Decision Tree with R

Azeez Olalekan, Baki

28/12/2021

## The Data: Cardiotocographic

The data is a medical data compiled by cardiologists on patients to determine the heart disease. Target variable from the data is NSP. NSP is coded: 1- Normal patient 2- Suspect 3- Pathology.

There are total of 22 variables in the data with 2127 rows i.e. the number of patients sampled.

# Objective of the Analysis

The objective is to build a Decision Tree to predict the heart disease of the a patient with the variable highlighted.

# The Step:

To start our Analysis, let load the data and do some exploratory analysis on it.

Let get started!

Note, the target variable NSP is an integer but it should be factor. So, let create another column in our dataset convert it to factor.

heart$NSP\_factor = as.factor(heart$NSP)

Now, we have converted our target variable to character showing 1:normal, 2:suspect and 3:pathology.

## Exploration of the Data

Let start with the summary statistics of the data

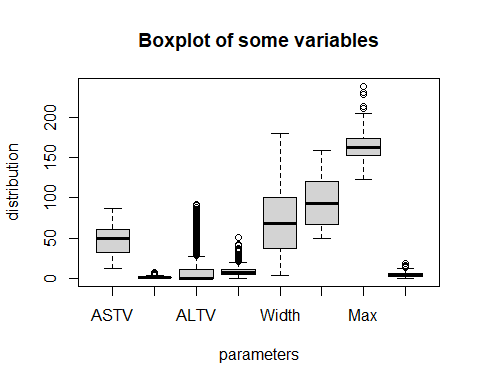
summary(heart[,1:21])

## LB AC FM UC   
## Min. :106.0 Min. :0.000000 Min. :0.000000 Min. :0.000000   
## 1st Qu.:126.0 1st Qu.:0.000000 1st Qu.:0.000000 1st Qu.:0.001876   
## Median :133.0 Median :0.001630 Median :0.000000 Median :0.004482   
## Mean :133.3 Mean :0.003170 Mean :0.009474 Mean :0.004357   
## 3rd Qu.:140.0 3rd Qu.:0.005631 3rd Qu.:0.002512 3rd Qu.:0.006525   
## Max. :160.0 Max. :0.019284 Max. :0.480634 Max. :0.014925   
## DL DS DP ASTV   
## Min. :0.000000 Min. :0.000e+00 Min. :0.0000000 Min. :12.00   
## 1st Qu.:0.000000 1st Qu.:0.000e+00 1st Qu.:0.0000000 1st Qu.:32.00   
## Median :0.000000 Median :0.000e+00 Median :0.0000000 Median :49.00   
## Mean :0.001885 Mean :3.585e-06 Mean :0.0001566 Mean :46.99   
## 3rd Qu.:0.003264 3rd Qu.:0.000e+00 3rd Qu.:0.0000000 3rd Qu.:61.00   
## Max. :0.015385 Max. :1.353e-03 Max. :0.0053476 Max. :87.00   
## MSTV ALTV MLTV Width   
## Min. :0.200 Min. : 0.000 Min. : 0.000 Min. : 3.00   
## 1st Qu.:0.700 1st Qu.: 0.000 1st Qu.: 4.600 1st Qu.: 37.00   
## Median :1.200 Median : 0.000 Median : 7.400 Median : 67.50   
## Mean :1.333 Mean : 9.847 Mean : 8.188 Mean : 70.45   
## 3rd Qu.:1.700 3rd Qu.:11.000 3rd Qu.:10.800 3rd Qu.:100.00   
## Max. :7.000 Max. :91.000 Max. :50.700 Max. :180.00   
## Min Max Nmax Nzeros   
## Min. : 50.00 Min. :122 Min. : 0.000 Min. : 0.0000   
## 1st Qu.: 67.00 1st Qu.:152 1st Qu.: 2.000 1st Qu.: 0.0000   
## Median : 93.00 Median :162 Median : 3.000 Median : 0.0000   
## Mean : 93.58 Mean :164 Mean : 4.068 Mean : 0.3236   
## 3rd Qu.:120.00 3rd Qu.:174 3rd Qu.: 6.000 3rd Qu.: 0.0000   
## Max. :159.00 Max. :238 Max. :18.000 Max. :10.0000   
## Mode Mean Median Variance   
## Min. : 60.0 Min. : 73.0 Min. : 77.0 Min. : 0.00   
## 1st Qu.:129.0 1st Qu.:125.0 1st Qu.:129.0 1st Qu.: 2.00   
## Median :139.0 Median :136.0 Median :139.0 Median : 7.00   
## Mean :137.5 Mean :134.6 Mean :138.1 Mean : 18.81   
## 3rd Qu.:148.0 3rd Qu.:145.0 3rd Qu.:148.0 3rd Qu.: 24.00   
## Max. :187.0 Max. :182.0 Max. :186.0 Max. :269.00   
## Tendency   
## Min. :-1.0000   
## 1st Qu.: 0.0000   
## Median : 0.0000   
## Mean : 0.3203   
## 3rd Qu.: 1.0000   
## Max. : 1.0000

The summary results show the mean of each variable, the minimum value of the variable, the maximum the first and the third quartile as well as median, i.e. the second quartile. From here, we know the mean value of the variables.

let plot the statistics of the result using boxplot.

boxplot(heart[,8:15], ylab = "distribution", xlab = "parameters", main= "Boxplot of some variables")



## Decision Tree

For the Decision Tree we will be using the package party. Let start by installing and calling the library. Before that let split the data into test and train(validate)data.

set.seed(1234)  
heart1 <- sample(2, replace = TRUE, prob = c(0.9, 0.1), nrow(heart))  
htrain <- heart[heart1==1,]  
htest <- heart[heart1==2,]

We have divided our data into two sizes, labelled 1 and 2. 1 is train data and 2 is validate(test) data. Note, the function set.seed: this is done to make sure we get same number for our train and test data. Next is to build the decision tree model For the Decision Tree we will be using the package party. Let start by installing and calling the library.

library("party")

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

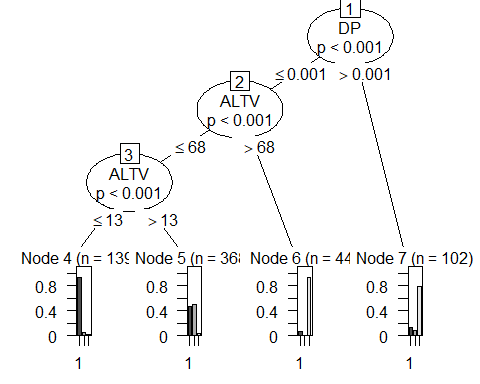
## Loading required package: sandwich

Good!. We have successfully installed the required package. Next is to build the decision tree mode

tree <- ctree(NSP\_factor~LB+AC+FM+UC+DL+DS+DP+ASTV+MSTV+ALTV+MLTV+Width+Min+Max+Nmax+Nzeros+Mode+Mean+Median+Variance+Tendency, data=htrain, controls = ctree\_control(minsplit = 1500, mincriterion = 0.99))  
tree

##   
## Conditional inference tree with 4 terminal nodes  
##   
## Response: NSP\_factor   
## Inputs: LB, AC, FM, UC, DL, DS, DP, ASTV, MSTV, ALTV, MLTV, Width, Min, Max, Nmax, Nzeros, Mode, Mean, Median, Variance, Tendency   
## Number of observations: 1908   
##   
## 1) DP <= 0.001422475; criterion = 1, statistic = 637.355  
## 2) ALTV <= 68; criterion = 1, statistic = 666.13  
## 3) ALTV <= 13; criterion = 1, statistic = 537.628  
## 4)\* weights = 1394   
## 3) ALTV > 13  
## 5)\* weights = 368   
## 2) ALTV > 68  
## 6)\* weights = 44   
## 1) DP > 0.001422475  
## 7)\* weights = 102

plot(tree)

 ## Interpretation of the Decision tree The result of the tree shows that if the Dp of the patient is greater than 0.001, the patient is likely to have a pathogenic heart problem. But if less than or equal 0.001, we should check the ALTV, if that > 68, there is 90% chance the patient is a pathogenic patient other </= 68 and ALTV>13, 55% of suspect.

## Prediction of test data from the model

obs <- predict(tree, htest)  
obs

## [1] 1 3 1 1 2 1 1 3 2 2 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 2 2 2 2 2 2 1 3 2  
## [38] 2 2 2 2 2 1 1 2 1 2 1 1 2 1 1 2 2 2 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1  
## [75] 1 3 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 3 1 1  
## [112] 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 1 2 1 3  
## [149] 1 3 1 1 1 1 2 2 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3  
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 1 2 2 1 3 2 1 1 2  
## Levels: 1 2 3

# Interpretation

From the obs result, the model predict the outcome of the test data. It was observed that for the first patient, s/he is a normal patient, patient 2 is pathogenic heart problem, patient 34, is suspect… if we have new dataset, we can easily predict the outcome of the patient given all the predictors. we can do that using “predict(tree, newdata)”.

## Accuracy of the model

acc <- table(predict(tree), htrain$NSP\_factor)  
print(acc)

##   
## 1 2 3  
## 1 1297 69 28  
## 2 171 182 15  
## 3 16 9 121

##Interpretation The result of the accuracy shows that out of the normal patient sampled, 1297 was predicted correctly, 171 was predicted to be suspect while 16 was predicted to be pathogenic. For the suspect, 69 was predicted to be normal, 182 to be suspect and 9 to be pathogen.

## Random Forest Model

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

trc <- trainControl(method = "repeatedcv", number = 10, repeats = 5)  
model <- train(NSP\_factor~LB+AC+FM+UC+DL+DS+DP+ASTV+MSTV+ALTV+MLTV+Width+Min+Max+Nmax+Nzeros+Mode+Mean+Median+Variance+Tendency,data=htrain, method = "rf", trControl = trc)  
obs <- predict(model, htrain)  
obs

## [1] 2 1 1 1 1 3 3 3 3 3 2 2 1 1 1 1 2 1 1 3 1 3 3 3 3 3 2 1 1 1 1 1 1 1 1 1 1  
## [38] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2  
## [75] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [112] 1 1 2 2 2 2 2 1 1 1 1 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 2 2 2 1 2 2 1 1 1 1 1 1 1  
## [186] 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 2 2 2 1 1  
## [223] 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2  
## [260] 2 2 1 2 2 2 1 2 2 3 3 3 2 2 2 2 2 2 2 3 2 2 2 2 2 2 3 2 3 3 2 2 3 3 2 2 2  
## [297] 3 2 2 2 3 3 2 2 3 3 3 2 2 2 2 2 2 3 3 2 2 1 2 1 2 2 2 3 3 2 2 2 2 1 2 2 2  
## [334] 2 2 1 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 2 2 1 2 3 2 2 2 2 2 2 2 2 1 1 1  
## [371] 2 2 3 2 2 2 2 1 1 1 1 1 1 1 1 2 2 2 2 2 1 1 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1  
## [408] 1 2 1 1 1 2 3 3 3 3 3 3 3 3 2 2 2 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [445] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 2 2 1  
## [482] 2 1 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [519] 1 1 1 1 1 1 1 1 1 2 3 3 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [556] 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 3 3 2 2 2 2 3 3 3 3 3 1 1 1 1 1 2 2 2 2 2 2  
## [593] 2 1 1 2 1 1 1 1 1 1 1 3 3 1 1 1 1 1 2 3 1 1 1 1 1 1 3 3 2 3 3 3 3 3 2 3 1  
## [630] 1 1 1 1 1 1 1 2 2 3 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 3 3 3 1 1  
## [667] 1 1 1 1 2 1 1 2 1 2 2 3 3 3 3 3 2 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1  
## [704] 1 2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 2 2 2 2 3 3 2 1 1 2 1 1 2  
## [741] 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [778] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [815] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [852] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [889] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [926] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [963] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1000] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1037] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2  
## [1074] 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1111] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1148] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1185] 1 1 1 1 1 1 1 1 1 1 1 2 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2  
## [1222] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 2 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 2 2  
## [1259] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 2 1  
## [1296] 1 1 1 1 1 2 1 1 1 1 1 1 2 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 3 2 3 1 1 1 1 1  
## [1333] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1  
## [1370] 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1407] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 2 1 1 1 1 1 1 1 1 1  
## [1444] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1481] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1  
## [1518] 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1  
## [1555] 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [1592] 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1629] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1666] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1703] 1 1 1 1 1 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1  
## [1740] 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1777] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1814] 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1  
## [1851] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [1888] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1  
## Levels: 1 2 3

obs1 <- predict(model, htest)  
obs1

## [1] 1 3 1 1 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 2 2 2 2 2 2 3 2  
## [38] 2 2 2 2 2 1 1 2 2 2 1 1 2 1 1 1 3 3 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1  
## [75] 1 3 1 1 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [112] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1  
## [149] 1 1 2 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3  
## [186] 1 2 1 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1  
## Levels: 1 2 3

confusionMatrix(obs, htrain$NSP\_factor)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3  
## 1 1484 2 0  
## 2 0 258 0  
## 3 0 0 164  
##   
## Overall Statistics  
##   
## Accuracy : 0.999   
## 95% CI : (0.9962, 0.9999)  
## No Information Rate : 0.7778   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9972   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3  
## Sensitivity 1.0000 0.9923 1.00000  
## Specificity 0.9953 1.0000 1.00000  
## Pos Pred Value 0.9987 1.0000 1.00000  
## Neg Pred Value 1.0000 0.9988 1.00000  
## Prevalence 0.7778 0.1363 0.08595  
## Detection Rate 0.7778 0.1352 0.08595  
## Detection Prevalence 0.7788 0.1352 0.08595  
## Balanced Accuracy 0.9976 0.9962 1.00000

#result <- predict(model, data.frame(LB=150, AC=0.003564), interval = "confidence")