Machine learning models for cancer predictive analysis

Natalia

28 May 2019

```r
data <- read.csv("C://Users//Natalia//Desktop//ITMO//R//R project//cancer data//breastcancer//cancer_da
colnames(data) <- c("id", "Class", "radius_mean", "texture_mean", "perimeter_mean", "area_mean", "smoot
                    "compactness_wors", "concavity_worst", "concave points_worst","symmetry_worst","fra
View(data)
```

Analyse the dataset and tidy it up.

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

```
##        id                Class         radius_mean      texture_mean
##  Min.   :     8670   Min.   :0.0000   Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869222   1st Qu.:0.0000   1st Qu.:11.697   1st Qu.:16.18
##  Median :   906157   Median :0.0000   Median :13.355   Median :18.86
##  Mean   : 30423820   Mean   :0.3715   Mean   :14.120   Mean   :19.31
##  3rd Qu.:  8825022   3rd Qu.:1.0000   3rd Qu.:15.780   3rd Qu.:21.80
##  Max.   :911320502   Max.   :1.0000   Max.   :28.110   Max.   :39.28
##  perimeter_mean     area_mean      smoothness_mean   compactness_mean
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 75.14   1st Qu.: 420.2   1st Qu.:0.08629   1st Qu.:0.06481
##  Median : 86.21   Median : 548.8   Median :0.09587   Median :0.09252
##  Mean   : 91.91   Mean   : 654.3   Mean   :0.09632   Mean   :0.10404
##  3rd Qu.:103.88   3rd Qu.: 782.6   3rd Qu.:0.10530   3rd Qu.:0.13040
##  Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##  concavity_mean    concave_points_mean symmetry_mean
##  Min.   :0.00000   Min.   :0.00000     Min.   :0.1060
##  1st Qu.:0.02954   1st Qu.:0.02031     1st Qu.:0.1619
##  Median :0.06140   Median :0.03345     Median :0.1792
##  Mean   :0.08843   Mean   :0.04875     Mean   :0.1811
##  3rd Qu.:0.12965   3rd Qu.:0.07373     3rd Qu.:0.1956
##  Max.   :0.42680   Max.   :0.20120     Max.   :0.3040
##  fractal_dimension_mean   radius_se        texture_se      perimeter_se
##  Min.   :0.04996        Min.   :0.1115   Min.   :0.3602   Min.   : 0.757
##  1st Qu.:0.05770        1st Qu.:0.2324   1st Qu.:0.8331   1st Qu.: 1.605
##  Median :0.06152        Median :0.3240   Median :1.1095   Median : 2.285
##  Mean   :0.06277        Mean   :0.4040   Mean   :1.2174   Mean   : 2.856
##  3rd Qu.:0.06612        3rd Qu.:0.4773   3rd Qu.:1.4743   3rd Qu.: 3.337
##  Max.   :0.09744        Max.   :2.8730   Max.   :4.8850   Max.   :21.980
##     area_se         smoothness_se       compactness_se      concavity_se
##  Min.   :  6.802   Min.   :0.001713   Min.   :0.002252   Min.   :0.00000
##  1st Qu.: 17.850   1st Qu.:0.005166   1st Qu.:0.013048   1st Qu.:0.01506
##  Median : 24.485   Median :0.006374   Median :0.020435   Median :0.02587
##  Mean   : 40.138   Mean   :0.007042   Mean   :0.025437   Mean   :0.03186
##  3rd Qu.: 45.017   3rd Qu.:0.008151   3rd Qu.:0.032218   3rd Qu.:0.04176
##  Max.   :542.200   Max.   :0.031130   Max.   :0.135400   Max.   :0.39600
##  concave_points_se   symmetry_se        fractal_dimension_se
##  Min.   :0.000000   Min.   :0.007882   Min.   :0.0008948
##  1st Qu.:0.007634   1st Qu.:0.015128   1st Qu.:0.0022445
##  Median :0.010920   Median :0.018725   Median :0.0031615
##  Mean   :0.011789   Mean   :0.020526   Mean   :0.0037907
##  3rd Qu.:0.014710   3rd Qu.:0.023398   3rd Qu.:0.0045258
##  Max.   :0.052790   Max.   :0.078950   Max.   :0.0298400
##   radius_worst   texture_worst   perimeter_worst    area_worst
##  Min.   : 7.93   Min.   :12.02   Min.   : 50.41   Min.   : 185.2
##  1st Qu.:13.01   1st Qu.:21.09   1st Qu.: 84.10   1st Qu.: 515.0
##  Median :14.96   Median :25.43   Median : 97.66   Median : 685.5
##  Mean   :16.25   Mean   :25.69   Mean   :107.13   Mean   : 878.6
##  3rd Qu.:18.77   3rd Qu.:29.76   3rd Qu.:125.17   3rd Qu.:1073.5
##  Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0
##  smoothness_worst  compactness_wors  concavity_worst  concave points_worst
##  Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.11660   1st Qu.:0.14690   1st Qu.:0.1145   1st Qu.:0.06473
##  Median :0.13130   Median :0.21185   Median :0.2266   Median :0.09984
##  Mean   :0.13232   Mean   :0.25354   Mean   :0.2714   Mean   :0.11434
##  3rd Qu.:0.14600   3rd Qu.:0.33760   3rd Qu.:0.3814   3rd Qu.:0.16132
##  Max.   :0.22260   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100
##  symmetry_worst   fractal_dimension_worst
##  Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.2504   1st Qu.:0.07141
##  Median :0.2821   Median :0.08002
##  Mean   :0.2898   Mean   :0.08388
##  3rd Qu.:0.3177   3rd Qu.:0.09206
##  Max.   :0.6638   Max.   :0.20750
```

```r
str(data)
```

```
## 'data.frame':    568 obs. of  32 variables:
##  $ id                     : int  842517 84300903 84348301 84358402 843786 844359 84458202 844981 845
##  $ Class                  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ radius_mean            : num  20.6 19.7 11.4 20.3 12.4 ...
##  $ texture_mean           : num  17.8 21.2 20.4 14.3 15.7 ...
##  $ perimeter_mean         : num  132.9 130 77.6 135.1 82.6 ...
##  $ area_mean              : num  1326 1203 386 1297 477 ...
##  $ smoothness_mean        : num  0.0847 0.1096 0.1425 0.1003 0.1278 ...
##  $ compactness_mean       : num  0.0786 0.1599 0.2839 0.1328 0.17 ...
##  $ concavity_mean         : num  0.0869 0.1974 0.2414 0.198 0.1578 ...
##  $ concave_points_mean    : num  0.0702 0.1279 0.1052 0.1043 0.0809 ...
##  $ symmetry_mean          : num  0.181 0.207 0.26 0.181 0.209 ...
##  $ fractal_dimension_mean : num  0.0567 0.06 0.0974 0.0588 0.0761 ...
##  $ radius_se              : num  0.543 0.746 0.496 0.757 0.335 ...
##  $ texture_se             : num  0.734 0.787 1.156 0.781 0.89 ...
##  $ perimeter_se           : num  3.4 4.58 3.44 5.44 2.22 ...
##  $ area_se                : num  74.1 94 27.2 94.4 27.2 ...
##  $ smoothness_se          : num  0.00522 0.00615 0.00911 0.01149 0.00751 ...
##  $ compactness_se         : num  0.0131 0.0401 0.0746 0.0246 0.0335 ...
##  $ concavity_se           : num  0.0186 0.0383 0.0566 0.0569 0.0367 ...
##  $ concave_points_se      : num  0.0134 0.0206 0.0187 0.0188 0.0114 ...
##  $ symmetry_se            : num  0.0139 0.0225 0.0596 0.0176 0.0216 ...
##  $ fractal_dimension_se   : num  0.00353 0.00457 0.00921 0.00511 0.00508 ...
##  $ radius_worst           : num  25 23.6 14.9 22.5 15.5 ...
##  $ texture_worst          : num  23.4 25.5 26.5 16.7 23.8 ...
##  $ perimeter_worst        : num  158.8 152.5 98.9 152.2 103.4 ...
##  $ area_worst             : num  1956 1709 568 1575 742 ...
##  $ smoothness_worst       : num  0.124 0.144 0.21 0.137 0.179 ...
##  $ compactness_wors       : num  0.187 0.424 0.866 0.205 0.525 ...
##  $ concavity_worst        : num  0.242 0.45 0.687 0.4 0.535 ...
##  $ concave points_worst   : num  0.186 0.243 0.258 0.163 0.174 ...
##  $ symmetry_worst         : num  0.275 0.361 0.664 0.236 0.399 ...
##  $ fractal_dimension_worst: num  0.089 0.0876 0.173 0.0768 0.1244 ...
```

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

```
##       Class radius_mean texture_mean perimeter_mean area_mean
## 1 malignant       20.57        17.77         132.90    1326.0
## 2 malignant       19.69        21.25         130.00    1203.0
## 3 malignant       11.42        20.38          77.58     386.1
## 4 malignant       20.29        14.34         135.10    1297.0
## 5 malignant       12.45        15.70          82.57     477.1
## 6 malignant       18.25        19.98         119.60    1040.0
##   smoothness_mean compactness_mean concavity_mean concave_points_mean
## 1         0.08474          0.07864         0.0869             0.07017
## 2         0.10960          0.15990         0.1974             0.12790
## 3         0.14250          0.28390         0.2414             0.10520
## 4         0.10030          0.13280         0.1980             0.10430
## 5         0.12780          0.17000         0.1578             0.08089
## 6         0.09463          0.10900         0.1127             0.07400
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.1812                0.05667    0.5435     0.7339        3.398
## 2        0.2069                0.05999    0.7456     0.7869        4.585
## 3        0.2597                0.09744    0.4956     1.1560        3.445
## 4        0.1809                0.05883    0.7572     0.7813        5.438
## 5        0.2087                0.07613    0.3345     0.8902        2.217
## 6        0.1794                0.05742    0.4467     0.7732        3.180
##   area_se smoothness_se compactness_se concavity_se concave_points_se
## 1   74.08      0.005225        0.01308      0.01860           0.01340
## 2   94.03      0.006150        0.04006      0.03832           0.02058
## 3   27.23      0.009110        0.07458      0.05661           0.01867
## 4   94.44      0.011490        0.02461      0.05688           0.01885
## 5   27.19      0.007510        0.03345      0.03672           0.01137
## 6   53.91      0.004314        0.01382      0.02254           0.01039
##   symmetry_se fractal_dimension_se radius_worst texture_worst
## 1     0.01389             0.003532        24.99         23.41
## 2     0.02250             0.004571        23.57         25.53
## 3     0.05963             0.009208        14.91         26.50
## 4     0.01756             0.005115        22.54         16.67
## 5     0.02165             0.005082        15.47         23.75
## 6     0.01369             0.002179        22.88         27.66
##   perimeter_worst area_worst smoothness_worst compactness_wors
## 1          158.80     1956.0           0.1238           0.1866
## 2          152.50     1709.0           0.1444           0.4245
## 3           98.87      567.7           0.2098           0.8663
## 4          152.20     1575.0           0.1374           0.2050
## 5          103.40      741.6           0.1791           0.5249
## 6          153.20     1606.0           0.1442           0.2576
##   concavity_worst concave points_worst symmetry_worst
## 1          0.2416               0.1860         0.2750
## 2          0.4504               0.2430         0.3613
## 3          0.6869               0.2575         0.6638
## 4          0.4000               0.1625         0.2364
## 5          0.5355               0.1741         0.3985
## 6          0.3784               0.1932         0.3063
##   fractal_dimension_worst        Cl
## 1                 0.08902 malignant
## 2                 0.08758 malignant
## 3                 0.17300 malignant
## 4                 0.07678 malignant
## 5                 0.12440 malignant
## 6                 0.08368 malignant
```

```
dim(data)
```

```
## [1] 568  32
```

```
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)))
```

```
##                   Class             radius_mean            texture_mean
##                       0                       0                       0
##          perimeter_mean               area_mean         smoothness_mean
##                       0                       0                       0
##        compactness_mean          concavity_mean     concave_points_mean
##                       0                       0                       0
##           symmetry_mean  fractal_dimension_mean               radius_se
##                       0                       0                       0
##              texture_se            perimeter_se                 area_se
##                       0                       0                       0
##           smoothness_se          compactness_se            concavity_se
##                       0                       0                       0
##       concave_points_se             symmetry_se    fractal_dimension_se
##                       0                       0                       0
##            radius_worst           texture_worst         perimeter_worst
##                       0                       0                       0
##              area_worst        smoothness_worst        compactness_wors
##                       0                       0                       0
##         concavity_worst    concave points_worst          symmetry_worst
##                       0                       0                       0
## fractal_dimension_worst                      Cl
##                       0                       0
```

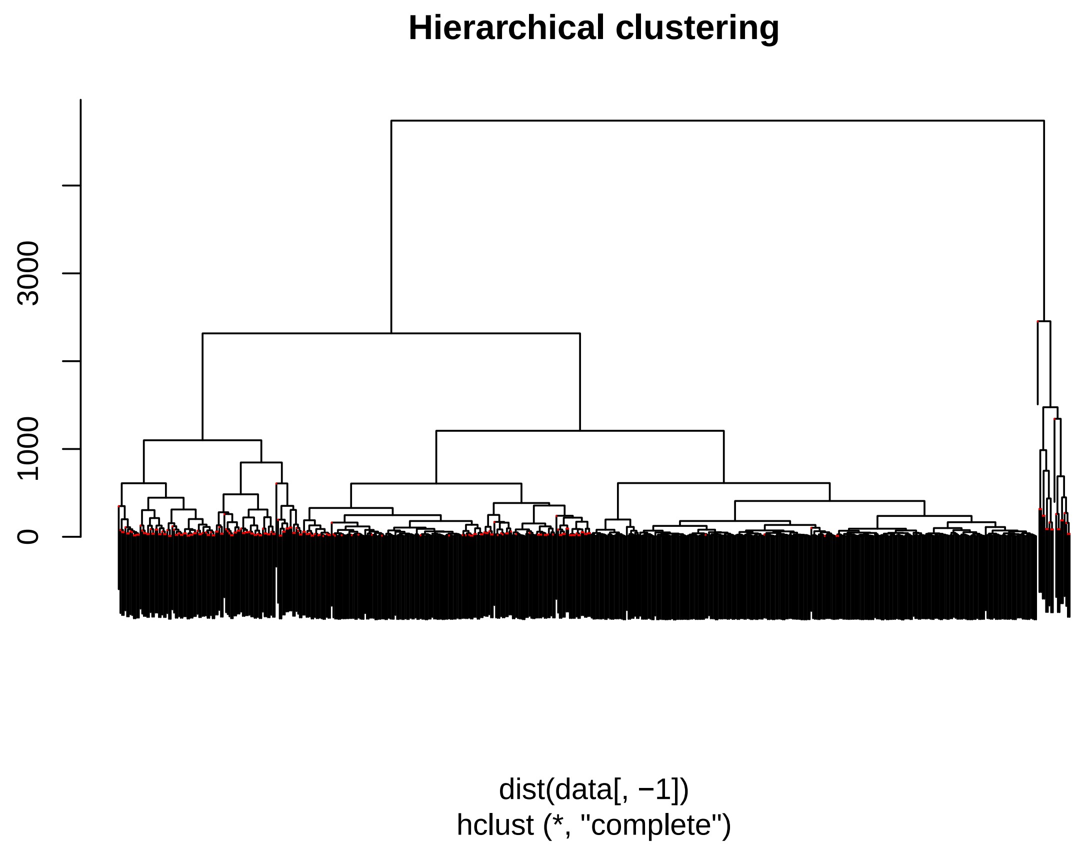
```
# Data type "Class" as factor:

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

```
##       Class radius_mean texture_mean perimeter_mean area_mean
## 1 malignant       20.57        17.77         132.90    1326.0
## 2 malignant       19.69        21.25         130.00    1203.0
## 3 malignant       11.42        20.38          77.58     386.1
## 4 malignant       20.29        14.34         135.10    1297.0
## 5 malignant       12.45        15.70          82.57     477.1
## 6 malignant       18.25        19.98         119.60    1040.0
##   smoothness_mean compactness_mean concavity_mean concave_points_mean
## 1         0.08474          0.07864         0.0869             0.07017
## 2         0.10960          0.15990         0.1974             0.12790
## 3         0.14250          0.28390         0.2414             0.10520
## 4         0.10030          0.13280         0.1980             0.10430
## 5         0.12780          0.17000         0.1578             0.08089
## 6         0.09463          0.10900         0.1127             0.07400
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.1812                0.05667    0.5435     0.7339        3.398
## 2        0.2069                0.05999    0.7456     0.7869        4.585
## 3        0.2597                0.09744    0.4956     1.1560        3.445
## 4        0.1809                0.05883    0.7572     0.7813        5.438
## 5        0.2087                0.07613    0.3345     0.8902        2.217
## 6        0.1794                0.05742    0.4467     0.7732        3.180
##   area_se smoothness_se compactness_se concavity_se concave_points_se
## 1   74.08      0.005225        0.01308      0.01860           0.01340
## 2   94.03      0.006150        0.04006      0.03832           0.02058
## 3   27.23      0.009110        0.07458      0.05661           0.01867
## 4   94.44      0.011490        0.02461      0.05688           0.01885
## 5   27.19      0.007510        0.03345      0.03672           0.01137
## 6   53.91      0.004314        0.01382      0.02254           0.01039
##   symmetry_se fractal_dimension_se radius_worst texture_worst
## 1     0.01389             0.003532        24.99         23.41
## 2     0.02250             0.004571        23.57         25.53
## 3     0.05963             0.009208        14.91         26.50
## 4     0.01756             0.005115        22.54         16.67
## 5     0.02165             0.005082        15.47         23.75
## 6     0.01369             0.002179        22.88         27.66
##   perimeter_worst area_worst smoothness_worst compactness_wors
## 1          158.80     1956.0           0.1238           0.1866
## 2          152.50     1709.0           0.1444           0.4245
## 3           98.87      567.7           0.2098           0.8663
## 4          152.20     1575.0           0.1374           0.2050
## 5          103.40      741.6           0.1791           0.5249
## 6          153.20     1606.0           0.1442           0.2576
##   concavity_worst concave points_worst symmetry_worst
## 1          0.2416               0.1860         0.2750
## 2          0.4504               0.2430         0.3613
## 3          0.6869               0.2575         0.6638
## 4          0.4000               0.1625         0.2364
## 5          0.5355               0.1741        0.3985
## 6          0.3784               0.1932        0.3063
##   fractal_dimension_worst
## 1                 0.08902
## 2                 0.08758
## 3                 0.17300
## 4                 0.07678
## 5                 0.12440
## 6                 0.08368
```

DATA EXPLORATION

Hierarchical clustering

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



Not very obvious where are different clusters. It seems that "red" malignant cluster intersects with "benign" cluster.

K-means clustering

```r
fit <- kmeans(data[,c(2:31)], 2)
names(fit)
```

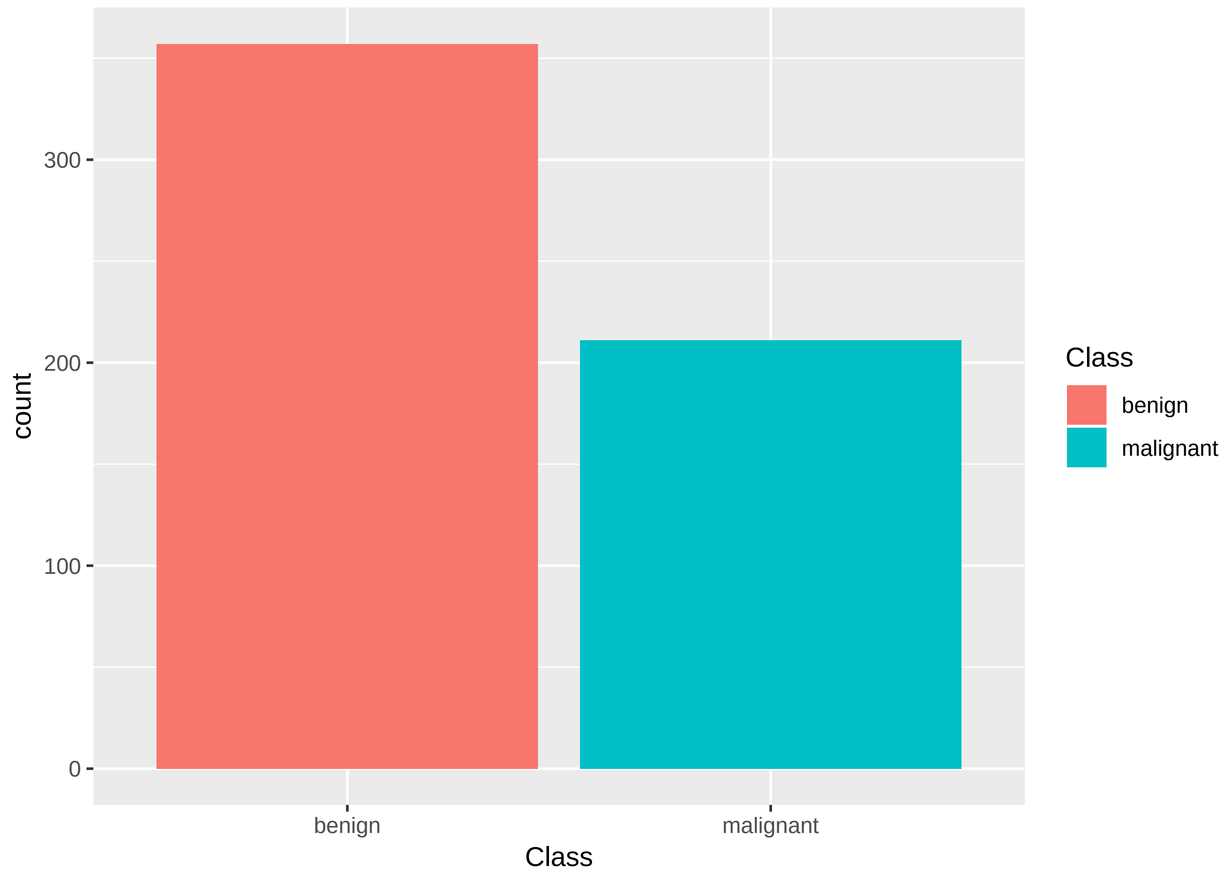
```
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
#k-means did a fairly good job
table(data.frame(fit$cluster,data[,1]))
```

```
##            data...1.
## fit.cluster benign malignant
##           1      1       129
##           2    356        82
```

Response variable for classification.

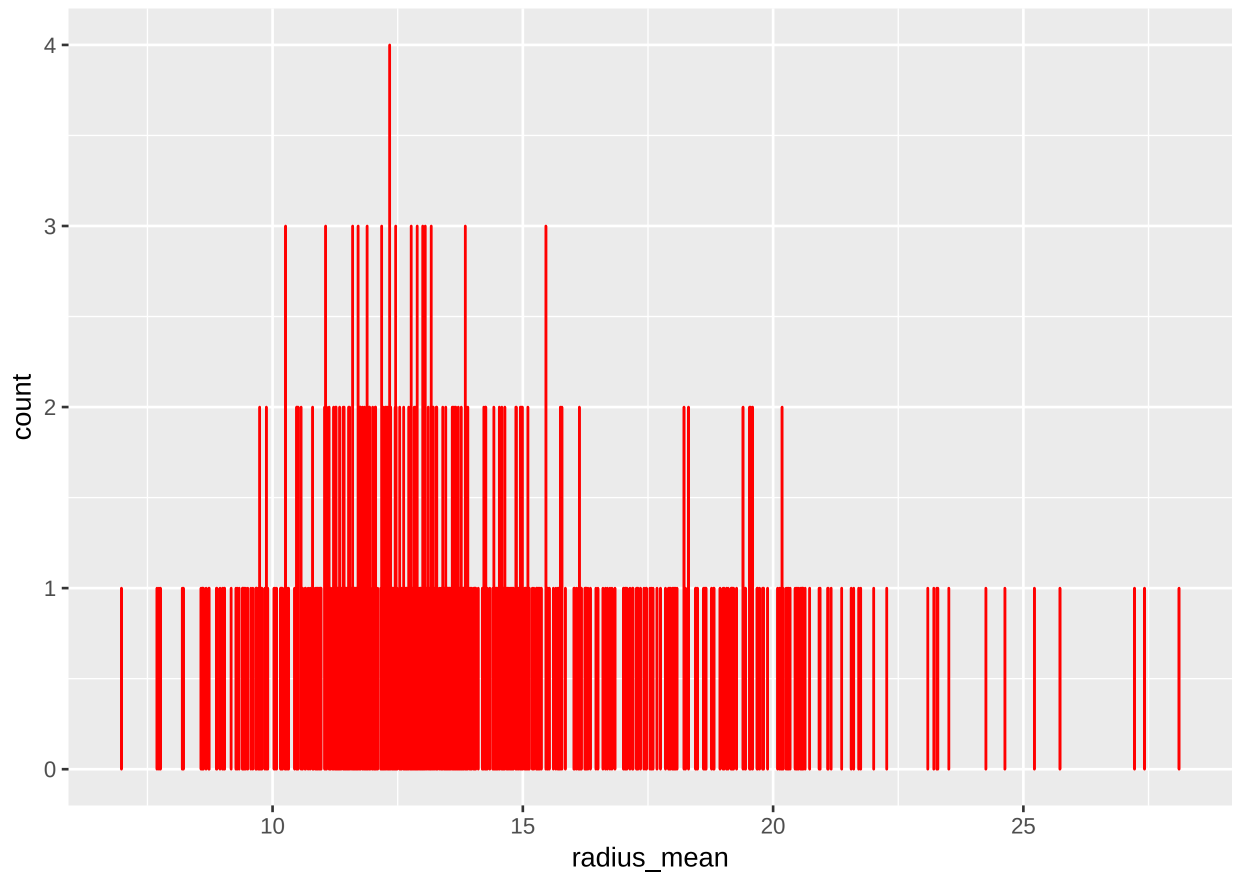
```r
library(ggplot2)

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



Response variable for regression.

```r
ggplot(data, aes(x = radius_mean)) +
  geom_histogram(stat = "count", color = "red")
```

```
## 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':
##
```

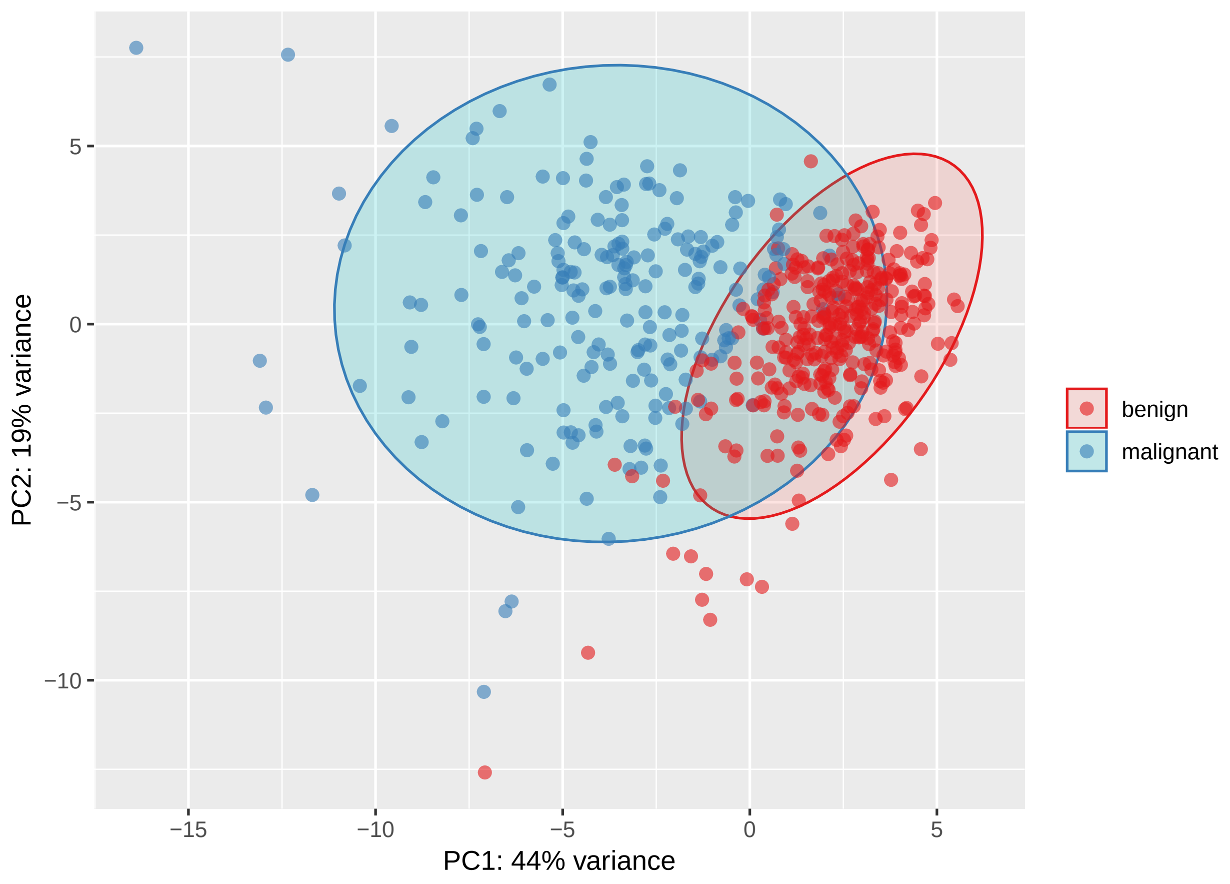
```
##     windows

## 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)
```

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

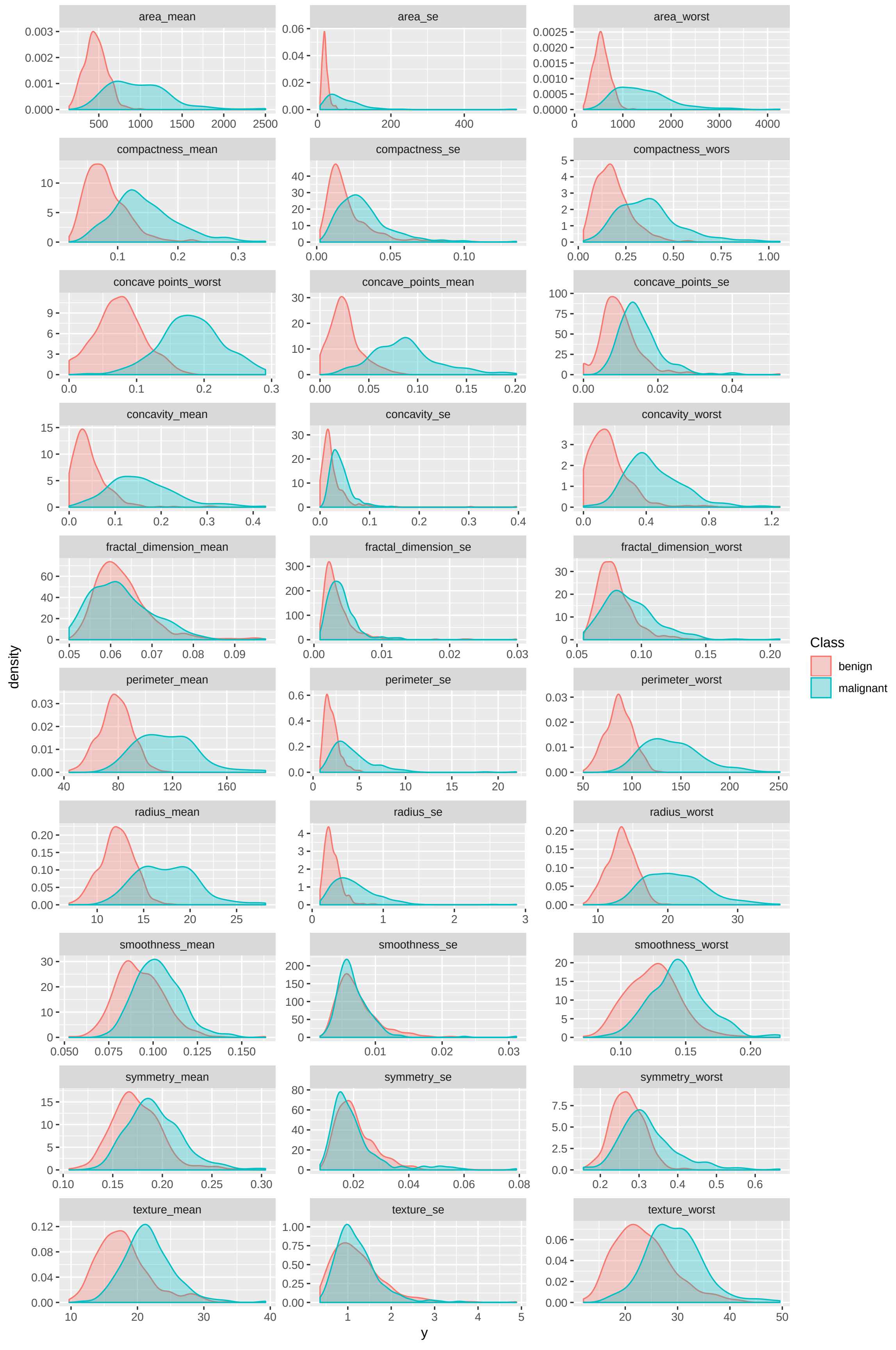
```
# perform pca and extract scores:

pcaOutput <- pca(t(data[,2:31]), 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

```r
library(tidyr)


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


area_mean
area_se
area_worst
compactness_mean
compactness_se
compactness_wors
concave points_worst
concave_points_mean
concave_points_se
concavity_mean
concavity_se
concavity_worst
fractal_dimension_mean
fractal_dimension_se
fractal_dimension_worst
perimeter_mean
perimeter_se
perimeter_worst
radius_mean
radius_se
radius_worst
smoothness_mean
smoothness_se
smoothness_worst
symmetry_mean
symmetry_se
symmetry_worst
texture_mean
texture_se
texture_worst
density
y
Class
benign
malignant

MACHINE LEARNING PACKAGES FOR R

caret

```r
# 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
```

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

library(caret)
```

```
## Loading required package: lattice
```

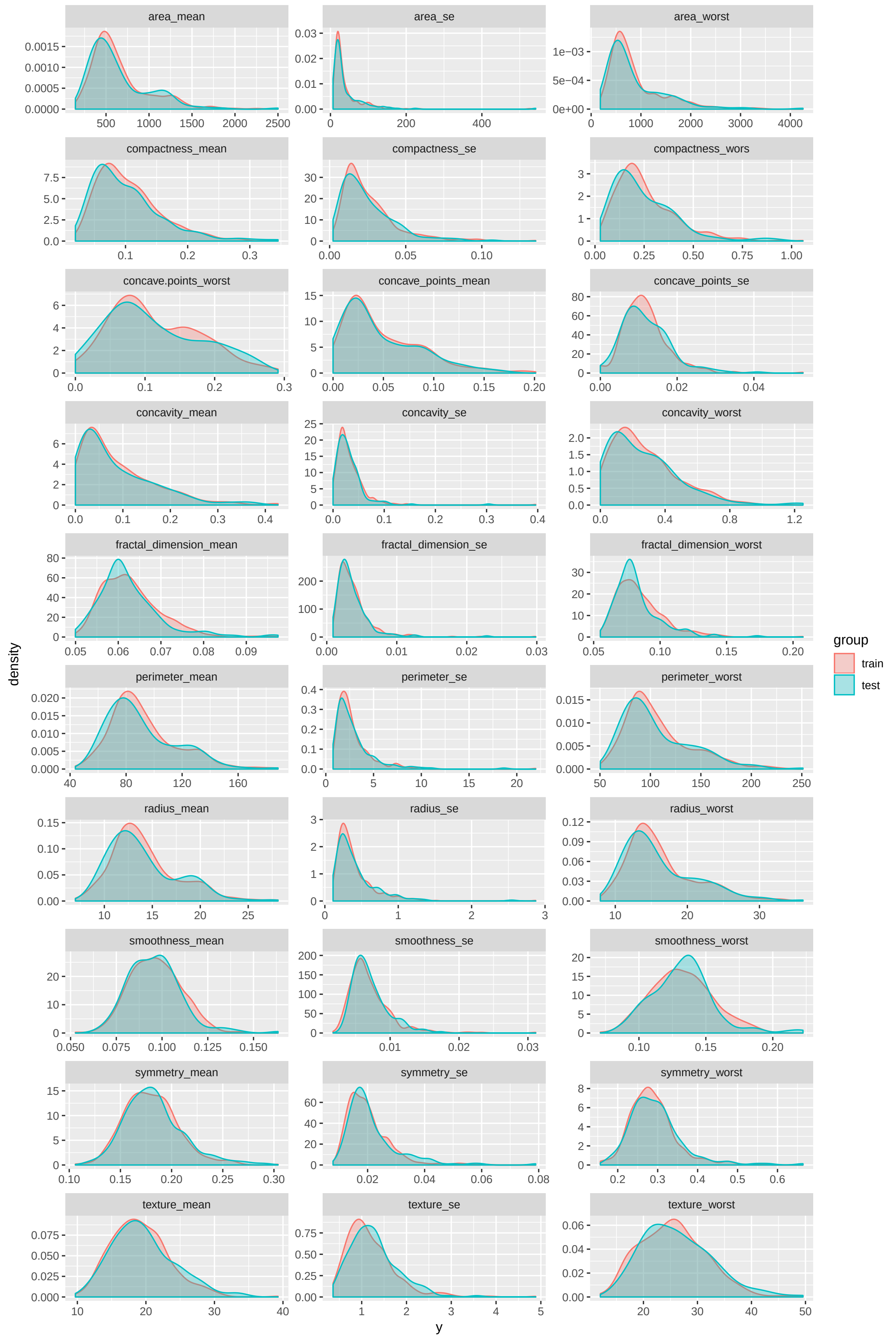
```
##
## Attaching package: 'caret'
```

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

Training, validation and test data

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

```r
library(dplyr)

rbind(data.frame(group = "train", train_data),
      data.frame(group = "test", test_data)) %>%
  gather(x, y, radius_mean:fractal_dimension_worst) %>%
  ggplot(aes(x = y, color = group, fill = group)) +
    geom_density(alpha = 0.3) +
    facet_wrap( ~ x, scales = "free", ncol = 3)
```


area_mean
area_se
area_worst
compactness_mean
compactness_se
compactness_wors
concave.points_worst
concave_points_mean
concave_points_se
concavity_mean
concavity_se
concavity_worst
fractal_dimension_mean
fractal_dimension_se
fractal_dimension_worst
perimeter_mean
perimeter_se
perimeter_worst
radius_mean
radius_se
radius_worst
smoothness_mean
smoothness_se
smoothness_worst
symmetry_mean
symmetry_se
symmetry_worst
texture_mean
texture_se
texture_worst
density
y
group
train
test

REGRESSION

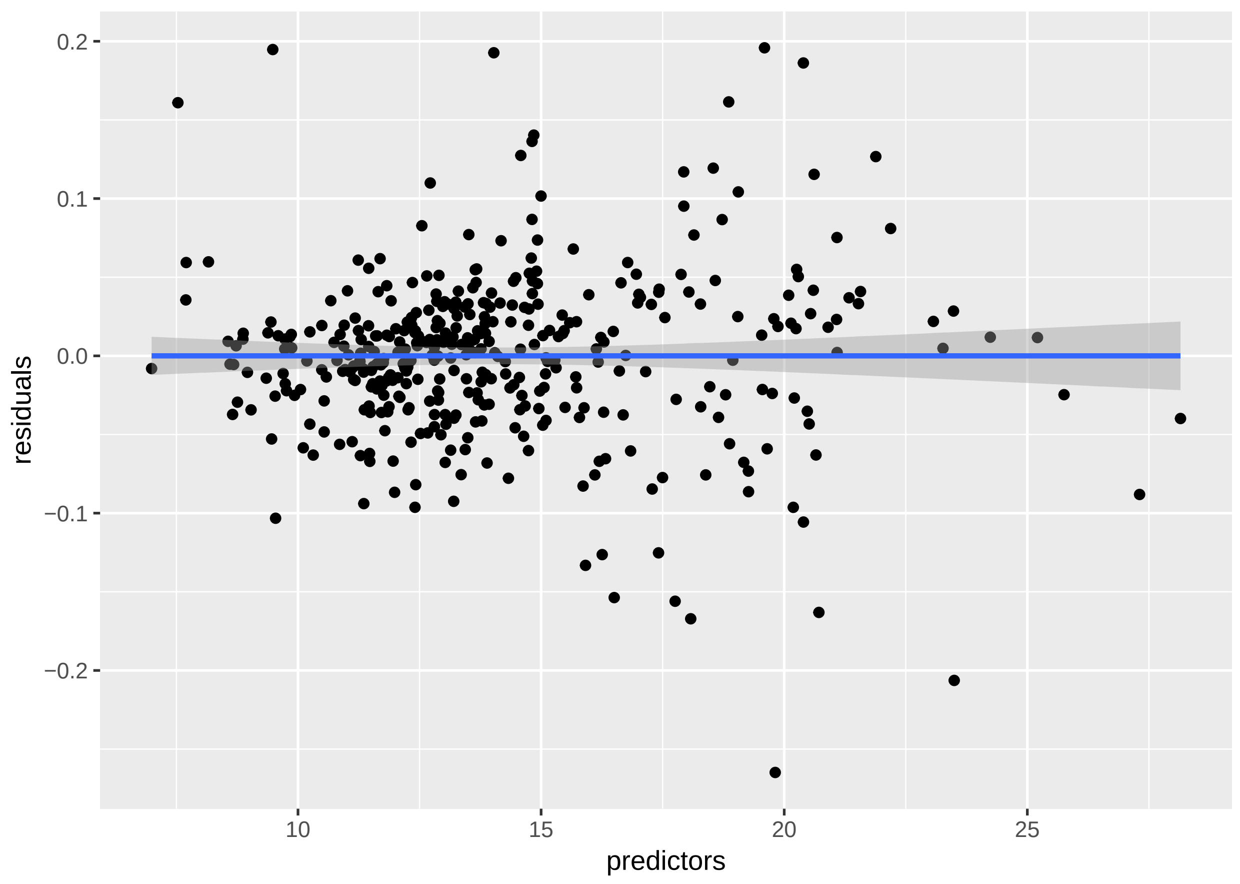
```
set.seed(42)
model_glm <- caret::train(radius_mean ~ .,
                          data = train_data,
                          method = "glm",
                          preProcess = c("scale", "center"),
                          trControl = trainControl(method = "repeatedcv",
                  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

```
model_glm
```

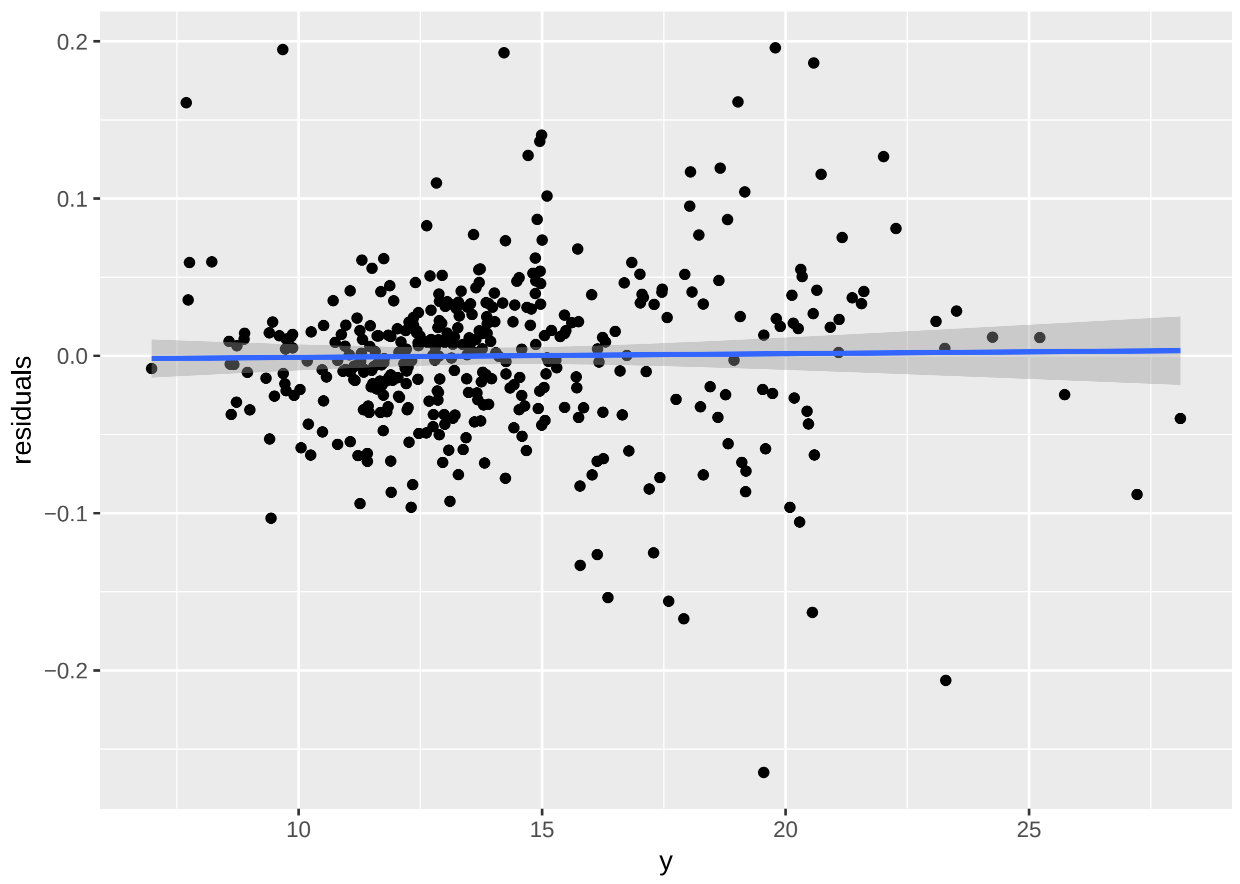
```
## Generalized Linear Model
##
## 398 samples
##  30 predictor
##
## Pre-processing: scaled (30), centered (30)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 358, 358, 359, 358, 358, 358, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.06443886  0.9996674  0.04360507
```

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

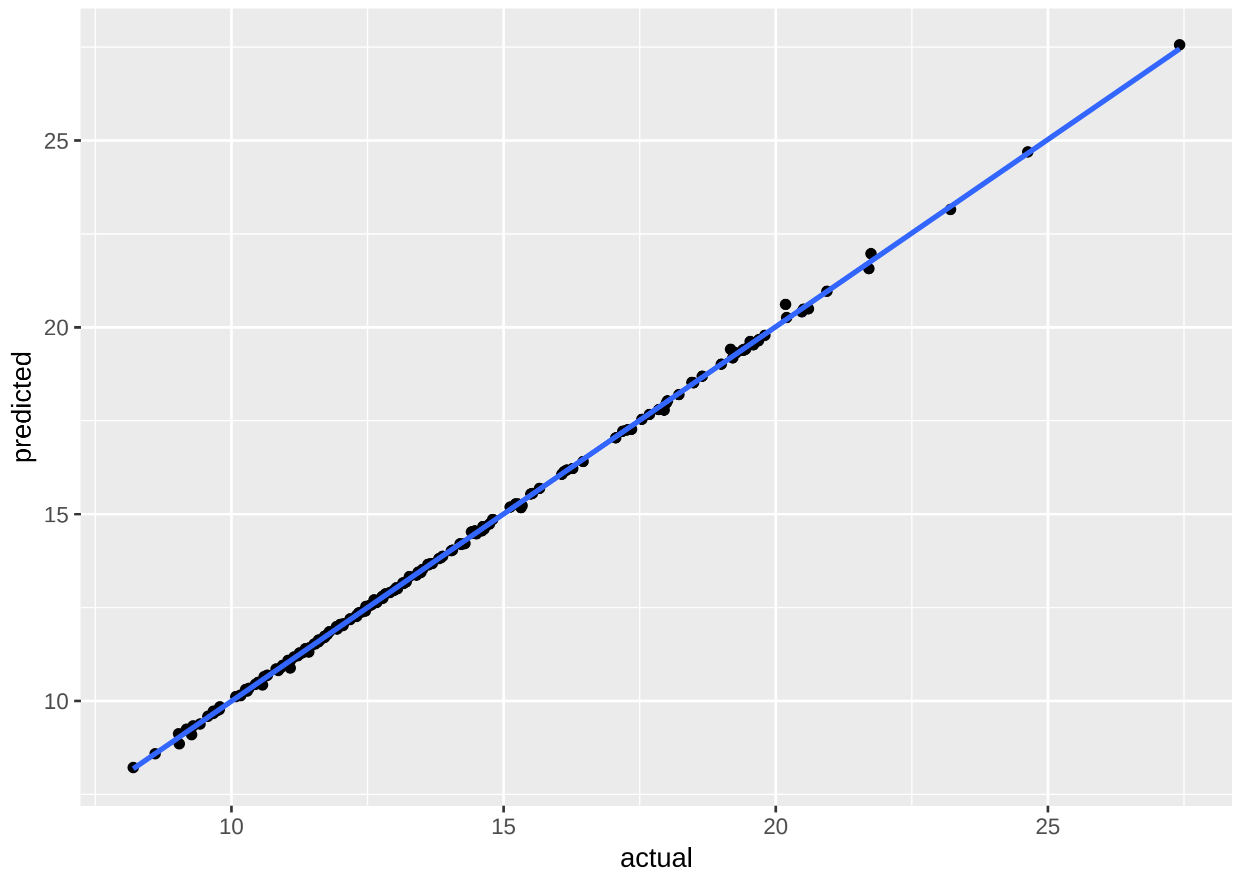
```
# model_glm$finalModel$linear.predictors == model_glm$finalModel$fitted.values
data.frame(residuals = resid(model_glm),
           predictors = model_glm$finalModel$linear.predictors) %>%
  ggplot(aes(x = predictors, y = residuals)) +
    geom_jitter() +
    geom_smooth(method = "lm")
```



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



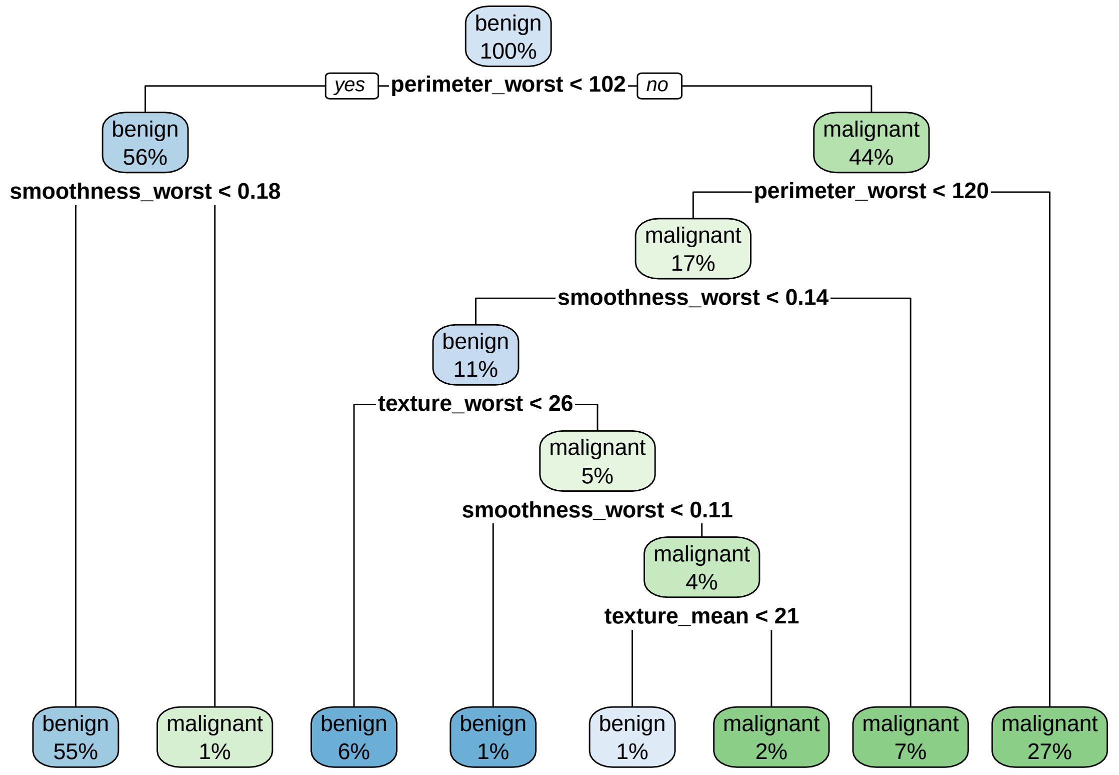
```
data.frame(actual = test_data$radius_mean,
           predicted = predictions) %>%
  ggplot(aes(x = actual, y = predicted)) +
    geom_jitter() +
    geom_smooth(method = "lm")
```



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
```

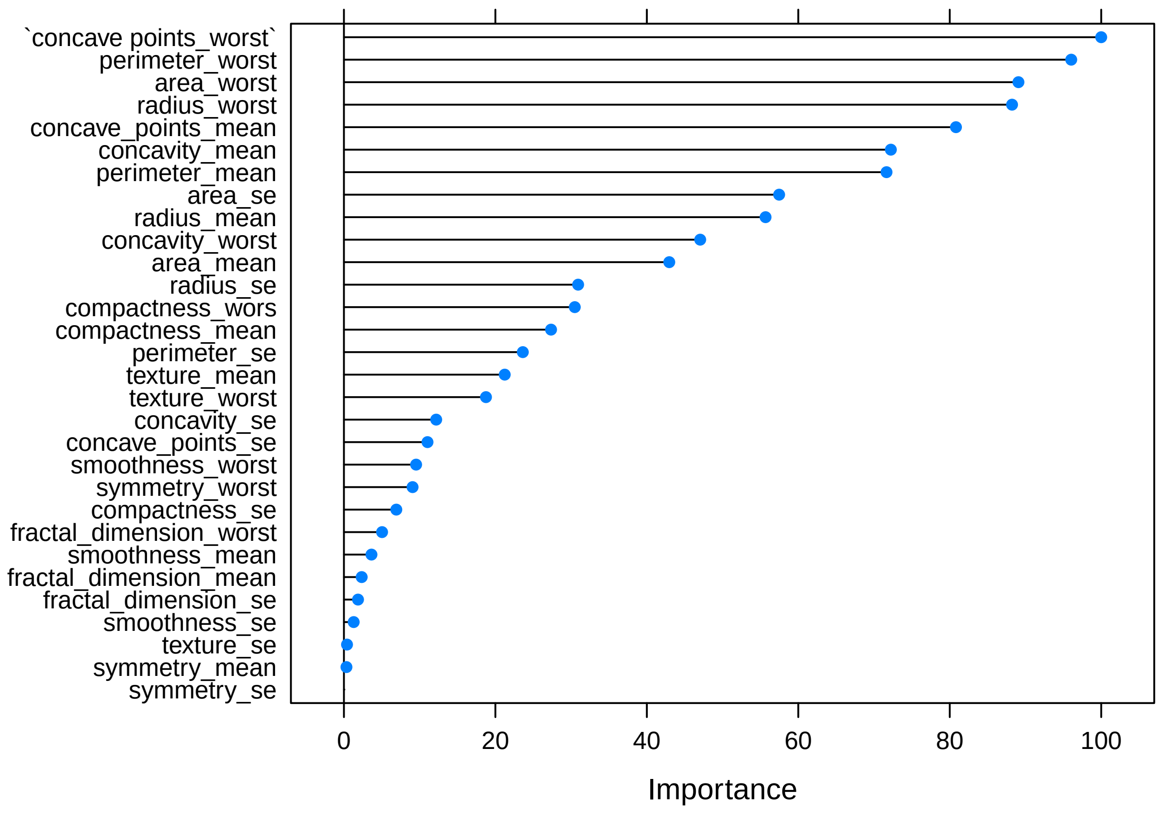
```
##           benign malignant class.error
## benign       243         7  0.02800000
## malignant      8       140  0.05405405
```

Feature Importance

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

```
##  `concave points_worst`         perimeter_worst              area_worst
##               15.841712               15.262762               14.242963
##            radius_worst     concave_points_mean          concavity_mean
##               14.119926               13.036611               11.779089
##          perimeter_mean                 area_se             radius_mean
##               11.695456                9.621609                9.359654
##         concavity_worst               area_mean               radius_se
##                8.097168                7.499848                5.739611
##        compactness_wors        compactness_mean            perimeter_se
##                5.675184                5.216820                4.671186
##            texture_mean           texture_worst            concavity_se
##                4.322024                3.959726                2.999013
##       concave_points_se        smoothness_worst          symmetry_worst
##                2.830949                2.610734                2.542498
##          compactness_se fractal_dimension_worst         smoothness_mean
##                2.228469                1.953428                1.748571
##  fractal_dimension_mean    fractal_dimension_se           smoothness_se
##                1.559781                1.488882                1.405341
##              texture_se           symmetry_mean             symmetry_se
##                1.276355                1.265452                1.216515
```

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

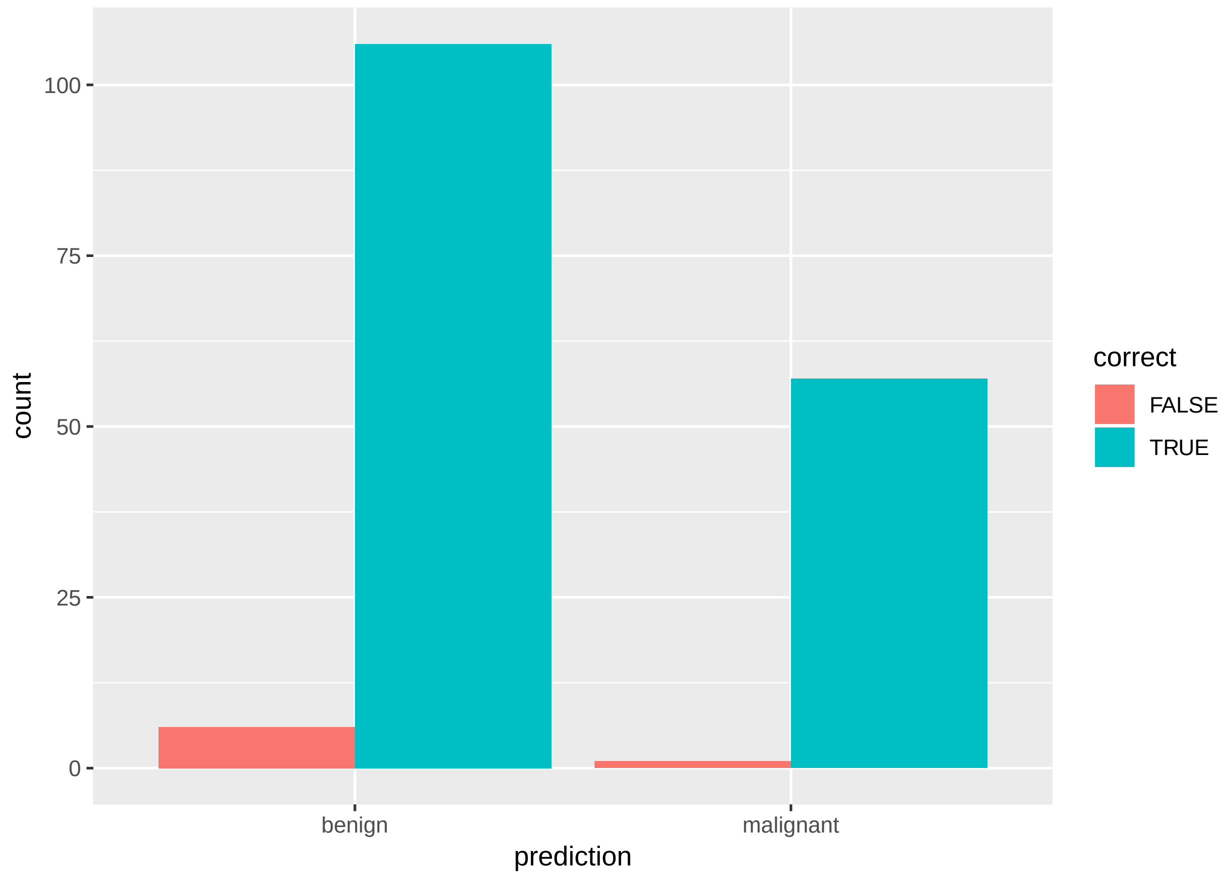


Predicting test data

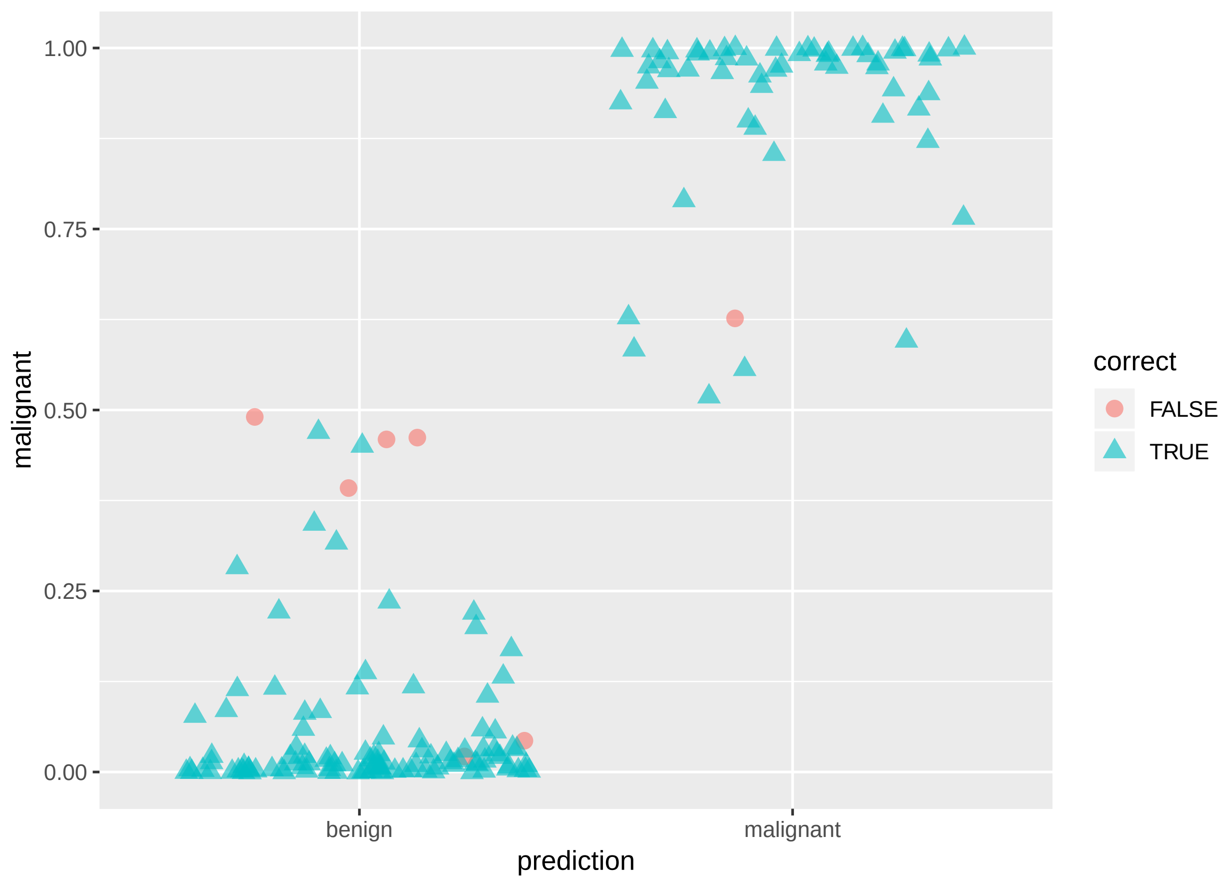
```
confusionMatrix(predict(model_rf, test_data), test_data$Class)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       106         6
##   malignant      1        57
##
##                Accuracy : 0.9588
##                  95% CI : (0.917, 0.9833)
##     No Information Rate : 0.6294
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9103
##
##  Mcnemar's Test P-Value : 0.1306
##
##             Sensitivity : 0.9907
##             Specificity : 0.9048
##          Pos Pred Value : 0.9464
##          Neg Pred Value : 0.9828
##               Prevalence : 0.6294
##           Detection Rate : 0.6235
##     Detection Prevalence : 0.6588
##        Balanced Accuracy : 0.9477
##
##         '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)
```

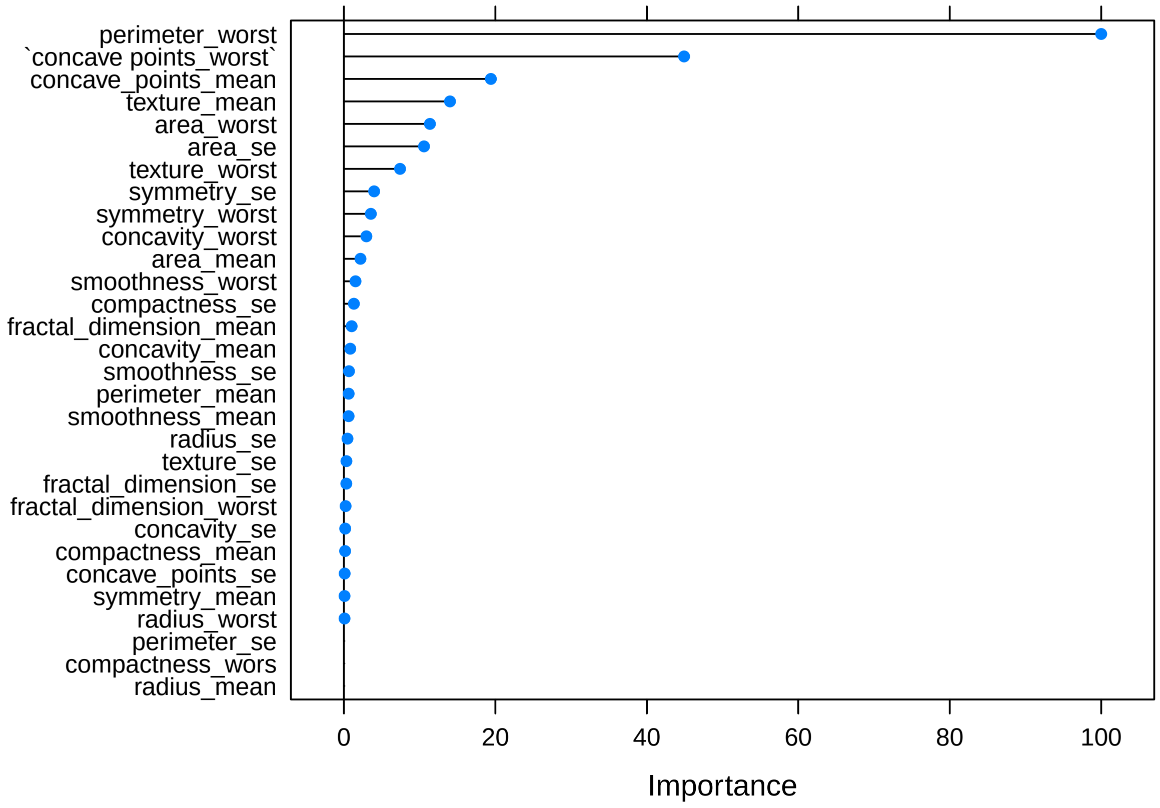
```
##
## Attaching package: 'xgboost'

## 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)
```

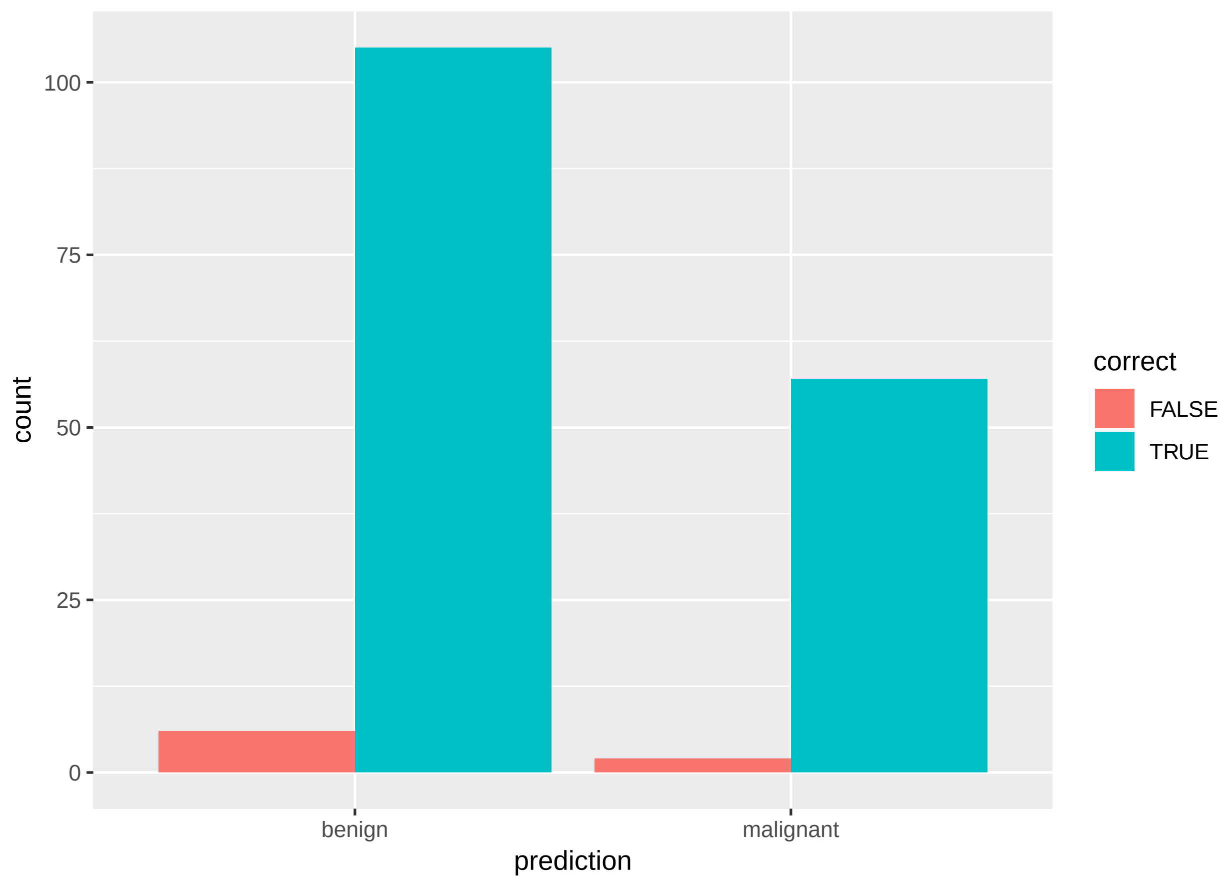


Predicting test data

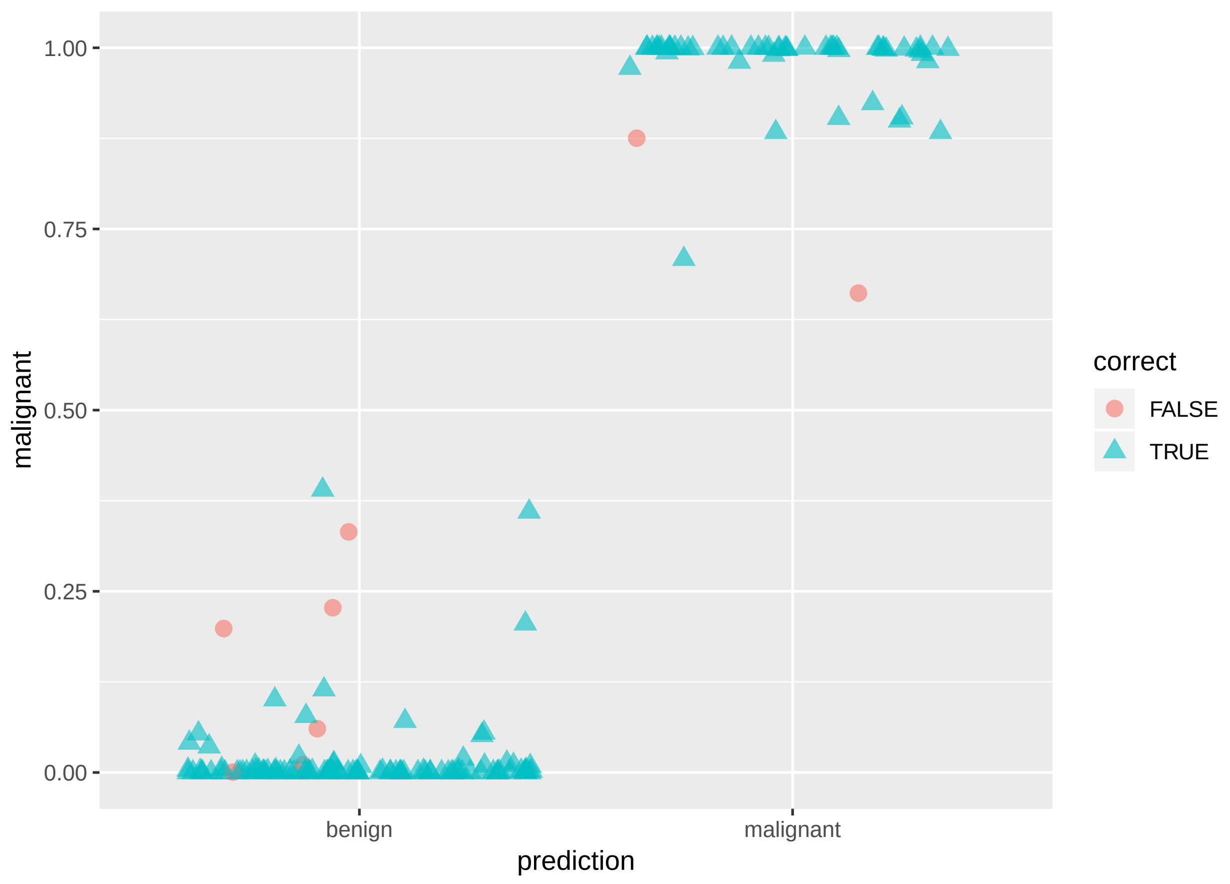
```
confusionMatrix(predict(model_xgb, test_data), test_data$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  benign malignant
##   benign       105         6
##   malignant      2        57
##
##                Accuracy : 0.9529
##                  95% CI : (0.9094, 0.9795)
##     No Information Rate : 0.6294
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8978
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.9813
##             Specificity : 0.9048
##          Pos Pred Value : 0.9459
##          Neg Pred Value : 0.9661
##              Prevalence : 0.6294
##          Detection Rate : 0.6176
##    Detection Prevalence : 0.6529
##       Balanced Accuracy : 0.9430
##
##        '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")
```



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



FEATURE SELECTION

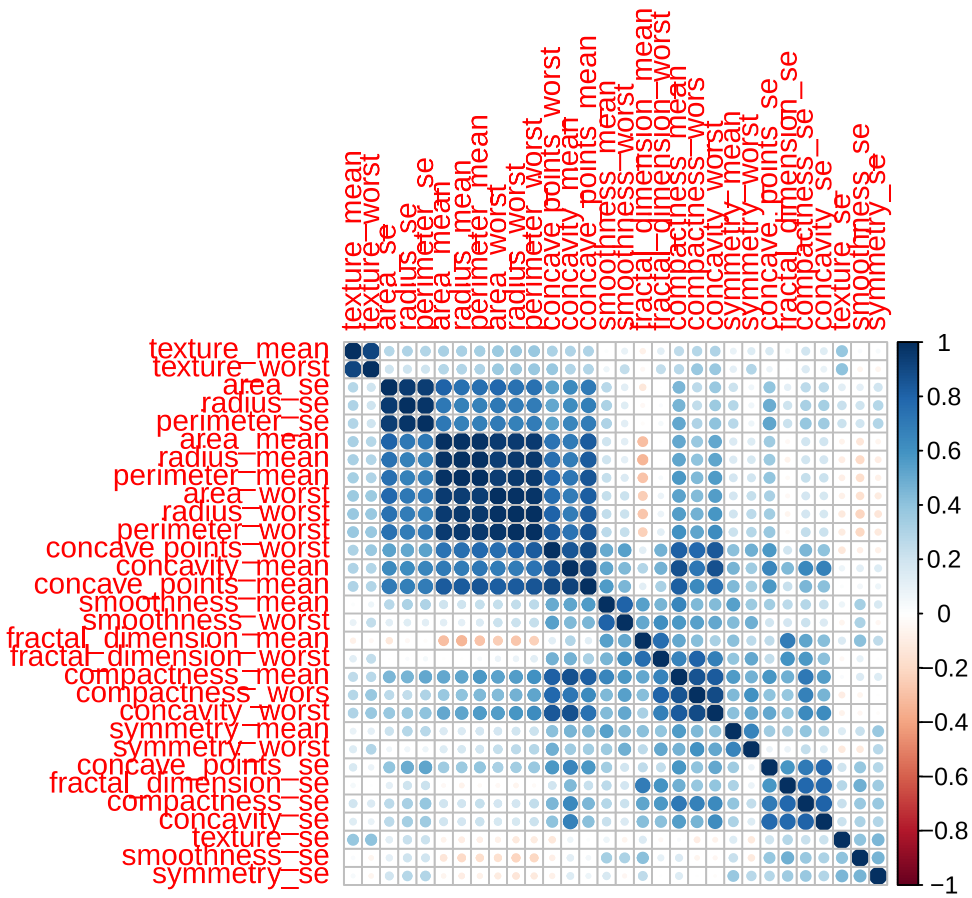
Performing feature selection on the whole dataset would lead to prediction bias, we therefore need to run the whole modeling process on the training data alone!

Correlation

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
# calculate correlation matrix
corMatMy <- cor(train_data[,2:31])
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 7  and column  8 with corr  0.921
##   Means:  0.564 vs 0.384 so flagging column 7
## Compare row 8  and column  28 with corr  0.905
##   Means:  0.534 vs 0.372 so flagging column 8
## Compare row 28  and column  6 with corr  0.819
##   Means:  0.514 vs 0.36 so flagging column 28
## Compare row 6  and column  27 with corr  0.831
##   Means:  0.502 vs 0.35 so flagging column 6
## Compare row 23  and column  21 with corr  0.994
##   Means:  0.462 vs 0.338 so flagging column 23
## Compare row 27  and column  26 with corr  0.893
##   Means:  0.443 vs 0.329 so flagging column 27
## Compare row 21  and column  3 with corr  0.967
##   Means:  0.422 vs 0.319 so flagging column 21
## Compare row 3  and column  24 with corr  0.943
##   Means:  0.383 vs 0.311 so flagging column 3
## Compare row 24  and column  1 with corr  0.943
##   Means:  0.359 vs 0.306 so flagging column 24
## Compare row 1  and column  4 with corr  0.988
##   Means:  0.315 vs 0.302 so flagging column 1
## Compare row 26  and column  30 with corr  0.803
##   Means:  0.379 vs 0.3 so flagging column 26
## Compare row 4  and column  13 with corr  0.719
##   Means:  0.269 vs 0.289 so flagging column 13
## Compare row 4  and column  11 with corr  0.719
##   Means:  0.243 vs 0.285 so flagging column 11
## Compare row 4  and column  14 with corr  0.805
##   Means:  0.213 vs 0.288 so flagging column 14
## Compare row 16  and column  18 with corr  0.709
##   Means:  0.404 vs 0.289 so flagging column 16
## Compare row 18  and column  17 with corr  0.772
##   Means:  0.309 vs 0.273 so flagging column 18
## Compare row 17  and column  20 with corr  0.773
##   Means:  0.286 vs 0.268 so flagging column 17
## Compare row 5  and column  25 with corr  0.806
##   Means:  0.322 vs 0.264 so flagging column 5
## Compare row 10  and column  30 with corr  0.761
##   Means:  0.353 vs 0.255 so flagging column 10
## Compare row 22  and column  2 with corr  0.912
##   Means:  0.263 vs 0.242 so flagging column 22
## All correlations <= 0.7
```

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

```
##  [1] "concavity_mean"          "concave_points_mean"
##  [3] "concave points_worst"    "compactness_mean"
##  [5] "perimeter_worst"         "concavity_worst"
##  [7] "radius_worst"            "perimeter_mean"
##  [9] "area_worst"              "radius_mean"
## [11] "compactness_wors"        "perimeter_se"
## [13] "radius_se"               "area_se"
## [15] "compactness_se"          "concave_points_se"
## [17] "concavity_se"            "smoothness_mean"
## [19] "fractal_dimension_mean" "texture_worst"
```

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

GRID SEARCH WITH CARET

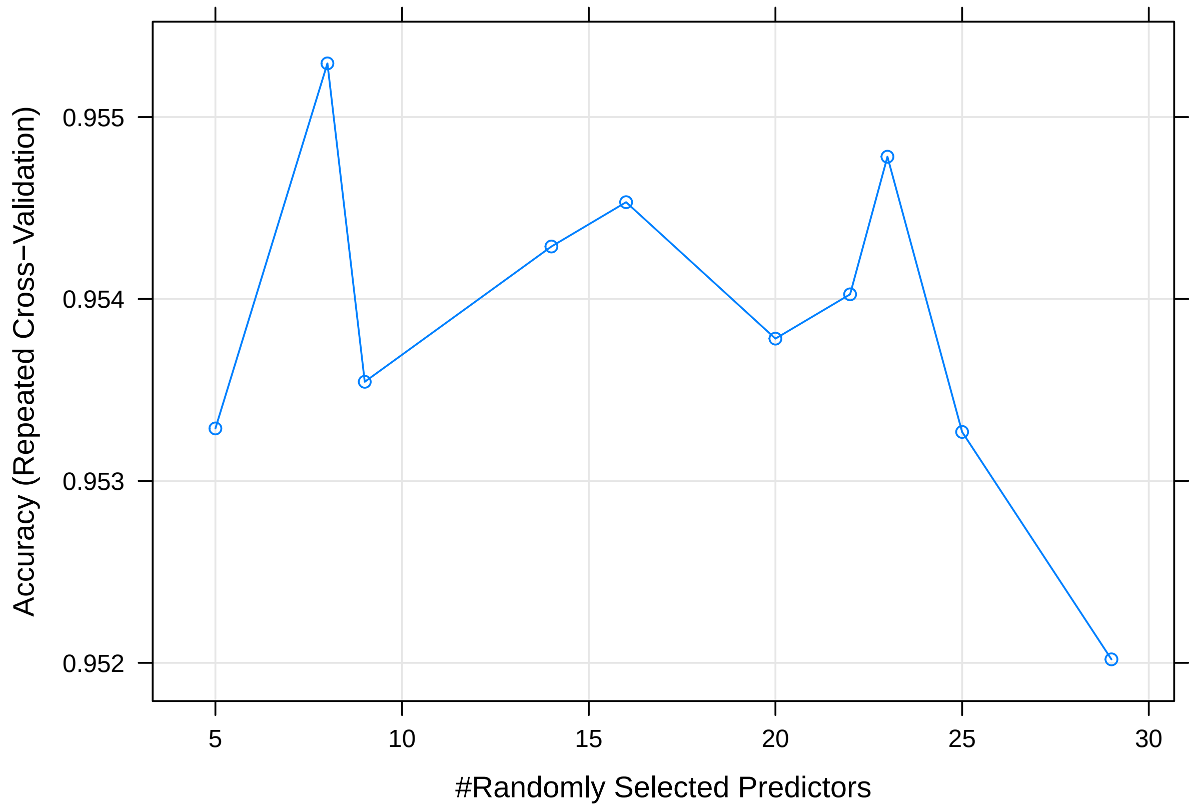
Automatic Grid

```
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
##
## 398 samples
##  30 predictor
##   2 classes: 'benign', 'malignant'
##
## Pre-processing: scaled (30), centered (30)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 358, 358, 359, 358, 358, 358, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    5    0.9532885  0.8998301
##    8    0.9552949  0.9042609
##    9    0.9535449  0.9005992
##   14    0.9542885  0.9022334
##   16    0.9545321  0.9027631
##   20    0.9537821  0.9011350
##   22    0.9540256  0.9015140
##   23    0.9547821  0.9032354
##   25    0.9532692  0.9000211
##   29    0.9520192  0.8971656
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
```

```
plot(model_rf_tune_auto)
```



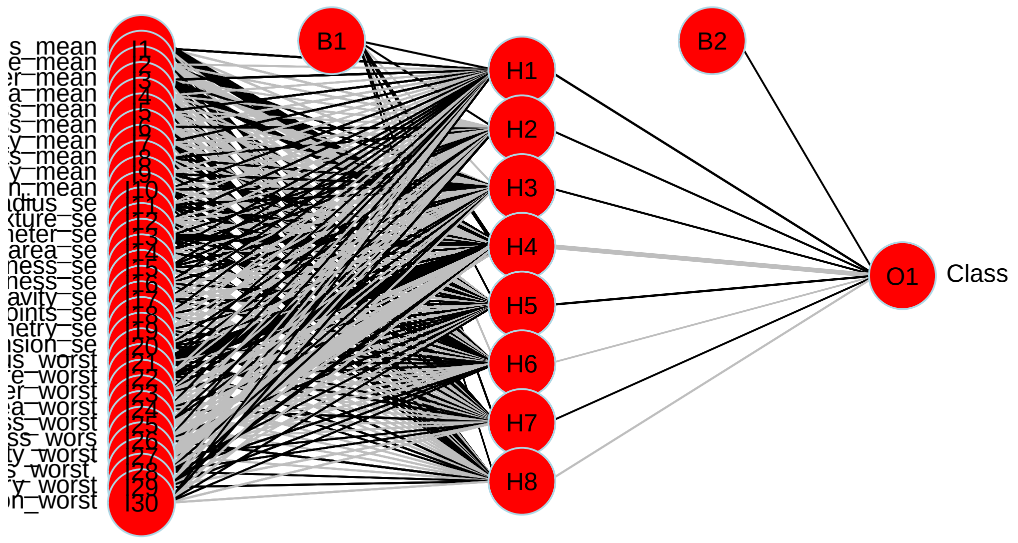
NEURAL NETWORK MODEL

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

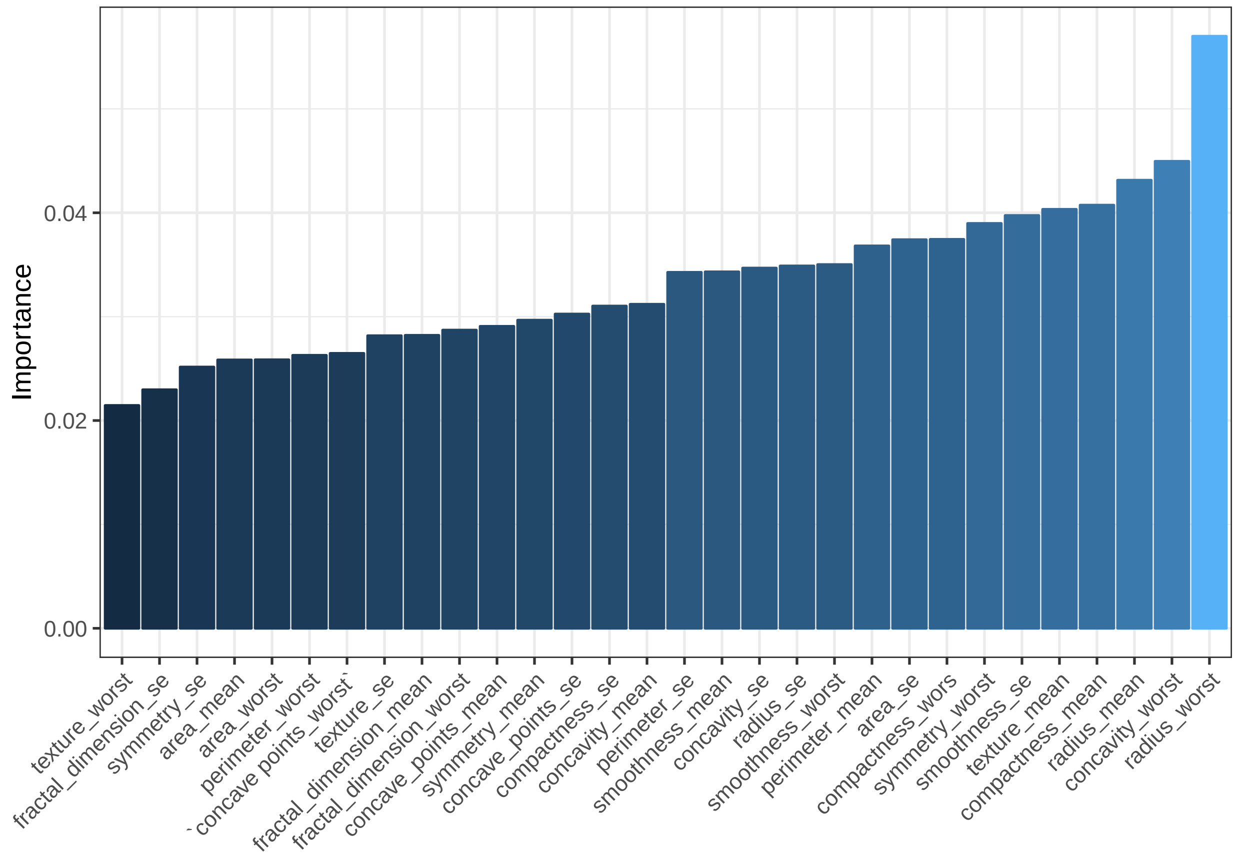
```
## # weights:  257
## initial  value 289.897687
## iter  10 value 249.315010
## iter  20 value 189.002331
## iter  30 value 137.130717
## iter  40 value 96.998986
## iter  50 value 79.761801
## iter  60 value 75.251591
## iter  70 value 71.625598
## iter  80 value 71.542152
## iter  90 value 71.531759
## iter 100 value 70.877250
## final  value 70.877250
## stopped after 100 iterations
```

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

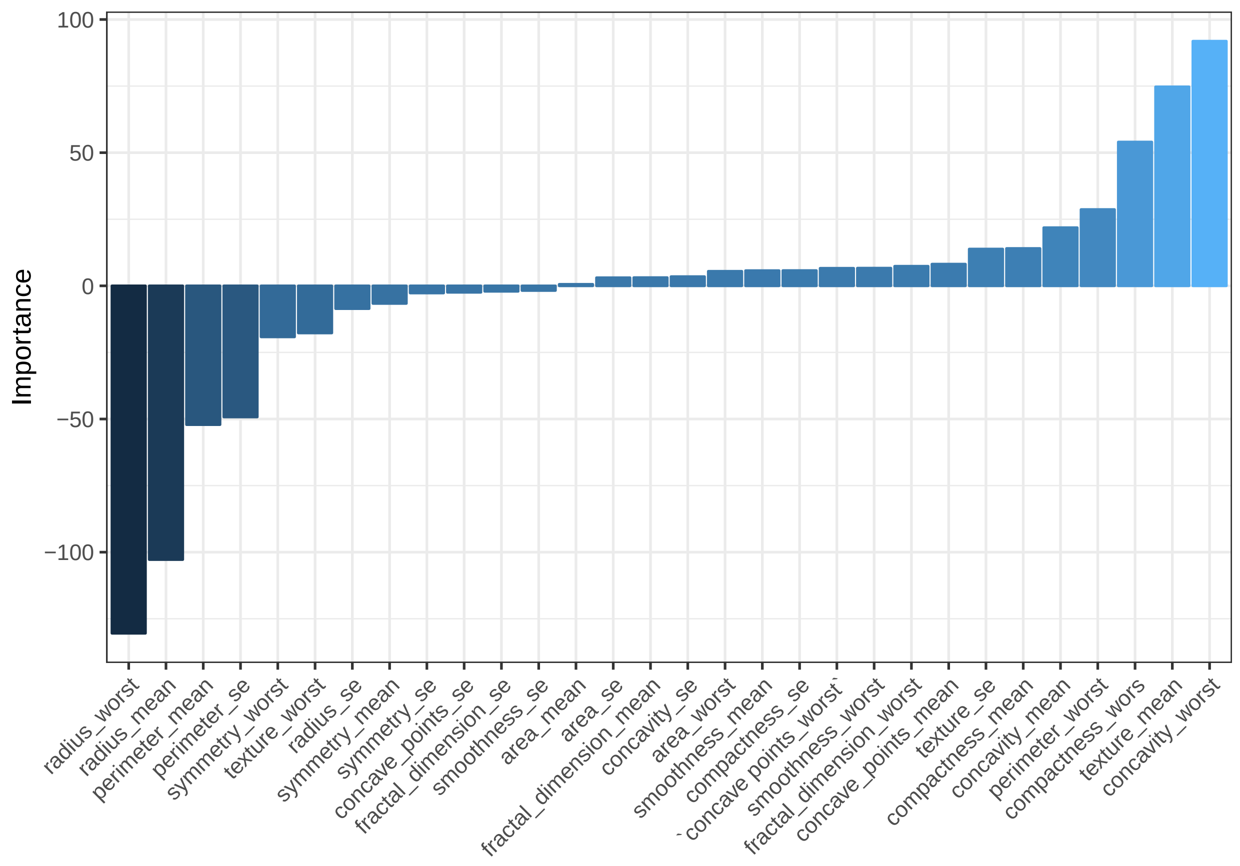
```r
plotnet(model_nnet, cex_val =.8,max_sp=T,circle_cex=5,circle_col = 'red')
```



```r
#Relative importance of input variables in neural networks using Garson's algorithm:
garson(model_nnet, las = 2) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
olden(model_nnet) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



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

Prediction

```
#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:  170
##
##
```

```
##                  | Predict Diagnosis 
## Actual Diagnosis |    benign | malignant | Row Total | 
## -----------------|-----------|-----------|-----------|
##           benign |        96 |        11 |       107 | 
##                  |     0.565 |     0.065 |           | 
## -----------------|-----------|-----------|-----------|
##        malignant |         7 |        56 |        63 | 
##                  |     0.041 |     0.329 |           | 
## -----------------|-----------|-----------|-----------|
##     Column Total |       103 |        67 |       170 | 
## -----------------|-----------|-----------|-----------|
## 
## 
```