Indian Liver Study - CYO

Rei Romero

10/7/2022

Executive Summary

Liver health is important to maintaining a good quality of life as there can be many detrimental effects that can be borne from poor liver health such as fatigue, nausea, vomiting, and jaundice (eyes and skin appearing yellow).

Thus, it would be useful to construct a model or multiple models even that would accurately predict the onset of liver disease in a patient. That is the objective of this study.

We will be using a data set collected from test samples in North East of Andhra Pradesh, India wherein there is a total of 583 observations, 416 of which suffer from some sort of liver disease, while 142 of which do not.

The following is the data set to be used:

```
data <- read.csv("./indian_liver_patient.csv")

str(data)
```

```
## 'data.frame':    583 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender                    : chr  "Female" "Male" "Male" "Male" ...
##  $ 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 ...
```

The data set contains 441 male patients and 142 female patients. The variables include basic demographic data such as age and gender, while the rest of the variables are those that relate to liver health in that they are compounds, enzymes, or other such biological

phenomena that are indicators of liver health. Note that the variable to be predicted is the "Dataset" variable where a "1" denotes the presence of liver disease in a patient and a "2" indicates the lack of liver disease in a patient.

Key steps

An overview for the steps that were taken is as follows:

1. Data cleaning (observations with NAs were removed).
2. A 70-30 training-test split was used on the data.
3. Data exploration and visualization were employed comparing the group which had liver disease and the group which did not have liver disease.
4. Logistic regression was employed on the training data, and the best logistic regression model was chosen through backward elimination and by removing models which did not satisfy the assumptions of logistic regression.
5. Predictions for liver disease for the testing data were computed using the logistic regression model found in step 4, and the corresponding confusion matrix was computed (using the the default threshold of labeling an observation to have liver disease if the probability of said observation was greater than 0.5).
6. The logistic regression model found in step 4 was designated as the final logistic regression model and its threshold for labeling an observation as having liver disease was tweaked to maximize sensitivity, while keeping the model's informedness score above 0.
7. The ROC Curve for the final logistic regression model was constructed.
8. XGBoost was applied to the training data using various sets of hyperparameters and the best performing XGBoost model in terms of accuracy was chosen to be the final XGBoost model.
9. The testing data's Dataset values were predicted using the final XGBoost model and its corresponding confusion matrix and ROC curve were constructed.
10. We compare the performance of the final logistic regression model and the final XGBoost model using their respective confusion matrix and ROC curve.

Analysis

We will now be going step-by-step through the process that was undertaken for this study.

We first start with loading all the necessary packages:

```
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-
project.org")
```

```
if(!require(caTools)) install.packages("caTools", repos = "http://cran.us.r-
project.org")
if(!require(ROCR)) install.packages("ROCR", repos = "http://cran.us.r-
project.org")
if(!require(pscl)) install.packages("pscl", repos = "http://cran.us.r-
project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-
project.org")
if(!require(car)) install.packages("car", repos = "http://cran.us.r-
project.org")
if(!require(broom)) install.packages("broom", repos = "http://cran.us.r-
project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos =
"http://cran.us.r-project.org")
if(!require(boot)) install.packages("boot", repos = "http://cran.us.r-
project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-
project.org")

library(dplyr)
library(caTools)
library(ROCR)
library(pscl)
library(caret)
library(car)
library(broom)
library(tidyverse)
library(boot)
library(ggplot2)
```

We then proceed with data cleaning and splitting the data into a training set and a testing set. Note that we choose a 70-30 train-test split since there are only a few hundred observations and we want to keep the proportion of the test set quite high since overfitting could easily happen with this relatively small data set, while also giving the logistic regression models and XGBoost models enough data to be trained on. The following is the code for data cleaning and splitting the data:

Data cleaning and splitting

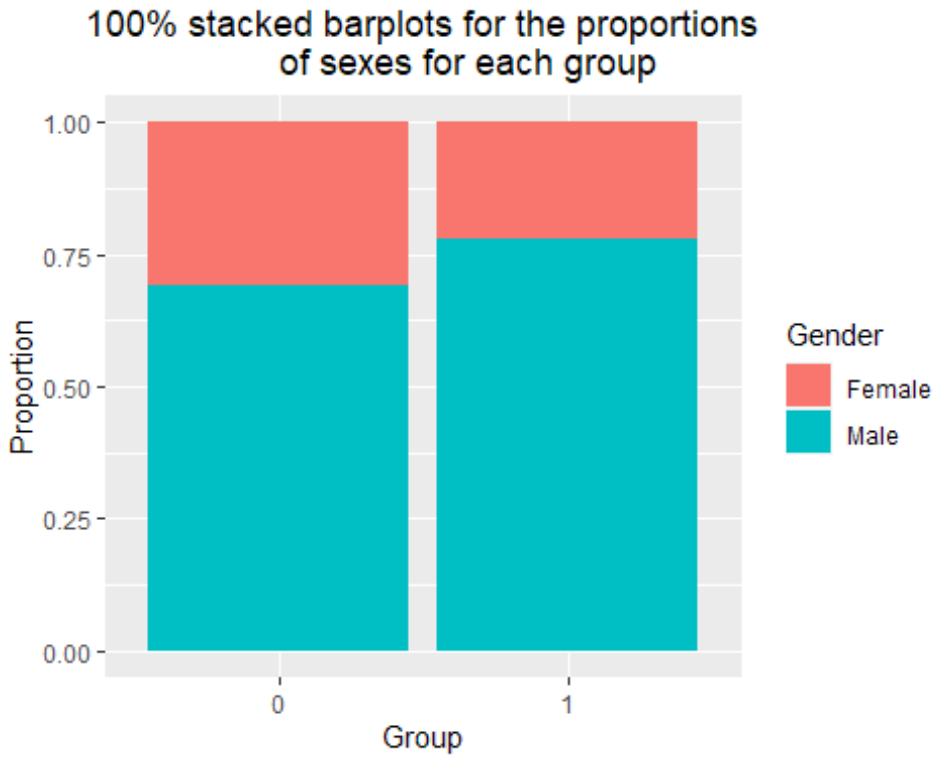
```
sum(is.na(data))

## [1] 4

# Removing NAs since there are only four observations with NAs, at most
data <- na.omit(data)

# Turn 2's in data$Dataset into "0" which means the patient does not have
# liver disease
data$Dataset[data$Dataset == 2] <- 0

# Turn outcome variables into factor variables
```

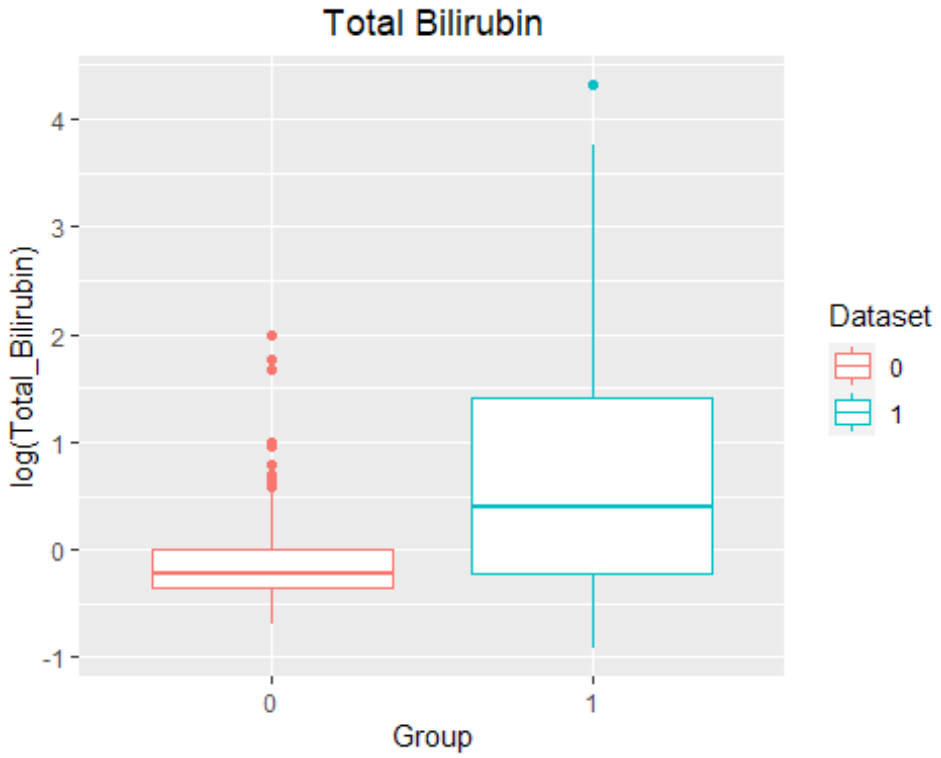
```
data$Dataset <- as.factor(data$Dataset)

# Splitting data into training set and testing set
set.seed(1)

# We use 70% of dataset as training set and 30% as test set
sample <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.7,0.3))
train  <- data[sample, ]
test   <- data[!sample, ]

str(train)
```

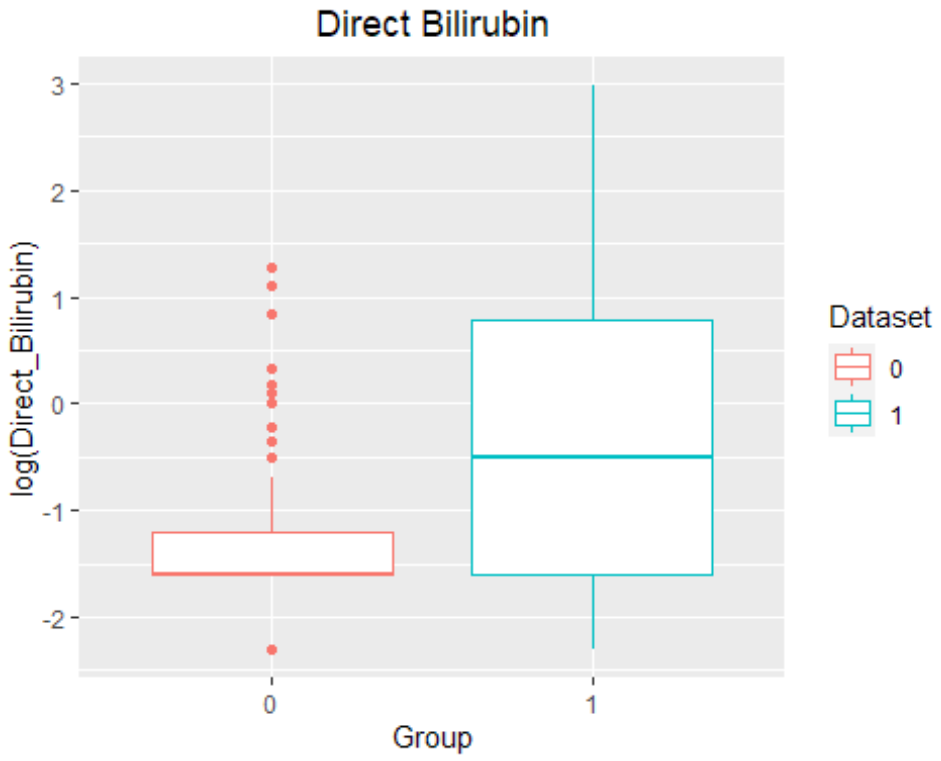
```
## 'data.frame':    408 obs. of  11 variables:
##  $ Age                       : int  65 62 62 72 29 17 55 57 72 64 ...
##  $ Gender                    : chr  "Female" "Male" "Male" "Male" ...
##  $ Total_Bilirubin           : num  0.7 10.9 7.3 3.9 0.9 0.9 0.7 0.6 2.7
0.9 ...
##  $ Direct_Bilirubin          : num  0.1 5.5 4.1 2 0.3 0.3 0.2 0.1 1.3 0.3
...
##  $ Alkaline_Phosphotase      : int  187 699 490 195 202 202 290 210 260
310 ...
##  $ Alamine_Aminotransferase  : int  16 64 60 27 14 22 53 51 31 61 ...
##  $ Aspartate_Aminotransferase: int  18 100 68 59 11 19 58 59 56 58 ...
##  $ Total_Protiens            : num  6.8 7.5 7 7.3 6.7 7.4 6.8 5.9 7.4 7
...
##  $ Albumin                   : num  3.3 3.2 3.3 2.4 3.6 4.1 3.4 2.7 3 3.4
...
##  $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 0.4 1.1 1.2 1 0.8 0.6
0.9 ...
##  $ Dataset                   : Factor w/ 2 levels "0","1": 2 2 2 2 2 1 2 2
2 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:4] 210 242 254 313
##   ..- attr(*, "names")= chr [1:4] "210" "242" "254" "313"
```

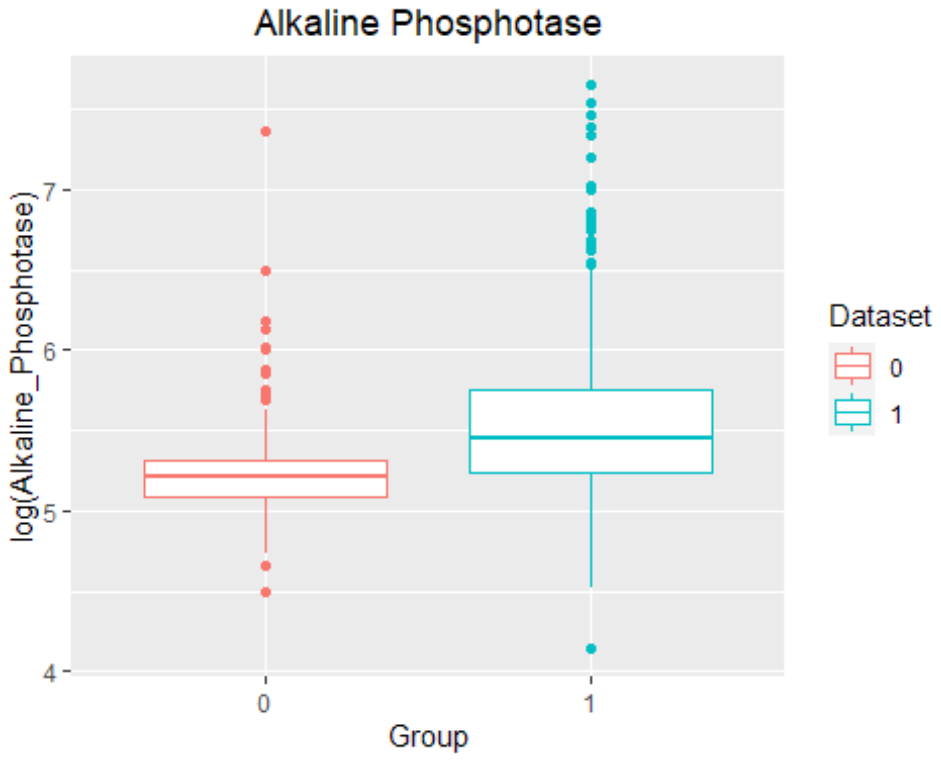
Data exploration and visualization

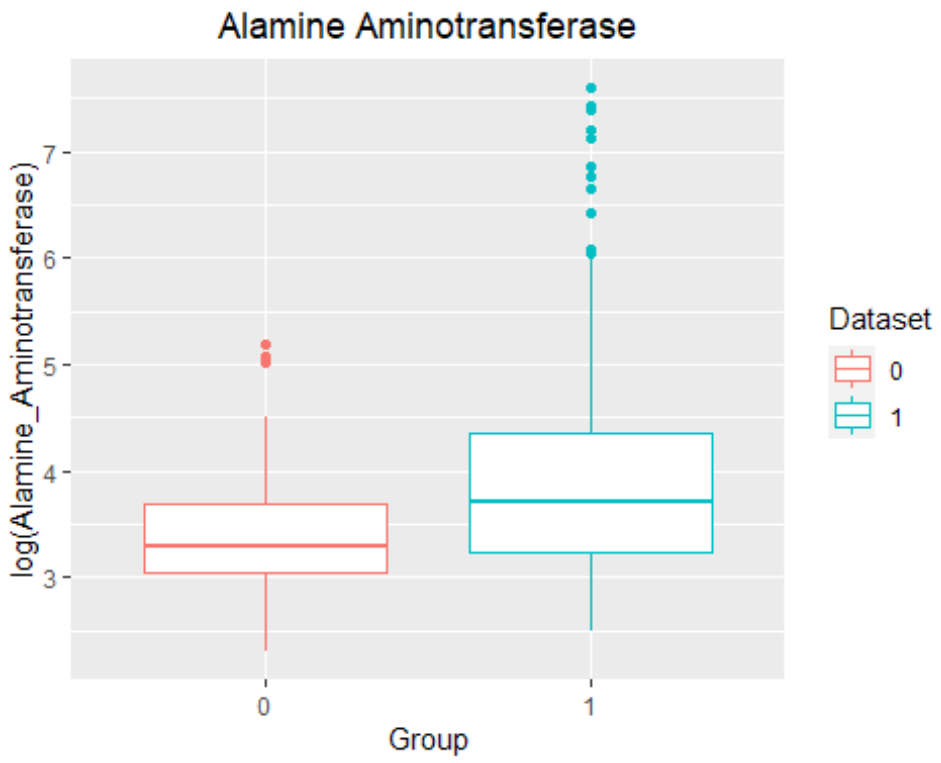
Next, we proceed to the data exploration and visualization of the training data:

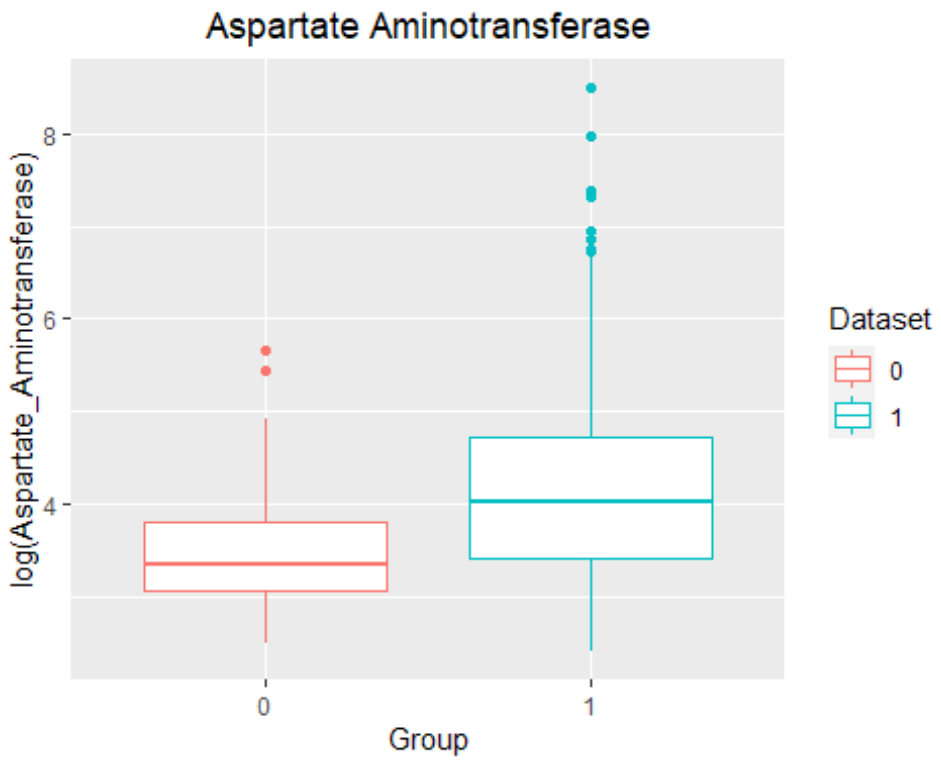

100% stacked barplots for the proportions of sexes for each group
1.00
0.75
0.50
0.25
0.00
Proportion
0
1
Group
Gender
Female
Male

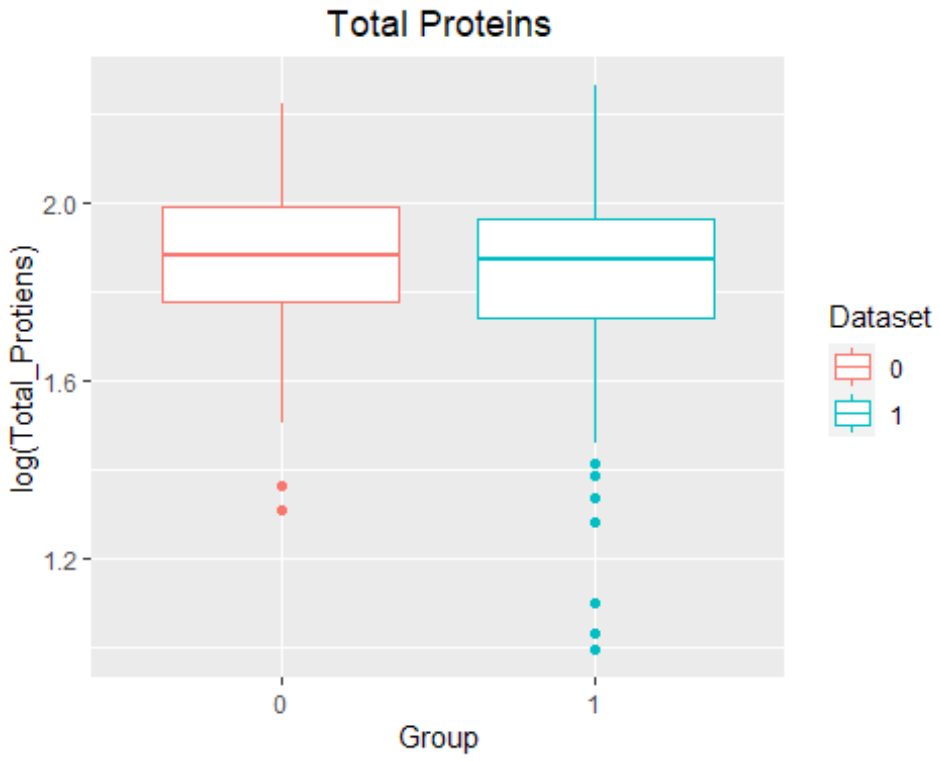

Total Bilirubin
4
3
2
1
0
-1
log(Total_Bilirubin)
0
1
Group
Dataset
0
1

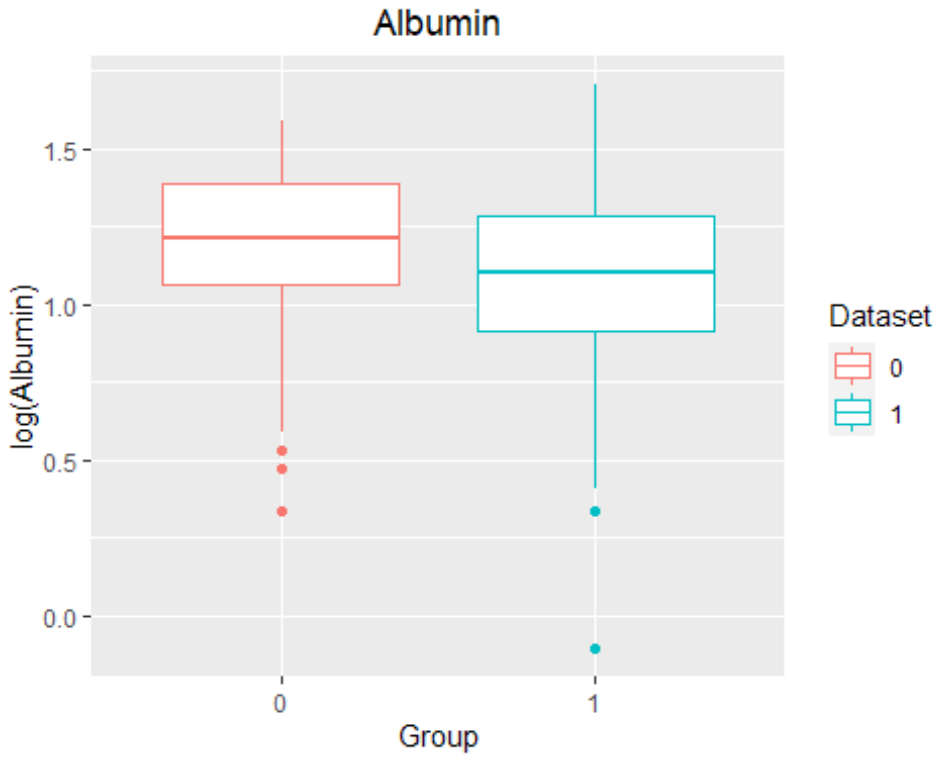

Direct Bilirubin
log(Direct_Bilirubin)
3
2
1
0
-1
-2
0
1
Group
Dataset
0
1

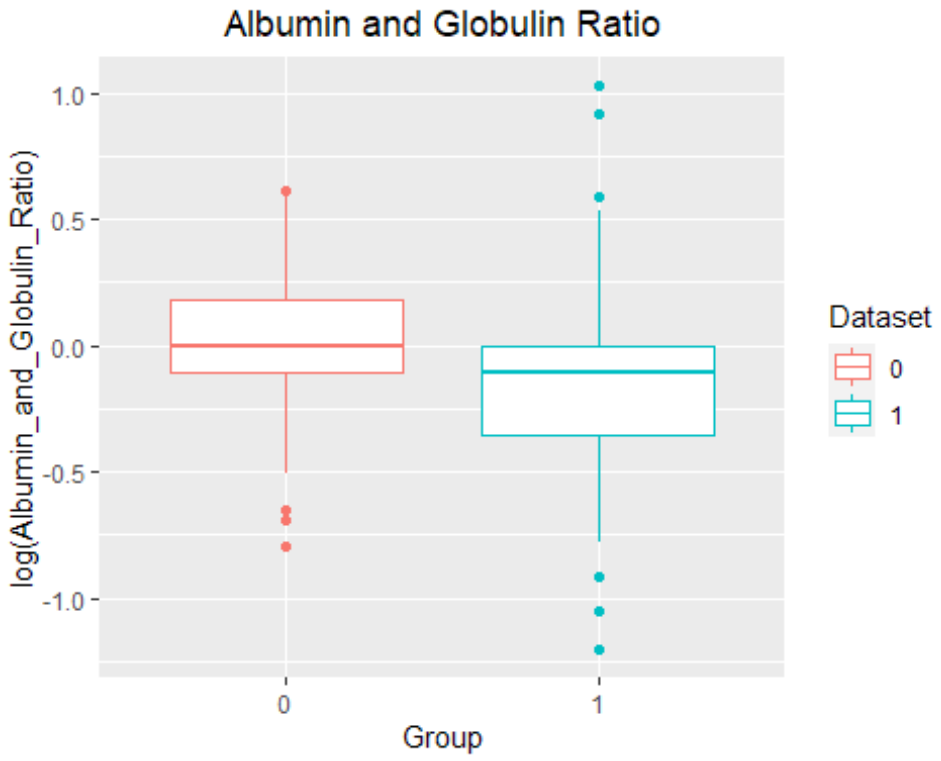

Alkaline Phosphotase
log(Alkaline_Phosphotase)
7
6
5
4
0
1
Group
Dataset
0
1


Alamine Aminotransferase
log(Alamine_Aminotransferase)
7
6
5
4
3
0
1
Group
Dataset
0
1


Aspartate Aminotransferase
log(Aspartate_Aminotransferase)
8
6
4
0
1
Group
Dataset
0
1


Total Proteins
log(Total_Protiens)
2.0
1.6
1.2
0
1
Group
Dataset
0
1


Albumin
log(Albumin)
1.5
1.0
0.5
0.0
0
1
Group
Dataset
0
1



We show these visualizations to have a rough idea about which variables could possibly be the best indicators when it comes to predicting whether a patient has a liver disease or not. Note that the plots for the total biliribun until albumin and globulin ratio are in logarithms for the sake of easier comparison.

It is clear from the graphs that gender, total proteins, and albumin are probably not very effective predictors for liver disease based on this data set. However, we need to dive deeper with more robust analyses.

Logistic regression model building

Thus, we move on to logistic regression. First, we fit a logistic regression model where the dependent variable is the Dataset variable and the independent variables are all the other variables:

```
log_model1 <- glm(Dataset ~ ., data = train, family = 'binomial')

summary(log_model1)

##
## Call:
## glm(formula = Dataset ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2388  -1.0421   0.3400   0.8813   1.5369
##
```

```
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -3.595379   1.657405  -2.169   0.0301 *
## Age                         0.018765   0.007814   2.401   0.0163 *
## GenderMale                  0.016255   0.281248   0.058   0.9539
## Total_Bilirubin             0.024655   0.142984   0.172   0.8631
## Direct_Bilirubin            0.457973   0.330306   1.387   0.1656
## Alkaline_Phosphotase        0.001736   0.001068   1.625   0.1041
## Alamine_Aminotransferase    0.008218   0.005468   1.503   0.1328
## Aspartate_Aminotransferase  0.003125   0.003550   0.880   0.3787
## Total_Protiens              0.998469   0.479963   2.080   0.0375 *
## Albumin                    -1.907660   0.952827  -2.002   0.0453 *
## Albumin_and_Globulin_Ratio  1.968780   1.447784   1.360   0.1739
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 483.40  on 407  degrees of freedom
## Residual deviance: 389.24  on 397  degrees of freedom
## AIC: 411.24
##
## Number of Fisher Scoring iterations: 7
```

However, the variable albumin is clearly related and multicollinear with the variable albumin and globulin ratio. Therefore, we must take out albumin or the albumin and globulin ratio. We take out the albumin and globulin ratio since the information about globulin in the ratio might still be multicollinear with the other variables.

```
log_model2 <- glm(Dataset ~ .-Albumin_and_Globulin_Ratio,
                  data = train,
                  family = 'binomial'
                  )


summary(log_model2)
```

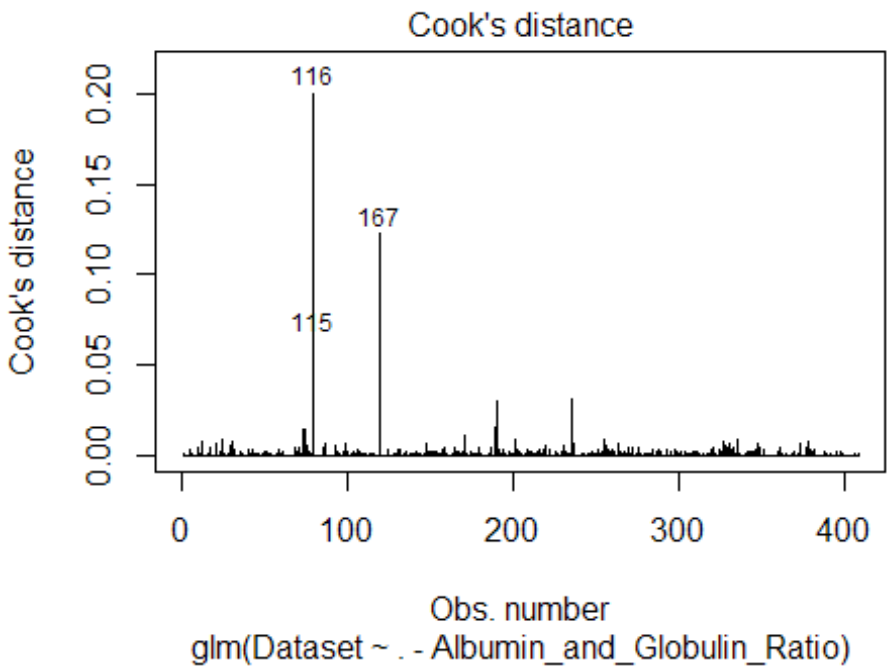
```
##
## Call:
## glm(formula = Dataset ~ . - Albumin_and_Globulin_Ratio, family =
"binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1768  -1.0419   0.3550   0.8672   1.5194
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -1.793320   0.992780  -1.806   0.0709 .
```

```
## Age                          0.018191   0.007814   2.328   0.0199 *
## GenderMale                   0.066653   0.278442   0.239   0.8108
## Total_Bilirubin              0.021069   0.131781   0.160   0.8730
## Direct_Bilirubin             0.474435   0.314365   1.509   0.1313
## Alkaline_Phosphotase         0.001491   0.001003   1.486   0.1373
## Alamine_Aminotransferase     0.007311   0.005386   1.357   0.1746
## Aspartate_Aminotransferase   0.003763   0.003631   1.036   0.3000
## Total_Protiens               0.416065   0.217425   1.914   0.0557 .
## Albumin                     -0.684978   0.314743  -2.176   0.0295 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 483.40  on 407  degrees of freedom
## Residual deviance: 391.14  on 398  degrees of freedom
## AIC: 411.14
##
## Number of Fisher Scoring iterations: 7
```

Outlier detection and removal

Then we check for influential values/outliers that could affect the logistic regression. The rationale behind this step is that influential values/outliers can be detrimental to logistic regression models by making it harder for them to figure out the best fitting thresholds that would lead to the accurate classification of observations according to their independent variables.

```
plot(log_model2, which = 4, id.n = 3)
```




```
model.train <- augment(log_model2) %>%
  mutate(index = 1:n())

# We filter for observations with standard residuals greater than 3
model.train %>%
  filter(abs(.std.resid) > 3)
```

```
## # A tibble: 2 x 19
##   .rownames Dataset   Age Gender Total_Bilirubin Direct_Bilirubin
##   <chr>     <fct>   <int> <chr>            <dbl>            <dbl>
## 1 115       0          50 Male               5.8              3
## 2 116       0          50 Male               7.3              3.6
## # ... with 13 more variables: Alkaline_Phosphotase <int>,
## #   Alamine_Aminotransferase <int>, Aspartate_Aminotransferase <int>,
## #   Total_Protiens <dbl>, Albumin <dbl>, Albumin_and_Globulin_Ratio <dbl>,
## #   .fitted <dbl>, .resid <dbl>, .std.resid <dbl>, .hat <dbl>, .sigma
<dbl>,
## #   .cooksd <dbl>, index <int>
```

Thus, we remove these observations:

```
new_train <- log_model2 %>% augment() %>% filter(abs(.std.resid) <= 3)
new_train <- new_train[, 2:12]

str(new_train)
```

```
## tibble [406 x 11] (S3: tbl_df/tbl/data.frame)
##  $ Dataset                   : Factor w/ 2 levels "0","1": 2 2 2 2 2 1 2 2
2 1 ...
##  $ Age                       : int [1:406] 65 62 62 72 29 17 55 57 72 64
...
##  $ Gender                    : chr [1:406] "Female" "Male" "Male" "Male"
...
##  $ Total_Bilirubin           : num [1:406] 0.7 10.9 7.3 3.9 0.9 0.9 0.7
0.6 2.7 0.9 ...
##  $ Direct_Bilirubin          : num [1:406] 0.1 5.5 4.1 2 0.3 0.3 0.2 0.1
1.3 0.3 ...
##  $ Alkaline_Phosphotase      : int [1:406] 187 699 490 195 202 202 290 210
260 310 ...
##  $ Alamine_Aminotransferase  : int [1:406] 16 64 60 27 14 22 53 51 31 61
...
##  $ Aspartate_Aminotransferase: int [1:406] 18 100 68 59 11 19 58 59 56 58
...
##  $ Total_Protiens            : num [1:406] 6.8 7.5 7 7.3 6.7 7.4 6.8 5.9
7.4 7 ...
##  $ Albumin                   : num [1:406] 3.3 3.2 3.3 2.4 3.6 4.1 3.4 2.7
3 3.4 ...
##  $ Albumin_and_Globulin_Ratio: num [1:406] 0.9 0.74 0.89 0.4 1.1 1.2 1 0.8
0.6 0.9 ...
##  - attr(*, "terms")=Classes 'terms', 'formula'  language Dataset ~ (Age +
Gender + Total_Bilirubin + Direct_Bilirubin + Alkaline_Phosphotase +
Alamine_Aminotransfera| __truncated__ ...
##   .. ..- attr(*, "variables")= language list(Dataset, Age, Gender,
Total_Bilirubin, Direct_Bilirubin, Alkaline_Phosphotase,
Alamine_Aminotransferase| __truncated__ ...
##   .. ..- attr(*, "factors")= int [1:11, 1:9] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:11] "Dataset" "Age" "Gender" "Total_Bilirubin" ...
##   .. .. .. ..$ : chr [1:9] "Age" "Gender" "Total_Bilirubin"
"Direct_Bilirubin" ...
##   .. ..- attr(*, "term.labels")= chr [1:9] "Age" "Gender"
"Total_Bilirubin" "Direct_Bilirubin" ...
##   .. ..- attr(*, "order")= int [1:9] 1 1 1 1 1 1 1 1 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(Dataset, Age, Gender,
Total_Bilirubin, Direct_Bilirubin, Alkaline_Phosphotase,
Alamine_Aminotransferase| __truncated__ ...
##   .. ..- attr(*, "dataClasses")= Named chr [1:11] "factor" "numeric"
"character" "numeric" ...
##   .. .. ..- attr(*, "names")= chr [1:11] "Dataset" "Age" "Gender"
"Total_Bilirubin" ...
```

As can be seen, the two outliers were removed since the original training data had 408 rows while the new training data now has 406 rows. We will check that the two rows that were removed were truly the outliers later on.

Using training data with no outliers

Now, we continue on to logistic regression with albumin and globulin ratio removed but, presumably, with no outliers:

```
log_model2b <- glm(Dataset ~ .-Albumin_and_Globulin_Ratio,
                   data = new_train,
                   family = 'binomial'
                   )

summary(log_model2b)

##
## Call:
## glm(formula = Dataset ~ . - Albumin_and_Globulin_Ratio, family =
"binomial",
##     data = new_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1869  -0.9902   0.2651   0.8611   1.6555
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -2.366606   1.052763  -2.248   0.0246 *
## Age                         0.019917   0.008094   2.461   0.0139 *
## GenderMale                  0.016141   0.288427   0.056   0.9554
## Total_Bilirubin             0.013512   0.171715   0.079   0.9373
## Direct_Bilirubin            0.714380   0.423330   1.688   0.0915 .
## Alkaline_Phosphotase        0.004145   0.001637   2.532   0.0113 *
## Alamine_Aminotransferase    0.008184   0.006028   1.358   0.1746
## Aspartate_Aminotransferase  0.003854   0.004065   0.948   0.3431
## Total_Protiens              0.368136   0.232213   1.585   0.1129
## Albumin                    -0.631716   0.334723  -1.887   0.0591 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 478.27  on 405  degrees of freedom
## Residual deviance: 365.95  on 396  degrees of freedom
## AIC: 385.95
##
## Number of Fisher Scoring iterations: 8
```

We check for outliers again and find that there are none now.

```
log_model2b %>% augment() %>% filter(abs(.std.resid) > 3)
```

```
## # A tibble: 0 x 17
## # ... with 17 variables: Dataset <fct>, Age <int>, Gender <chr>,
## #   Total_Bilirubin <dbl>, Direct_Bilirubin <dbl>, Alkaline_Phosphotase
<int>,
## #   Alamine_Aminotransferase <int>, Aspartate_Aminotransferase <int>,
## #   Total_Protiens <dbl>, Albumin <dbl>, Albumin_and_Globulin_Ratio <dbl>,
## #   .fitted <dbl>, .resid <dbl>, .std.resid <dbl>, .hat <dbl>, .sigma
<dbl>,
## #   .cooksd <dbl>
```

Backward elimination

Moving on, we start backward elimination and take out the variable Gender as it is currently the most insignificant variable since it has the highest p-value (using alpha = 0.05). We use the usual alpha of 0.05 since the focus of this study is not strict hypothesis testing and figuring out which variables are statistically significant with respect to affecting liver disease onset. The focus instead is providing a model or models that would accurately predict the onset of liver disease.

The following is the code to execute the backward elimination of the logistic regression models until the final logistic regression model where all variables are statistically significant, using p-values, with an alpha of 0.05.

We first remove the Gender variable:

```
log_model2c <- glm(Dataset ~ .-Albumin_and_Globulin_Ratio -Gender,
                   data = new_train,
                   family = 'binomial'
                   )

summary(log_model2c)
```

```
##
## Call:
## glm(formula = Dataset ~ . - Albumin_and_Globulin_Ratio - Gender,
##     family = "binomial", data = new_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1867  -0.9946   0.2653   0.8567   1.6563
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -2.350575   1.013122  -2.320   0.0203 *
## Age                         0.019911   0.008093   2.460   0.0139 *
## Total_Bilirubin             0.013570   0.172183   0.079   0.9372
## Direct_Bilirubin            0.716288   0.422805   1.694   0.0902 .
## Alkaline_Phosphotase        0.004150   0.001636   2.537   0.0112 *
## Alamine_Aminotransferase    0.008205   0.006022   1.362   0.1731
```

```
## Aspartate_Aminotransferase  0.003859   0.004067   0.949   0.3427
## Total_Protiens               0.366292   0.229856   1.594   0.1110
## Albumin                     -0.630210   0.333599  -1.889   0.0589 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 478.27  on 405  degrees of freedom
## Residual deviance: 365.95  on 397  degrees of freedom
## AIC: 383.95
##
## Number of Fisher Scoring iterations: 8
```

We take out the variable Total Bilirubin next:

```
log_model2d <- glm(Dataset ~ .-Albumin_and_Globulin_Ratio -Gender -
Total_Bilirubin,
                   data = new_train,
                   family = 'binomial'
                   )

summary(log_model2d)

##
## Call:
## glm(formula = Dataset ~ . - Albumin_and_Globulin_Ratio - Gender -
##     Total_Bilirubin, family = "binomial", data = new_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1871  -0.9948   0.2634   0.8567   1.6560
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -2.344001   1.009159  -2.323  0.02019 *
## Age                         0.019910   0.008093   2.460  0.01389 *
## Direct_Bilirubin            0.742433   0.275534   2.695  0.00705 **
## Alkaline_Phosphotase        0.004154   0.001636   2.540  0.01110 *
## Alamine_Aminotransferase    0.008205   0.006024   1.362  0.17320
## Aspartate_Aminotransferase  0.003859   0.004068   0.949  0.34283
## Total_Protiens              0.365740   0.229789   1.592  0.11147
## Albumin                    -0.629809   0.333638  -1.888  0.05907 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 478.27  on 405  degrees of freedom
## Residual deviance: 365.96  on 398  degrees of freedom
```

```
## AIC: 381.96
##
## Number of Fisher Scoring iterations: 8
```

We take out the variable Aspartate Aminotransferase next:

```
log_model2e <- glm(Dataset ~ .-Albumin_and_Globulin_Ratio -Total_Bilirubin -
Gender -
                     Aspartate_Aminotransferase,
                   data = new_train,
                   family = 'binomial'
                   )

summary(log_model2e)

##
## Call:
## glm(formula = Dataset ~ . - Albumin_and_Globulin_Ratio - Total_Bilirubin -
##     Gender - Aspartate_Aminotransferase, family = "binomial",
##     data = new_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2384  -1.0043   0.2578   0.8620   1.6402
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -2.260797   1.000634  -2.259  0.02386 *
## Age                        0.019516   0.008061   2.421  0.01548 *
## Direct_Bilirubin           0.787424   0.273888   2.875  0.00404 **
## Alkaline_Phosphotase       0.004174   0.001645   2.537  0.01117 *
## Alamine_Aminotransferase   0.012461   0.004671   2.668  0.00763 **
## Total_Protiens             0.359528   0.229919   1.564  0.11789
## Albumin                   -0.638305   0.333418  -1.914  0.05557 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 478.27  on 405  degrees of freedom
## Residual deviance: 366.98  on 399  degrees of freedom
## AIC: 380.98
##
## Number of Fisher Scoring iterations: 8
```

We take out the variable Total Protiens next:

```
log_model2f <- glm(Dataset ~ .-Albumin_and_Globulin_Ratio -Total_Bilirubin -
Gender -
                     Aspartate_Aminotransferase -Total_Protiens,
                   data = new_train,
```

```
                  family = 'binomial'
                  )

summary(log_model2f)

##
## Call:
## glm(formula = Dataset ~ . - Albumin_and_Globulin_Ratio - Total_Bilirubin -
##     Gender - Aspartate_Aminotransferase - Total_Protiens, family =
"binomial",
##     data = new_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1898  -1.0300   0.2605   0.8782   1.6015
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.495506   0.866511  -1.726  0.08437 .
## Age                       0.019985   0.008025   2.491  0.01276 *
## Direct_Bilirubin          0.843711   0.272624   3.095  0.00197 **
## Alkaline_Phosphotase      0.004668   0.001673   2.790  0.00527 **
## Alamine_Aminotransferase  0.011594   0.004500   2.576  0.00999 **
## Albumin                  -0.190599   0.168712  -1.130  0.25859
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 478.27  on 405  degrees of freedom
## Residual deviance: 369.45  on 400  degrees of freedom
## AIC: 381.45
##
## Number of Fisher Scoring iterations: 7
```

Finally, we take out the variable Albumin and construct the final logistic regression model:

Final logistic regression model

```
log_model2g <- glm(Dataset ~ .-Albumin_and_Globulin_Ratio -Total_Bilirubin -
Gender -
                     Aspartate_Aminotransferase -Total_Protiens -Albumin,
                   data = new_train,
                   family = 'binomial'
                   )

summary(log_model2g)

##
## Call:
## glm(formula = Dataset ~ . - Albumin_and_Globulin_Ratio - Total_Bilirubin -
```

```
##     Gender - Aspartate_Aminotransferase - Total_Protiens - Albumin,
##     family = "binomial", data = new_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2528  -1.0119   0.2598   0.8865   1.5740
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -2.263842   0.545475  -4.150 3.32e-05 ***
## Age                       0.022374   0.007745   2.889 0.003868 **
## Direct_Bilirubin          0.885325   0.267627   3.308 0.000939 ***
## Alkaline_Phosphotase      0.004753   0.001665   2.856 0.004296 **
## Alamine_Aminotransferase  0.011611   0.004492   2.584 0.009754 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 478.27  on 405  degrees of freedom
## Residual deviance: 370.73  on 401  degrees of freedom
## AIC: 380.73
##
## Number of Fisher Scoring iterations: 7
```

Observe that the final logistic regression model has the lowest AIC value, and all of its variables are statistically significant which make it the most preferred model out of all the presented logistic regression models.

Model Evaluation

Now that the logistic regression model has been finalized, we evaluate it based on its pseudo r-squared value (McFadden's score), check the importance of each variable, and check for multicollinearity between variables.

The last two relate more to satisfying the assumptions of logistic regression while checking the pseudo r-squared value of the model will give us an idea of how much the model is able to explain whether a patient has liver disease or not.

```
# McFadden's of the final model
pR2(log_model2g)["McFadden"]
```

```
## fitting null model for pseudo-r2
```

```
##  McFadden
## 0.2248602
```

```
# Variable importance
varImp(log_model2g)
```

```
##                            Overall
## Age                       2.888705
## Direct_Bilirubin          3.308061
## Alkaline_Phosphotase      2.855533
## Alamine_Aminotransferase 2.584428
```

```
# Variance inflation factors
vif(log_model2g)
```

```
##                       Age         Direct_Bilirubin     Alkaline_Phosphotase
##                  1.031117                 1.028128                 1.052662
## Alamine_Aminotransferase
##                  1.066873
```

Note that none of the VIF values are above 5 (which would indicate multicollinearity), and that the McFadden score for the final logistic regression model is 0.2248602.

Test data set prediction

Then, we predict the presence of liver disease for the patients in the test data set (using the usual probability threshold of 0.5) and also construct its coresponding confusion matrix:

```
# Predict test data based on final logistic regression model
predict_reg <- predict(log_model2g,
                       test,
                       type = "link"
                       )

# Changing probabilities
predict_reg <- ifelse(predict_reg > 0.5, 1, 0)

# Constructing a confusion matrix and making predict_reg into factors
predict_reg <- as.factor(predict_reg)

logreg_conmatrix <- confusionMatrix(data = predict_reg,
                                    reference = test$Dataset,
                                    positive = "1"
                                    )

# Logistic regression confusion matrix
logreg_conmatrix
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  0  1
##          0 33 44
##          1 18 76
##
##                Accuracy : 0.6374
##                  95% CI : (0.5606, 0.7094)
```

```
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 0.971025
##
##                   Kappa : 0.2445
##
##  Mcnemar's Test P-Value : 0.001498
##
##             Sensitivity : 0.6333
##             Specificity : 0.6471
##          Pos Pred Value : 0.8085
##          Neg Pred Value : 0.4286
##              Prevalence : 0.7018
##          Detection Rate : 0.4444
##    Detection Prevalence : 0.5497
##       Balanced Accuracy : 0.6402
##
##        'Positive' Class : 1
##
```

Now, we can see that the final logistic regression model, with the default threshold, has an accuracy of 0.6374 which is actually lower than the no information rate (NIR) of 0.7018 which is just the proportion of those who have liver disease in the testing data set. The NIR is also a measure of how often one would be correct if they just blindly assumed that any patient they come across has liver disease. As can be seen, this logistic regression model does not do so well since its accuracy is lower than its NIR.

Thus, to remedy this, we will tweak the probability threshold for classification of this final logistic regression model. To do this, we will define a function that will take a threshold level as input and have the corresponding confusion matrix as its output:

```
# Defining a function that takes threshold level as input and have the
corresponding confusion matrix as its output
conf_matrix <- function(threshold){
  predict_reg <- as.numeric(predict_reg)
  predict_reg <- predict(log_model2g,
                         test,
                         type = "link")
  predict_reg <- ifelse(predict_reg > threshold, 1, 0)
  predict_reg <- as.factor(predict_reg)
  logreg_conmatrix <- confusionMatrix(data = predict_reg,
                                      reference = test$Dataset,
                                      positive = "1")
  print(logreg_conmatrix)
}
```

Then we use this function for multiple values of the probability threshold.

```
sapply(c(0.5, 0.005, 0.0005, 0.00005), conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 33 44
##          1 18 76
##
##                Accuracy : 0.6374
##                  95% CI : (0.5606, 0.7094)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 0.971025
##
##                   Kappa : 0.2445
##
##  Mcnemar's Test P-Value : 0.001498
##
##             Sensitivity : 0.6333
##             Specificity : 0.6471
##          Pos Pred Value : 0.8085
##          Neg Pred Value : 0.4286
##              Prevalence : 0.7018
##          Detection Rate : 0.4444
##    Detection Prevalence : 0.5497
##       Balanced Accuracy : 0.6402
##
##        'Positive' Class : 1
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  19  19
##          1  32 101
##
##                Accuracy : 0.7018
##                  95% CI : (0.6272, 0.7692)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 0.53773
##
##                   Kappa : 0.2312
##
##  Mcnemar's Test P-Value : 0.09289
##
##             Sensitivity : 0.8417
##             Specificity : 0.3725
##          Pos Pred Value : 0.7594
##          Neg Pred Value : 0.5000
##              Prevalence : 0.7018
##          Detection Rate : 0.5906
##    Detection Prevalence : 0.7778
```

```
##       Balanced Accuracy : 0.6071
##
##        'Positive' Class : 1
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  19  18
##          1  32 102
##
##                Accuracy : 0.7076
##                  95% CI : (0.6333, 0.7745)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 0.47117
##
##                   Kappa : 0.2416
##
##  Mcnemar's Test P-Value : 0.06599
##
##             Sensitivity : 0.8500
##             Specificity : 0.3725
##          Pos Pred Value : 0.7612
##          Neg Pred Value : 0.5135
##              Prevalence : 0.7018
##          Detection Rate : 0.5965
##    Detection Prevalence : 0.7836
##       Balanced Accuracy : 0.6113
##
##        'Positive' Class : 1
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  19  18
##          1  32 102
##
##                Accuracy : 0.7076
##                  95% CI : (0.6333, 0.7745)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 0.47117
##
##                   Kappa : 0.2416
##
##  Mcnemar's Test P-Value : 0.06599
##
##             Sensitivity : 0.8500
##             Specificity : 0.3725
##          Pos Pred Value : 0.7612
```

```
##          Neg Pred Value : 0.5135
##              Prevalence : 0.7018
##          Detection Rate : 0.5965
##    Detection Prevalence : 0.7836
##       Balanced Accuracy : 0.6113
##
##        'Positive' Class : 1
##

##          [,1]        [,2]        [,3]        [,4]
## positive "1"         "1"         "1"         "1"
## table    table,4     table,4     table,4     table,4
## overall  numeric,7   numeric,7   numeric,7   numeric,7
## byClass  numeric,8   numeric,8   numeric,8   numeric,8
## mode     "sens_spec" "sens_spec" "sens_spec" "sens_spec"
## dots     list,0      list,0      list,0      list,0
```

As can be seen from the confusion matrices, the accuracy and sensitivity stop improving after a threshold of 0.0005. Thus, we choose a threshold of 0.0005 for our final logistic regression model. We focus on accuracy and sensitivity since it is desirable to maximize them in this case. It is obvious why we would want to maximize accuracy, but the reason why we want to also maximize sensitivity is because it is defined as the probability of a positive test conditioned on truly being positive.

We obviously want to maximize this probability because it is of the utmost importance to detect the presence of a patient's liver disease to mitigate the harm caused by said disease, even if doing so would mean an increase in the number of false positives (as evidenced by the confusion matrices).

With this final logistic regression model with a threshold of 0.0005 and a sensitivity of 0.85, we have an 85% probability of classifying a patient as having liver disease, given that they truly have a liver disease. This logistic regression model also has an accuracy of 0.7076.

However, and as expected, the specificity or the probability of achieving a negative test given that a patient is truly negative is quite low, which means that false positives will happen quite often. But producing a false positive is preferable to obtaining a false negative since producing a false negative would mean a patient would not receive any treatment for their liver disease.

We now specify the final logistic regression model to have a threshold of 0.0005:

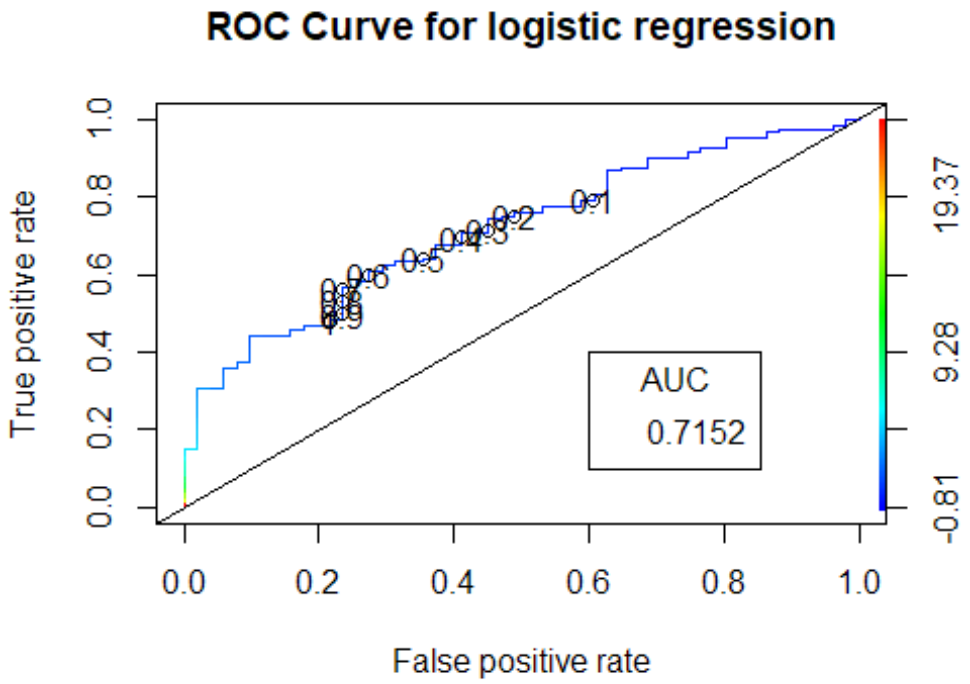
```
predict_reg <- predict(log_model2g,
                       test,
                       type = "link"
                       )

predict_reg <- ifelse(predict_reg > 0.0005, 1, 0)

predict_reg <- as.factor(predict_reg)
```

```
logreg_conmatrix <- confusionMatrix(data = predict_reg,
                                    reference = test$Dataset,
                                    positive = "1"
                                    )
```

ROC Curve

Finally, we will plot an ROC curve along with its area under the curve to show the overview and overall performance of the final logistic regression model.

```
# ROC-AUC Curve
ROCPred <- prediction(predict(log_model2g, test), test$Dataset)
ROCPer <- performance(ROCPred, measure = "tpr",
                      x.measure = "fpr"
                      )

auc <- performance(ROCPred, measure = "auc")
auc <- auc@y.values[[1]]
auc

## [1] 0.7151961

# Plotting curve
par(mfrow = c(1, 1))

plot(ROCPer, colorize = TRUE,
     print.cutoffs.at = seq(0.1, by = 0.1),
     main = "ROC Curve for logistic regression")
abline(a = 0, b = 1)

auc <- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```



Thus, the AUC of the final logistic regression model is 0.7152.

XGBoost models

To compare and contrast with the logistic regression model, we will be applying XGBoost models with the same data used by the logistic regression models.

Model building

```
xgb_train <- as.data.frame(new_train)

# We change the gender character strings into 1 for Female, 0
# for male, and make both of them into numeric variables

xgb_train$Gender[xgb_train$Gender == "Female"] <- 1
xgb_train$Gender[xgb_train$Gender == "Male"] <- 0
xgb_train$Gender <- as.numeric(xgb_train$Gender)

# We specify reasonable options for hyperparameters that will
# be used to check for the best XGBoost model

grid_tune <- expand.grid(nrounds = c(100, 400, 600),
                         max_depth = c(4, 6, 8, 10),
                         eta = c(0.1, 0.3, 0.5),
                         gamma = c(0, 1, 2),
                         colsample_bytree = c(1, 0.7, 0.5),
                         min_child_weight = c(1, 2, 3),
                         subsample = c(0.5, 0.75, 1)
```

```
                          )

# We set a three-fold cross validation (due to the computational
# resources that grid_tune is already taking)

train_control <- trainControl(method = "cv",
                              number= 3,
                              verboseIter = FALSE,
                              allowParallel = TRUE
                              )

# We fit the XGBoost model with the presence of liver disease
# being the y variable and all other variables as the x variables

xgb_model <- train(x = xgb_train[, -1],
                   y = xgb_train$Dataset,
                   trControl = train_control,
                   tuneGrid = grid_tune,
                   method= "xgbTree",
                   verbose = FALSE,
                   verbosity = 0
                   )

xgb_model$bestTune
```

```
##      nrounds max_depth eta gamma colsample_bytree min_child_weight
subsample
## 2161     100         4 0.5     2                1                1
0.5
```

As can be seen from the results, the best performing XGBoost model has parameters nrounds = 100, max_depth = 4, eta = 0.5, gamma = 2, colsample_bytree = 1, min_child_weight = 1 and subsample = 0.5. This best performing model will be the final XGBoost model.

XGBoost model evaluation

Now, we move on to evaluating the performance of the final XGBoost model using its confusion matrix and ROC curve.

```
xgb_test <- test
xgb_test$Gender[xgb_test$Gender == "Female"] <- 1
xgb_test$Gender[xgb_test$Gender == "Male"] <- 0
xgb_test$Gender <- as.numeric(xgb_test$Gender)

xgb_pred <- predict(xgb_model, xgb_test)

# Confusion matrix for the final XGBoost model
xgb_conmatrix <- confusionMatrix(data = xgb_pred,
                                 reference = xgb_test$Dataset,
```

```
                                  positive = "1"
                                  )

xgb_conmatrix
```

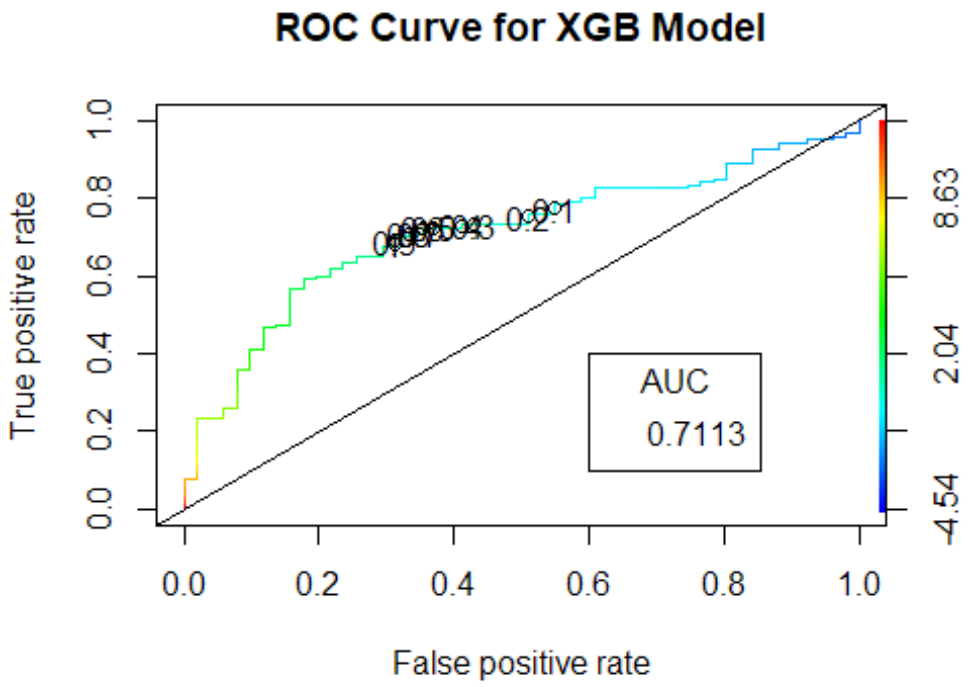
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23 27
##          1 28 93
##
##                Accuracy : 0.6784
##                  95% CI : (0.6028, 0.7476)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 0.7755
##
##                   Kappa : 0.2273
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.7750
##             Specificity : 0.4510
##          Pos Pred Value : 0.7686
##          Neg Pred Value : 0.4600
##              Prevalence : 0.7018
##          Detection Rate : 0.5439
##    Detection Prevalence : 0.7076
##       Balanced Accuracy : 0.6130
##
##        'Positive' Class : 1
##
```

```
# ROC-AUC Curve

xgb_prob <- predict(xgb_model, xgb_test, type = "prob")
log_odds_xgb_prob <- log(xgb_prob[, 2]/xgb_prob[ , 1])

xgb_ROCPred <- prediction(log_odds_xgb_prob, xgb_test$Dataset)
xgb_ROCPer <- performance(xgb_ROCPred, measure = "tpr",
                      x.measure = "fpr")

xgb_auc <- performance(xgb_ROCPred, measure = "auc")
xgb_auc <- xgb_auc@y.values[[1]]
xgb_auc
```

```
## [1] 0.7112745
```

```
# Plotting curve
plot(xgb_ROCPer, colorize = TRUE,
```

```
     print.cutoffs.at = seq(0.1, by = 0.1),
     main = "ROC Curve for XGB Model"
     )

abline(a = 0, b = 1)

xgb_auc <- round(xgb_auc, 4)
legend(.6, .4, xgb_auc, title = "AUC", cex = 1)
```



Observe that for the final model: its accuracy is 0.6784, its sensitivity is 0.7750, and its AUC is 0.7113.

With all of these results, we will now move on to the results section which compares and contrasts the final logistic regression model and the final XGBoost model.

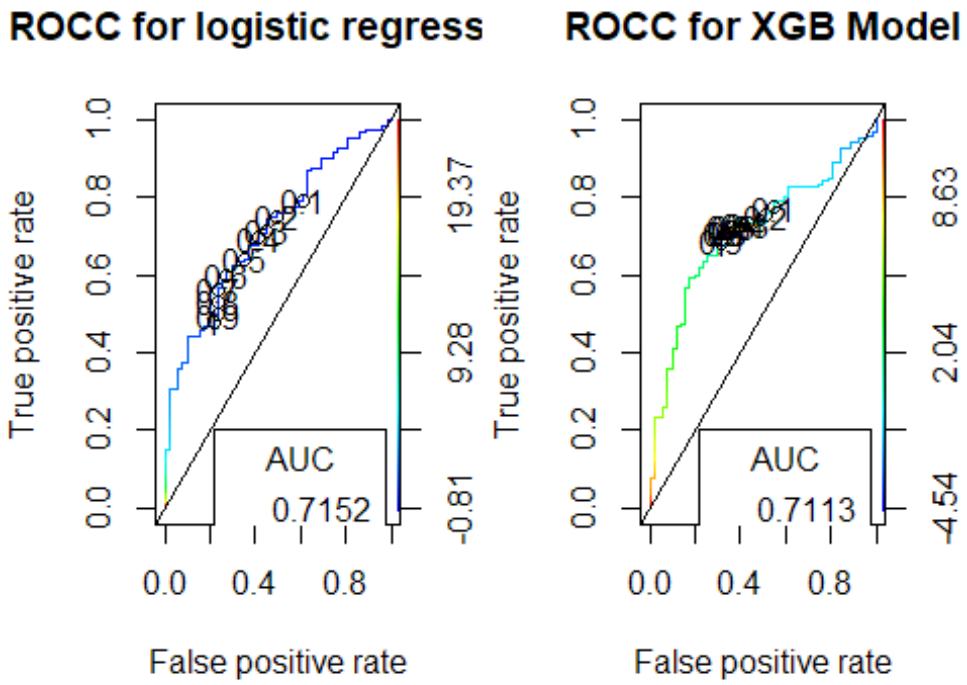
Results

To facilitate the discussion of the results of both models, we will refer to the past confusion matrices and ROC curves of both models.

Let us focus on the ROC curve for each model first:

```
# We compare the ROC-AUC Curve for each model
par(mfrow = c(1, 2))

# Logistic regression
```

```
plot(ROCPer, colorize = TRUE,
     print.cutoffs.at = seq(0.1, by = 0.1),
     main = "ROCC for logistic regression"
     )

abline(a = 0, b = 1)

auc <- round(auc, 4)
legend(.22, .2, auc, title = "AUC", cex = 1)

# XGB Model

plot(xgb_ROCPer, colorize = TRUE,
     print.cutoffs.at = seq(0.1, by = 0.1),
     main = "ROCC for XGB Model"
     )

abline(a = 0, b = 1)

xgb_auc <- round(xgb_auc, 4)
legend(.22, .2, xgb_auc, title = "AUC", cex = 1)
```



As we can see from the plots, the final logistic regression model actually has a slightly higher area under the curve (AUC) value than the final XGBoost model. This implies that the logistic regression model actually performs better than the XGBoost model, since the

former is able to attain a higher true positive rate or sensitivity without having to increase the false positive rate as much as the latter.

Now, we move on to the confusion matrix for each model.

```
# Logistic Regression
logreg_conmatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  19  18
##          1  32 102
##
##                Accuracy : 0.7076
##                  95% CI : (0.6333, 0.7745)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 0.47117
##
##                   Kappa : 0.2416
##
##  Mcnemar's Test P-Value : 0.06599
##
##             Sensitivity : 0.8500
##             Specificity : 0.3725
##          Pos Pred Value : 0.7612
##          Neg Pred Value : 0.5135
##              Prevalence : 0.7018
##          Detection Rate : 0.5965
##    Detection Prevalence : 0.7836
##       Balanced Accuracy : 0.6113
##
##        'Positive' Class : 1
##
```

```
# XGB Model
xgb_conmatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23 27
##          1 28 93
##
##                Accuracy : 0.6784
##                  95% CI : (0.6028, 0.7476)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 0.7755
##
```

```
##                   Kappa : 0.2273
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.7750
##             Specificity : 0.4510
##          Pos Pred Value : 0.7686
##          Neg Pred Value : 0.4600
##              Prevalence : 0.7018
##          Detection Rate : 0.5439
##    Detection Prevalence : 0.7076
##       Balanced Accuracy : 0.6130
##
##        'Positive' Class : 1
##
```

The final logistic regression model has an accuracy of 0.7076 and a sensitivity of 0.85, while the final XGBoost model has an accuracy of 0.6784 and a sensitivity of 0.7750.

It is important to note that tweaking the threshold as was done earlier could lead to a lack of informedness which is equal to specificity + sensitivity - 1. If the informedness score is < 0 for a model, then it means that the use of that model represents a perverse use of information and that it is very unlikely for a person to make an informed decision between two classes.

Fortunately, for both the logistic regression model and the XGBoost model, their informedness scores are greater than 0. More specifically, the informedness score for logistic regression is (0.8500 + 0.3725 - 1) = 0.2225, while the informedness score of the XGBoost model is (0.7500 + 0.4510 - 1) = 0.2010.

From what can be gleaned from the results, it is clear that that the logistic regression model does a bit better than the XGBoost model when it comes to the accuracies, sensitivities, AUCs, and informedness scores of the two models even with parameter tuning on the XGBoost models.

Therefore, not only is the final logistic regression model developed in this study able to accurately predict the onset of liver disease 85% of the time (if the patient actually has a liver disease) but said model is also preferable to the final XGBoost model in this study when it comes to detecting whether a patient has liver disease or not.

However, it must be noted that, on one hand, the logistic regression model needed more refinement than the XGBoost model in that backward elimination, multicollinearity checks, and outlier detection and removal had to be applied. On the other hand, the XGBoost model just had to be tuned and the same outlier detection and removal that was applied to logistic regression, was also applied to the XGBoost model.

Conclusion

To conclude, we ran two different types of models: Logistic Regression and XGBoost, in order to develop a model or models which would accurately predict the onset of liver disease in patients. While both types of models performed satisfactorily, after some modifications, the final logistic regession model with a threshold of 0.0005 was chosen to be the best model overall due to having the highest accuracy, sensitivity, and AUC.

While it was possible to produce models which could accurately predict liver disease, if a patient truly had liver disease, it was not possible with the current data and variables to make a model with either logistic regression or XGBoost that would be able to elicit both high sensitivity and specificity.

However, this was not a huge obstacle since, for liver disease patients, we want to maximize the probability of predicting a patient who has liver disease if they actually have liver disease, for the sake of early treatment and prevention of further harm to liver disease patients.

The potential impact of this chosen best model, the final logistic regression model, is that such a model could be used by hospitals who need to know how likely it is that a patient has liver disease without having to resort to expensive and invasive surgery.

The limitations of this model is that, again, it does not have a very high specificity and thus, it does not predict the onset of liver disease very accurately if a patient is not suffering from liver disease. Furthermore, there are only a few hundred observations and only a handful of variables. Both of these facts limit how much can be used to accurately predict the presence of liver disease in patients.

For further work, it would be prudent to collect more data from different countries, and to also include more variables that could possibly help shed light when it comes to the nature of liver disease onset.

Resources:

https://www.geeksforgeeks.org/logistic-regression-in-r-programming/
https://www.youtube.com/watch?v=qjeUhuvkbHY&t=723s