Machine learning models for cancer predictive analysis

Natalia

28 May 2019

```
data <- read.csv("C://Users//Natalia//Desktop//ITMO//R//R project//cancer data//mammo//mammographic_mass
View(data)
```

```
#Comments for dataset:

writeLines(readLines("C://Users//Natalia//Desktop//ITMO//R//R project//cancer data//mammo//mammo.txt"))
```

```
## Warning in readLines("C://Users//Natalia//Desktop//ITMO//R//R project//
## cancer data//mammo//mammo.txt"): incomplete final line found on 'C://
## Users//Natalia//Desktop//ITMO//R//R project//cancer data//mammo//mammo.txt'
```

```
## Number of Instances
## 961
##
## Number of Attributes: 6
## 6 (1 goal field, 1 non-predictive, 4 predictive attributes)
##
## Attribute Information:
## BI-RADS assessment: 1 to 5 (ordinal)
## Age: patient's age in years (continuous)
## Shape (Mass shape, categorical):
##    round=1
##    oval=2
##    lobular=3
##    irregular=4
## Margin: mass margin (categorical):
##    circumscribed=1
##    microlobulated=2
##    obscured=3
##    ill-defined=4
##    spiculated=5
## Density: mass density (ordinal)
##    high=1
##    iso=2
##    low=3
##    fat-containing=4
## Severity: benign=0 or malignant=1 (binominal)
## Missing Attribute Values: Yes
## BI-RADS assessment: 2
## Age: 5
## Shape: 31
## Margin: 48
## Density: 76
## Severity: 0
## Class Distribution: benign:
## Benign: 516
## Malignant: 445
```

Analyse data and tidy it up.

```
# Analyse the data - checking for values, NAs, data type.
summary(data)
```

```
##      Score          Age      Shape   Margin  Density   Malignant
##  4      :547   59     : 36   ?: 31   ?: 48   ?: 76   Min.   :0.0000
##  5      :345   57     : 32   1:224   1:357   1: 16   1st Qu.:0.0000
##  3      : 36   67     : 32   2:211   2: 24   2: 59   Median :0.0000
##  2      : 14   66     : 31   3: 95   3:116   3:798   Mean   :0.4631
##  6      : 11   46     : 28   4:400   4:280   4: 12   3rd Qu.:1.0000
##  0      :  5   64     : 27           5:136           Max.   :1.0000
##  (Other):  3   (Other):775
```

```
str(data)
```

```
## 'data.frame':    961 obs. of  6 variables:
##  $ Score    : Factor w/ 8 levels "?","0","2","3",..: 2 2 2 2 2 3 3 3 3 3 ...
##  $ Age      : Factor w/ 74 levels "?","18","19",..: 29 53 42 56 55 50 43 39 60 41 ...
##  $ Shape    : Factor w/ 5 levels "?","1","2","3",..: 3 5 5 5 5 2 2 2 2 2 ...
##  $ Margin   : Factor w/ 6 levels "?","1","2","3",..: 5 6 5 4 5 2 2 1 2 2 ...
##  $ Density  : Factor w/ 5 levels "?","1","2","3",..: 4 4 4 4 4 1 1 2 3 4 ...
##  $ Malignant: int  0 1 0 1 1 0 1 0 0 0 ...
```

```
head(data)
```

```
##   Score Age Shape Margin Density Malignant
## 1     0  45     2      4       3         0
## 2     0  69     4      5       3         1
## 3     0  58     4      4       3         0
## 4     0  72     4      3       3         1
## 5     0  71     4      4       3         1
## 6     2  66     1      1       ?         0
```

```
dim(data)
```

```
## [1] 961   6
```

```
data$Malignant <- factor(data$Malignant)
data$Class <- ifelse(data$Malignant == "0", "benign", ifelse(data$Malignant == 1, "malignant", NA))
data$Malignant <- NULL
head(data)
```

```
##   Score Age Shape Margin Density     Class
## 1     0  45     2      4       3    benign
## 2     0  69     4      5       3 malignant
## 3     0  58     4      4       3    benign
## 4     0  72     4      3       3 malignant
## 5     0  71     4      4       3 malignant
## 6     2  66     1      1       ?    benign
```

```
data[ data == "?" ] <- NA
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------- tidyverse 1.2
```

```
## v ggplot2 3.1.1       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
```

```
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
map_int(data, function(.x) sum(is.na(.x)))
```

```
##   Score     Age   Shape  Margin Density   Class
##       2       5      31      48      76       0
```

```
data <- na.omit(data)
map_int(data, function(.x) sum(is.na(.x)))
```

```
##   Score     Age   Shape  Margin Density   Class
##       0       0       0       0       0       0
```

```
sapply(data, mode)
```

```
##       Score         Age       Shape      Margin     Density       Class
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric" "character"
```

```
data <- as.data.frame(data, stringsAsFactors=T)
data$Class <- as.factor(as.character(data$Class))

sapply(data,mode)
```

```
##     Score       Age     Shape    Margin   Density     Class
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
```

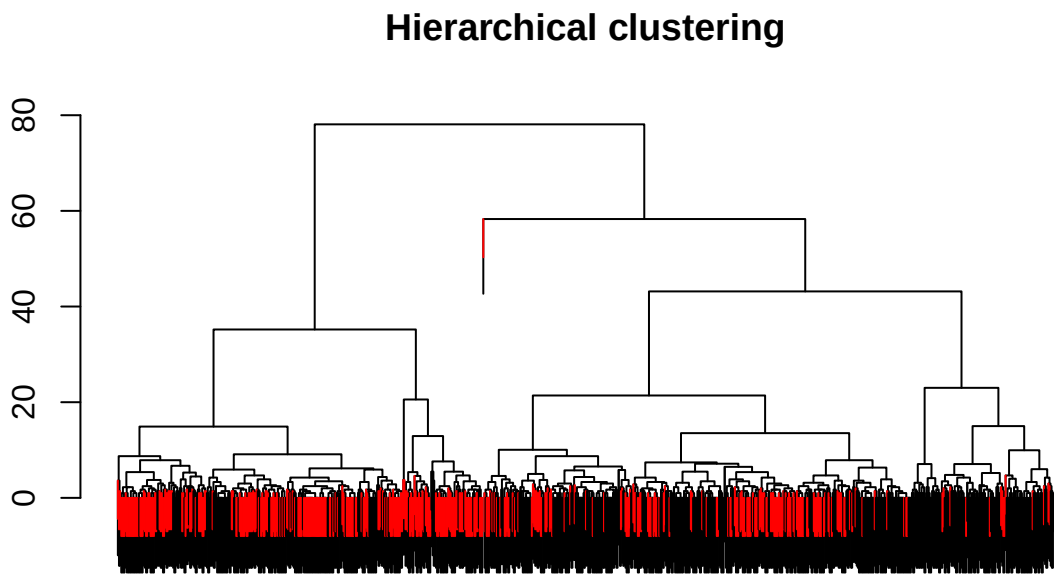
```
head(dt)
```

```
##
## 1 function (x, df, ncp, log = FALSE)
## 2 {
## 3     if (missing(ncp))
## 4         .Call(C_dt, x, df, log)
## 5     else .Call(C_dnt, x, df, ncp, log)
## 6 }
```

DATA EXPLORATION

Hierarchical clustering

```
library(sparcl)
hc <- hclust(dist(data[,1:5]), method = "complete")
ColorDendrogram(hc,y=factor(data$Class), main = "Hierarchical clustering", branchlength=8)
```



Most of the benign (black) and malignant (red) samples cluster together.

K-means clustering

```
fit <- kmeans(data[,c(1:5)], 2)
names(fit)
```

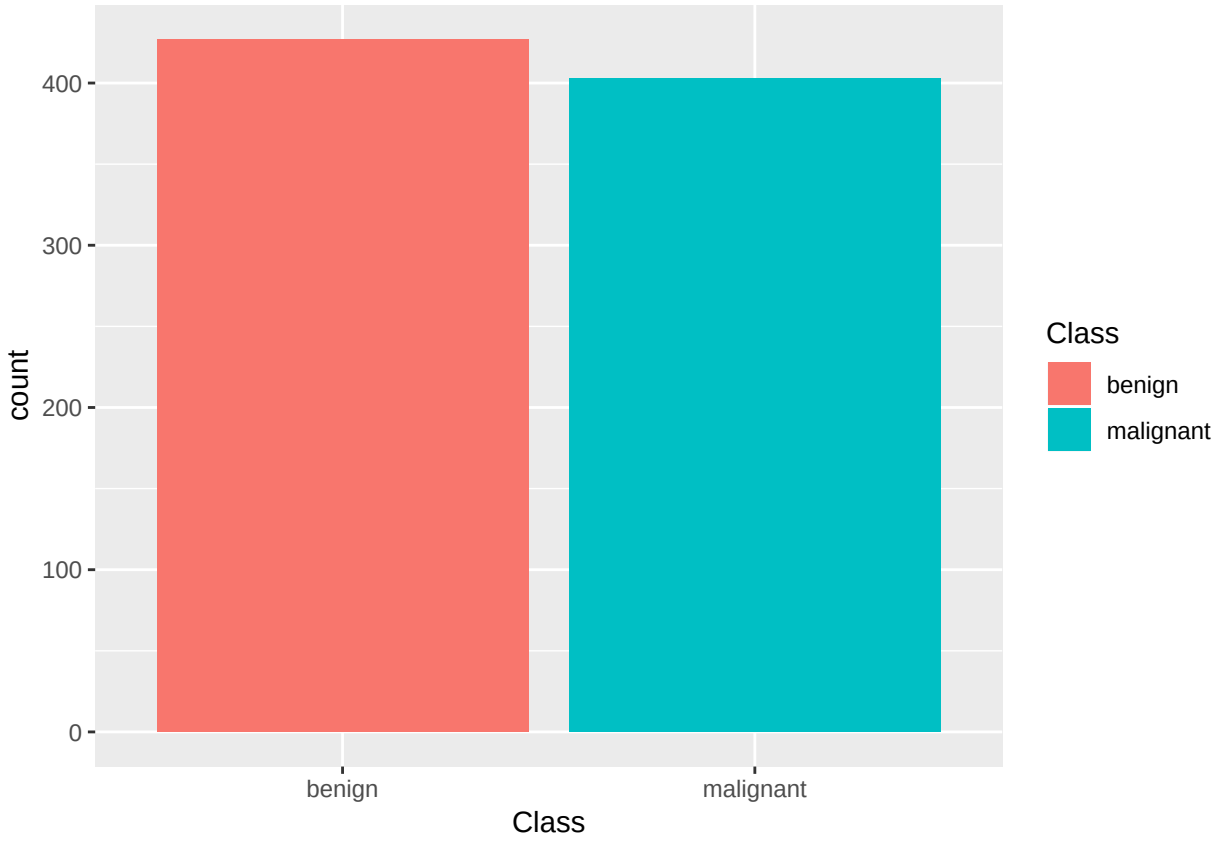
```
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
#k-means did a fairly good job
table(data.frame(fit$cluster,data[,6]))
```

```
##             data...6.
## fit.cluster benign malignant
##           1    265        99
##           2    162       304
```

Response variable for classification

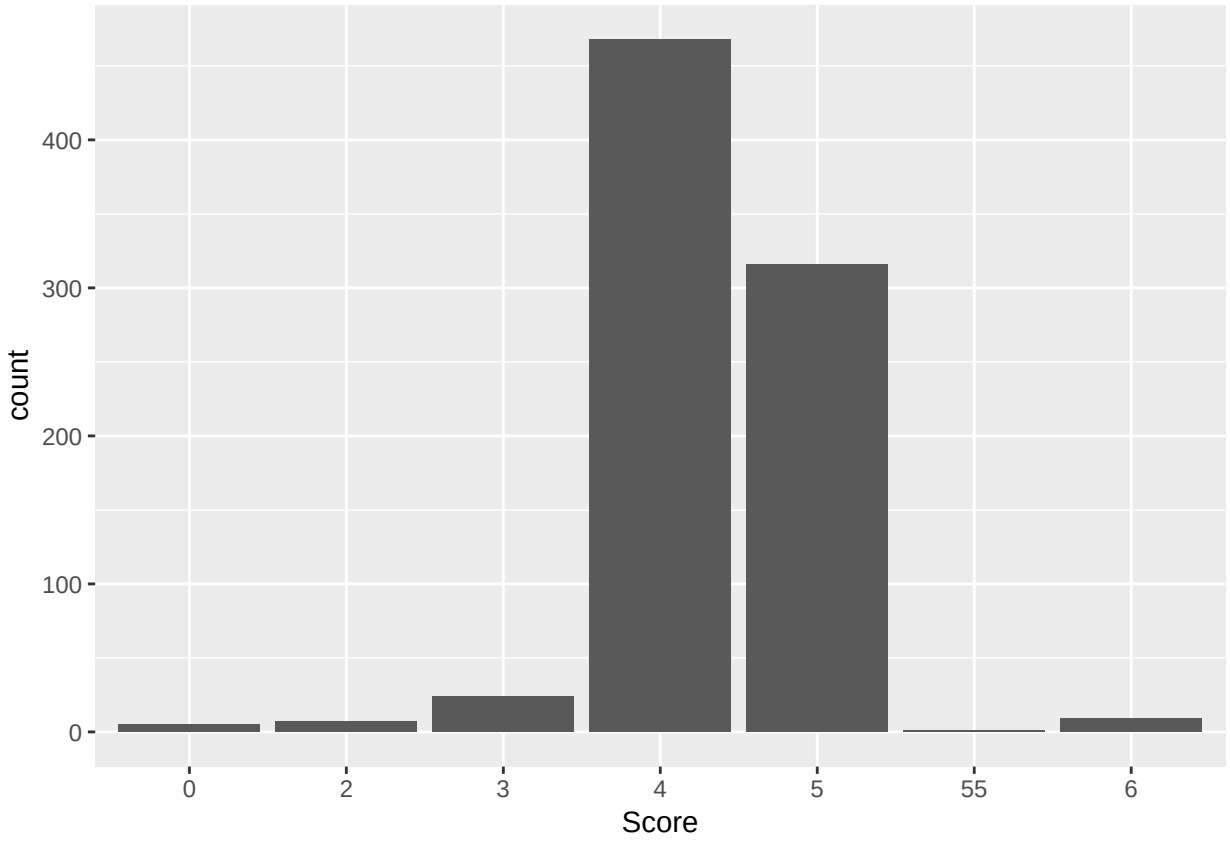
```
library(ggplot2)

ggplot(data, aes(x = Class, fill = Class)) +
  geom_bar()
```



Response variable for regression

```
ggplot(data, aes(x = Score)) +
  geom_histogram(stat = "count")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



Principal Component Analysis

```
library(pcaGoPromoter)
```

```
## Loading required package: ellipse
```

```
##
## Attaching package: 'ellipse'
```

```
## The following object is masked from 'package:graphics':
##
##     pairs
```

```
## Loading required package: Biostrings
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:dplyr':
```

```
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min

## Loading required package: S4Vectors

## Loading required package: stats4

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:tidyr':
##
##     expand

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## The following object is masked from 'package:purrr':
##
##     reduce

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: XVector

##
## Attaching package: 'XVector'

## The following object is masked from 'package:purrr':
##
##     compact
```

```
##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':
##
##     strsplit
```

```
library(ellipse)
```

```
library(mice)
```

```
## Loading required package: lattice

##
## Attaching package: 'mice'

## The following objects are masked from 'package:IRanges':
##
##     cbind, rbind

## The following objects are masked from 'package:S4Vectors':
##
##     cbind, rbind

## The following objects are masked from 'package:BiocGenerics':
##
##     cbind, rbind

## The following object is masked from 'package:tidyr':
##
##     complete

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

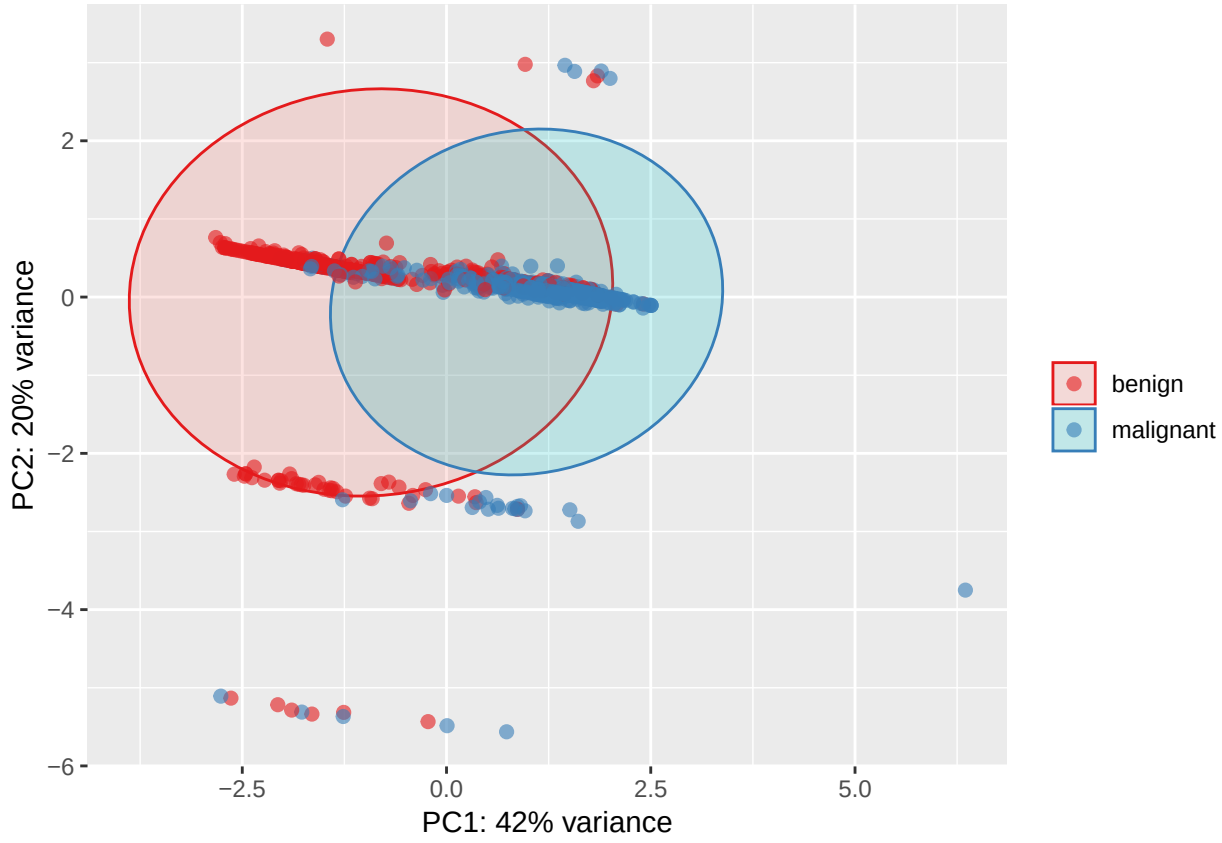
```
data[,1:5] <- apply(data[, 1:5], 2, function(x) as.numeric(as.character(x)))
dataset_impute <- mice(data[, 1:5], print = FALSE)
data <- cbind(data[, 6, drop = FALSE], mice::complete(dataset_impute, 1))
data$Class <- as.factor(data$Class)
```

```
data <- na.omit(data)
```

```
# perform pca and extract scores
pcaOutput <- pca(t(data[, 2:6]), printDropped = FALSE, scale = TRUE, center = TRUE)
pcaOutput2 <- as.data.frame(pcaOutput$scores)
```

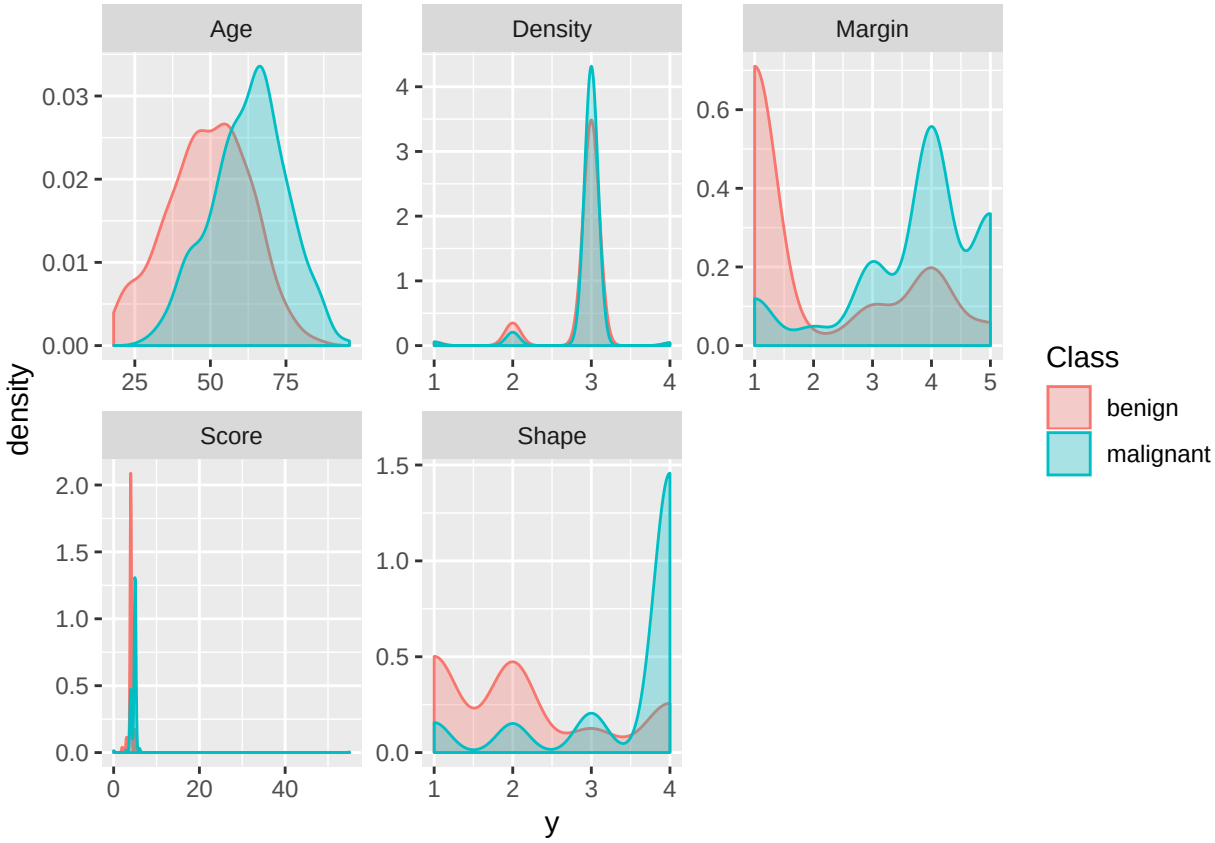
```
# define groups for plotting:

pcaOutput2$groups <- data$Class

centroids <- aggregate(cbind(PC1, PC2) ~ groups, pcaOutput2, mean)

conf.rgn  <- do.call(rbind, lapply(unique(pcaOutput2$groups), function(t)
  data.frame(groups = as.character(t),
             ellipse(cov(pcaOutput2[pcaOutput2$groups == t, 1:2]),
                   centre = as.matrix(centroids[centroids$groups == t, 2:3]),
                   level = 0.95),
             stringsAsFactors = FALSE)))
```

```r
#Plot PCA with variance %:

ggplot(data = pcaOutput2, aes(x = PC1, y = PC2, group = groups, color = groups)) +
    geom_polygon(data = conf.rgn, aes(fill = groups), alpha = 0.2) +
    geom_point(size = 2, alpha = 0.6) +
    scale_color_brewer(palette = "Set1") +
    labs(color = "",
         fill = "",
         x = paste0("PC1: ", round(pcaOutput$pov[1], digits = 2) * 100, "% variance"),
         y = paste0("PC2: ", round(pcaOutput$pov[2], digits = 2) * 100, "% variance"))
```



Features

```r
library(tidyr)

gather(data, x, y, Score:Density) %>%
  ggplot(aes(x = y, color = Class, fill = Class)) +
    geom_density(alpha = 0.3) +
    facet_wrap( ~ x, scales = "free")
```



MACHINE LEARNING PACKAGES FOR R

caret

```
# configure multicore
library(doParallel)
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: iterators
```

```
cl <- makeCluster(detectCores())
registerDoParallel(cl)

library(caret)
```

```
##
## Attaching package: 'caret'
```

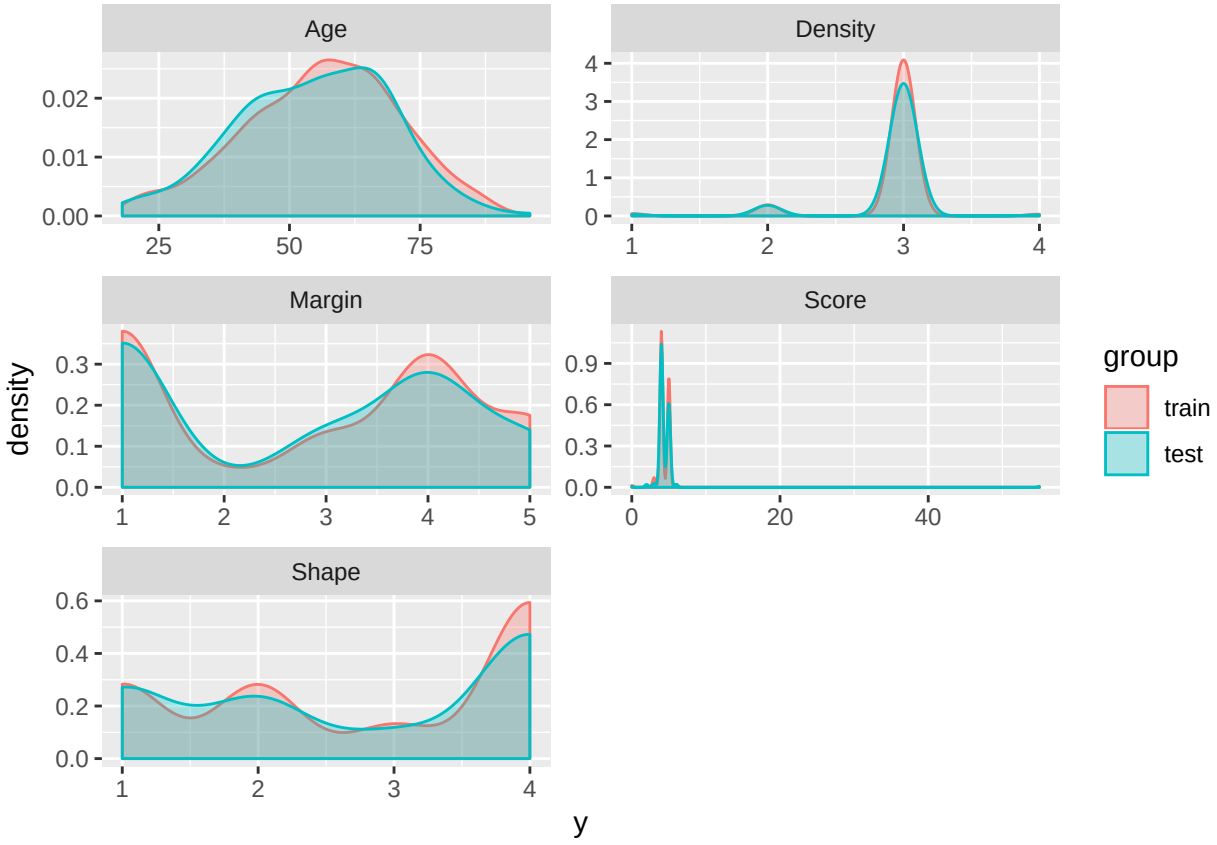
```
## The following object is masked from 'package:purrr':
```

```
##
##     lift
```

Training, validation and test data

```
set.seed(42)
index <- createDataPartition(data$Class, p = 0.7, list = FALSE)
train_data <- data[index, ]
test_data  <- data[-index, ]
```

```
library(dplyr)

rbind(data.frame(group = "train", train_data),
      data.frame(group = "test", test_data)) %>%
  gather(x, y, Score:Density) %>%
  ggplot(aes(x = y, color = group, fill = group)) +
    geom_density(alpha = 0.3) +
    facet_wrap( ~ x, scales = "free", ncol = 2)
```



Regression

```
set.seed(42)
model_glm <- caret::train(Score ~ .,
                          data = train_data,
```

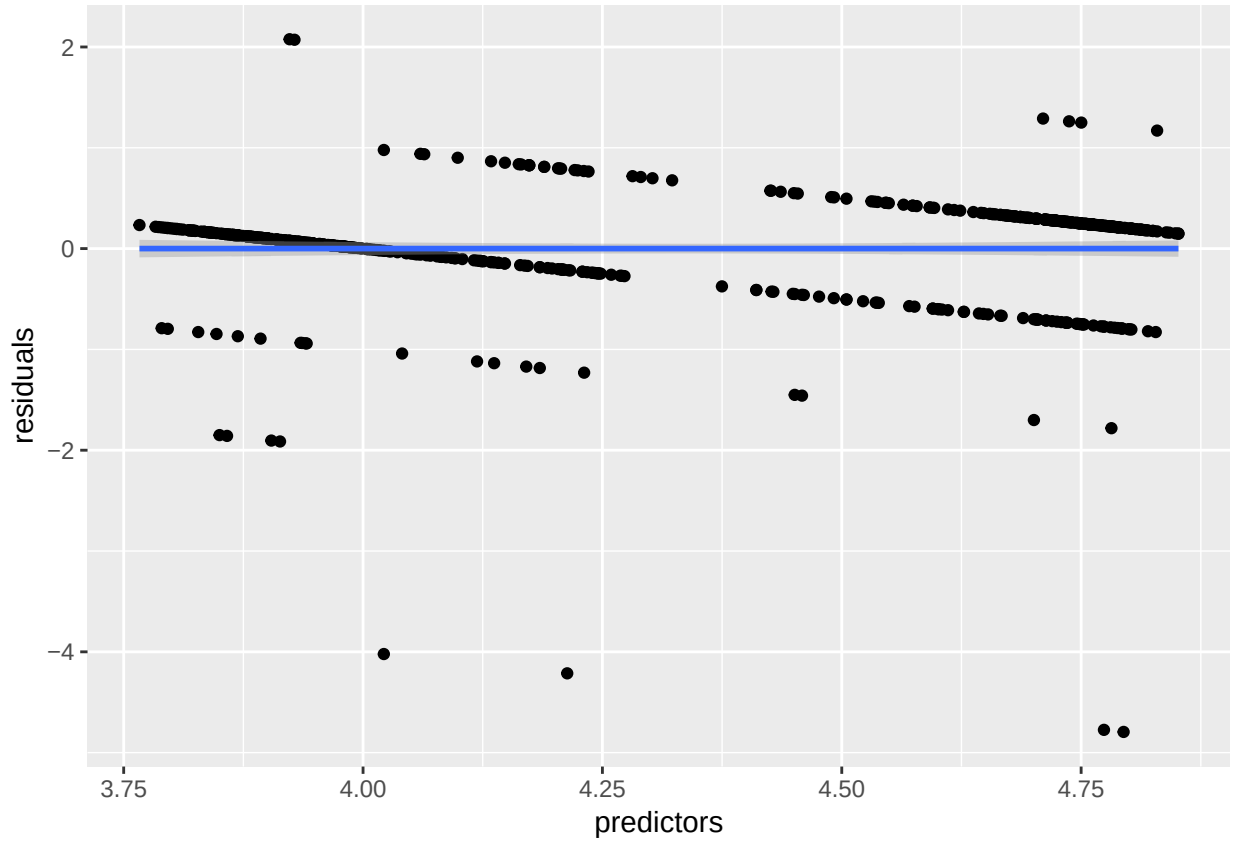
```
                          method = "glm",
                          preProcess = c("scale", "center"),
                          trControl = trainControl(method = "repeatedcv",
                                                   number = 10,
                                                   repeats = 10,
                                                   savePredictions = TRUE,
                                                   verboseIter = FALSE))
```

```
model_glm
```

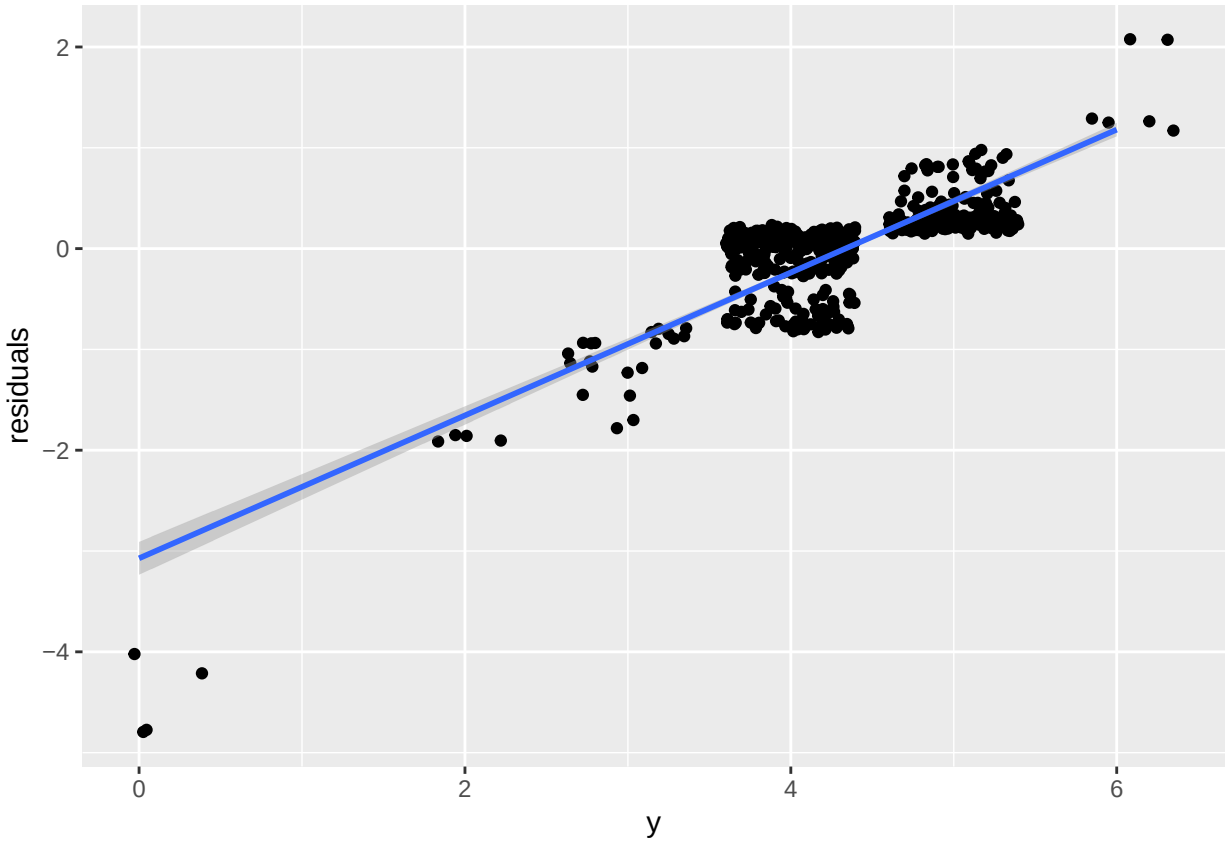
```
## Generalized Linear Model
##
## 582 samples
##   5 predictor
##
## Pre-processing: scaled (5), centered (5)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 524, 525, 524, 524, 523, 523, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.5734065  0.3340204  0.3559011
```

```
predictions <- predict(model_glm, test_data)
```

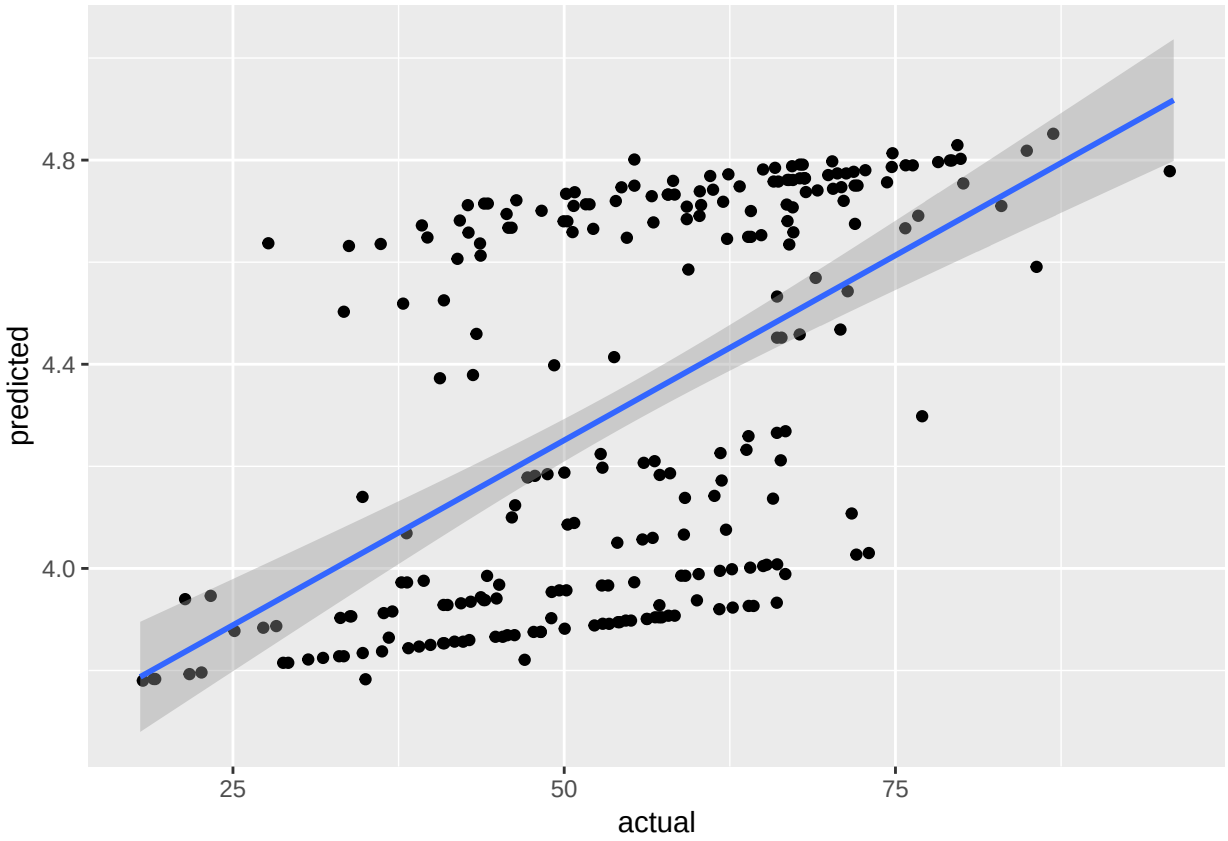
```
# model_glm$finalModel$linear.predictors == model_glm$finalModel$fitted.values
data.frame(residuals = resid(model_glm),
           predictors = model_glm$finalModel$linear.predictors) %>%
  ggplot(aes(x = predictors, y = residuals)) +
    geom_jitter() +
    geom_smooth(method = "lm")
```




```
# y == train_data$Age
data.frame(residuals = resid(model_glm),
           y = model_glm$finalModel$y) %>%
  ggplot(aes(x = y, y = residuals)) +
    geom_jitter() +
    geom_smooth(method = "lm")
```




```
data.frame(actual = test_data$Age,
           predicted = predictions) %>%
  ggplot(aes(x = actual, y = predicted)) +
    geom_jitter() +
    geom_smooth(method = "lm")
```

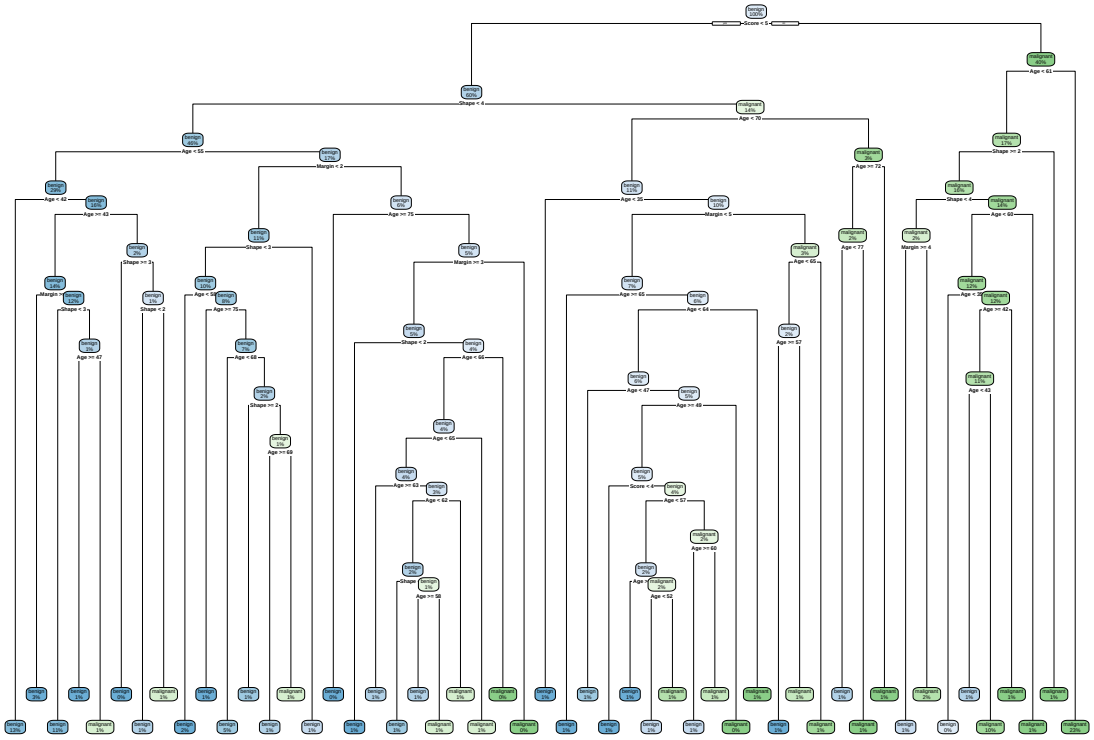


CLASSIFICATION

Decision trees

```
library(rpart)
library(rpart.plot)

set.seed(42)
fit <- rpart(Class ~ .,
             data = train_data,
             method = "class",
             control = rpart.control(xval = 10,
                                     minbucket = 2,
                                     cp = 0),
              parms = list(split = "information"))

rpart.plot(fit, extra = 100)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



RANDOM FORESTS

```
#Random Forests predictions are based on the generation of
#multiple classification trees.
#They can be used for both, classification and regression tasks.
#Here, it is classification task.
```

```
set.seed(42)
library(randomForest)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:BiocGenerics':
##
##     combine
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
model_rf <- caret::train(Class ~ .,
                         data = train_data,
                         method = "rf",
                         preProcess = c("scale", "center"),
                         trControl = trainControl(method = "repeatedcv",
                                                  number = 10,
                                                  repeats = 10,
                                                  savePredictions = TRUE,
                                                  verboseIter = FALSE))
```

```
#When savePredictions = TRUE is specified,
#can access the cross-validation resuls with model_rf$pred.

model_rf$finalModel$confusion
```

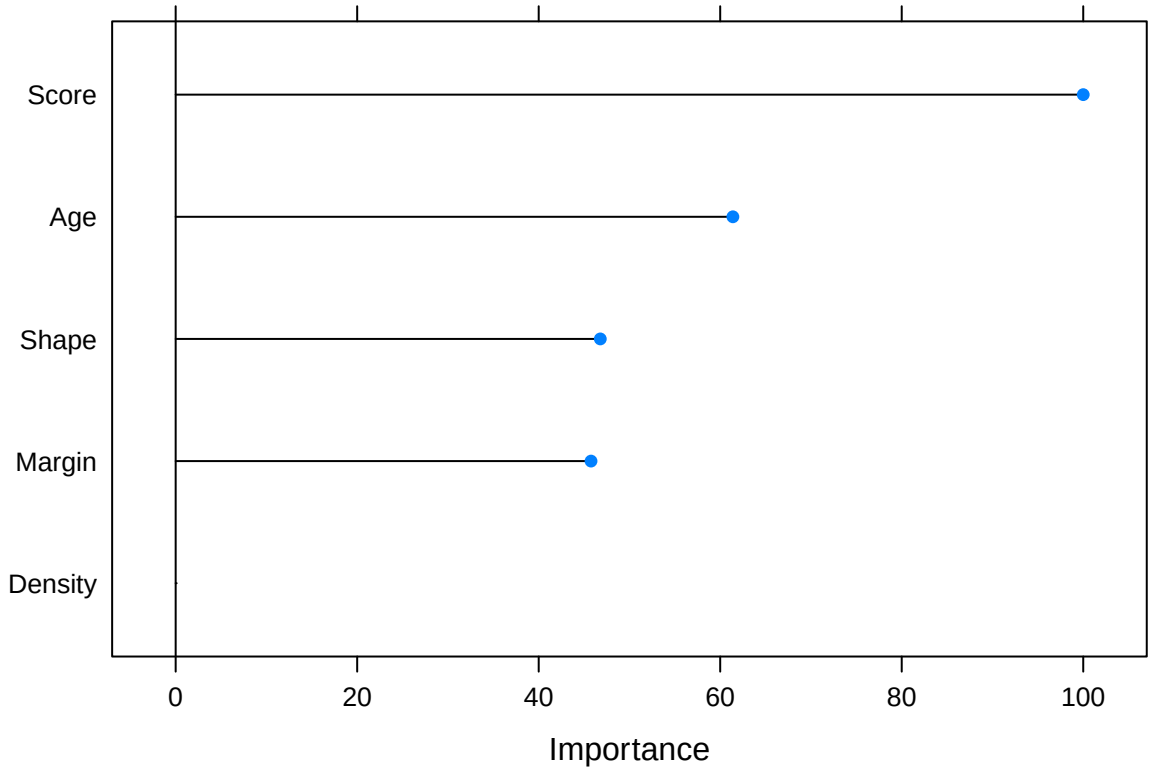
```
##           benign malignant class.error
## benign       247        52   0.1739130
## malignant     60       223   0.2120141
```

Feature Importance

```
imp <- model_rf$finalModel$importance
imp[order(imp, decreasing = TRUE), ]
```

```
##     Score       Age     Shape    Margin   Density
## 80.177655 51.670337 40.881514 40.127439  6.328954
```

```
# estimate variable importance
importance <- varImp(model_rf, scale = TRUE)
plot(importance)
```



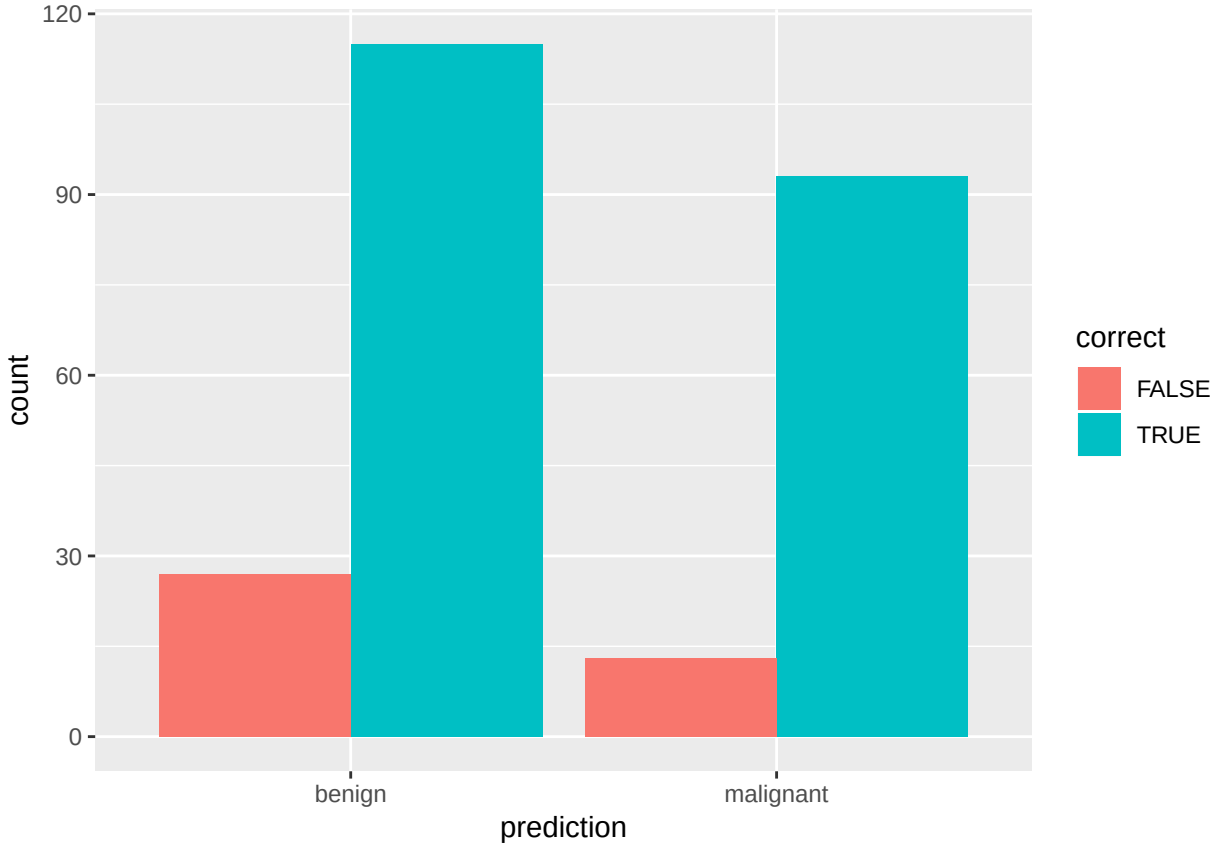
Pedicting test data

```
confusionMatrix(predict(model_rf, test_data), test_data$Class)
```

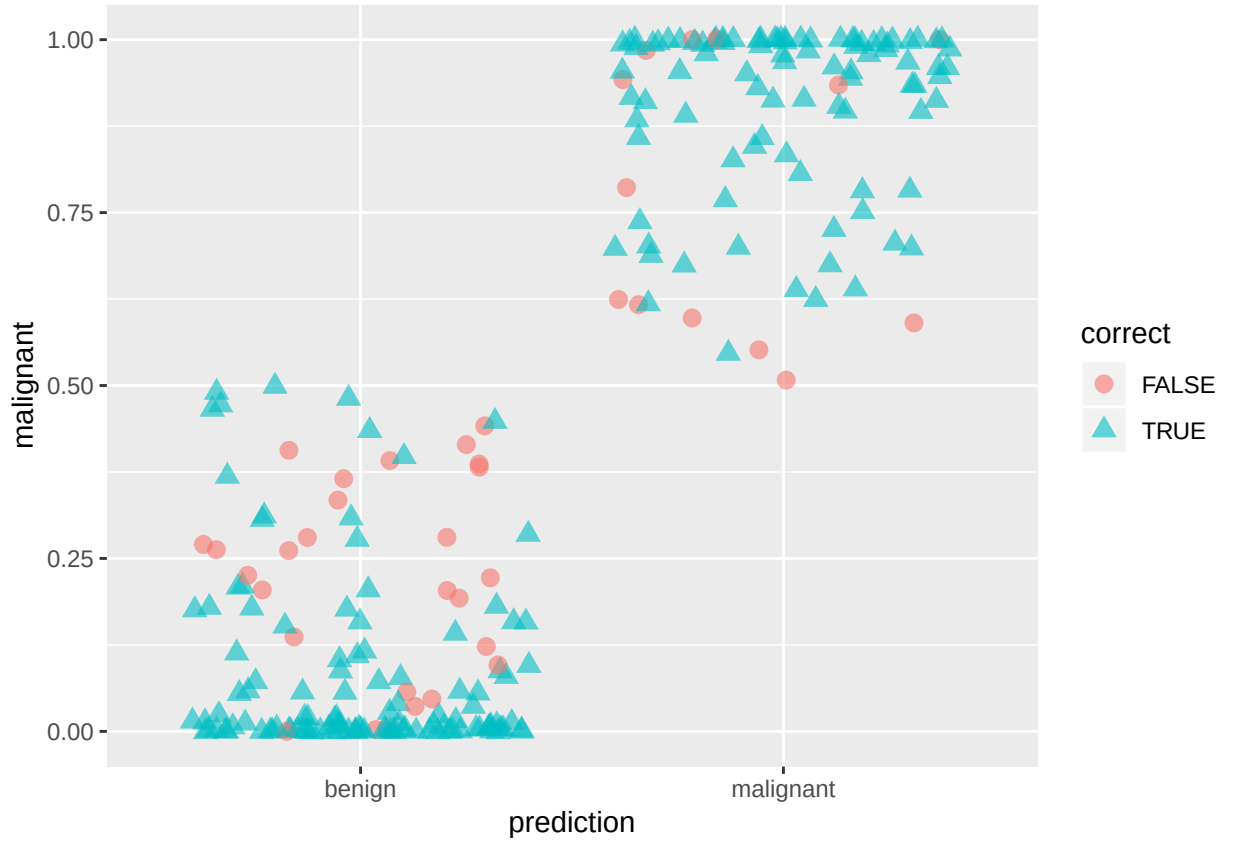
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       115        27
##   malignant     13        93
##
##                Accuracy : 0.8387
##                  95% CI : (0.7869, 0.8822)
##     No Information Rate : 0.5161
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6759
##
##  Mcnemar's Test P-Value : 0.03983
##
##             Sensitivity : 0.8984
##             Specificity : 0.7750
##          Pos Pred Value : 0.8099
##          Neg Pred Value : 0.8774
##              Prevalence : 0.5161
```

```
##          Detection Rate : 0.4637
##    Detection Prevalence : 0.5726
##       Balanced Accuracy : 0.8367
##
##        'Positive' Class : benign
##
```

```r
results <- data.frame(actual = test_data$Class,
                      predict(model_rf, test_data, type = "prob"))

results$prediction <- ifelse(results$benign > 0.5, "benign",
                             ifelse(results$malignant > 0.5, "malignant", NA))

results$correct <- ifelse(results$actual == results$prediction, TRUE, FALSE)

ggplot(results, aes(x = prediction, fill = correct)) +
  geom_bar(position = "dodge")
```




```r
ggplot(results, aes(x = prediction, y = malignant, color = correct, shape = correct)) +
  geom_jitter(size = 3, alpha = 0.6)
```



#EXTREME GRADIENT BOOSTING.

Extreme gradient boosting (XGBoost) is a faster and improved implementation of gradient boosting for supervised learning.

```
#XGBoost is a tree ensemble model, which means the sum of predictions
#from a set of classification and regression trees (CART).
#In that, XGBoost is similar to Random Forests but it uses a different approach
#to model training: it uses a combination of "weak" functions during iteration process,
#for each next iteration step, the model learns using the "mistakes" data of previous steps.
```

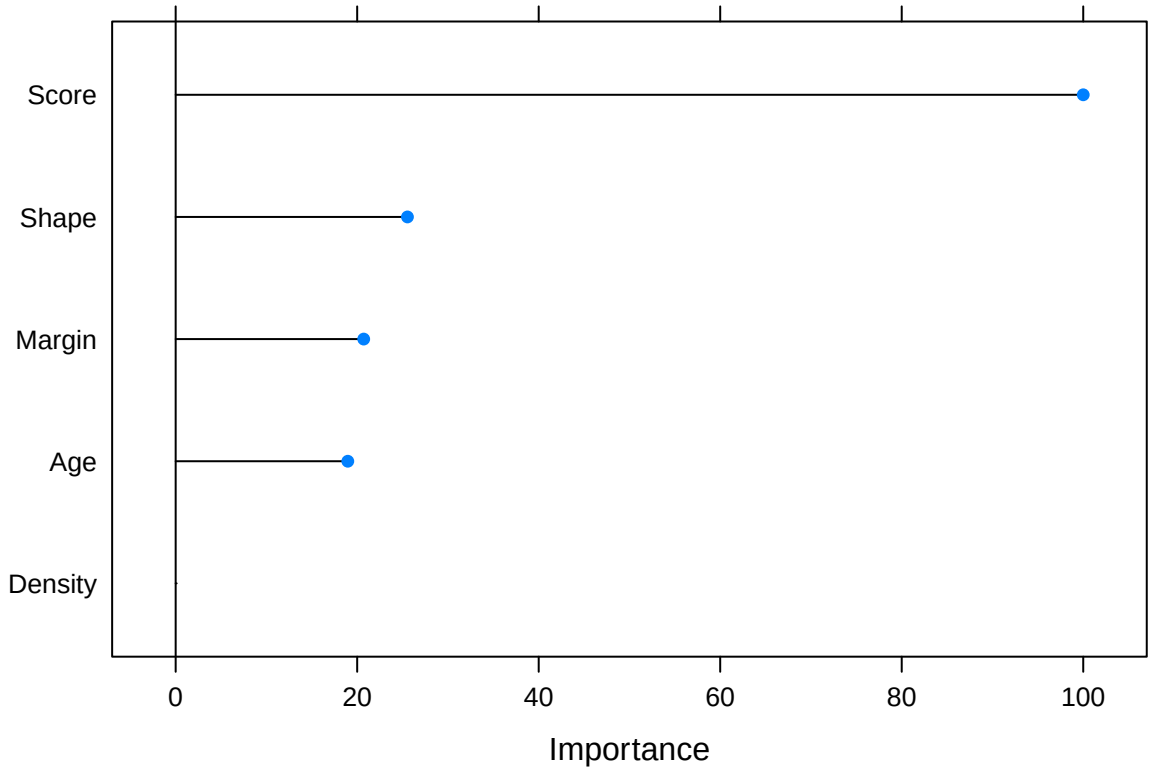
```
set.seed(42)
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
## The following object is masked from 'package:XVector':
##
##     slice
## The following object is masked from 'package:IRanges':
##
##     slice
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
model_xgb <- caret::train(Class ~ .,
                          data = train_data,
                          method = "xgbTree",
                          preProcess = c("scale", "center"),
                          trControl = trainControl(method = "repeatedcv",
                                                   number = 10,
                                                   repeats = 10,
                                                   savePredictions = TRUE,
                                                   verboseIter = FALSE))
```

Feature Importance

```
importance <- varImp(model_xgb, scale = TRUE)
plot(importance)
```



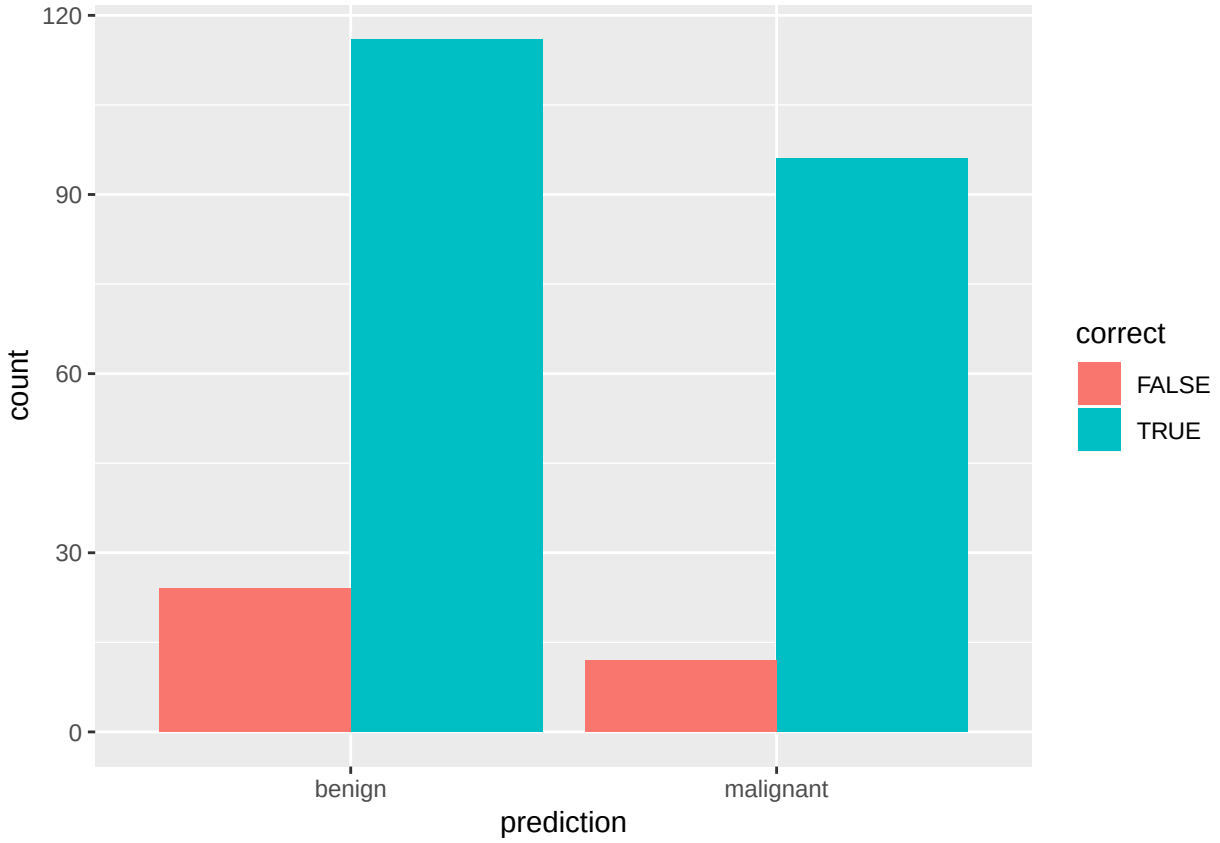
#Predicting test data

```
confusionMatrix(predict(model_xgb, test_data), test_data$Class)
```

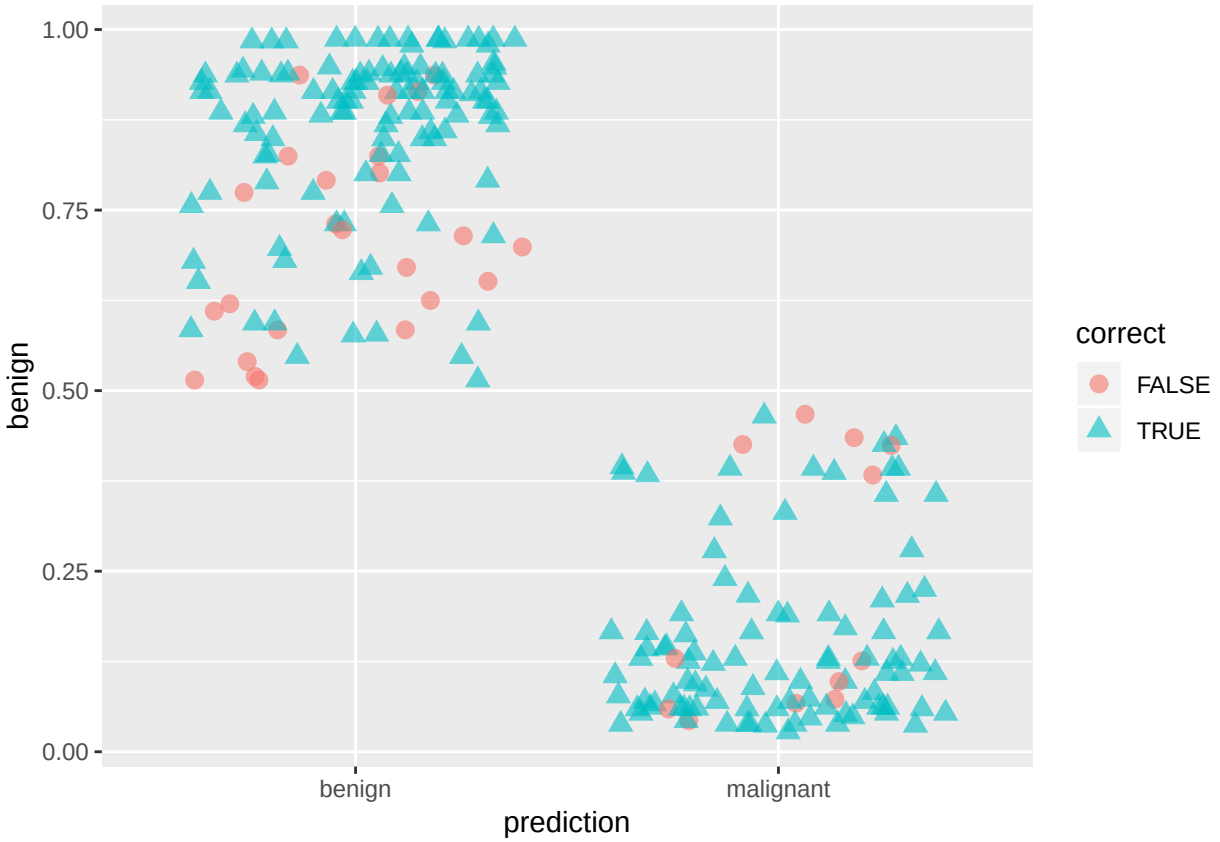
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       116        24
##   malignant     12        96
##
```

```
##                Accuracy : 0.8548
##                  95% CI : (0.8047, 0.8962)
##     No Information Rate : 0.5161
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7085
##
##  Mcnemar's Test P-Value : 0.06675
##
##             Sensitivity : 0.9062
##             Specificity : 0.8000
##          Pos Pred Value : 0.8286
##          Neg Pred Value : 0.8889
##              Prevalence : 0.5161
##          Detection Rate : 0.4677
##    Detection Prevalence : 0.5645
##       Balanced Accuracy : 0.8531
##
##        'Positive' Class : benign
##
```

```r
results <- data.frame(actual = test_data$Class,
                      predict(model_xgb, test_data, type = "prob"))

results$prediction <- ifelse(results$benign > 0.5, "benign",
                             ifelse(results$malignant > 0.5, "malignant", NA))

results$correct <- ifelse(results$actual == results$prediction, TRUE, FALSE)

ggplot(results, aes(x = prediction, fill = correct)) +
  geom_bar(position = "dodge")
```




```
ggplot(results, aes(x = prediction, y = benign, color = correct, shape = correct)) +
  geom_jitter(size = 3, alpha = 0.6)
```



#FEATURE SELECTION

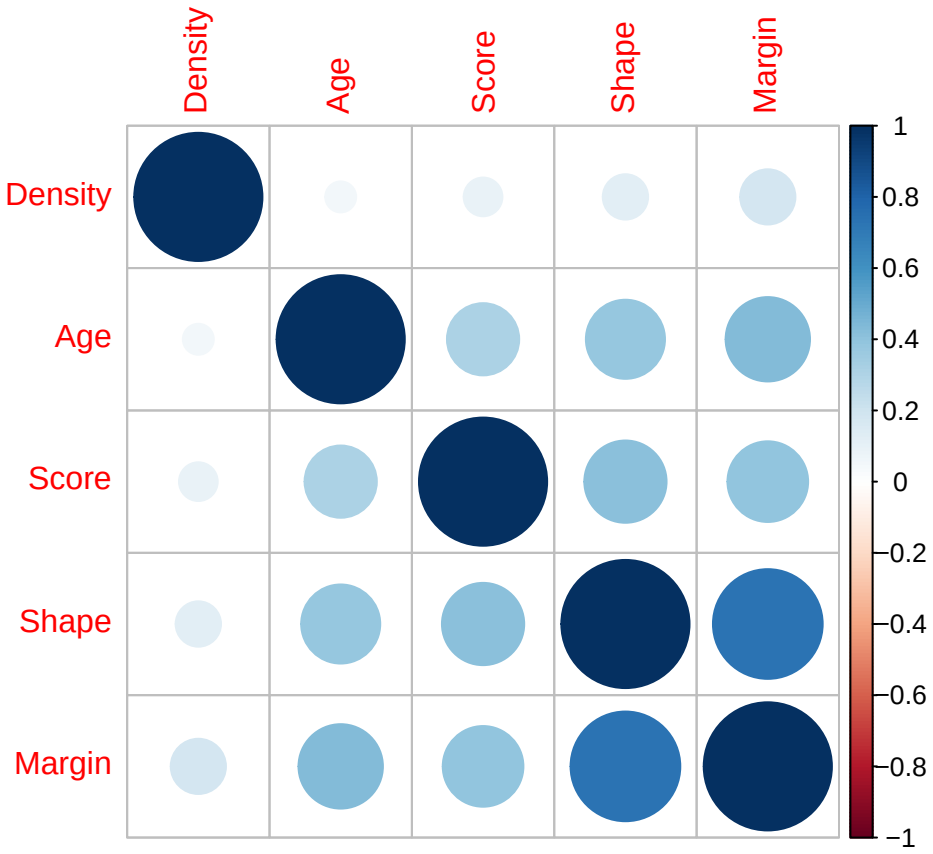
Performing feature selection on the whole dataset would lead to prediction bias, we therefore need to run the whole modeling process on the training data alone!

Correlation

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
# calculate correlation matrix
corMatMy <- cor(train_data[, 2:6])
corrplot(corMatMy, order = "hclust")
```




```
#Apply correlation filter at 0.70:

highlyCor <- colnames(train_data[, -1])[findCorrelation(corMatMy, cutoff = 0.7, verbose = TRUE)]
```

```
## Compare row 4  and column  3 with corr  0.733
##   Means:  0.437 vs 0.288 so flagging column 4
## All correlations <= 0.7
```

```
# which variables are flagged for removal?
highlyCor
```

```
## [1] "Margin"
```

```
#then we remove these variables
train_data_cor <- train_data[, which(!colnames(train_data) %in% highlyCor)]
```

GRID SEARCH WITH CARET

Automatic Grid

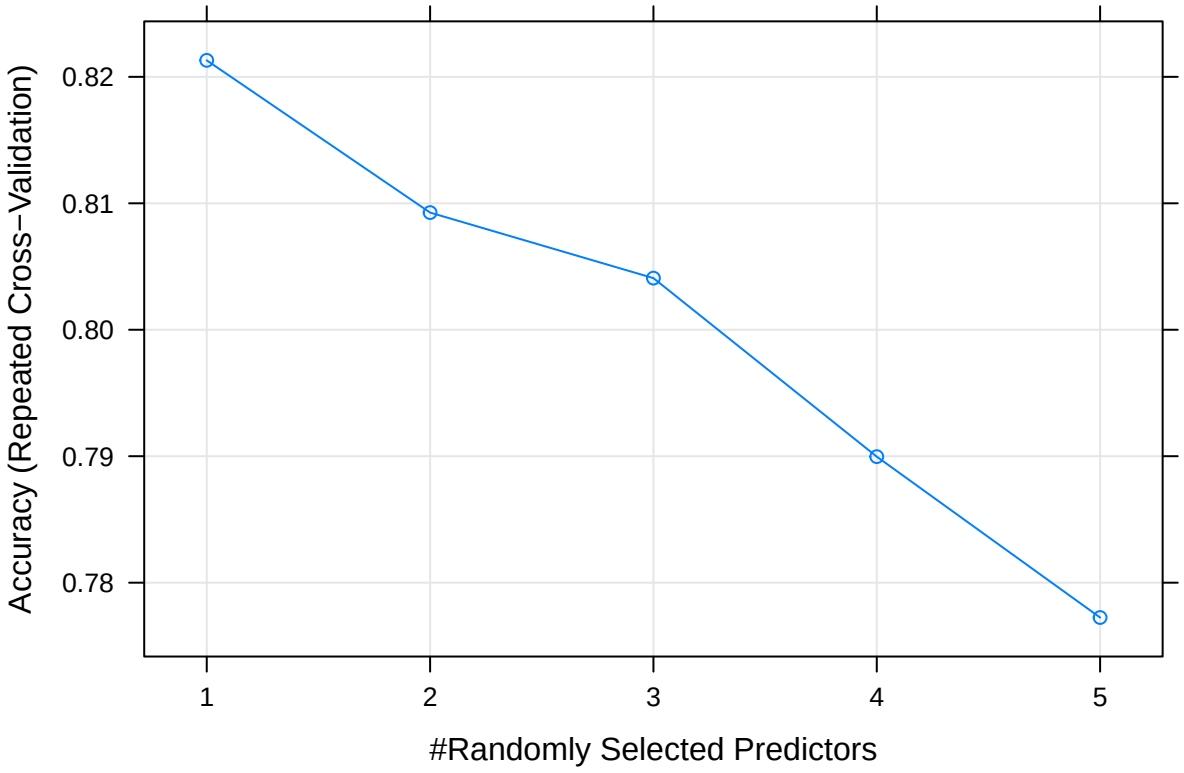
```
set.seed(42)
model_rf_tune_auto <- caret::train(Class ~ .,
                         data = train_data,
                         method = "rf",
                         preProcess = c("scale", "center"),
                         trControl = trainControl(method = "repeatedcv",
```

```
                                                    number = 10,
                                                    repeats = 10,
                                                    savePredictions = TRUE,
                                                    verboseIter = FALSE,
                                                    search = "random"),
                          tuneLength = 15)
```

```
model_rf_tune_auto
```

```
## Random Forest
##
## 582 samples
##   5 predictor
##   2 classes: 'benign', 'malignant'
##
## Pre-processing: scaled (5), centered (5)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 524, 524, 524, 524, 523, 523, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   1     0.8213038  0.6424800
##   2     0.8092664  0.6179492
##   3     0.8040816  0.6077005
##   4     0.7899664  0.5795103
##   5     0.7772449  0.5538580
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 1.
```

```
plot(model_rf_tune_auto)
```



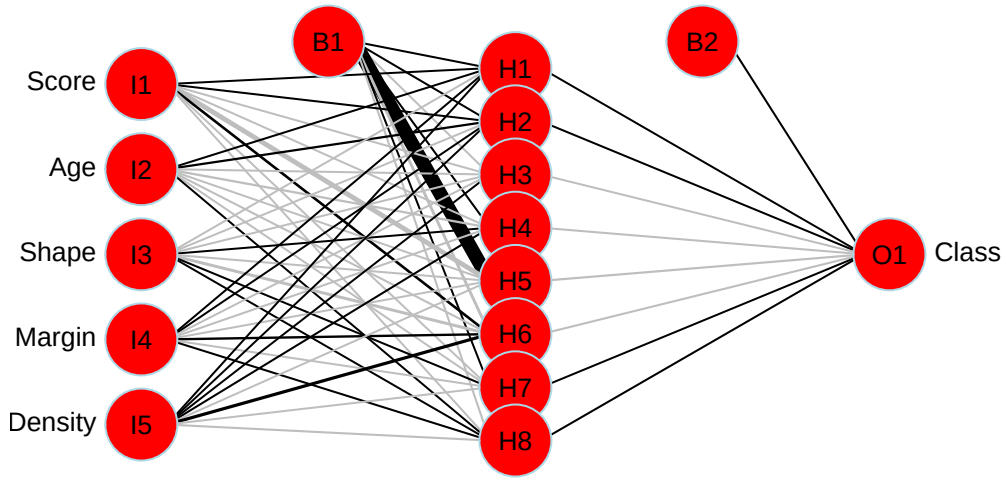
NEURAL NETWORK MODEL

```
library(nnet)
model_nnet<-nnet(Class ~. ,
                 data= train_data,
                 size=8
)
```

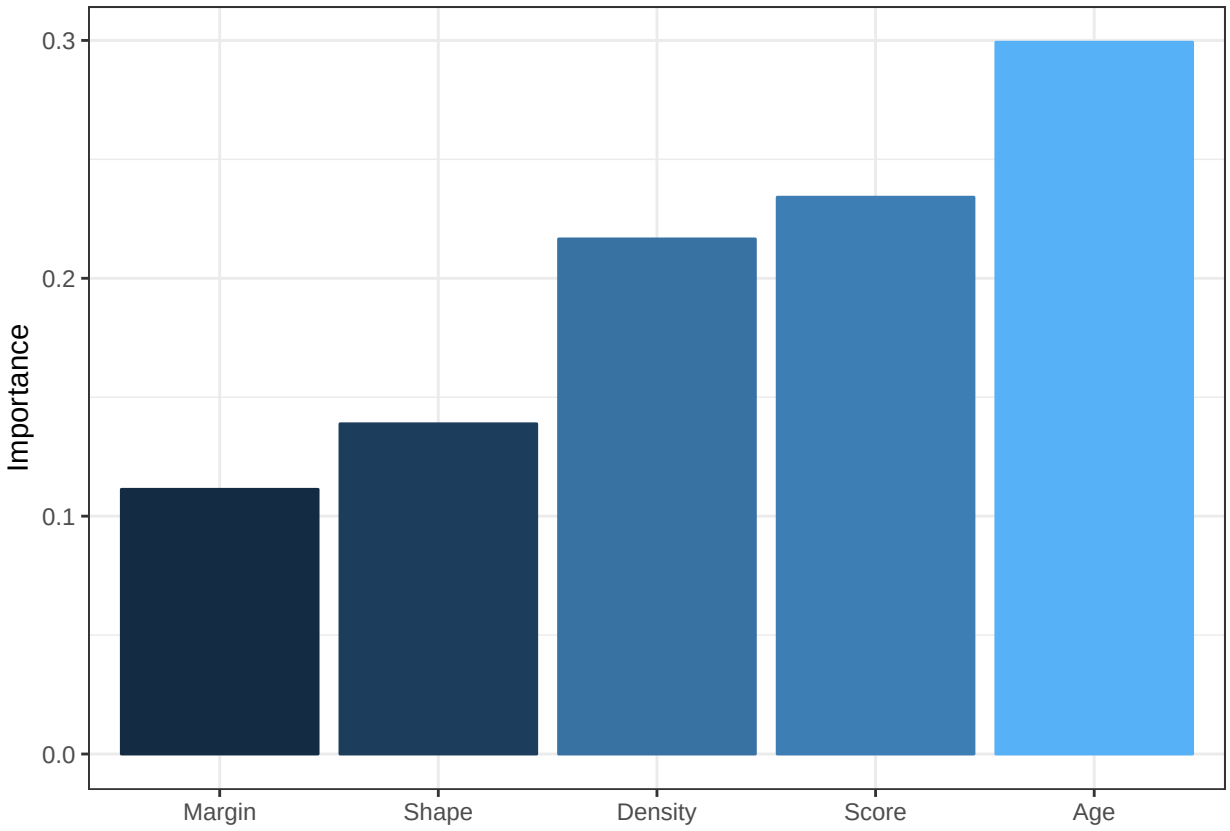
```
## # weights:  57
## initial  value 405.825721
## iter  10 value 378.993447
## iter  20 value 241.736812
## iter  30 value 230.185181
## iter  40 value 221.980350
## iter  50 value 219.683287
## iter  60 value 218.698452
## iter  70 value 217.889981
## iter  80 value 217.242696
## iter  90 value 217.024266
## iter 100 value 215.601013
## final  value 215.601013
## stopped after 100 iterations
```

```
library(NeuralNetTools)
# Plot a neural interpretation diagram for a neural network object
```

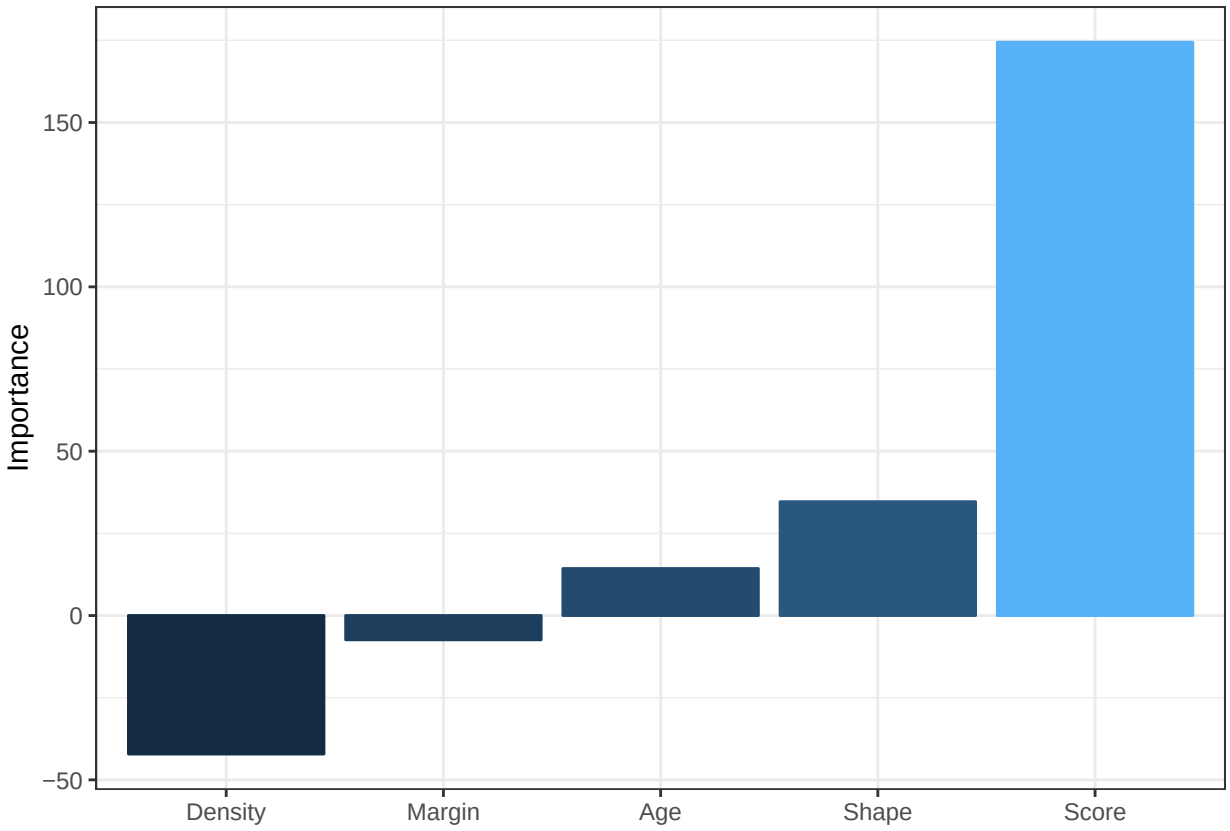
```
plotnet(model_nnet, cex_val =.8,max_sp=T,circle_cex=5,circle_col = 'red')
```




```
#Relative importance of input variables in neural networks using Garson's algorithm
garson(model_nnet)
```




```
olden(model_nnet)
```



Here both the positve and negative value represents relative contibutions of each connection weight among the variables

```
#Predict
predict_nnet <- predict(model_nnet,test_data, type = "class")
```

```
#Draw the crosstable

library(gmodels)
CrossTable(test_data$Class,predict_nnet,prop.chisq = F,prop.r = F,prop.c = F,dnn =c("Actual Diagnosis",
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  248
##
##
##                  | Predict Diagnosis
## Actual Diagnosis |    benign | malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           benign |       117 |        11 |       128 |
##                  |     0.472 |     0.044 |           |
```

```
## -----------------|-----------|-----------|-----------|
##        malignant |        24 |        96 |       120 |
##                  |     0.097 |     0.387 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |       141 |       107 |       248 |
## -----------------|-----------|-----------|-----------|
##
##
```