Breast Cancer Wisconsin (Diagnostic) Data Set

HarvardX - PH125.9x Data Science - Choose your own project

Eduardo Sthory

29 de noviembre de 2019

1. Overview

In this project, the HarvardX "Choose-your-own: PH125.9x Data Science: Capstone Project" assignment will be fulfilled.

We will begin with an Overview and an introduction to the topic, then an Analysis section consisting of data exploration, data preparation for models, generation, adjustment and evaluation of machine learning models to predict whether a breast cancer cell is benign or malignant

Then it will show the results and finally the conclusions.

2. Introduction

Breast cancer is a pathology in which a malignant tumor It develops in the breast tissue and is one of the most common types of cancer, especially in women, more than 411,000 deaths annually worldwide.

The global incidence is thought to be more than 23.6 million new cases of cancer each year by 2030.

Mammography is used to detect breast cancer early, then proceed to the biopsy test with the "fine needle aspirates" (FNA) method that is the subject of this project, which is a safe method and accurate.

After aspirating a drop of fluid into the breast mass, it is analyzed with the help of the miscroscope and photographed and analyzed with an "Xcyt" image analysis program, the edges of the nuclei are determined from initial points placed manually near these edges to then perform the interactive diagnostic process

This project will train and evaluate the performance of several machine learning models that can predict the malignancy or not of a tumor, it is a binary classification problem where we will evaluate the accuracy, sensitivity and specificity and thus to find the best model.

This is very useful since the diagnosis at an early stage facilitates the subsequent clinical management of patients and It can increase the survival rate of breast cancer patients.

Dataset

The dataset is in https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/version/2.

The .csv format file that contains the data (data2.csv) can be loaded from the github account. The dataset is in my github in this address: https://github.com/sthory/Harvad-Data-Science/tree/master/Breast%20Cancer%20Project

The characteristics of the data set that are the cell nuclei in the image are described below.

Description:

idID number: Id number

diagnosis: The diagnosis of breast tissues (M = malignant, B = benign)

radius_mean: mean of distances from center to points on the perimeter

texture_mean: standard deviation of gray-scale values

perimeter_mean: mean size of the core tumor

area_mean: mean area

smoothness_mean: mean of local variation in radius lengths

compactness_mean: mean of perimeter^2 / area - 1.0

concavity_mean: mean of severity of concave portions of the contour

concave points_mean: mean for number of concave portions of the contour

symmetry_mean: symmetry mean

fractal_dimension_mean: mean for 'coastline approximation' - 1

radius_se: standard error for the mean of distances from center to points on the perimeter

texture_se: standard error for standard deviation of gray-scale values

perimeter_se: standard error of perimeter

area_se: standard error of area

smoothness_se: standard error for local variation in radius lengths

compactness_se: standard error for perimeter^2 / area - 1.0

concavity_se: standard error for severity of concave portions of the contour

concave points_se: standard error for number of concave portions of the contour

symmetry_se: standard error for symmetry

fractal_dimension_se: standard error for 'coastline approximation' - 1

radius_worst: 'worst' or largest mean value for mean of distances from center to points on the perimeter

texture_worst: 'worst' or largest mean value for standard deviation of gray-scale values

perimeter_worst: 'worst' perimeter

area_worst: 'worst' area

smoothness_worst: 'worst' or largest mean value for local variation in radius lengths

compactness_worst: 'worst' or largest mean value for perimeter^2 / area - 1.0

concavity_worst: 'worst' or largest mean value for severity of concave portions of the contour

concave points_worst: 'worst' or largest mean value for number of concave portions of the contour

symmetry_worst: 'worst' symmetry

fractal_dimension_worst: 'worst' or largest mean value for 'coastline approximation' - 1"

Load Libraries

```
if(!require(tidyverse))
  install.packages("tidyverse",
                   repos = "http://cran.us.r-project.org")
if(!require(caret))
  install.packages("caret",
                   repos = "http://cran.us.r-project.org")
if(!require(ggplot2))
  install.packages("ggplot2",
                   repos = "http://cran.us.r-project.org")
if(!require(funModeling))
  install.packages("funModeling",
                   repos = "http://cran.us.r-project.org")
if(!require(corrplot))
  install.packages("corrplot",
                   repos = "http://cran.us.r-project.org")
library(tidyverse)
library(caret)
```

```
library(ggplot2)
library(funModeling)
library(corrplot)
```

Load data

```
datacancer = read.csv("data2.csv")  # read csv file
```

Check for missing values

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

```
                     id               diagnosis             radius_mean
                      0                       0                       0
           texture_mean          perimeter_mean               area_mean
                      0                       0                       0
        smoothness_mean         compactness_mean          concavity_mean
                      0                       0                       0
     concave.points_mean           symmetry_mean  fractal_dimension_mean
                      0                       0                       0
              radius_se              texture_se            perimeter_se
                      0                       0                       0
                area_se           smoothness_se          compactness_se
                      0                       0                       0
           concavity_se       concave.points_se             symmetry_se
                      0                       0                       0
    fractal_dimension_se            radius_worst           texture_worst
                      0                       0                       0
         perimeter_worst              area_worst        smoothness_worst
                      0                       0                       0
       compactness_worst         concavity_worst    concave.points_worst
                      0                       0                       0
          symmetry_worst fractal_dimension_worst                       X
                      0                       0                     569
```

Validation Dataset, split out validation dataset and create a list of 80% of the rows in the original dataset we can use for training. Remove the Id column and 33 number columns, after convert the data to numeric.

```
datacancer <- datacancer[,-1]
datacancer <- datacancer[,-ncol(datacancer)]
```

Create dataset of training and test validation

```
set.seed(1)
validationIndex <- createDataPartition(datacancer$diagnosis,
                                       p=0.80,
                                       list=FALSE)
```

Select 20% of the data for validation

```
validation <- datacancer[-validationIndex,]
```

Use the remaining 80% of data to training and testing the models

```
dataset <- datacancer[validationIndex,]
```

3. Analysis

The objective of this step in the process is to better understand the problem.

Descriptive Statistics

Let's start off by confirming the dimensions of the dataset.

Explore dataset

Dataset dimensions

```
dim(dataset)
```

```
[1] 456  31
```

First ten rows

```
head(dataset, 10)
```

```
   diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
2          M       20.57        17.77         132.90    1326.0         0.08474
3          M       19.69        21.25         130.00    1203.0         0.10960
4          M       11.42        20.38          77.58     386.1         0.14250
5          M       20.29        14.34         135.10    1297.0         0.10030
7          M       18.25        19.98         119.60    1040.0         0.09463
8          M       13.71        20.83          90.20     577.9         0.11890
10         M       12.46        24.04          83.97     475.9         0.11860
11         M       16.02        23.24         102.70     797.8         0.08206
12         M       15.78        17.89         103.60     781.0         0.09710
13         M       19.17        24.80         132.40    1123.0         0.09740
   compactness_mean concavity_mean concave.points_mean symmetry_mean
2           0.07864        0.08690             0.07017        0.1812
3           0.15990        0.19740             0.12790        0.2069
4           0.28390        0.24140             0.10520        0.2597
5           0.13280        0.19800             0.10430        0.1809
7           0.10900        0.11270             0.07400        0.1794
8           0.16450        0.09366             0.05985        0.2196
10          0.23960        0.22730             0.08543        0.2030
11          0.06669        0.03299             0.03323        0.1528
12          0.12920        0.09954             0.06606        0.1842
13          0.24580        0.20650             0.11180        0.2397
   fractal_dimension_mean radius_se texture_se perimeter_se area_se
2                 0.05667    0.5435     0.7339        3.398   74.08
3                 0.05999    0.7456     0.7869        4.585   94.03
4                 0.09744    0.4956     1.1560        3.445   27.23
5                 0.05883    0.7572     0.7813        5.438   94.44
7                 0.05742    0.4467     0.7732        3.180   53.91
8                 0.07451    0.5835     1.3770        3.856   50.96
10                0.08243    0.2976     1.5990        2.039   23.94
11                0.05697    0.3795     1.1870        2.466   40.51
12                0.06082    0.5058     0.9849        3.564   54.16
13                0.07800    0.9555     3.5680       11.070  116.20
   smoothness_se compactness_se concavity_se concave.points_se symmetry_se
2       0.005225       0.013080      0.01860          0.013400     0.01389
3       0.006150       0.040060      0.03832          0.020580     0.02250
4       0.009110       0.074580      0.05661          0.018670     0.05963
5       0.011490       0.024610      0.05688          0.018850     0.01756
7       0.004314       0.013820      0.02254          0.010390     0.01369
8       0.008805       0.030290      0.02488          0.014480     0.01486
10      0.007149       0.072170      0.07743          0.014320     0.01789
11      0.004029       0.009269      0.01101          0.007591     0.01460
12      0.005771       0.040610      0.02791          0.012820     0.02008
13      0.003139       0.082970      0.08890          0.040900     0.04484
   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
```

```
2              0.003532        24.99         23.41          158.80     1956.0
3              0.004571        23.57         25.53          152.50     1709.0
4              0.009208        14.91         26.50           98.87      567.7
5              0.005115        22.54         16.67          152.20     1575.0
7              0.002179        22.88         27.66          153.20     1606.0
8              0.005412        17.06         28.14          110.60      897.0
10             0.010080        15.09         40.68           97.65      711.4
11             0.003042        19.19         33.88          123.80     1150.0
12             0.004144        20.42         27.28          136.50     1299.0
13             0.012840        20.96         29.94          151.70     1332.0
   smoothness_worst compactness_worst concavity_worst concave.points_worst
2            0.1238            0.1866          0.2416               0.18600
3            0.1444            0.4245          0.4504               0.24300
4            0.2098            0.8663          0.6869               0.25750
5            0.1374            0.2050          0.4000               0.16250
7            0.1442            0.2576          0.3784               0.19320
8            0.1654            0.3682          0.2678               0.15560
10           0.1853            1.0580          1.1050               0.22100
11           0.1181            0.1551          0.1459               0.09975
12           0.1396            0.5609          0.3965               0.18100
13           0.1037            0.3903          0.3639               0.17670
   symmetry_worst fractal_dimension_worst
2          0.2750                 0.08902
3          0.3613                 0.08758
4          0.6638                 0.17300
5          0.2364                 0.07678
7          0.3063                 0.08368
8          0.3196                 0.11510
10         0.4366                 0.20750
11         0.2948                 0.08452
12         0.3792                 0.10480
13         0.3176                 0.10230
```

Field class

```
sapply(dataset, class)
```

```
             diagnosis            radius_mean           texture_mean
              "factor"              "numeric"              "numeric"
        perimeter_mean              area_mean        smoothness_mean
              "numeric"              "numeric"              "numeric"
       compactness_mean         concavity_mean     concave.points_mean
              "numeric"              "numeric"              "numeric"
         symmetry_mean  fractal_dimension_mean              radius_se
              "numeric"              "numeric"              "numeric"
            texture_se           perimeter_se                area_se
              "numeric"              "numeric"              "numeric"
         smoothness_se         compactness_se           concavity_se
              "numeric"              "numeric"              "numeric"
      concave.points_se            symmetry_se    fractal_dimension_se
              "numeric"              "numeric"              "numeric"
          radius_worst          texture_worst        perimeter_worst
              "numeric"              "numeric"              "numeric"
            area_worst        smoothness_worst       compactness_worst
              "numeric"              "numeric"              "numeric"
       concavity_worst    concave.points_worst         symmetry_worst
              "numeric"              "numeric"              "numeric"
fractal_dimension_worst
              "numeric"
```

Convert input values to numeric

```
for(i in 2:ncol(dataset)) {
  dataset[,i] <- as.numeric(as.character(dataset[,i]))
}
```

Dataset Summary

```
summary(dataset)
```

```
 diagnosis  radius_mean      texture_mean   perimeter_mean     area_mean
 B:286      Min.   : 6.981   Min.   : 9.71   Min.   : 43.79   Min.   : 143.5
 M:170      1st Qu.:11.697   1st Qu.:16.20   1st Qu.: 74.83   1st Qu.: 419.9
            Median :13.415   Median :18.89   Median : 86.42   Median : 557.0
            Mean   :14.137   Mean   :19.26   Mean   : 92.01   Mean   : 656.4
            3rd Qu.:15.797   3rd Qu.:21.60   3rd Qu.:103.72   3rd Qu.: 784.1
            Max.   :28.110   Max.   :33.81   Max.   :188.50   Max.   :2501.0
 smoothness_mean   compactness_mean  concavity_mean    concave.points_mean
 Min.   :0.05263   Min.   :0.01938   Min.   :0.00000   Min.   :0.00000
 1st Qu.:0.08604   1st Qu.:0.06362   1st Qu.:0.02874   1st Qu.:0.01998
 Median :0.09502   Median :0.09094   Median :0.06071   Median :0.03333
 Mean   :0.09614   Mean   :0.10350   Mean   :0.08876   Mean   :0.04881
 3rd Qu.:0.10495   3rd Qu.:0.13050   3rd Qu.:0.12965   3rd Qu.:0.07373
 Max.   :0.16340   Max.   :0.34540   Max.   :0.42680   Max.   :0.20120
 symmetry_mean    fractal_dimension_mean   radius_se        texture_se
 Min.   :0.1060   Min.   :0.04996        Min.   :0.1115   Min.   :0.3602
 1st Qu.:0.1619   1st Qu.:0.05768        1st Qu.:0.2315   1st Qu.:0.8282
 Median :0.1784   Median :0.06132        Median :0.3194   Median :1.1285
 Mean   :0.1808   Mean   :0.06271        Mean   :0.4048   Mean   :1.2214
 3rd Qu.:0.1953   3rd Qu.:0.06604        3rd Qu.:0.4731   3rd Qu.:1.4783
 Max.   :0.3040   Max.   :0.09744        Max.   :2.8730   Max.   :4.8850
  perimeter_se       area_se         smoothness_se      compactness_se
 Min.   : 0.757   Min.   :  6.802   Min.   :0.001713   Min.   :0.002252
 1st Qu.: 1.573   1st Qu.: 17.858   1st Qu.:0.005247   1st Qu.:0.012752
 Median : 2.276   Median : 24.195   Median :0.006382   Median :0.019960
 Mean   : 2.848   Mean   : 40.556   Mean   :0.007025   Mean   :0.025213
 3rd Qu.: 3.288   3rd Qu.: 44.665   3rd Qu.:0.008118   3rd Qu.:0.031715
 Max.   :21.980   Max.   :542.200   Max.   :0.031130   Max.   :0.135400
  concavity_se     concave.points_se  symmetry_se       fractal_dimension_se
 Min.   :0.00000   Min.   :0.00000   Min.   :0.007882   Min.   :0.0008948
 1st Qu.:0.01464   1st Qu.:0.00743   1st Qu.:0.015125   1st Qu.:0.0022032
 Median :0.02574   Median :0.01089   Median :0.018610   Median :0.0031035
 Mean   :0.03157   Mean   :0.01168   Mean   :0.020313   Mean   :0.0037915
 3rd Qu.:0.04168   3rd Qu.:0.01483   3rd Qu.:0.022845   3rd Qu.:0.0044468
 Max.   :0.39600   Max.   :0.05279   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.:21.19   1st Qu.: 84.10   1st Qu.: 515.4
 Median :14.97   Median :25.47   Median : 97.61   Median : 686.5
 Mean   :16.28   Mean   :25.73   Mean   :107.27   Mean   : 882.9
 3rd Qu.:18.77   3rd Qu.:30.13   3rd Qu.:125.03   3rd Qu.:1073.5
 Max.   :36.04   Max.   :47.16   Max.   :251.20   Max.   :4254.0
 smoothness_worst  compactness_worst concavity_worst  concave.points_worst
 Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.11620   1st Qu.:0.14415   1st Qu.:0.1139   1st Qu.:0.06398
 Median :0.13150   Median :0.21070   Median :0.2290   Median :0.09881
 Mean   :0.13256   Mean   :0.25331   Mean   :0.2731   Mean   :0.11460
 3rd Qu.:0.14683   3rd Qu.:0.33565   3rd Qu.:0.3856   3rd Qu.:0.16315
 Max.   :0.21840   Max.   :1.05800   Max.   :1.1700   Max.   :0.29100
 symmetry_worst   fractal_dimension_worst
```

```
 Min.   :0.1565   Min.   :0.05504
 1st Qu.:0.2497   1st Qu.:0.07109
 Median :0.2821   Median :0.08005
 Mean   :0.2897   Mean   :0.08403
 3rd Qu.:0.3174   3rd Qu.:0.09219
 Max.   :0.6638   Max.   :0.20750
```

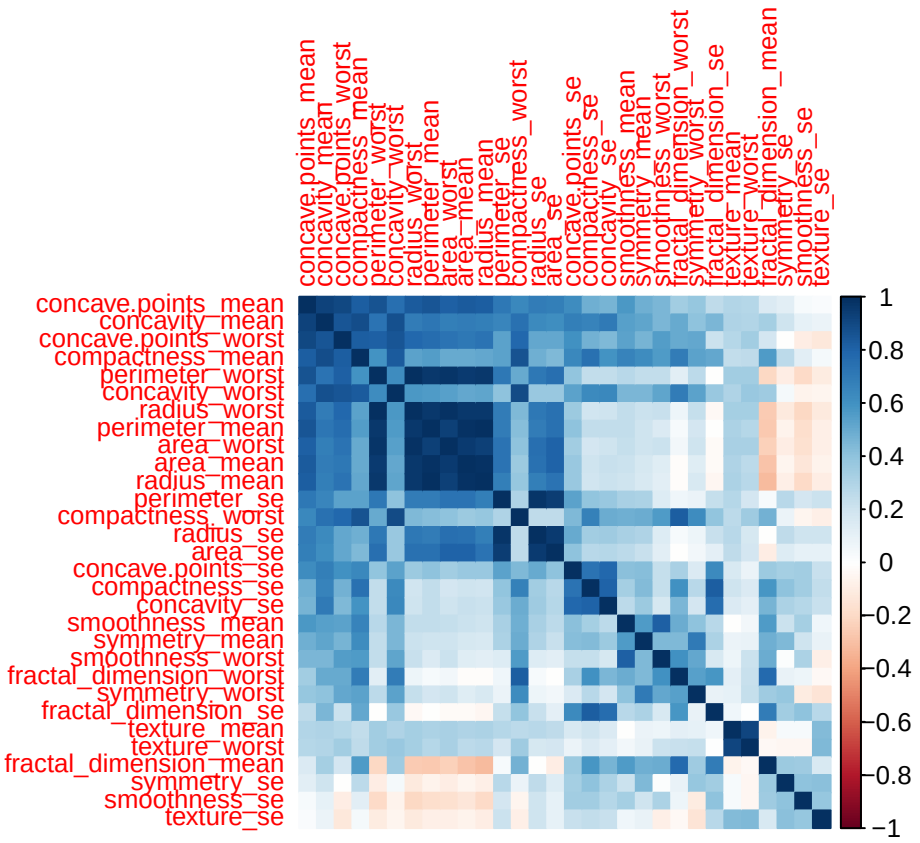
Diagnosis distribution

```
cbind(freq=table(dataset$diagnosis),
      percentage=prop.table(table(dataset$diagnosis))*100)
```

```
  freq percentage
B  286    62.7193
M  170    37.2807
```

We see the correlation between the attributes. Summarize correlations between input variables.

```
#cor(dataset[,2:ncol(dataset)])

# Correlation plot
correlationMatrix <- cor(dataset[,2:ncol(dataset)])
corrplot(correlationMatrix,
         method = "color",
         order = "FPC",
         tl.cex = 0.8)
```

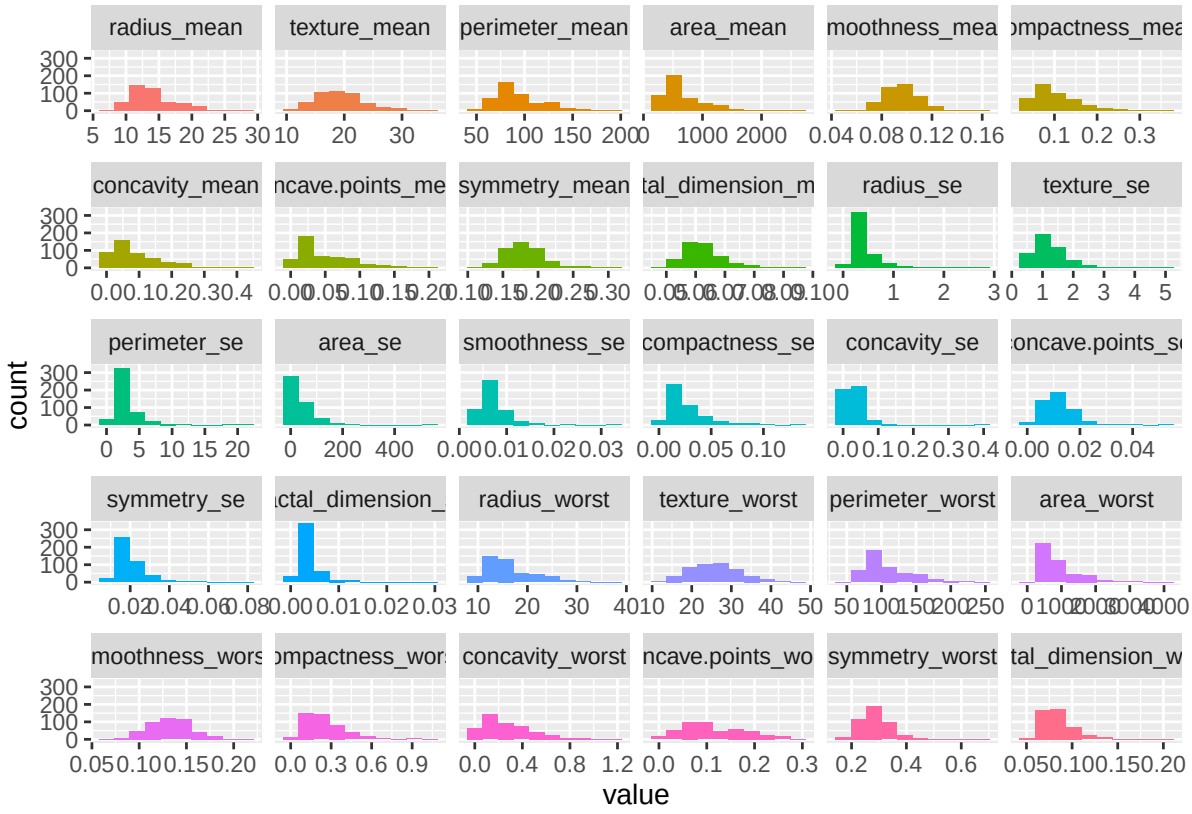


The graph shows that many variables are highly correlated with each other.

Unimodal Data Visualizations

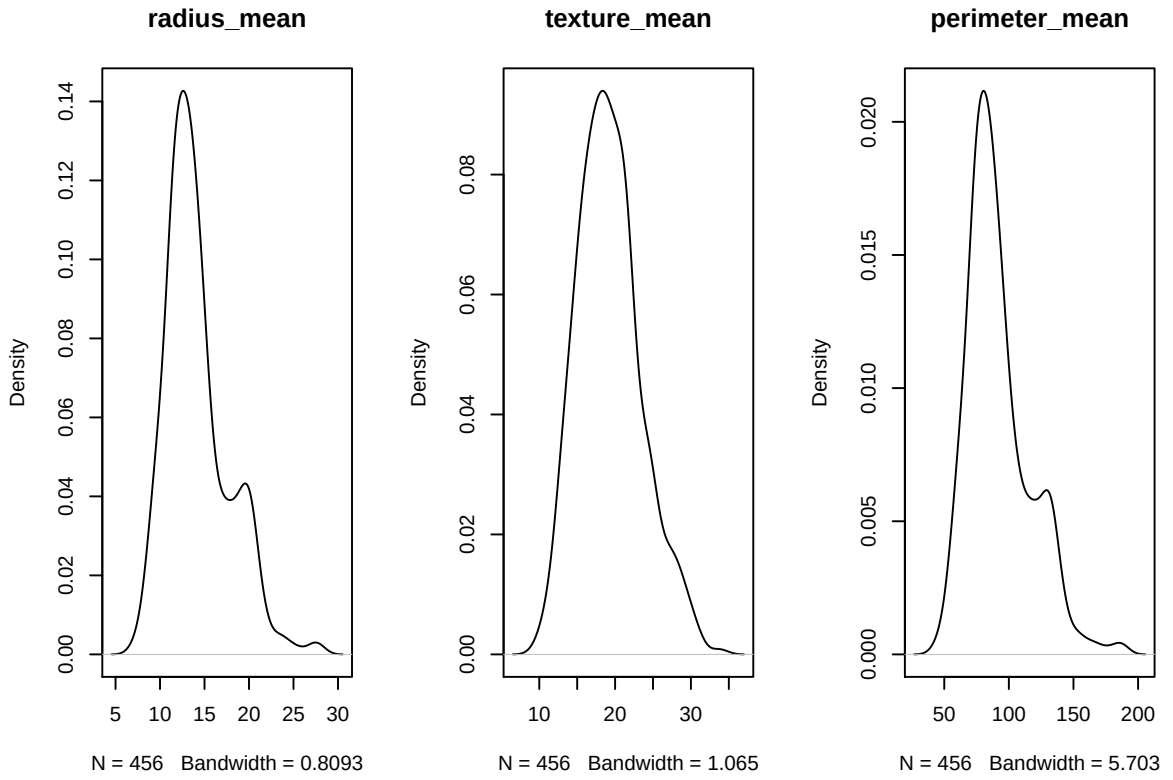
This is the distribution of individual attributes with histograms in the data set.

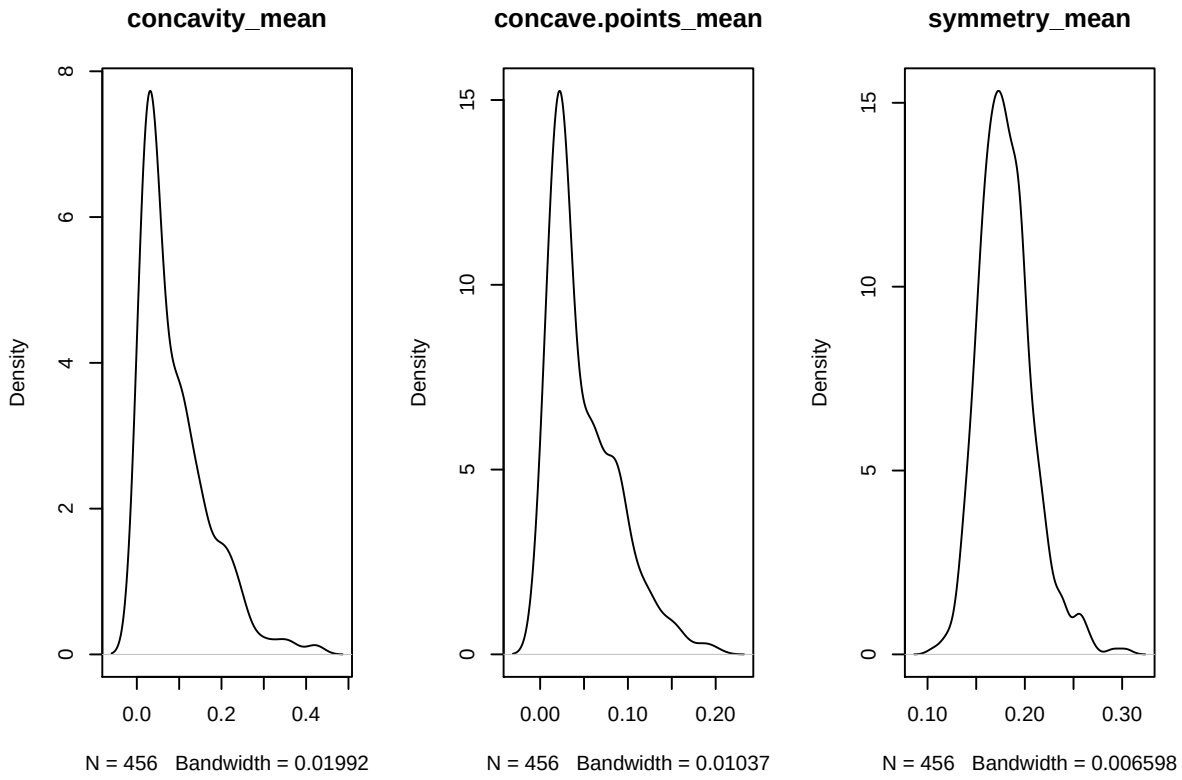
```
# Graphics values
lineas <- 5
columnas <- 6

plot_num(dataset %>% select(-diagnosis),
         bins=10)
```

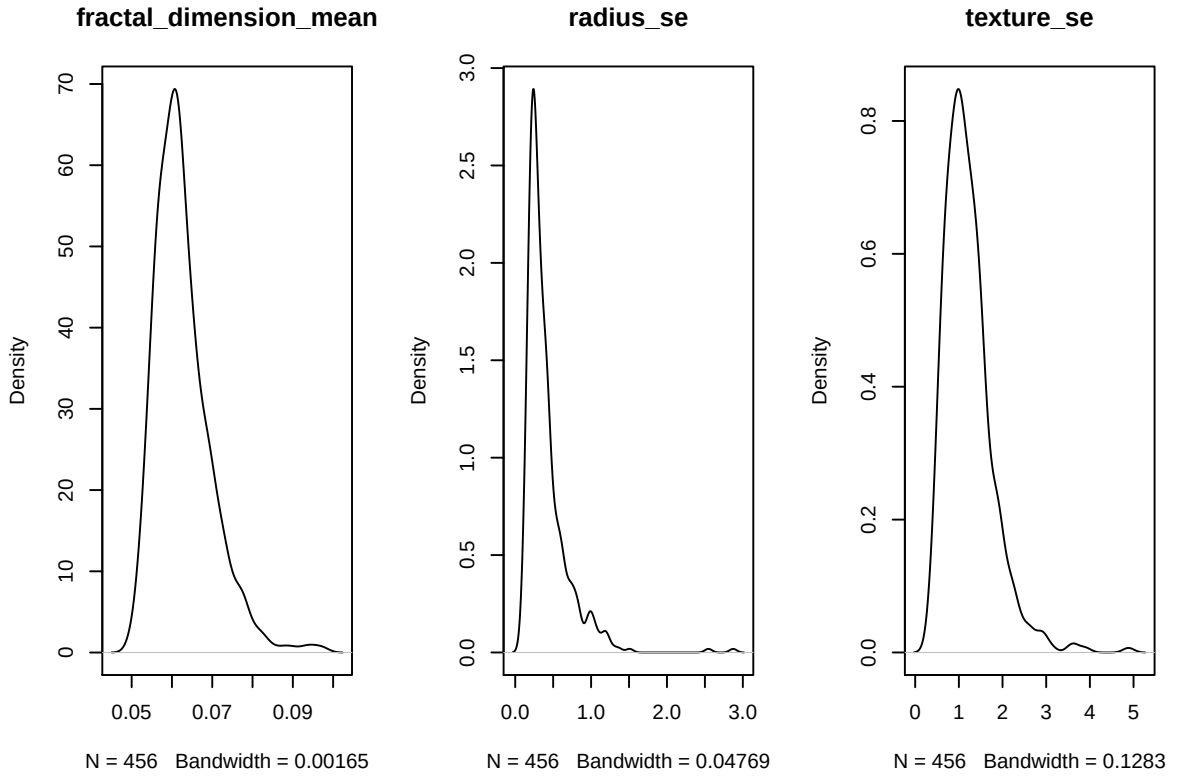


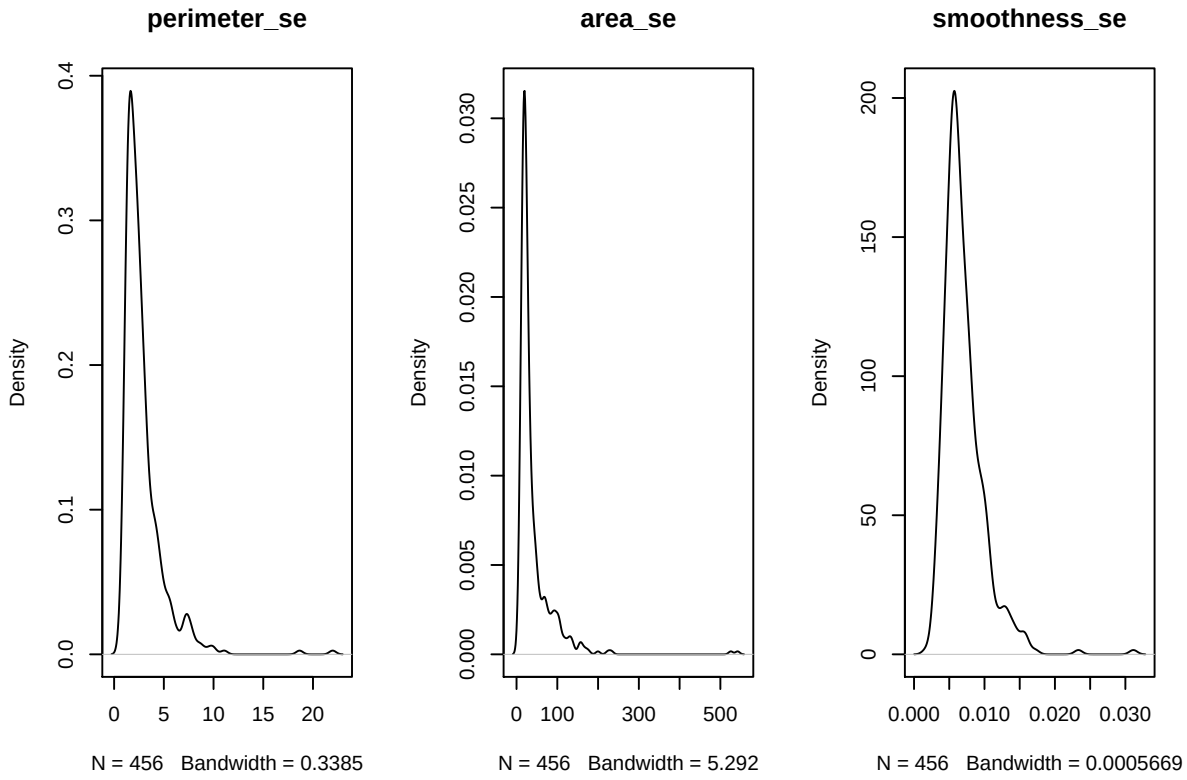
Bimodal and normal distributions can be observed. To get a more uniform view of the distributions we will see density graphs

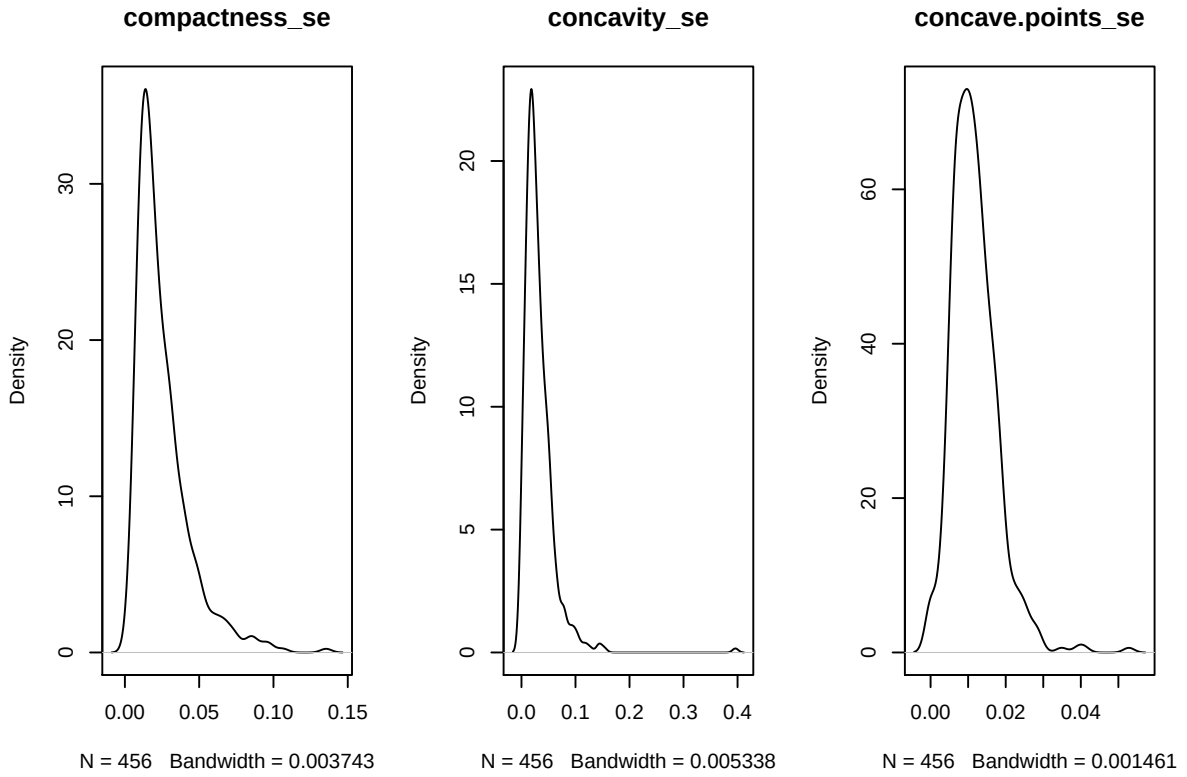
```
# density plot for each attribute

lineas <- 1
columnas <- 3
# density plot for each attribute
par(mfrow=c(lineas,columnas))
for(i in 2:ncol(dataset)) {
    plot(density(dataset[,i]),
         main=names(dataset)[i])
}
```

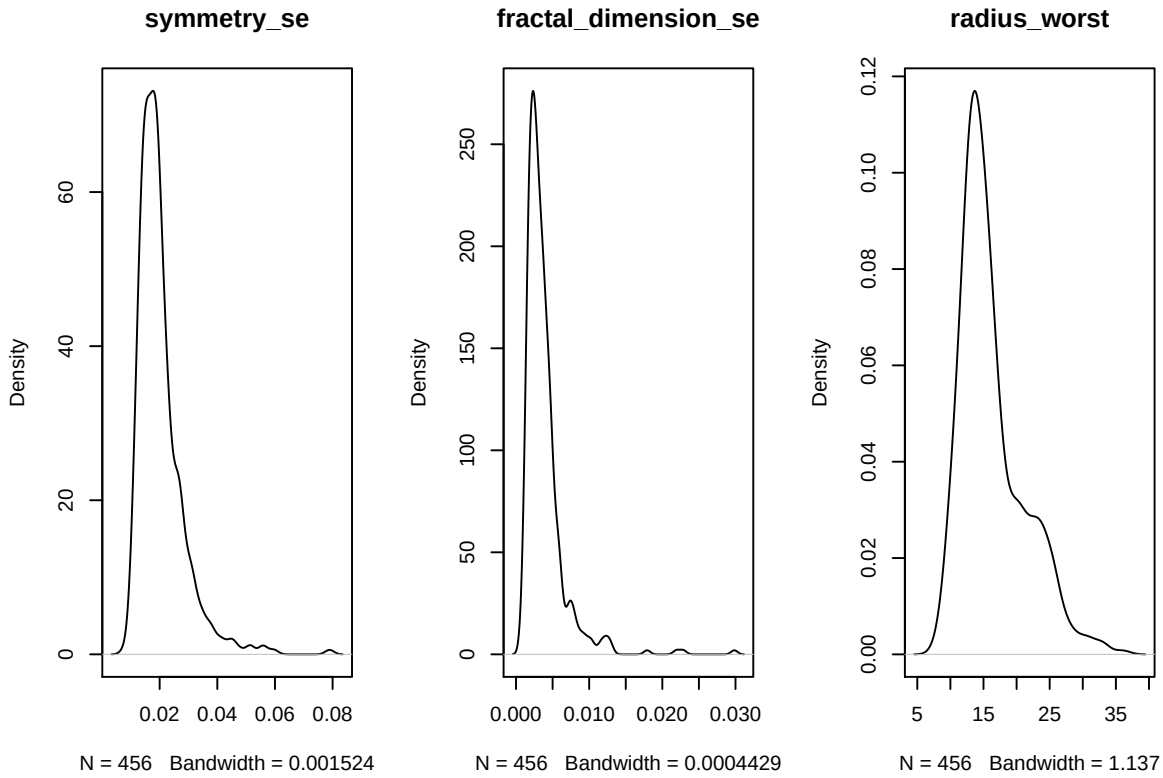

radius_mean
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
5
10
15
20
25
30
N = 456 Bandwidth = 0.8093
texture_mean
Density
0.00
0.02
0.04
0.06
0.08
10
20
30
N = 456 Bandwidth = 1.065
perimeter_mean
Density
0.000
0.005
0.010
0.015
0.020
50
100
150
200
N = 456 Bandwidth = 5.703
area_mean
Density
0.0000
0.0005
0.0010
0.0015
0
500
1500
2500
N = 456 Bandwidth = 71.9
smoothness_mean
Density
0
5
10
15
20
25
0.04
0.08
0.12
0.16
N = 456 Bandwidth = 0.003733
compactness_mean
Density
0
2
4
6
8
0.0
0.1
0.2
0.3
0.4
N = 456 Bandwidth = 0.0132

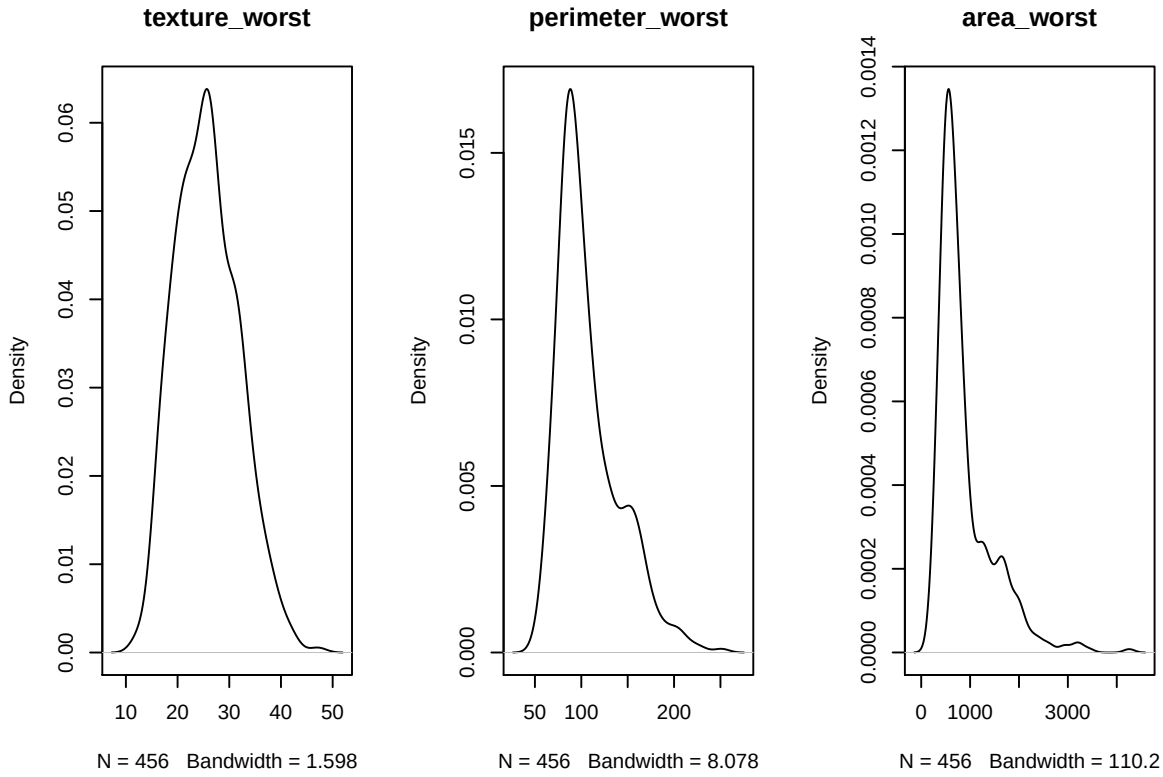

concavity_mean
Density
N = 456 Bandwidth = 0.01992
concave.points_mean
Density
N = 456 Bandwidth = 0.01037
symmetry_mean
Density
N = 456 Bandwidth = 0.006598

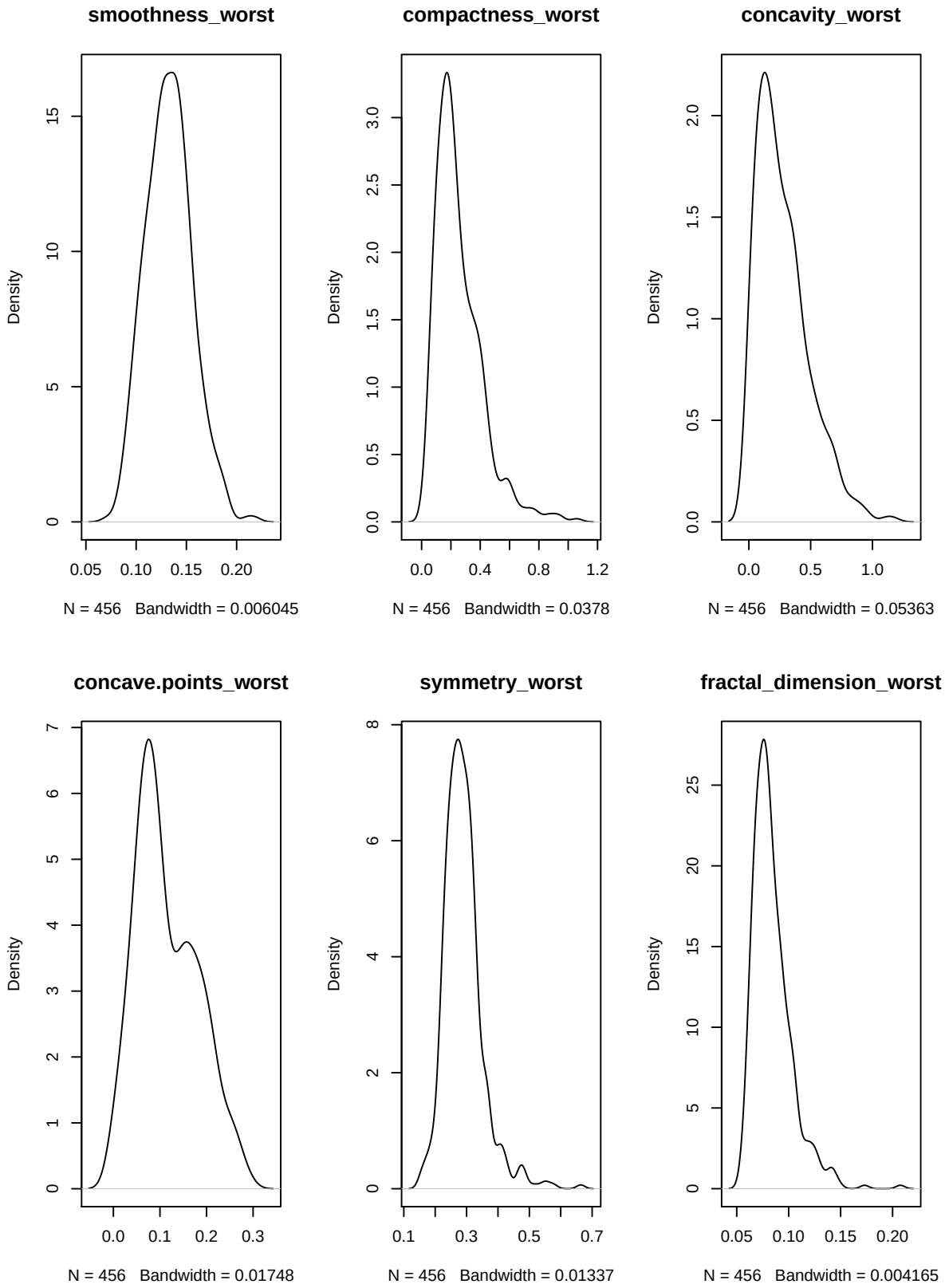

fractal_dimension_mean
Density
N = 456 Bandwidth = 0.00165
radius_se
Density
N = 456 Bandwidth = 0.04769
texture_se
Density
N = 456 Bandwidth = 0.1283


perimeter_se
Density
0.0
0.1
0.2
0.3
0.4
0
5
10
15
20
N = 456 Bandwidth = 0.3385
area_se
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0
100
300
500
N = 456 Bandwidth = 5.292
smoothness_se
Density
0
50
100
150
200
0.000
0.010
0.020
0.030
N = 456 Bandwidth = 0.0005669


compactness_se
Density
0
10
20
30
0.00
0.05
0.10
0.15
N = 456 Bandwidth = 0.003743
concavity_se
Density
0
5
10
15
20
0.0
0.1
0.2
0.3
0.4
N = 456 Bandwidth = 0.005338
concave.points_se
Density
0
20
40
60
0.00
0.02
0.04
N = 456 Bandwidth = 0.001461


symmetry_se
Density
0
20
40
60
0.02
0.04
0.06
0.08
N = 456 Bandwidth = 0.001524
fractal_dimension_se
Density
0
50
100
150
200
250
0.000
0.010
0.020
0.030
N = 456 Bandwidth = 0.0004429
radius_worst
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
5
15
25
35
N = 456 Bandwidth = 1.137

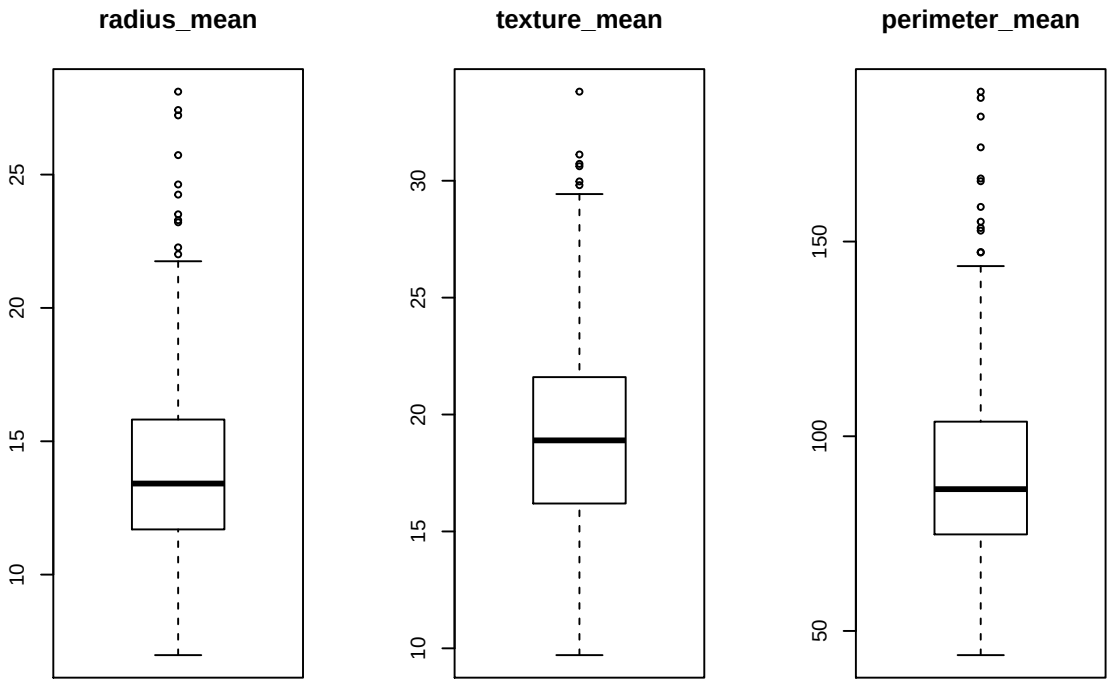

texture_worst
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
10
20
30
40
50
N = 456 Bandwidth = 1.598
perimeter_worst
Density
0.000
0.005
0.010
0.015
50
100
200
N = 456 Bandwidth = 8.078
area_worst
Density
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0.0014
0
1000
3000
N = 456 Bandwidth = 110.2

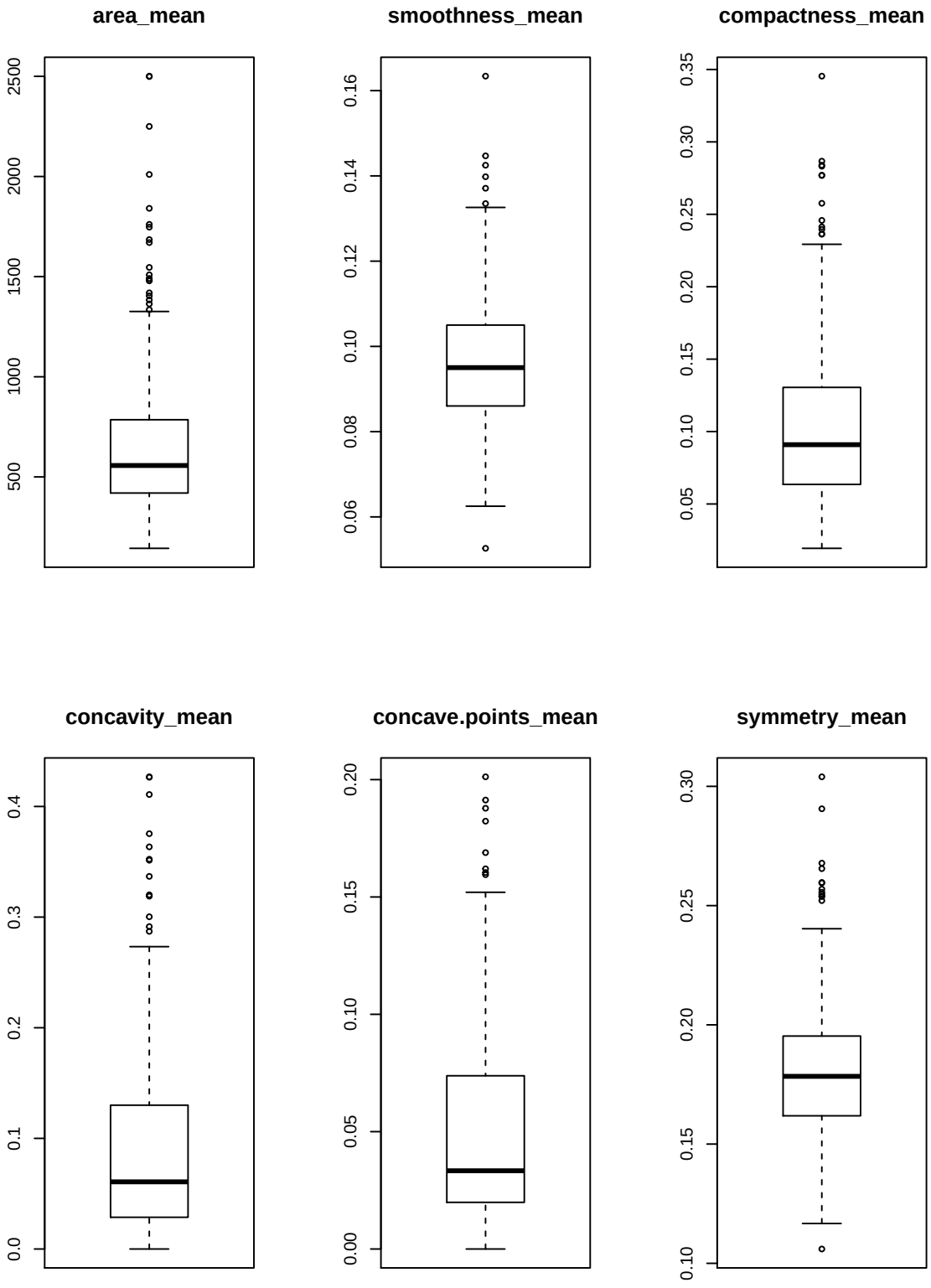


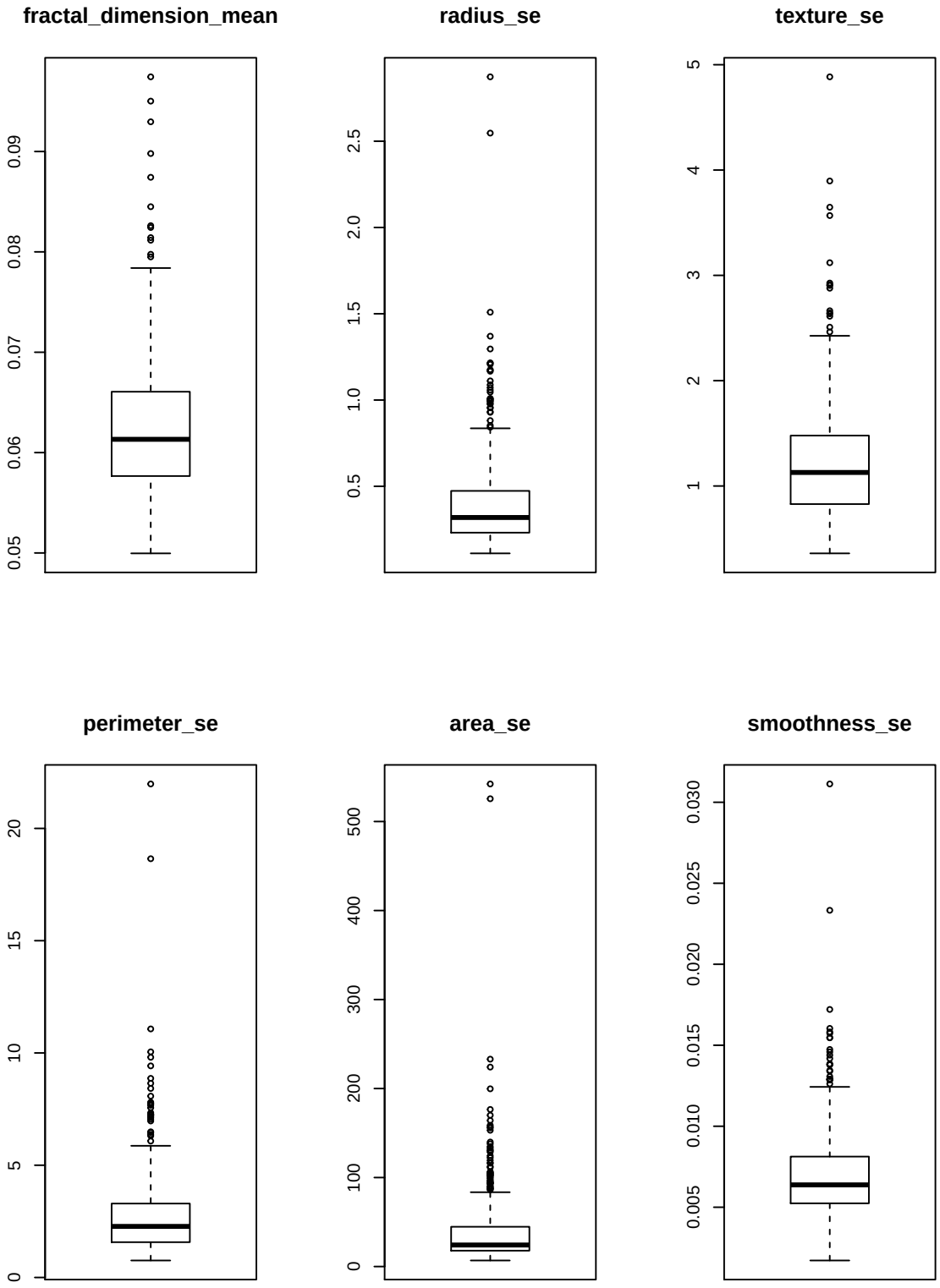
These frames give us the reason for the initial idea, you can see normal distributions and a few with two bumps (bimodal).

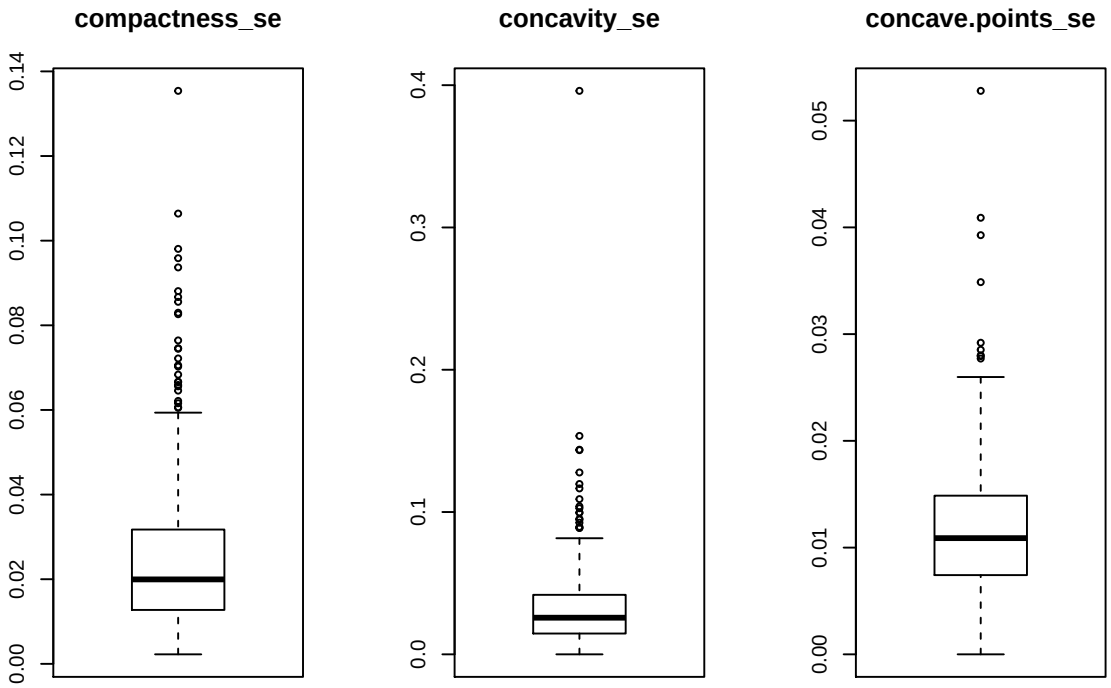
Now we see the distributions using box and whisker plots

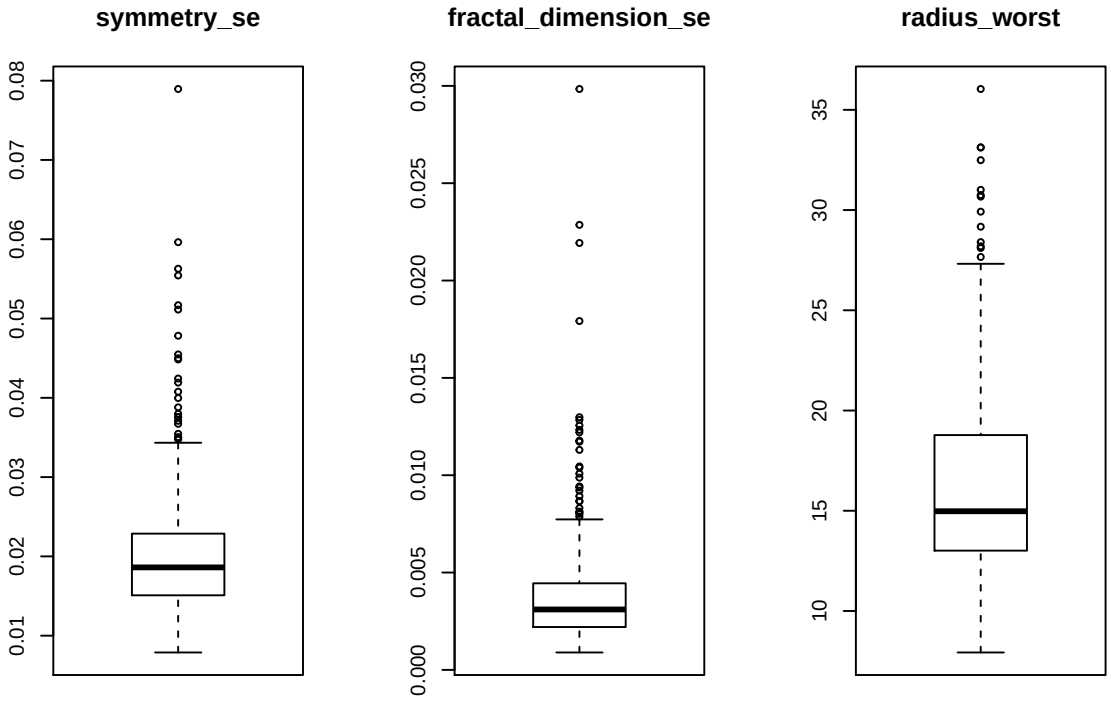
```r
# boxplots for each attribute
par(mfrow=c(lineas,columnas))
for(i in 2:ncol(dataset)) {
  boxplot(dataset[,i],
          main=names(dataset)[i])
}
```

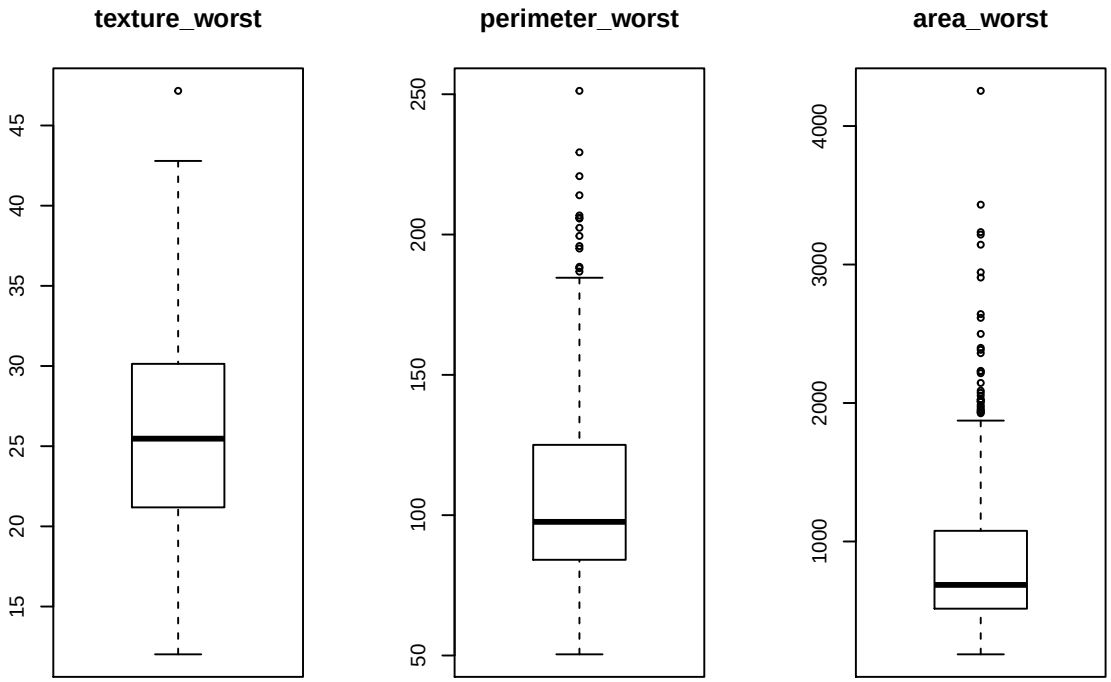



area_mean
500
1000
1500
2000
2500
smoothness_mean
0.06
0.08
0.10
0.12
0.14
0.16
compactness_mean
0.05
0.10
0.15
0.20
0.25
0.30
0.35
concavity_mean
0.0
0.1
0.2
0.3
0.4
concave.points_mean
0.00
0.05
0.10
0.15
0.20
symmetry_mean
0.10
0.15
0.20
0.25
0.30


fractal_dimension_mean
0.05
0.06
0.07
0.08
0.09
radius_se
0.5
1.0
1.5
2.0
2.5
texture_se
1
2
3
4
5
perimeter_se
0
5
10
15
20
area_se
0
100
200
300
400
500
smoothness_se
0.005
0.010
0.015
0.020
0.025
0.030

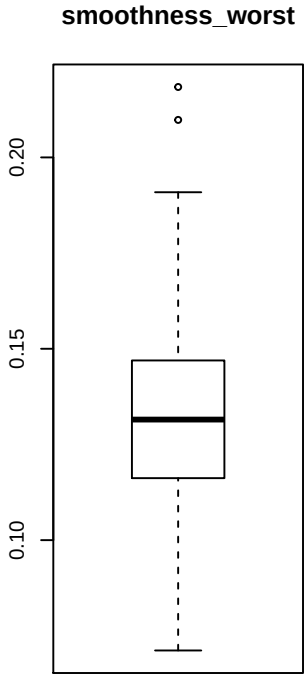

compactness_se
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
concavity_se
0.0
0.1
0.2
0.3
0.4
concave.points_se
0.00
0.01
0.02
0.03
0.04
0.05

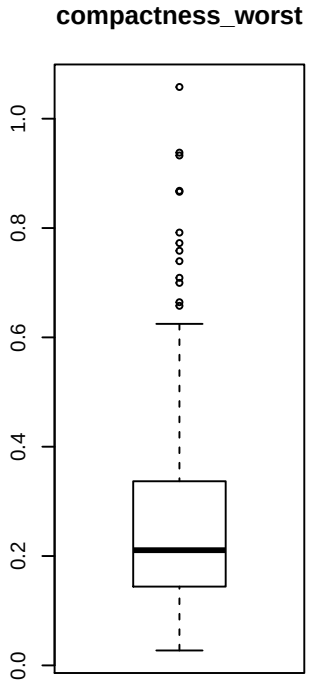

symmetry_se
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
fractal_dimension_se
0.000
0.005
0.010
0.015
0.020
0.025
0.030
radius_worst
10
15
20
25
30
35

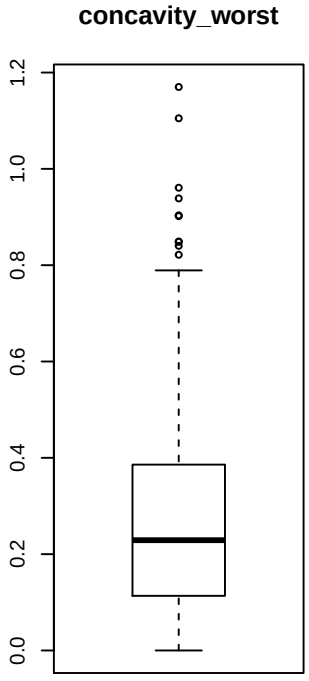

texture_worst
15
20
25
30
35
40
45

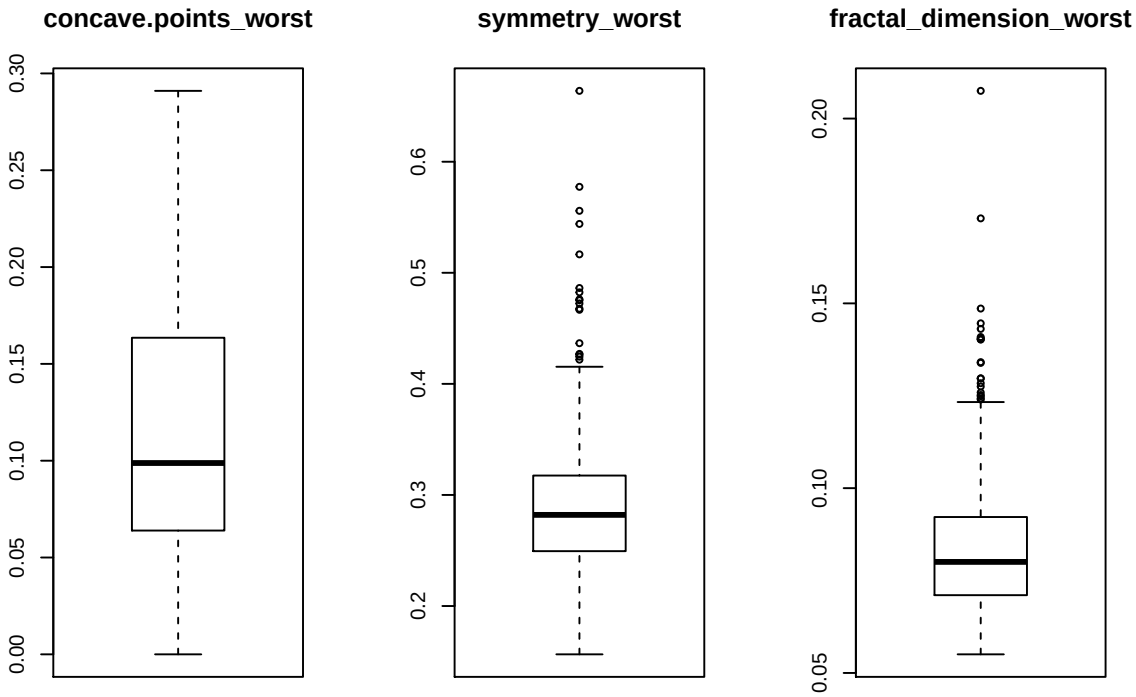
perimeter_worst
50
100
150
200
250

area_worst
1000
2000
3000
4000


smoothness_worst
0.10
0.15
0.20


compactness_worst
0.0
0.2
0.4
0.6
0.8
1.0

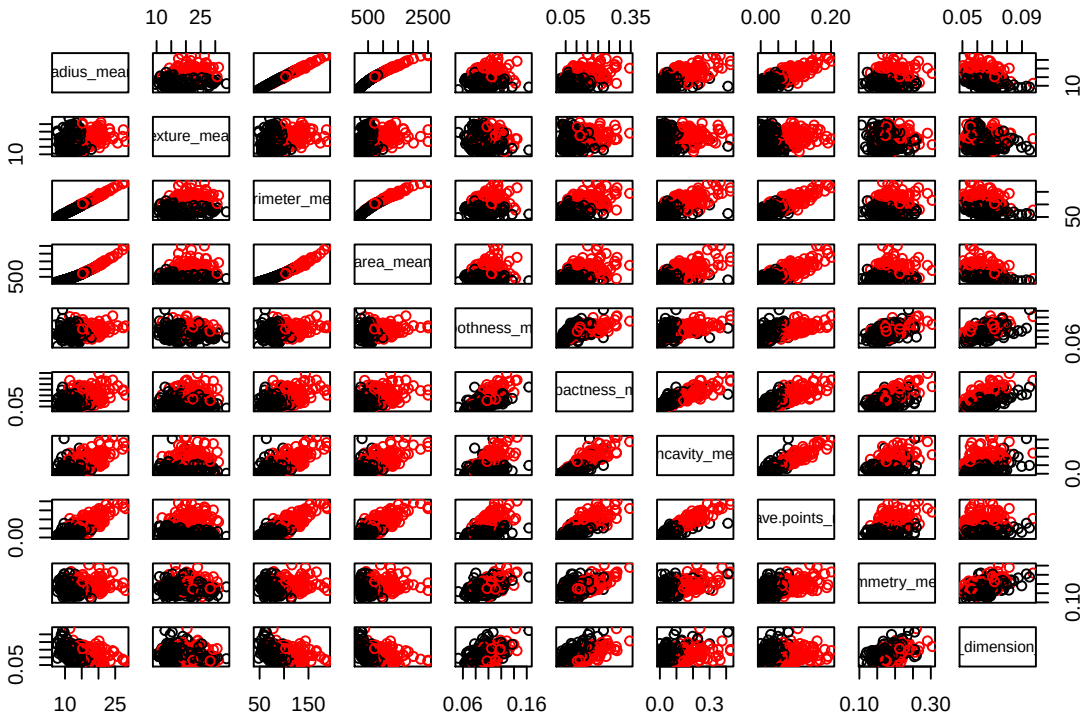

concavity_worst
0.0
0.2
0.4
0.6
0.8
1.0
1.2



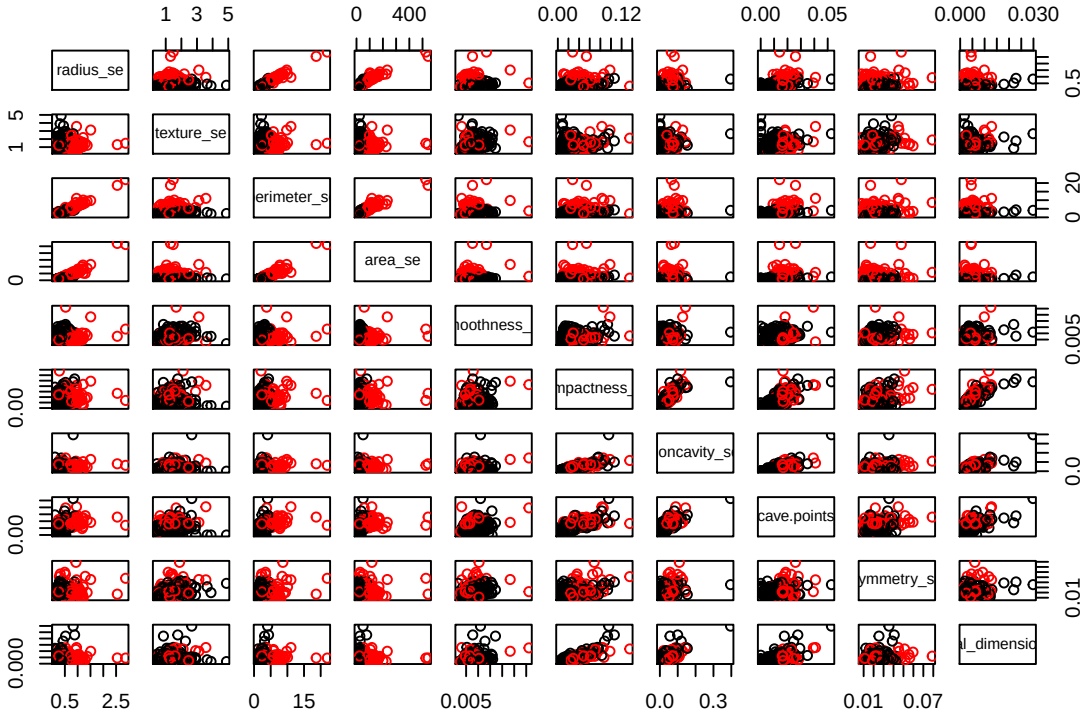
Multimodal Data Visualizations

Next we will see the interactions between the attributes. First with a scatterplot matrix of the attributes colored by the 'diagnosis' values.

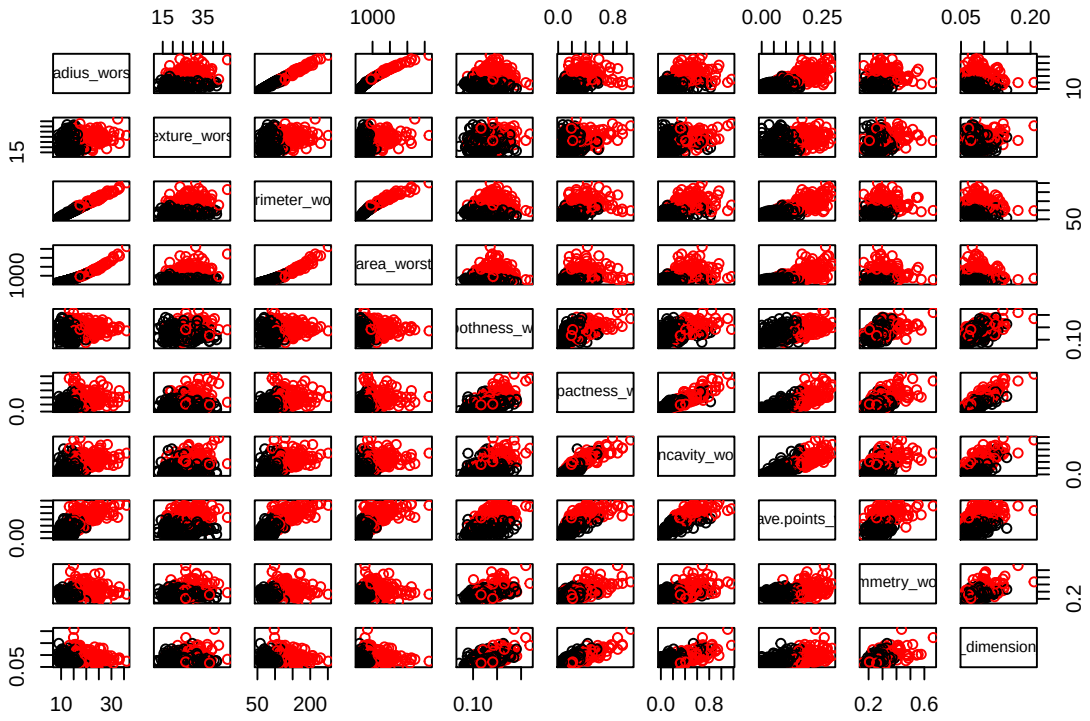
```
# scatterplot matrix

# mean
pairs(dataset[,2:11],
      names(dataset[,2:11]),
      col=dataset$diagnosis)
```




```
# Standard error
pairs(dataset[,12:21],
      names(dataset[,12:21]),
      col=dataset$diagnosis)
```



```
# Worst
pairs(dataset[,22:ncol(dataset)],
      names(dataset[,22:ncol(dataset)]),
      col=dataset$diagnosis)
```



We can see that black (benign) clusters around the lower left corner (smaller values) and red (malignant) is everywhere.

Evaluating models

As we do not know which algorithms will work well in this data, we will try several models and check which ones are the best.

We will define how we will do the tests, we will use cross validation 10 times with 3 repetitions. Since it is a binary diagnosisification problem, we will use 'Accuracy' and 'Kappa' metrics.

We define this function as follows:

```
# 10-fold cross validation with 3 repeats

trainControl <- trainControl(method="repeatedcv",
                             number=10,
                             repeats=3)
metric <- "Accuracy"
```

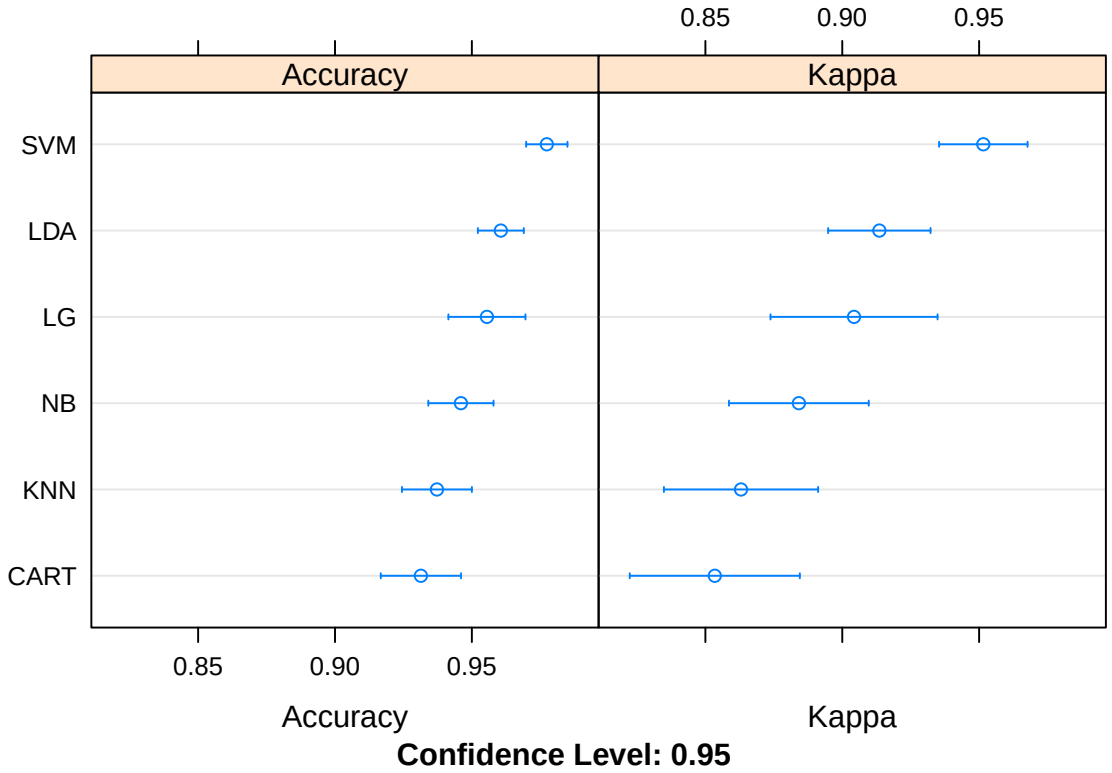
Now we will create the models that we are going to evaluate, we will use the default parameters and then we will consider the respective settings. In order for each algorithm to be evaluated in exactly the same data divisions, we need to reset the seed of random numbers before training the models.

```
# Generalized Linear Model
set.seed(1)
fit.glm <- dataset %>%
  train(diagnosis~.,
        data=.,
```

```
        method="glm",
        metric=metric,
        trControl=trainControl)

# SVM - Support Vector Machines
set.seed(1)
fit.svm <- dataset %>%
  train(diagnosis~.,
        data=.,
        method="svmRadial",
        metric=metric,
        trControl=trainControl)

# Linear Discriminant Analysis
set.seed(1)
fit.lda <- dataset %>%
  train(diagnosis~.,
        data=.,
        method="lda",
        metric=metric,
        trControl=trainControl)

# Naive Bayes
set.seed(1)
fit.nb <- dataset %>%
  train(diagnosis~.,
        data=.,
        method="nb",
        metric=metric,
        trControl=trainControl)

# K-nearest Neighbors
set.seed(1)
fit.knn <- dataset %>%
  train(diagnosis~.,
        data=.,
        method="knn",
        metric=metric,
        trControl=trainControl)

# CART - diagnosisification and Regression Trees
set.seed(1)
fit.cart <- dataset %>%
  train(diagnosis~.,
        data=.,
        method="rpart",
        metric=metric,
        trControl=trainControl)

# Comparing Models
results <- resamples(list(LG=fit.glm,
                          SVM=fit.svm,
                          LDA=fit.lda,
                          NB=fit.nb,
                          KNN=fit.knn,
                          CART=fit.cart))
summary(results)
```

```
Call:
summary.resamples(object = results)

Models: LG, SVM, LDA, NB, KNN, CART
Number of resamples: 30

Accuracy
          Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
LG   0.8478261 0.9336957 0.9560386 0.9555072 0.9782609    1    0
SVM  0.9347826 0.9565217 0.9777778 0.9773913 1.0000000    1    0
LDA  0.9130435 0.9555556 0.9565217 0.9605958 0.9781401    1    0
NB   0.8888889 0.9347826 0.9555556 0.9460064 0.9565217    1    0
KNN  0.8695652 0.9130435 0.9347826 0.9372464 0.9565217    1    0
CART 0.8444444 0.9115942 0.9347826 0.9313849 0.9565217    1    0

Kappa
          Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
LG   0.6610526 0.8586904 0.9037959 0.9043005 0.9532710    1    0
SVM  0.8583162 0.9066937 0.9530216 0.9515366 1.0000000    1    0
LDA  0.8038380 0.9032258 0.9043659 0.9135503 0.9526237    1    0
NB   0.7551687 0.8569808 0.9043659 0.8841477 0.9089109    1    0
KNN  0.7057569 0.8087318 0.8583162 0.8630111 0.9054622    1    0
CART 0.6728972 0.8119920 0.8583162 0.8534308 0.9066937    1    0
```

```
dotplot(results)
```



We can see that SVM (97.73%) and LDA (96.05%) had the highest 'Accuracy'. In the Kappa score also SVM (95.15%) was the best, followed by LG (91.35%).

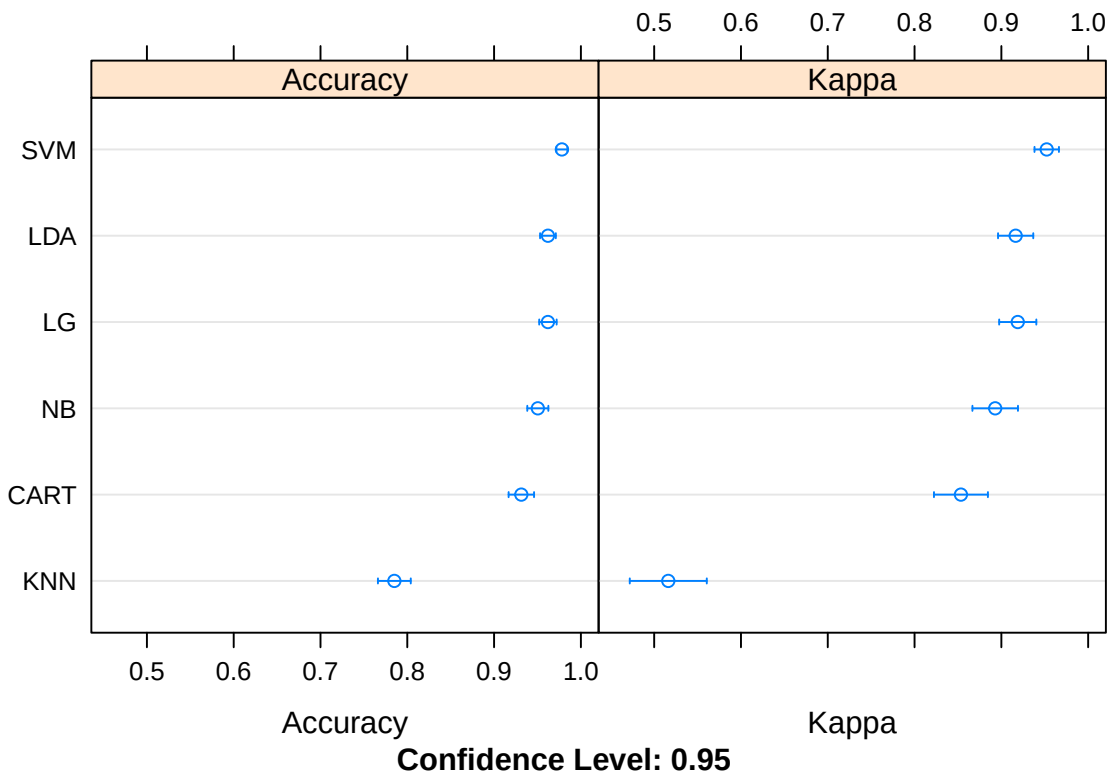
It is very possible that we have some skewed distributions, we will use the Box-Cox method to adjust and normalize these distributions, the data will be transformed using a Box-Cox power transformation to flatten the distributions.

```
# 10-fold cross validation with 3 repeats
trainControl <- trainControl(method="repeatedcv",
                             number=10,
                             repeats=3)
metric <- "Accuracy"

# LG
set.seed(1)
fit.glm <- dataset %>%
  train(diagnosis~.,
        data = .,
        method = "glm",
        metric = metric,
        preProc = c("BoxCox"),
        trControl = trainControl)

# SVM
set.seed(1)
fit.svm <- dataset %>%
  train(diagnosis~.,
        data=.,
        method="svmRadial",
        metric=metric,
        preProc=c("BoxCox"), trControl=trainControl)

# LDA
set.seed(1)
fit.lda <- dataset %>%
  train(diagnosis~.,
        data=.,
        method="lda",
        metric=metric,
        preProc=c("BoxCox"),
        trControl=trainControl)

# Naive Bayes
set.seed(1)
fit.nb <- dataset %>%
  train(diagnosis~.,
        data=.,
        method="nb",
        metric=metric,
        preProc=c("BoxCox"),
        trControl=trainControl)

# KNN
set.seed(1)
fit.knn <- dataset %>%
  train(diagnosis~.,
        data=.,
        method="knn",
        metric=metric,
        preProc=c("BoxCox"),
        trControl=trainControl)

# CART
set.seed(1)
fit.cart <- dataset %>%
```

```
  train(diagnosis~.,
        data=.,
        method="rpart",
        metric=metric,
        preProc=c("BoxCox"), trControl=trainControl)

# Compare algorithms
transformResults <- resamples(list(LG=fit.glm,
                                   SVM=fit.svm,
                                   LDA=fit.lda,
                                   NB=fit.nb,
                                   KNN=fit.knn,
                                   CART=fit.cart))
summary(transformResults)
```

```
Call:
summary.resamples(object = transformResults)

Models: LG, SVM, LDA, NB, KNN, CART
Number of resamples: 30

Accuracy
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
LG   0.8695652 0.9555556 0.9565217 0.9620290 0.9781401 1.0000000    0
SVM  0.9555556 0.9565217 0.9780193 0.9781320 1.0000000 1.0000000    0
LDA  0.9130435 0.9555556 0.9565217 0.9620612 0.9782609 1.0000000    0
NB   0.8888889 0.9347826 0.9555556 0.9503865 0.9782609 1.0000000    0
KNN  0.6739130 0.7568841 0.7826087 0.7850725 0.8177536 0.8913043    0
CART 0.8444444 0.9115942 0.9347826 0.9313849 0.9565217 1.0000000    0

Kappa
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
LG   0.7200811 0.9043659 0.9082542 0.9188865 0.9531463 1.0000000    0
SVM  0.9032258 0.9043659 0.9524753 0.9522619 1.0000000 1.0000000    0
LDA  0.8038380 0.9032258 0.9043659 0.9164779 0.9527721 1.0000000    0
NB   0.7551687 0.8583162 0.9037959 0.8928990 0.9527721 1.0000000    0
KNN  0.2350333 0.4206939 0.5276490 0.5162435 0.5876613 0.7578947    0
CART 0.6728972 0.8119920 0.8583162 0.8534308 0.9066937 1.0000000    0
```

```
dotplot(transformResults)
```



We continue to see how the SVM (97.81%) and LDA (95.22%) models still have the best 'Accuracy', similarly in the 'Kappa', SVM (95.15%) and LDA (91.35%) are the best.

Model Tuning

Now, knowing that SVM is the best results in our tests, we will make some adjustments to try to improve accuracy. Note: LDA has no parameters to adjust

Tuning SVM

We see which parameters adjust

```
modelLookup("svmRadial")
```

```
      model parameter label forReg forClass probModel
1 svmRadial     sigma Sigma   TRUE     TRUE      TRUE
2 svmRadial         C  Cost   TRUE     TRUE      TRUE
```

We can adjust 'sigma' and 'C'

The SVM model in the 'caret' package has two parameters that can be adjusted, these are: 1: sigma which is a smoothing term and 2: C which is a cost constraint. We will test for C a range of values between 1 and 15. For sigma small values, approximately 0.1.

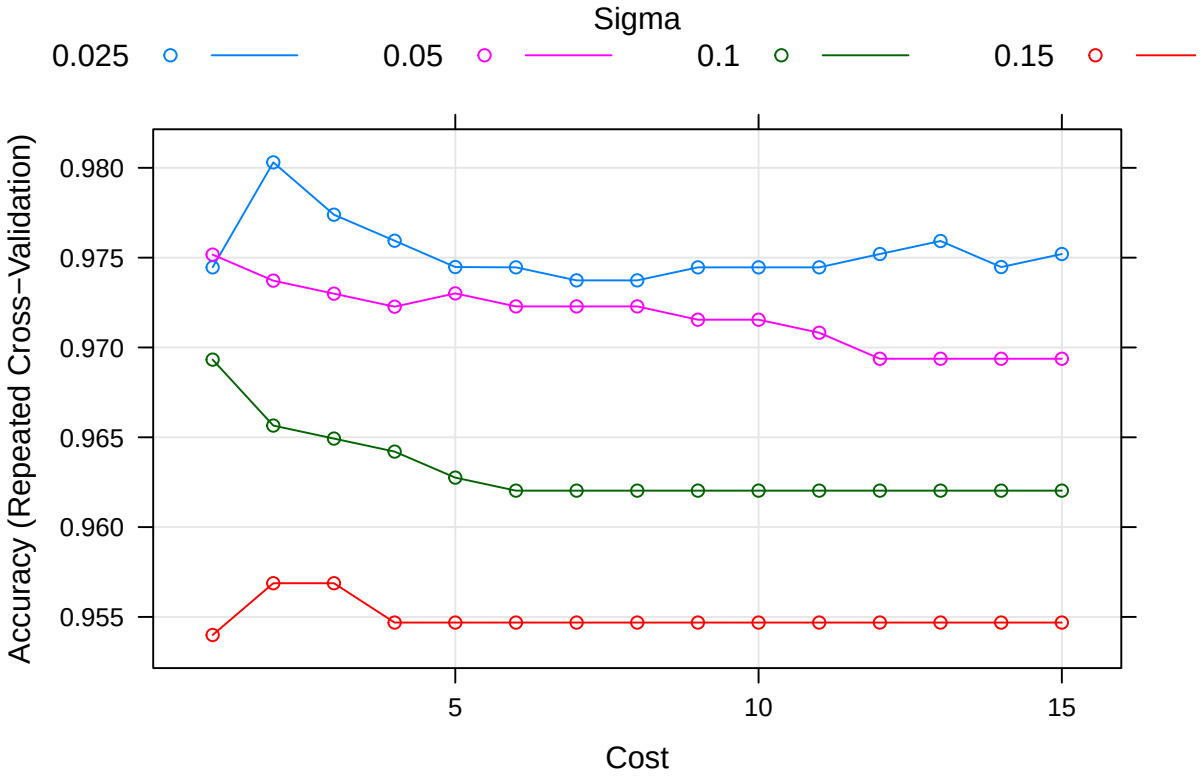
```
# 10-fold cross validation with 3 repeats

trainControl <- trainControl(method = "repeatedcv",
                             number=10,
                             repeats=3)
metric <- "Accuracy"

set.seed(1)

grid <- expand.grid(.sigma = c(0.025, 0.05, 0.1, 0.15),
```

```
                    .C = seq(1, 15, by=1))

fit.svm <- dataset %>%
  train(diagnosis~.,
        data = .,
        method = "svmRadial",
        metric = metric,
        tuneGrid = grid,
        preProc = c("BoxCox"),
        trControl = trainControl)

print(fit.svm)
```

```
Support Vector Machines with Radial Basis Function Kernel

456 samples
 30 predictor
  2 classes: 'B', 'M'

Pre-processing: Box-Cox transformation (24)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 410, 410, 411, 410, 410, 411, ...
Resampling results across tuning parameters:

  sigma  C   Accuracy   Kappa
  0.025   1  0.9744605  0.9448201
  0.025   2  0.9803060  0.9575488
  0.025   3  0.9773913  0.9513066
  0.025   4  0.9759420  0.9482329
  0.025   5  0.9744767  0.9451033
  0.025   6  0.9744605  0.9450812
  0.025   7  0.9737359  0.9435069
  0.025   8  0.9737359  0.9435429
  0.025   9  0.9744605  0.9450428
  0.025  10  0.9744605  0.9450428
  0.025  11  0.9744605  0.9449716
  0.025  12  0.9752013  0.9465658
  0.025  13  0.9759259  0.9481022
  0.025  14  0.9744767  0.9450632
  0.025  15  0.9752013  0.9466370
  0.050   1  0.9751691  0.9463786
  0.050   2  0.9737198  0.9432692
  0.050   3  0.9729952  0.9415004
  0.050   4  0.9722705  0.9399654
  0.050   5  0.9730113  0.9415587
  0.050   6  0.9722866  0.9400940
  0.050   7  0.9722866  0.9400940
  0.050   8  0.9722866  0.9400940
  0.050   9  0.9715459  0.9385752
  0.050  10  0.9715459  0.9385752
  0.050  11  0.9708213  0.9370388
  0.050  12  0.9693720  0.9340076
  0.050  13  0.9693720  0.9340076
  0.050  14  0.9693720  0.9340076
  0.050  15  0.9693720  0.9340076
  0.100   1  0.9693237  0.9331108
  0.100   2  0.9656522  0.9254129
  0.100   3  0.9649275  0.9238683
```

```
0.100   4  0.9642029  0.9222940
0.100   5  0.9627536  0.9192619
0.100   6  0.9620290  0.9177269
0.100   7  0.9620290  0.9177269
0.100   8  0.9620290  0.9177269
0.100   9  0.9620290  0.9177269
0.100  10  0.9620290  0.9177269
0.100  11  0.9620290  0.9177269
0.100  12  0.9620290  0.9177269
0.100  13  0.9620290  0.9177269
0.100  14  0.9620290  0.9177269
0.100  15  0.9620290  0.9177269
0.150   1  0.9539936  0.8985295
0.150   2  0.9568760  0.9055657
0.150   3  0.9568760  0.9056396
0.150   4  0.9546860  0.9009771
0.150   5  0.9546860  0.9009771
0.150   6  0.9546860  0.9009771
0.150   7  0.9546860  0.9009771
0.150   8  0.9546860  0.9009771
0.150   9  0.9546860  0.9009771
0.150  10  0.9546860  0.9009771
0.150  11  0.9546860  0.9009771
0.150  12  0.9546860  0.9009771
0.150  13  0.9546860  0.9009771
0.150  14  0.9546860  0.9009771
0.150  15  0.9546860  0.9009771

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.025 and C = 2.
```

```
plot(fit.svm)
```

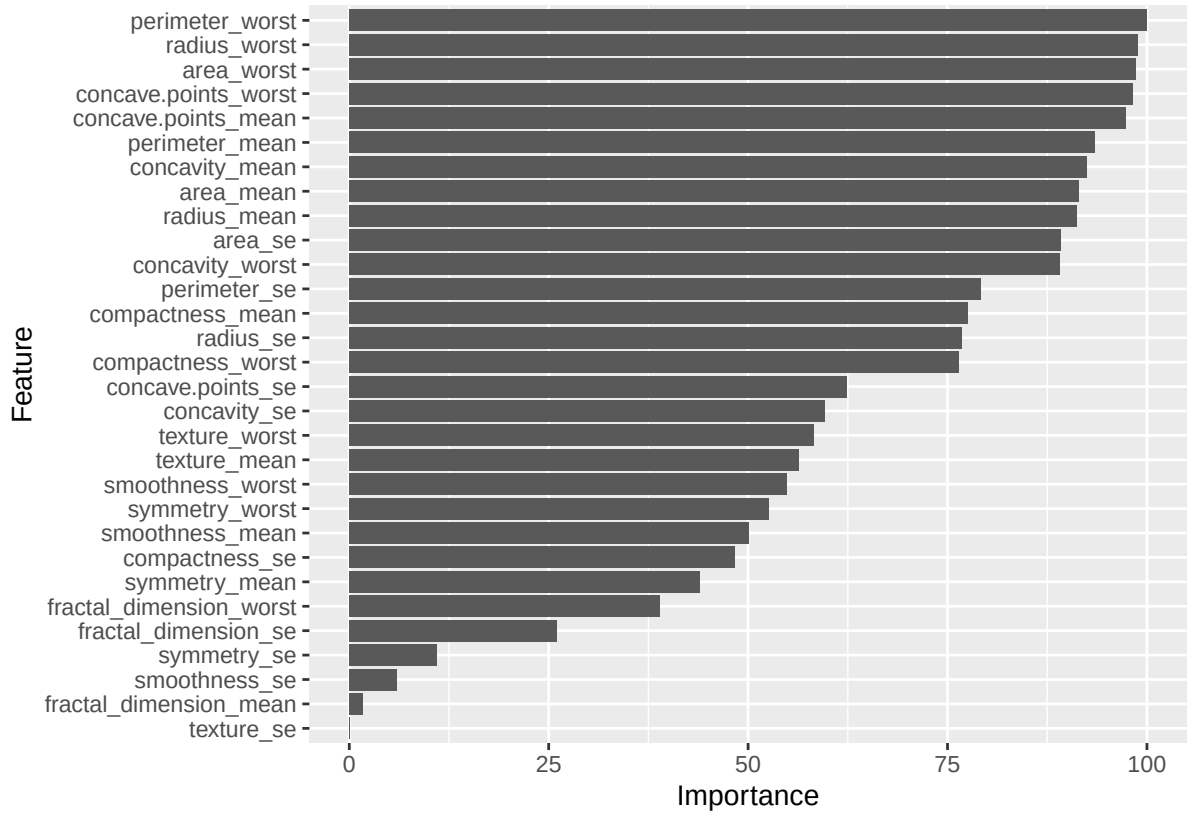


```
fit.svm$bestTune
```

```
  sigma C
2 0.025 2
```

ROC curve variable importance

```
ggplot(varImp(fit.svm))
```




```
varImp(fit.svm)
```

```
ROC curve variable importance

  only 20 most important variables shown (out of 30)

                     Importance
perimeter_worst          100.00
radius_worst              98.80
area_worst                98.65
concave.points_worst      98.20
concave.points_mean       97.33
perimeter_mean            93.44
concavity_mean            92.47
area_mean                 91.43
radius_mean               91.24
area_se                   89.18
concavity_worst           89.13
perimeter_se              79.18
compactness_mean          77.60
radius_se                 76.81
compactness_worst         76.41
concave.points_se         62.31
```

```
concavity_se              59.61
texture_worst             58.28
texture_mean              56.36
smoothness_worst          54.86
```

Best accuracy: sigma: 0.025 C: 1 Accuracy: 0.9838808 Kappa: 0.9651788

We can see that we made an improvement with the adjustments: in the SVM now it is 98.03%. And in the Kappa score it is 95.75%.

Ensemble Methods

A boosting and bagging ensemble algorithm will be applied. Since the decision tree methods underlie these models, it is possible to infer that CART (BAG) and 'Random Forest (RF)' like Bagging and 'Stochastic Gradient Boosting (GBM)' and C5.0 (C50) like Boosting can do an excellent job let's try them and see. We will use the same test data as before, including Box-Cox that flattens the distributions

```
# 10-fold cross validation with 3 repeats

trainControl <-
  trainControl(method = "repeatedcv",
               number = 10,
               repeats = 3)


metric <- "Accuracy"

# Bagged CART
set.seed(1)
fit.treebag <- dataset %>%
  train(diagnosis ~ .,
        data = .,
        method = "treebag",
        metric = metric,
        trControl = trainControl)

# Random Forest
set.seed(1)
fit.rf <- dataset %>%
  train(diagnosis ~ .,
        data = .,
        method = "rf",
        metric = metric,
        preProc = c("BoxCox"),
        trControl = trainControl)

# Stochastic Gradient Boosting
set.seed(1)
fit.gbm <- dataset %>%
  train(diagnosis ~ .,
        data = .,
        method = "gbm",
        metric = metric,
        preProc = c("BoxCox"),
        trControl = trainControl,
        verbose = FALSE)

# C5.0
set.seed(1)
fit.c50 <- dataset %>%
```

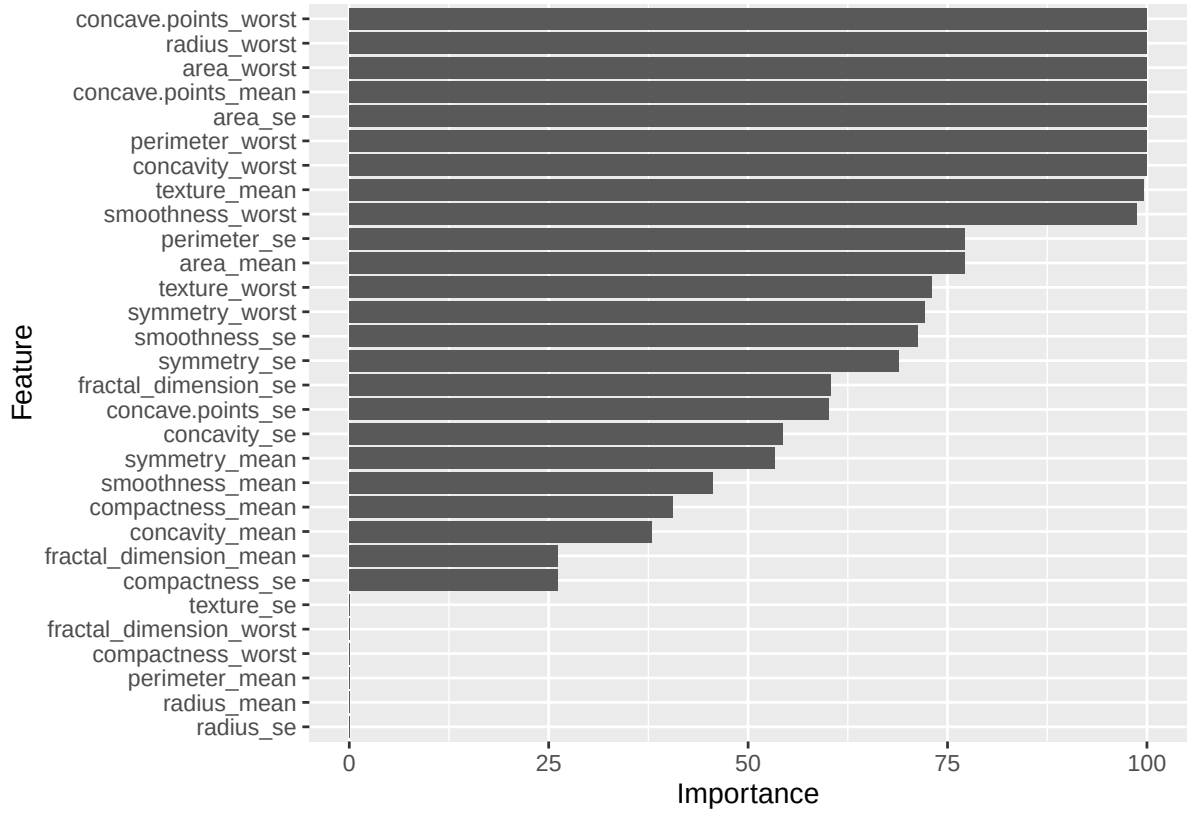
```
  train(diagnosis ~ .,
        data = .,
        method="C5.0",
        metric = metric,
        preProc = c("BoxCox"),
        trControl = trainControl)

# Compare results
ensembleResults <- resamples(list(BAG=fit.treebag,
                                  RF=fit.rf,
                                  GBM=fit.gbm,
                                  C50=fit.c50))
```

ROC curve variable importance

```
varImp(fit.c50)
```

```
C5.0 variable importance

  only 20 most important variables shown (out of 30)

                     Overall
perimeter_worst       100.00
area_se               100.00
area_worst            100.00
concave.points_worst  100.00
concavity_worst       100.00
radius_worst          100.00
concave.points_mean   100.00
texture_mean           99.56
smoothness_worst       98.68
perimeter_se           77.19
area_mean              77.19
texture_worst          73.03
symmetry_worst         72.15
smoothness_se          71.27
symmetry_se            68.86
fractal_dimension_se   60.31
concave.points_se      60.09
concavity_se           54.39
symmetry_mean          53.29
smoothness_mean        45.61
```

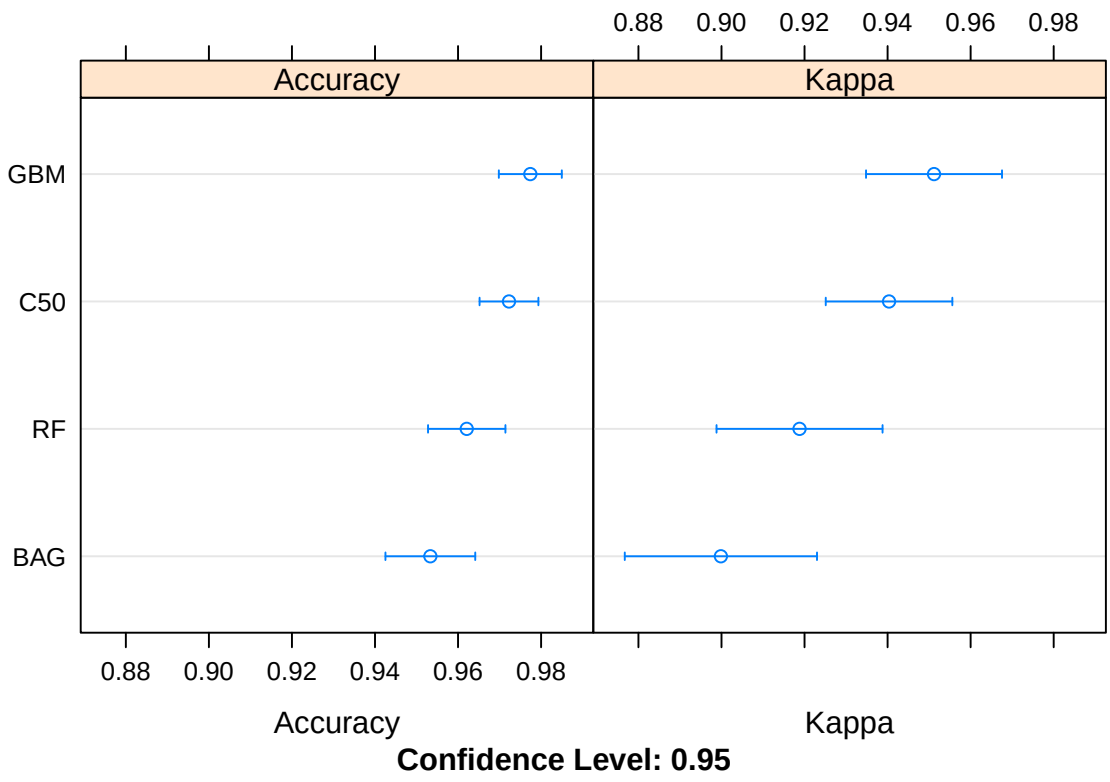
```
ggplot(varImp(fit.c50))
```




```
summary(ensembleResults)
```

```
Call:
summary.resamples(object = ensembleResults)

Models: BAG, RF, GBM, C50
Number of resamples: 30

Accuracy
         Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
BAG 0.8695652 0.9347826 0.9565217 0.9533333 0.9777778    1    0
RF  0.9111111 0.9399758 0.9565217 0.9620934 0.9782609    1    0
GBM 0.9333333 0.9565217 0.9782609 0.9774074 1.0000000    1    0
C50 0.9347826 0.9565217 0.9777778 0.9722705 0.9782609    1    0

Kappa
         Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
BAG 0.7200811 0.8586904 0.9054622 0.8998541 0.9521785    1    0
RF  0.8064516 0.8745344 0.9078023 0.9187975 0.9531463    1    0
GBM 0.8565356 0.9066937 0.9527721 0.9511848 1.0000000    1    0
C50 0.8583162 0.9066937 0.9527721 0.9403526 0.9537486    1    0
```

```
dotplot(ensembleResults)
```



The best Ensemble Methods is GBM with an Accuracy of 97.74% (the best previous model was 98.03% of SVM tuning) and regarding the Kappa score with the GBM model the score is 95.11% and the previous one It was 95.75%%.

Final Model

We will finish making the final model, our model with better accuracy and Kappa score was the SVM, that's why it will be the final model. However, we will also test the C5.0 Ensemble Methods (It also performed well) to compare its execution with the validation set with the other chosen model (SVM).

We are going to capture the parameters of the Box-Cox transformation and prepare the data, previously we eliminate the unused attributes, and we convert all the entries to a numerical format but in the training set, now we have to do the same with the validation portion and remove missing values (Na).

Preparing parameters for data transform

```
set.seed(1)

preprocessParams <- preProcess(dataset,
                               method = c("BoxCox"))

x <- predict(preprocessParams, dataset)
```

Now we will proceed to prepare the validation data set, for this we will eliminate the attributes that we will not use, convert all the input attributes to numerical and finally we will apply the Box-Cox transformation to the input attributes using the parameters prepared in training data set. Also remove Na values.

```
# prepare parameters for data transform
set.seed(1)
datasetNoMissing <- dataset[complete.cases(dataset),]
x <- datasetNoMissing[,2:11]
```

```
preprocessParams <- preProcess(x, method=c("BoxCox"))
x <- predict(preprocessParams, x)

# Preparing the validation dataset ---------------------------------
set.seed(1)

# remove missing values (not allowed in this implementation of knn)
validation <- validation[complete.cases(validation),]

# convert to numeric
for(i in 2:ncol(validation)) {
  validation[,i] <- as.numeric(as.character(validation[,i]))
}

# transform the validation dataset
validationX <- predict(preprocessParams,
                       validation[,2:ncol(validation)])
```

Run final models

First, we test with the C5.0 model

```
library(C50)
set.seed(1)

# Tuning some parameters that previously did not tuning.
C5.0.Grid <-expand.grid(interaction.depth = c(1,5,9),
                        n.trees = (1:30)*50,
                        shrinkage = 0.1,
                        n.minosinnode =20)

# Train model
c5model <- C5.0(x = dataset[,-1],
                y = dataset$diagnosis,
                preProc = c("BoxCox"),
                Grid =C5.0.Grid,
                tuneLength=30,
                trControl = trainControl)

# Model C5.0 predictions with the validation set
predictions <- predict.C5.0(c5model, validationX)

# Confusion Matrix for C5.0 model
confusionMatrix(as.factor(predictions),
                validation$diagnosis)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 68  5
         M  3 37

               Accuracy : 0.9292
                 95% CI : (0.8653, 0.9689)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : 1.372e-13
```

```
                 Kappa : 0.8469

 Mcnemar's Test P-Value : 0.7237

            Sensitivity : 0.9577
            Specificity : 0.8810
         Pos Pred Value : 0.9315
         Neg Pred Value : 0.9250
             Prevalence : 0.6283
         Detection Rate : 0.6018
   Detection Prevalence : 0.6460
      Balanced Accuracy : 0.9193

       'Positive' Class : B
```

Model SVM predictions with validation set

```
# This is previously tuning
set.seed(1)
predictions <- predict(fit.svm, validation[,-1])

# Confusion Matrix for SVM model
confusionMatrix(as.factor(predictions),
                validation$diagnosis)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 69  2
         M  2 40

               Accuracy : 0.9646
                 95% CI : (0.9118, 0.9903)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9242

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9718
            Specificity : 0.9524
         Pos Pred Value : 0.9718
         Neg Pred Value : 0.9524
             Prevalence : 0.6283
         Detection Rate : 0.6106
   Detection Prevalence : 0.6283
      Balanced Accuracy : 0.9621

       'Positive' Class : B
```

4. Results

With the C5.0 in the previous train the score was 97.22%, now with the validation set the Accuracy is 92.92%, much smaller. The kappa score was 84.69%, sensitivity of 95.77% and the specificity of 88.10%.

Our best and final model is SVM, it has 96.46% of Accuracy and 92.42% in kappa score. The Sensitivity is 97.18% and the Specificity is 95.24%

5. Conclusion

In this project, the diagnosis data of Wisconsin Madison breast cancer was studied as a Binary Classification problem.

We investigated several models and the less precise ones were discarded and we advanced with those that best suited our case.

The best model turned out to be SVM, at the start we obtained an Accuracy of 98.03% and a kappa score of 95.75%. When we tested them with the validation set, the result was 96.46% of Accuracy and 92.42% in kappa score, with a Sensitivity of 97.18% and a Specificity of 95.24%, it is a good performance for our final model