
Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique


University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

# *2021 Summer DTI 5126:*

# *Fundamentals for Applied Data Science*

## *Assignment 2*

*Reyad Melies*

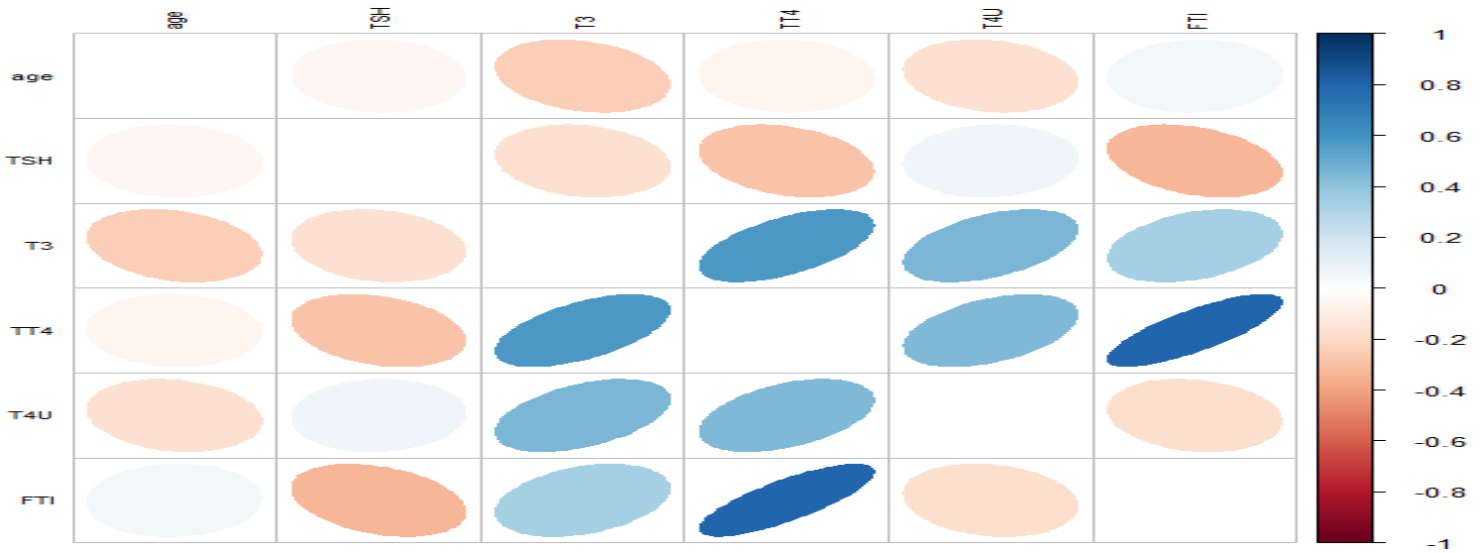
# Part A: Decision Trees:

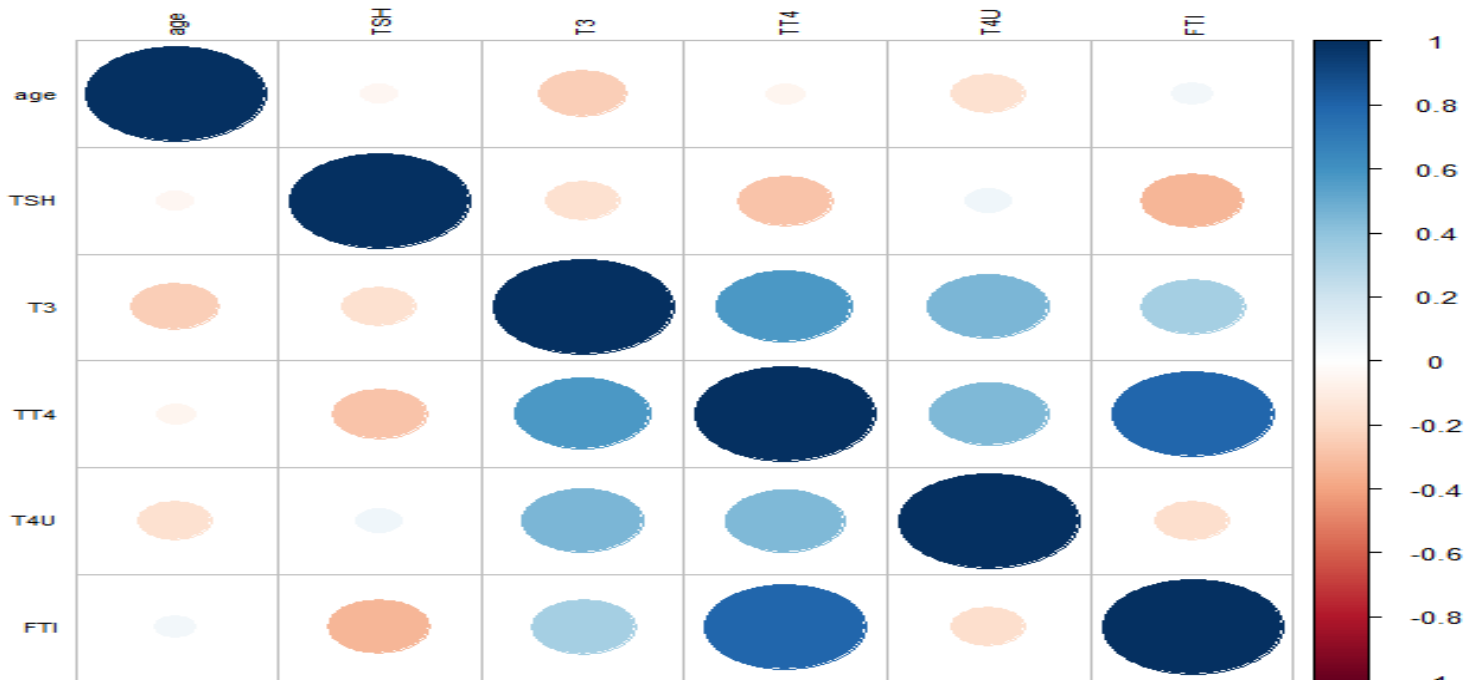
a. There are some missing values in the dataset. Several strategies can be used to handle them, e.g., remove cases with unknowns. Apply one of these methods to address the missing values.

Drop TGB as it all contains NAN to be able to remove rows that contains NAN.

```
#########################################################
dataset[dataset == '?'] <- NA
dataset = dataset[,!(names(dataset) %in% c("TBG"))]
dataset=dataset %>% drop_na()
#########################################################
```

b. Perform attribute selection on the dataset and state briefly why attribute selection is sometimes important.



```
      age              sex          on_thyroxine query_on_thyroxine on_antithyroid_medication sick       pregnant thyroid_surgery I131_treatment
 Min.   :  1.00   ?:   0   f:2377       f:2619           f:2610                  f:2530   f:2602   f:2610         f:2603
 1st Qu.: 37.00   F:1764   t: 266       t:  24           t:  33                  t: 113   t:  41   t:  33         t:  40
 Median : 55.00   M: 879
 Mean   : 53.08
 3rd Qu.: 69.00
 Max.   :455.00
 query_hypothyroid query_hyperthyroid lithium  goitre   tumor    hypopituitary psych    TSH_measured      TSH              T3_measured       T3
 f:2497            f:2480             f:2629   f:2623   f:2585   f:2642        f:2473   f:   0       Min.   :  0.005   f:   0       Min.   : 0.050
 t: 146            t: 163             t:  14   t:  20   t:  58   t:   1        t: 170   t:2643       1st Qu.:  0.500   t:2643       1st Qu.: 1.500
                                                                                                     Median :  1.300                Median : 2.000
                                                                                                     Mean   :  5.036                Mean   : 2.001
                                                                                                     3rd Qu.:  2.600                3rd Qu.: 2.300
                                                                                                     Max.   :530.000                Max.   :10.600
 TT4_measured      TT4              T4U_measured     T4U              FTI_measured     FTI              TBG_measured  referral_source                 Class
 f:   0       Min.   :  2.0   f:   0       Min.   :0.2500   f:   0       Min.   :  2.0   f:2643        other:1237      compensated_hypothyroid: 136
 t:2643       1st Qu.: 88.0   t:2643       1st Qu.:0.8700   t:2643       1st Qu.: 93.0                 STMW :  84      negative                 :2427
              Median :103.0                Median :0.9800                Median :107.0                 SVHC : 359      primary_hypothyroid      :  79
              Mean   :107.9                Mean   :0.9957                Mean   :109.4                 SVHD :  34      secondary_hypothyroid    :   1
              3rd Qu.:124.0                3rd Qu.:1.0900                3rd Qu.:124.0                 SVI  : 929
              Max.   :430.0                Max.   :2.1200                Max.   :395.0
```

1, 2: Remove outliers

9 : one record in label that could never be detected

```
#remove outliers
dataset=dataset[dataset$age != 455, ]
dataset=dataset[dataset$hypopituitary != "t", ]

dataset=dataset[dataset$Class != "secondary_hypothyroid", ]
```

3,4,5,6,7,8: always true columns

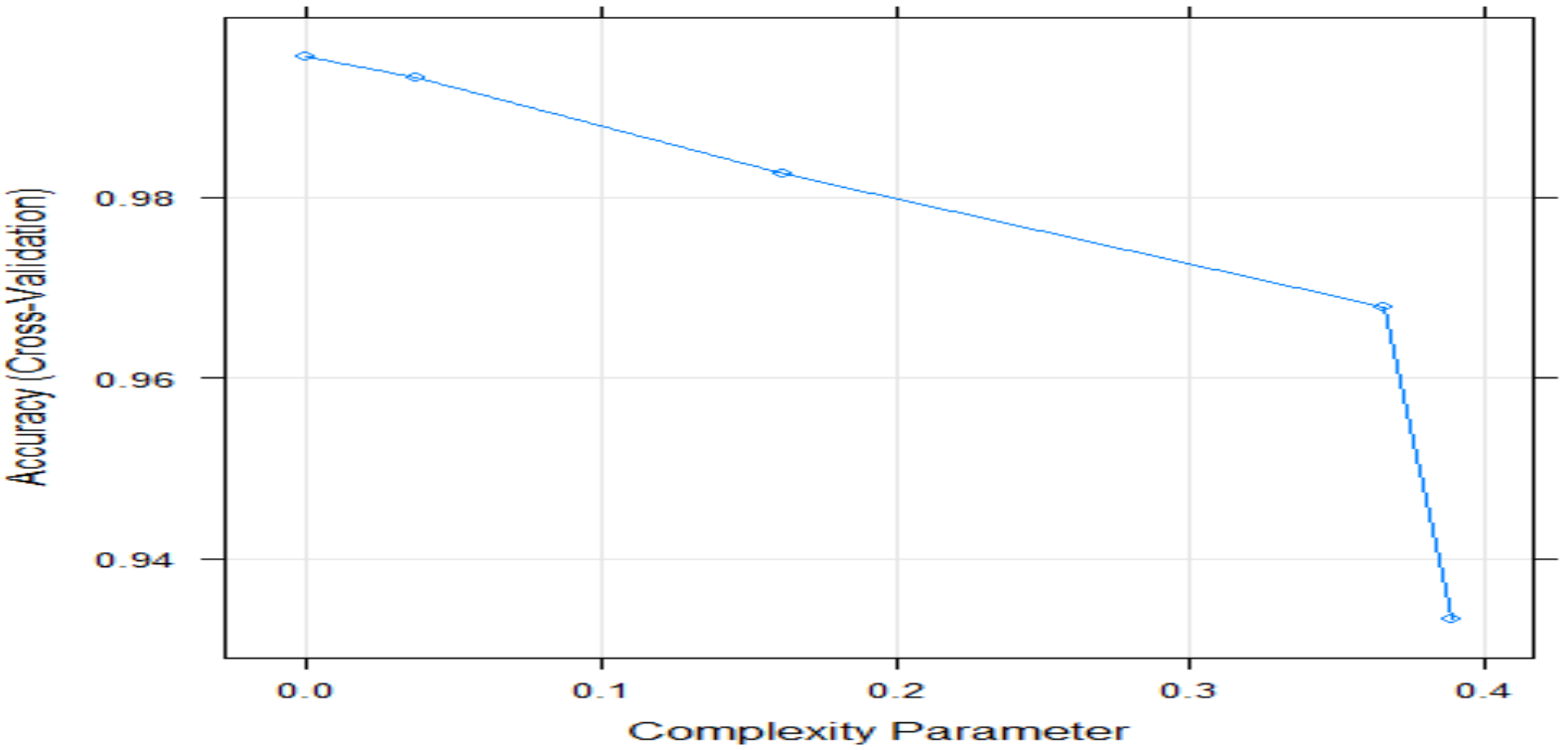
```
#drop these columns
dataset = dataset[,!(names(dataset) %in%
                       c('hypopituitary',"TSH_measured","T3_measured",
                         "TT4_measured","T4U_measured","FTI_measured","TBG_measured"))]
```

There is high correlation between FTI and TT4 but not above 80%

Attribute selection is important because we train the model with only important features and not to overload the model with unimportant ones that could cause the model to need more computational power and more time and can even cause false and wrong prediction.

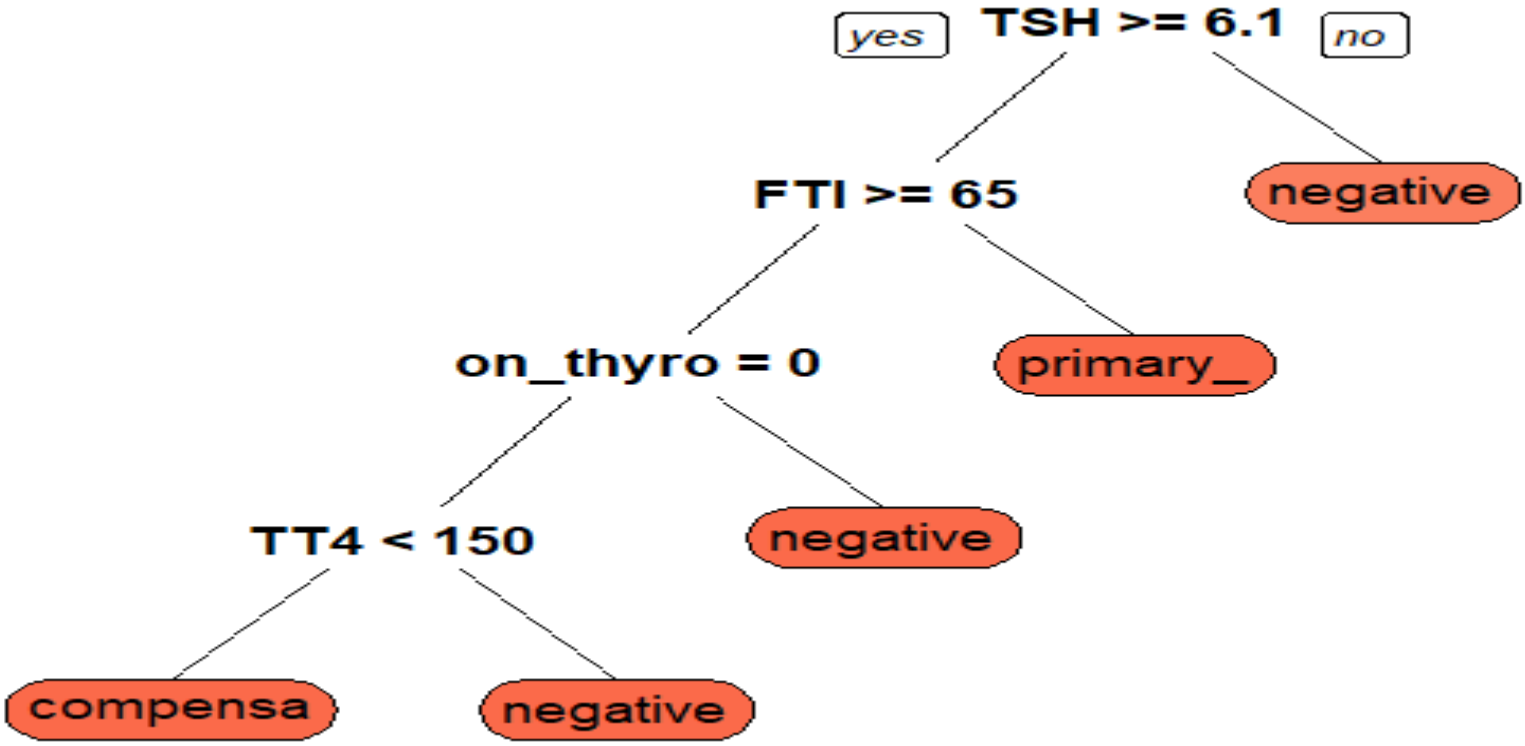
c. Split the dataset into a train and test set using k-fold cross-validation (k= 10). Create a decision tree model using the selected attributes from your dataset that can predict the type of thyroid disease a patient has.

```
#############################################################Splitting
set.seed(123) #generates a reproducible random sampling
ctrl <- trainControl(method = "cv", number = 10)
#fit a decision tree model and use k-fold CV to evaluate performance
dtree_fit_gini <- train(Class~., data = dataset, method = "rpart", parms = list(split = "gini"), trControl = ctrl, tuneLength = 5)
#############################################
```




| | Accuracy | Kappa | Resample |
|---|---|---|---|
| 1 | 0.9962264 | 0.9752336 | Fold09 |
| 2 | 1.0000000 | 1.0000000 | Fold04 |
| 3 | 1.0000000 | 1.0000000 | Fold06 |
| 4 | 0.9924528 | 0.9534271 | Fold01 |
| 5 | 0.9962121 | 0.9762291 | Fold02 |
| 6 | 0.9962264 | 0.9762353 | Fold10 |
| 7 | 0.9962121 | 0.9762162 | Fold05 |
| 8 | 0.9962121 | 0.9762162 | Fold07 |
| 9 | 0.9924242 | 0.9534145 | Fold08 |
| 10 | 0.9962121 | 0.9752089 | Fold03 |

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




When the values of TSH for every record is less than 6.1 all cases are considered negative without even considering the rest of attributes. Moreover, if TSH is greater than or equal 6.1 and the FTI is less than 65 it is considered primary. However, if FTI is greater than or equal 65 it needs to observe another attribute to make decision so next In line is on _thyro where it is considered not equal zero, it is considered negative. Last but not least, if TSH is greater than or equal 6.1 , FTI is greater than or equal 65, and _thyro equal zero and TT4 less than 150 it is considered compensa else it is greater than TT4 it is considered negative.

e. Try different ways to improve your decision tree algorithm, e.g., use different splitting strategies, prune tree after splitting.

## Tried Reduction and CHI as splitting strategies instead of GINI and pruning trees:

```
#########################################################################    | Reduction
dtree_fit_reduction <- train(Class~., data = dataset, method = "rpart", parms = list(split = 'reduction'), trControl = ctrl, tuneLength = 10)
#Step 5: Evaluate - view summary of k-fold CV
print(dtree_fit_reduction) #metrics give us an idea of how well the model performed on previously unseen data

#view final model
dtree_fit_reduction$finalModel
prp(dtree_fit_reduction$finalModel, box.palette = "Reds", tweak = 1.2) #view the tree using prop() function

#view predictions for each fold
dtree_fit_reduction$resample
#########################################################################      CHI
dtree_fit_chi <- train(Class~., data = dataset, method = "rpart", parms = list(split = 'reduction'), trControl = ctrl, tuneLength = 10)
#Step 5: Evaluate - view summary of k-fold CV
print(dtree_fit_chi) #metrics give us an idea of how well the model performed on previously unseen data

#view final model
dtree_fit_chi$finalModel
prp(dtree_fit_chi$finalModel, box.palette = "Reds", tweak = 1.2) #view the tree using prop() function

#view predictions for each fold
dtree_fit_chi$resample
#########################################################################      Puring
```

## Reduction

```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2375, 2378, 2377, 2377, 2378, 2378, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.00000000  0.9954645  0.9715088
  0.04320988  0.9924284  0.9527827
  0.08641975  0.9924284  0.9527827
  0.12962963  0.9924284  0.9527827
  0.17283951  0.9799483  0.8842970
  0.21604938  0.9791907  0.8799199
  0.25925926  0.9791907  0.8799199
  0.30246914  0.9791907  0.8799199
  0.34567901  0.9791907  0.8799199
  0.38888889  0.9276907  0.2685085

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

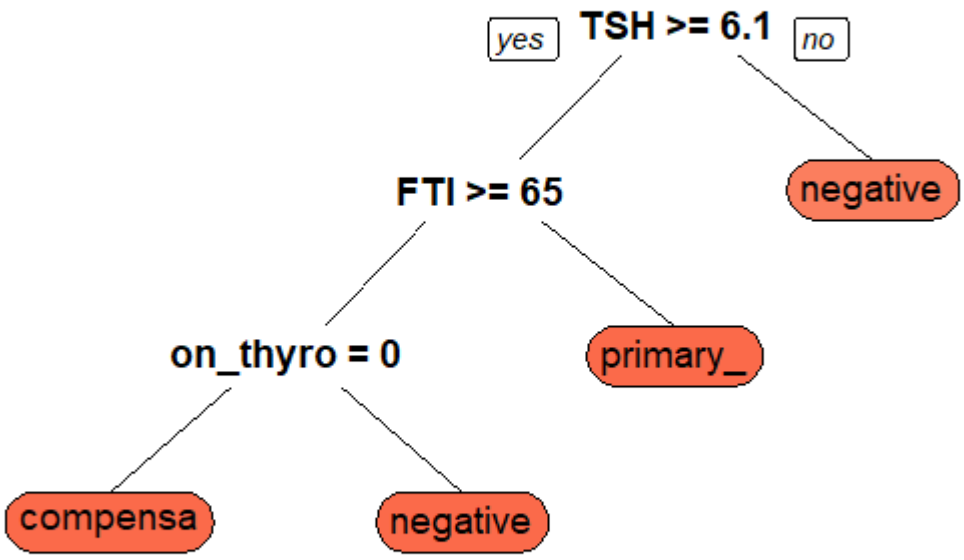
## CHI

```
cp          Accuracy   Kappa
0.00000000  0.9962077  0.9759050
0.04320988  0.9931860  0.9574470
0.08641975  0.9931860  0.9574470
0.12962963  0.9931860  0.9574470
0.17283951  0.9803128  0.8857117
0.21604938  0.9799340  0.8834303
0.25925926  0.9799340  0.8834303
0.30246914  0.9799340  0.8834303
0.34567901  0.9799340  0.8834303
0.38888889  0.9284509  0.2715964

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

CHI produces slightly better accuracy than reduction algorithm but not better than Gini.

# Pruning

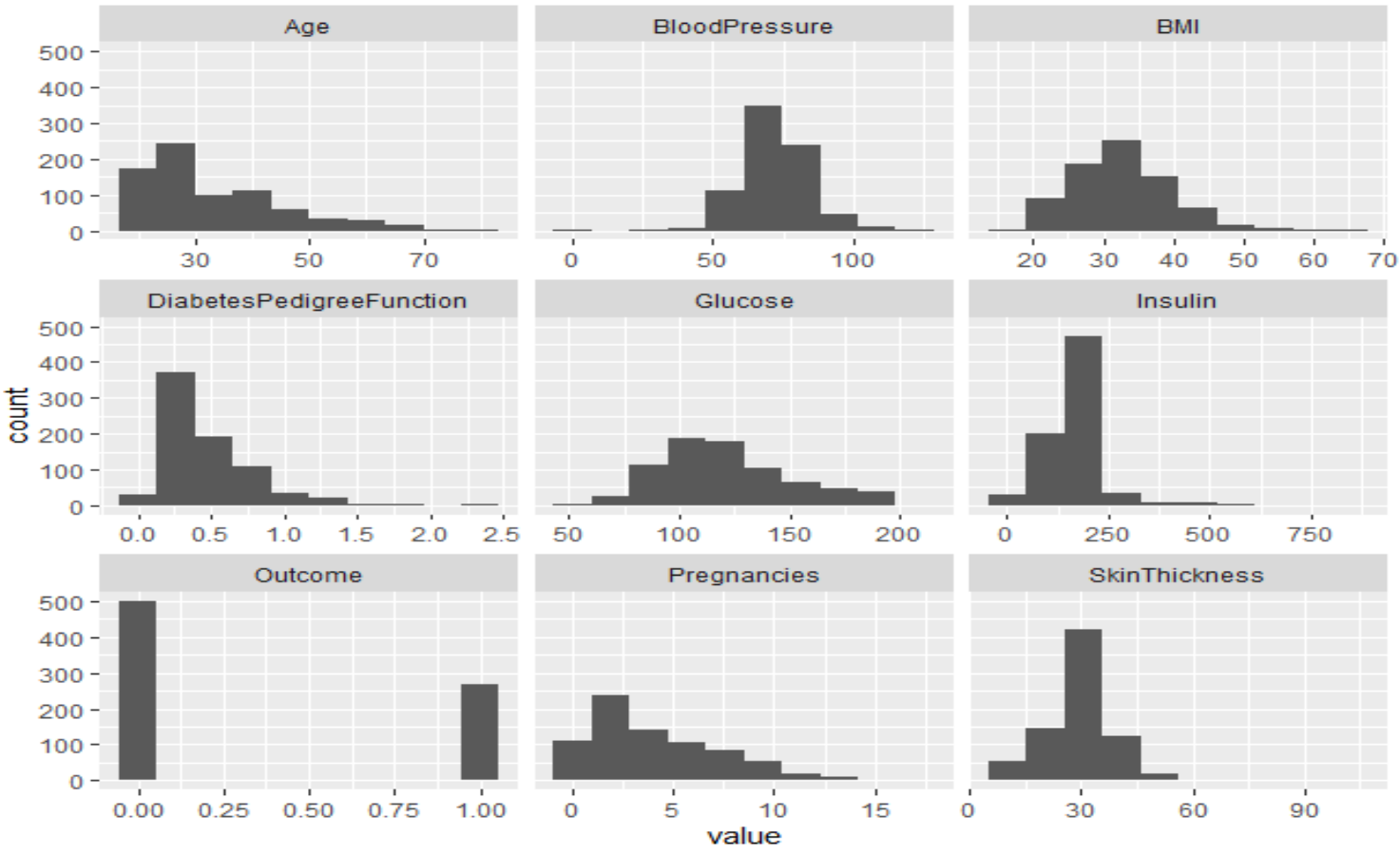
```
##############################################################################Puring
prune=prune.rpart(dtree_fit_gini$finalModel,0.1)
prp(prune, box.palette = "Reds", tweak = 1.2) #view the tree using prop() function
##############################################################################
```



# Part B: Support Vector Machines

a. Some data points are not available, handle the missing data by applying central measure of tendency to derive the missing value.

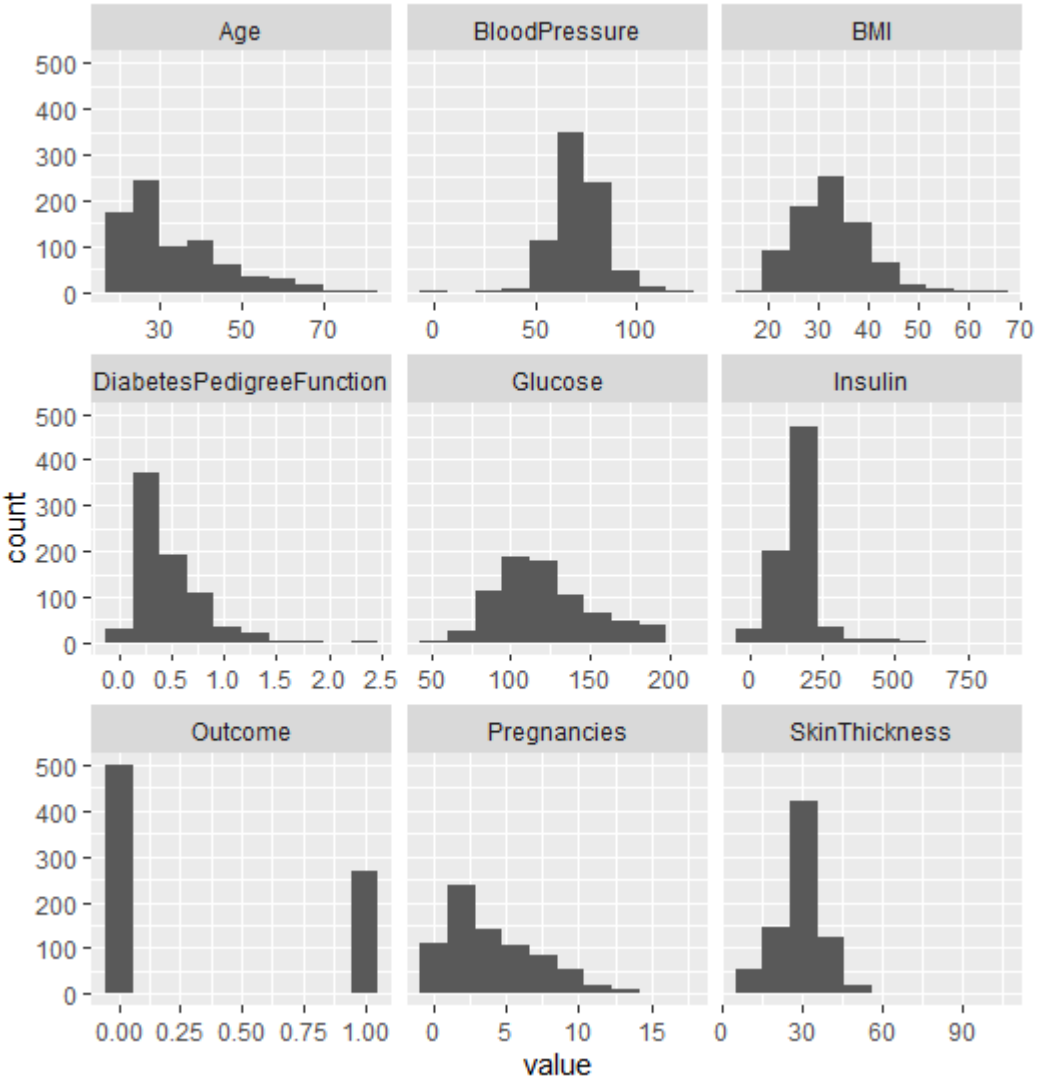
## Original Data




Replace with Mean for Glucose , Skin Thickness and BMI (Normally Distributed)

Median for Pregnancies, age, Diabetes Pedigree Function, and Insulin (Skewness)

```
dataset[ , 1][is.na(dataset[ , 1])] <- median(dataset[ , i], na.rm = TRUE)
dataset[ , 4][is.na(dataset[ , 4])] <- median(dataset[ , i], na.rm = TRUE)
dataset[ , 8][is.na(dataset[ , 8])] <- median(dataset[ , i], na.rm = TRUE)
dataset=na_mean(dataset)
```




b. 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.

```
set.seed(42)
sample_split <- sample.split(Y = dataset, SplitRatio = 0.75)
train_set <- subset(x = dataset, sample_split == TRUE)
test_set <- subset(x = dataset, sample_split == FALSE)
```

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

```
#########################################################################
max = apply(dataset , 2 , max)
min = apply(dataset, 2 , min)
dataset = as.data.frame(scale(dataset, center = min, scale = max - min))
dataset
#########################################################################
```

d. 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?

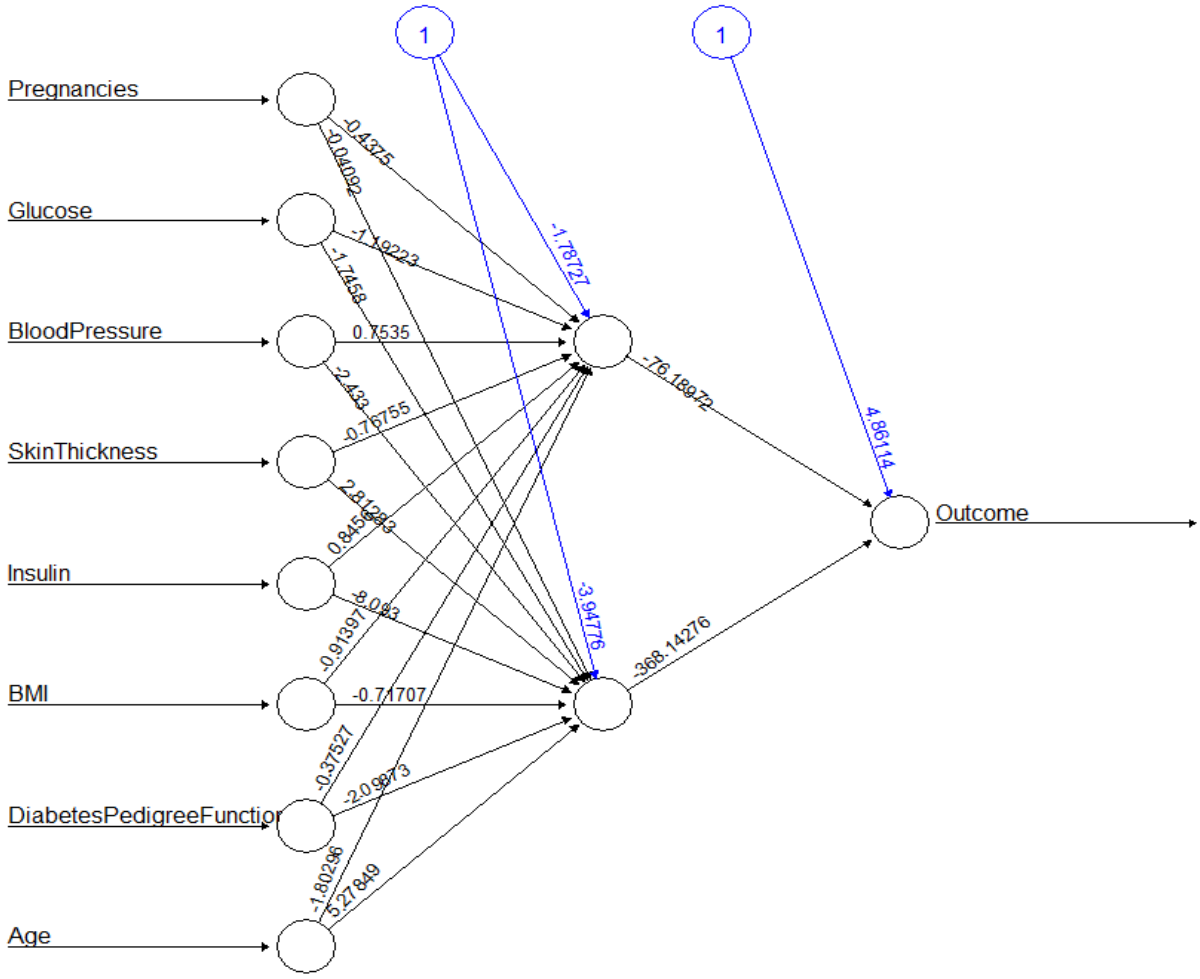
## 2 nodes one hidden layer:

Accuracy:

0.7276265

```
predict_testNN
   0   1
0 134  27
1  43  53
```

```
NN = neuralnet(Outcome
               ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
               train_set, hidden = 2,linear.output = FALSE, err.fct = 'ce',likelihood = TRUE,stepmax = 9999999999)

plot(NN, rep = 'best')

predict_testNN = compute(NN, test_set[,c(1:8)])
predict_testNN = round(predict_testNN$net.result)
predict_testNN <- sapply(predict_testNN,as.numeric)


table(test_set$Outcome,predict_testNN)
accuracy <- mean(test_set$Outcome == predict_testNN)
accuracy
```



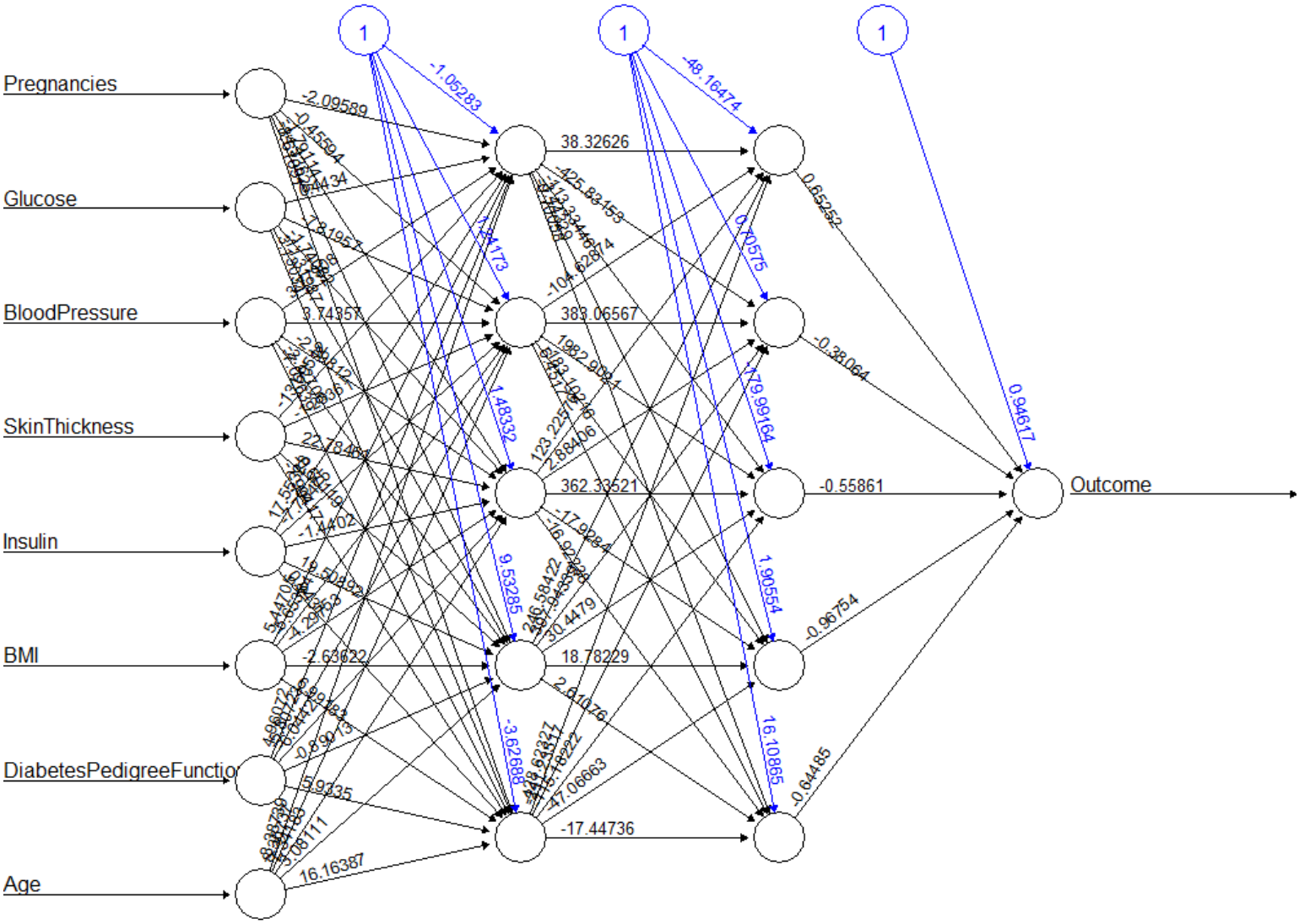

Error: 205.329362 Steps: 61164

2 hidden with 5 nodes each:

```
NN2 = neuralnet(Outcome
                ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
                data=train_set, hidden = c(5,5),linear.output = TRUE,stepmax = 9999999999)

predict_testNN2 = compute(NN2, test_set[,c(1:8)])
predict_testNN2 =round(predict_testNN2$net.result)
predict_testNN2 <- sapply(predict_testNN2,as.numeric)

table(test_set$Outcome,predict_testNN2)
accuracy <- mean(test_set$Outcome == predict_testNN2)
accuracy

plot(NN2, rep = 'best')
```

Accuracy:

0.6692607




Error: 22.432293 Steps: 130459

Accuracy dropped from 0.6692607 to 0.7276265 because I think it is a clear case of overfitting as we increased number of nodes and hidden layer the variance increase.

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

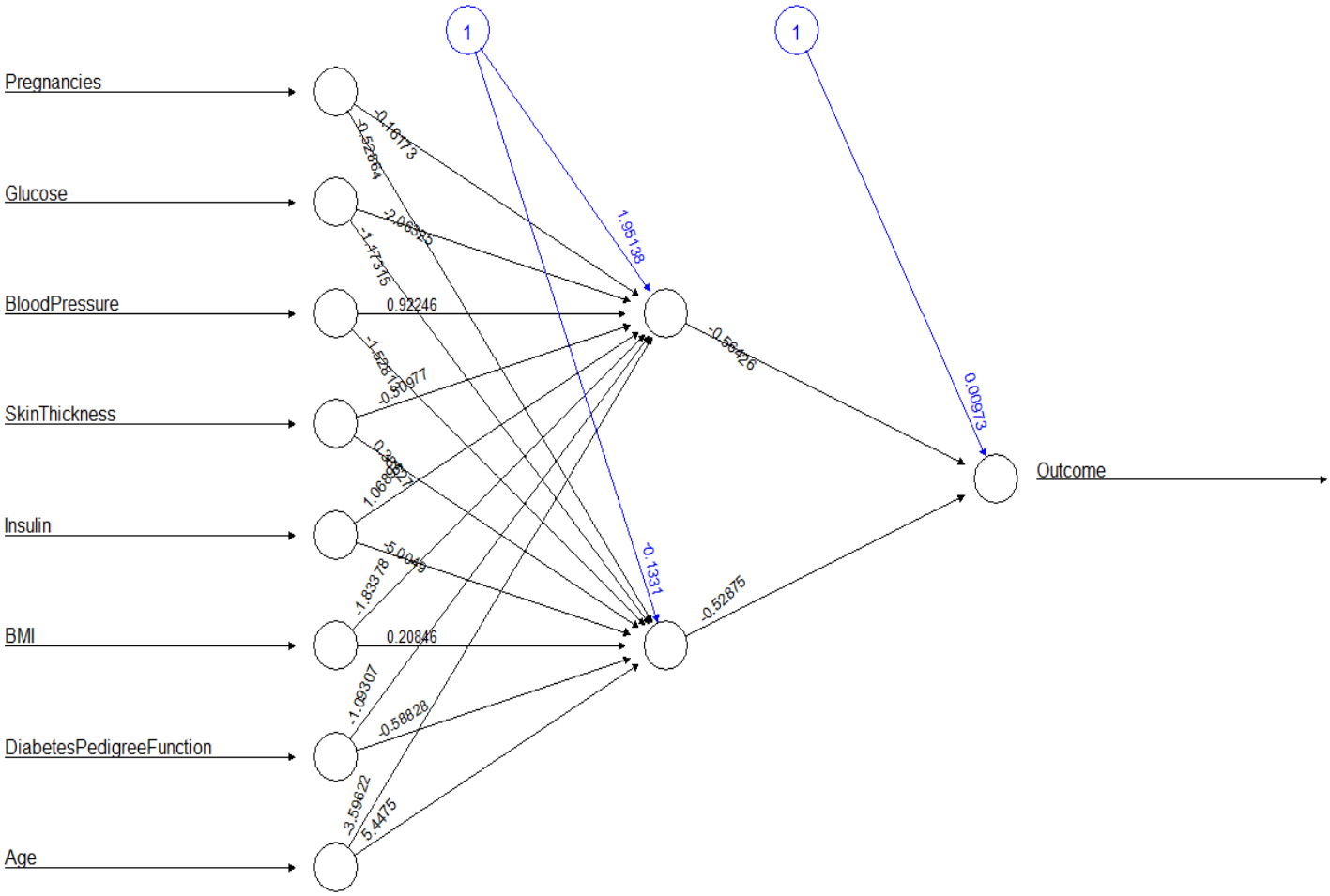
## Activation function tanh

Accuracy

0.7315175

```
predict_testNN
   0   1
0 133  28
1  41  55
```

```
NNtanh = neuralnet(Outcome
                ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
                train_set, hidden = 2,linear.output = TRUE, likelihood = TRUE,stepmax = 9999999999,act.fct="tanh")


plot(NNtanh, rep = 'best')

predict_testNN = compute(NNtanh, test_set[,c(1:8)])
predict_testNN = round(predict_testNN$net.result)
predict_testNN <- sapply(predict_testNN,as.numeric)

table(test_set$Outcome,predict_testNN)
accuracy <- mean(test_set$Outcome == predict_testNN)
accuracy
```



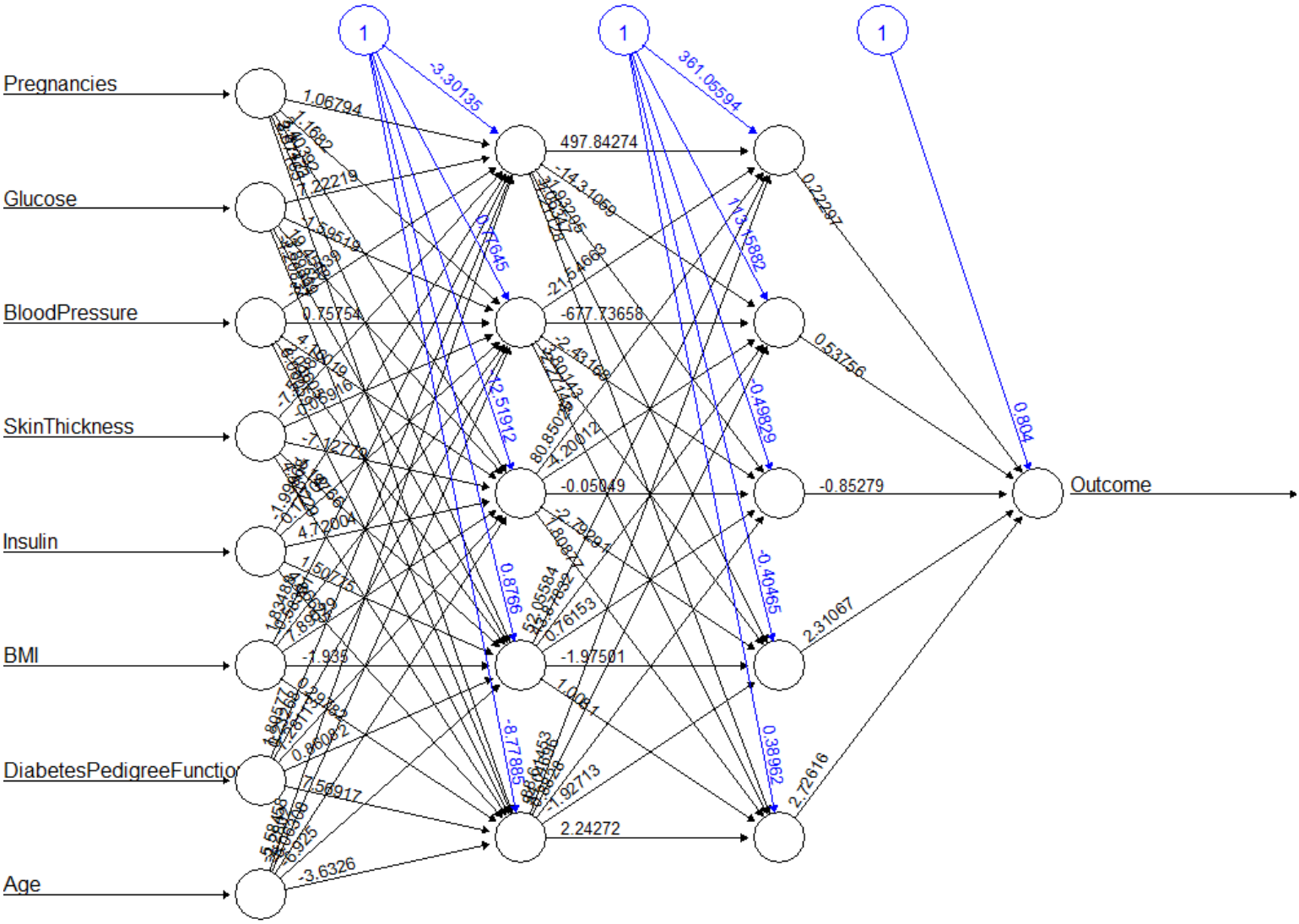

Error: 33.719883 Steps: 6022

# 2 Hidden Layers and 5 nodes each

```
NNtanh = neuralnet(Outcome
                   ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
                   train_set, hidden = c(5,5),linear.output = TRUE, likelihood = TRUE,stepmax = 9999999999,act.fct="tanh")


plot(NNtanh, rep = 'best')

predict_testNN = compute(NNtanh, test_set[,c(1:8)])


predict_testNN = round(predict_testNN$net.result)


predict_testNN <- sapply(predict_testNN,as.numeric)

table(test_set$Outcome,predict_testNN)


accuracy <- mean(test_set$Outcome == predict_testNN)
accuracy
```

Accuracy

0.7315175

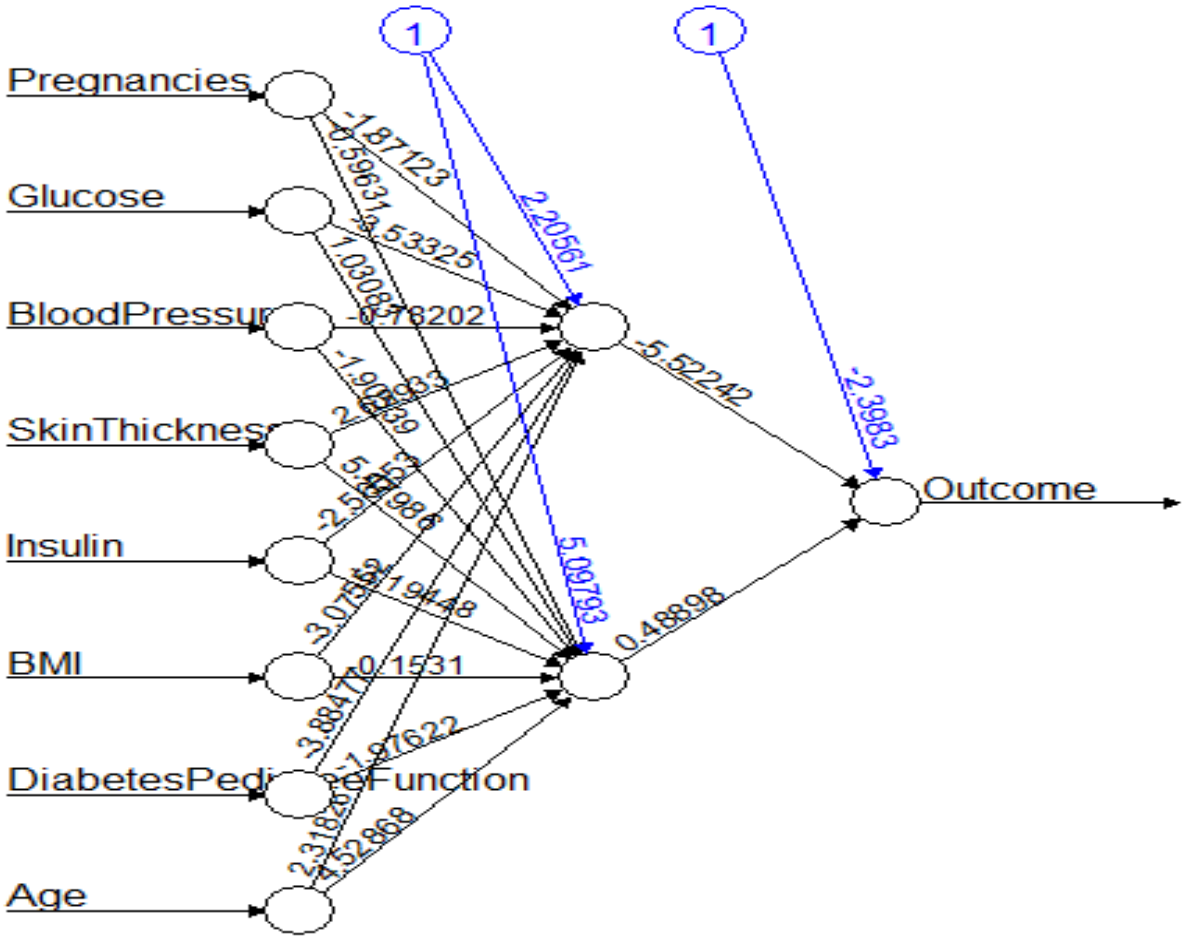



Error: 20.707845 Steps: 2572186

# Activation Function soft plus

```
softplus <- function(x) log(1 + exp(x))
NNsoftplus = neuralnet(Outcome
                  ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
                  train_set, hidden = 2,linear.output = FALSE, likelihood = TRUE,stepmax = 9999999999,act.fct=softplus)

plot(NNsoftplus, rep = 'best')


predict_testNN = compute(NNtanh, test_set[,c(1:8)])
predict_testNN = round(predict_testNN$net.result)
predict_testNN <- sapply(predict_testNN,as.numeric)

table(test_set$Outcome,predict_testNN)
accuracy <- mean(test_set$Outcome == predict_testNN)
accuracy
```

```
> accuracy
[1] 0.7315175
```

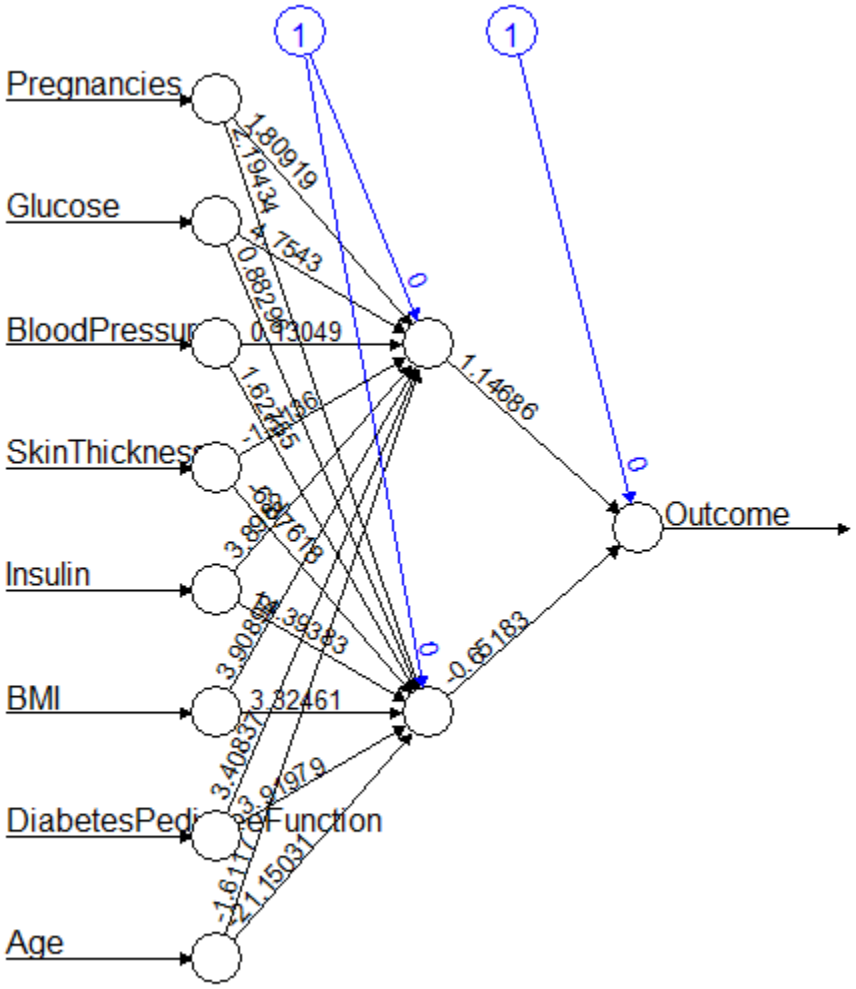


# learning rate

```
NNlr = neuralnet(Outcome
                 ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
                 train_set, hidden = 2,linear.output = FALSE, likelihood = TRUE,stepmax = 9999999999,learningrate = 0.1)

plot(NNlr, rep = 'best')

predict_testNN = compute(NNlr, test_set[,c(1:8)])
predict_testNN = round(predict_testNN$net.result)
predict_testNN <- sapply(predict_testNN,as.numeric)

table(test_set$Outcome,predict_testNN)
accuracy <- mean(test_set$Outcome == predict_testNN)
accuracy
```

```
> accuracy
[1] 0.7237354
```

# Execluding Bias




Error: 33.934334 Steps: 6593

```
NN2 = neuralnet(Outcome
                ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
                data=train_set, hidden = 2,linear.output = TRUE,stepmax = 9999999999)

NN2$weights[[c(1, 1)]][1,]=0
NN2$weights[[c(1, 2)]][1,]=0
NN2$weights

predict_testNN2 = compute(NN2, test_set[,c(1:8)])
predict_testNN2 =round(predict_testNN2$net.result)
predict_testNN2 <- sapply(predict_testNN2,as.numeric)

table(test_set$Outcome,predict_testNN2)
accuracy <- mean(test_set$Outcome == predict_testNN2)
accuracy

plot(NN2, rep = 'best')
```

```
> accuracy
[1] 0.4980545
```

| | **2 nodes one hidden layer Accuracy** |
|---|---|
| *First one* | 0. 7276265 |
| *Tanh Activation Function* | 0.7315175 |
| *Soft plus activationFunction* | 0.7315175 |
| *learning rate = 0.1* | 0.7237354 |
| *learning rate = 0.01* | 0.7042802 |
| *learning rate = 0.8* | 0.7276265 |
| *learning rate = 0.9* | 0.7003891 |
| *Epoch=10* | 0.7276265 |
| *Epoch=40* | 0.7315175 |
| *Epoch=100* | 0.7237354 |
| *Execluding Bias* | 0.4980545 |
| ***BEST Model***<br>*Tanh Activation Function +*<br>*Epoch=40 +*<br>*learning rate = 0.1* | 0.7315175 |

Accuracy did not increase when using all parameters combined. The most change that caused increase in accuarcy is the activation function.