>% kable()
```

model	accuracy	sensitivity	specificity	f_score	AUC
GLM (all predictors)	0.7119	0.2059	0.9167	0.2917	0.7384
GLM (6 predictors)	0.7542	0.2941	0.9405	0.4082	0.7496
KNN on liver	0.7458	0.4118	0.8810	0.4828	0.7518
KNN on liver(norm)	0.6356	0.2941	0.7738	0.3175	0.6066
KNN on liver(log)	0.7458	0.5882	0.8095	0.5714	0.8009

The best results for now. We have achieved accuracy of 74.58% and AUC of 0.8 (max is 1).

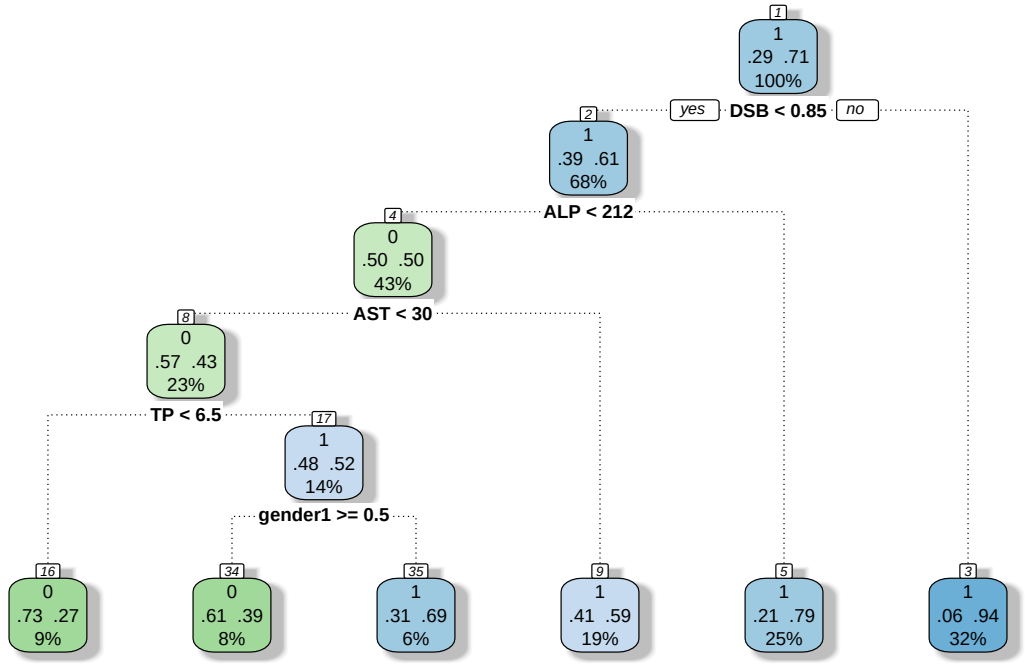
Classification tree

The following results are obtained by applying the classification tree on `liver` data set:

```
set.seed(14, sample.kind = "Rounding")

fit_tree <- train(patient ~ . ,
                  method="rpart",
                  data = train)
y_hat <- predict(fit_tree, test)
tbl <- bind_rows(tbl,metrics(y_hat, test$patient, AUC(fit_tree, test),"TREE on liver"))
tbl %>% kable()
```

model	accuracy	sensitivity	specificity	f_score	AUC
GLM (all predictors)	0.7119	0.2059	0.9167	0.2917	0.7384
GLM (6 predictors)	0.7542	0.2941	0.9405	0.4082	0.7496
KNN on liver	0.7458	0.4118	0.8810	0.4828	0.7518
KNN on liver(norm)	0.6356	0.2941	0.7738	0.3175	0.6066
KNN on liver(log)	0.7458	0.5882	0.8095	0.5714	0.8009
TREE on liver	0.6271	0.1176	0.8333	0.1538	0.5746

```
fancyRpartPlot(fit_tree$finalModel, sub = "")
```



Classification tree perform pretty bad. When we check what is accuracy of predicting outcome on train set (0.77) we see again that overfitting is happend. When trying to solve this problem through tuning parameter cp we do not get satisfactory results. AUC is at the lower limit because recall is 0 and specificity is 1.

```
set.seed(14, sample.kind = "Rounding")
fit_tree <- train(patient ~ . ,
                  method="rpart",
                  data = train,
                  trControl = trainControl(method= "cv", number = 10, p=0.9),
                  tuneGrid=data.frame(cp=seq(0.05,0.1, len=30)))

y_hat <- predict(fit_tree, test)
tbl <- bind_rows(tbl,metrics(y_hat, test$patient, AUC(fit_tree, test),
                             "TREE on liver(trainControl)"))
tbl %>% kable()
```

model	accuracy	sensitivity	specificity	f_score	AUC
GLM (all predictors)	0.7119	0.2059	0.9167	0.2917	0.7384
GLM (6 predictors)	0.7542	0.2941	0.9405	0.4082	0.7496
KNN on liver	0.7458	0.4118	0.8810	0.4828	0.7518
KNN on liver(norm)	0.6356	0.2941	0.7738	0.3175	0.6066
KNN on liver(log)	0.7458	0.5882	0.8095	0.5714	0.8009
TREE on liver	0.6271	0.1176	0.8333	0.1538	0.5746
TREE on liver(trainControl)	0.7119	0.0000	1.0000	NA	0.5000

The goal is to improve prediction performance and reduce instability by averaging multiple decision trees. This is achieved by a forest made of trees constructed with randomness.

Random forest

Algoritam Random Forest with default parameters is train in the next lines of code.

```
set.seed(14, sample.kind = "Rounding")
fit_rf <- train(patient~.,
                  method="rf",
                  data = train)

m <- fit_rf$finalModel$mtry
y_hat <- predict(fit_rf,test)
tbl <- bind_rows(tbl,metrics(y_hat, test$patient, AUC(fit_rf, test),"RANDOM FOREST on liver"))
tbl %>% kable()
```

model	accuracy	sensitivity	specificity	f_score	AUC
GLM (all predictors)	0.7119	0.2059	0.9167	0.2917	0.7384
GLM (6 predictors)	0.7542	0.2941	0.9405	0.4082	0.7496
KNN on liver	0.7458	0.4118	0.8810	0.4828	0.7518
KNN on liver(norm)	0.6356	0.2941	0.7738	0.3175	0.6066
KNN on liver(log)	0.7458	0.5882	0.8095	0.5714	0.8009
TREE on liver	0.6271	0.1176	0.8333	0.1538	0.5746
TREE on liver(trainControl)	0.7119	0.0000	1.0000	NA	0.5000
RANDOM FOREST on liver	0.6441	0.2059	0.8214	0.2500	0.6747

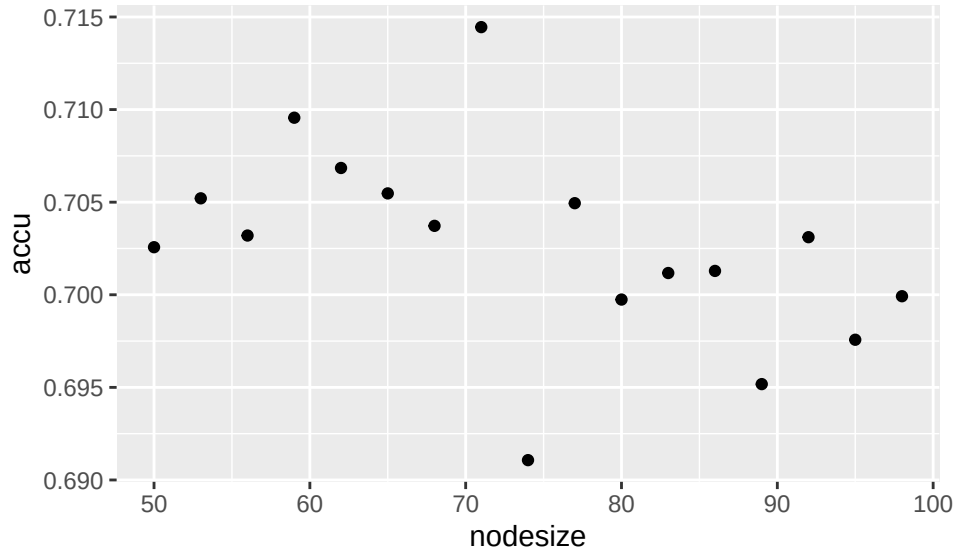
The result is not so good and checking the accuracy on the train set (1), it turns out this algorithm adapted perfectly to the data in the train set, so it is unusable, due to high variance. Let's try to improve the algorithm by adjusting the parameters.

Only those algorithm parameters that have a large effect are available for tuning in caret. As such, only mtry parameter is available in caret for tuning. The caret package is used below to optimize over the minimum node size(parameter that controls the minimum number of data points in the nodes of the tree). The larger this minimum, the smoother the final estimate will be.

```
set.seed(14, sample.kind = "Rounding")
nodesize <- seq(50, 100, 3)
accu <- sapply(nodesize, function(ns){
  train(patient ~ .,
        method = "rf",
        data = train,
        tuneGrid = data.frame(mtry = m),
        nodesize = ns)$results$Accuracy
})
```

The following plot shows the dependence of accuracy on the nodesize parameter.

```
qplot(nodesize, accu)
```



Now, fit the random forest with the optimized minimun node size and train control function to the train data and evaluate performance on the test data.

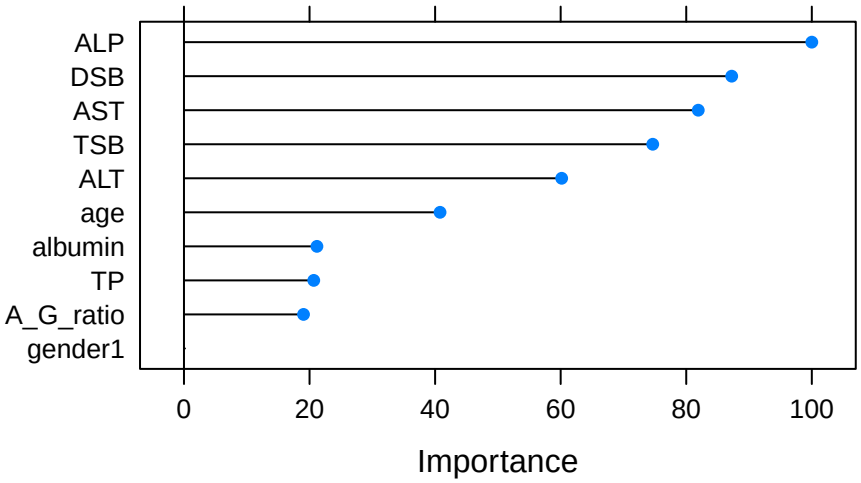
```
set.seed(14, sample.kind = "Rounding")
fit_rf_tune <- train(patient ~ .,
                     data= train,
                     method = "rf",
                     nodesize = nodesize[which.max(accu)],
                     trControl = trainControl(method= "cv", number = 10, p=0.9))

y_hat <- predict(fit_rf_tune, test)
tbl <- bind_rows(tbl,metrics(y_hat, test$patient, AUC(fit_rf_tune, test),"RANDOM FOREST tune"))
tbl %>% kable()
```

model	accuracy	sensitivity	specificity	f_score	AUC
GLM (all predictors)	0.7119	0.2059	0.9167	0.2917	0.7384
GLM (6 predictors)	0.7542	0.2941	0.9405	0.4082	0.7496
KNN on liver	0.7458	0.4118	0.8810	0.4828	0.7518
KNN on liver(norm)	0.6356	0.2941	0.7738	0.3175	0.6066
KNN on liver(log)	0.7458	0.5882	0.8095	0.5714	0.8009
TREE on liver	0.6271	0.1176	0.8333	0.1538	0.5746
TREE on liver(trainControl)	0.7119	0.0000	1.0000	NA	0.5000
RANDOM FOREST on liver	0.6441	0.2059	0.8214	0.2500	0.6747
RANDOM FOREST tune	0.6525	0.1471	0.8571	0.1961	0.6838

The selected model doesn't improves accuracy so much, but based on train accuracy (0.78) provides a smoother estimate and reduce overtraining effect, although it still exists.

The `varImp` function provides information on which features influenced the prediction the most.

```
plot(varImp(fit_rf_tune))
```



CONCLUSION

Four ML models were applied to solve the problem of predicting liver patients based on age, gender and biochemical blood analysis. Then, a table was obtained which contains metrics of interest for each model (accuracy, sensitivity, specificity, F1 score and AUC). Based on the values from the table, we conclude that the **knn** algorithm applied to the transformed liver table (liver_log) proved to be the best.

model	accuracy	sensitivity	specificity	f_score	AUC
KNN on liver(log)	0.7458	0.5882	0.8095	0.5714	0.8009

With more data, better results as well as better model stability would be achieved. Further work on the model may take into account data imbalances (oversample the minority class or undersample the majority class) and further features selection.