Predicting Liver Disease

Executive Summary

The dataset was downloaded from the UCI ML Repository:Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. University of California, School of Information and Computer Science. The data set can also be found on my Github account https://github.com/huraaa/Liver_Disease/

Objective: **Given the set of variables predict if a patient has liver disease or not**

This dataset contains 416 liver patient records and 167 non liver patient records. The data set was collected from north east of Andhra Pradesh, India.The "Dataset" column is a class label used to divide groups into liver patient (liver disease) or not (no disease). 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".

We used various models with varing accuracy as mentioned below

- Classification and Regression Trees ~ 68.97% Accuracy
- Random Forest (Original Data) ~ 69.8% Accuracy
- Random Forest (Categorised variabes) ~ 71.55% Accuracy
- XGBoost ~ 88.5% Accuracy

It is important to note that in real world, specially in medical world, accuracy is not the only metric of importance. What is more important is to minimize False Negatives in the model. Meaning if we diagnose a patient as negative there should be minimal chance that the patient was in fact positive. This ill diagnosis can prove a costly mistake and cost lives. However, in this paper, given the limited set of variable and observations, we only target accuracy as our main metric for model performance.

Exploratory Data Analysis

In the first step I ran a summary function to give the overview of the dataset. We observe that only 1 column has missing values and the rest are polulated. We impute the missing values in Albumin_and_Globulin_Ratio by replacing them with the mean of the column. In the next step involves converting the numerical values to factor format so that they are easily read by the models employed.

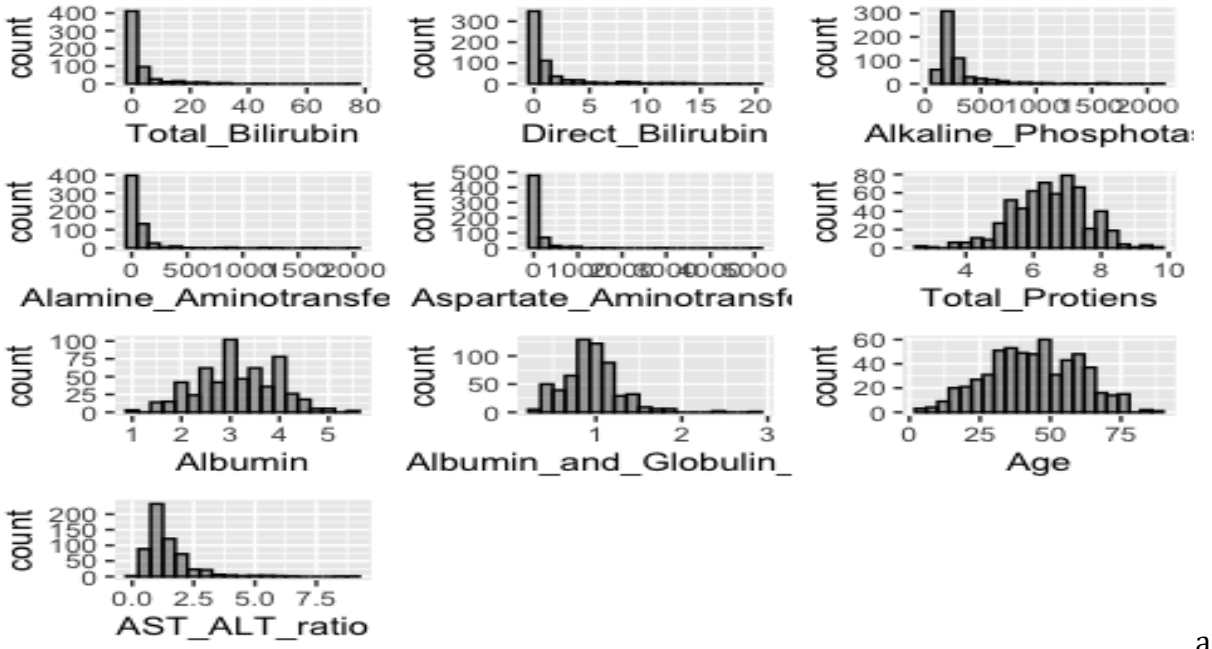
I create a new variable AST_ALT_ratio, since the ratio of AST (Aspartate_Aminotransferase) to ALT (Alamine_Aminotransferase) is used as a clinical parameter for liver disease condition and is considered very important by the medical community.

In order to get a better understanding of how different variables change for patients with liver disease, I broke down target variable into liver_disease and no_liver_disease. Summary function breaks down the data into quintiles and is useful to determine outliers.

```
summary(data)
```

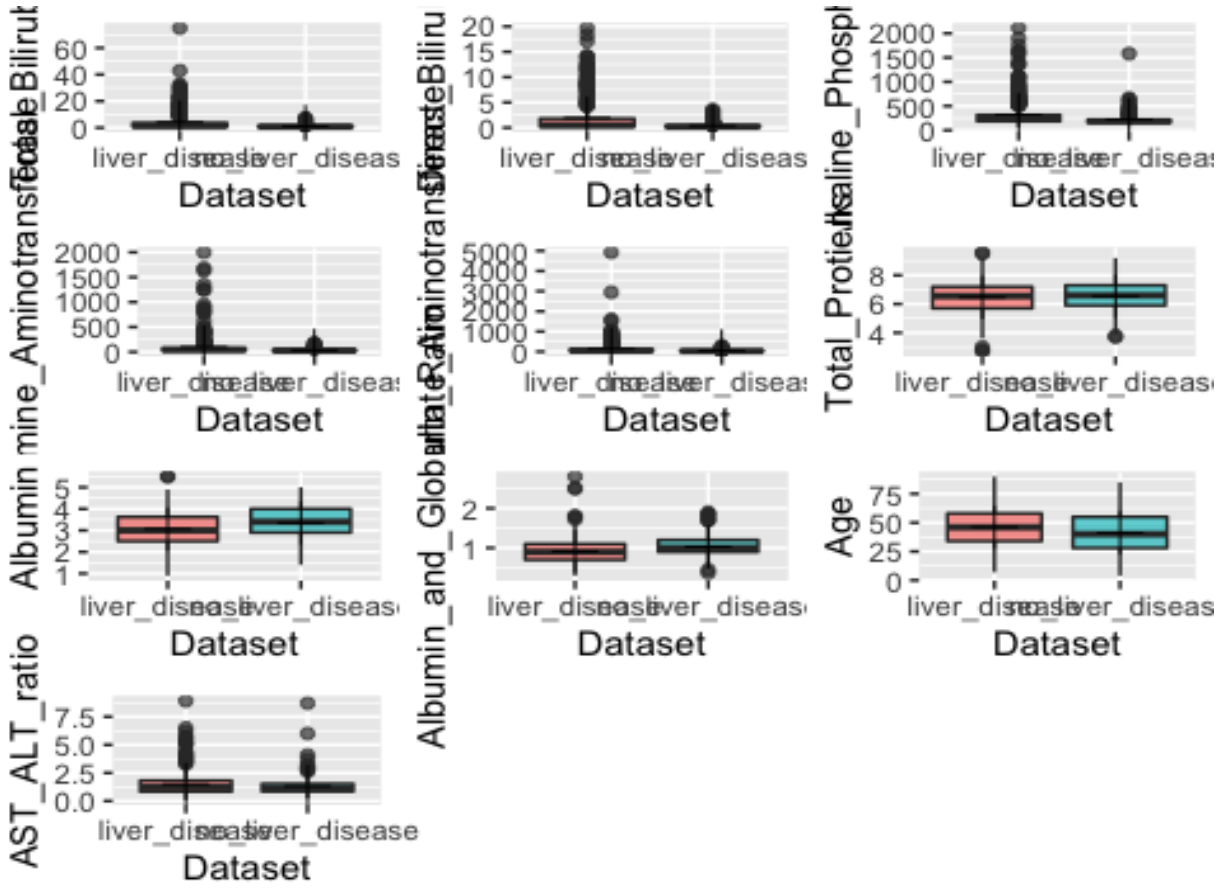
```
##       Age            Gender      Total_Bilirubin  Direct_Bilirubin
##  Min.   : 4.00   Min.   :1.000   Min.   : 0.400   Min.   : 0.100
##  1st Qu.:33.00   1st Qu.:2.000   1st Qu.: 0.800   1st Qu.: 0.200
##  Median :45.00   Median :2.000   Median : 1.000   Median : 0.300
##  Mean   :44.75   Mean   :1.756   Mean   : 3.299   Mean   : 1.486
##  3rd Qu.:58.00   3rd Qu.:2.000   3rd Qu.: 2.600   3rd Qu.: 1.300
##  Max.   :90.00   Max.   :2.000   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    Dataset
##  Min.   :2.700   Min.   :0.900   Min.   :0.3000             Min.   :1.000
##  1st Qu.:5.800   1st Qu.:2.600   1st Qu.:0.7000             1st Qu.:1.000
##  Median :6.600   Median :3.100   Median :0.9471             Median :1.000
##  Mean   :6.483   Mean   :3.142   Mean   :0.9471             Mean   :1.286
##  3rd Qu.:7.200   3rd Qu.:3.800   3rd Qu.:1.1000             3rd Qu.:2.000
##  Max.   :9.600   Max.   :5.500   Max.   :2.8000             Max.   :2.000
##  AST_ALT_ratio
##  Min.   :0.08995
##  1st Qu.:0.84000
##  Median :1.17391
##  Mean   :1.43734
##  3rd Qu.:1.72197
##  Max.   :8.92308
```

Lets see the distribution of different variables in the data



a

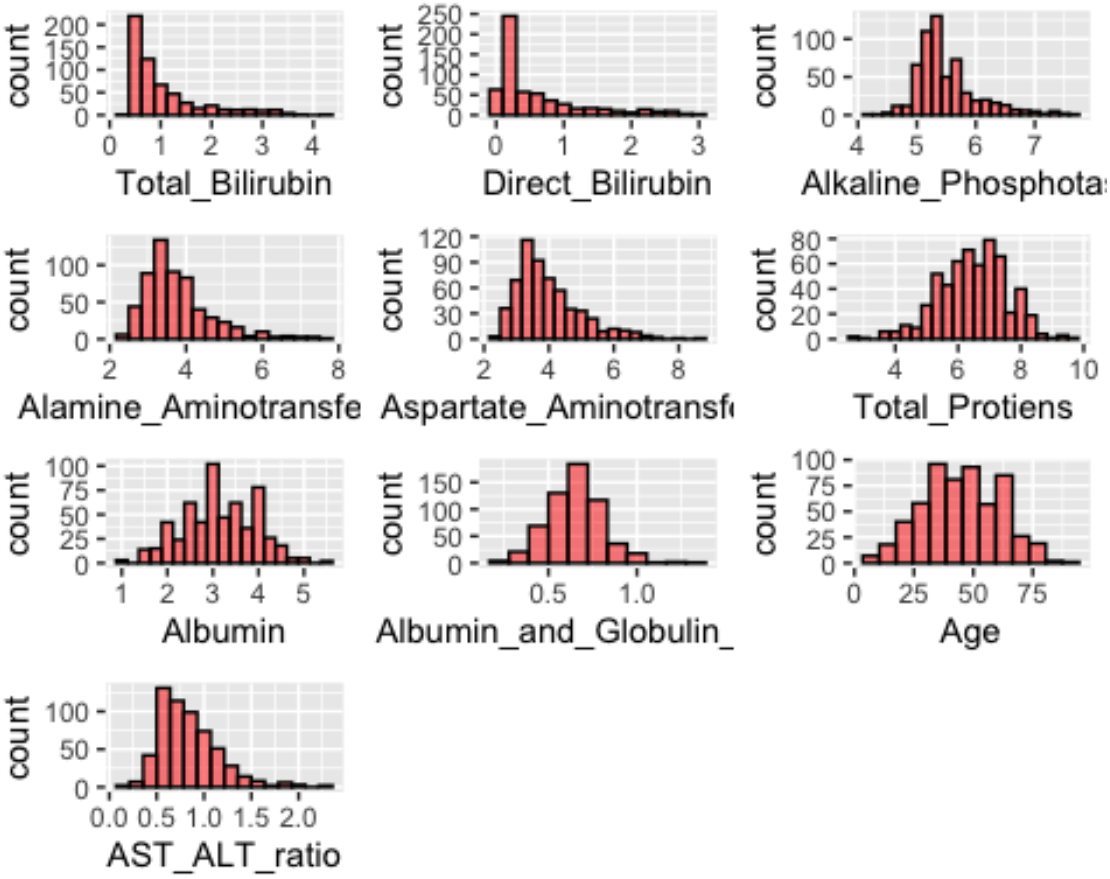
While come variables follow a normal distribution, a big majority of the variables show distribution similar to Poission. The plots look a bit messy and do not present a clear picture of what is going on. Box plot, divided by disease or not might be able to resolve this and present a better outcome to analyse.



looking at these plots, it looks like some of the variables have a high degree of dependency and can deviate by a big margin from their mean values if the patient has liver disease. We also notice that the some of the variales are highly skwed and need to be normalised before proceeding with the analysis.

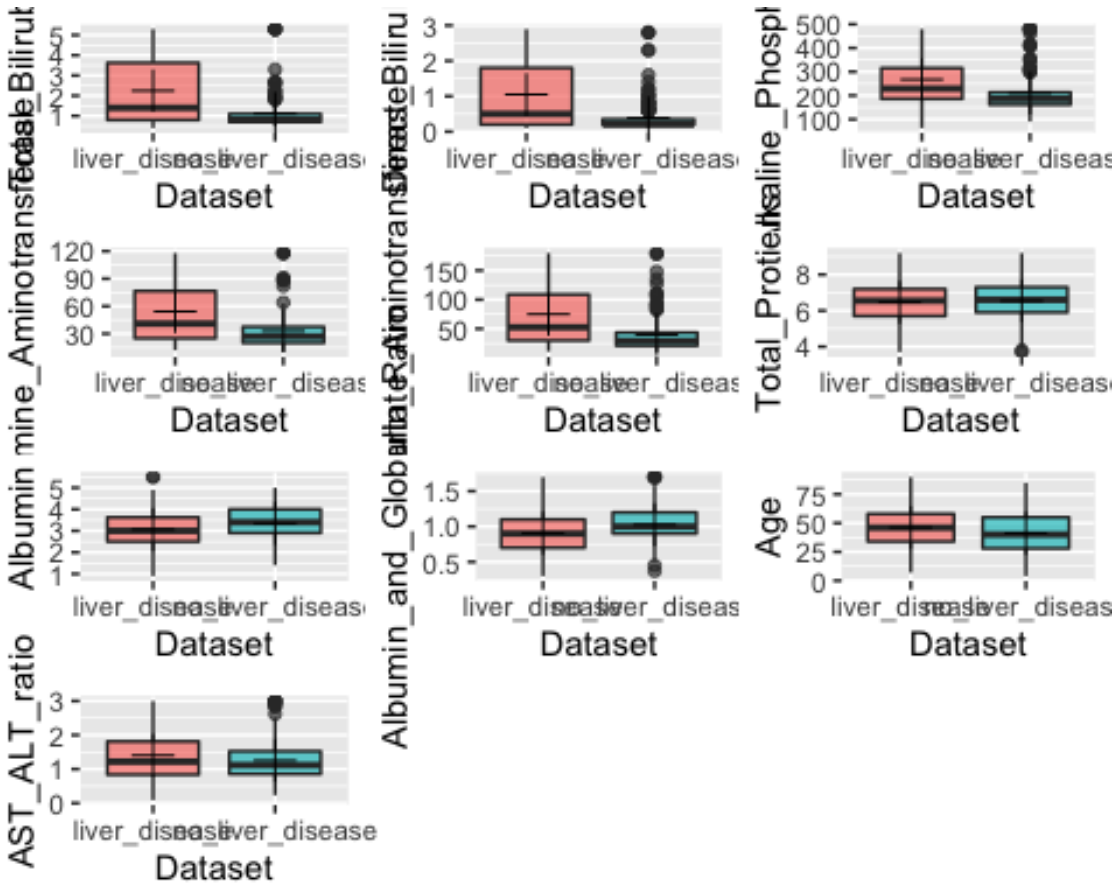
Applying log transformation on the data

Data distribution post log transformation



I decided to perform another transformation in order to tackle with extreme values, which could bias the statistic inferences and the predict models. For this, I used the boxplot.stats function and the rule that a data point is an outlier/extreme value if it is more than 1.5 * IQR (interquartile range) above the third quartile or below the first quartile.

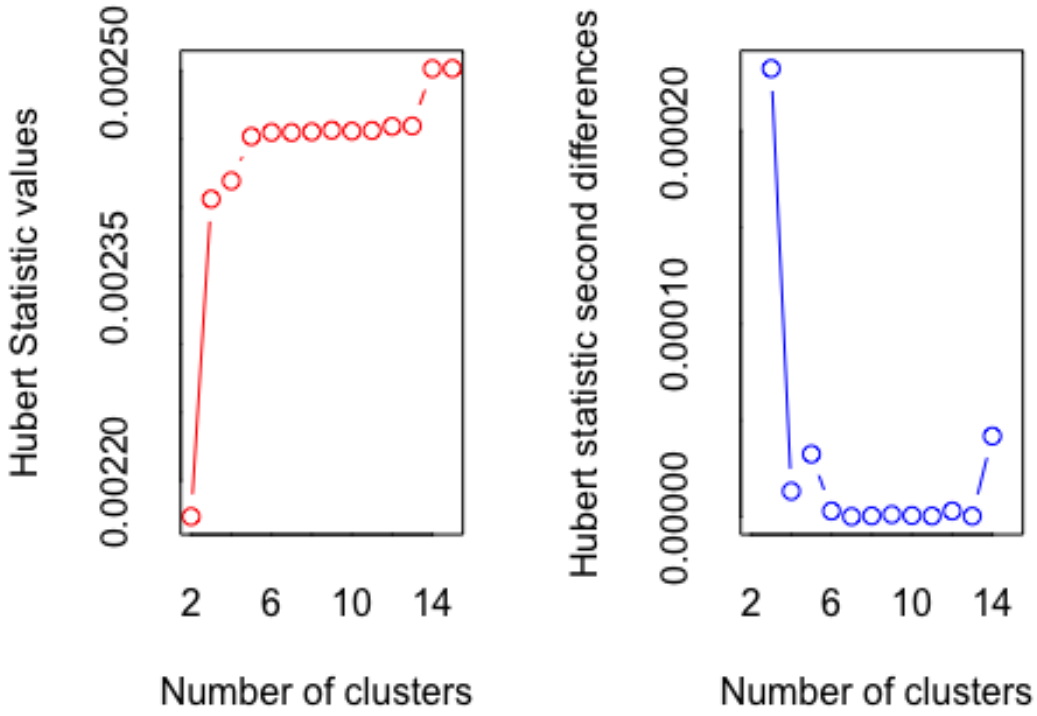
Box Plots after removing extreme values



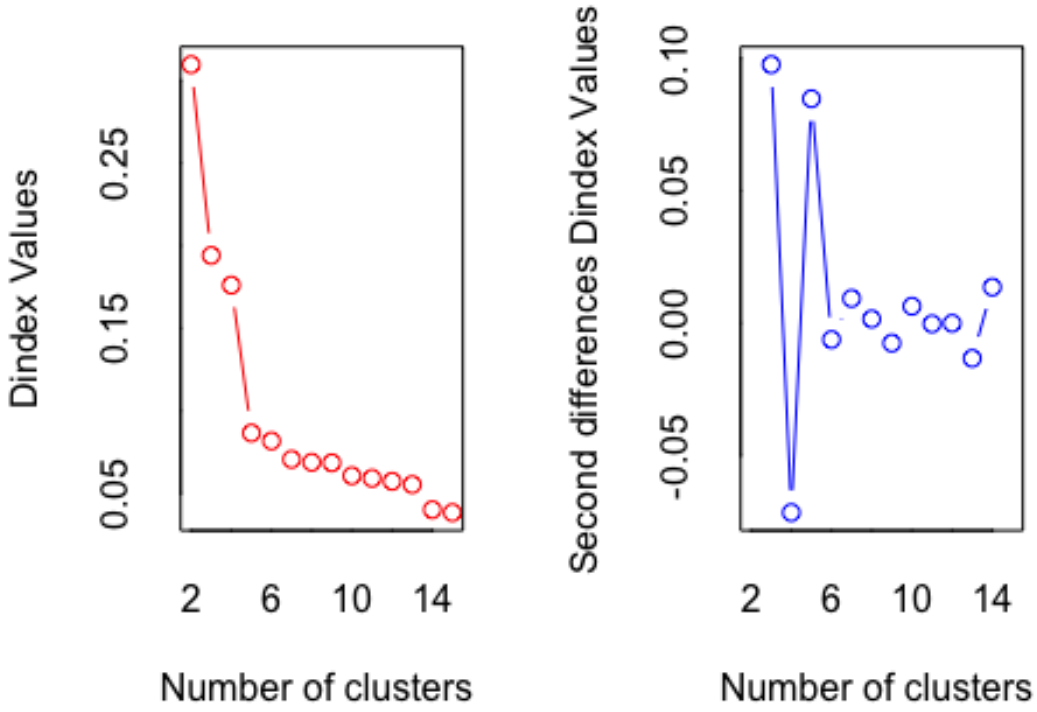
Data Clustering

I decided to perform the categorization of each column through the clustering approach in order to improve the statistic inferences and the predict models.

For this cluster analysis, we used the NbClust R package, which provides 30 indices for determining the best number of clusters with the Euclidean distance. The method of data clustering was hierarchical clustering with average linkage.

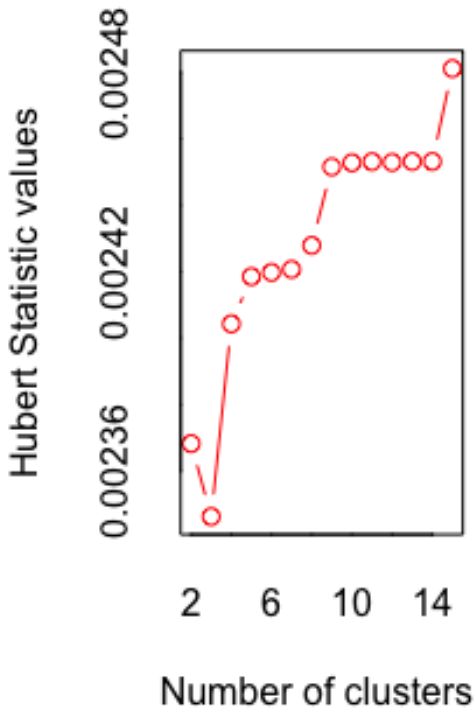


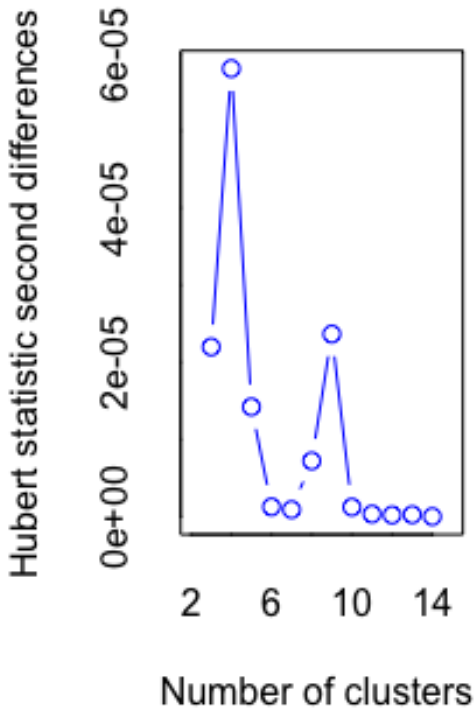

```
## *** : The Hubert index is a graphical method of determining the number of
clusters.
##                 In the plot of Hubert index, we seek a significant knee th
at corresponds to a
##                 significant increase of the value of the measure i.e the s
ignificant peak in Hubert
##                 index second differences plot.
##

## *** : The D index is a graphical method of determining the number of clust
ers.
##                 In the plot of D index, we seek a significant knee (the si
gnificant peak in Dindex
##                 second differences plot) that corresponds to a significant
increase of the value of
##                 the measure.
##
```

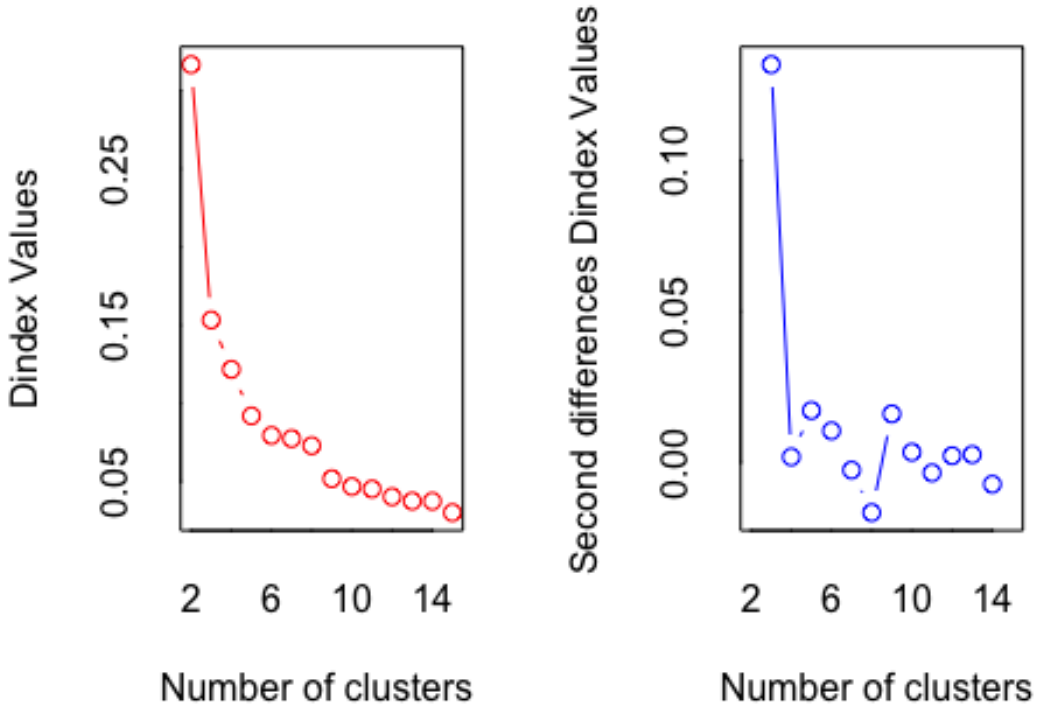



```
## *****************************************************************
## * Among all indices:
## * 1 proposed 2 as the best number of clusters
## * 4 proposed 5 as the best number of clusters
## * 1 proposed 14 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  5
##
##
## *****************************************************************
```

```
nc4 <- NbClust(scale(data.cat[, 4]), distance="euclidean", min.nc=2, max.nc=1
5, method="average") #5
```

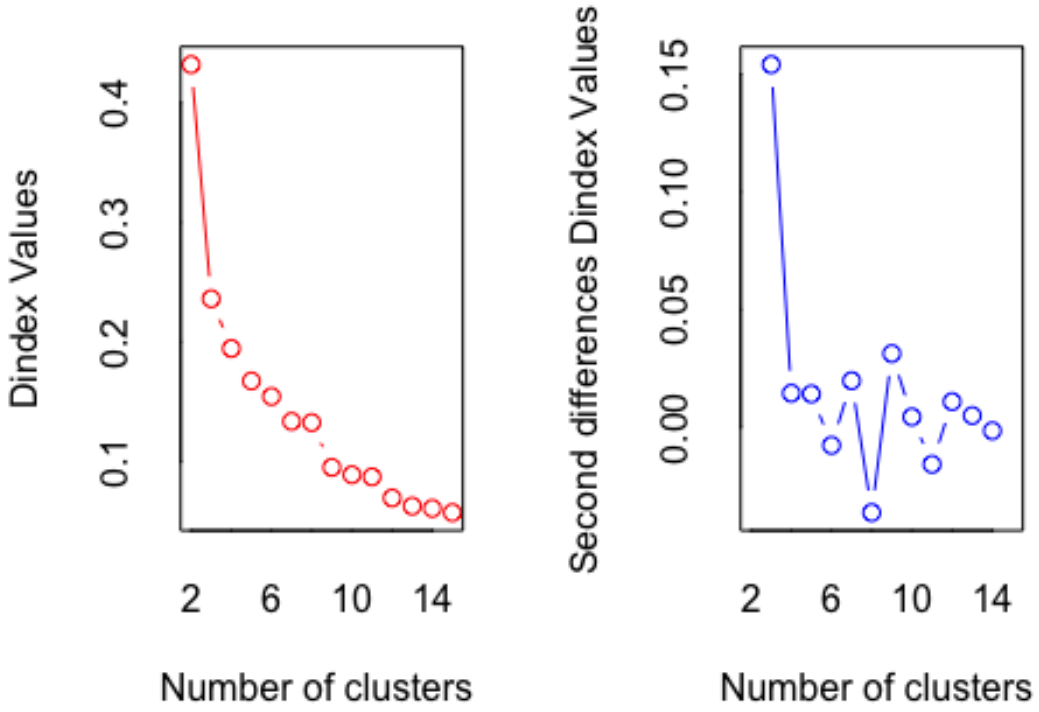

Hubert Statistic values
0.00236
0.00242
0.00248
2
6
10
14
Number of clusters


Hubert statistic second differences
0e+00
2e-05
4e-05
6e-05
2
6
10
14
Number of clusters



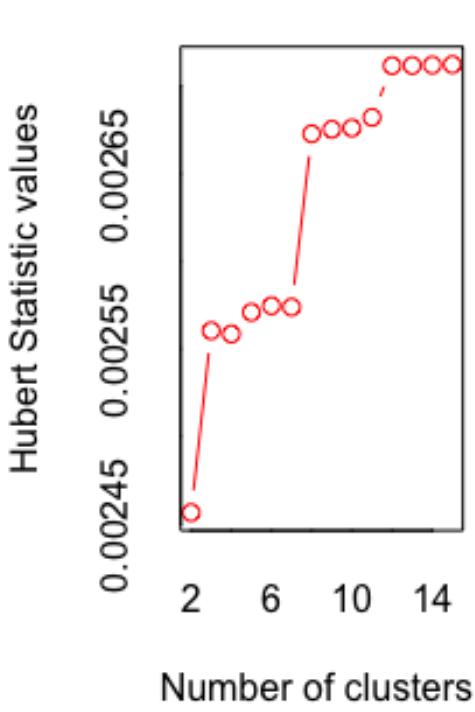

```
## *****************************************************************
## * Among all indices:
## * 3 proposed 3 as the best number of clusters
## * 1 proposed 9 as the best number of clusters
## * 1 proposed 14 as the best number of clusters
## * 1 proposed 15 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *****************************************************************
```

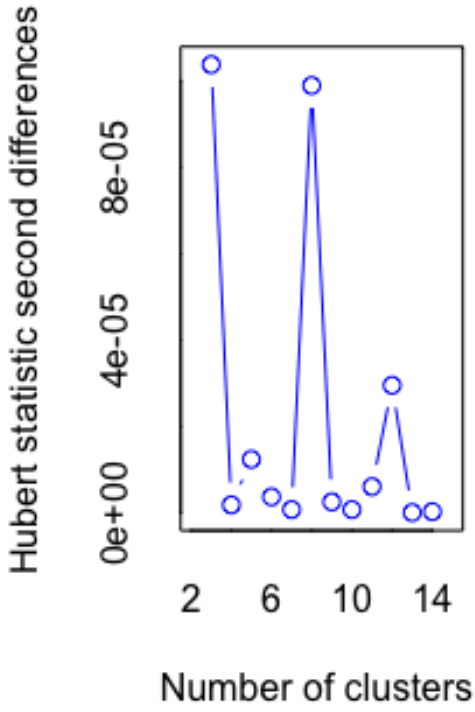
```
nc5 <- NbClust(scale(data.cat[, 5]), distance="euclidean", min.nc=2, max.nc=1
5, method="average") #8
```

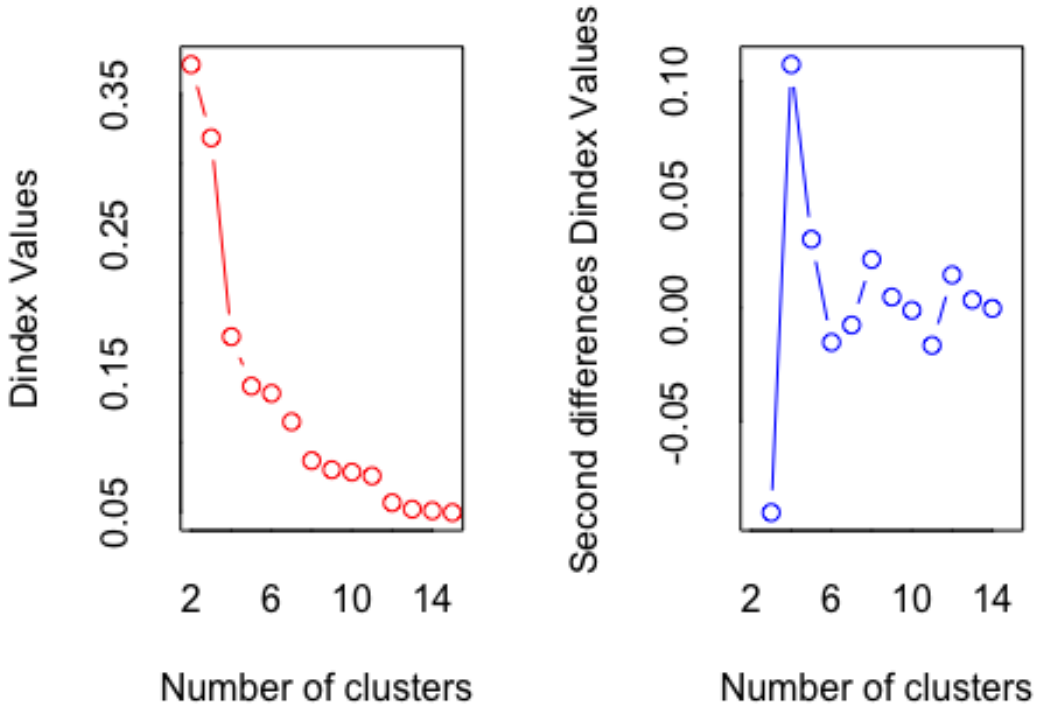



```
## *****************************************************************
## * Among all indices:
## * 3 proposed 3 as the best number of clusters
## * 2 proposed 9 as the best number of clusters
## * 1 proposed 14 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *****************************************************************
```

```
nc6 <- NbClust(scale(data.cat[, 6]), distance="euclidean", min.nc=2, max.nc=1
5, method="average") #8
```

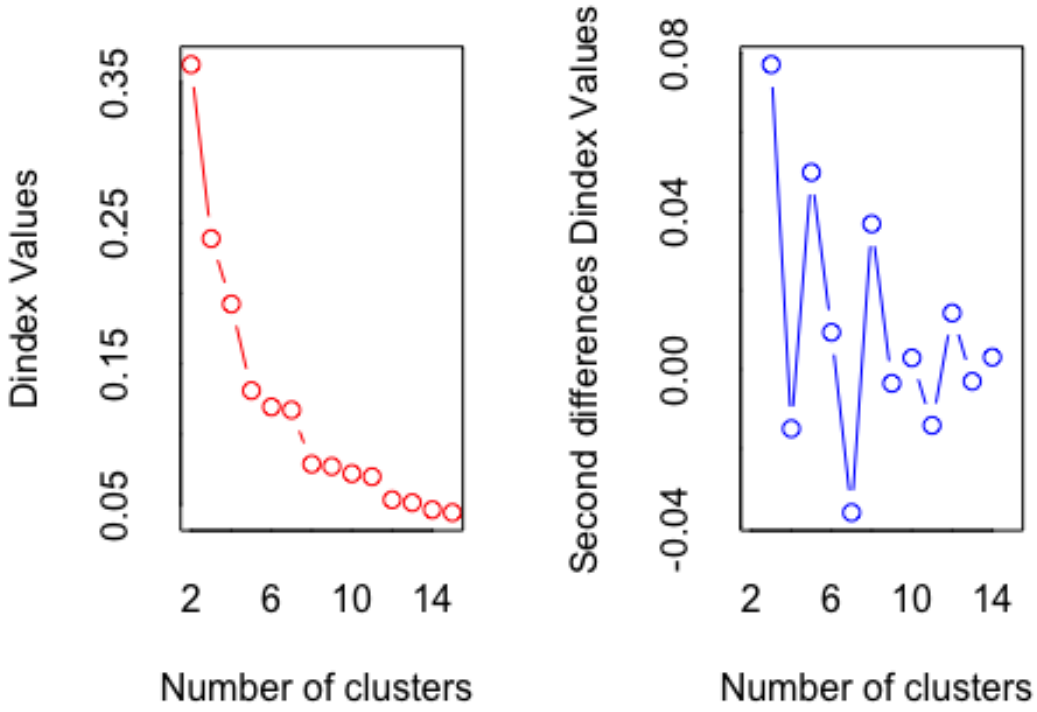

Hubert Statistic values
0.00245
0.00255
0.00265
2
6
10
14
Number of clusters


Hubert statistic second differences
0e+00
4e-05
8e-05
2
6
10
14
Number of clusters



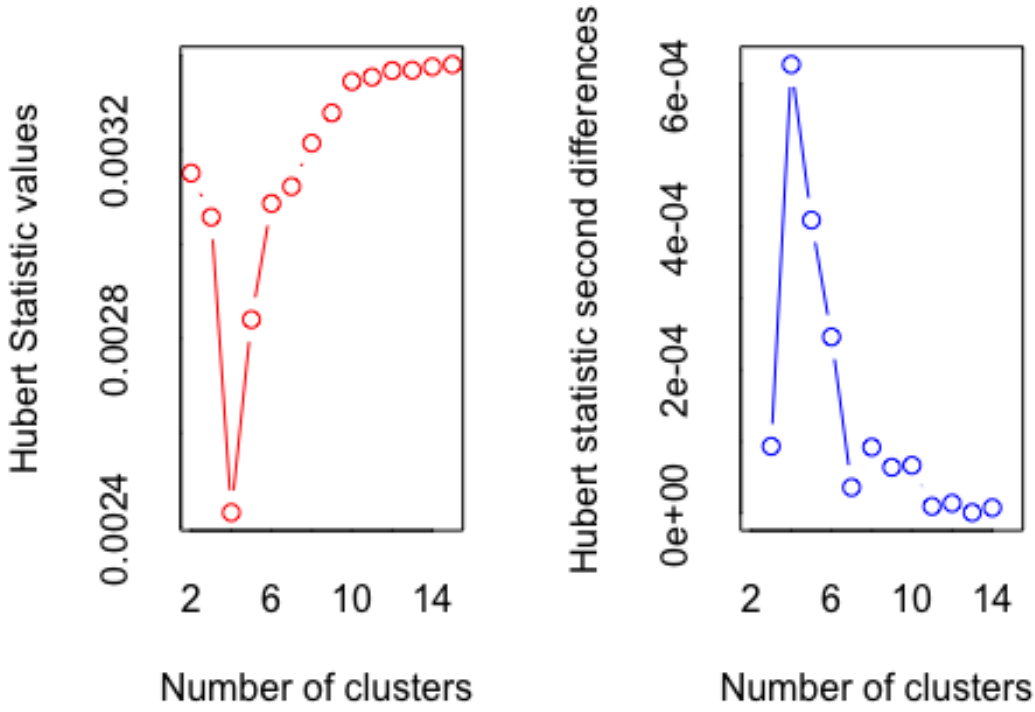

```
## *****************************************************************
## * Among all indices:
## * 2 proposed 4 as the best number of clusters
## * 3 proposed 8 as the best number of clusters
## * 1 proposed 13 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  8
##
##
## *****************************************************************
```

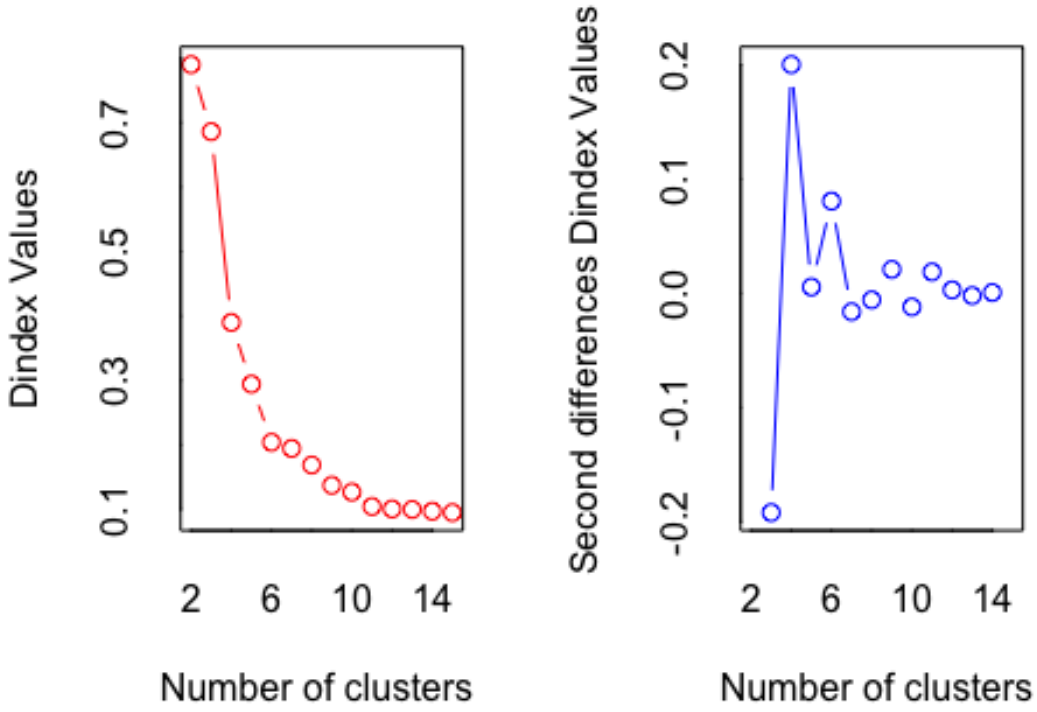
```
nc7 <- NbClust(scale(data.cat[, 7]), distance="euclidean", min.nc=2, max.nc=1
5, method="average") #5
```




```
## *****************************************************************
## * Among all indices:
## * 1 proposed 3 as the best number of clusters
## * 3 proposed 8 as the best number of clusters
## * 2 proposed 14 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  8
##
##
## *****************************************************************
```

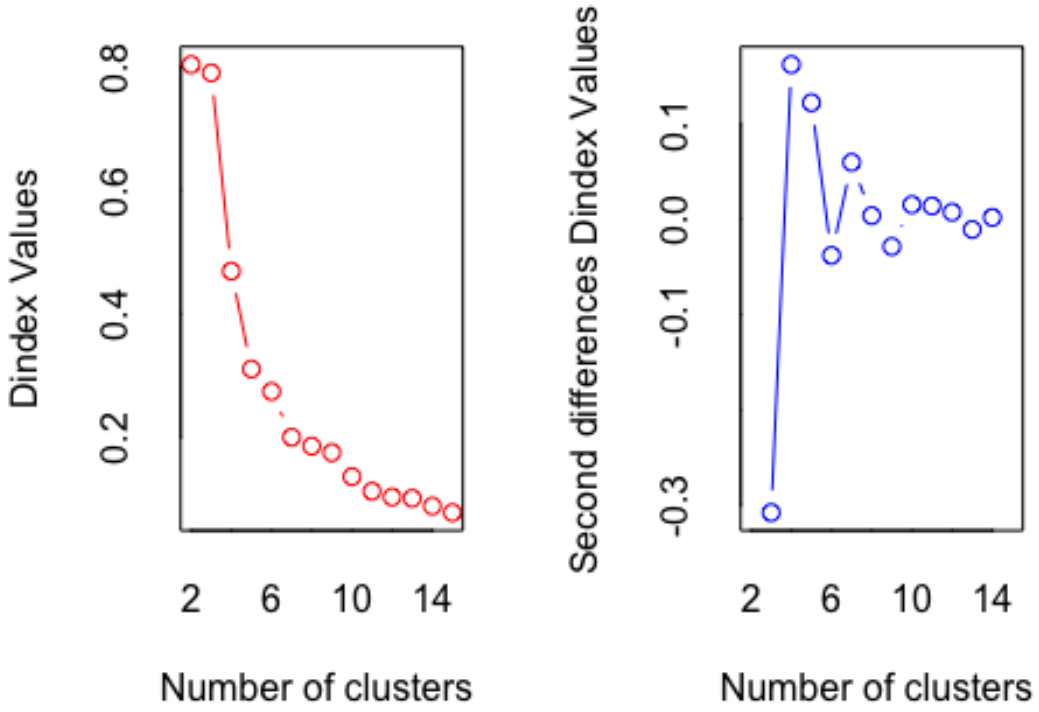
```
nc8 <- NbClust(scale(data.cat[, 8]), distance="euclidean", min.nc=2, max.nc=1
5, method="average") #4
```


Hubert Statistic values
0.0024
0.0028
0.0032
2
6
10
14
Number of clusters
Hubert statistic second differences
0e+00
2e-04
4e-04
6e-04
2
6
10
14
Number of clusters



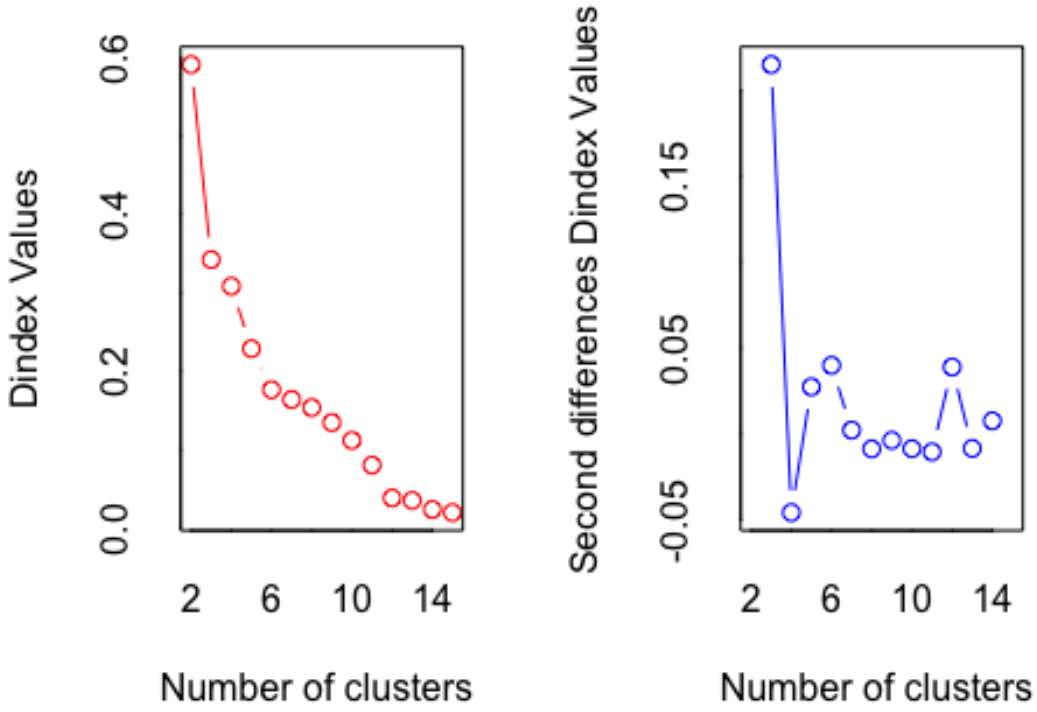

```
## *****************************************************************
## * Among all indices:
## * 2 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 2 proposed 11 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  4
##
##
## *****************************************************************
```

```
nc9 <- NbClust(scale(data.cat[, 9]), distance="euclidean", min.nc=2, max.nc=1
5, method="average") #15
```



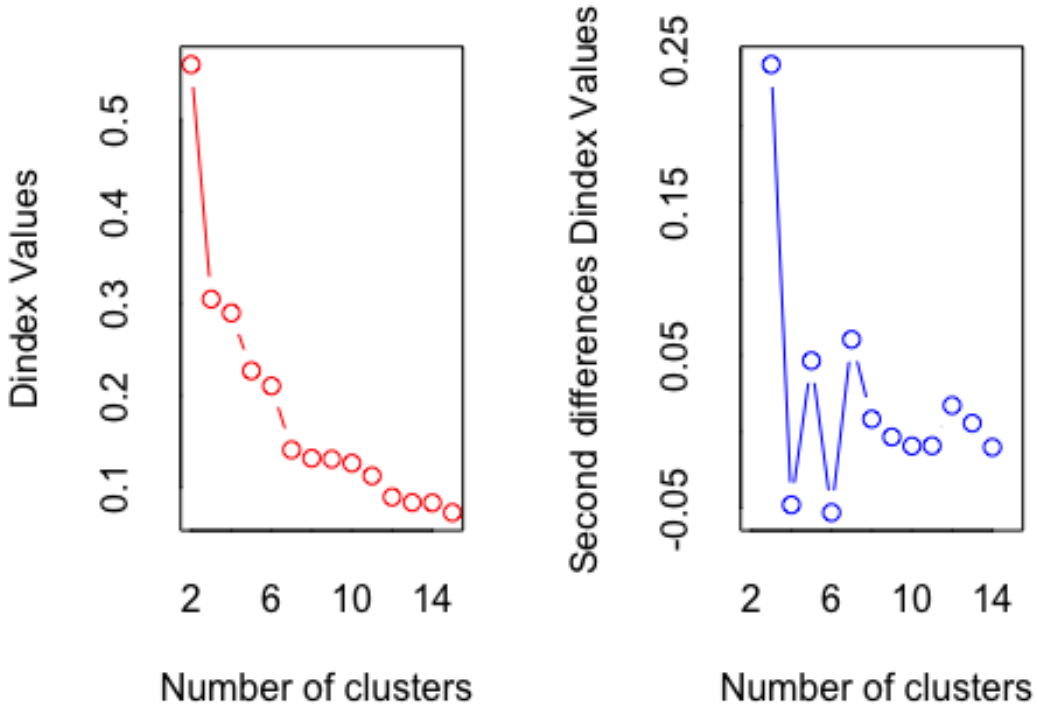

```
## *****************************************************************
## * Among all indices:
## * 2 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 1 proposed 14 as the best number of clusters
## * 2 proposed 15 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  4
##
##
## *****************************************************************
```

```
nc10 <- NbClust(scale(data.cat[, 10]), distance="euclidean", min.nc=2, max.nc
=15, method="average") #3
```

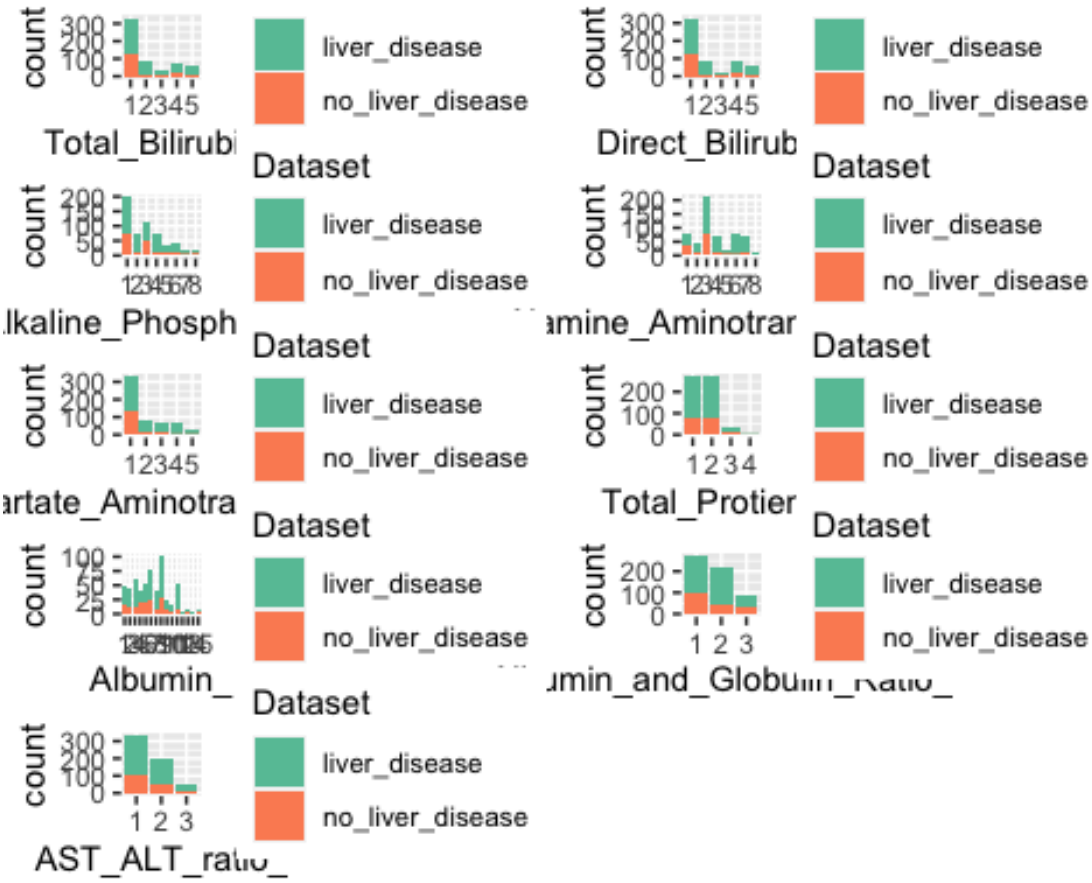



```
## *****************************************************************
## * Among all indices:
## * 2 proposed 3 as the best number of clusters
## * 1 proposed 11 as the best number of clusters
## * 1 proposed 12 as the best number of clusters
## * 2 proposed 15 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *****************************************************************
```

```
nc11 <- NbClust(scale(data.cat[, 11]), distance="euclidean", min.nc=2, max.nc
=15, method="average") #3
```




```
## *****************************************************************
## * Among all indices:
## * 3 proposed 3 as the best number of clusters
## * 1 proposed 6 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *****************************************************************
```



As this graph clearly shows there are variables with clearly show with 100% confidence that a patient has disease of not. Eg- High values of Total_Bilirub and Direct Bilirub indicare a patient has liver disease.

After all these procedures, we had the following 4 dataframes:

-> ORIGINAL DATAFRAME : data.o

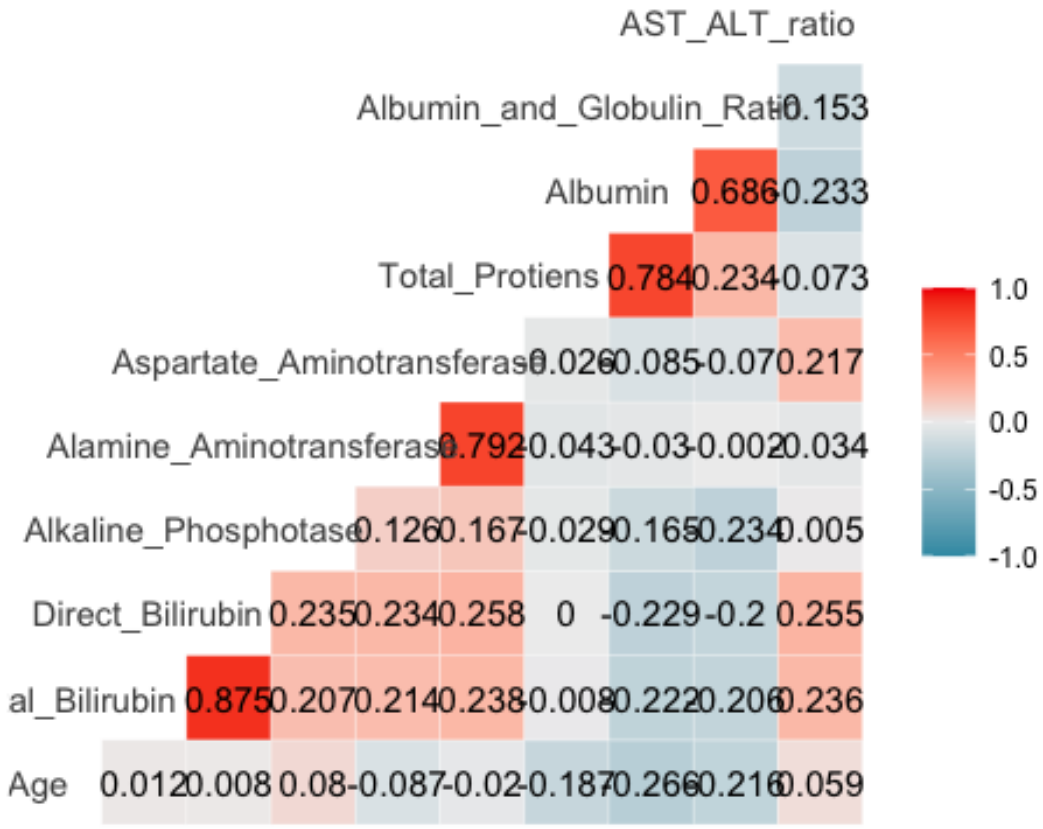
-> ORIGINAL DATAFRAME with LOG TRANSFORMATION: data.o.log

-> ORIGINAL DATAFRAME with EXTREME VALUES FITTED: data.o.wo

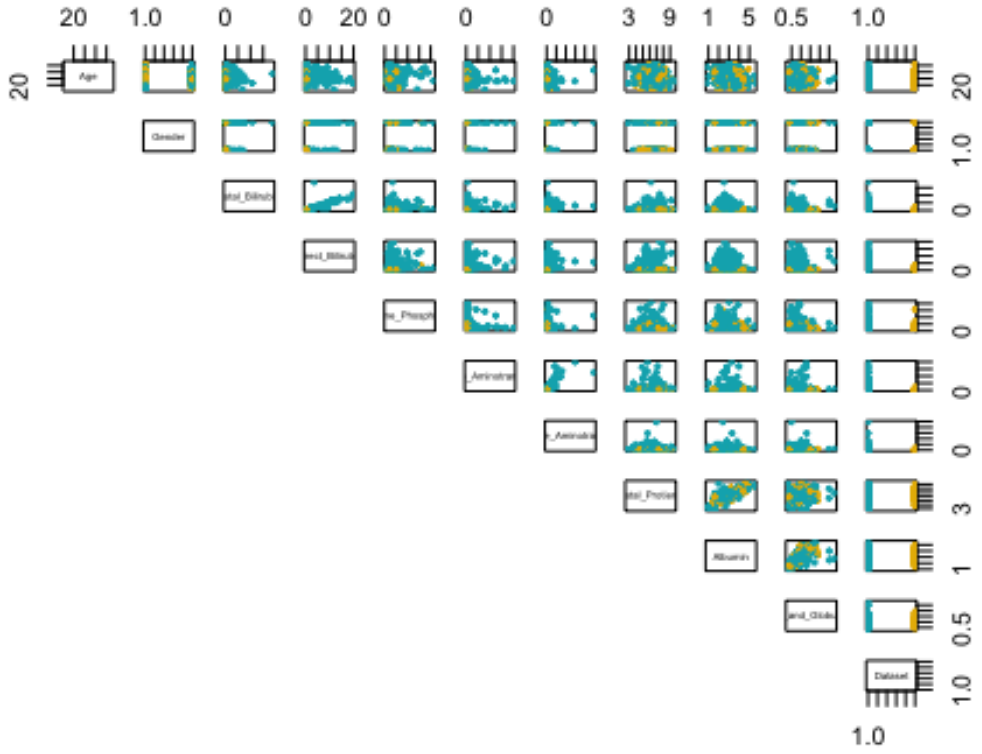
-> ORIGINAL DATAFRAME CATEGORIZATED BY CLUSTERING: data.cat.f

Chekcing for correlation

Correlations between continuous variables (original dataframe):




```
cols <- c("#00AFBB", "#E7B800")
pairs(data[,1:11], pch = 19,  cex = 0.4,
      col = cols[data.o$Dataset],
      lower.panel=NULL,
      cex.labels = 0.4)
```



In this case we used only the original dataframe, since the others showed similar results.

Some variables are strong directly correlated:

-> Total Bilirubin x Direct Bilirubin: 0.874

-> Alamine Aminotransferase x Aspartate Aminotransferease: 0.792

-> Total Protiens x Albumin: 0.783

Statistical Inference

Some variables show a skewed distribution. Many of statistical tests require the data to follow a normal distribution (parametric tests). Before using a statistical test, we plotted the Q-Q plot and Shapiro-Wilk's method (shapiro.test()) to make sure that the test assumptions were met. The data do not present a normal distribution, thus we applied non-paramatric tests.

In order to compare the two groups (patients with and without liver) and several variables not normally distributed, we chose the unpaired two-samples Wilcoxon test. Using the Test, we can decide whether the population distributions are identical without assuming them to follow the normal distribution.

We tested 3 dataframes (original, original log transformed and original with extreme values fitted) through the Mann-Whitney-Wilcoxon Test.

```
##                             wilcox.tests.o wilcox.tests.o.log wilcox.tests.
o.wo
## Total_Bilirubin            2.289519e-13   2.289519e-13       4.12924e-13
## Direct_Bilirubin           7.431126e-13   7.431126e-13       1.001788e-12
## Alkaline_Phosphotase       4.347234e-11   4.347234e-11       5.277923e-11
## Alamine_Aminotransferase   2.332935e-12   2.332935e-12       2.892522e-12
## Aspartate_Aminotransferase 9.209662e-14   9.209662e-14       9.742278e-14
## Total_Protiens             0.4371466      0.4371466          0.4377885
## Albumin                    5.567004e-05   5.567004e-05       5.567004e-05
## Albumin_and_Globulin_Ratio 6.179967e-06   6.179967e-06       6.052684e-06
## Age                        0.001774368    0.001774368        0.001774368
## AST_ALT_ratio              0.06012985     0.06012985         0.05988864
```

The results were similar for the 3 dataframes (original, log transformed and original with extreme values fitted).

-> Total_Bilirubin (e.g. The p-value less than 0.05. Hence, we reject the null hypothesis. There are significant differences in the median of Total_Bilirubin lab test for liver_disease and no_liver_disease groups. This supports our initial hypothesis that Total_Bilirubin is important.

After that, we got the original dataframe with categorization (data.cat.f) and compared categorical variables through the chi-square test of independence (used to analyze the frequency table). The chi-squared test is a statistical test used to discover whether there is a relationship between categorical variables.

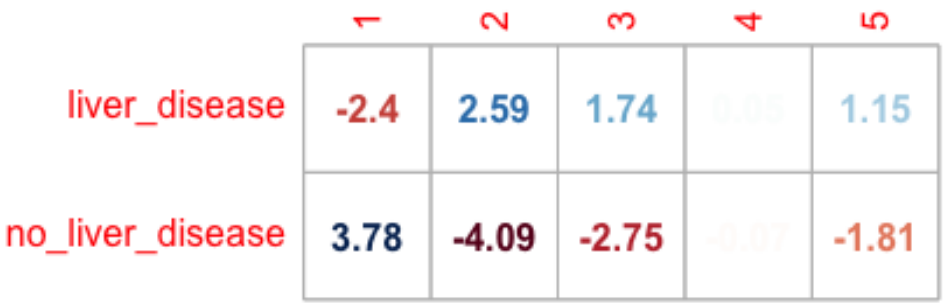
```
chisq.tests

## $Gender
## [1] 0.05966585
##
## $Total_Bilirubin_
## [1] 5.316031e-12
##
## $Direct_Bilirubin_
## [1] 1.569975e-12
##
## $Alkaline_Phosphotase_
## [1] 2.099307e-07
##
## $Alamine_Aminotransferase_
## [1] 1.483902e-09
##
## $Aspartate_Aminotransferase_
## [1] 1.688917e-10
##
## $Total_Protiens_
```

```
## [1] 0.8170174
##
## $Albumin_
## [1] 0.002262557
##
## $Albumin_and_Globulin_Ratio_
## [1] 3.180095e-05
##
## $AST_ALT_ratio_
## [1] 0.05087892
```

The standardized residuals is also important to interpret the association between rows and columns.

```
##                   Total_Bilirubin_
## Dataset                          1           2           3           4
5
##   liver_disease     -2.39734647  2.59311168  1.74193837  0.04747403  1.1489
8107
##   no_liver_disease   3.78372212 -4.09269754 -2.74929420 -0.07492807 -1.8134
3212
```

```
corrplot(chisq.tests.ind$residuals, is.cor = FALSE, method="number",  cl.pos
= "n")
```



For example, the residuals of chi-squared test between Dataset (liver_disease and no_liver_disease) and Total Bilirubin shows attraction (positive association) and repulsion (negative association). The group 1 (lower values of Total_Bilirubin), for example, shows a positive association with no_liver_disease and a negative association with liver_disease.

Feature selection

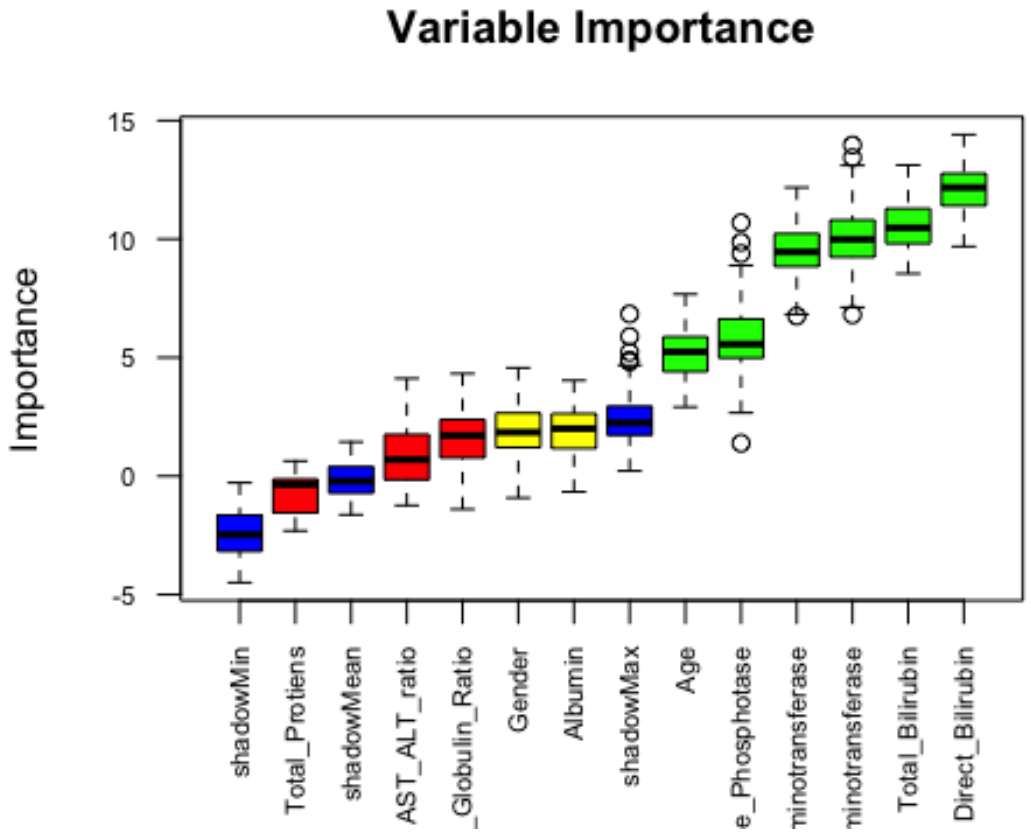
The results from statistical analysis showed that the variables Total Protiens, Gender and AST_ALT_ratio are not significantly different between no_liver_disease and liver_disease groups.

Boruta: It is a feature ranking and selection algorithm based on random forests. It clearly decides if a variable is important or not.

```
set.seed(123)
boruta_output <- Boruta(Dataset ~ ., data=data, doTrace=0)
boruta_output$finalDecision
```

```
##                       Age                    Gender
##                 Confirmed                 Tentative
##           Total_Bilirubin           Direct_Bilirubin
##                 Confirmed                 Confirmed
##      Alkaline_Phosphotase   Alamine_Aminotransferase
##                 Confirmed                 Confirmed
## Aspartate_Aminotransferase             Total_Protiens
##                 Confirmed                  Rejected
##                   Albumin Albumin_and_Globulin_Ratio
##                 Tentative                  Rejected
##              AST_ALT_ratio
##                  Rejected
## Levels: Tentative Confirmed Rejected
```

```
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")
```




```
##                         Age                      Gender
##                   Confirmed                    Rejected
##             Total_Bilirubin            Direct_Bilirubin
##                   Confirmed                   Confirmed
##       Alkaline_Phosphotase    Alamine_Aminotransferase
##                   Confirmed                   Confirmed
## Aspartate_Aminotransferase              Total_Protiens
##                   Confirmed                    Rejected
##                     Albumin Albumin_and_Globulin_Ratio
##                    Rejected                    Rejected
##               AST_ALT_ratio
##                    Rejected
## Levels: Tentative Confirmed Rejected
```

Boruta results show similar importance of results for all the data types i.e original, extreme value treated and categorized. Perhaps Variable Importance Through Random Forest might genereate a different result. Random forests are based on decision trees and use bagging to come up with a model over the data.

```
##                               Overall
## Age                         27.113885
## Gender                       3.839569
## Total_Bilirubin             21.783235
```

```
## Direct_Bilirubin           18.437218
## Alkaline_Phosphotase       32.185661
## Alamine_Aminotransferase   27.443977
## Aspartate_Aminotransferase 27.613227
## Total_Protiens             18.771252
## Albumin                    19.644155
## Albumin_and_Globulin_Ratio 16.684994
## AST_ALT_ratio              24.306470
```

Using Regression to Calculate Variable Importance: The summary function in regression also describes features and how they affect the dependent feature through significance.

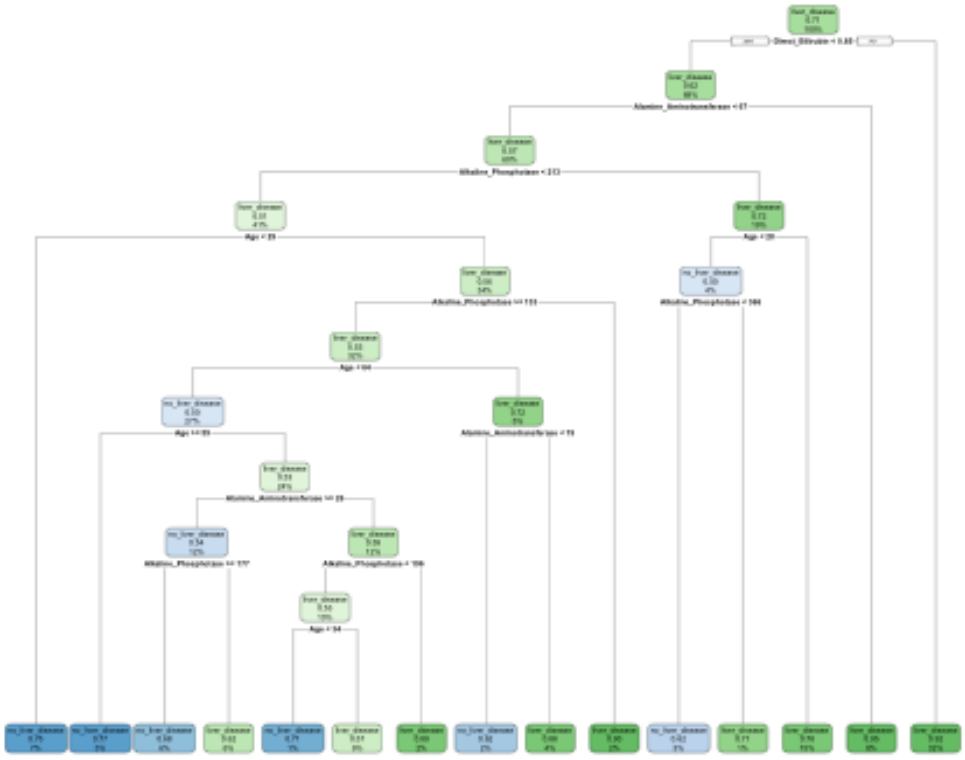
```
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 3.3077771  1.3232680   2.500  0.01243 *
## Age                        -0.0188251  0.0063849  -2.948  0.00319 **
## Gender                     -0.0211301  0.2320807  -0.091  0.92746
## Total_Bilirubin            -0.0093870  0.0839437  -0.112  0.91096
## Direct_Bilirubin           -0.4644644  0.2392147  -1.942  0.05218 .
## Alkaline_Phosphotase       -0.0012903  0.0008186  -1.576  0.11498
## Alamine_Aminotransferase   -0.0060047  0.0062295  -0.964  0.33509
## Aspartate_Aminotransferase -0.0076048  0.0050526  -1.505  0.13229
## Total_Protiens             -0.9483630  0.3655852  -2.594  0.00948 **
## Albumin                     1.7573139  0.7145085   2.459  0.01391 *
## Albumin_and_Globulin_Ratio -1.8840079  1.0896254  -1.729  0.08380 .
## AST_ALT_ratio               0.2763261  0.1999229   1.382  0.16692
```

The variable Gender is a non essential variable and was not used.

Modeling Liver Disease

Modelling for this problem was done by 3 different models. Each model used its own set of variables and determined its own level of importance. Each model successively improved of the previous model and generated better accuracy. Each of the 3 models were run though the original data, categorized data and extreme value treated data. Hence, there were a total of 9 models. Only the relavant models are shown below, the others were discarded but are present in the code. The 3 models that were used along with their accuracy rate based on confusion matrix.

Classification and regression trees



Model results in 68.97% Accuracy

```
# Prediction
predicted.classes <- model.tree.o %>% predict(data.Test, type = "class")
matrix.tree.o <- confusionMatrix(predicted.classes, data.Test$Dataset)
matrix.tree.o

## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         no_liver_disease liver_disease
```

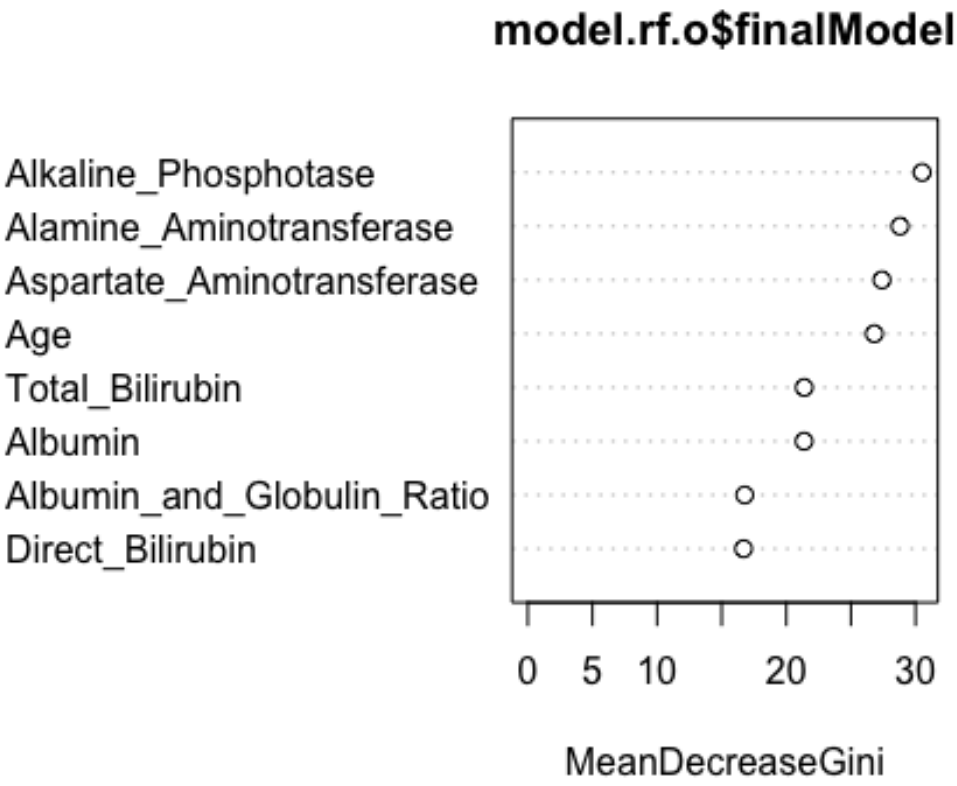
```
##   no_liver_disease               11            20
##   liver_disease                  22            63
##
##                Accuracy : 0.6379
##                  95% CI : (0.5435, 0.7251)
##     No Information Rate : 0.7155
##     P-Value [Acc > NIR] : 0.9726
##
##                   Kappa : 0.0941
##
##  Mcnemar's Test P-Value : 0.8774
##
##             Sensitivity : 0.33333
##             Specificity : 0.75904
##          Pos Pred Value : 0.35484
##          Neg Pred Value : 0.74118
##              Prevalence : 0.28448
##          Detection Rate : 0.09483
##    Detection Prevalence : 0.26724
##       Balanced Accuracy : 0.54618
##
##        'Positive' Class : no_liver_disease
##
```

Random Forests

Original dataframe

Model Results: model has an accuracy of 69.8%

```
# Plot MeanDecreaseGini
varImpPlot(model.rf.o$finalModel, type = 2)
```




```
varImp(model.rf.o)
```

```
## rf variable importance
##
##                              Overall
## Alkaline_Phosphotase       100.0000
## Alamine_Aminotransferase    87.6295
## Aspartate_Aminotransferase  77.5000
## Age                         73.2239
## Total_Bilirubin             33.9746
## Albumin                     33.8705
## Albumin_and_Globulin_Ratio   0.6548
## Direct_Bilirubin             0.0000
```

```
# Prediction
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         no_liver_disease liver_disease
##   no_liver_disease               11            11
##   liver_disease                  22            72
##
##                Accuracy : 0.7155
```

```
##                  95% CI : (0.6243, 0.7954)
##     No Information Rate : 0.7155
##     P-Value [Acc > NIR] : 0.54677
##
##                   Kappa : 0.2232
##
##  Mcnemar's Test P-Value : 0.08172
##
##             Sensitivity : 0.33333
##             Specificity : 0.86747
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.76596
##              Prevalence : 0.28448
##          Detection Rate : 0.09483
##    Detection Prevalence : 0.18966
##       Balanced Accuracy : 0.60040
##
##        'Positive' Class : no_liver_disease
##
```

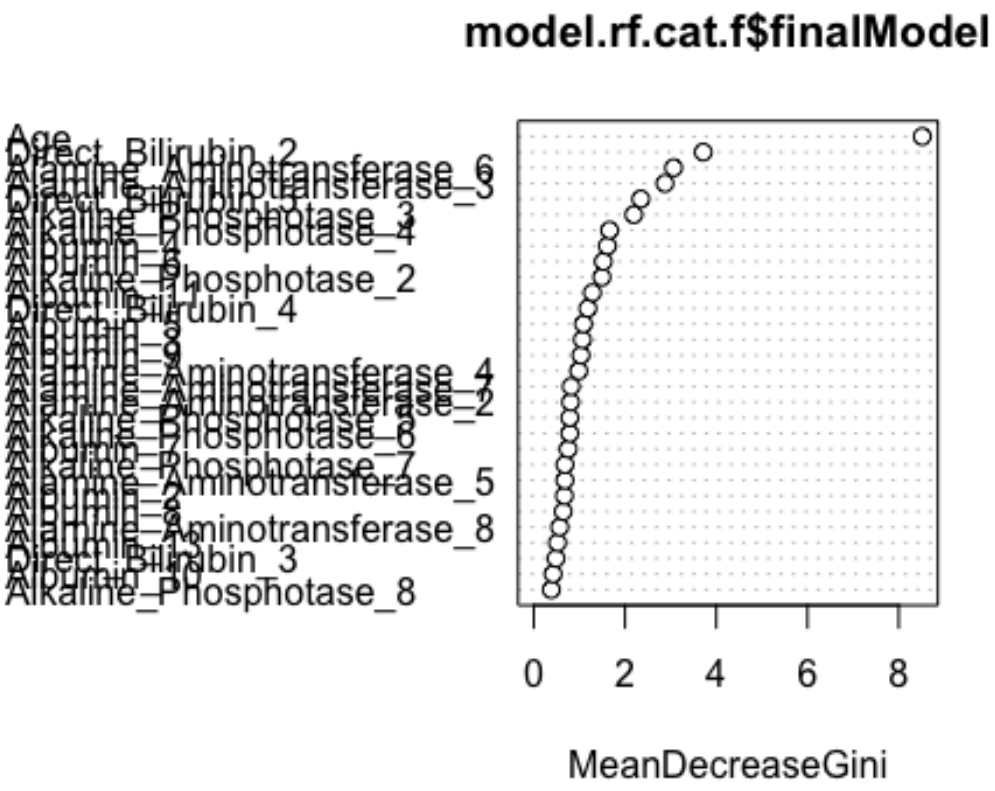
Random Forst with original dataframe categorized by:

Model accuracy = 71.55%

```
# Variable Importance
varImp(model.rf.cat.f)

## rf variable importance
##
##   only 20 most important variables shown (out of 33)
##
##                             Overall
## Age                         100.000
## Direct_Bilirubin_2           42.870
## Alamine_Aminotransferase_6   35.145
## Alamine_Aminotransferase_3   32.886
## Direct_Bilirubin_5           26.600
## Alkaline_Phosphotase_3       24.766
## Alkaline_Phosphotase_4       18.464
## Albumin_4                    17.889
## Albumin_6                    16.813
## Alkaline_Phosphotase_2       16.446
## Albumin_11                   13.980
## Direct_Bilirubin_4           12.794
## Albumin_5                    11.645
## Albumin_3                    11.380
## Albumin_9                    11.004
## Alamine_Aminotransferase_4   10.411
## Alamine_Aminotransferase_7    8.439
## Alamine_Aminotransferase_2    8.297
```

```
## Alkaline_Phosphotase_5       8.090
## Alkaline_Phosphotase_6       8.071
```

```
# Plot MeanDecreaseGini
varImpPlot(model.rf.cat.f$finalModel, type = 2)
```




```
# Prediction
predicted.classes <- model.rf.cat.f %>% predict(data.Test)
matrix.rf.cat.f <- confusionMatrix(predicted.classes, data.Test$Dataset)
matrix.rf.cat.f
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         no_liver_disease liver_disease
##   no_liver_disease                1             0
##   liver_disease                  32            83
##
##                Accuracy : 0.7241
##                  95% CI : (0.6334, 0.803)
##     No Information Rate : 0.7155
##     P-Value [Acc > NIR] : 0.4649
##
##                   Kappa : 0.0428
```

```
##
##  Mcnemar's Test P-Value : 4.251e-08
##
##             Sensitivity : 0.030303
##             Specificity : 1.000000
##          Pos Pred Value : 1.000000
##          Neg Pred Value : 0.721739
##              Prevalence : 0.284483
##          Detection Rate : 0.008621
##    Detection Prevalence : 0.008621
##       Balanced Accuracy : 0.515152
##
##        'Positive' Class : no_liver_disease
```

Train XGBoost model

To use XGBoost the data needs to be converted to a different format called DMatrix. DMatrix is an internal data structure used by XGBoost which is optimized for both memory efficiency and training speed.

```
## Stopping. Best iteration:
## [3]  train-error:0.116705    test-error:0.301370
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 388  67
##          1  28 100
##
##                Accuracy : 0.837
##                  95% CI : (0.8045, 0.8661)
##     No Information Rate : 0.7136
##     P-Value [Acc > NIR] : 2.377e-12
##
##                   Kappa : 0.5714
##
##  Mcnemar's Test P-Value : 9.670e-05
##
##             Sensitivity : 0.9327
##             Specificity : 0.5988
##          Pos Pred Value : 0.8527
##          Neg Pred Value : 0.7812
##              Prevalence : 0.7136
##          Detection Rate : 0.6655
##    Detection Prevalence : 0.7804
```

```
##       Balanced Accuracy : 0.7657
##
##        'Positive' Class : 0
##
```

```
#Accuracy 88.5%

#which features were most important
xgb.importance(colnames(fulldata), model = xgbModel)
```

```
##                       Feature       Gain      Cover  Frequency
## 1:       Alkaline_Phosphotase 0.20084921 0.21913435 0.20782938
## 2:           Direct_Bilirubin 0.19534923 0.17166063 0.02862155
## 3:                        Age 0.18406963 0.17454623 0.19453421
## 4: Aspartate_Aminotransferase 0.09612948 0.13403000 0.12279568
## 5:             Total_Protiens 0.08653312 0.03828992 0.13239775
## 6:   Alamine_Aminotransferase 0.08291753 0.10564068 0.09371249
## 7:                   Albumin 0.06174611 0.06242364 0.09168129
## 8: Albumin_and_Globulin_Ratio 0.05289526 0.05439954 0.08152525
## 9:                    Gender 0.03951043 0.03987501 0.04690241
```

Conclusion

- Classification and Regression Trees ~ 68.97% Accuracy
- Random Forest (Original Data) ~ 68.97% Accuracy
- Random Forest (Categorised variabes) ~ 71.55% Accuracy
- XGBoost ~ 88.5% Accuracy

In general, the models performed with a range accuracy of 68% - 88% and a number of predictors between 4 and 8. The most accurate outcome was through the XGBoost model. For a real world applicaton it is not just the accuracy that is important but the confusin matrix should yield minimal false negatives. As a patient diagnosed was “negative” but is positive will have escaped the system.

We were able to train a model to diagnose whether a patient has liver disease or not based on a set of available data points. Our model achieved an accuracy of 88.51%.

The dataset we used was indeed limited, and to truly have a model which generalizes well we would need to collect much more data but the results we achieved are very promising indeed. And hospitals and health authorities would clearly have more of the data we require to make our model achieve (or even surpass) human-level diagnosis accuracy. With more information our model will be able to perform better.