Own project - indian liver

Florian Kern

14 June 2019

Table of contents

1. Summary

Reliable diagnosis are essential in medical care. Machine learning provides a great tool to assess the readily available patient data to predict disease state. The indian liver dataset was used to demonstrate this ability. The dataset consists of 416 patients with liver disease and 167 patients with no liver disease and 11 reported variables for each patient. The data set was split into a training and a testing set. After running a few different models an ensembler approach was chosen to use 23 different models to predict disease status and define a cumulated score across all models. The ensembler prediction yieled a an accuracy of 72.17%, which is great improvement over just guessing and could very well support a diagnosis decision.

2. Introduction

Predicting disease outcome based on several measurable variables is one of the best use cases for predicitve statistics and machine learning. There is already a lot of data available that includes numerous observations linked to a disease status e.g. diagnosis. In many cases there are disease postive and negative cases reported.

Observations can range from sequencing results, protein levels, protein activity to general features like Age and Gender. Putting all this together is one of the main efforts undertaken in medical science.

In this project a publicly available data set containing records of liver disease patients and liver-disease free patients is used. It contains 583 observations with 11 variables. 416 records of liver disease patients and 167 records of liver-disease free patients. The 11 observations include Age, Gender as well as a panel of measured protein activities and protein amounts. For further details refer to the Data Set Information part in 3. Methods.

The task was to find a reliable algorithm to predict disease state in respect to the provided variables.

3. Methods

Load libraries:

```
library(dplyr) # a grammar for data manipulation
library(ggplot2) # grammar of graphics
library(caret) # classification and regression training package
library(reshape2) # data reshaping
```

Download data from: https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset)

Data Set Information:

This data set contains 416 liver patient records and 167 non liver patient records.The data set was collected from north east of Andhra Pradesh, India. Selector is a class label used to divide into groups(liver patient or not). This data set contains 441 male patient records and 142 female patient records.

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

Attribute Information:

1. Age Age of the patient
2. Gender Gender of the patient
3. TB Total Bilirubin
4. DB Direct Bilirubin
5. Alkphos Alkaline Phosphotase
6. Sgpt Alamine Aminotransferase
7. Sgot Aspartate Aminotransferase
8. TP Total Protiens
9. ALB Albumin
10. A/G Ratio Albumin and Globulin Ratio
11. Selector field used to split the data into two sets (labeled by the experts)

Load and read the data from csv file:

```
# read csv file
liver_data <- read.csv("indian_liver_patient.csv", sep = ",", header = TRUE)
```

Let's explore the dataset:

```
# data set structure
str(liver_data)
```

```
## 'data.frame':    583 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender                    : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 2 1 1 2 2 ...
##  $ Total_Bilirubin           : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7 ...
##  $ Direct_Bilirubin          : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2 ...
##  $ Alkaline_Phosphotase      : int  187 699 490 182 195 208 154 202 202 290 ...
##  $ Alamine_Aminotransferase  : int  16 64 60 14 27 19 16 14 22 53 ...
##  $ Aspartate_Aminotransferase: int  18 100 68 20 59 14 12 11 19 58 ...
##  $ Total_Protiens            : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 ...
##  $ Albumin                   : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.4 ...
##  $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 ...
##  $ Dataset                   : int  1 1 1 1 1 1 1 1 2 1 ...
```

It contains 583 observations with 11 variables.

```
# data set summary
summary(liver_data)
```

```
##       Age           Gender    Total_Bilirubin  Direct_Bilirubin
##  Min.   : 4.00   Female:142   Min.   : 0.400   Min.   : 0.100
##  1st Qu.:33.00   Male  :441   1st Qu.: 0.800   1st Qu.: 0.200
##  Median :45.00                Median : 1.000   Median : 0.300
##  Mean   :44.75                Mean   : 3.299   Mean   : 1.486
##  3rd Qu.:58.00                3rd Qu.: 2.600   3rd Qu.: 1.300
##  Max.   :90.00                Max.   :75.000   Max.   :19.700
##
```

```
##  Alkaline_Phosphotase Alamine_Aminotransferase Aspartate_Aminotransferase
##  Min.   :  63.0       Min.   :  10.00          Min.   :  10.0
##  1st Qu.: 175.5       1st Qu.:  23.00          1st Qu.:  25.0
##  Median : 208.0       Median :  35.00          Median :  42.0
##  Mean   : 290.6       Mean   :  80.71          Mean   : 109.9
##  3rd Qu.: 298.0       3rd Qu.:  60.50          3rd Qu.:  87.0
##  Max.   :2110.0       Max.   :2000.00          Max.   :4929.0
##
##  Total_Protiens     Albumin      Albumin_and_Globulin_Ratio
##  Min.   :2.700   Min.   :0.900   Min.   :0.3000
##  1st Qu.:5.800   1st Qu.:2.600   1st Qu.:0.7000
##  Median :6.600   Median :3.100   Median :0.9300
##  Mean   :6.483   Mean   :3.142   Mean   :0.9471
##  3rd Qu.:7.200   3rd Qu.:3.800   3rd Qu.:1.1000
##  Max.   :9.600   Max.   :5.500   Max.   :2.8000
##                                  NA's   :4
##     Dataset
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.286
##  3rd Qu.:2.000
##  Max.   :2.000
##
```

Feature names available.

```
# vars in dataset
colnames(liver_data)
```

```
##  [1] "Age"                          "Gender"
##  [3] "Total_Bilirubin"              "Direct_Bilirubin"
##  [5] "Alkaline_Phosphotase"         "Alamine_Aminotransferase"
##  [7] "Aspartate_Aminotransferase" "Total_Protiens"
##  [9] "Albumin"                      "Albumin_and_Globulin_Ratio"
## [11] "Dataset"
```

The columnnames are Age, Gender, Total_Bilirubin, Direct_Bilirubn, Alkaline_Phosphatase, ...

```
# disease distribution
table(liver_data$Dataset)
```

```
##
##   1   2
## 416 167
```

The Dataset column contains the disease outcome variable. 1 means liver disease and 2 means no liver disease. There are 416 patients with disease and 167 patients without.

```
# gender distribution
table(liver_data$Gender)
```

```
##
## Female   Male
##    142    441
```

The dataset contains 142 Female and 441 Male patients.

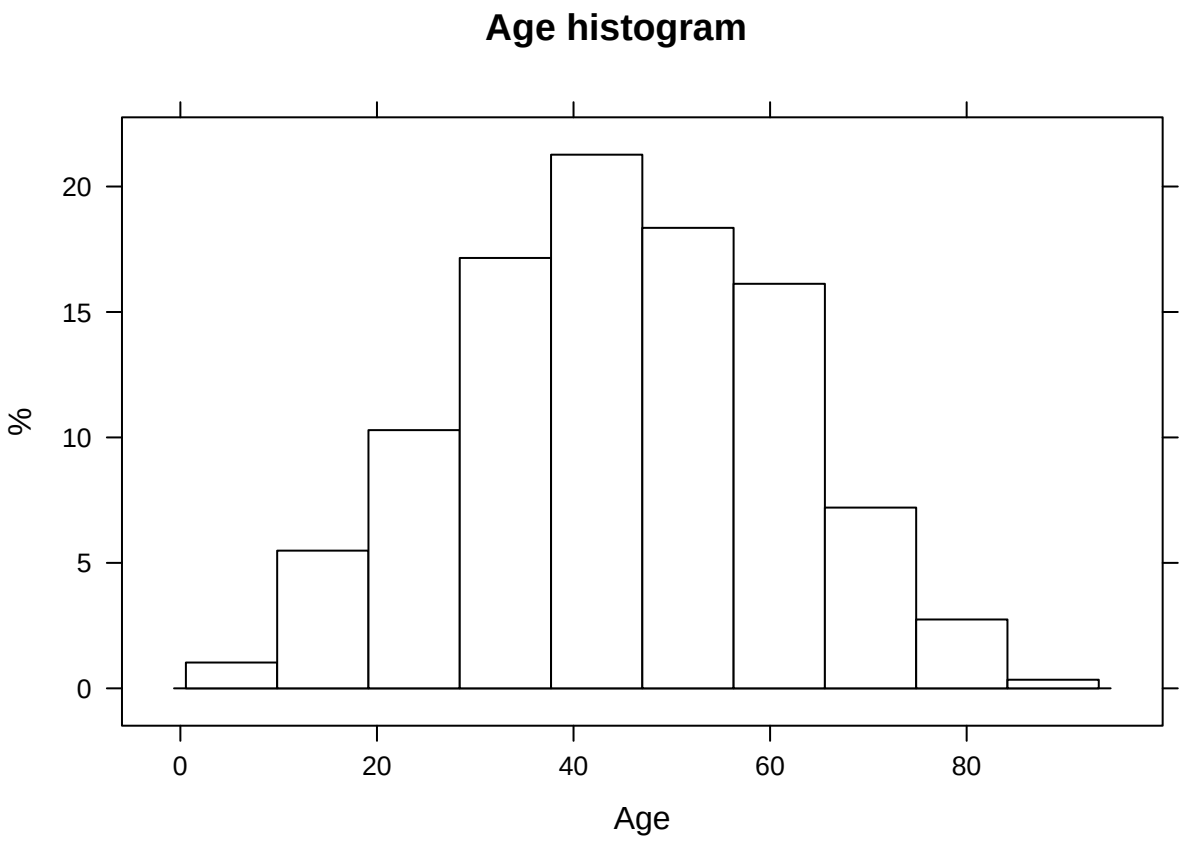
Let's convert "Gender" to a factor:

```r
# factorize gender
liver_data$Gender <- as.factor(liver_data$Gender)
class(liver_data$Gender)
```

```
## [1] "factor"
```

Let's look at the age distribution:

```r
# plot age histogram
histogram(liver_data$Age, main = "Age histogram", xlab = "Age", ylab = "%", col = "white")
```



The majority of cases are between 30 and 60 years of age.

Let's change the columnname from Dataset to Liver Disease and change the values to 0 and 1 instead of 1 and 2. 0 for no disease and 1 for disease.

```r
# change column name
colnames(liver_data)[colnames(liver_data) == "Dataset"] = "Liver_Disease"
colnames(liver_data)
```

```
##  [1] "Age"                        "Gender"
##  [3] "Total_Bilirubin"            "Direct_Bilirubin"
##  [5] "Alkaline_Phosphotase"       "Alamine_Aminotransferase"
##  [7] "Aspartate_Aminotransferase" "Total_Protiens"
##  [9] "Albumin"                    "Albumin_and_Globulin_Ratio"
## [11] "Liver_Disease"
```

```r
# change disease status to 0 an 1
liver_data$Liver_Disease[liver_data$Liver_Disease == 2] = 0
table(liver_data$Liver_Disease)
```

```
##
##   0   1
## 167 416
```

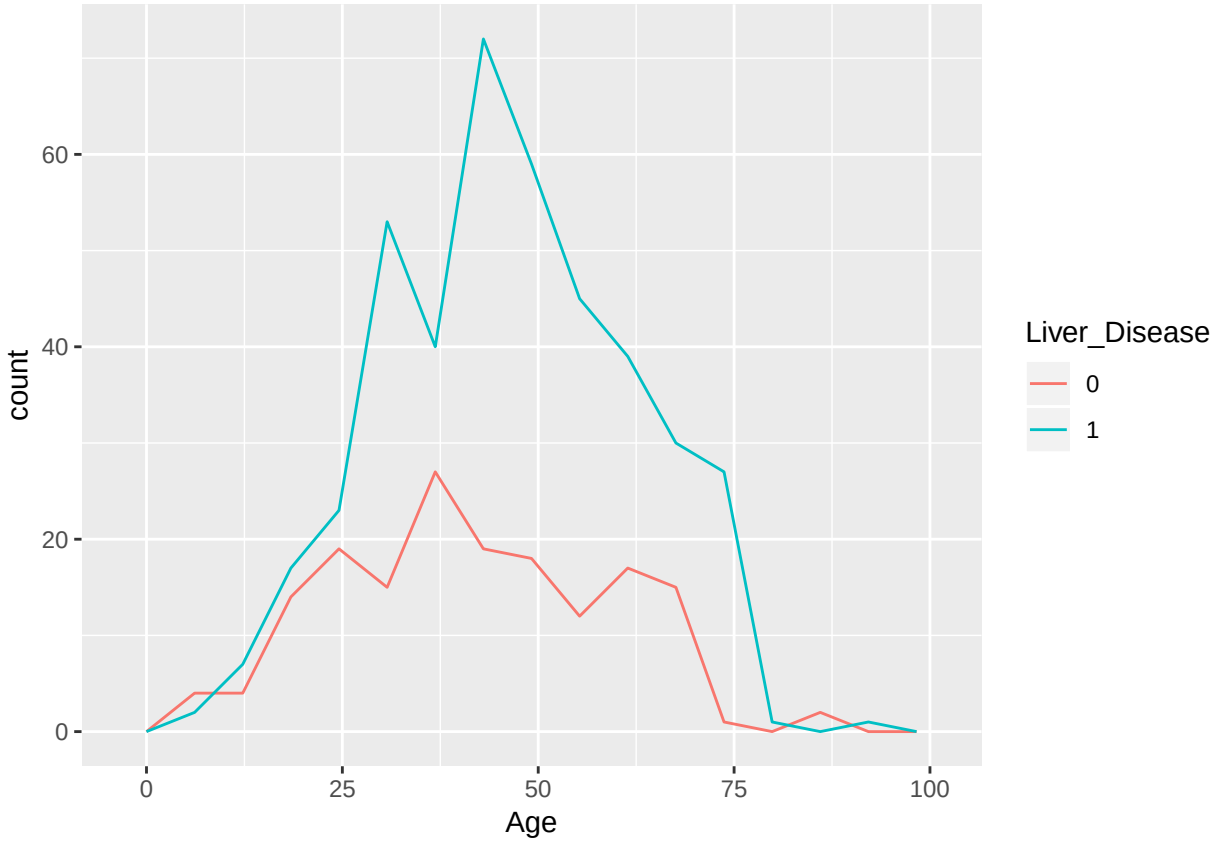
Convert Liver_Disease to a factor:

```
# factorize disease status
liver_data$Liver_Disease = as.factor(liver_data$Liver_Disease)
str(liver_data)
```

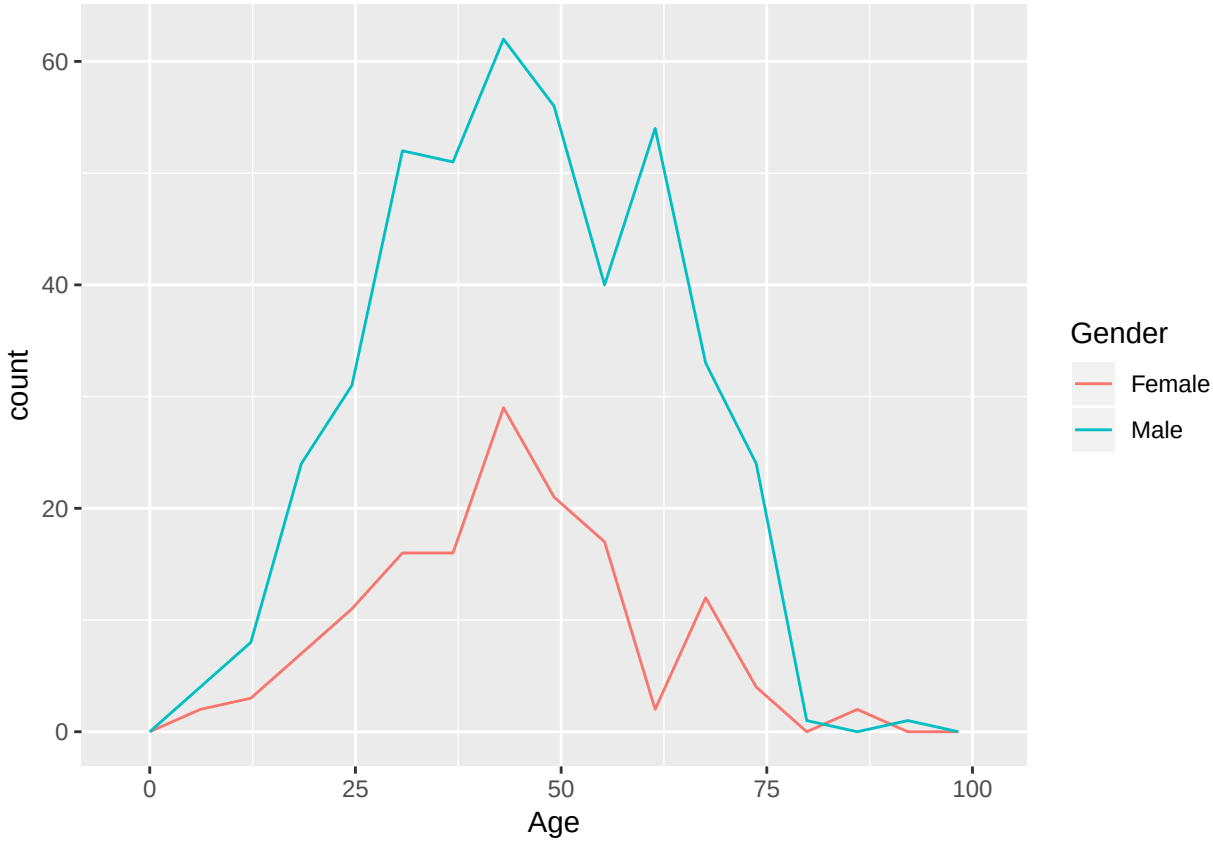
```
## 'data.frame':    583 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender                    : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 2 1 1 2 2 ...
##  $ Total_Bilirubin           : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7 ...
##  $ Direct_Bilirubin          : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2 ...
##  $ Alkaline_Phosphotase      : int  187 699 490 182 195 208 154 202 202 290 ...
##  $ Alamine_Aminotransferase  : int  16 64 60 14 27 19 16 14 22 53 ...
##  $ Aspartate_Aminotransferase: int  18 100 68 20 59 14 12 11 19 58 ...
##  $ Total_Protiens            : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 ...
##  $ Albumin                   : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.4 ...
##  $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 ...
##  $ Liver_Disease             : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1 2 ...
```

Let's look at age and gender in the context of disease status

```
liver_data %>% ggplot(aes(Age, color= Liver_Disease)) +
  geom_freqpoly(bins=15)
```




```
liver_data %>% ggplot(aes(Age, color= Gender)) +
  geom_freqpoly(bins=15)
```



Both look more or less equally distributed.

Let's check for NAs in each column:

```
# check for NAs
sapply(liver_data, function(x) sum(is.na(x)))
```

```
##                         Age                      Gender
##                           0                           0
##             Total_Bilirubin            Direct_Bilirubin
##                           0                           0
##       Alkaline_Phosphotase   Alamine_Aminotransferase
##                           0                           0
## Aspartate_Aminotransferase             Total_Protiens
##                           0                           0
##                     Albumin Albumin_and_Globulin_Ratio
##                           0                           4
##               Liver_Disease
##                           0
```

There are 4 NAs in the Albumin_and_Globulin_Ratio column.

```
# calculate the percentage of NAs
sum(is.na(liver_data)/nrow(liver_data))
```

```
## [1] 0.006861063
```

4 rows account for less than 0.69% of the data. Let's remove them.

```r
# remove NA entries
liver_data = na.omit(liver_data)
dim(liver_data)
```

```
## [1] 579  11
```

Let's look at correlations:

```r
# calculate the correlations of variables without Gender and Disease Status
correlations <- cor(liver_data[,c(-2,-11)])
correlations
```

```
##                                    Age Total_Bilirubin Direct_Bilirubin
## Age                         1.000000000     0.011000374     6.784303e-03
## Total_Bilirubin             0.011000374     1.000000000     8.744810e-01
## Direct_Bilirubin            0.006784303     0.874480969     1.000000e+00
## Alkaline_Phosphotase        0.078878350     0.205739173     2.340076e-01
## Alamine_Aminotransferase   -0.087799162     0.213375493     2.331801e-01
## Aspartate_Aminotransferase -0.020498946     0.237323055     2.570224e-01
## Total_Protiens             -0.186248122    -0.007905923     3.270877e-05
## Albumin                    -0.264210935    -0.222086570    -2.284092e-01
## Albumin_and_Globulin_Ratio -0.216408346    -0.206267186    -2.001247e-01
##                            Alkaline_Phosphotase Alamine_Aminotransferase
## Age                                  0.07887835              -0.08779916
## Total_Bilirubin                      0.20573917               0.21337549
## Direct_Bilirubin                     0.23400757               0.23318008
## Alkaline_Phosphotase                 1.00000000               0.12477671
## Alamine_Aminotransferase             0.12477671               1.00000000
## Aspartate_Aminotransferase           0.16657999               0.79186215
## Total_Protiens                      -0.02706202              -0.04243210
## Albumin                             -0.16341865              -0.02865750
## Albumin_and_Globulin_Ratio          -0.23416650              -0.00237499
##                            Aspartate_Aminotransferase Total_Protiens
## Age                                        -0.02049895  -1.862481e-01
## Total_Bilirubin                             0.23732305  -7.905923e-03
## Direct_Bilirubin                            0.25702239   3.270877e-05
## Alkaline_Phosphotase                        0.16657999  -2.706202e-02
## Alamine_Aminotransferase                    0.79186215  -4.243210e-02
## Aspartate_Aminotransferase                  1.00000000  -2.575101e-02
## Total_Protiens                             -0.02575101   1.000000e+00
## Albumin                                    -0.08491457   7.831122e-01
## Albumin_and_Globulin_Ratio                 -0.07003983   2.348872e-01
##                               Albumin Albumin_and_Globulin_Ratio
## Age                        -0.26421094                -0.21640835
## Total_Bilirubin            -0.22208657                -0.20626719
## Direct_Bilirubin           -0.22840915                -0.20012469
## Alkaline_Phosphotase       -0.16341865                -0.23416650
## Alamine_Aminotransferase   -0.02865750                -0.00237499
## Aspartate_Aminotransferase -0.08491457                -0.07003983
## Total_Protiens              0.78311217                 0.23488718
## Albumin                     1.00000000                 0.68963234
## Albumin_and_Globulin_Ratio  0.68963234                 1.00000000
```

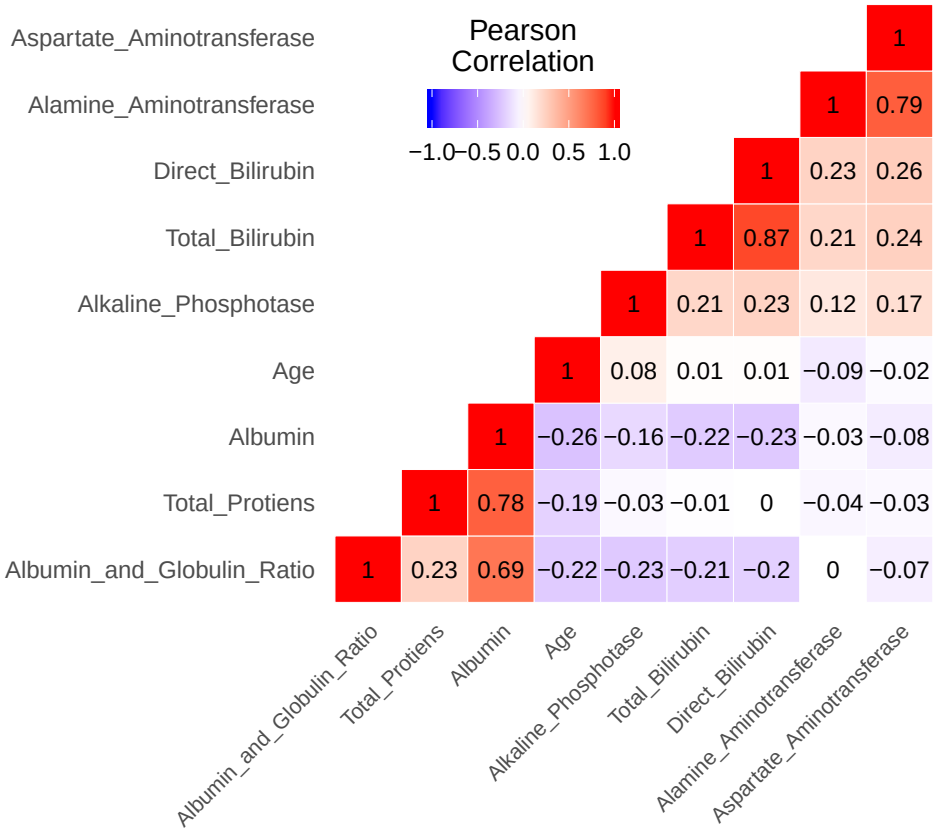
```r
# plot the correlation triangle as heatmap
# function to get lower triangle of the correlation matrix
get_lower_tri<-function(cormat){
```

```
    cormat[upper.tri(cormat)] <- NA
    return(cormat)
  }
# function to get upper triangle of the correlation matrix
get_upper_tri <- function(cormat){
    cormat[lower.tri(cormat)]<- NA
    return(cormat)
  }

# function to reorder data
reorder_cormat <- function(cormat){
# Use correlation between variables as distance
dd <- as.dist((1-cormat)/2)
hc <- hclust(dd)
cormat <-cormat[hc$order, hc$order]
}

# reorder the correlation matrix
cormat <- round(reorder_cormat(correlations),2)
# get the upper triangle
upper_tri <- get_upper_tri(cormat)
# melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)
# create a ggheatmap
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0,
                       limit = c(-1,1), space = "Lab", name="Pearson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 8, hjust = 1)) +
  coord_fixed()

# add correlation coefficient labels onto the heatmap
ggheatmap +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 3) +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    axis.ticks = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.5, 0.7),
    legend.direction = "horizontal") +
  guides(fill = guide_colorbar(barwidth = 5, barheight = 1, title.position = "top", title.hjust = 0.5))
```



Let's find columns that have high pairwise correlation and remove them.

```
# find highly correlated variables
high_coor_cols <- findCorrelation(correlations, cutoff = 0.7, names = TRUE)
high_coor_cols
```

```
## [1] "Albumin"                    "Direct_Bilirubin"
## [3] "Aspartate_Aminotransferase"
```

```
# remove highly correlated (redundant) data
liver_data <- liver_data[, !names(liver_data) %in% high_coor_cols]
dim(liver_data)
```

```
## [1] 579   8
```

Let's split the data to obtain a training and a testing set:

```
# set a seed
set.seed(1)
# split data set into train and test
index <- createDataPartition(liver_data$Liver_Disease, p = 0.8, list = FALSE)
train = liver_data[index,]
test = liver_data[-index,]
dim(train)
```

```
## [1] 464   8
```

```
dim(test)
```

```
## [1] 115   8
```

The training set contains 80% of the data.

Many models do not support non-numerical values, therefore we convert the Gender column to numbers.

```
# remove gender factor
train$Gender <- as.numeric(train$Gender)
test$Gender <- as.numeric(test$Gender)
```

4. Results

4.1 Naive Bayes

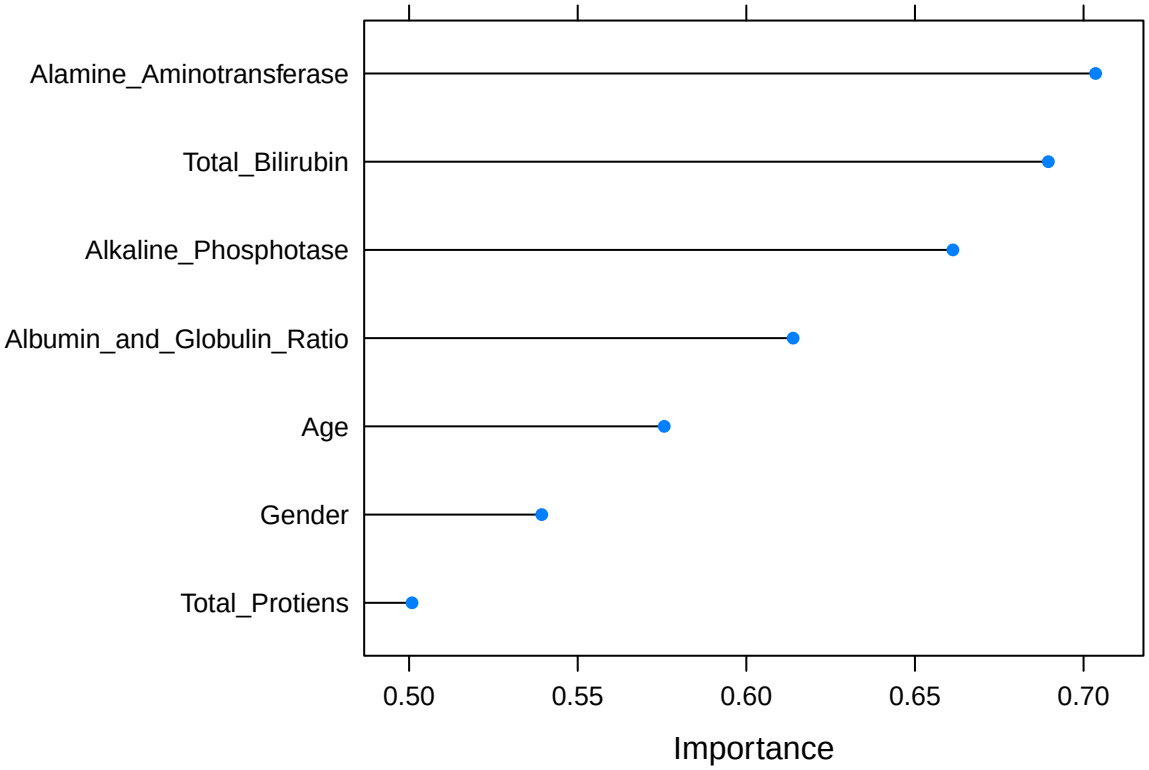
```
# set seed
set.seed(1)
# fit/train the model
model1_nb_fit <- train(Liver_Disease ~ ., data = train, method = "nb")
model1_nb_fit
```

```
## Naive Bayes
##
## 464 samples
##   7 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 464, 464, 464, 464, 464, 464, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.5812435  0.2634180
##    TRUE      0.6478134  0.2361091
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = TRUE
##  and adjust = 1.
```

```
# var importance
model1_nb_imp <- varImp(model1_nb_fit, scale = FALSE)
model1_nb_imp
```

```
## ROC curve variable importance
##
##                            Importance
## Alamine_Aminotransferase       0.7036
## Total_Bilirubin                0.6896
## Alkaline_Phosphotase           0.6613
## Albumin_and_Globulin_Ratio     0.6139
## Age                            0.5757
## Gender                         0.5393
## Total_Protiens                 0.5009
```

```r
# plot var importance
plot(model1_nb_imp)
```



The important variables in this model are Alamine_Aminotransferase, Total_Bilirubin, and Alkaline_Phosphotase.

Let's predict the test data and look at the confusion matrix and accuracy.

```r
# predict outcomes on test
model1_nb_pred <- predict(model1_nb_fit, test)
# confusion matrix
model1_cm <- confusionMatrix(model1_nb_pred, test$Liver_Disease)
model1_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23 24
##          1 10 58
##
##                Accuracy : 0.7043
##                  95% CI : (0.6121, 0.7858)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.62638
##
##                   Kappa : 0.3588
##
##  Mcnemar's Test P-Value : 0.02578
##
```

```
##             Sensitivity : 0.6970
##             Specificity : 0.7073
##          Pos Pred Value : 0.4894
##          Neg Pred Value : 0.8529
##              Prevalence : 0.2870
##          Detection Rate : 0.2000
##    Detection Prevalence : 0.4087
##       Balanced Accuracy : 0.7021
##
##        'Positive' Class : 0
##
```

```
# model accuracy
model1_acc <- model1_cm$overall["Accuracy"]
model1_acc
```

```
##  Accuracy
## 0.7043478
```

The accuracy of the naive bayes model is about 70.43%. which seems alright.

4.2 Logistic Regression - GLM

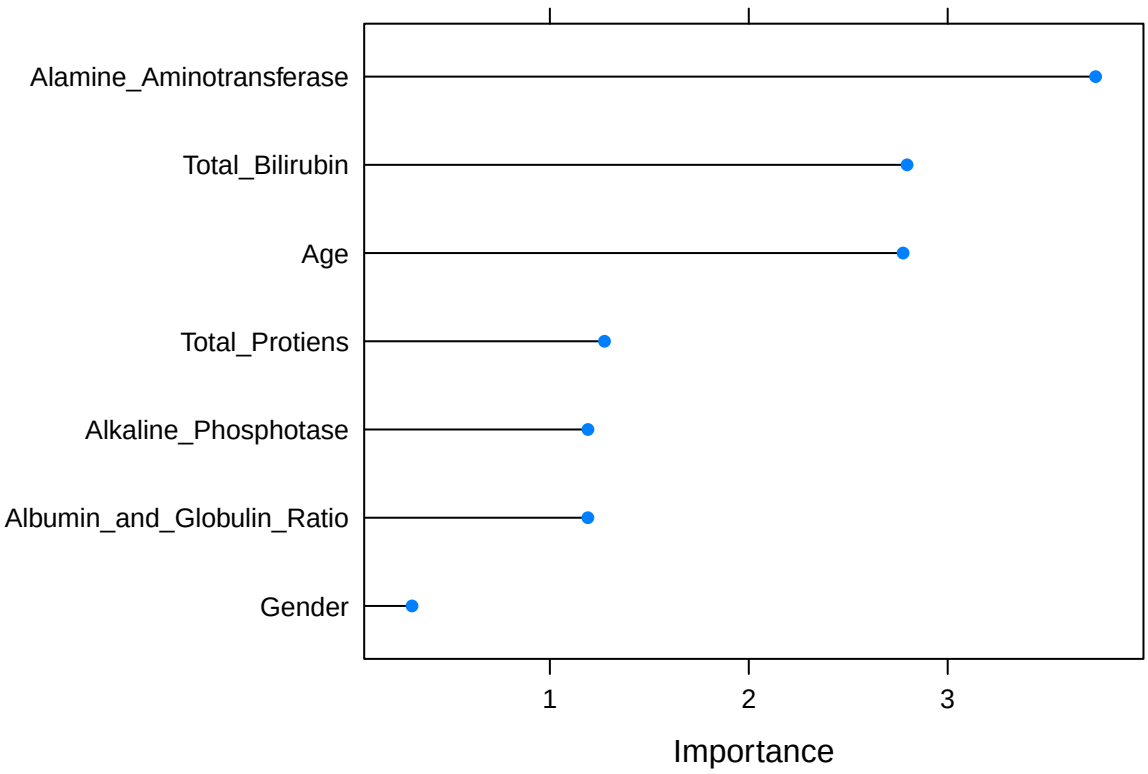
```
# set seed
set.seed(2)
# fit/train the model
model2_glm_fit <- train(Liver_Disease ~., data = train, method = "glm", family = "binomial")
model2_glm_fit
```

```
## Generalized Linear Model
##
## 464 samples
##   7 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 464, 464, 464, 464, 464, 464, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7116226  0.1816424
```

Importance

```
# var importance
model2_glm_imp <- varImp(model2_glm_fit, scale = FALSE)
```

```
# plot var importance
plot(model2_glm_imp)
```



The important variables in this model are Alamine_Aminotransferase, Total_Bilirubin, and Age.

Let's predict the test data and look at the confusion matrix and accuracy.

```
# predict outcomes on test
model2_glm_pred <- predict(model2_glm_fit, test)
# confusion matrix
model2_cm <- confusionMatrix(model2_glm_pred, test$Liver_Disease)
model2_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  3  3
##          1 30 79
##
##                Accuracy : 0.713
##                  95% CI : (0.6212, 0.7935)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.5468
##
##                   Kappa : 0.0719
##
##  Mcnemar's Test P-Value : 6.011e-06
##
##             Sensitivity : 0.09091
##             Specificity : 0.96341
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.72477
##              Prevalence : 0.28696
```

```
##          Detection Rate : 0.02609
##    Detection Prevalence : 0.05217
##       Balanced Accuracy : 0.52716
##
##        'Positive' Class : 0
##
```

```
# model accuracy
model2_acc <- model2_cm$overall["Accuracy"]
model2_acc
```

```
##  Accuracy
## 0.7130435
```

The accuracy of the glm model is 71.31%.

4.3 knn

```
# set seed
set.seed(2)
# # fit/train the model
model3_knn_fit <- train(Liver_Disease ~ ., data = train, method = "knn")
model3_knn_fit
```

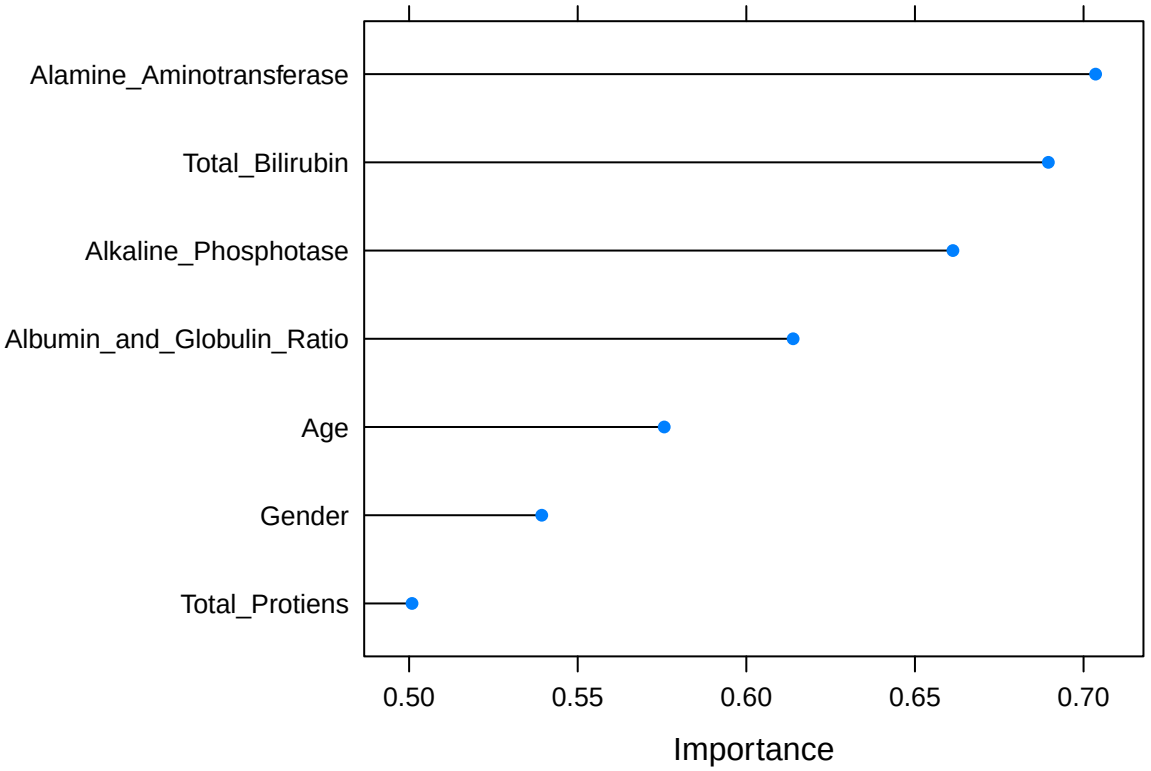
```
## k-Nearest Neighbors
##
## 464 samples
##   7 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 464, 464, 464, 464, 464, 464, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6628229  0.1402657
##   7  0.6715226  0.1457653
##   9  0.6722574  0.1382695
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
# var importance
model3_knn_imp <- varImp(model3_knn_fit, scale = FALSE)
model3_knn_imp
```

```
## ROC curve variable importance
##
##                            Importance
## Alamine_Aminotransferase       0.7036
## Total_Bilirubin                0.6896
## Alkaline_Phosphotase           0.6613
## Albumin_and_Globulin_Ratio     0.6139
```

```
## Age                              0.5757
## Gender                           0.5393
## Total_Protiens                   0.5009
```

```
# plot var importance
plot(model3_knn_imp)
```



The important variables are Alamine_Aminotransferase, Total_Bilirubin, and Alkaline_Phosphotase.

Let's predict the test data and look at the confusion matrix and accuracy.

```
# predict outcomes on test
model3_knn_pred <- predict(model3_knn_fit, test)
# confusion matrix
model3_cm <- confusionMatrix(model3_knn_pred, test$Liver_Disease)
model3_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  7  8
##          1 26 74
##
##                Accuracy : 0.7043
##                  95% CI : (0.6121, 0.7858)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.626380
##
##                   Kappa : 0.1369
##
```

```
##  Mcnemar's Test P-Value : 0.003551
##
##             Sensitivity : 0.21212
##             Specificity : 0.90244
##          Pos Pred Value : 0.46667
##          Neg Pred Value : 0.74000
##              Prevalence : 0.28696
##          Detection Rate : 0.06087
##    Detection Prevalence : 0.13043
##       Balanced Accuracy : 0.55728
##
##        'Positive' Class : 0
##
```

```
# model accuracy
model3_acc <- model3_cm$overall["Accuracy"]
model3_acc
```

```
##  Accuracy
## 0.7043478
```

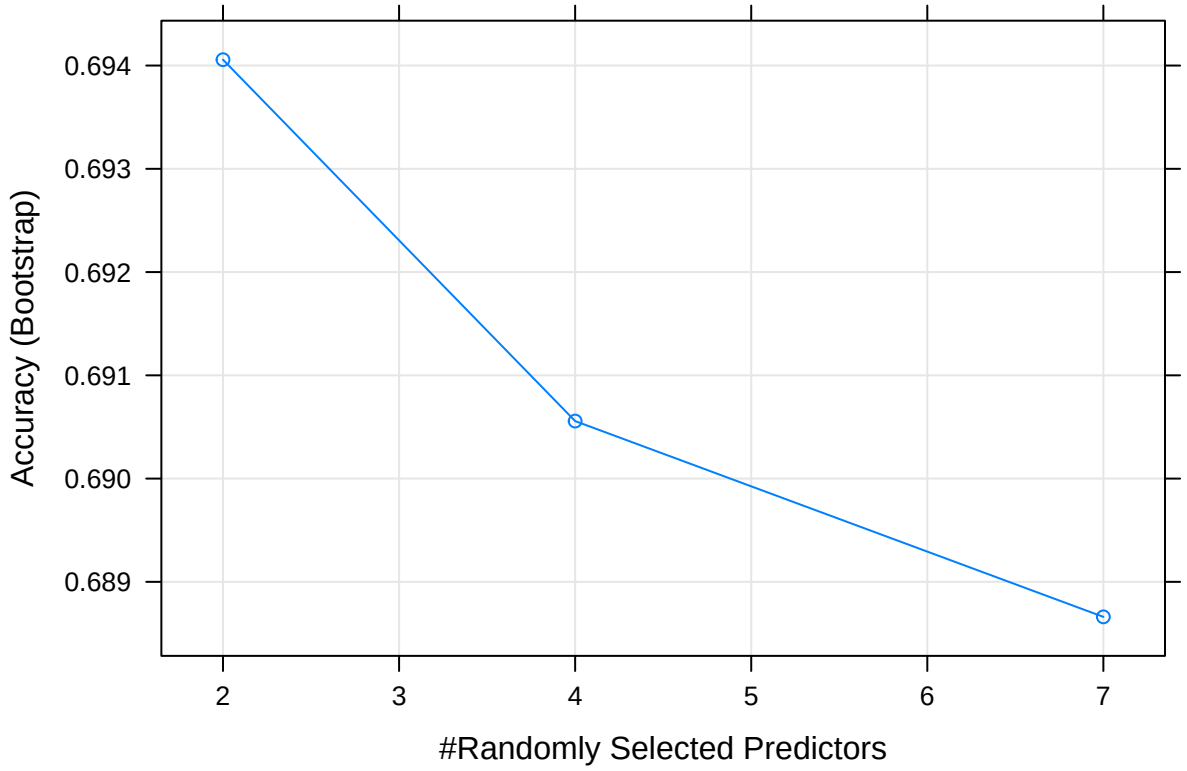
The accuracy of the knn model is 70.34%

4.4 Random Forest

```
# set seed
set.seed(3)
# # fit/train the model
model4_rf = train(Liver_Disease ~ ., method = "rf", data = train, prox = TRUE)
model4_rf
```

```
## Random Forest
##
## 464 samples
##   7 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 464, 464, 464, 464, 464, 464, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.6940568  0.1725523
##   4     0.6905570  0.1811999
##   7     0.6886609  0.1850418
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

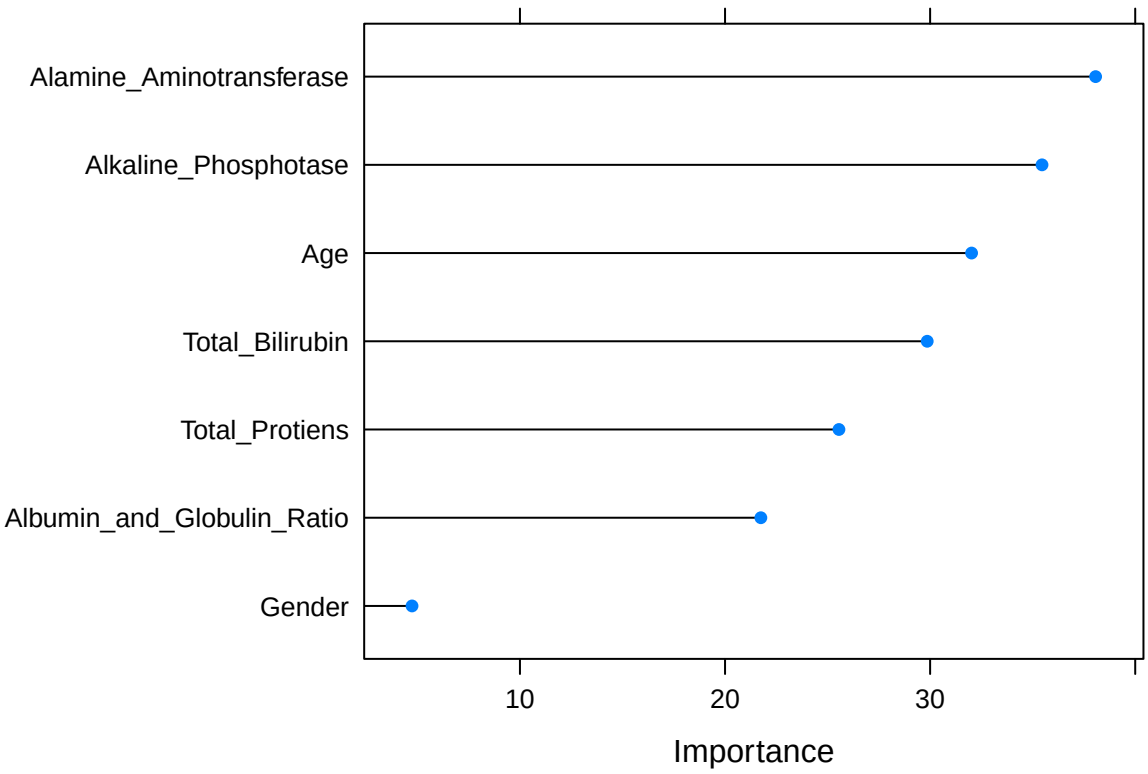
```
# plot trained models mtrys
plot(model4_rf)
```




```
# var importance
model4_rf_imp <- varImp(model4_rf, scale = FALSE)
model4_rf_imp
```

```
## rf variable importance
##
##                            Overall
## Alamine_Aminotransferase    38.069
## Alkaline_Phosphotase        35.456
## Age                         32.022
## Total_Bilirubin             29.858
## Total_Protiens              25.554
## Albumin_and_Globulin_Ratio  21.746
## Gender                       4.737
```

```
# plot var importance
plot(model4_rf_imp)
```



The important variables are Alamine_Aminotransferase, Alkaline_Phosphotase, and Age.

Let's predict the test data and look at the confusion matrix and accuracy.

```
# predict outcomes on test
model4_pred <- predict(model4_rf, test)
# confusion matrix
model4_cm <- confusionMatrix(model4_pred, test$Liver_Disease)
model4_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  9  7
##          1 24 75
##
##                Accuracy : 0.7304
##                  95% CI : (0.6397, 0.8089)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.383747
##
##                   Kappa : 0.2214
##
##  Mcnemar's Test P-Value : 0.004057
##
##             Sensitivity : 0.27273
##             Specificity : 0.91463
##          Pos Pred Value : 0.56250
##          Neg Pred Value : 0.75758
##              Prevalence : 0.28696
```

```
##          Detection Rate : 0.07826
##    Detection Prevalence : 0.13913
##       Balanced Accuracy : 0.59368
##
##        'Positive' Class : 0
##
```

```
# model accuracy
model4_acc <- model4_cm$overall["Accuracy"]
model4_acc
```

```
##  Accuracy
## 0.7304348
```

The random forest model yields a 73,04% accuracy.

Overview accuracy

```
# create table with accuracies overview
tibble(method = c("naive bayes", "glm", "knn", "random forest"),
       Accuracy = c(model1_acc, model2_acc, model3_acc, model4_acc))
```

```
## # A tibble: 4 x 2
##   method        Accuracy
##   <chr>            <dbl>
## 1 naive bayes      0.704
## 2 glm              0.713
## 3 knn              0.704
## 4 random forest    0.730
```

Naive Bayes and knn models give the lowest accuracy with 70.04%. The random forest model yielded the highes accuracy with 73.04% accuracy.

Testing each model is getting tidious pretty quickly. Let's use an ensembler with different models.

4.5 Ensembler

Several models are available in the caret package. We will use 22 models and see how the ensembler as well as the individual models perform.

```
# list of models
models <- c("glm", "lda",  "naive_bayes",  "svmLinear", "gamLoess",
            "qda", "knn", "kknn", "loclda", "gam", "rf", "ranger",
            "wsrf", "Rborist", "avNNet", "mlp", "monmlp", "adaboost",
            "gbm", "svmRadial", "svmRadialCost", "svmRadialSigma")
```

Let's train each model on our training data:

```
# set seed
set.seed(1)
# fit/train models
fits <- lapply(models, function(model){
  print(model)
```

```
  train(Liver_Disease ~ ., method = model, data = train)
})
```

Use the trained models to predict the testing data:

```
# predict with each model
fits_predicts <- sapply(fits, function(fits){
  predict(fits, test)
})
```

Summarize the accuracies for each model on the test set.

```
# calculate accuracies for each model
acc <- colMeans(fits_predicts == test$Liver_Disease)
acc
```

```
##            glm            lda    naive_bayes      svmLinear       gamLoess
##      0.7130435      0.7130435      0.7043478      0.7130435      0.7043478
##            qda            knn           kknn         loclda            gam
##      0.4782609      0.7043478      0.6869565      0.7304348      0.7043478
##             rf         ranger           wsrf        Rborist         avNNet
##      0.7217391      0.7304348      0.6956522      0.7130435      0.7130435
##            mlp         monmlp       adaboost            gbm      svmRadial
##      0.7130435      0.6347826      0.7217391      0.7043478      0.7130435
##  svmRadialCost svmRadialSigma
##      0.7130435      0.7130435
```

The accuracy ranges from very low 47.83% to 73.04%. Below 50% is worse than guessing.

Let's use the individual predictions of the ensembler to calculate an accumulated prediction:

```
# obtain mean score of predictions
votes <- rowMeans(fits_predicts == 1)
# accumulated voting, below mean of 0.5 vote is 0
y_hat <- ifelse(votes > 0.5, 1, 0)
# ensembler accuracy mean is mean of accumulated votes accuracy
ensembler_mean <- mean(y_hat == test$Liver_Disease)
ensembler_mean
```

```
## [1] 0.7217391
```

The ensembler yields an average accuracy of 72.17%.

For a quick sanity check, which models perform individually better than the mean of the ensembler:

```
# which individual models are better than the ensembler mean
better <- ensembler_mean < acc
acc[which(better==T)]
```

```
##    loclda    ranger
## 0.7304348 0.7304348
```

Only the loclda and the ranger model with an accuracy of 73.04% have a higher accuracy in predicting liver disease than the accumulated ensembler prediction with an accuracy of 72.17%.

5. Conclusions

Overall the accuracy of 72.17% using the ensembler approach is a lot higher than just guessing but certainly not enough for a deployment. The localda and ranger model yielded the highest accuracy of 73.04%. A much larger dataset would be needed to test the reliability of the ensembler. Anyway it is a proof of concept that an ensembler consisting of numerous models is the way to go forward and with further optimization e.g tweaking the model parameters or data transformation would be needed to make this a reliable tool for a real world application. In any case it could already be used to support a diagnosis decision.