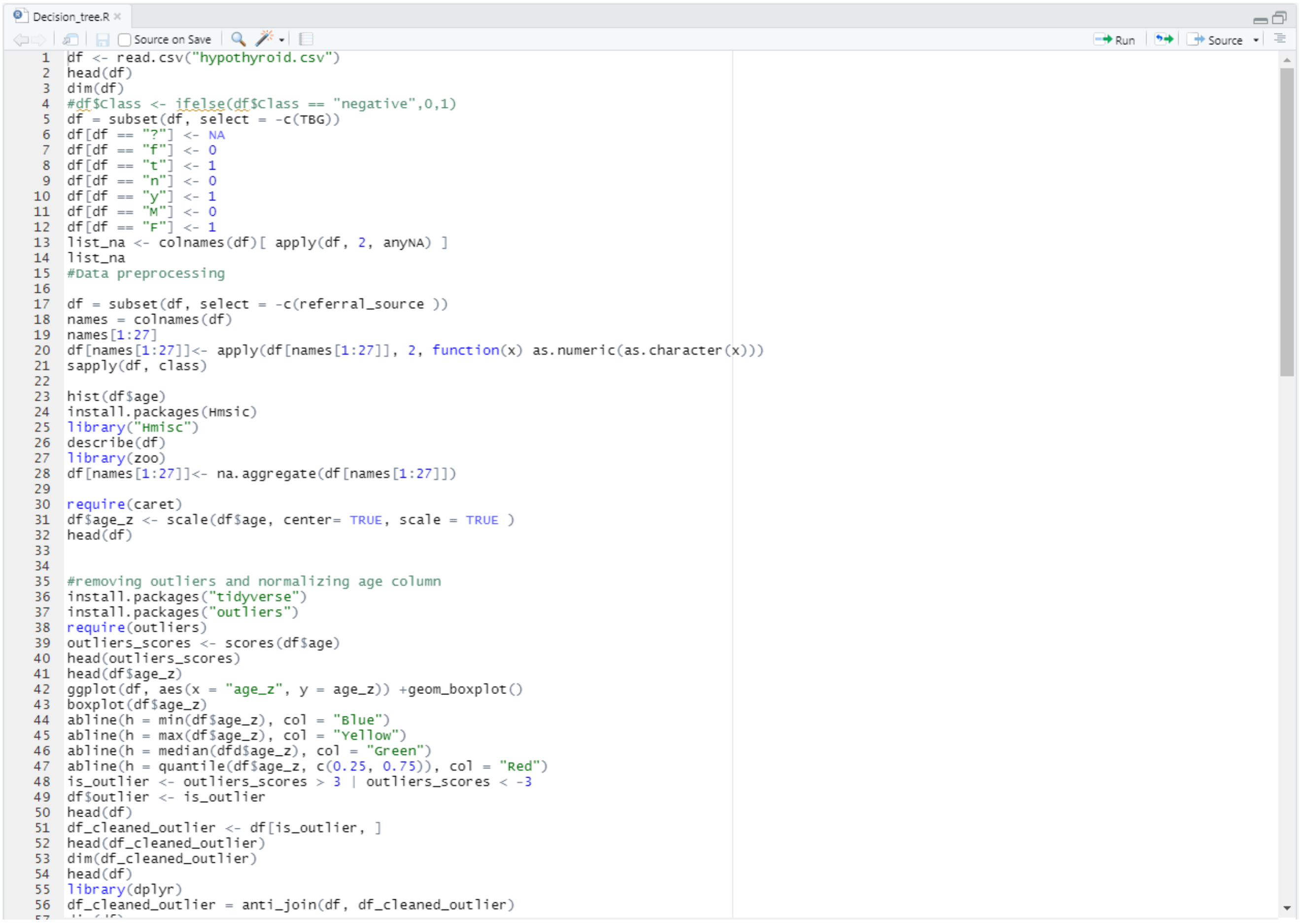
Assignment 2

PROF. BISI RUNSEWE

Mohammed Elnamory

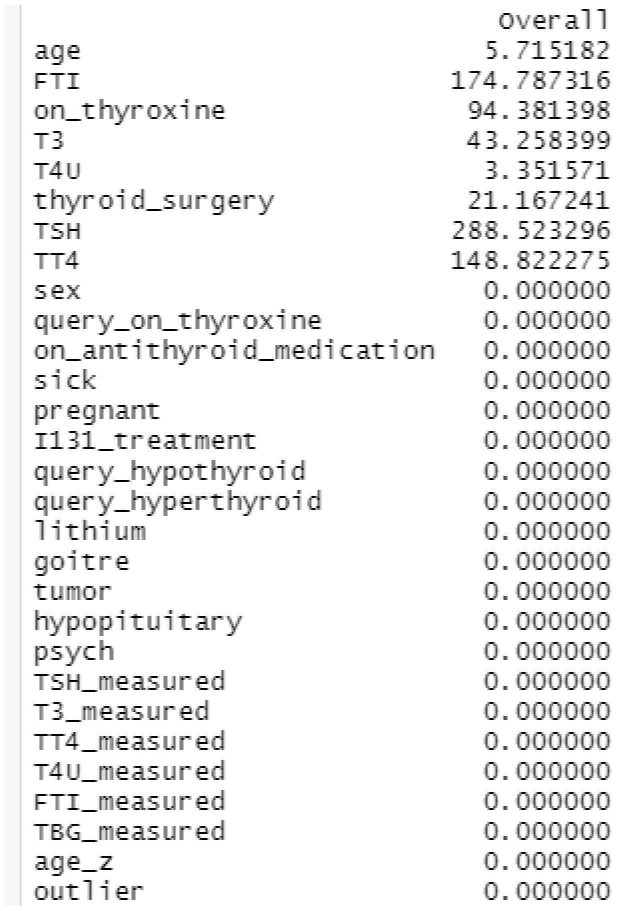
Part A

- There are some missing values in the dataset. Several strategies can be used to handle them, e.g., remove cases with unknowns. Apply one of these methods to address the missing values.
    1. **1.** When I went to check the data for NA's I fount out that the missing values was entered as "?" so first I went to change "?" to NA's values as the first early begging.
    2. **2.** Converting "?" to NA's lead to have a lot of missing data values in the dataset so I though of imputing the missing values with the mean but I faced a problem that even the numbers in the dataset were entered as characters so first I had to change the type of each column in the dataset and after that I went for imputing the missing values using "na.aggregate()"
    3. **3.** When I was checking the dataset too, I found one outlier in the age column, so I used to go for standardizing the age column and remove the outlier then using the data.



```r
df <- read.csv("hypothyroid.csv")
head(df)
dim(df)
#df$Class <- ifelse(df$Class == "negative",0,1)
df = subset(df, select = -c(TBG))
df[df == "?"] <- NA
df[df == "f"] <- 0
df[df == "t"] <- 1
df[df == "n"] <- 0
df[df == "y"] <- 1
df[df == "M"] <- 0
df[df == "F"] <- 1
list_na <- colnames(df)[ apply(df, 2, anyNA) ]
list_na
#Data preprocessing

df = subset(df, select = -c(referral_source ))
names = colnames(df)
names[1:27]
df[names[1:27]]<- apply(df[names[1:27]], 2, function(x) as.numeric(as.character(x)))
sapply(df, class)

hist(df$age)
install.packages(Hmsic)
library("Hmisc")
describe(df)
library(zoo)
df[names[1:27]]<- na.aggregate(df[names[1:27]])

require(caret)
df$age_z <- scale(df$age, center= TRUE, scale = TRUE )
head(df)


#removing outliers and normalizing age column
install.packages("tidyverse")
install.packages("outliers")
require(outliers)
outliers_scores <- scores(df$age)
head(outliers_scores)
head(df$age_z)
ggplot(df, aes(x = "age_z", y = age_z)) +geom_boxplot()
boxplot(df$age_z)
abline(h = min(df$age_z), col = "Blue")
abline(h = max(df$age_z), col = "Yellow")
abline(h = median(dfd$age_z), col = "Green")
abline(h = quantile(df$age_z, c(0.25, 0.75)), col = "Red")
is_outlier <- outliers_scores > 3 | outliers_scores < -3
df$outlier <- is_outlier
head(df)
df_cleaned_outlier <- df[is_outlier, ]
head(df_cleaned_outlier)
dim(df_cleaned_outlier)
head(df)
library(dplyr)
df_cleaned_outlier = anti_join(df, df_cleaned_outlier)
```

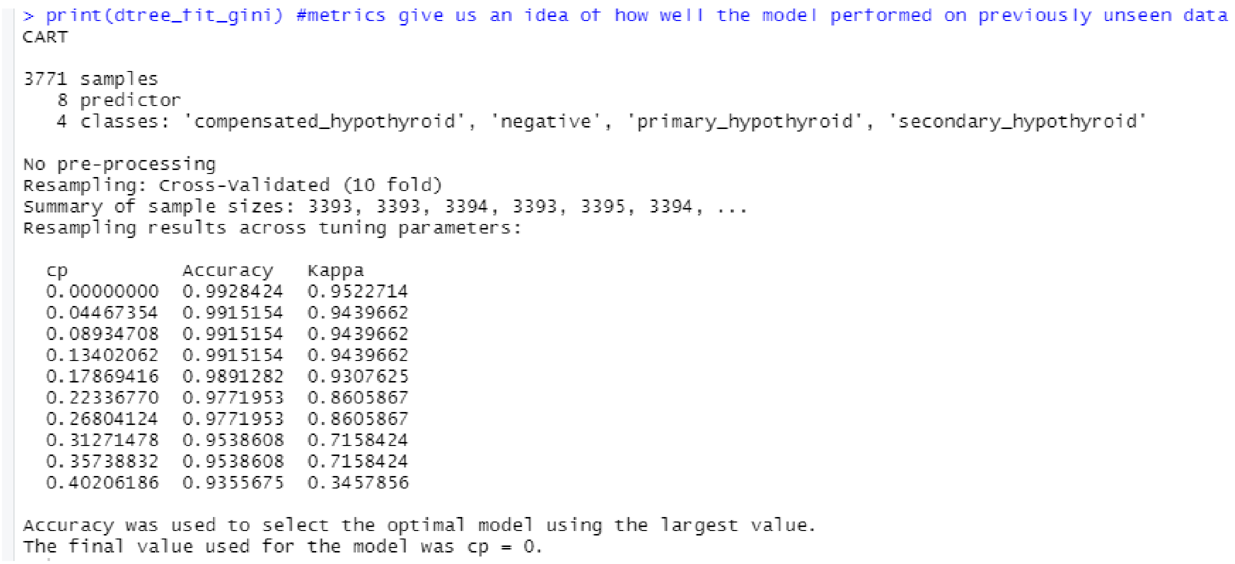
- Perform attribute selection on the dataset and state briefly why attribute selection is sometimes important.
    1. I used feature importance as my guidance, and the result came out like this.



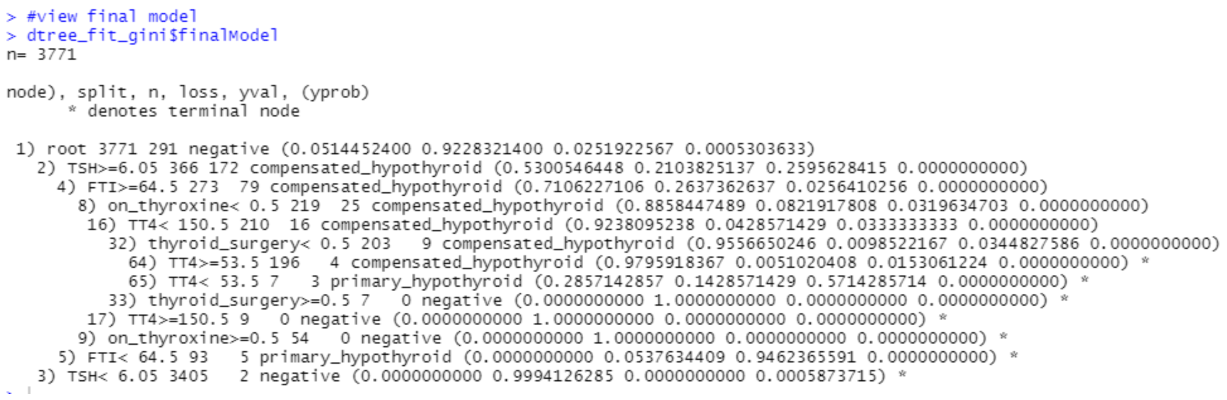
```
                              Overall
age                          5.715182
FTI                        174.787316
on_thyroxine                94.381398
T3                          43.258399
T4U                          3.351571
thyroid_surgery             21.167241
TSH                        288.523296
TT4                        148.822275
sex                          0.000000
query_on_thyroxine           0.000000
on_antithyroid_medication    0.000000
sick                         0.000000
pregnant                     0.000000
I131_treatment               0.000000
query_hypothyroid            0.000000
query_hyperthyroid           0.000000
lithium                      0.000000
goitre                       0.000000
tumor                        0.000000
hypopituitary                0.000000
psych                        0.000000
TSH_measured                 0.000000
T3_measured                  0.000000
TT4_measured                 0.000000
T4U_measured                 0.000000
FTI_measured                 0.000000
TBG_measured                 0.000000
age_z                        0.000000
outlier                      0.000000
```

This output indicates the ranking of the importance of the features, so I decided to Take “age, FTI, on_thyroxine, T3, T4U, thyroid_surgery, TSH, TT4” as my input Features of the model.

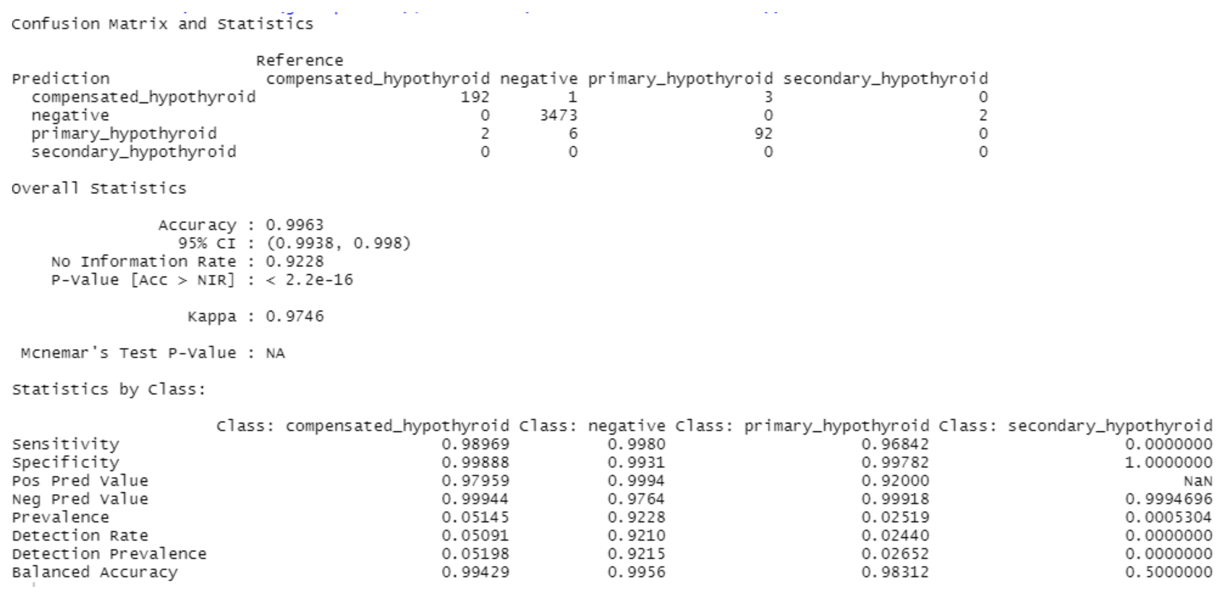
- **Split the dataset into a train and test set using k-fold cross-validation (k= 10). Create a decision tree model using the selected attributes from your dataset that can predict the type of thyroid disease a patient has.**



```
> print(dtree_fit_gini) #metrics give us an idea of how well the model performed on previously unseen data
CART

3771 samples
   8 predictor
   4 classes: 'compensated_hypothyroid', 'negative', 'primary_hypothyroid', 'secondary_hypothyroid'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3393, 3393, 3394, 3393, 3395, 3394, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.00000000  0.9928424  0.9522714
  0.04467354  0.9915154  0.9439662
  0.08934708  0.9915154  0.9439662
  0.13402062  0.9915154  0.9439662
  0.17869416  0.9891282  0.9307625
  0.22336770  0.9771953  0.8605867
  0.26804124  0.9771953  0.8605867
  0.31271478  0.9538608  0.7158424
  0.35738832  0.9538608  0.7158424
  0.40206186  0.9355675  0.3457856

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.
```



```
> #view final model
> dtree_fit_gini$finalModel
n= 3771

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 3771 291 negative (0.0514452400 0.9228321400 0.0251922567 0.0005303633)
   2) TSH>=6.05 366 172 compensated_hypothyroid (0.5300546448 0.2103825137 0.2595628415 0.0000000000)
     4) FTI>=64.5 273  79 compensated_hypothyroid (0.7106227106 0.2637362637 0.0256410256 0.0000000000)
       8) on_thyroxine< 0.5 219  25 compensated_hypothyroid (0.8858447489 0.0821917808 0.0319634703 0.0000000000)
        16) TT4< 150.5 210  16 compensated_hypothyroid (0.9238095238 0.0428571429 0.0333333333 0.0000000000)
          32) thyroid_surgery< 0.5 203   9 compensated_hypothyroid (0.9556650246 0.0098522167 0.0344827586 0.0000000000)
            64) TT4>=53.5 196   4 compensated_hypothyroid (0.9795918367 0.0051020408 0.0153061224 0.0000000000) *
            65) TT4< 53.5 7   3 primary_hypothyroid (0.2857142857 0.1428571429 0.5714285714 0.0000000000) *
          33) thyroid_surgery>=0.5 7   0 negative (0.0000000000 1.0000000000 0.0000000000 0.0000000000) *
        17) TT4>=150.5 9   0 negative (0.0000000000 1.0000000000 0.0000000000 0.0000000000) *
       9) on_thyroxine>=0.5 54   0 negative (0.0000000000 1.0000000000 0.0000000000 0.0000000000) *
     5) FTI< 64.5 93   5 primary_hypothyroid (0.0000000000 0.0537634409 0.9462365591 0.0000000000) *
   3) TSH< 6.05 3405   2 negative (0.0000000000 0.9994126285 0.0000000000 0.0005873715) *
```

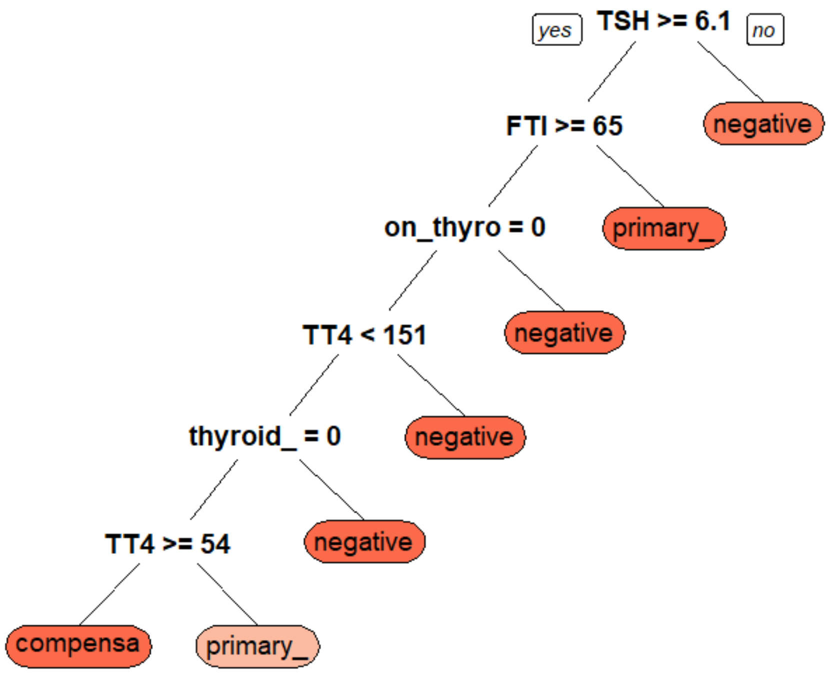


```
Confusion Matrix and Statistics

                         Reference
Prediction                compensated_hypothyroid negative primary_hypothyroid secondary_hypothyroid
  compensated_hypothyroid                     192        1                   3                     0
  negative                                      0     3473                   0                     2
  primary_hypothyroid                           2        6                  92                     0
  secondary_hypothyroid                         0        0                   0                     0

Overall Statistics

               Accuracy : 0.9963
                 95% CI : (0.9938, 0.998)
    No Information Rate : 0.9228
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9746

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: compensated_hypothyroid Class: negative Class: primary_hypothyroid Class: secondary_hypothyroid
Sensitivity                                 0.98969          0.9980                    0.96842                    0.0000000
Specificity                                 0.99888          0.9931                    0.99782                    1.0000000
Pos Pred Value                              0.97959          0.9994                    0.92000                          NaN
Neg Pred Value                              0.99944          0.9764                    0.99918                    0.9994696
Prevalence                                  0.05145          0.9228                    0.02519                    0.0005304
Detection Rate                              0.05091          0.9210                    0.02440                    0.0000000
Detection Prevalence                        0.05198          0.9215                    0.02652                    0.0000000
Balanced Accuracy                           0.99429          0.9956                    0.98312                    0.5000000
```

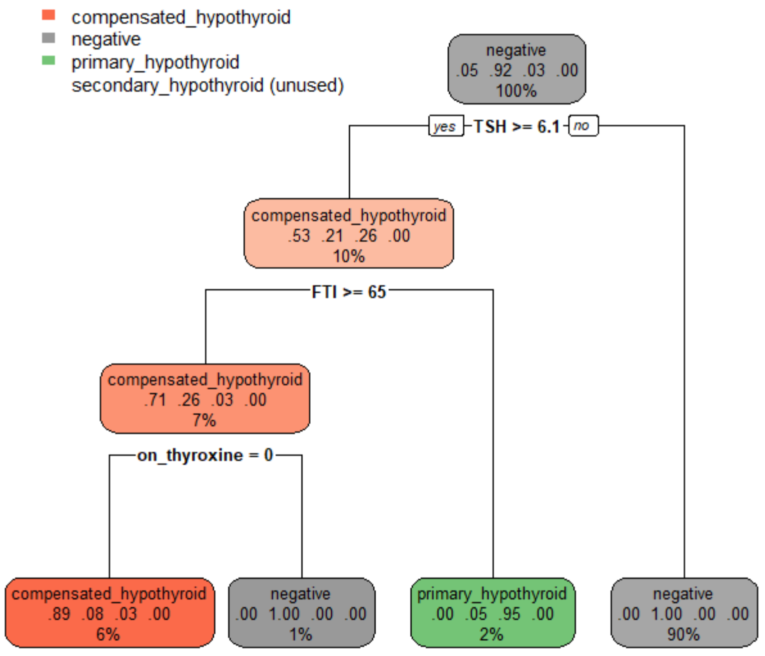
So as shown in the above output that the best accuracy came out with cp = 0 and accuracy =0.9946809, kappa = 0.9620394, Fold_number = Fold07 and this was the optimal choice for kfold.

The confusion matrix overall statistics show that the overall accuracy is 0.9963 and kappa = 0.9746

- **Visualize and describe the first few splits in the decision tree. Extract some rules.**



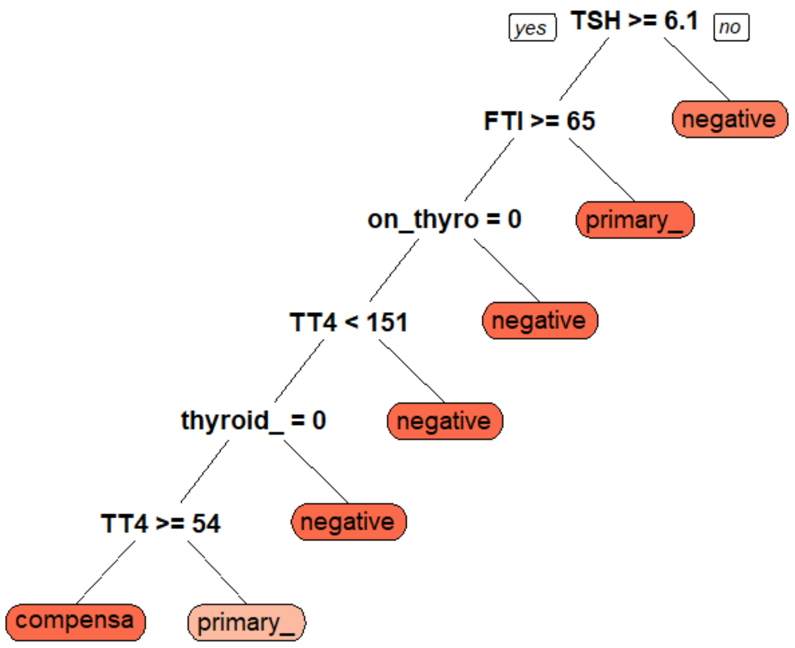
Decision Tree using geni.



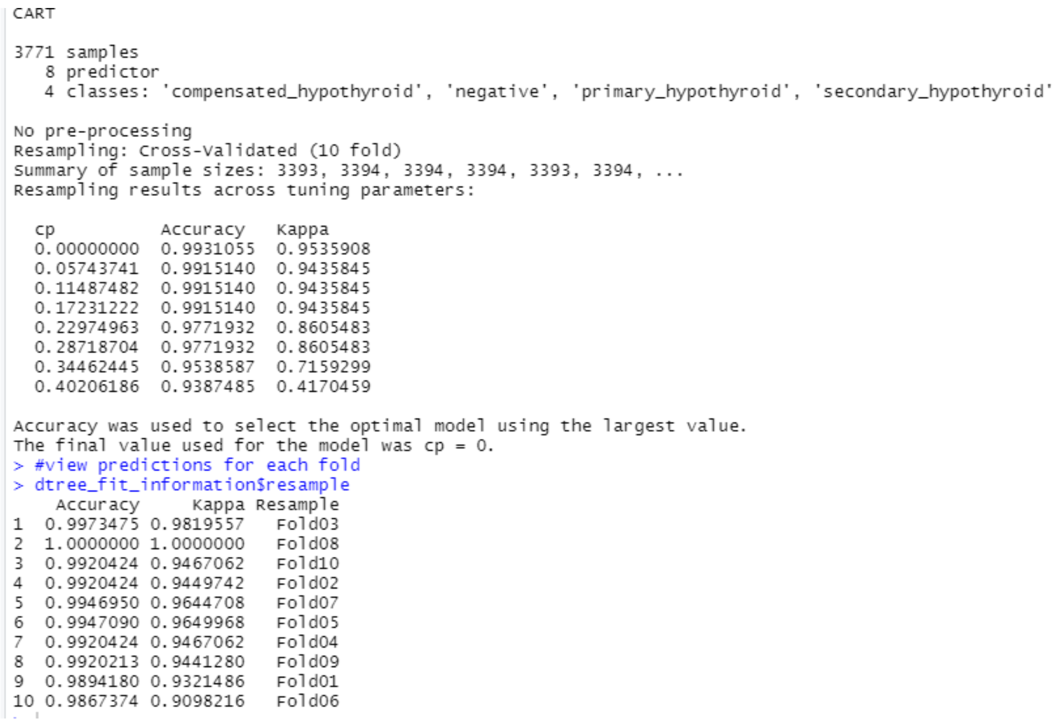
compensated_hypothyroid
negative
primary_hypothyroid
secondary_hypothyroid (unused)
negative .05 .92 .03 .00 100%
yes TSH >= 6.1 no
compensated_hypothyroid .53 .21 .26 .00 10%
FTI >= 65
compensated_hypothyroid .71 .26 .03 .00 7%
on_thyroxine = 0
compensated_hypothyroid .89 .08 .03 .00 6%
negative .00 1.00 .00 .00 1%
primary_hypothyroid .00 .05 .95 .00 2%
negative .00 1.00 .00 .00 90%

Purned Decision Tree

- **Try different ways to improve your decision tree algorithm, e.g., use different splitting strategies, prune tree after splitting.**
  when I tried to use information as in splitting strategies the output did not came out different from the geni decision tree and here are the output results.



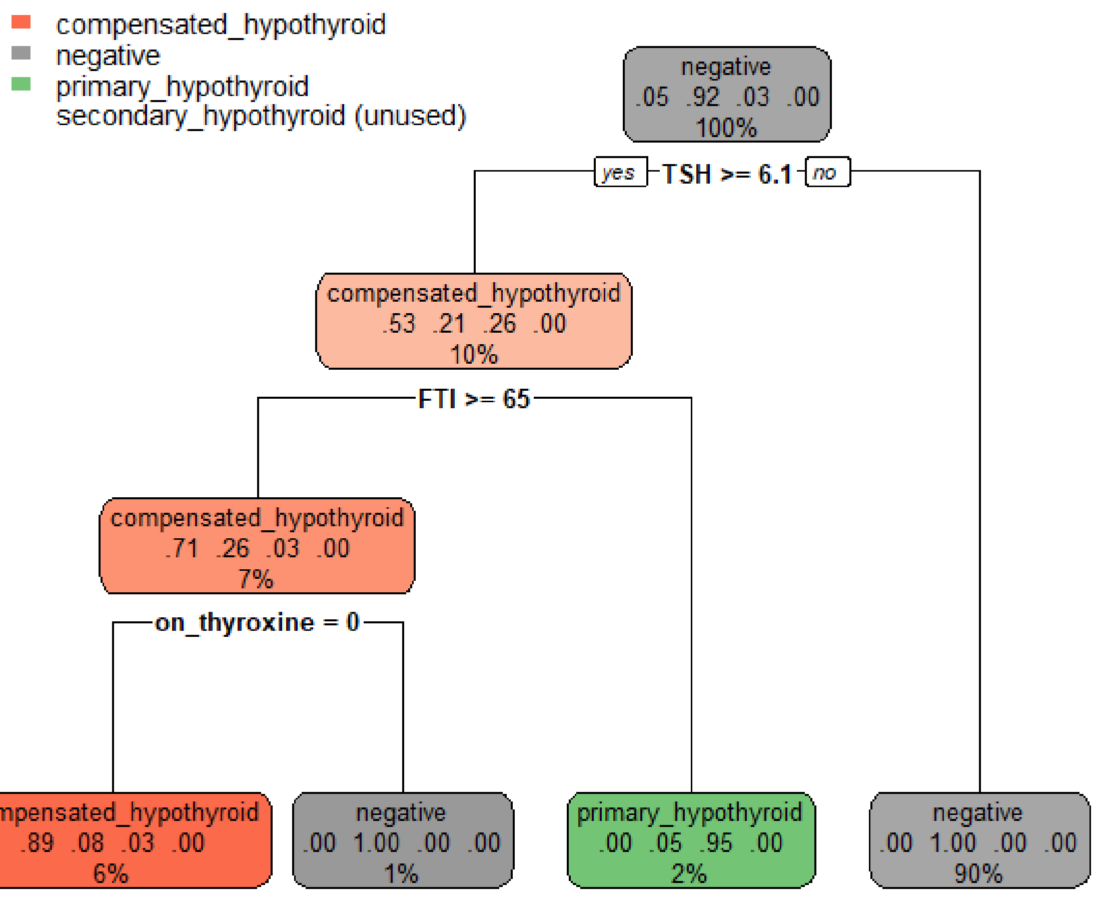
Decision Tree with information splitting method



```
CART

3771 samples
   8 predictor
   4 classes: 'compensated_hypothyroid', 'negative', 'primary_hypothyroid', 'secondary_hypothyroid'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3393, 3394, 3394, 3394, 3393, 3394, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.00000000  0.9931055  0.9535908
  0.05743741  0.9915140  0.9435845
  0.11487482  0.9915140  0.9435845
  0.17231222  0.9915140  0.9435845
  0.22974963  0.9771932  0.8605483
  0.28718704  0.9771932  0.8605483
  0.34462445  0.9538587  0.7159299
  0.40206186  0.9387485  0.4170459

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.
> #view predictions for each fold
> dtree_fit_information$resample
    Accuracy     Kappa Resample
1  0.9973475 0.9819557   Fold03
2  1.0000000 1.0000000   Fold08
3  0.9920424 0.9467062   Fold10
4  0.9920424 0.9449742   Fold02
5  0.9946950 0.9644708   Fold07
6  0.9947090 0.9649968   Fold05
7  0.9920424 0.9467062   Fold04
8  0.9920213 0.9441280   Fold09
9  0.9894180 0.9321486   Fold01
10 0.9867374 0.9098216   Fold06
```

After changing the method of splitting to information I got the optimal accuracy was for the model with cp = 0, accuracy = 0.9973475, kappa = 0.9819557, fold_number = Fold03

And the pruned tree for information splitting method came out like this.

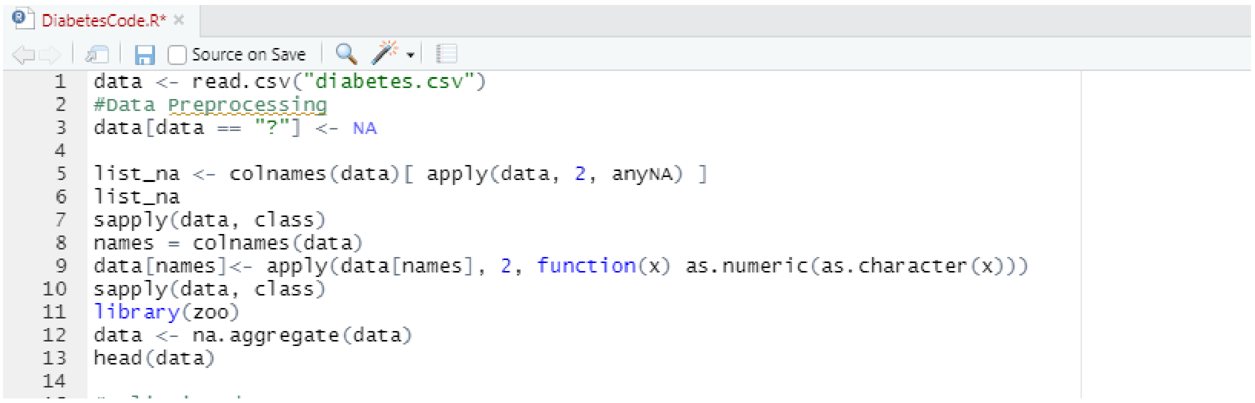


compensated_hypothyroid
negative
primary_hypothyroid
secondary_hypothyroid (unused)
negative .05 .92 .03 .00 100%
yes
TSH >= 6.1
no
compensated_hypothyroid .53 .21 .26 .00 10%
FTI >= 65
compensated_hypothyroid .71 .26 .03 .00 7%
on_thyroxine = 0
compensated_hypothyroid .89 .08 .03 .00 6%
negative .00 1.00 .00 .00 1%
primary_hypothyroid .00 .05 .95 .00 2%
negative .00 1.00 .00 .00 90%

Pruned information decision tree.

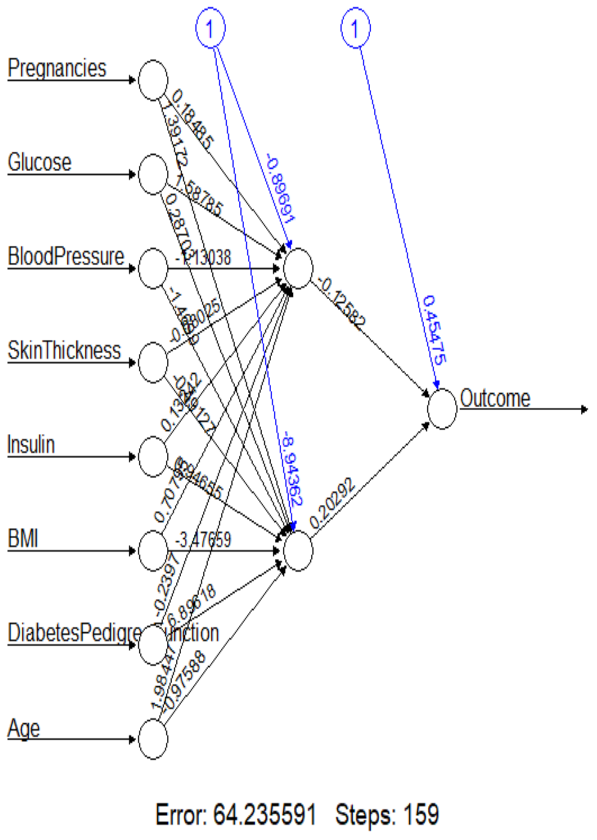
Part B “support vector Machine”

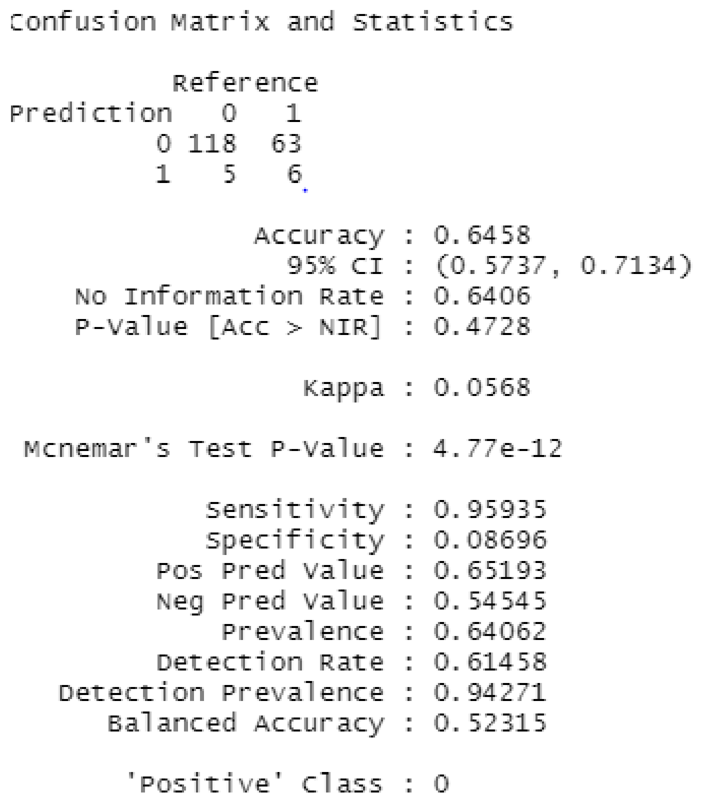
- Some data points are not available, handle the missing data by applying central measure of tendency to derive the missing value.
    1. When I went to check the data for NA’s I fount out that the missing values was entered as “?” so first I went to change “?” to NA’s values as the first early begging.
    2. Converting “?” to NA’s lead to have a lot of missing data values in the dataset so I though of imputing the missing values with the mean but I faced a problem that even the numbers in the dataset were entered as characters so first I had to change the type of each column in the dataset and after that I went for imputing the missing values using “na.aggregate()”


DiabetesCode.R*

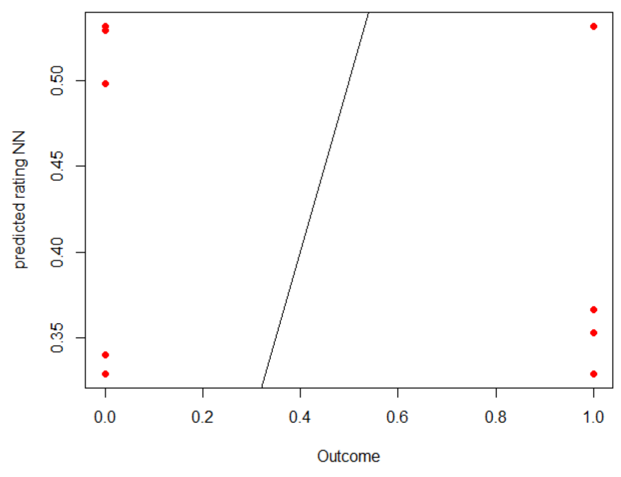
```r
data <- read.csv("diabetes.csv")
#Data Preprocessing
data[data == "?"] <- NA

list_na <- colnames(data)[ apply(data, 2, anyNA) ]
list_na
sapply(data, class)
names = colnames(data)
data[names]<- apply(data[names], 2, function(x) as.numeric(as.character(x)))
sapply(data, class)
library(zoo)
data <- na.aggregate(data)
head(data)

```

- Partition the dataset into a train dataset (75%) and test dataset (25%). Use the train dataset to build the Neural Network and the test dataset to evaluate how well the model generalizes to future results.

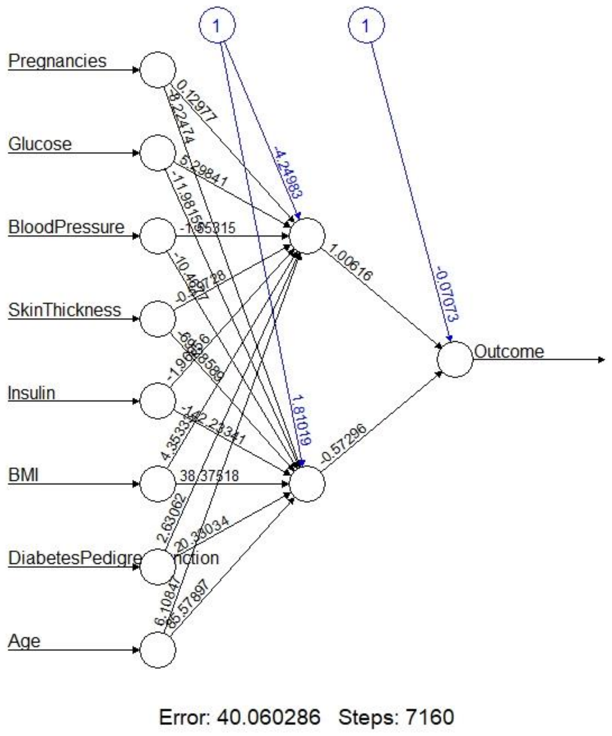




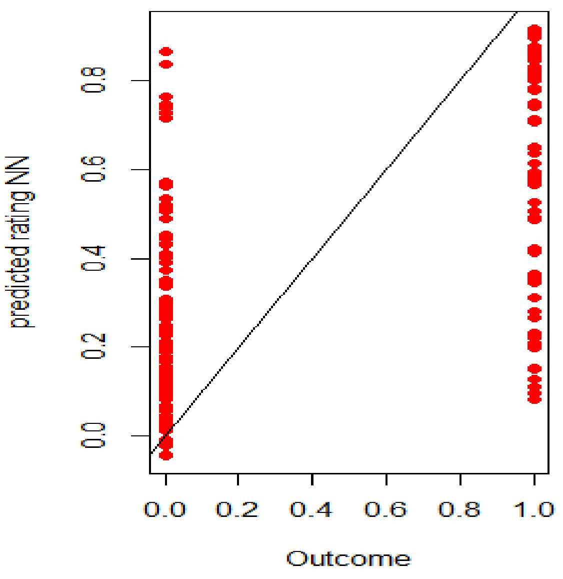
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 118  63
         1   5   6

               Accuracy : 0.6458
                 95% CI : (0.5737, 0.7134)
    No Information Rate : 0.6406
    P-Value [Acc > NIR] : 0.4728

                  Kappa : 0.0568

 Mcnemar's Test P-Value : 4.77e-12

            Sensitivity : 0.95935
            Specificity : 0.08696
         Pos Pred Value : 0.65193
         Neg Pred Value : 0.54545
             Prevalence : 0.64062
         Detection Rate : 0.61458
   Detection Prevalence : 0.94271
      Balanced Accuracy : 0.52315

       'Positive' Class : 0
```

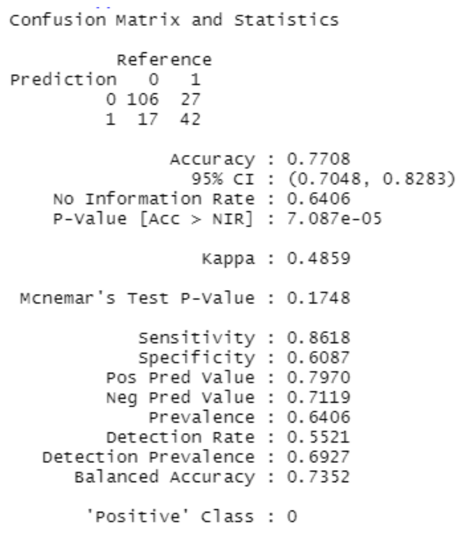


- **Neural networks work best when the input data are scaled to a narrow range around zero. Rescale the data with a normalizing (e.g., min_max normalization) or standardization (e.g., z_score standardization) function.**



After normalizing the data It came out with the best accuracy out of all models I have tried so it's the best champion.

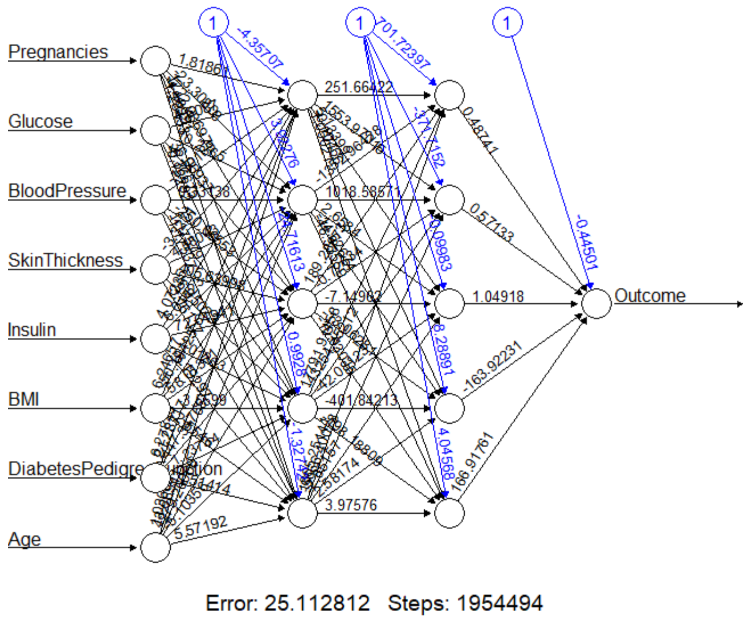


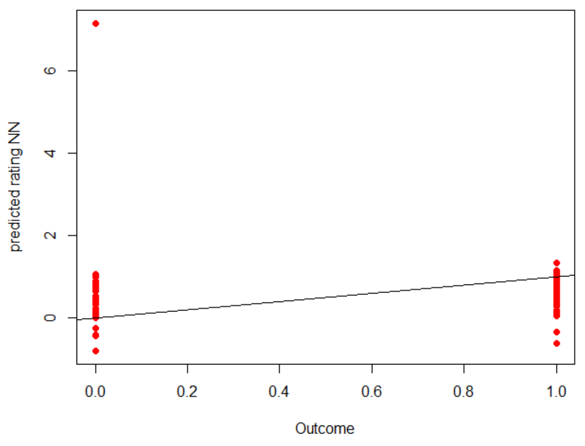


```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 106  27
         1  17  42

               Accuracy : 0.7708
                 95% CI : (0.7048, 0.8283)
    No Information Rate : 0.6406
    P-Value [Acc > NIR] : 7.087e-05

                  Kappa : 0.4859

 Mcnemar's Test P-Value : 0.1748

            Sensitivity : 0.8618
            Specificity : 0.6087
         Pos Pred Value : 0.7970
         Neg Pred Value : 0.7119
             Prevalence : 0.6406
         Detection Rate : 0.5521
   Detection Prevalence : 0.6927
      Balanced Accuracy : 0.7352

       'Positive' Class : 0
```

- **Train & plot a simple Neural Network with only 2 hidden nodes (not layer). Then, train & plot a multilayer perceptron with 2 layers & 5 nodes. What impact does the change in the number of layers & nodes have on the accuracy of your model?**

  The model performed worse when the 2 layers were added as it scored an accuracy of 0.69 less than the model with two nodes and scaled data as input the results came like the next figures.



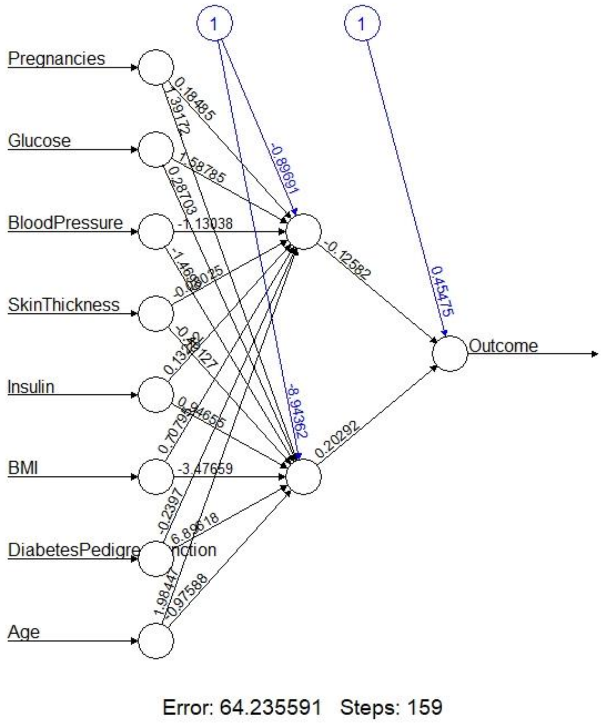


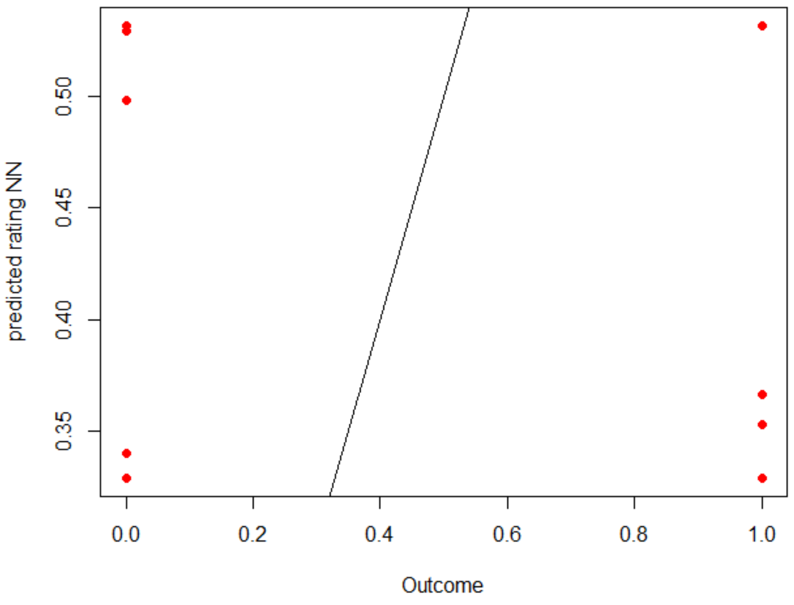
```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 99 34
         1 24 35

               Accuracy : 0.6979
                 95% CI : (0.6277, 0.7619)
    No Information Rate : 0.6406
    P-Value [Acc > NIR] : 0.05576

                  Kappa : 0.3224

 Mcnemar's Test P-Value : 0.23730

            Sensitivity : 0.8049
            Specificity : 0.5072
         Pos Pred Value : 0.7444
         Neg Pred Value : 0.5932
             Prevalence : 0.6406
         Detection Rate : 0.5156
   Detection Prevalence : 0.6927
      Balanced Accuracy : 0.6561

       'Positive' Class : 0
```

- **Try changing the activation function, varying the learning rate, epochs or removing the bias. What effects does any of these have on the result?**

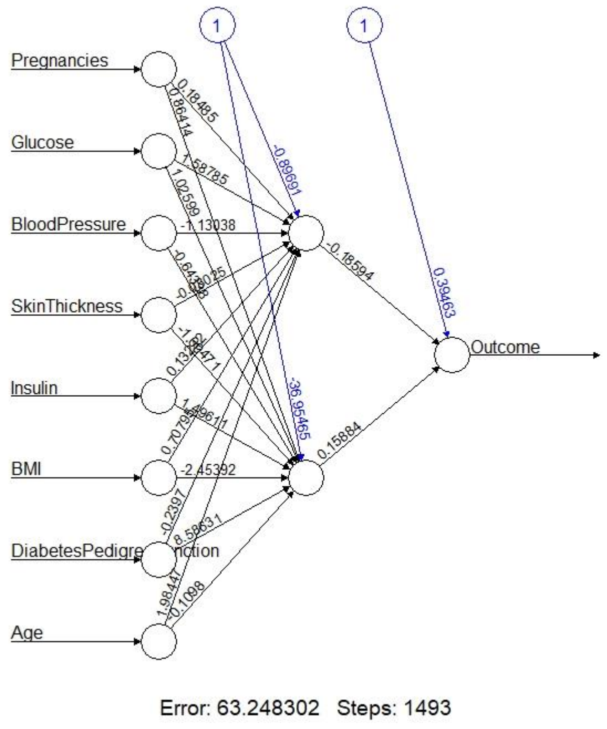
  I tried to change the learning rate one time and the other change was choosing another activate function tanh and compared the results.

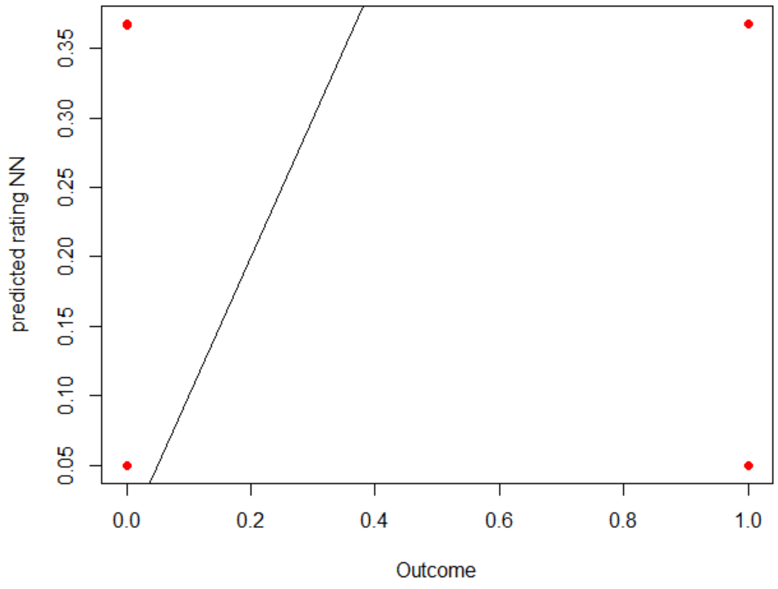
  1- Learning rate up to 0.5



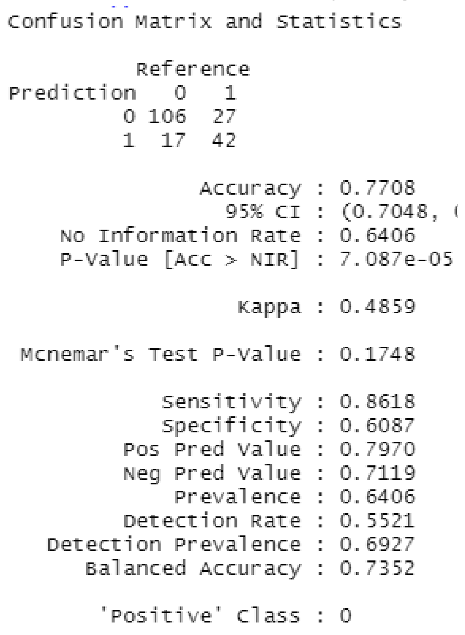


2- Tanh Activatio

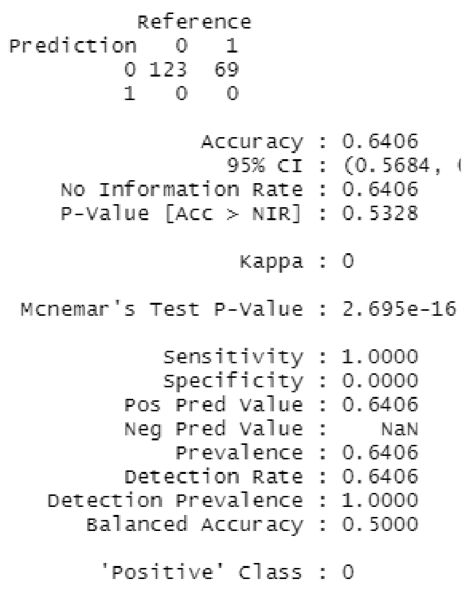




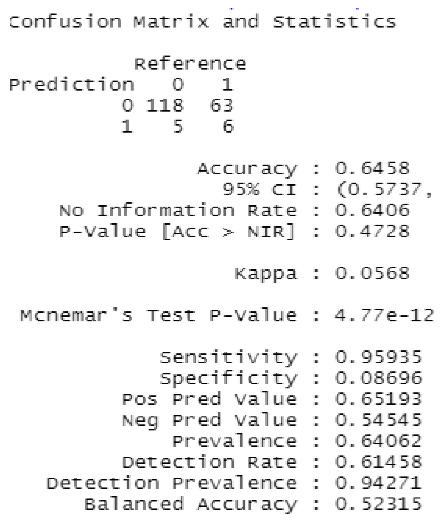
3- Comparing the confusion matrices of the three neural networks



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 106  27
         1  17  42

               Accuracy : 0.7708
                 95% CI : (0.7048, (
    No Information Rate : 0.6406
    P-Value [Acc > NIR] : 7.087e-05

                  Kappa : 0.4859

 Mcnemar's Test P-Value : 0.1748

            Sensitivity : 0.8618
            Specificity : 0.6087
         Pos Pred Value : 0.7970
         Neg Pred Value : 0.7119
             Prevalence : 0.6406
         Detection Rate : 0.5521
   Detection Prevalence : 0.6927
      Balanced Accuracy : 0.7352

       'Positive' Class : 0
```



```
          Reference
Prediction   0   1
         0 123  69
         1   0   0

               Accuracy : 0.6406
                 95% CI : (0.5684, (
    No Information Rate : 0.6406
    P-Value [Acc > NIR] : 0.5328

                  Kappa : 0

 Mcnemar's Test P-Value : 2.695e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.6406
         Neg Pred Value :    NaN
             Prevalence : 0.6406
         Detection Rate : 0.6406
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0
```



```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 118  63
         1   5   6

               Accuracy : 0.6458
                 95% CI : (0.5737,
    No Information Rate : 0.6406
    P-Value [Acc > NIR] : 0.4728

                  Kappa : 0.0568

 Mcnemar's Test P-Value : 4.77e-12

            Sensitivity : 0.95935
            Specificity : 0.08696
         Pos Pred Value : 0.65193
         Neg Pred Value : 0.54545
             Prevalence : 0.64062
         Detection Rate : 0.61458
   Detection Prevalence : 0.94271
      Balanced Accuracy : 0.52315
```

First scaled Neural Network

Tanh Neural Network

0.5 learning rate confusion matrix

As shown in the above table changing the learning rate or even choosing tanh as my activation function affected the accuracy of the neural network but it did not improve it has fallen from 0.7708 in the normal neural network to 0.64 and approximately the same in the tanh with accuracy 0.6458.

- **Conclusion:**

Making the neural network more complex or changing the activation function as the model champion here was the one with one layer and two nodes but normalizing the dataset guarantee better results most of time also using the mean as imputation method can help but the imputation of missing value using median can help within dataset full of outliers.