Machine learning models for cancer predictive analysis

Natalia

28 May 2019

```
library(mlbench)
data(BreastCancer)
data <- BreastCancer
View(data)
```

Analyse the dataset and tidy it up.

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

```
##       Id             Cl.thickness   Cell.size     Cell.shape
##  Length:699         1      :145   1      :384   1      :353
##  Class :character   5      :130   10     : 67   2      : 59
##  Mode  :character   3      :108   3      : 52   10     : 58
##                     4      : 80   2      : 45   3      : 56
##                     10     : 69   4      : 40   4      : 44
##                     2      : 50   5      : 30   5      : 34
##                     (Other):117   (Other): 81   (Other): 95
##  Marg.adhesion  Epith.c.size  Bare.nuclei   Bl.cromatin  Normal.nucleoli
##  1      :407   2      :386   1      :402   2      :166   1      :443
##  2      : 58   3      : 72   10     :132   3      :165   10     : 61
##  3      : 58   4      : 48   2      : 30   1      :152   3      : 44
##  10     : 55   1      : 47   5      : 30   7      : 73   2      : 36
##  4      : 33   6      : 41   3      : 28   4      : 40   8      : 24
##  8      : 25   5      : 39   (Other): 61   5      : 34   6      : 22
##  (Other): 63   (Other): 66   NA's   : 16   (Other): 69   (Other): 69
##     Mitoses          Class
##  1      :579   benign   :458
##  2      : 35   malignant:241
##  3      : 33
##  10     : 14
##  4      : 12
##  7      :  9
##  (Other): 17
```

```
str(data)
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ Id             : chr  "1000025" "1002945" "1015425" "1016277" ...
##  $ Cl.thickness   : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 5 5 3 6 4 8 1 2 2 4 ...
##  $ Cell.size      : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 1 1 2 ...
##  $ Cell.shape     : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 2 1 1 ...
##  $ Marg.adhesion  : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 5 1 1 3 8 1 1 1 1 ...
##  $ Epith.c.size   : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 2 7 2 3 2 7 2 2 2 2 ...
##  $ Bare.nuclei    : Factor w/ 10 levels "1","2","3","4",..: 1 10 2 4 1 10 10 1 1 1 ...
##  $ Bl.cromatin    : Factor w/ 10 levels "1","2","3","4",..: 3 3 3 3 3 9 3 3 1 2 ...
##  $ Normal.nucleoli: Factor w/ 10 levels "1","2","3","4",..: 1 2 1 7 1 7 1 1 1 1 ...
##  $ Mitoses        : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 5 1 ...
##  $ Class          : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
```

```r
head(data)
```

```
##        Id Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size
## 1 1000025            5         1          1             1            2
## 2 1002945            5         4          4             5            7
## 3 1015425            3         1          1             1            2
## 4 1016277            6         8          8             1            3
## 5 1017023            4         1          1             3            2
## 6 1017122            8        10         10             8            7
##   Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses     Class
## 1           1           3               1       1    benign
## 2          10           3               2       1    benign
## 3           2           3               1       1    benign
## 4           4           3               7       1    benign
## 5           1           3               1       1    benign
## 6          10           9               7       1 malignant
```

```r
dim(data)
```

```
## [1] 699  11
```

```r
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()
```

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

```
##              Id    Cl.thickness       Cell.size      Cell.shape
##               0               0               0               0
##   Marg.adhesion    Epith.c.size     Bare.nuclei     Bl.cromatin
##               0               0              16               0
## Normal.nucleoli         Mitoses           Class
##               0               0               0
```

```r
#clean up data
#remove NAs
data <- na.omit(data)
dim(data)
```

```
## [1] 683  11
```

```r
head(data)
```

```
##        Id Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size
## 1 1000025            5         1          1             1            2
## 2 1002945            5         4          4             5            7
## 3 1015425            3         1          1             1            2
## 4 1016277            6         8          8             1            3
## 5 1017023            4         1          1             3            2
## 6 1017122            8        10         10             8            7
##   Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses     Class
## 1           1           3               1       1    benign
## 2          10           3               2       1    benign
## 3           2           3               1       1    benign
## 4           4           3               7       1    benign
## 5           1           3               1       1    benign
## 6          10           9               7       1 malignant
```

```
# Data type is character:
data <- as.data.frame(data, stringsAsFactors=T)
head(data)
```

```
##        Id Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size
## 1 1000025            5         1          1             1            2
## 2 1002945            5         4          4             5            7
## 3 1015425            3         1          1             1            2
## 4 1016277            6         8          8             1            3
## 5 1017023            4         1          1             3            2
## 6 1017122            8        10         10             8            7
##   Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses     Class
## 1           1           3               1       1    benign
## 2          10           3               2       1    benign
## 3           2           3               1       1    benign
## 4           4           3               7       1    benign
## 5           1           3               1       1    benign
## 6          10           9               7       1 malignant
```

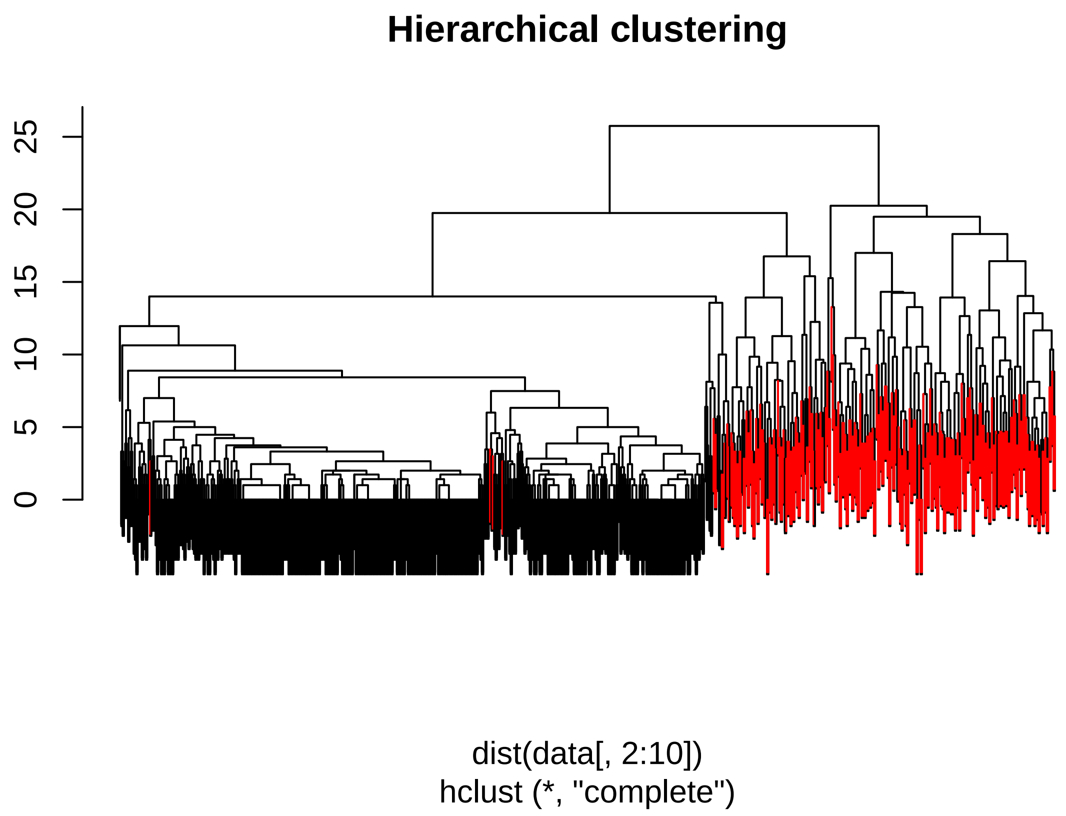
```
data$Class <- as.factor(data$Class)
sapply(data,mode)
```

```
##              Id    Cl.thickness       Cell.size      Cell.shape
##     "character"       "numeric"       "numeric"       "numeric"
##   Marg.adhesion    Epith.c.size     Bare.nuclei     Bl.cromatin
##       "numeric"       "numeric"       "numeric"       "numeric"
## Normal.nucleoli         Mitoses           Class
##       "numeric"       "numeric"       "numeric"
```

DATA EXPLORATION

Hierarchical clustering

```
library(sparcl)
hc <- hclust(dist(data[,2:10]), method = "complete")
ColorDendrogram(hc,y=data$Class, main = "Hierarchical clustering", branchlength=5)
```



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

K-means clustering

```
fit <- kmeans(data[,c(2:10)], 2)
names(fit)
```

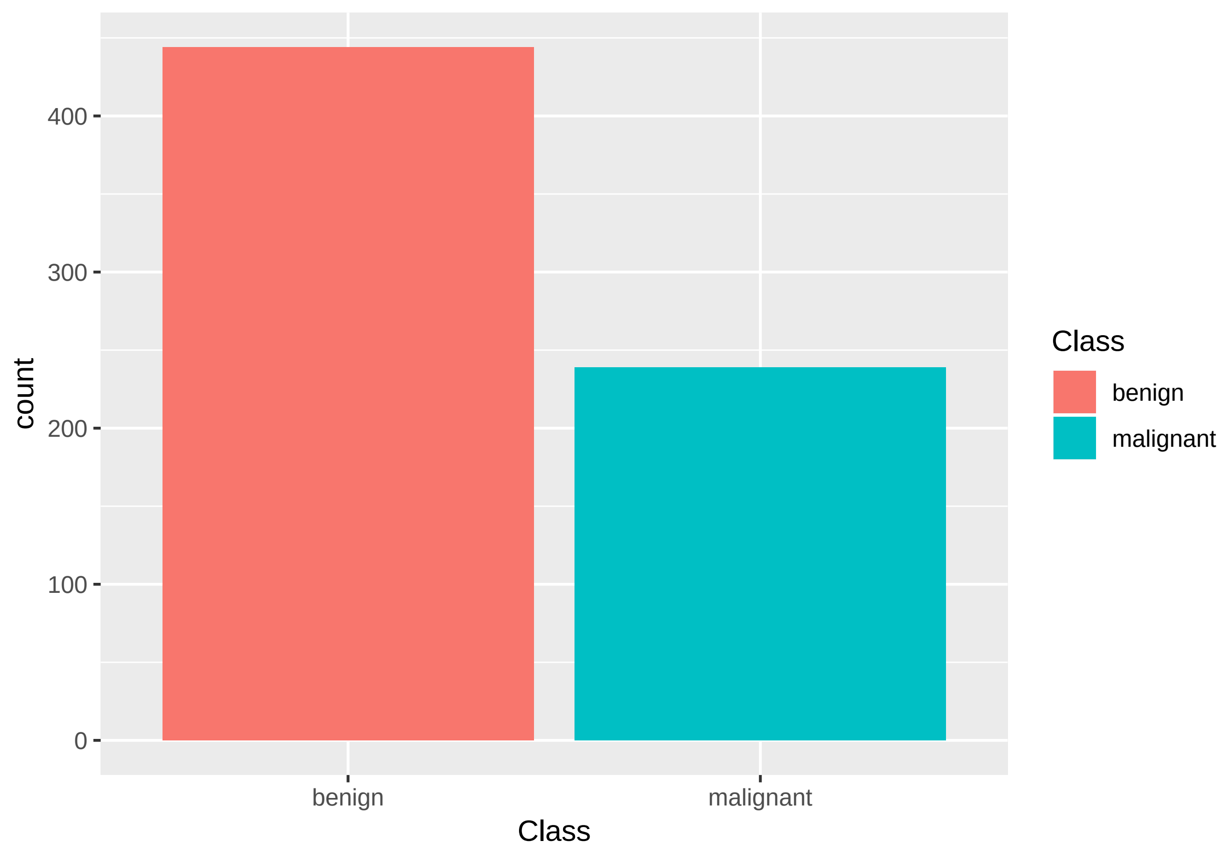
```
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
#k-means did a fairly good job
table(data.frame(fit$cluster,data[,11]))
```

```
##            data...11.
## fit.cluster benign malignant
##           1    435        18
##           2      9       221
```

Response variable for classification

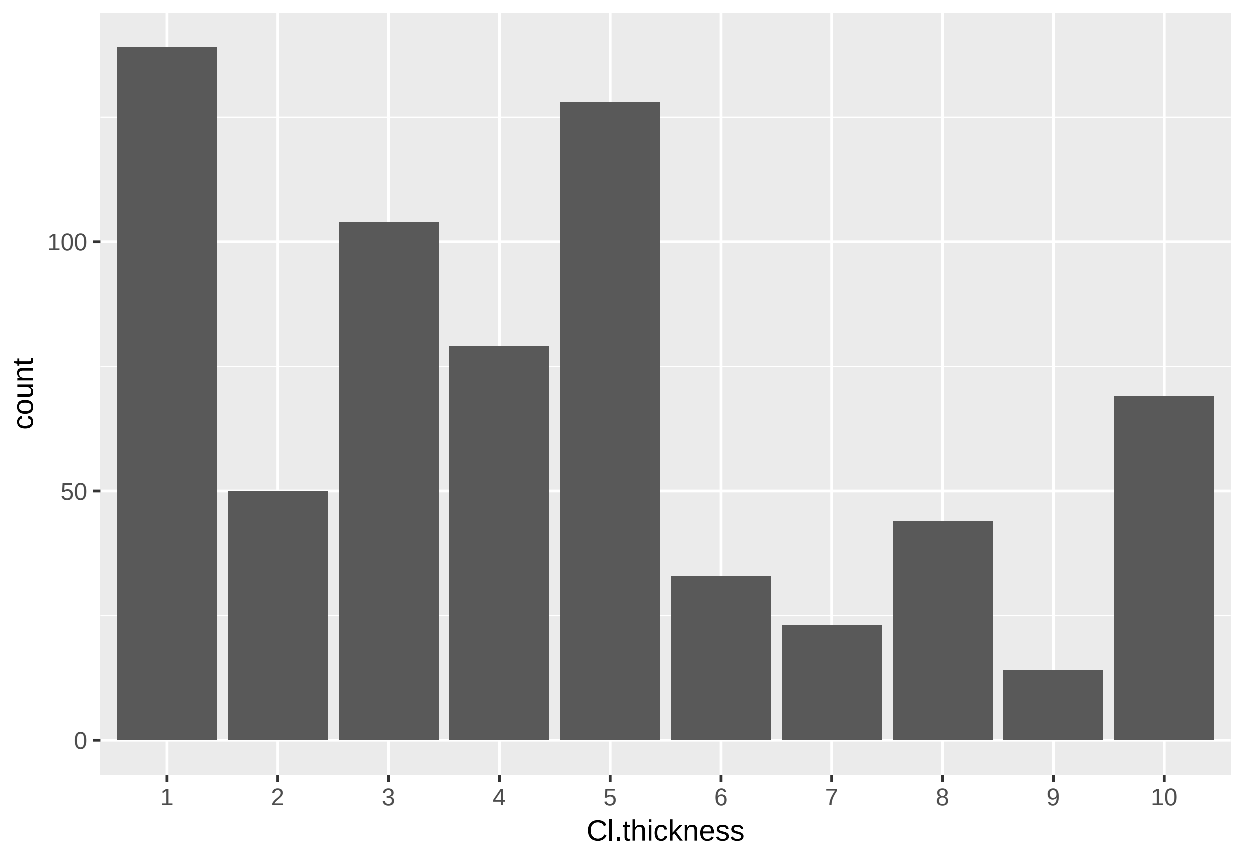
```
library(ggplot2)

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



Response variable for regression

```r
ggplot(data, aes(x = Cl.thickness)) +
  geom_histogram(binwidth = 1, stat = "count")
```

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



Principal Component Analysis

```r
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)
```

```
# impute missing data
library(mice)
```

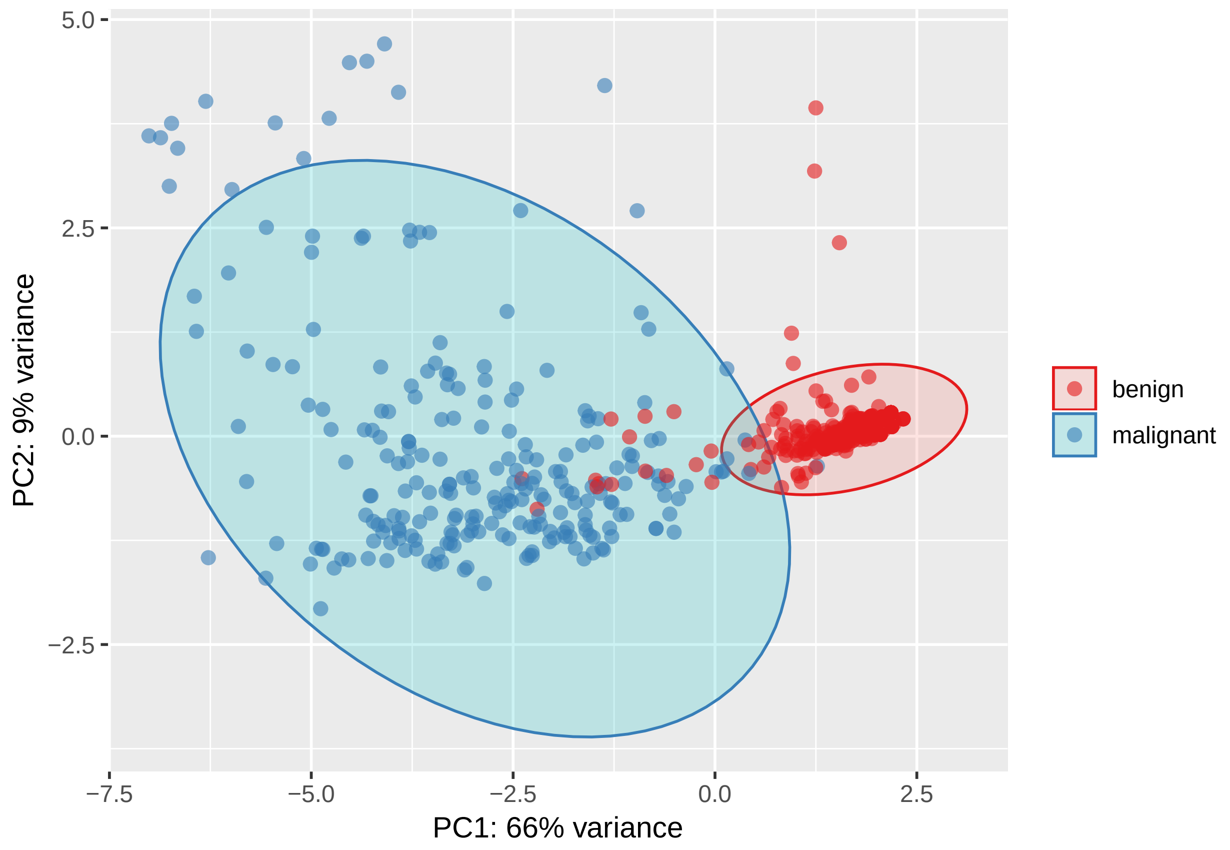
```
## Loading required package: lattice

##
## 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[,2:10] <- apply(data[, 2:10], 2, function(x) as.numeric(as.character(x)))
dataset_impute <- mice(data[, 2:10],  print = FALSE)
data <- cbind(data[, 11, drop = FALSE], mice::complete(dataset_impute, 1))
data$Class <- as.factor(data$Class)
```

```
# perform pca and extract scores:
pcaOutput <- pca(t(data[, -1]), printDropped = FALSE, scale = TRUE, center = TRUE)
pcaOutput2 <- as.data.frame(pcaOutput$scores)
```

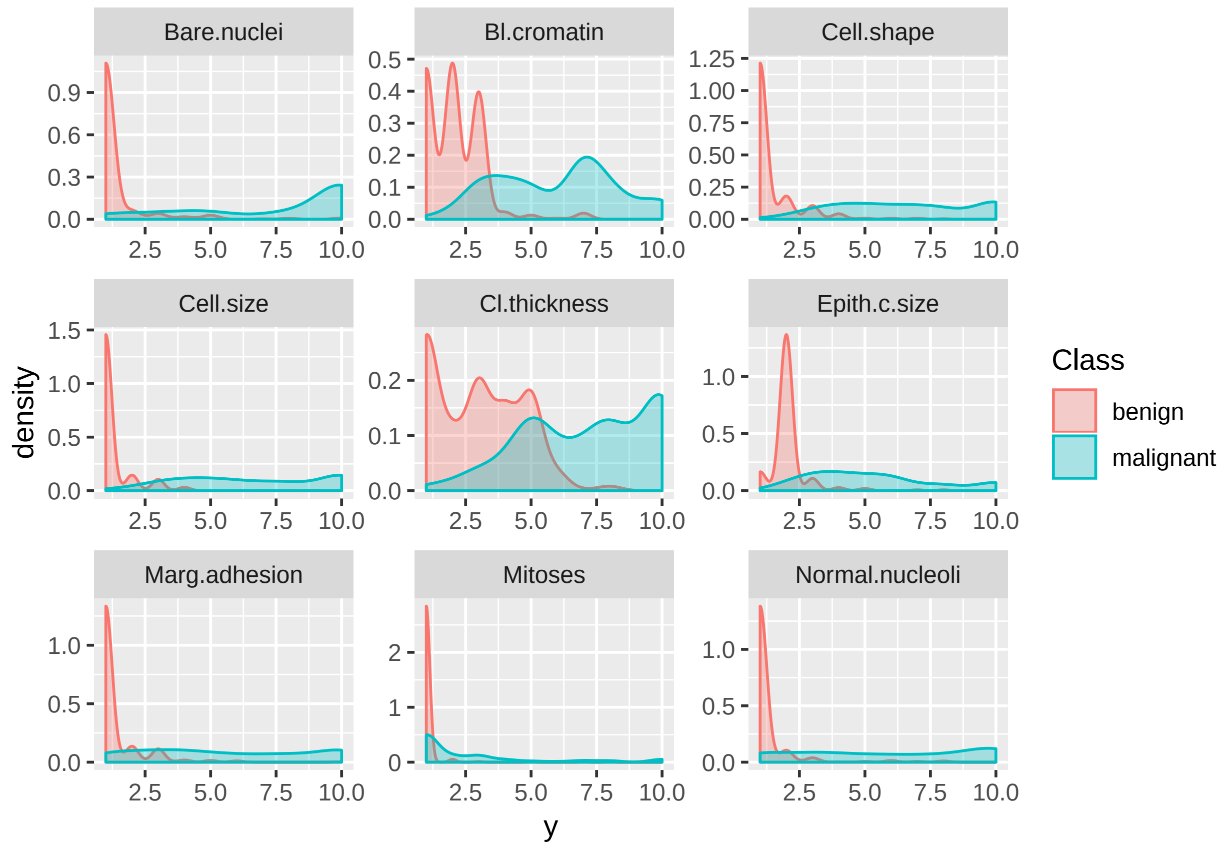
```
# define groups for plotting:
pcaOutput2$groups <- data$Class

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)))


#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

```
library(tidyr)

gather(data, x, y, Cl.thickness:Mitoses) %>%
  ggplot(aes(x = y, color = Class, fill = Class)) +
    geom_density(alpha = 0.3) +
    facet_wrap( ~ x, scales = "free", ncol = 3)
```



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)
```

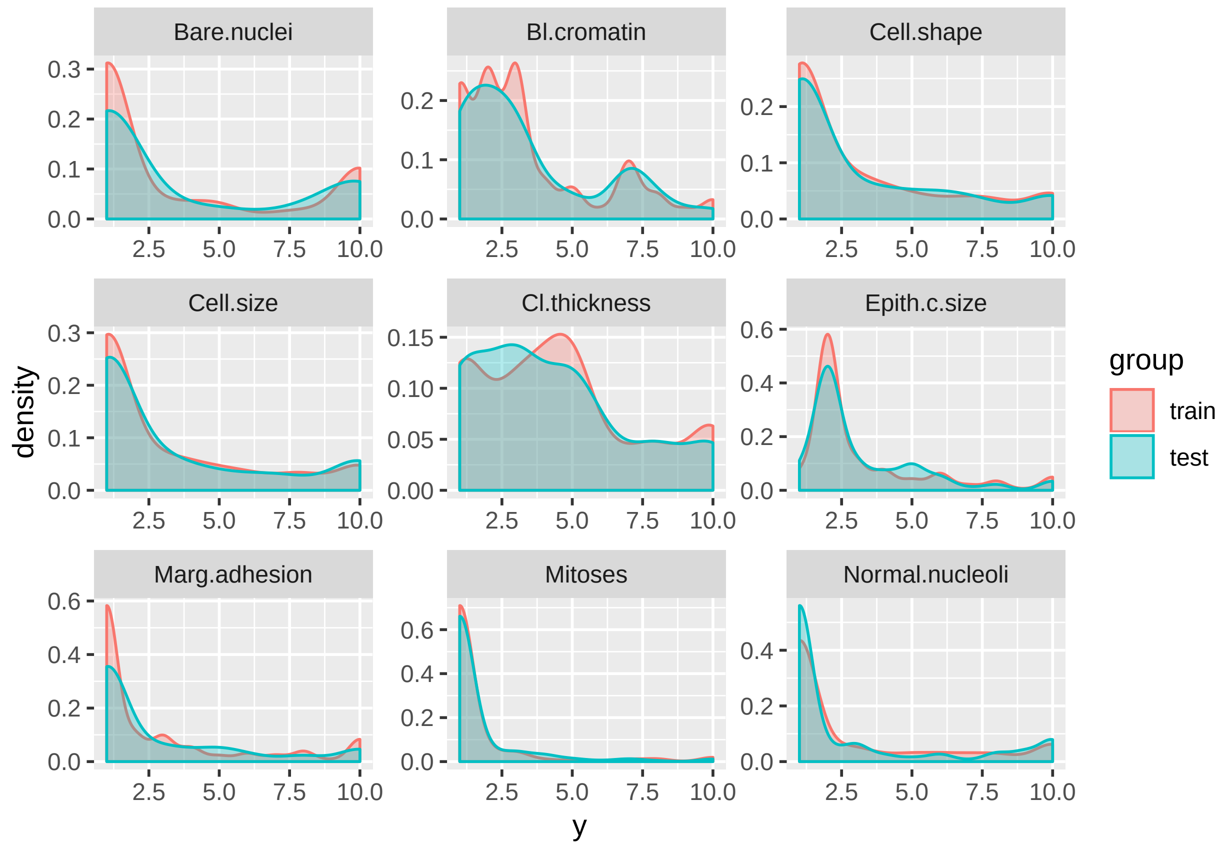
```
##
## Attaching package: 'caret'

## 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, Cl.thickness:Mitoses) %>%
  ggplot(aes(x = y, color = group, fill = group)) +
    geom_density(alpha = 0.3) +
    facet_wrap( ~ x, scales = "free", ncol = 3)
```



REGRESSION

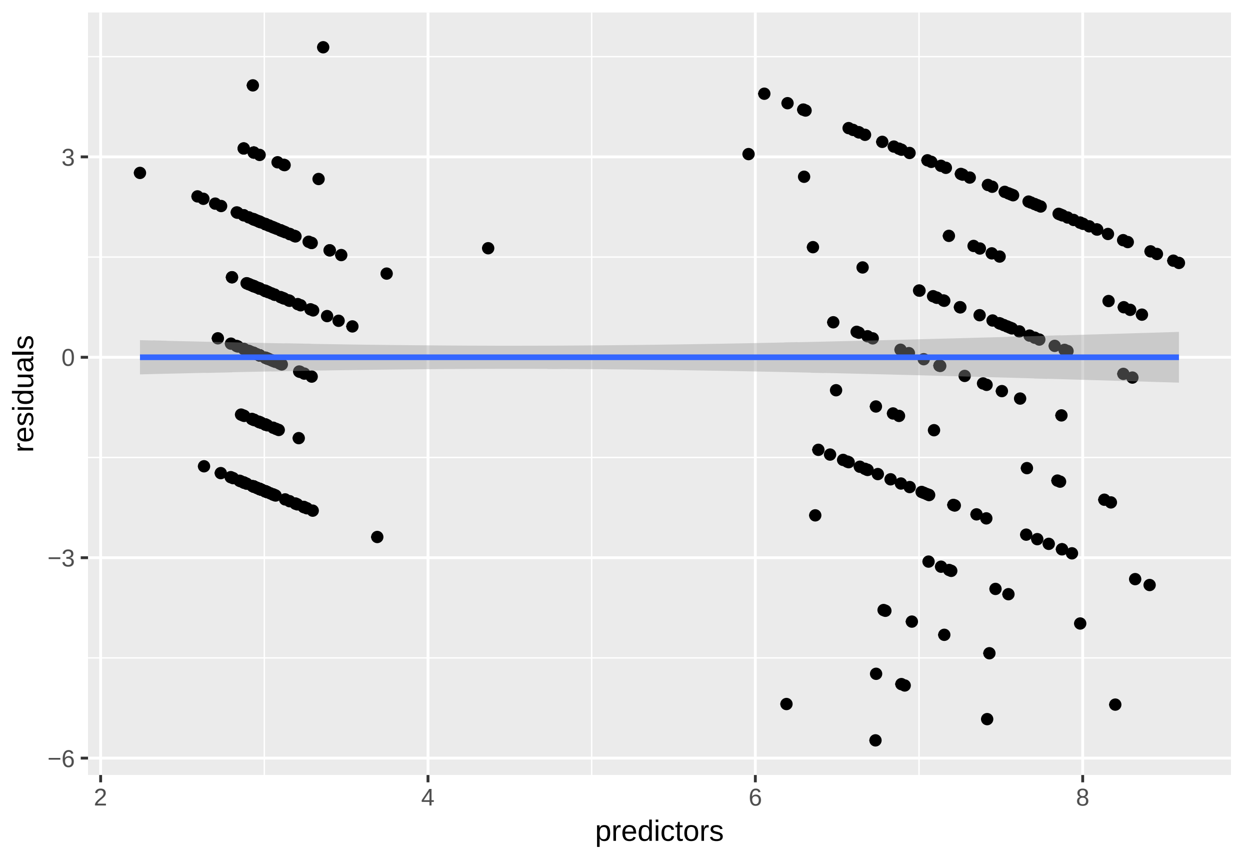
```
set.seed(42)
model_glm <- caret::train(Cl.thickness ~ .,
                          data = train_data,
                          method = "glm",
                          preProcess = c("scale", "center"),
                          trControl = trainControl(method = "repeatedcv",
                                                   number = 10,
                                                   repeats = 10,
                                                   savePredictions = TRUE,
                                                   verboseIter = FALSE))
```

```
model_glm
```

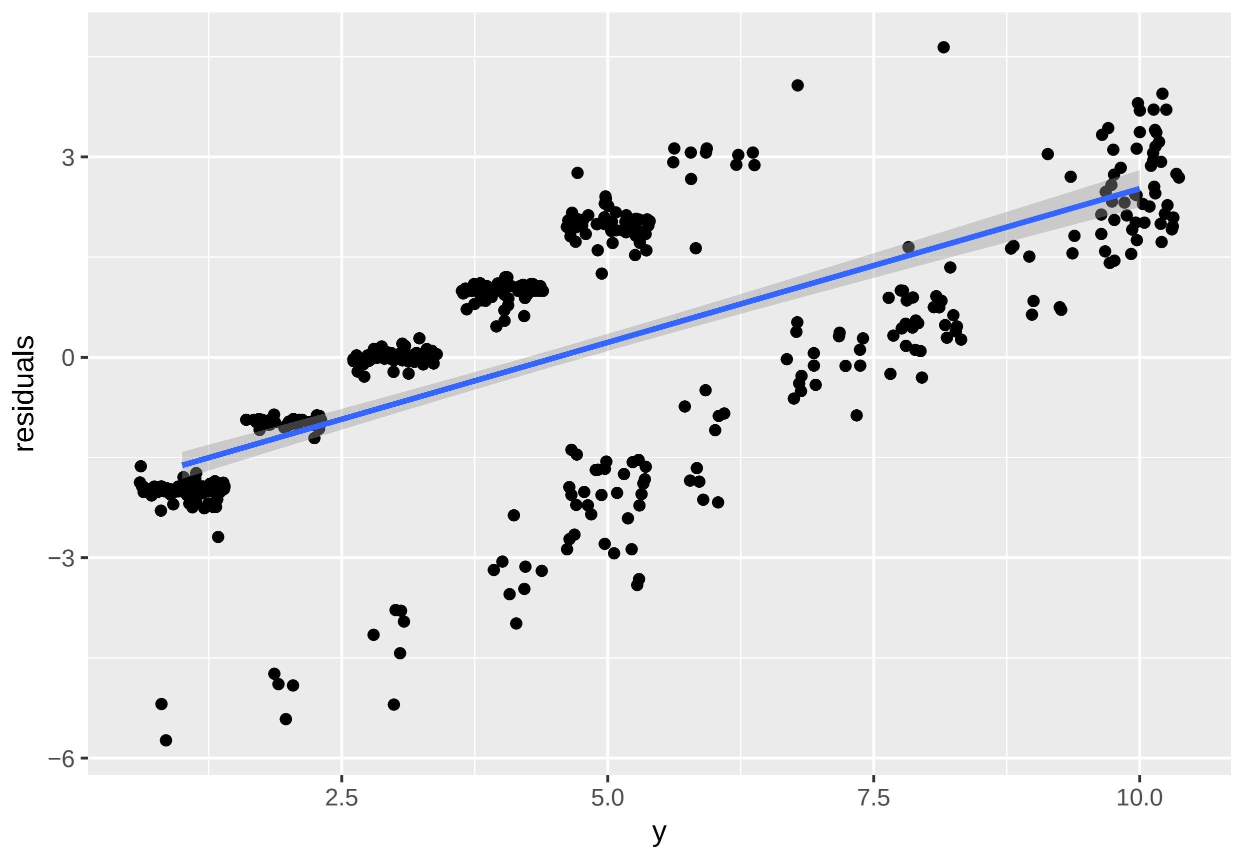
```
## Generalized Linear Model
##
## 479 samples
##   9 predictor
##
## Pre-processing: scaled (9), centered (9)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 432, 431, 432, 431, 431, 431, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   1.972314  0.5254215  1.648832
```

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

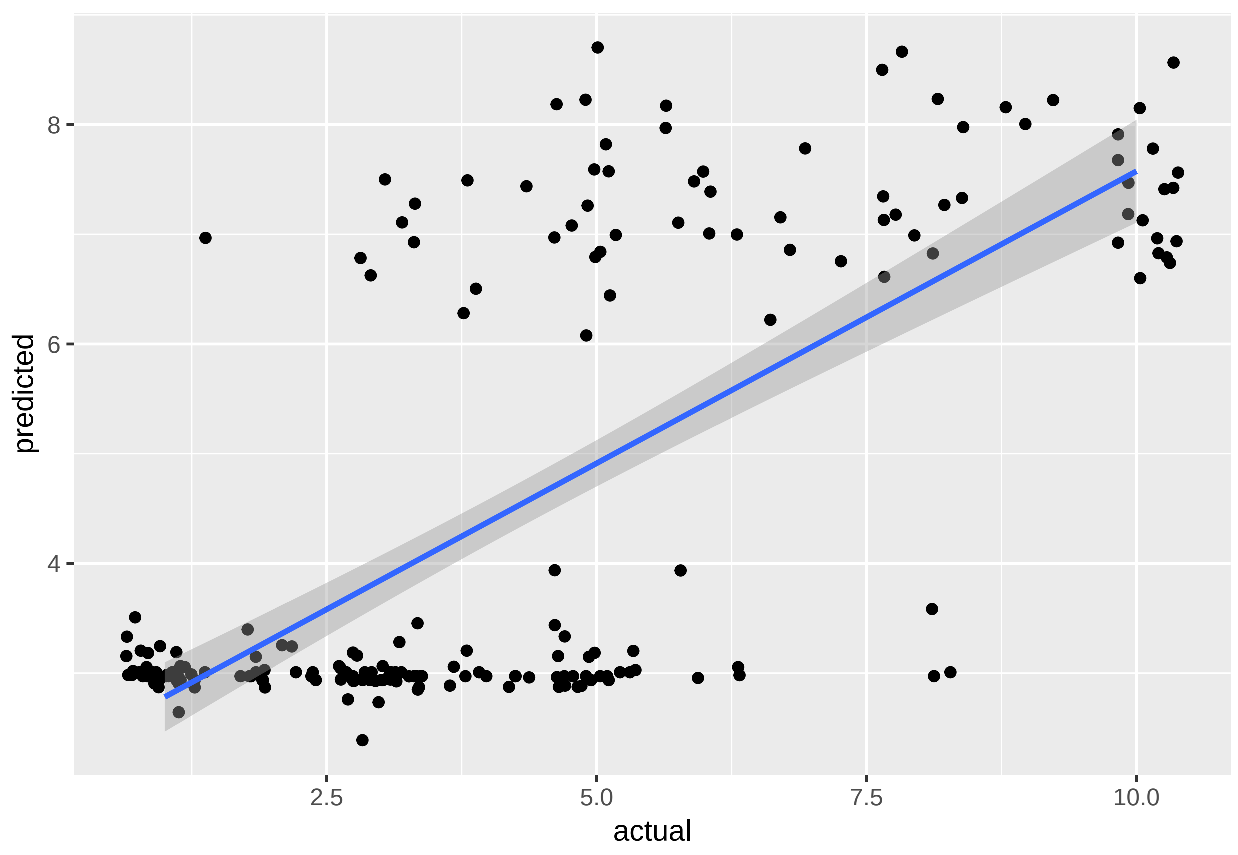
```
# model_glm$finalModel$linear.predictors == model_glm$finalModel$fitted.values
data.frame(residuals = resid(model_glm),
           predictors = model_glm$finalModel$linear.predictors) %>%
  ggplot(aes(x = predictors, y = residuals)) +
    geom_jitter() +
    geom_smooth(method = "lm")
```



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



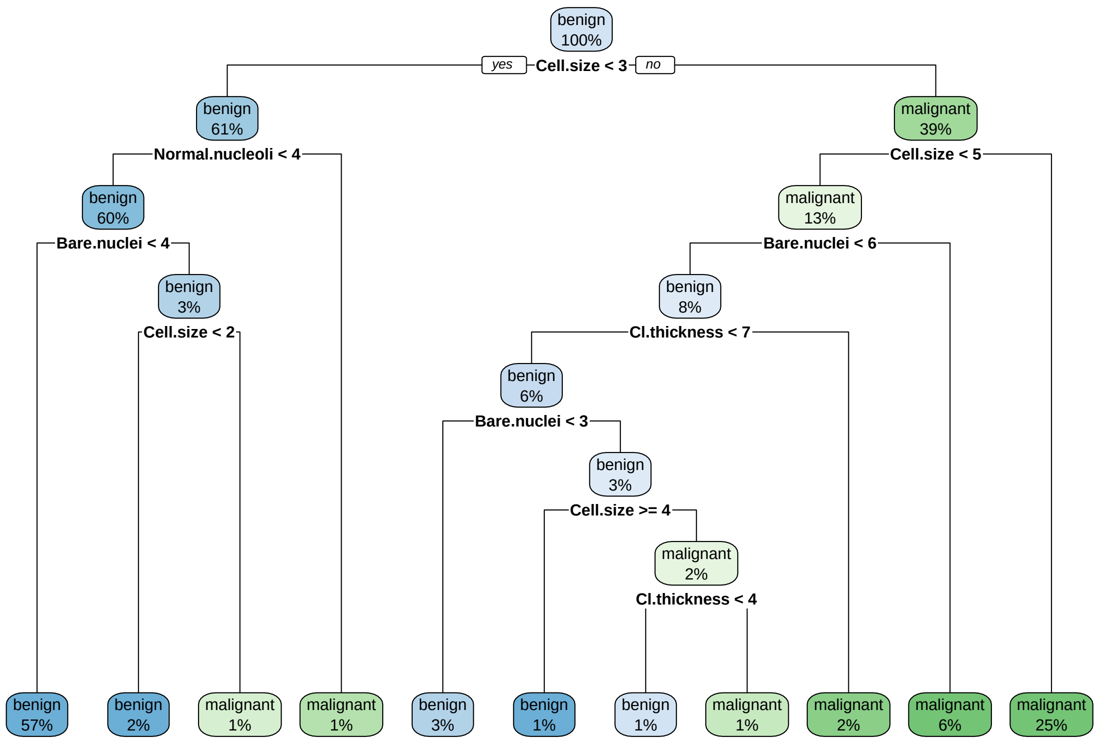
```
data.frame(actual = test_data$Cl.thickness,
           predicted = predictions) %>%
  ggplot(aes(x = actual, y = predicted)) +
    geom_jitter() +
    geom_smooth(method = "lm")
```



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)
```



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
```

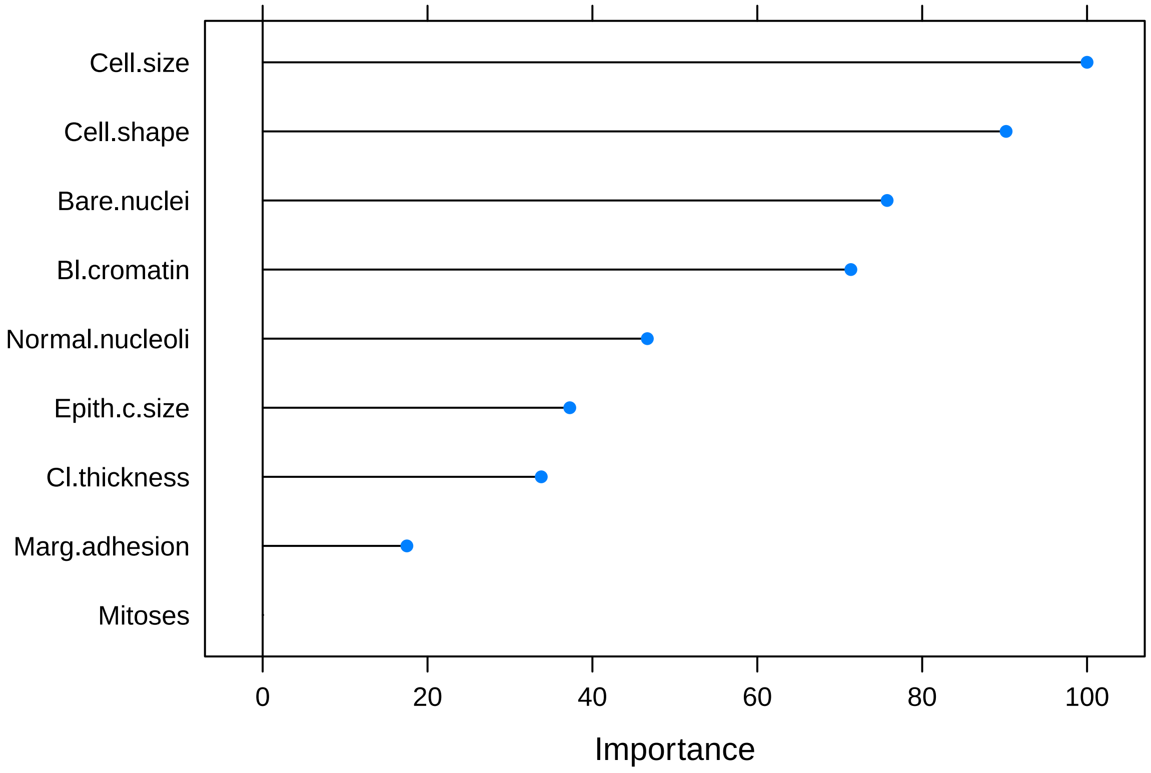
```
##           benign malignant class.error
## benign       304         7  0.02250804
## malignant      5       163  0.02976190
```

Feature Importance

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

```
##       Cell.size      Cell.shape     Bare.nuclei     Bl.cromatin
##       43.936945       39.840595       33.820345       31.984813
## Normal.nucleoli    Epith.c.size    Cl.thickness   Marg.adhesion
##       21.686039       17.761202       16.318817        9.518437
##         Mitoses
##        2.220633
```

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

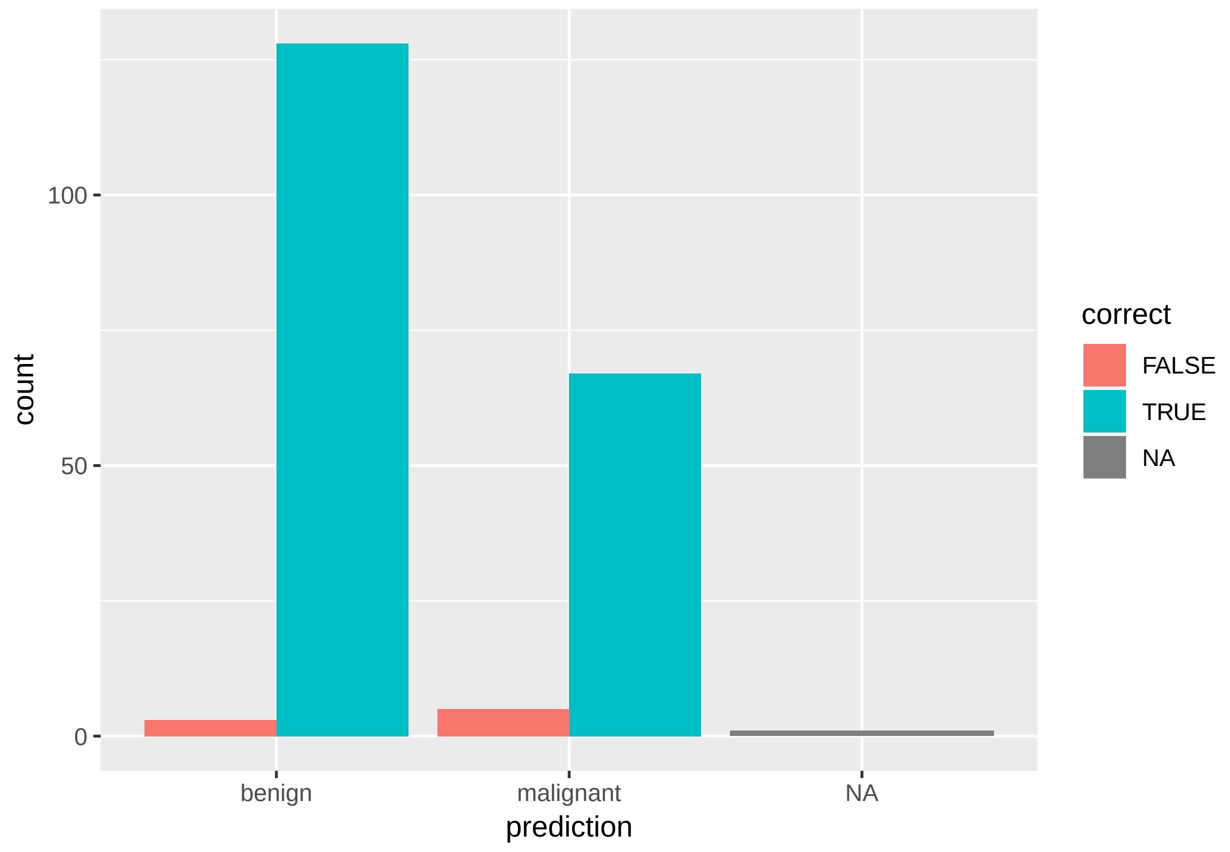


Predicting test data

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

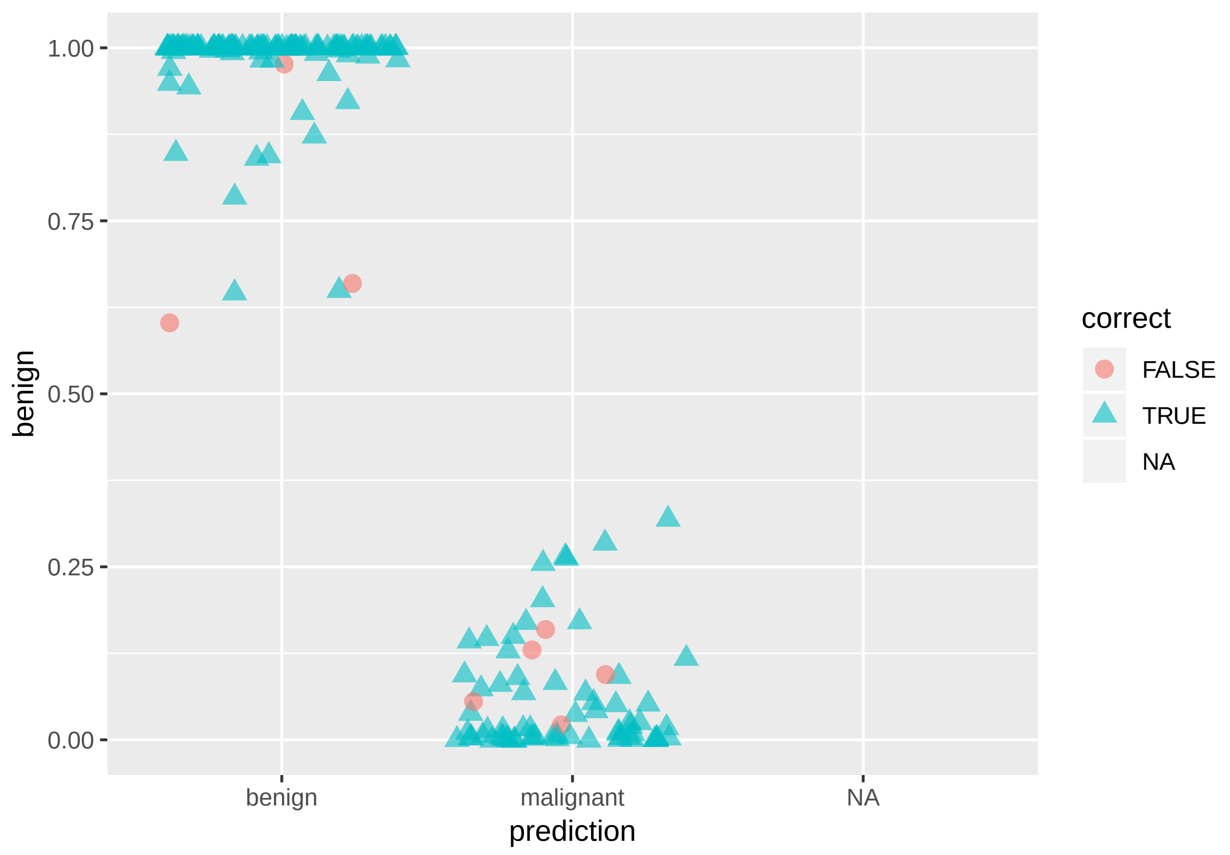
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       128         4
##   malignant      5        67
##
##                Accuracy : 0.9559
##                  95% CI : (0.9179, 0.9796)
##     No Information Rate : 0.652
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9031
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9624
##             Specificity : 0.9437
##          Pos Pred Value : 0.9697
##          Neg Pred Value : 0.9306
##              Prevalence : 0.6520
##          Detection Rate : 0.6275
##    Detection Prevalence : 0.6471
##       Balanced Accuracy : 0.9530
##
##        'Positive' Class : benign
##
```

```
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")
```



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

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



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.
```

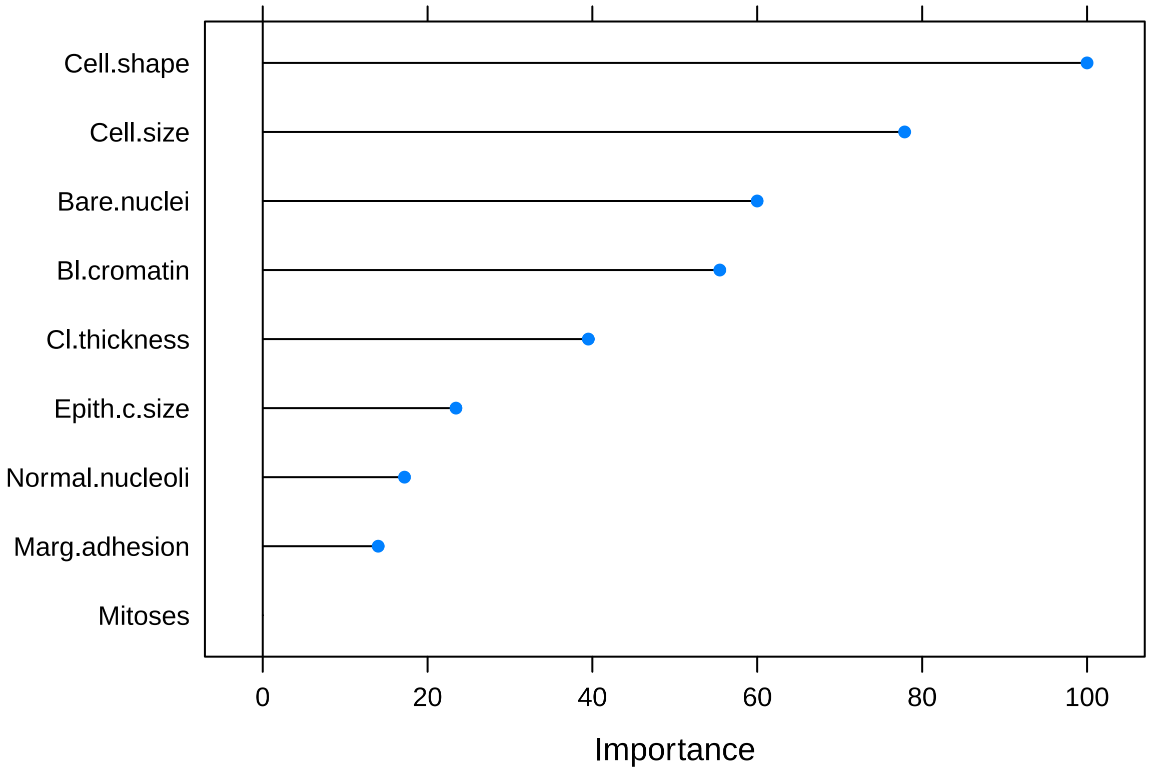
```
set.seed(42)
library(xgboost)
```

```
##
## 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)
```



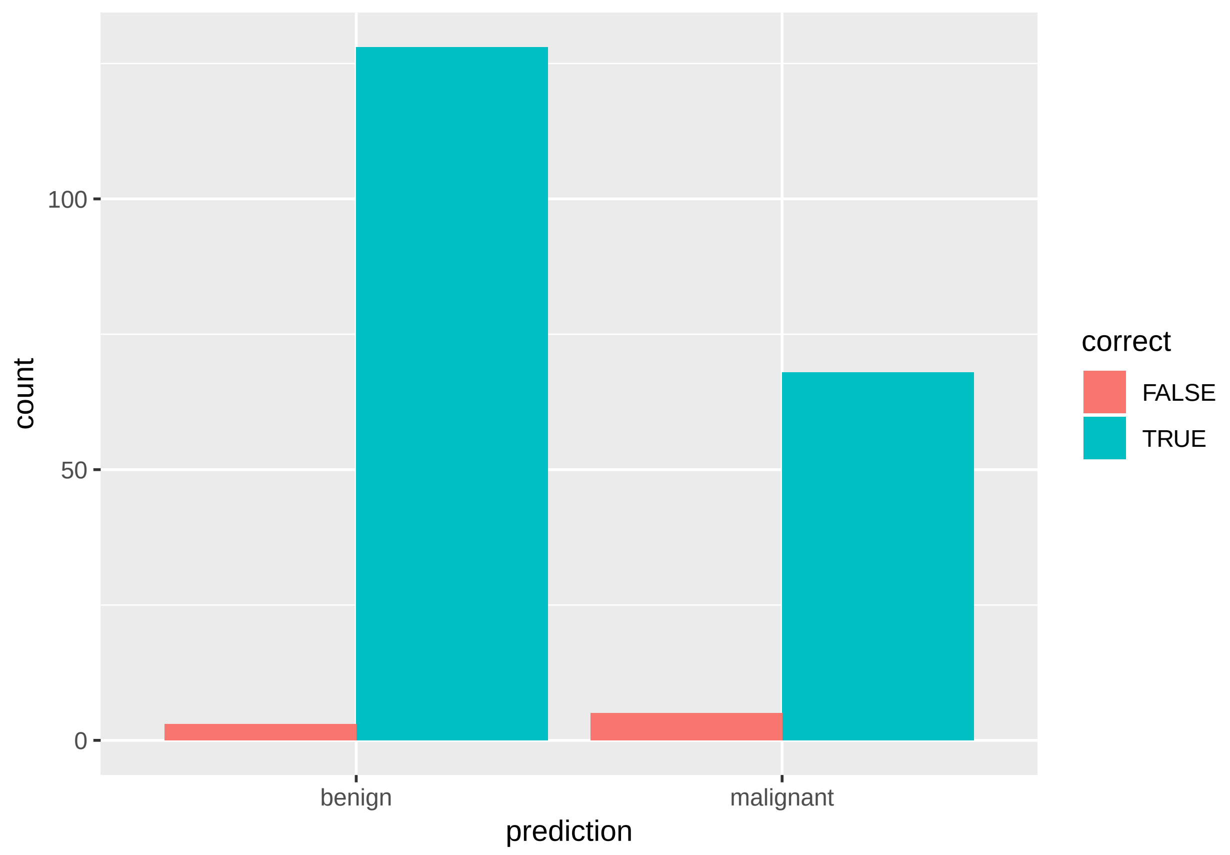
#Predicting test data

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

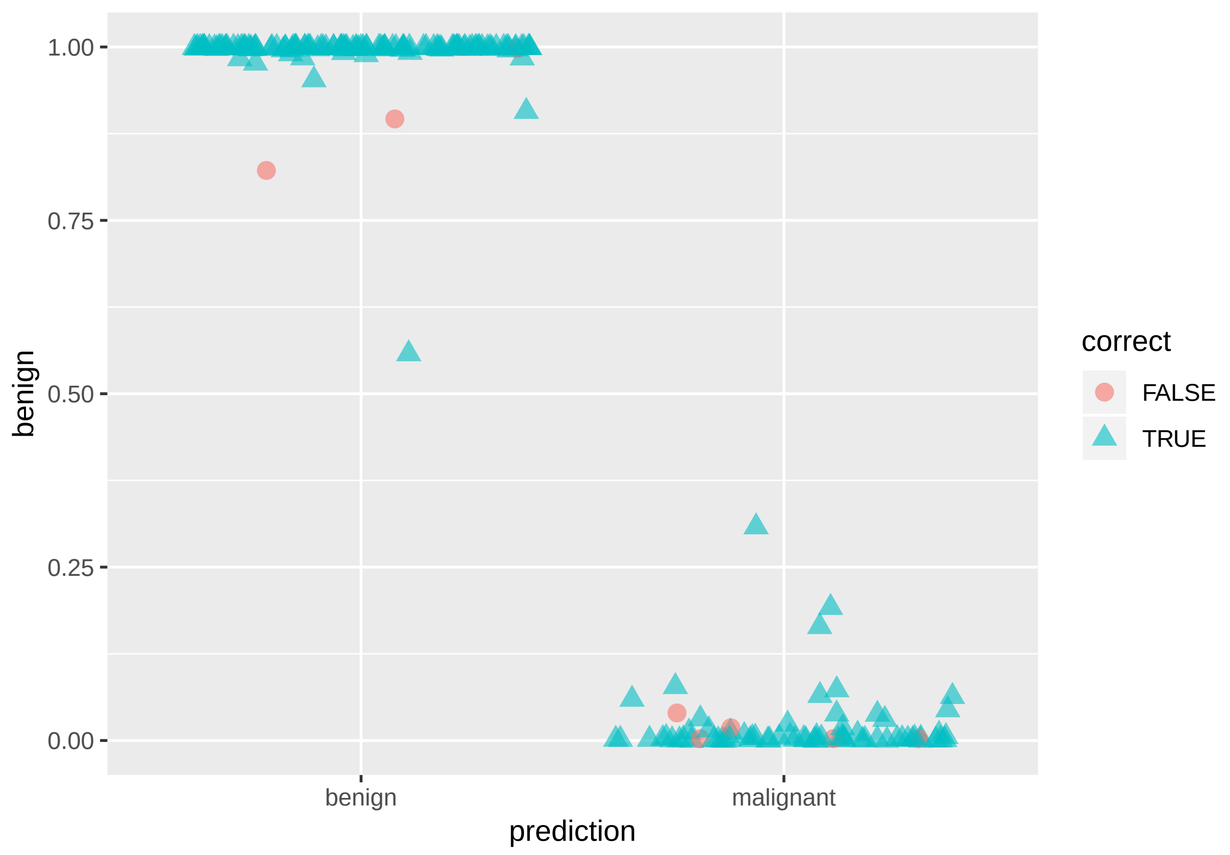
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       128         3
##   malignant      5        68
##
```

```
##                Accuracy : 0.9608
##                  95% CI : (0.9242, 0.9829)
##     No Information Rate : 0.652
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9142
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9624
##             Specificity : 0.9577
##          Pos Pred Value : 0.9771
##          Neg Pred Value : 0.9315
##              Prevalence : 0.6520
##          Detection Rate : 0.6275
##    Detection Prevalence : 0.6422
##       Balanced Accuracy : 0.9601
##
##        'Positive' Class : benign
##
```

```
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

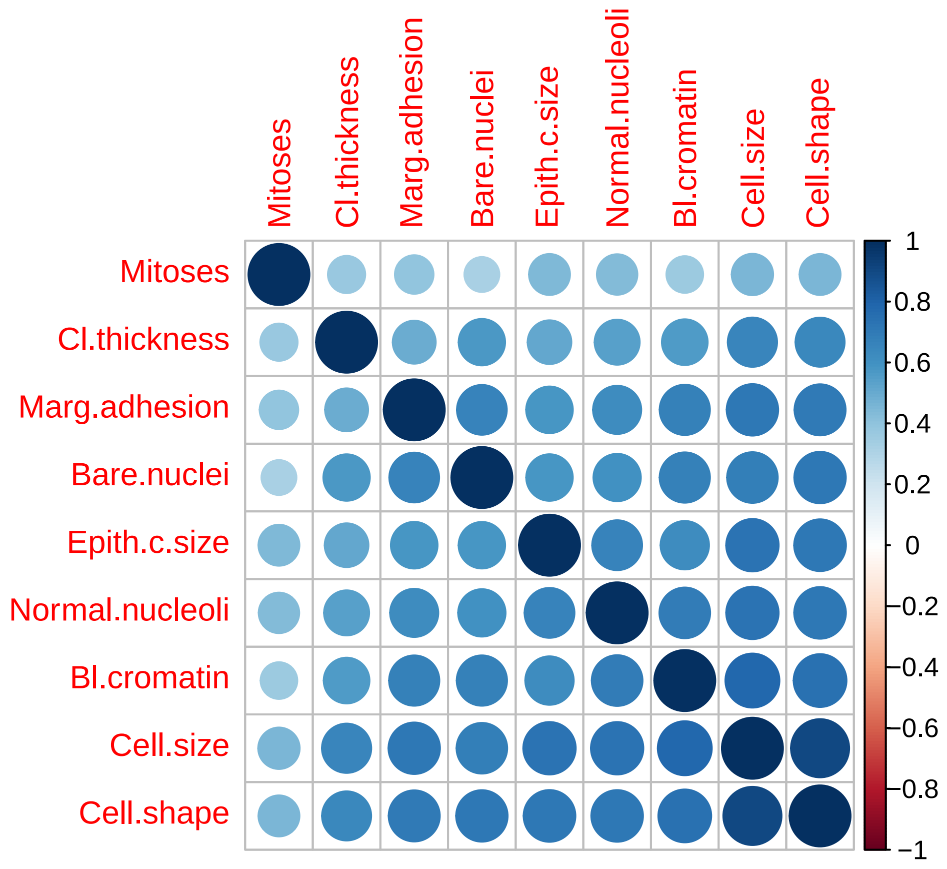
Performing feature selection on the whole dataset would lead to prediction bias, we therefore need to run the whole modeling process on the training data alone.

Correlation

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
# calculate correlation matrix
corMatMy <- cor(train_data[, -1])
corrplot(corMatMy, order = "hclust")
```



```
#Correlations between all features are calculated and visualised.
#Removing all features with a correlation higher than 0.7,
#keeping the feature with the lower mean.
```

```
#Apply correlation filter at 0.70:
highlyCor <- colnames(train_data[, -1])[findCorrelation(corMatMy, cutoff = 0.7, verbose = TRUE)]
```

```
## Compare row 2  and column  3 with corr  0.908
##   Means:  0.709 vs 0.594 so flagging column 2
## Compare row 3  and column  7 with corr  0.749
##   Means:  0.67 vs 0.569 so flagging column 3
## All correlations <= 0.7
```

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

```
## [1] "Cell.size"  "Cell.shape"
```

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

GRID SEARCH WITH CARET

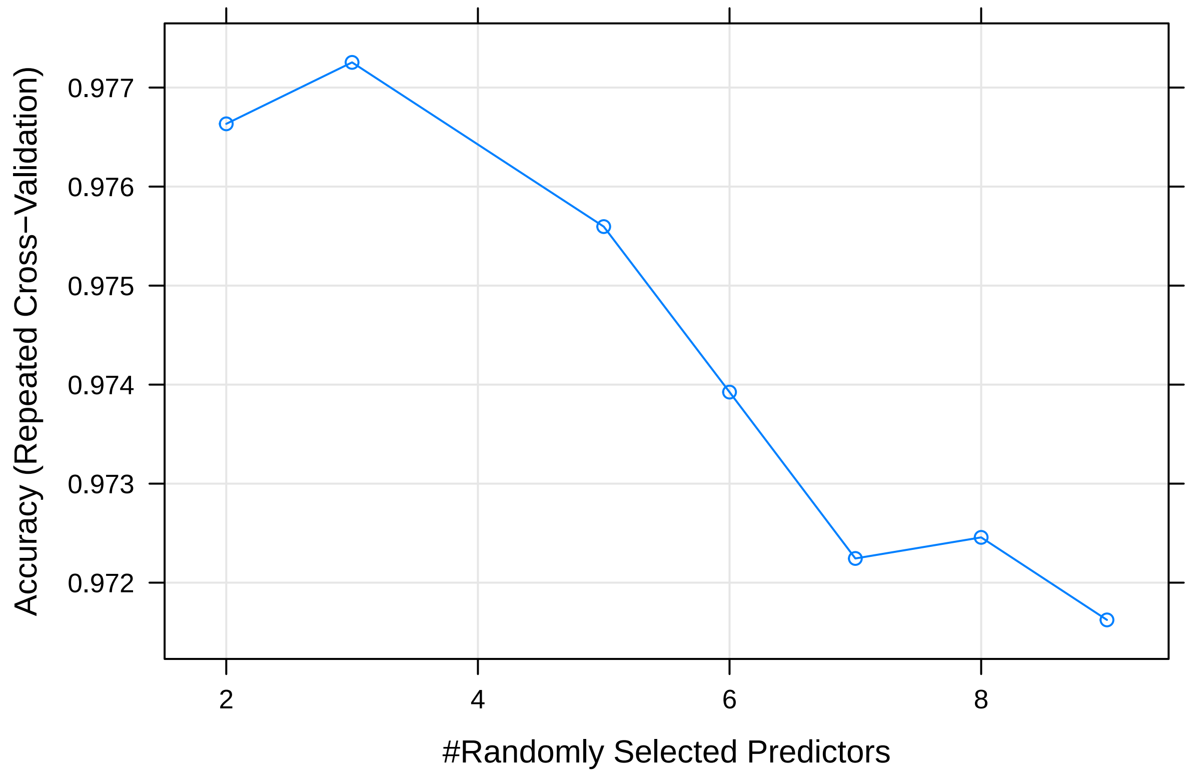
Automatic Grid

```
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
##
## 479 samples
##   9 predictor
##   2 classes: 'benign', 'malignant'
##
## Pre-processing: scaled (9), centered (9)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 432, 431, 431, 431, 431, 431, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9766336  0.9490429
##   3     0.9772542  0.9503366
##   5     0.9755961  0.9465320
##   6     0.9739246  0.9427764
##   7     0.9722446  0.9389782
##   8     0.9724574  0.9394126
##   9     0.9716239  0.9375003
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 3.
```

```
plot(model_rf_tune_auto)
```



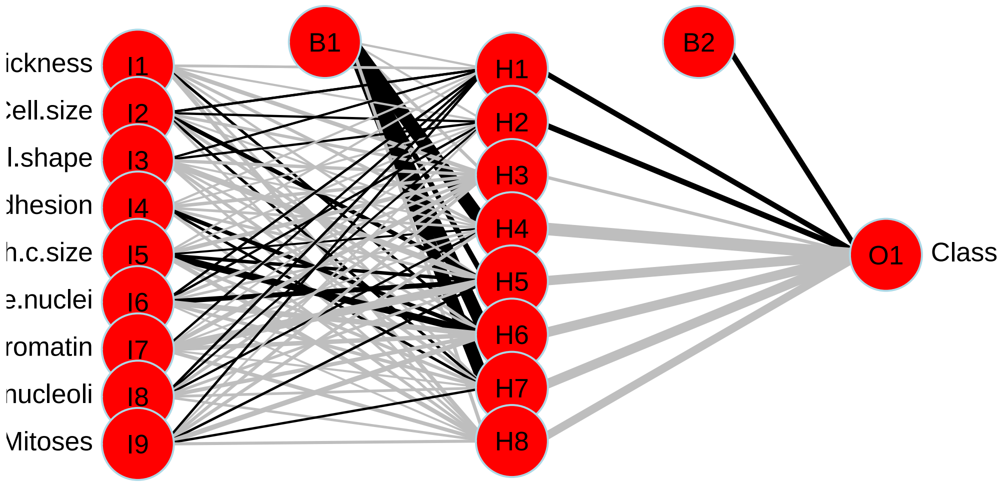
NEURAL NETWORK MODEL

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

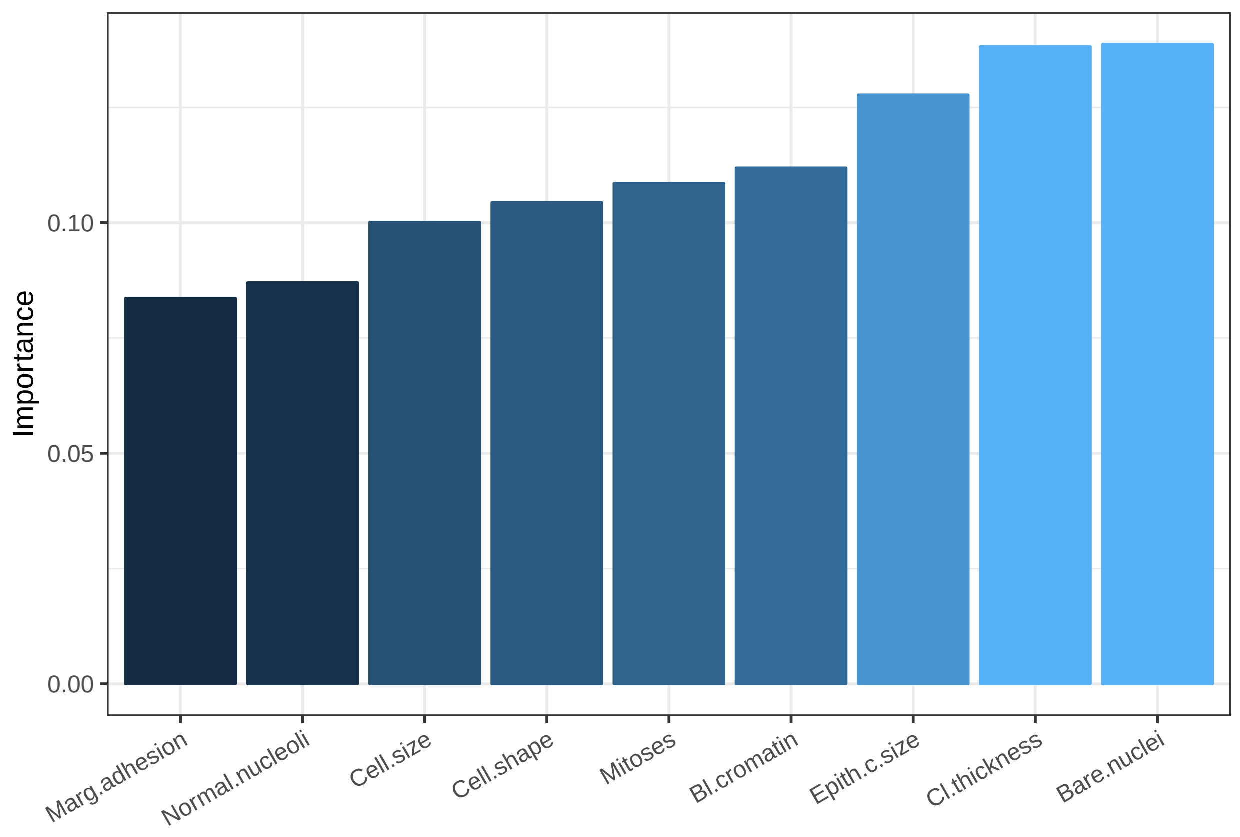
```
## # weights:  89
## initial  value 322.300354
## iter  10 value 78.473477
## iter  20 value 20.761423
## iter  30 value 11.718385
## iter  40 value 6.868799
## iter  50 value 4.461064
## iter  60 value 4.027394
## iter  70 value 3.895768
## iter  80 value 3.853760
## iter  90 value 3.831495
## iter 100 value 3.823310
## final  value 3.823310
## stopped after 100 iterations
```

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

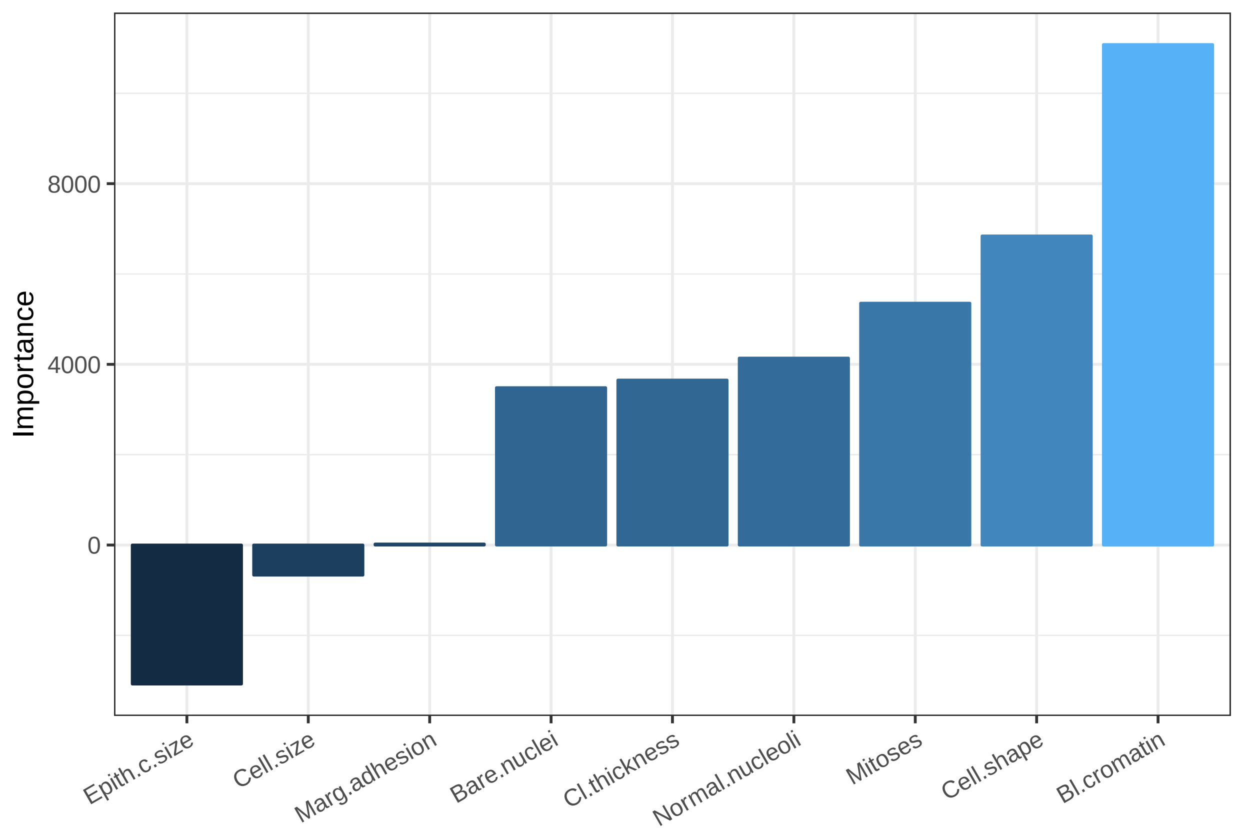
```
plotnet(model_nnet, cex_val =.8,max_sp=T,circle_cex=5,circle_col = 'red')
```



```
#Relative importance of input variables in neural networks using Garson's algorithm:
garson(model_nnet) +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```



```r
olden(model_nnet) +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```



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

Prediction

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

```r
#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:  204
##
##
##                    | Predict Diagnosis
## Actual Diagnosis |    benign | malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           benign |       127 |         6 |       133 |
##                  |     0.623 |     0.029 |           |
## -----------------|-----------|-----------|-----------|
##        malignant |         6 |        65 |        71 |
##                  |     0.029 |     0.319 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |       133 |        71 |       204 |
## -----------------|-----------|-----------|-----------|
##
##
```