Statistical Project A.A. 2022/2023

Alessia d'Addario, Alice Ronzoni

Contents

Problem presentation

In this project we present a binary classification problem based on the "Wisconsin Diagnostic Breast Cancer (WDBC)".
Given a series of observations (patients) and a series of characteristic (features) we were interested in classifying if a given tumor is benign or malignant based on the specific characteristic's values of every patient's diagnosis. Our aim was to identify the informative features and use them in several models to study how well those models could classify the two types of Tumor.

Dataset presentation

The dataset used was created by Dr. William H. Wolberg (General Surgery Dept., University of Wisconsin), Olvi L. Mangasarian (Computer Sciences Dept. University of Wisconsin) and W. Nick Street, (Computer Sciences Dept., University of Wisconsin), who also donated it in 1995. It can be found at the following web page: uci.edu.
The data are obtained from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image by three different parameters for every features:

- Mean
- Standard Error
- Worst

This Dataset contains 569 Instances with 32 features, two of them are informative

1. ID number
2. Diagnosis (M = malignant, B = benign)

and the remained (3-32) are real-valued features computed for each cell nucleus divided into the previous three parameters (Mean, SE, Worst):

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2/(area - 1.0))
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

Clean and Filter Data

As first step we checked for some missing values:

```
sum(is.na(wdbc))
```

```
## [1] 0
```

Since this is not the case, we could proceed with the analysis of the Dataset.
To make sure that we didn't have some redundant rows, we checked for repeating ID in the ID columns.

```
sum(duplicated(wdbc$id))
```

```
## [1] 0
```

There were no multiple observations for the same patient. Given that, we decided to remove the ID columns from the entire dataset as it was not helpful in our analysis.

As further manipulation, we used the *as.factor()* function to change the target column into factors and check all the levels.

```
wdbc_df$diagnosis<-as.factor(wdbc_df$diagnosis)
levels(wdbc_df$diagnosis)
```

```
## [1] "B" "M"
```

We also renamed the columns due to legibility problem:

```
colnames(wdbc) = c('id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
                   'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
                   'concave_pts_mean', 'symmetry_mean', 'fractal_dim_mean','radius_SE',
                   'texture_SE', 'perimeter_SE', 'area_SE', 'smoothness_SE',
                   'compactness_SE', 'concavity_SE', 'concave_pts_SE', 'symmetry_SE',
                   'fractal_dim_SE','radius_worst', 'texture_worst','perimeter_worst',
                   'area_worst', 'smoothness_worst', 'compactness_worst', 'concavity_worst',
                   'concave_pts_worst', 'symmetry_worst', 'fractal_dim_worst')
```

Data exploration

In this section we want to have a graphical representation of how data are distributed.
To start, we have a look at the summary to have a better idea of how data are distributed.

```
summary(wdbc_df)
```

```
##  diagnosis  radius_mean      texture_mean   perimeter_mean     area_mean
##  B:357     Min.   : 6.981   Min.   : 9.71   Min.   : 43.79   Min.   : 143.5
##  M:212     1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17   1st Qu.: 420.3
##            Median :13.370   Median :18.84   Median : 86.24   Median : 551.1
##            Mean   :14.127   Mean   :19.29   Mean   : 91.97   Mean   : 654.9
##            3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10   3rd Qu.: 782.7
##            Max.   :28.110   Max.   :39.28   Max.   :188.50   Max.   :2501.0
##  smoothness_mean   compactness_mean  concavity_mean    concave_pts_mean
##  Min.   :0.05263   Min.   :0.01938   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956   1st Qu.:0.02031
##  Median :0.09587   Median :0.09263   Median :0.06154   Median :0.03350
##  Mean   :0.09636   Mean   :0.10434   Mean   :0.08880   Mean   :0.04892
##  3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070   3rd Qu.:0.07400
##  Max.   :0.16340   Max.   :0.34540   Max.   :0.42680   Max.   :0.20120
##  symmetry_mean    fractal_dim_mean    radius_SE        texture_SE
##  Min.   :0.1060   Min.   :0.04996   Min.   :0.1115   Min.   :0.3602
##  1st Qu.:0.1619   1st Qu.:0.05770   1st Qu.:0.2324   1st Qu.:0.8339
##  Median :0.1792   Median :0.06154   Median :0.3242   Median :1.1080
##  Mean   :0.1812   Mean   :0.06280   Mean   :0.4052   Mean   :1.2169
##  3rd Qu.:0.1957   3rd Qu.:0.06612   3rd Qu.:0.4789   3rd Qu.:1.4740
##  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.: 1.606   1st Qu.: 17.850   1st Qu.:0.005169   1st Qu.:0.013080
##  Median : 2.287   Median : 24.530   Median :0.006380   Median :0.020450
##  Mean   : 2.866   Mean   : 40.337   Mean   :0.007041   Mean   :0.025478
##  3rd Qu.: 3.357   3rd Qu.: 45.190   3rd Qu.:0.008146   3rd Qu.:0.032450
##  Max.   :21.980   Max.   :542.200   Max.   :0.031130   Max.   :0.135400
##   concavity_SE     concave_pts_SE      symmetry_SE       fractal_dim_SE
##  Min.   :0.00000   Min.   :0.000000   Min.   :0.007882   Min.   :0.0008948
```

```
##  1st Qu.:0.01509   1st Qu.:0.007638   1st Qu.:0.015160   1st Qu.:0.0022480
##  Median :0.02589   Median :0.010930   Median :0.018730   Median :0.0031870
##  Mean   :0.03189   Mean   :0.011796   Mean   :0.020542   Mean   :0.0037949
##  3rd Qu.:0.04205   3rd Qu.:0.014710   3rd Qu.:0.023480   3rd Qu.:0.0045580
##  Max.   :0.39600   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.:21.08   1st Qu.: 84.11   1st Qu.: 515.3
##  Median :14.97   Median :25.41   Median : 97.66   Median : 686.5
##  Mean   :16.27   Mean   :25.68   Mean   :107.26   Mean   : 880.6
##  3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40   3rd Qu.:1084.0
##  Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0
##  smoothness_worst  compactness_worst concavity_worst  concave_pts_worst
##  Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145   1st Qu.:0.06493
##  Median :0.13130   Median :0.21190   Median :0.2267   Median :0.09993
##  Mean   :0.13237   Mean   :0.25427   Mean   :0.2722   Mean   :0.11461
##  3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829   3rd Qu.:0.16140
##  Max.   :0.22260   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100
##  symmetry_worst   fractal_dim_worst
##  Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2822   Median :0.08004
##  Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :0.6638   Max.   :0.20750
```

To have a graphical representation, we decided to plot them starting from the column diagnosis.

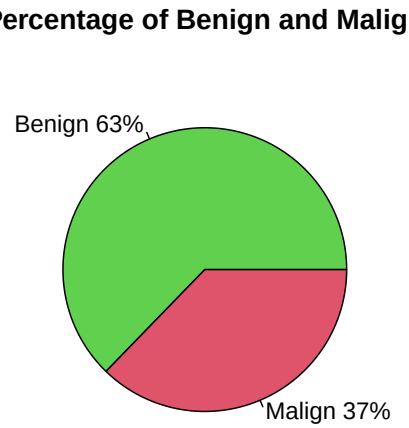
```
table(wdbc_df$diagnosis) # Numbers of 0 and 1 (M, B)
```

```
##
##   B   M
## 357 212
```

```
table(wdbc_df$diagnosis)/length(wdbc_df$diagnosis) # proportion of 0 and 1 (M, B)
```

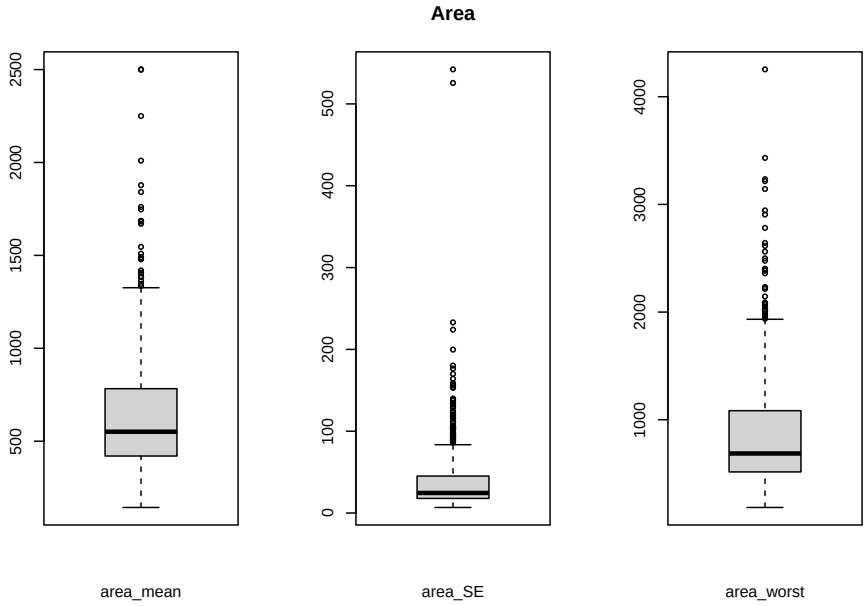
```
##
##         B         M
## 0.6274165 0.3725835
```

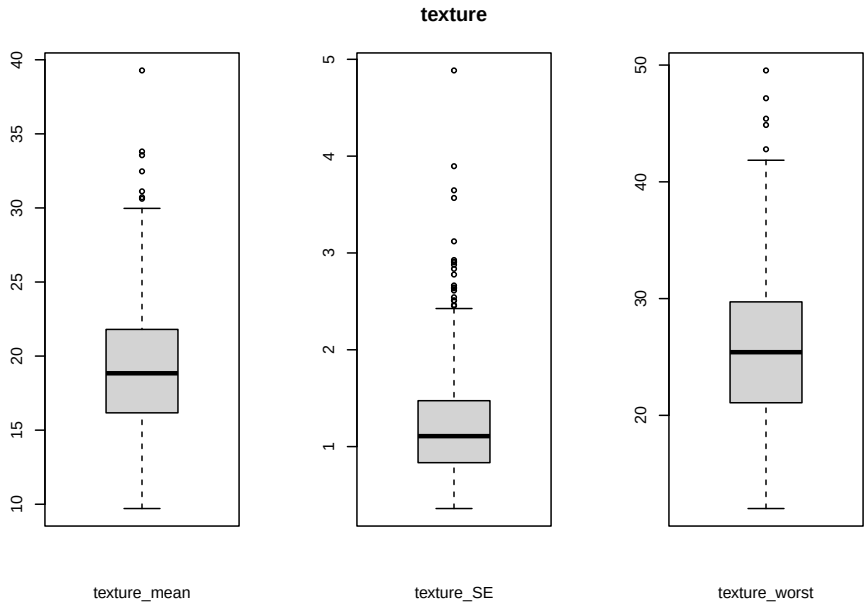
We can see that there are 357 (62,7%) Benign cases and 212 (37,2%) Malign cases.
To have a more clear understanding, we plotted them into a Pie Chart:



Data Representation

We now want to see the distribution of our data looking at the boxplot and at the density plot.
In order to have a complete display, we decided to have 10 different frame, one for each feature and to show the three different parameters adjusted in their specific scale. Here, for simplicity, we report just two of them.



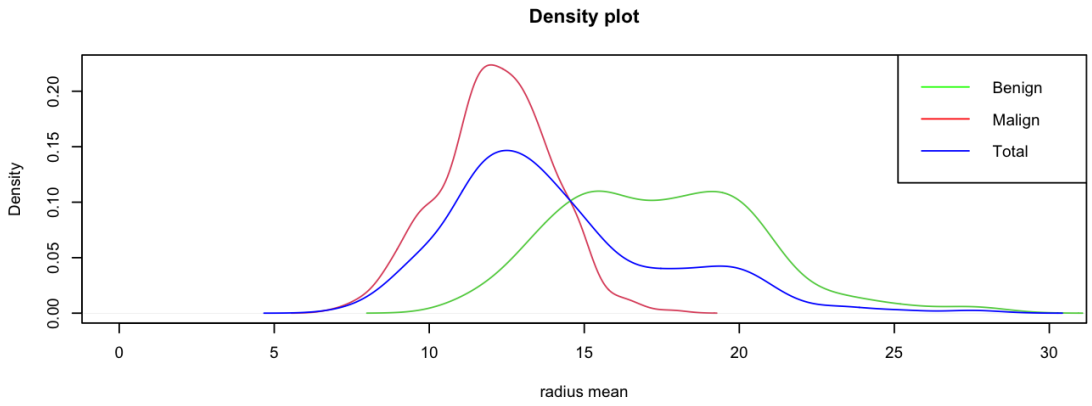


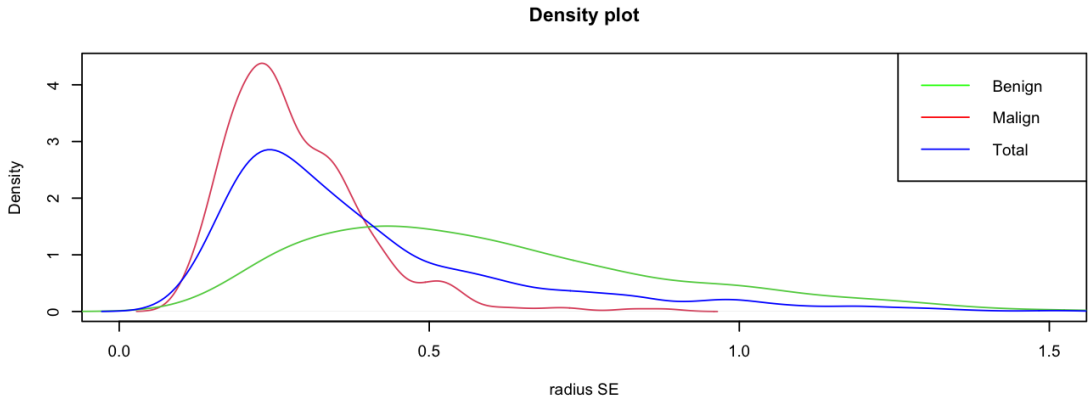
It is evident that our data are really skewed. At the same time we have to remember that these data are the result of measurements done on cells that can have different shapes and sizes. What's more, the dataset doesn't provide all the measurements, but only the mean, the standard error and the worst value.
Before further analysis we needed to check the normality of our feature, which is a needed assumption in most of the classification model. To do so, we first plotted the density of our features.
Here we report a small sample of them:

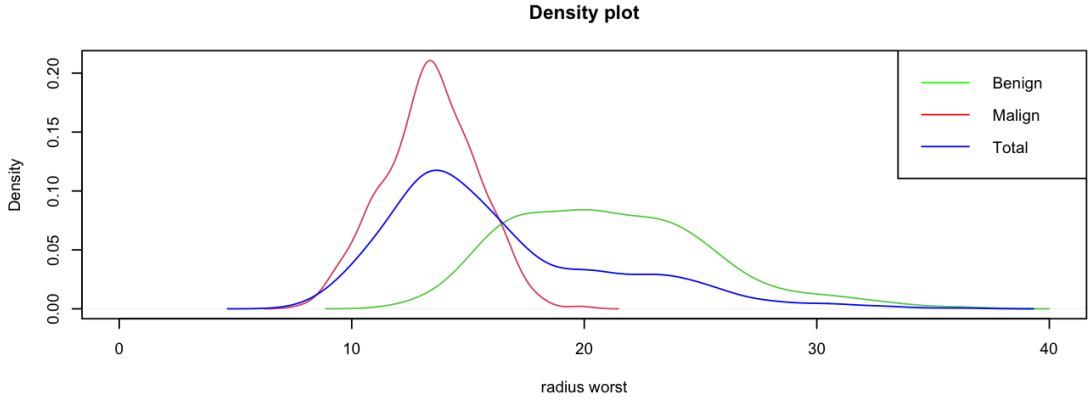
```
wdbc_as_matrix <- as.matrix(wdbc[-c(1, 2)])
wdbc_as_matrix <- as.matrix(wdbc_as_matrix)

wdbc_B <- wdbc_as_matrix[wdbc[,'diagnosis']=="B",]
wdbc_M <- wdbc_as_matrix[wdbc[,'diagnosis']=="M",]
wdbc_B <- as.data.frame(wdbc_B)
wdbc_M <- as.data.frame(wdbc_M)
```

```
par(mfrow=c(3, 1))
plot(density(wdbc_B$radius_mean), col = 2, main ="Density plot", xlab = "radius mean" ,
     xlim=c(0,30))
lines(density(wdbc_M$radius_mean), col = 3)
lines(density(wdbc$radius_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$radius_SE), col = 2, main ="Density plot",
     xlab = "radius SE",xlim=c(0,1.5))
lines(density(wdbc_M$radius_SE), col = 3)
lines(density(wdbc$radius_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
plot(density(wdbc_B$radius_worst), col = 2, main ="Density plot", xlab = "radius worst",
     xlim=c(0,40))
lines(density(wdbc_M$radius_worst), col = 3)
lines(density(wdbc$radius_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
```

```
       col = c("green", "red","blue"), lwd = 1)
```

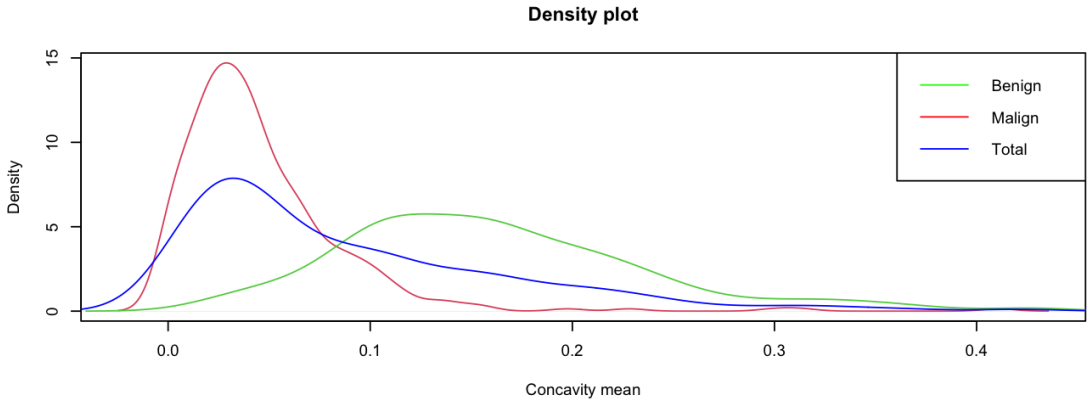


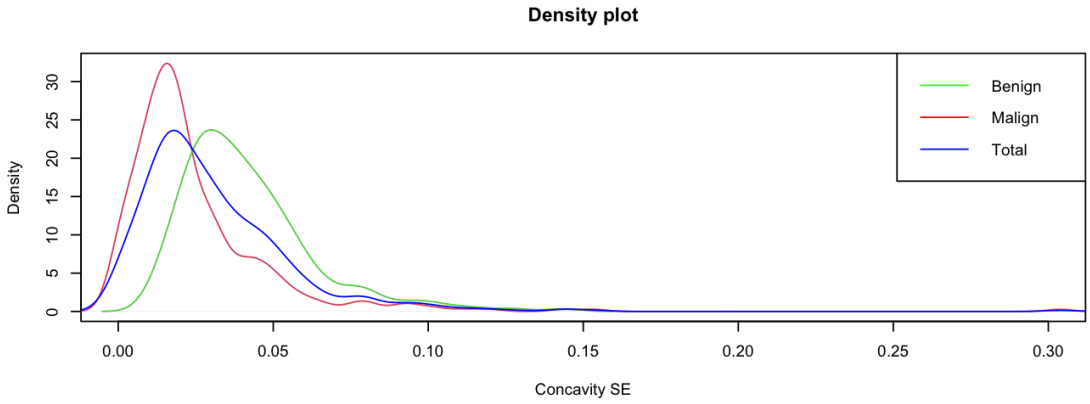


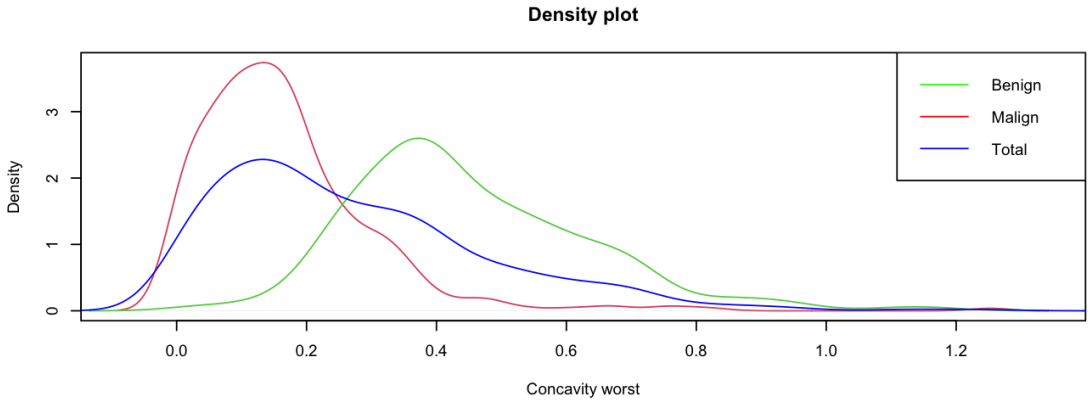



```
# Density plot of concave_pts
plot(density(wdbc_B$concave_pts_mean), col = 2, main ="Density plot",
     xlab = "concave_pts mean", ylim = c(0, 0.12))
lines(density(wdbc_M$concave_pts_mean), col = 3)
lines(density(wdbc$concave_pts_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
plot(density(wdbc_B$concave_pts_SE), col = 2, main ="Density plot",
     xlab = "concave_pts SE", ylim = c(0, 1.2))
```

```
lines(density(wdbc_M$concave_pts_SE), col = 3)
lines(density(wdbc$concave_pts_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
plot(density(wdbc_B$concave_pts_worst), col = 2, main ="Density plot",
     xlab = "concave_pts worst", ylim = c(0, 0.12))
lines(density(wdbc_M$concave_pts_worst), col = 3)
lines(density(wdbc$concave_pts_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
par(mfrow=c(1, 1))
```

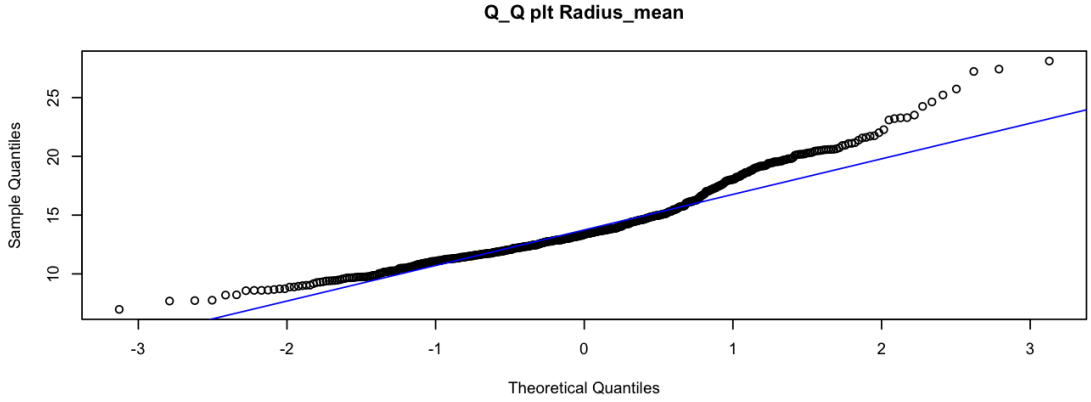


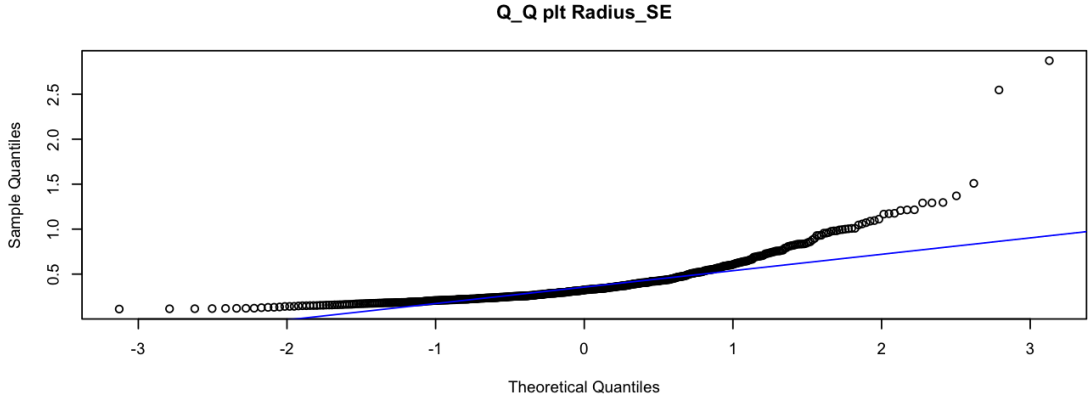


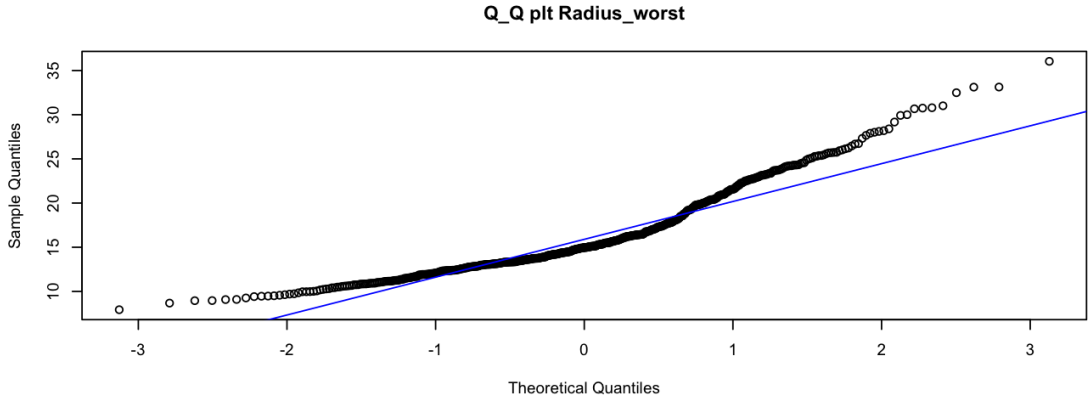


The plot suggest a bimodal tendency for what concerns the *radius_mean* and the *radius_worst* and some irregularities in the shape of the *concave_pts* plots.
Those observation suggest that they may not be normal distributed.
To assess this hypothesis we looked at the respective qqplot:

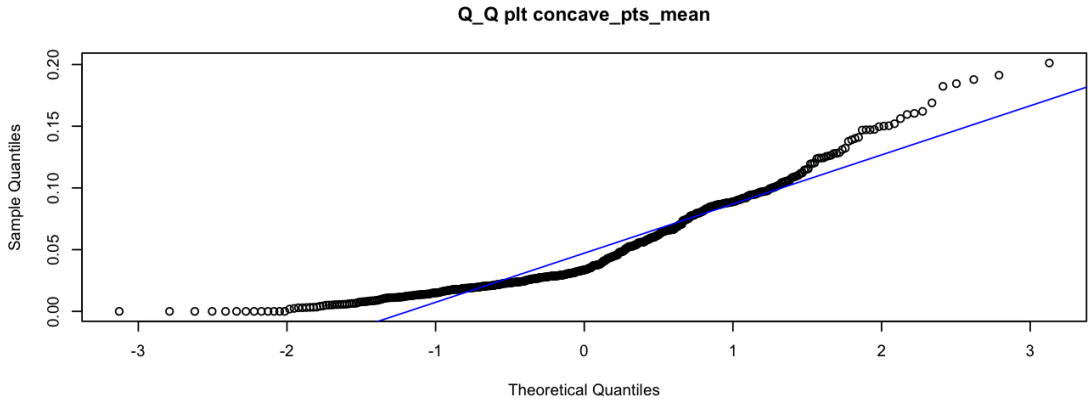
```
par(mfrow=c(3, 1))
### qqplot Radius
qqnorm(wdbc$radius_mean,main="Q_Q plt Radius_mean")
qqline(wdbc$radius_mean, col="blue")
qqnorm(wdbc$radius_SE,main="Q_Q plt Radius_SE")
qqline(wdbc$radius_SE, col="blue")
qqnorm(wdbc$radius_worst,main="Q_Q plt Radius_worst")
qqline(wdbc$radius_worst, col="blue")
```

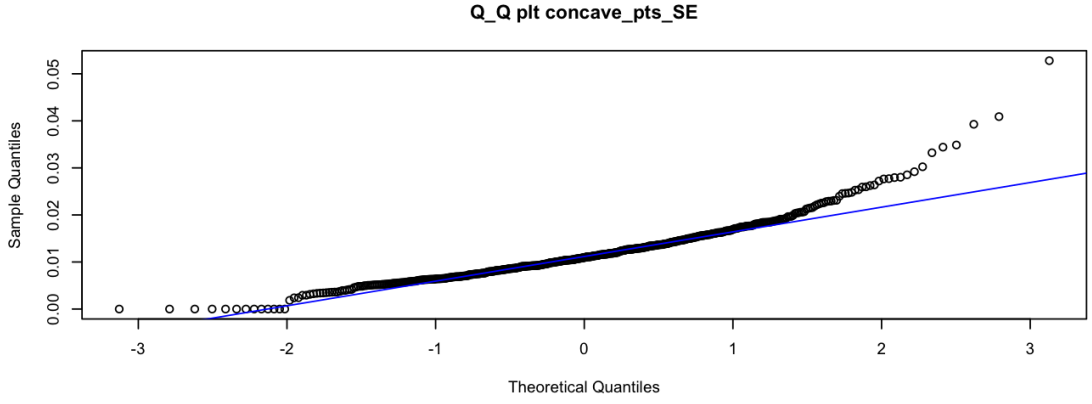


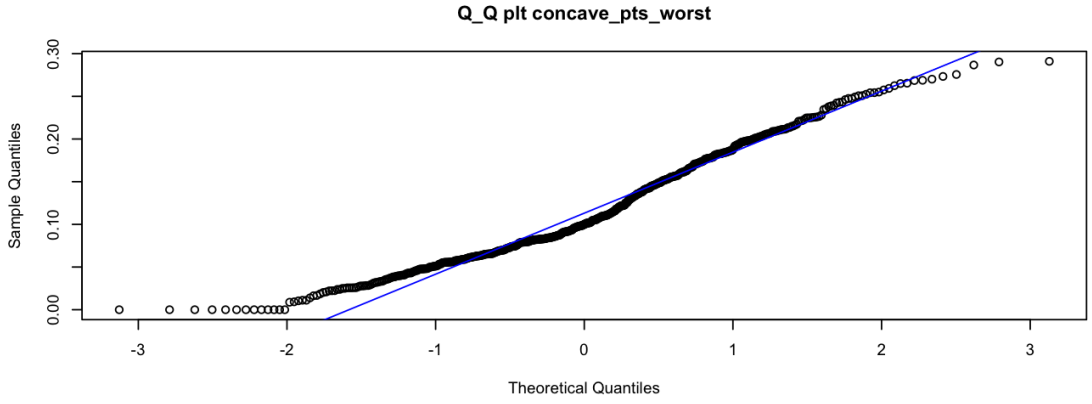





```
### qqplot concave_pts
qqnorm(wdbc$concave_pts_mean,main="Q_Q plt concave_pts_mean")
qqline(wdbc$concave_pts_mean, col="blue")
qqnorm(wdbc$concave_pts_SE,main="Q_Q plt concave_pts_SE")
qqline(wdbc$concave_pts_SE, col="blue")
qqnorm(wdbc$concave_pts_worst,main="Q_Q plt concave_pts_worst")
qqline(wdbc$concave_pts_worst, col="blue")



par(mfrow=c(1, 1))
```







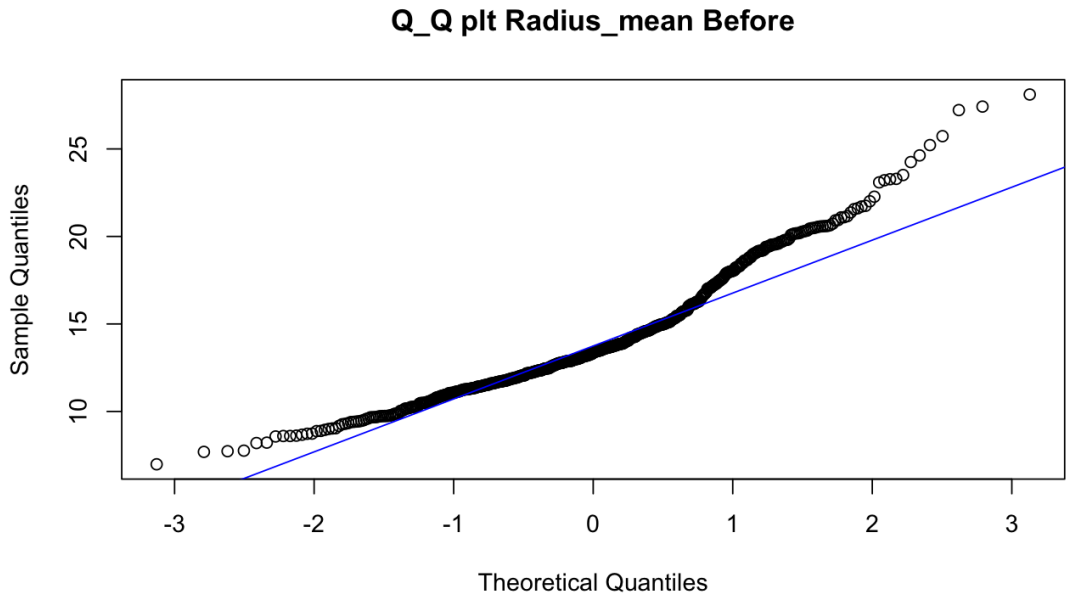
As supposed, the features lack of normality behavior.

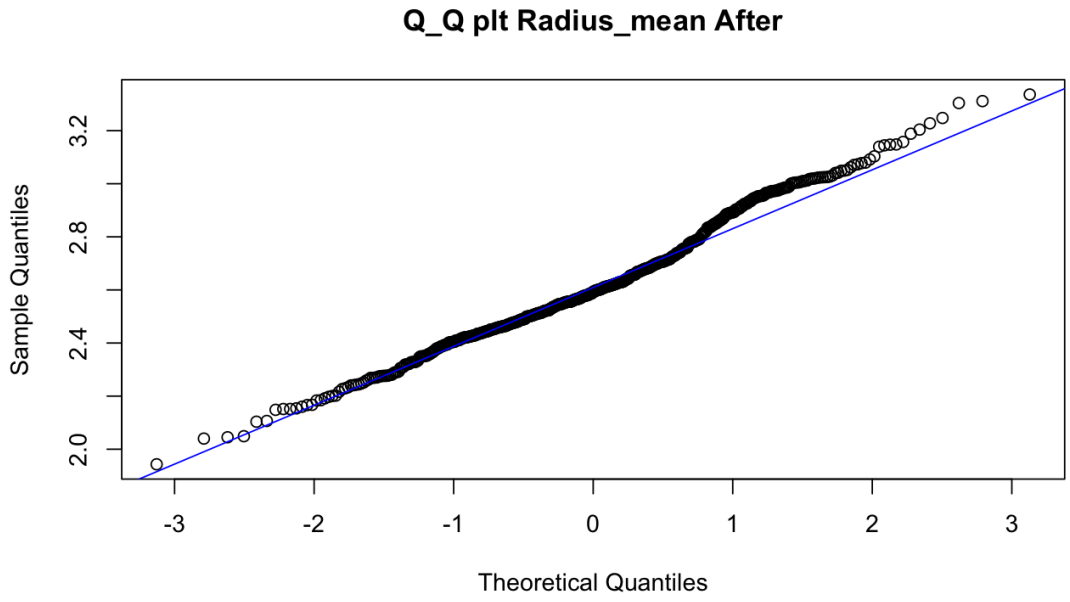
For this reason, before proceeding with the analysis, we decided to normalize them using the logarithm function.

Here are some example of feature where can be seen how the normalization impacted on the data:

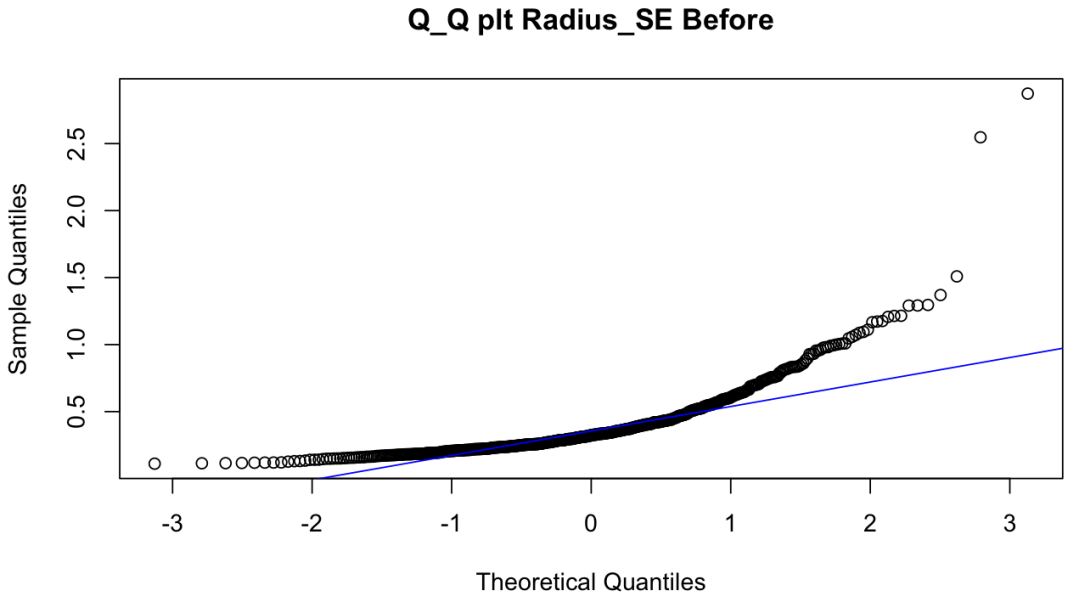
```
par(mfrow=c(2, 1))

#before
qqnorm(wdbc$radius_mean,main="Q_Q plt Radius_mean Before")
qqline(wdbc$radius_mean, col="blue")
#after
```

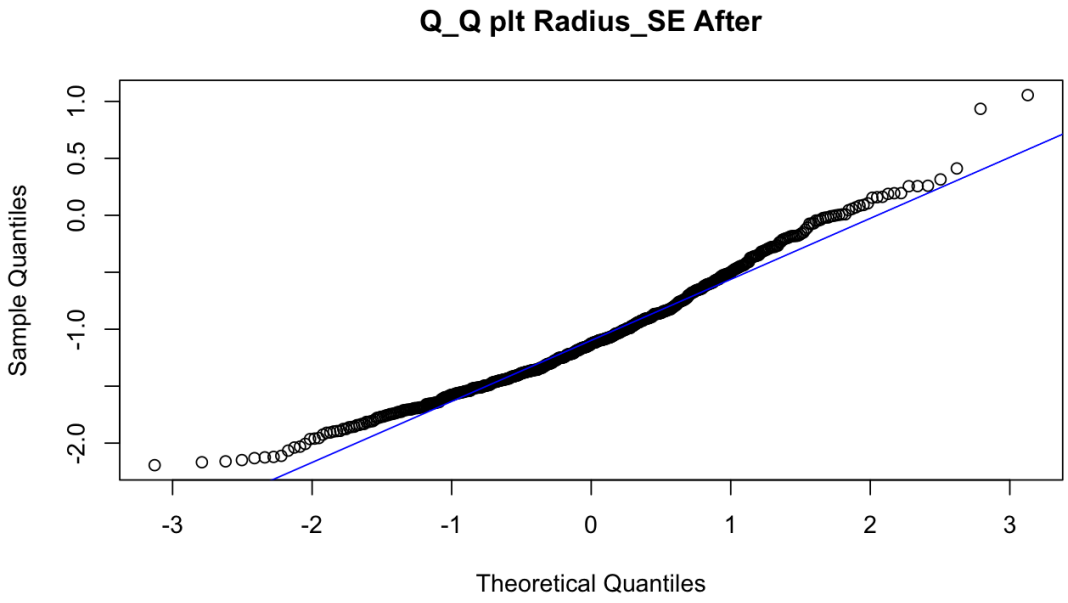
```
qqnorm(wdbc_df$radius_mean,main="Q_Q plt Radius_mean After")
qqline(wdbc_df$radius_mean, col="blue")
```



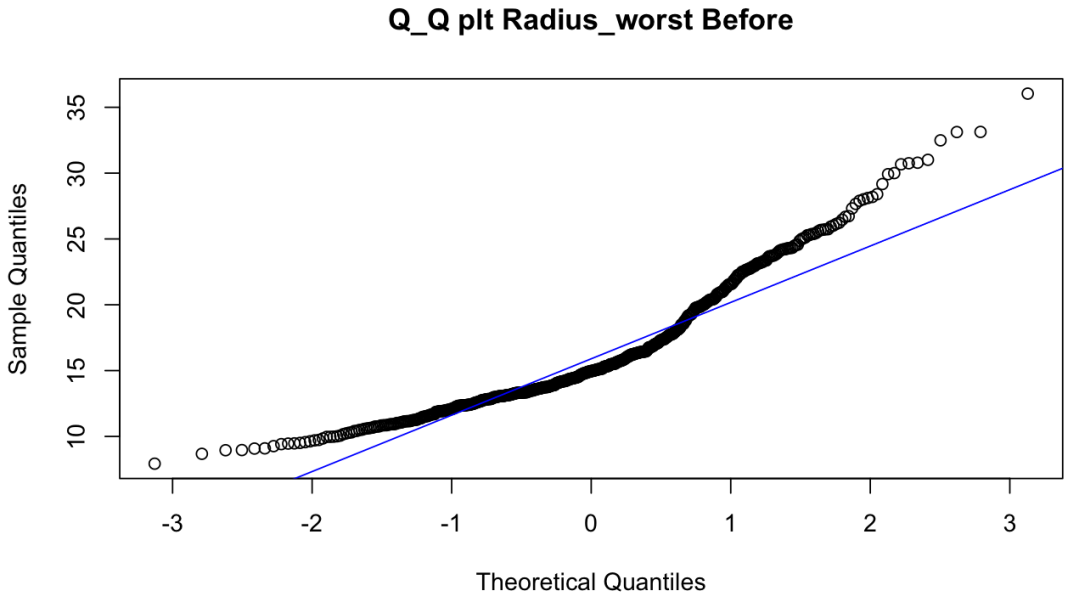


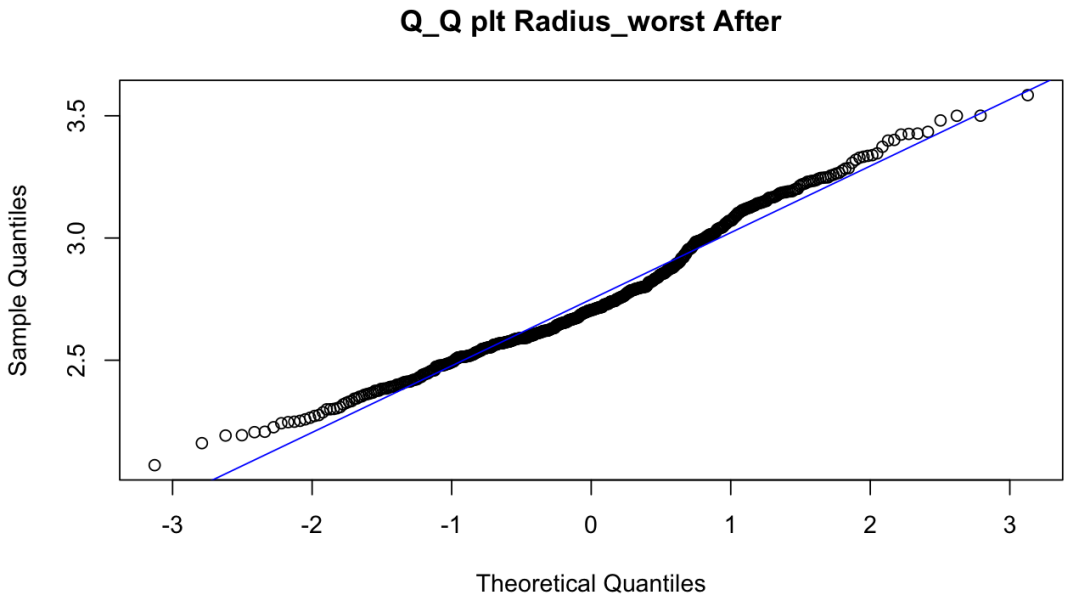

```
#before

qqnorm(wdbc$radius_SE,main="Q_Q plt Radius_SE Before")
qqline(wdbc$radius_SE, col="blue")
#after
qqnorm(wdbc_df$radius_SE,main="Q_Q plt Radius_SE After")
qqline(wdbc_df$radius_SE, col="blue")
```






```
#before
qqnorm(wdbc$radius_worst,main="Q_Q plt Radius_worst Before")
qqline(wdbc$radius_worst, col="blue")
#after
qqnorm(wdbc_df$radius_worst,main="Q_Q plt Radius_worst After")
qqline(wdbc_df$radius_worst, col="blue")
```

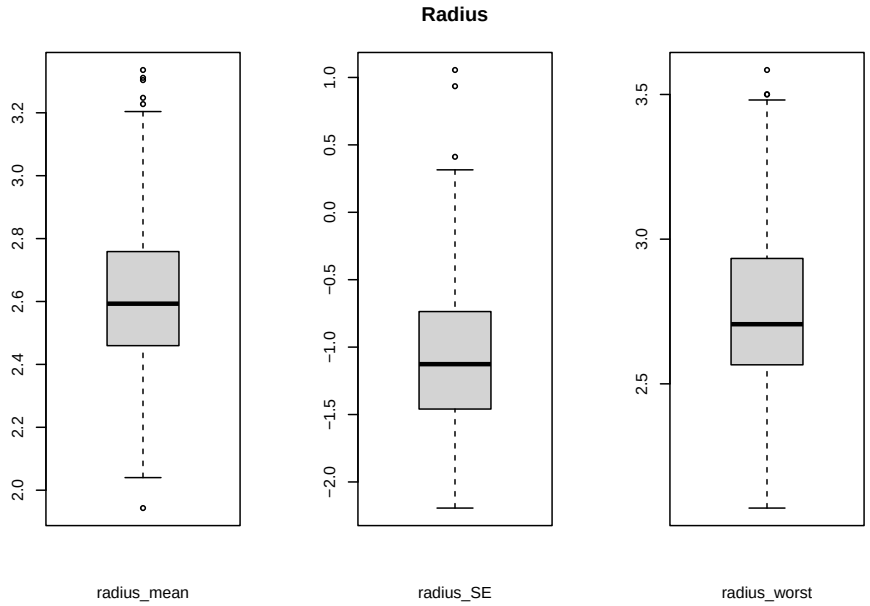

Q_Q plt Radius_worst Before
Sample Quantiles
10
15
20
25
30
35
-3
-2
-1
0
1
2
3
Theoretical Quantiles


Q_Q plt Radius_worst After
Sample Quantiles
2.5
3.0
3.5
-3
-2
-1
0
1
2
3
Theoretical Quantiles

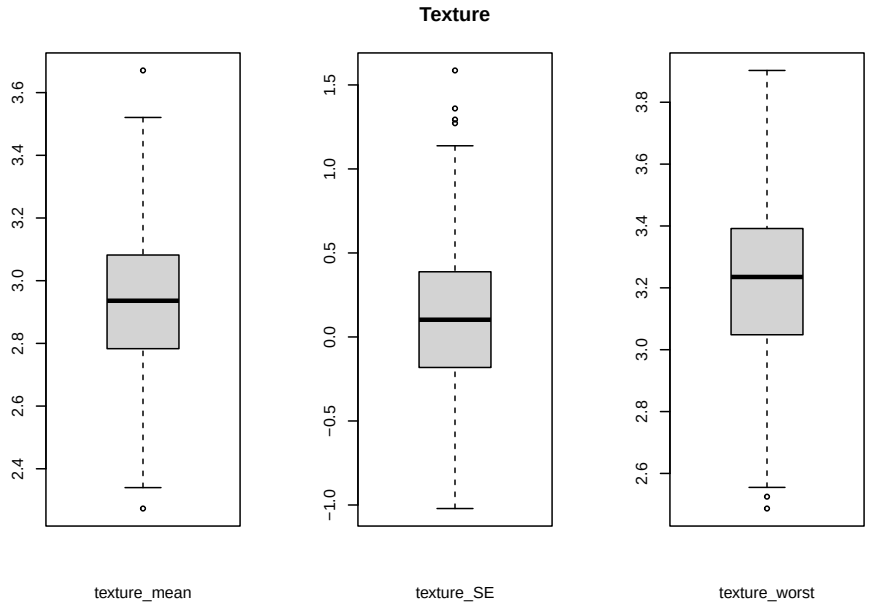
Boxplot

At this point we plotted again all the data.

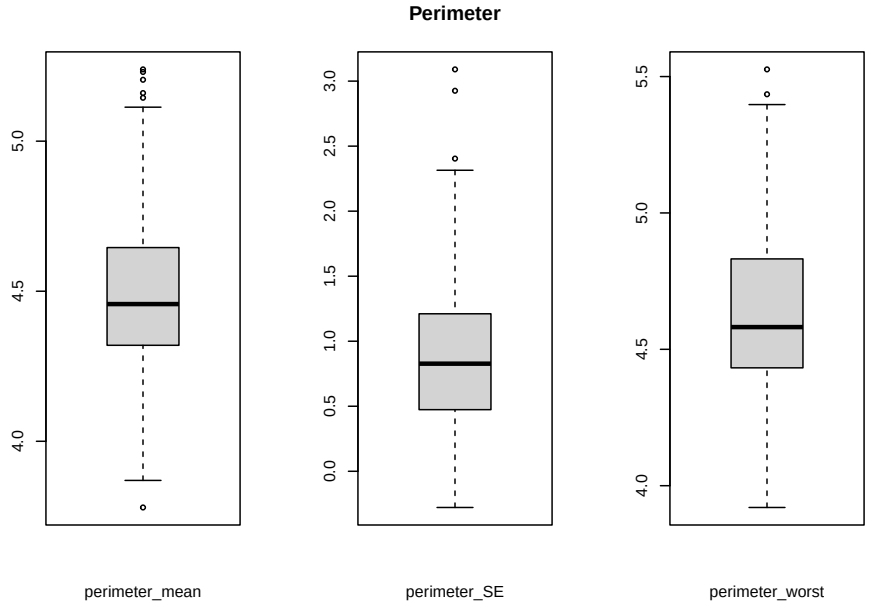
```
# boxplot dei dati normalizzati


##### BOX PLOT #####
par(mfrow=c(1, 3))
# Radius
boxplot(radius_mean,
        xlab = "radius_mean",
        main = "")
boxplot(radius_SE,
        xlab = "radius_SE",
        main = "Radius")
boxplot( radius_worst,
        xlab = "radius_worst",
        main = "")
```



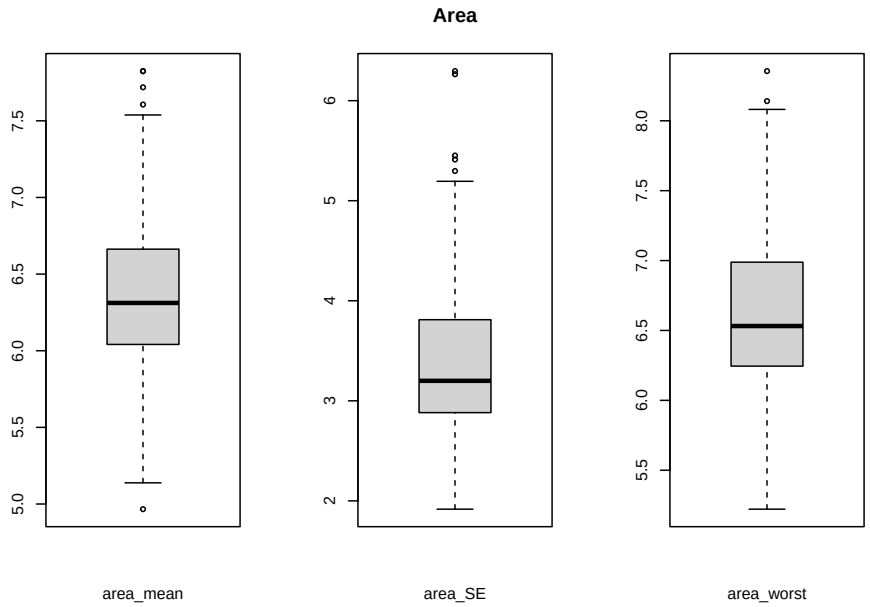

```
par(mfrow=c(1, 3))
# Texture
boxplot(texture_mean,
        xlab = "texture_mean",
        main = "")
boxplot(texture_SE,
        xlab = "texture_SE",
        main = "Texture")
boxplot(texture_worst,
        xlab = "texture_worst",
        main = "")
```



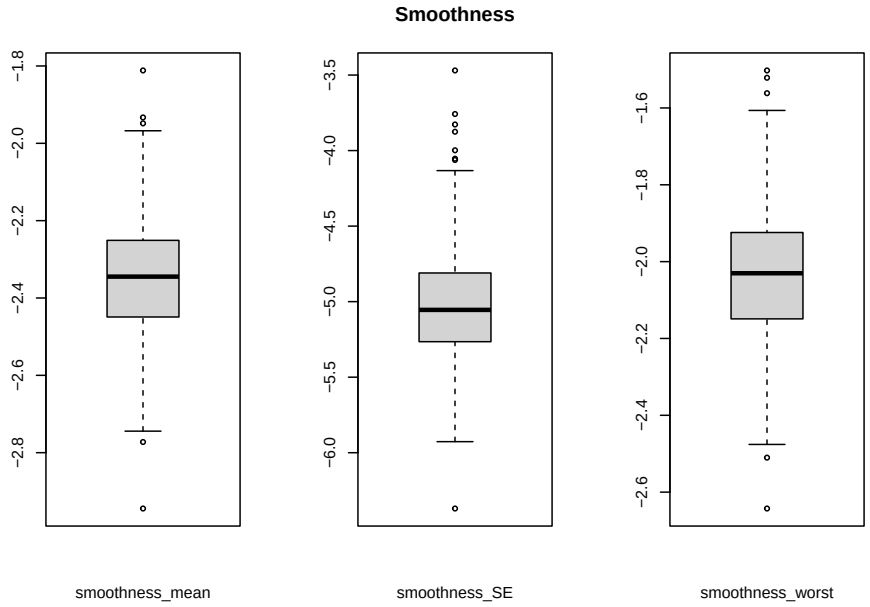

```
par(mfrow=c(1, 3))
# Perimeter
boxplot(perimeter_mean,
        xlab = "perimeter_mean",
        main = "")
boxplot( perimeter_SE,
        xlab = "perimeter_SE",
        main = "Perimeter")
boxplot( perimeter_worst,
        xlab = "perimeter_worst",
        main = "")
```



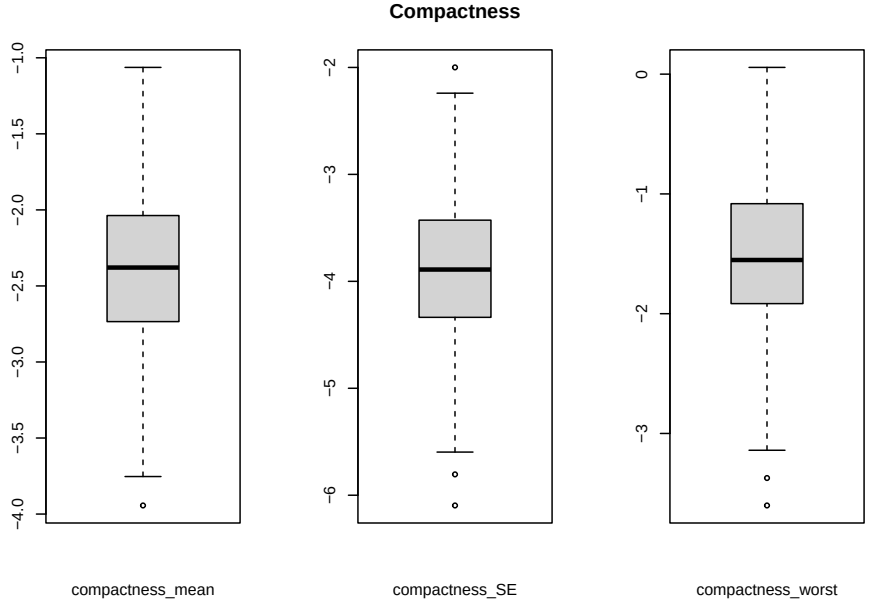
```
par(mfrow=c(1, 3))
# Area
boxplot( area_mean,
        xlab = "area_mean",
        main = "")
boxplot( area_SE,
        xlab = "area_SE",
        main = "Area")
boxplot(area_worst,
        xlab = "area_worst",
        main = "")
```



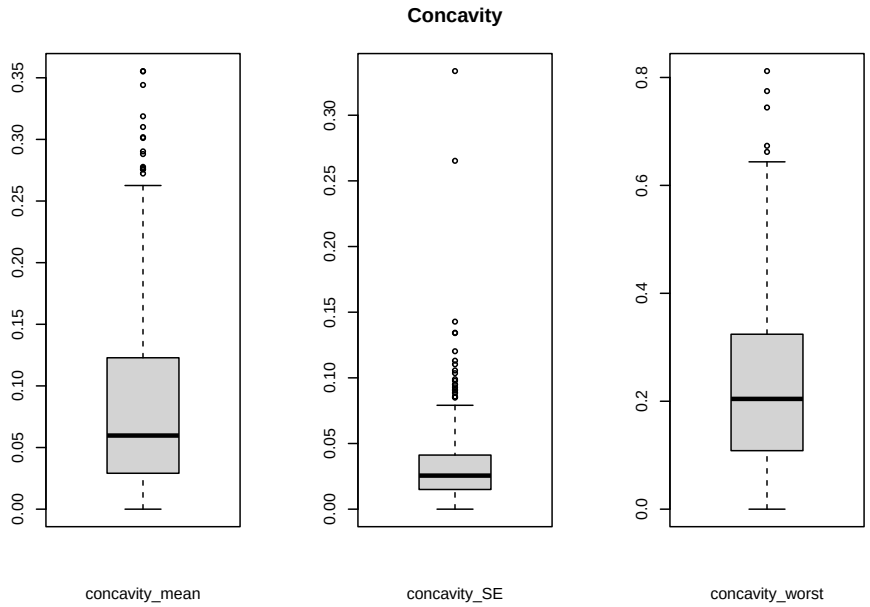

```
par(mfrow=c(1, 3))
# Smoothness
boxplot(smoothness_mean,
        xlab = "smoothness_mean",
        main = "")
boxplot(smoothness_SE,
        xlab = "smoothness_SE",
        main = "Smoothness")
boxplot(smoothness_worst,
        xlab = "smoothness_worst",
        main = "")
```



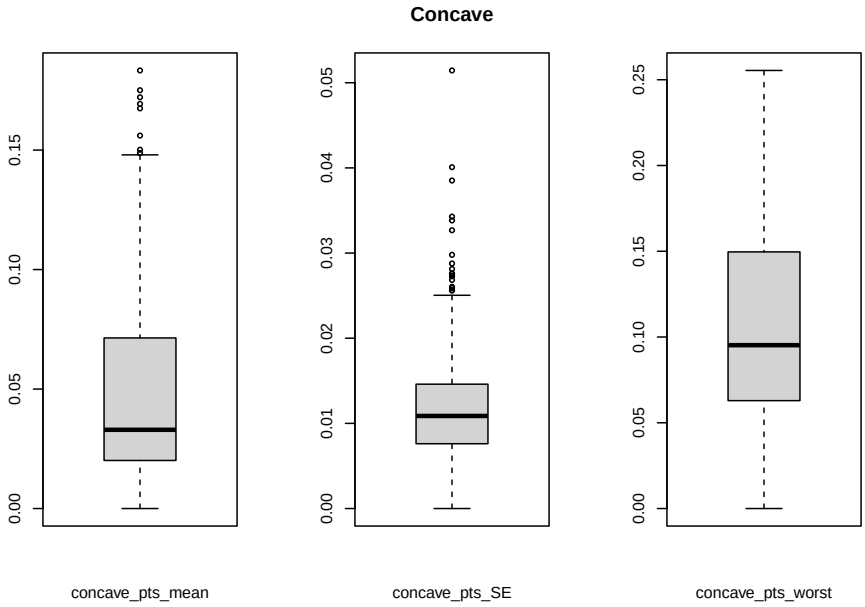

```
par(mfrow=c(1, 3))
# Compactness
boxplot(compactness_mean,
        xlab = "compactness_mean",
        main = "")
boxplot( compactness_SE,
        xlab = "compactness_SE",
        main = "Compactness")
boxplot( compactness_worst,
        xlab = "compactness_worst",
        main = "")
```



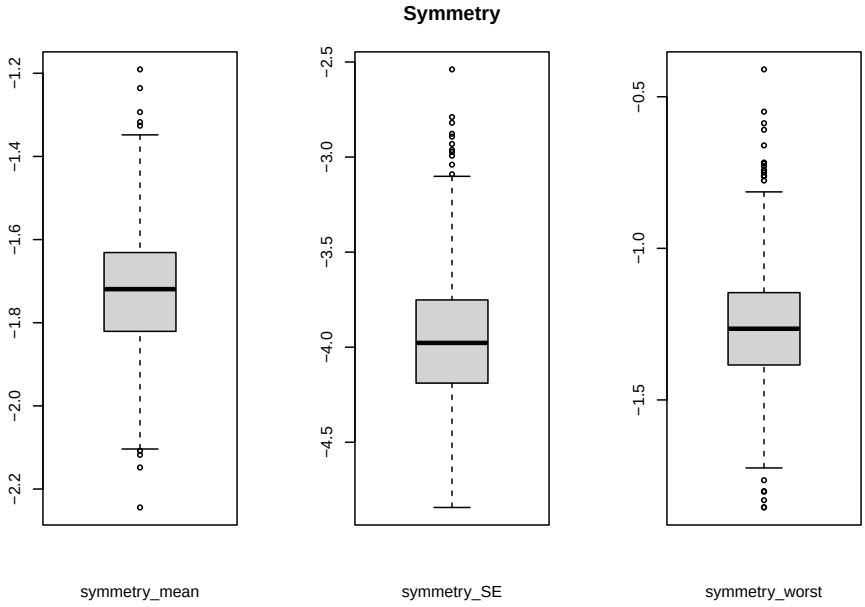
```r
par(mfrow=c(1, 3))
# Concavity
boxplot(concavity_mean,
        xlab = "concavity_mean",
        main = "")
boxplot( concavity_SE,
        xlab = "concavity_SE",
        main = "Concavity")
boxplot(concavity_worst,
        xlab = "concavity_worst",
        main = "")
```



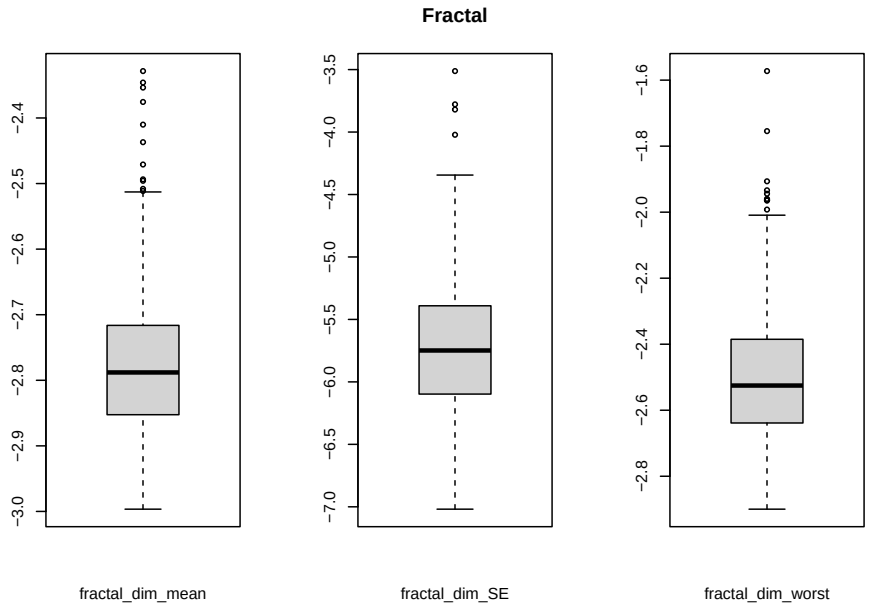

```r
par(mfrow=c(1, 3))
# Concave_pts_mean
boxplot(concave_pts_mean,
        xlab = "concave_pts_mean",
        main = "")
boxplot(concave_pts_SE,
        xlab = "concave_pts_SE",
        main = "Concave")
boxplot(concave_pts_worst,
        xlab = "concave_pts_worst",
        main = "")
```




```
par(mfrow=c(1, 3))
# compactness_mean
boxplot(symmetry_mean,
        xlab = "symmetry_mean",
        main = "")
boxplot(symmetry_SE,
        xlab = "symmetry_SE",
        main = "Symmetry")
boxplot(symmetry_worst,
        xlab = "symmetry_worst",
        main = "")
```



```r
par(mfrow=c(1, 3))
# Fractal_dim_mean
boxplot(fractal_dim_mean,
        xlab = "fractal_dim_mean",
        main = "")
boxplot( fractal_dim_SE,
        xlab = "fractal_dim_SE",
        main = "Fractal")
boxplot( fractal_dim_worst,
        xlab = "fractal_dim_worst",
        main = "")
```

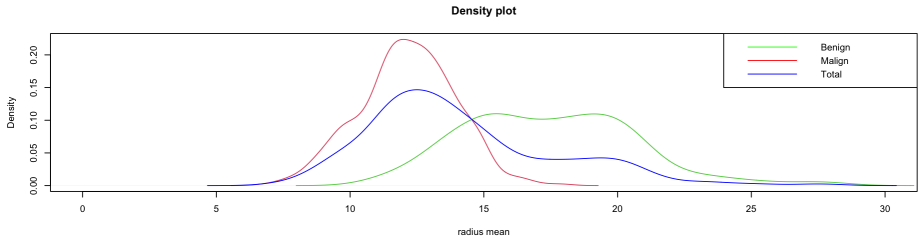


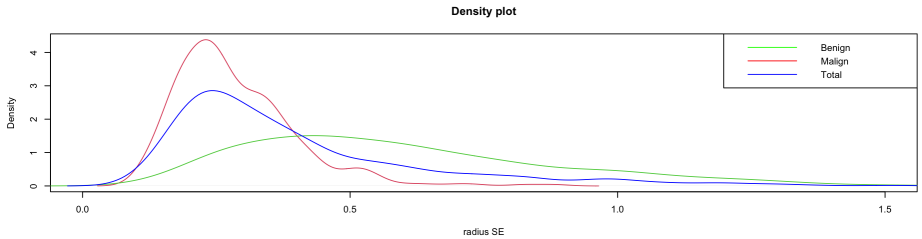
Density plot

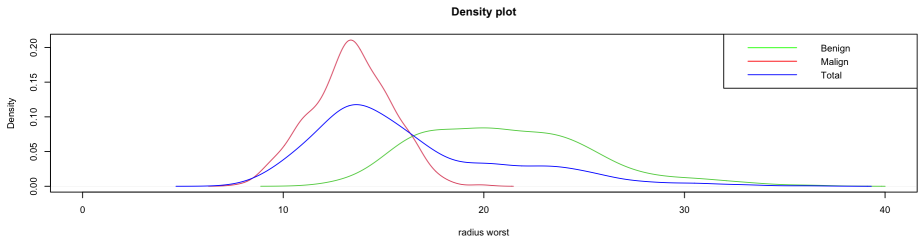
For the density plot, we kept the same scheme to arrange the plots and for each one we plotted the overall density (in Blue), the Benign Density (In Green) and the Malignant Density (In Red), in order to have a complete overview of the situation.

Radius

```
plot(density(wdbc_B$radius_mean), col = 2, main ="Density plot",
     xlab = "radius mean" ,xlim=c(0,30))
lines(density(wdbc_M$radius_mean), col = 3)
lines(density(wdbc$radius_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$radius_SE), col = 2, main ="Density plot",
     xlab = "radius SE",xlim=c(0,1.5))
lines(density(wdbc_M$radius_SE), col = 3)
lines(density(wdbc$radius_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$radius_worst), col = 2, main ="Density plot",
     xlab = "radius worst",xlim=c(0,40))
lines(density(wdbc_M$radius_worst), col = 3)
lines(density(wdbc$radius_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```

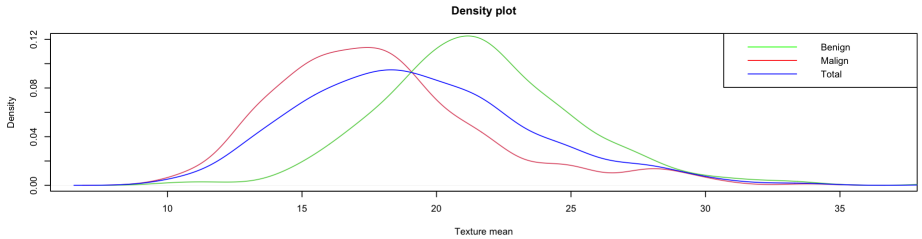


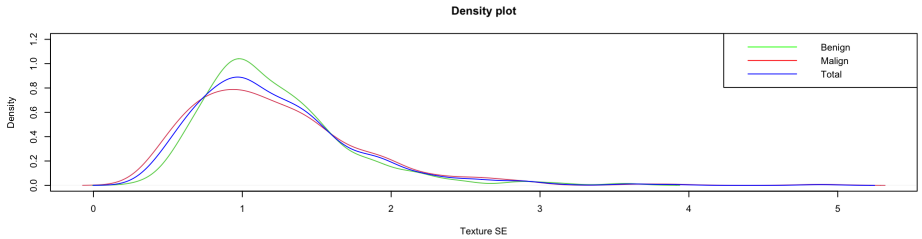


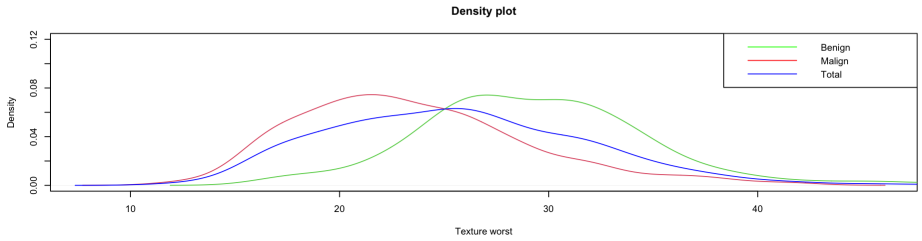


Texture

```
plot(density(wdbc_B$texture_mean), col = 2, main ="Density plot",
     xlab = "Texture mean", ylim = c(0, 0.12))
lines(density(wdbc_M$texture_mean), col = 3)
lines(density(wdbc$texture_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$texture_SE), col = 2, main ="Density plot",
     xlab = "Texture SE", ylim = c(0, 1.2))
lines(density(wdbc_M$texture_SE), col = 3)
lines(density(wdbc$texture_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$texture_worst), col = 2, main ="Density plot",
     xlab = "Texture worst", ylim = c(0, 0.12))
lines(density(wdbc_M$texture_worst), col = 3)
lines(density(wdbc$texture_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```



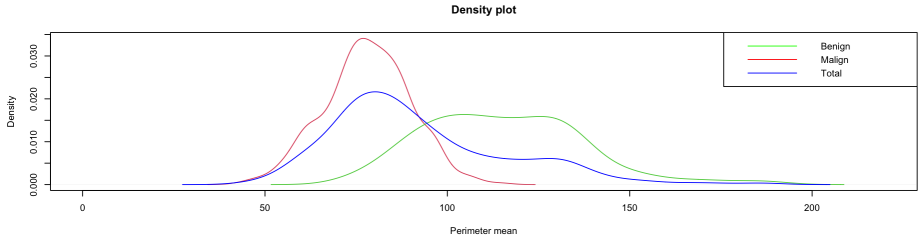


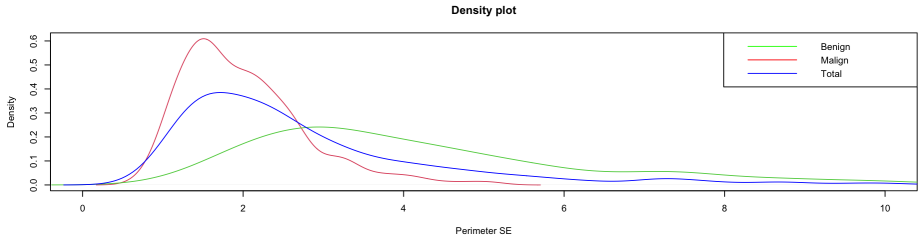


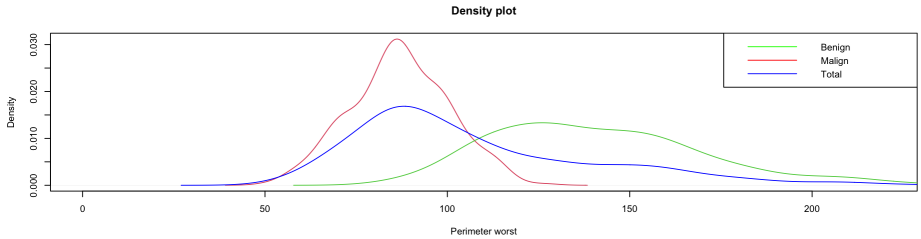
Perimeter

```
plot(density(wdbc_B$perimeter_mean), col = 2, main ="Density plot",
     xlab = "Perimeter mean",  xlim = c(0, 220) )
lines(density(wdbc_M$perimeter_mean), col = 3)
lines(density(wdbc$perimeter_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```

```
plot(density(wdbc_B$perimeter_SE), col = 2, main ="Density plot",
     xlab = "Perimeter SE",xlim=c(0,10) )
lines(density(wdbc_M$perimeter_SE), col = 3)
lines(density(wdbc$perimeter_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$perimeter_worst), col = 2, main ="Density plot",
     xlab = "Perimeter worst", xlim = c(0, 220) )
lines(density(wdbc_M$perimeter_worst), col = 3)
lines(density(wdbc$perimeter_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```



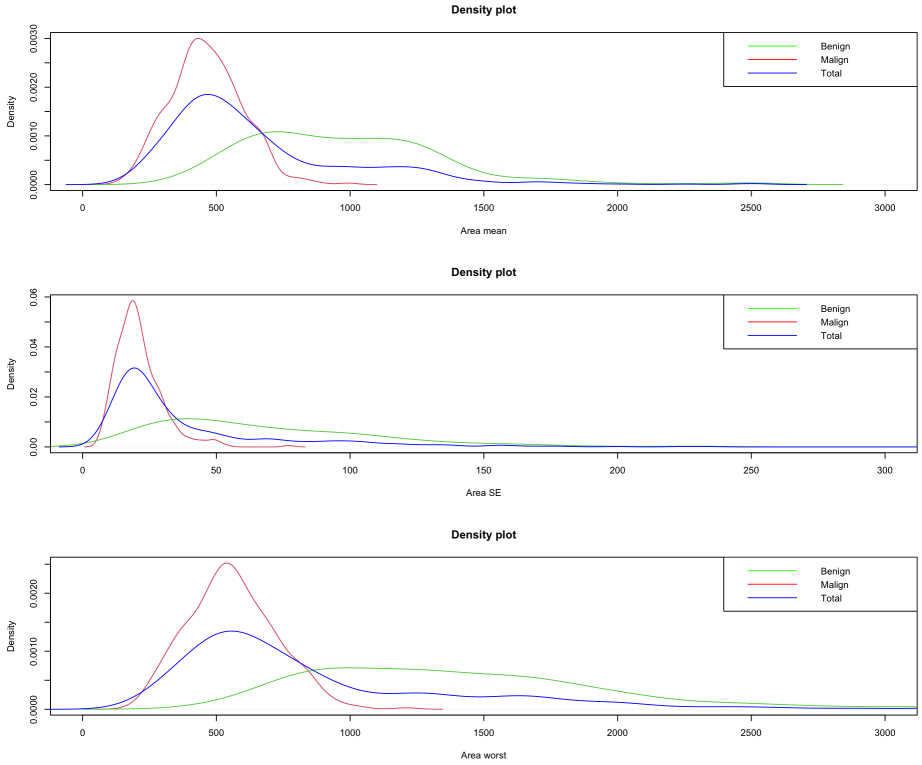




Area

```
plot(density(wdbc_B$area_mean), col = 2, main ="Density plot",
     xlab = "Area mean", xlim = c(0, 3000))
lines(density(wdbc_M$area_mean), col = 3)
lines(density(wdbc$area_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$area_SE), col = 2, main ="Density plot",
     xlab = "Area SE",  xlim = c(0, 300))
lines(density(wdbc_M$area_SE), col = 3)
lines(density(wdbc$area_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
```

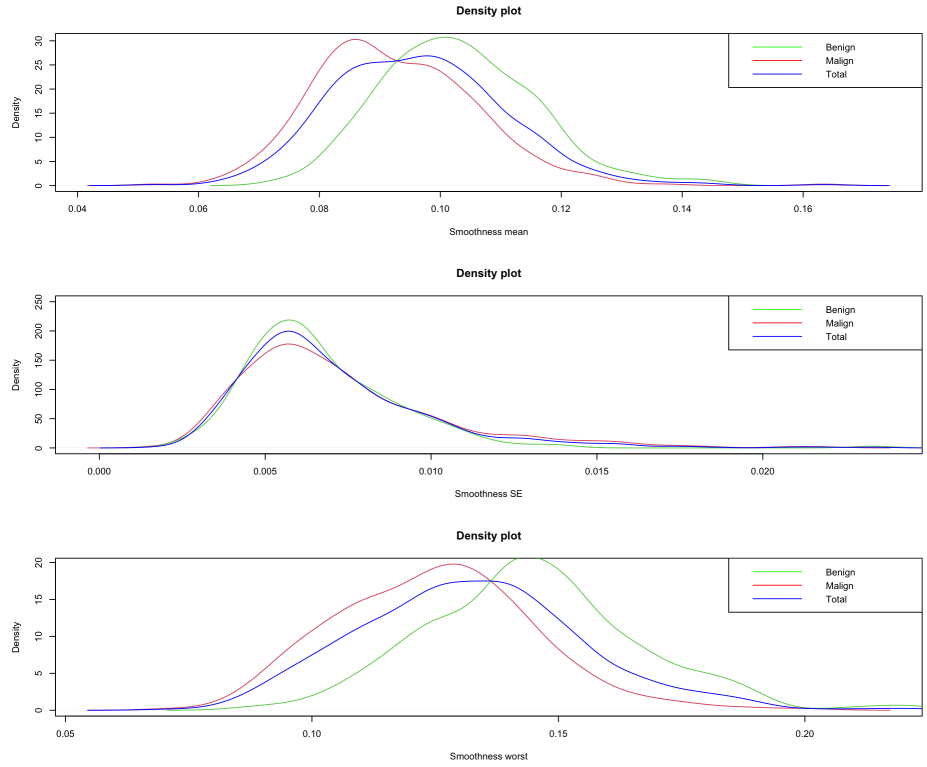
```
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$area_worst), col = 2, main ="Density plot",
     xlab = "Area worst",  xlim = c(0, 3000))
lines(density(wdbc_M$area_worst), col = 3)
lines(density(wdbc$area_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```



Smoothness

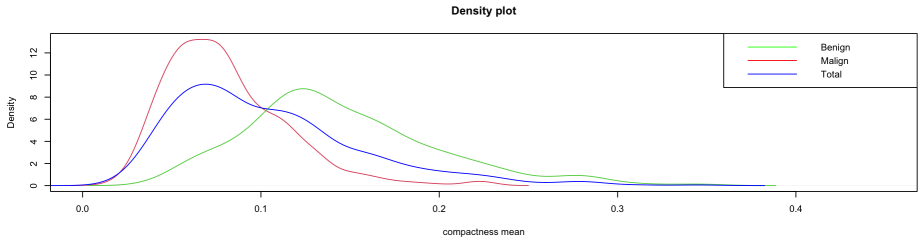
```
plot(density(wdbc_B$smoothness_mean), col = 2, main ="Density plot",
     xlab = "Smoothness mean")
lines(density(wdbc_M$smoothness_mean), col = 3)
lines(density(wdbc$smoothness_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$smoothness_SE), col = 2, main ="Density plot",
     xlab = "Smoothness SE",ylim=c(0,250))
lines(density(wdbc_M$smoothness_SE), col = 3)
lines(density(wdbc$smoothness_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$smoothness_worst), col = 2, main ="Density plot",
     xlab = "Smoothness worst")
lines(density(wdbc_M$smoothness_worst), col = 3)
lines(density(wdbc$smoothness_worst), col = "blue")
```

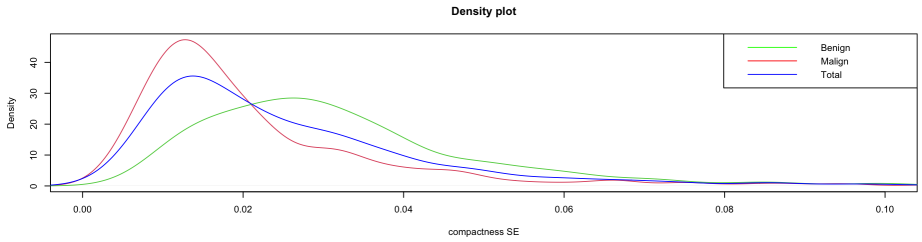
```
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```

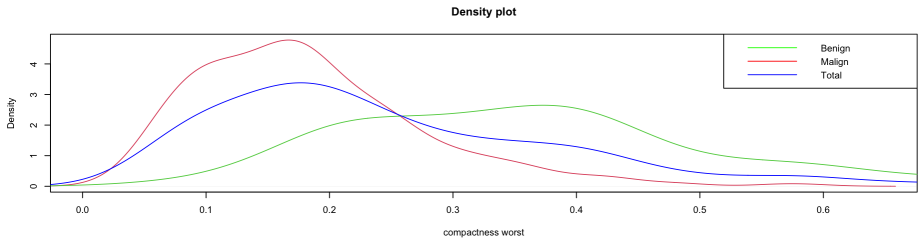


Compactness

```
plot(density(wdbc_B$compactness_mean), col = 2, main ="Density plot",
     xlab = "compactness mean", xlim = c(0, 0.45))
lines(density(wdbc_M$compactness_mean), col = 3)
lines(density(wdbc$compactness_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$compactness_SE), col = 2, main ="Density plot",
     xlab = "compactness SE",  xlim = c(0, 0.1))
lines(density(wdbc_M$compactness_SE), col = 3)
lines(density(wdbc$compactness_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$compactness_worst), col = 2, main ="Density plot",
     xlab = "compactness worst",  xlim = c(0, 0.65))
lines(density(wdbc_M$compactness_worst), col = 3)
lines(density(wdbc$compactness_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```

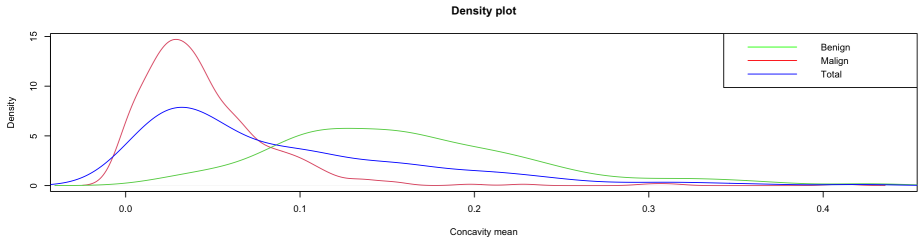


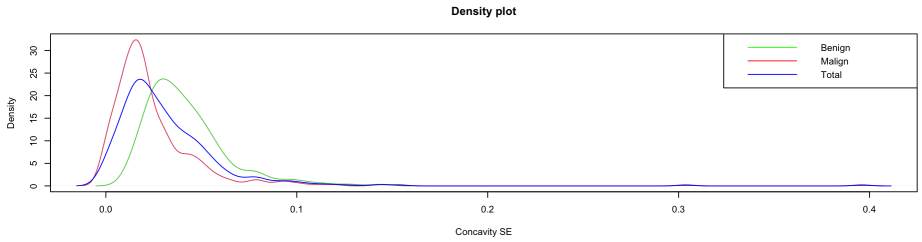


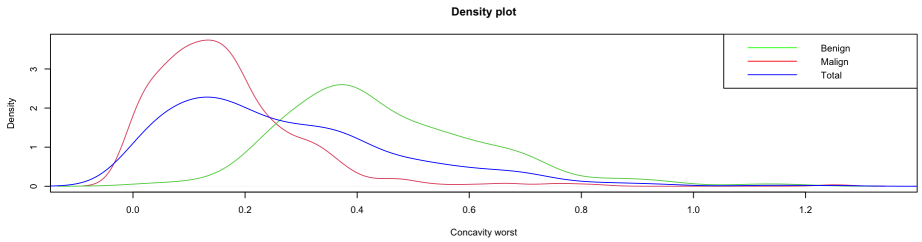


Concavity

```
plot(density(wdbc_B$concavity_mean), col = 2, main ="Density plot",
     xlab = "Concavity mean")
lines(density(wdbc_M$concavity_mean), col = 3)
lines(density(wdbc$concavity_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$concavity_SE), col = 2, main ="Density plot",
     xlab = "Concavity SE")
lines(density(wdbc_M$concavity_SE), col = 3)
lines(density(wdbc$concavity_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$concavity_worst), col = 2, main ="Density plot",
     xlab = "Concavity worst")
lines(density(wdbc_M$concavity_worst), col = 3)
lines(density(wdbc$concavity_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```

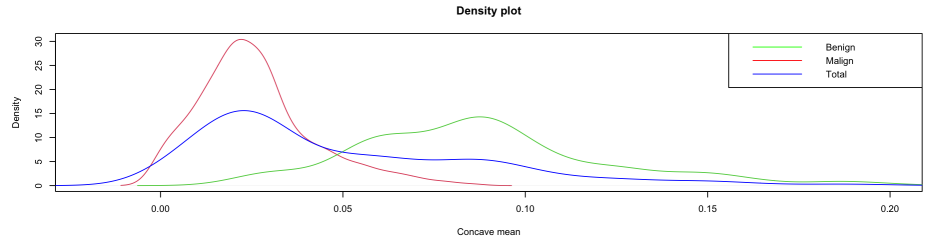


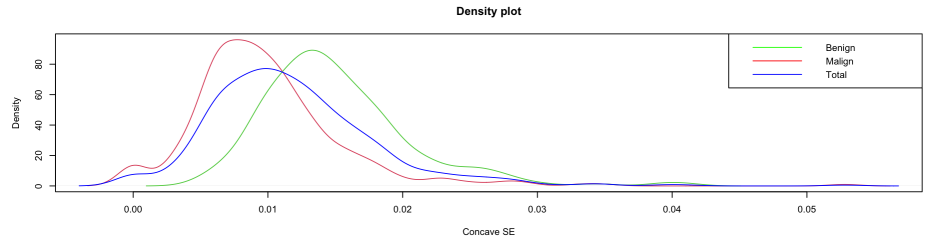


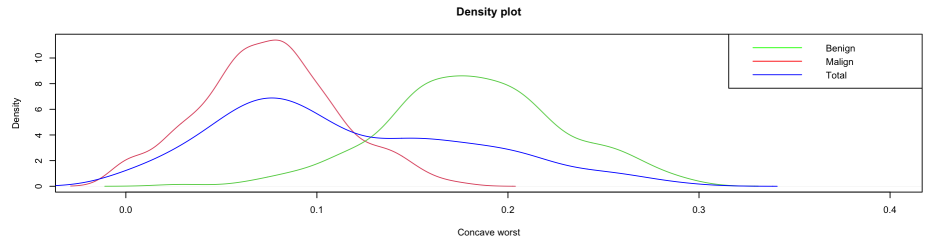


Concave_pts

```
plot(density(wdbc_B$concave_pts_mean), col = 2, main ="Density plot",
     xlab = "Concave mean",xlim=c(-0.02,0.2))
lines(density(wdbc_M$concave_pts_mean), col = 3)
lines(density(wdbc$concave_pts_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$concave_pts_SE), col = 2, main ="Density plot",
     xlab = "Concave SE")
lines(density(wdbc_M$concave_pts_SE), col = 3)
lines(density(wdbc$concave_pts_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$concave_pts_worst), col = 2, main ="Density plot",
     xlab = "Concave worst",xlim=c(-0.02,0.4))
lines(density(wdbc_M$concave_pts_worst), col = 3)
lines(density(wdbc$concave_pts_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```

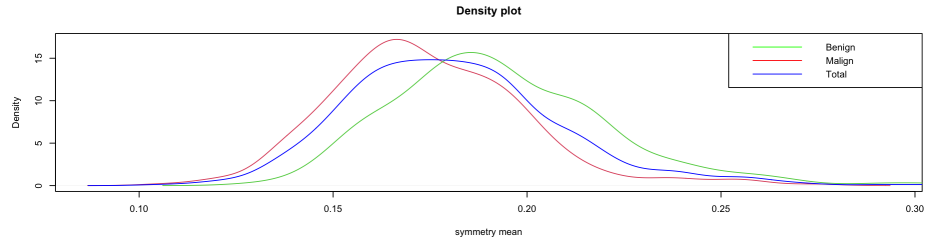


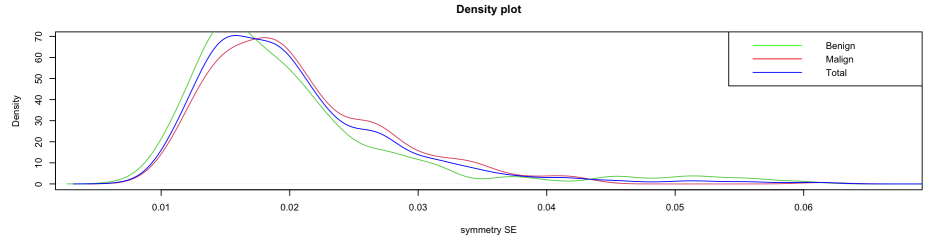


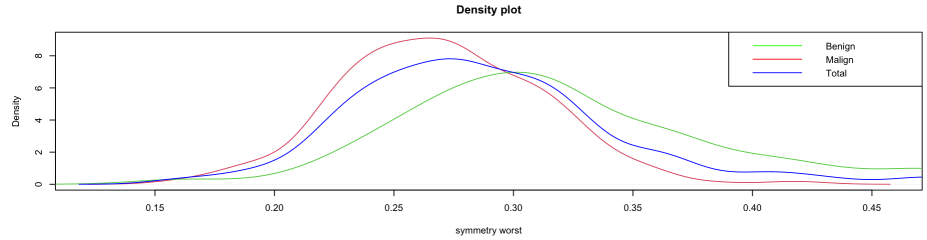


Symmetry

```
plot(density(wdbc_B$symmetry_mean), col = 2, main ="Density plot",
     xlab = "symmetry mean")
lines(density(wdbc_M$symmetry_mean), col = 3)
lines(density(wdbc$symmetry_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$symmetry_SE), col = 2, main ="Density plot",
     xlab = "symmetry SE")
lines(density(wdbc_M$symmetry_SE), col = 3)
lines(density(wdbc$symmetry_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$symmetry_worst), col = 2, main ="Density plot",
     xlab = "symmetry worst")
lines(density(wdbc_M$symmetry_worst), col = 3)
lines(density(wdbc$symmetry_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```

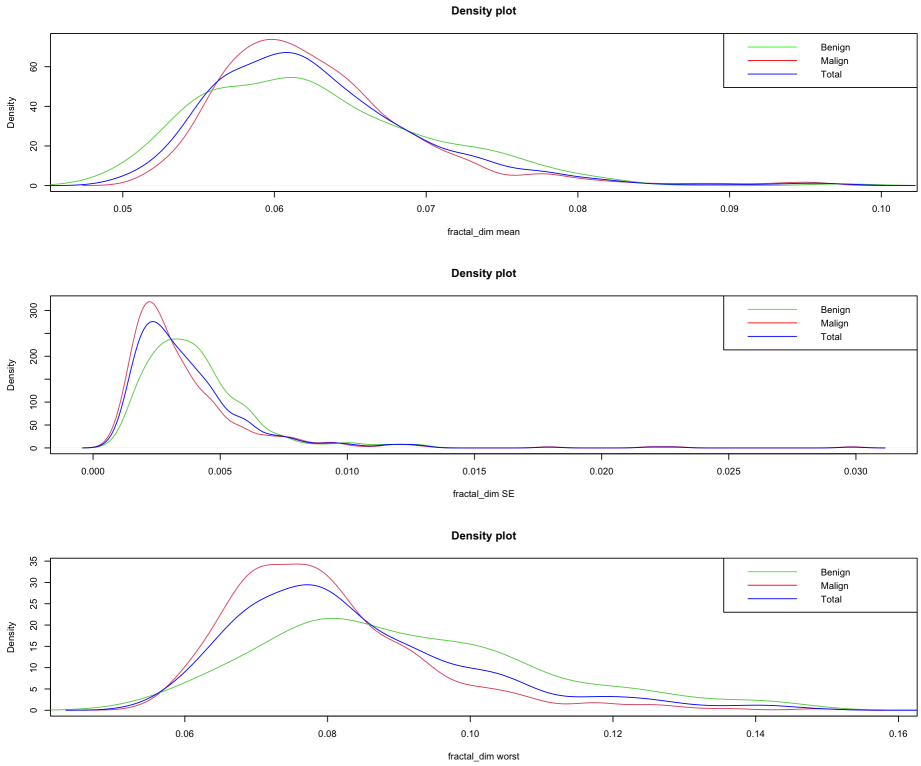






Fractal_dim

```
plot(density(wdbc_B$fractal_dim_mean), col = 2, main ="Density plot",
     xlab = "fractal_dim mean")
lines(density(wdbc_M$fractal_dim_mean), col = 3)
lines(density(wdbc$fractal_dim_mean), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$fractal_dim_SE), col = 2, main ="Density plot",
     xlab = "fractal_dim SE")
lines(density(wdbc_M$fractal_dim_SE), col = 3)
lines(density(wdbc$fractal_dim_SE), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)

plot(density(wdbc_B$fractal_dim_worst), col = 2, main ="Density plot",
     xlab = "fractal_dim worst")
lines(density(wdbc_M$fractal_dim_worst), col = 3)
lines(density(wdbc$fractal_dim_worst), col = "blue")
legend("topright", legend = c("Benign","Malign","Total"),
       col = c("green", "red","blue"), lwd = 1)
```

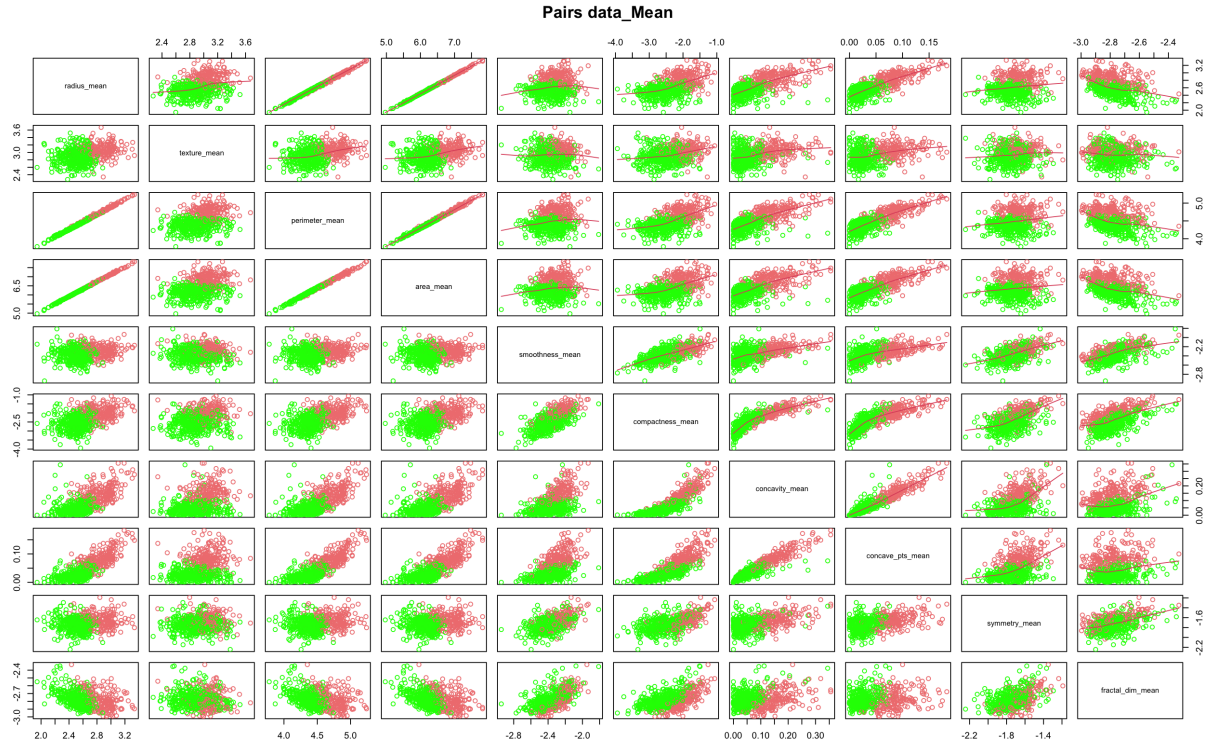


If we look at the density plots, we can notice that the Benign column, in general, doesn't have a clear bell shape, while Malignant has.
In general, thanks to the normalization we managed to have a unimodal distribution instead of the bimodal that we had before.

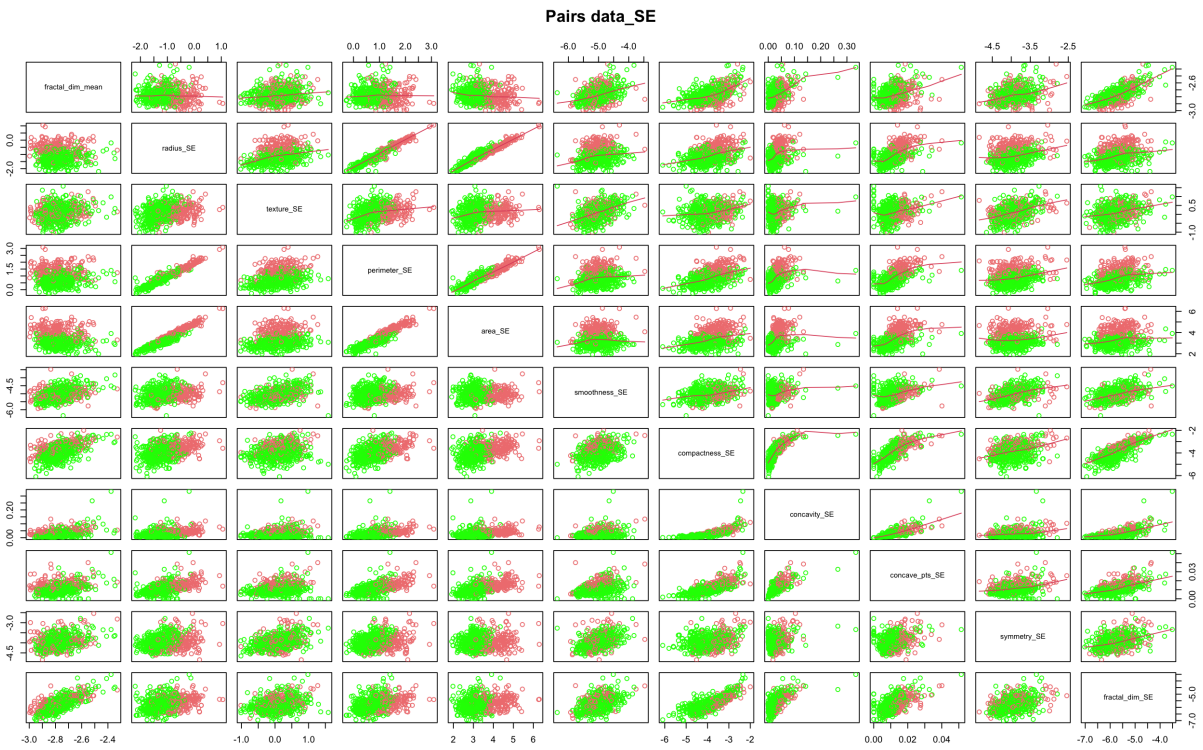
Pair plots

To see how the features are related to each other we called the *pairs()* function for every type of parameter.

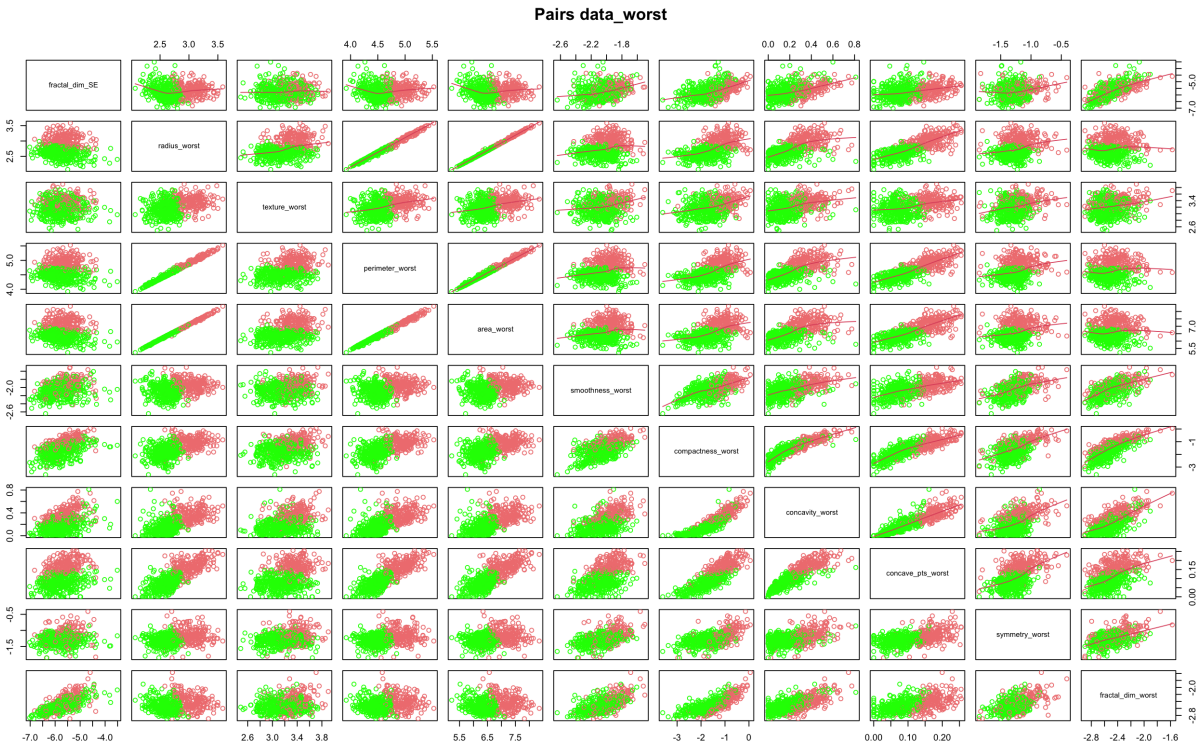
```
pairs(~.,data=wdbc_df[c(1:10)],
      col = c("green","lightcoral")[group],    # Change color by group
      pch = c(1, 1)[group],
      upper.panel=panel.smooth,
      main = "Pairs data_Mean")
```




```
pairs(~.,data=wdbc_df[c(10:20)],
      col = c("green","lightcoral")[group],    # Change color by group
      pch = c(1, 1)[group],                                # Change points by group
      upper.panel=panel.smooth,
      main = "Pairs data_SE")
```




```
pairs(~.,data=wdbc_df[c(20:30)],
      col = c("green","lightcoral")[group],   # Change color by group
      pch = c(1, 1)[group],                          # Change points by group
      upper.panel=panel.smooth,
      main = "Pairs data_worst")
```

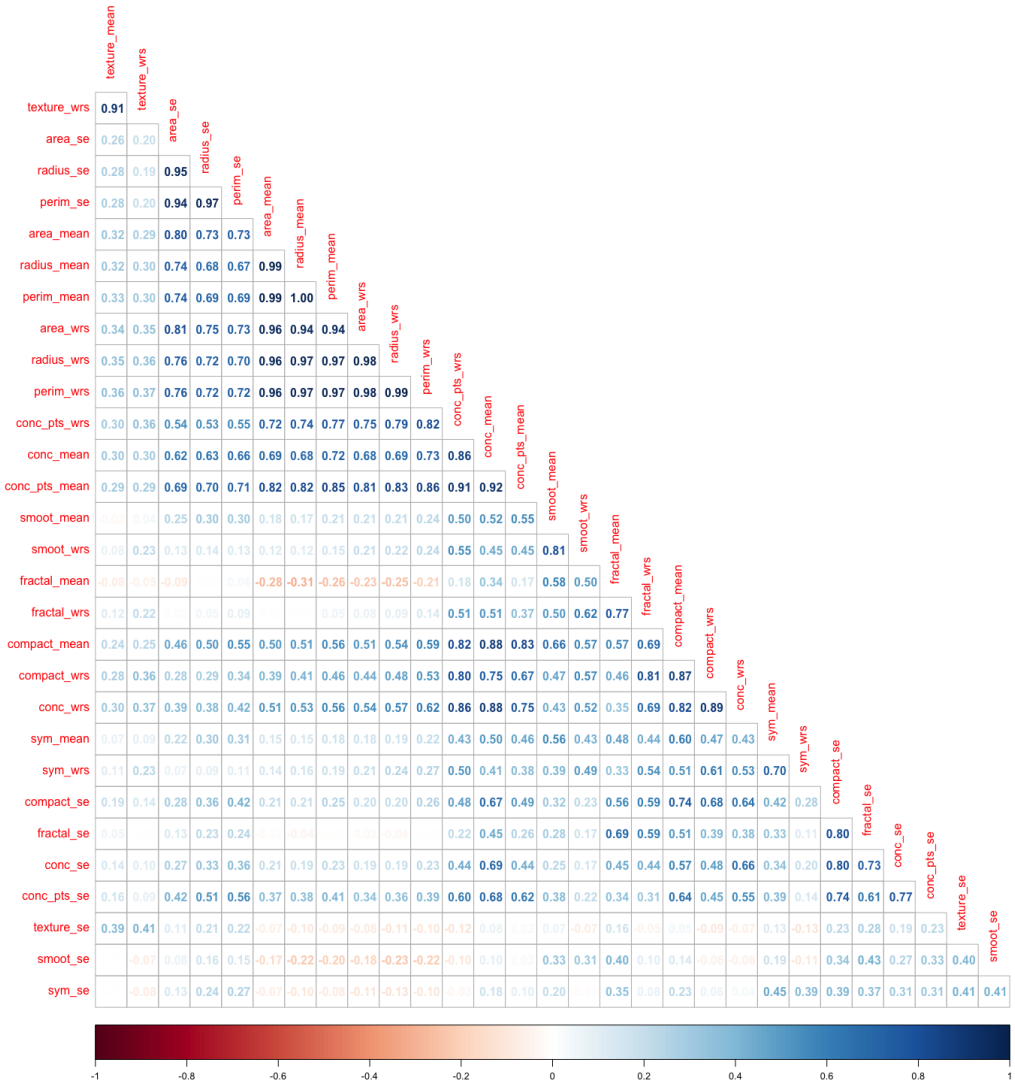


Using the same color scheme as before, we made clear the distinction between Benign and Malignant tumor in the scatter plots.
Just looking at these figures it is evident that there are some variables strongly correlated with each other such as *radius_mean/perimeter_mean* or *perimeter_mean/area_mean* and some less correlated such as *smoothness_mean/texture_mean* or *symmetry_worst/area_worst*.

Covariance Matrix

In order to have numerical values of the covariance between our features we constructed the covariance matrix.

```
cor_BM<- cor(wdbc[-c(1,2)])
#cambio nomi per leggibilita
colnames(cor_BM) = c( 'radius_mean', 'texture_mean', 'perim_mean', 'area_mean',
                      'smoot_mean', 'compact_mean', 'conc_mean', 'conc_pts_mean', 'sym_mean',
                      'fractal_mean','radius_se', 'texture_se', 'perim_se', 'area_se',
                      'smoot_se', 'compact_se', 'conc_se', 'conc_pts_se', 'sym_se',
                      'fractal_se','radius_wrs', 'texture_wrs', 'perim_wrs', 'area_wrs',
                      'smoot_wrs', 'compact_wrs', 'conc_wrs', 'conc_pts_wrs', 'sym_wrs',
                      'fractal_wrs')
rownames(cor_BM)=c( 'radius_mean', 'texture_mean', 'perim_mean', 'area_mean',
                    'smoot_mean', 'compact_mean', 'conc_mean', 'conc_pts_mean', 'sym_mean',
                    'fractal_mean','radius_se', 'texture_se', 'perim_se', 'area_se',
                    'smoot_se', 'compact_se', 'conc_se', 'conc_pts_se', 'sym_se',
                    'fractal_se','radius_wrs', 'texture_wrs', 'perim_wrs', 'area_wrs',
                    'smoot_wrs', 'compact_wrs', 'conc_wrs', 'conc_pts_wrs', 'sym_wrs',
                    'fractal_wrs')


corrplot(cor_BM, method = 'number', diag = FALSE, order = 'hclust', type='lower',
         addrect = 3, rect.col = 'blue', rect.lwd = 2 )
```



As we can see, some of the predicted relationships are confirmed by the presence of numbers ~1 in the matrix. During the analysis we will have to take into account these strong correlations, as we are going to explain in the next section.

Models

At this point, after looking at the data, we start doing classification. We took into account different models: logistic regression, k-NN, Bayes Classifier, LDA and QDA.

To have a better idea of how well each model was doing classification on unseen data, we divided the dataset into two different sets: train and test. The division is done using a different percentage: 80% of data are in the training set, while the 20% are in the test set.
We used the training set to train the model, while the test set has been used to do predictions and calculate the accuracy of the model.
For reproducibility purposes, we also set a seed.

Logistic Regression as Classification

The first model we used to do classification is Logistic regression.

At first we created a model using all the features but, as expected, we got warning messages.

```
set.seed(161)
sample <- sample(1:569, size=455, replace= FALSE)
wdbc_train <- wdbc_df[sample,] #training set
wdbc_test <- wdbc_df[-sample,] #test set

lr_model_0 <- glm(wdbc_train$diagnosis~ . ,data = wdbc_train, family = binomial)
```

```
## Warning: glm.fit: l'algoritmo non converge
```

```
## Warning: glm.fit: si sono verificate probabilità stimate numericamente pari a 0
## o 1
```

This is due to the fact that, as said before, in our dataset we have features that are strongly correlated and in a regression problem, strong correlation, can lead to uncertainty in the coefficient estimates.

We decided to apply the Variance Inflation Factor (VIF) to deal with this problem.
The VIF is defined as

$$\mathrm{VIF}(\hat{\beta}_j) = \frac{1}{1 - R^2_{x_j|x_{-j}}}$$

and, according to the output value, it helps to understand if there is collinearity or not. The value 1 indicates the complete absence of collinearity, while values greater than 10 indicate a problematic collinearity. These are the results obtained using the *vif()* function in R.

```
vif(lr_model_0)
```

```
##       radius_mean      texture_mean    perimeter_mean         area_mean
##       15218.44015         167.24240       43290.64843       21723.85577
##   smoothness_mean  compactness_mean    concavity_mean  concave_pts_mean
##         141.25014         179.55738         184.72538         201.90282
##     symmetry_mean  fractal_dim_mean         radius_SE        texture_SE
##          31.32979          55.41660        3202.03068          57.24564
##      perimeter_SE           area_SE     smoothness_SE    compactness_SE
##         422.93630        4076.49099          93.74082         169.58219
##      concavity_SE    concave_pts_SE       symmetry_SE    fractal_dim_SE
##         205.16199         127.87768         143.77884         274.01617
##      radius_worst     texture_worst   perimeter_worst        area_worst
##        4943.09284         276.74398        1223.58974        6382.22674
##  smoothness_worst compactness_worst   concavity_worst concave_pts_worst
##         200.64867         427.58625         118.53194         126.95973
##    symmetry_worst fractal_dim_worst
```

```
##          72.79374         440.19752
```

As we can notice, we have values that are much bigger than 10. We decided to proceed to eliminate, one by one, the feature with the biggest value to see how this would have changed the VIF value of the other features. Sometimes, eliminating the biggest value produced an increasing in the VIF, so in this case we decided to eliminate the second highest value.
After few attempts we obtained the following result.

```
vif(lr_model_18)
```

```
##      texture_mean   smoothness_mean  concave_pts_mean     symmetry_mean
##          6.772479          4.353268          4.337390          3.119615
##         radius_SE     smoothness_SE      concavity_SE       symmetry_SE
##          2.502346          3.163113          2.173610          2.559119
##    fractal_dim_SE      radius_worst     texture_worst concave_pts_worst
##          3.307410          3.098307          6.523176          4.013336
```

The features' values are closer to 10, so at this point we checked for the p-value calling the summary function.

```
summary(lr_model_18)
```

```
##
## Call:
## glm(formula = diagnosis ~ . - area_worst - radius_mean - perimeter_mean -
##     perimeter_worst - area_SE - concavity_mean - area_mean -
##     fractal_dim_worst - compactness_worst - concavity_worst -
##     texture_SE - perimeter_SE - fractal_dim_mean - concave_pts_SE -
##     symmetry_worst - compactness_mean - smoothness_worst - compactness_SE,
##     family = binomial, data = wdbc_train)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -106.0271    22.0412  -4.810 1.51e-06 ***
## texture_mean         5.0468     5.0837   0.993 0.320831
## smoothness_mean      6.9989     5.8815   1.190 0.234054
## concave_pts_mean    34.6539    48.7203   0.711 0.476909
## symmetry_mean        8.6504     4.6105   1.876 0.060623 .
## radius_SE            2.3649     1.6569   1.427 0.153499
## smoothness_SE        0.7031     1.7131   0.410 0.681465
## concavity_SE        44.0782    25.2837   1.743 0.081274 .
## symmetry_SE         -2.2835     1.6146  -1.414 0.157280
## fractal_dim_SE      -3.4908     1.4085  -2.478 0.013201 *
## radius_worst        22.4050     6.3451   3.531 0.000414 ***
## texture_worst        8.4325     4.6078   1.830 0.067242 .
## concave_pts_worst   48.3657    23.9301   2.021 0.043267 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 600.338  on 454  degrees of freedom
## Residual deviance:  48.891  on 442  degrees of freedom
## AIC: 74.891
##
## Number of Fisher Scoring iterations: 10
```

Since, as can be seen, the p-values are very high we decided to use backward selection with fixed stopping

rule. In our case we decided to set a significance level at 0.05.
According to backward selection, we started with all the variables in the model and remove one variable at the time, on the basis of the highest p-value.
We repeated these passages for 5 times until we reached the following result:

```
summary(lr_model_23)
```

```
##
## Call:
## glm(formula = diagnosis ~ . - area_worst - radius_mean - perimeter_mean -
##     perimeter_worst - area_SE - concavity_mean - area_mean -
##     fractal_dim_worst - compactness_worst - concavity_worst -
##     texture_SE - perimeter_SE - fractal_dim_mean - concave_pts_SE -
##     symmetry_worst - compactness_mean - smoothness_worst - compactness_SE -
##     smoothness_SE - concave_pts_mean - symmetry_SE - texture_mean -
##     symmetry_mean, family = binomial, data = wdbc_train)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -98.633     19.772  -4.989 6.08e-07 ***
## smoothness_mean      9.481      4.586   2.067 0.038720 *
## radius_SE            3.281      1.583   2.073 0.038210 *
## concavity_SE        38.238     19.172   1.994 0.046107 *
## fractal_dim_SE      -3.119      1.289  -2.420 0.015540 *
## radius_worst        21.034      5.368   3.919 8.90e-05 ***
## texture_worst       11.860      2.572   4.612 4.00e-06 ***
## concave_pts_worst   63.288     19.081   3.317 0.000911 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 600.338  on 454  degrees of freedom
## Residual deviance:  54.248  on 447  degrees of freedom
## AIC: 70.248
##
## Number of Fisher Scoring iterations: 10
```

At this point we used this model to do prediction on the test set.
In the first attempt, we fixed the threshold to 0.5: in this way everything higher than 0.5 is classified as Benign.
We obtained the following results:

```
CM <- table(logistic_predictions, wdbc_test$diagnosis)
CM <- addmargins(CM, margin = c(1, 2))
CM
```

```
##
## logistic_predictions   B   M Sum
##                  B    70   2  72
##                  M     1  41  42
##                  Sum  71  43 114
```

```
err <- mean(logistic_predictions != wdbc_test$diagnosis)
err
```

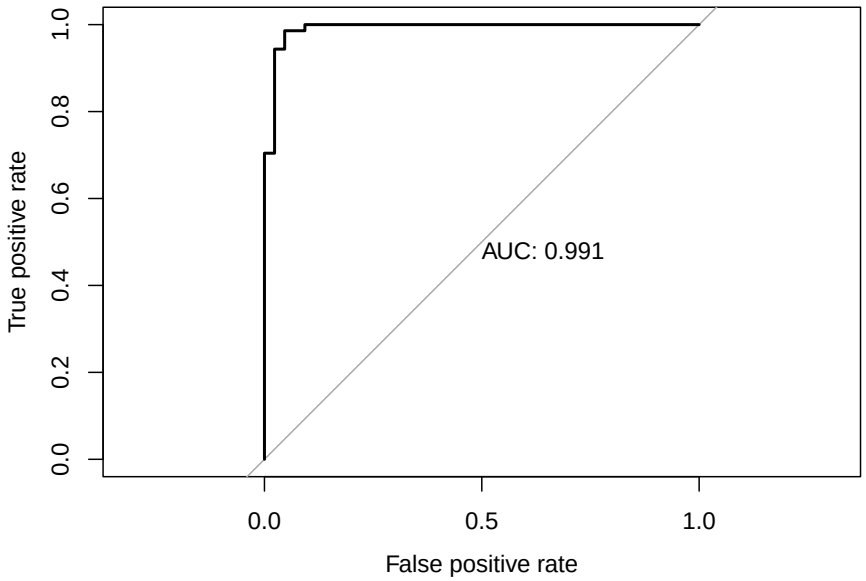
```
## [1] 0.02631579
```

```
precision <- 70/72
precision
```

```
## [1] 0.9722222
```

We printed the ROC curve:

```
## Setting direction: controls > cases
```




```
##   threshold specificity sensitivity
## 1       0.5   0.9534884   0.9859155
```

As we can see from the confusion matrix, the model has very high precision and good result in terms of specificity and sensitivity. Also the ROC curve is very close to 1 as desired.

Since we have some freedom in the choice of the threshold, we used the information provided by the ROC curve to select the best one.

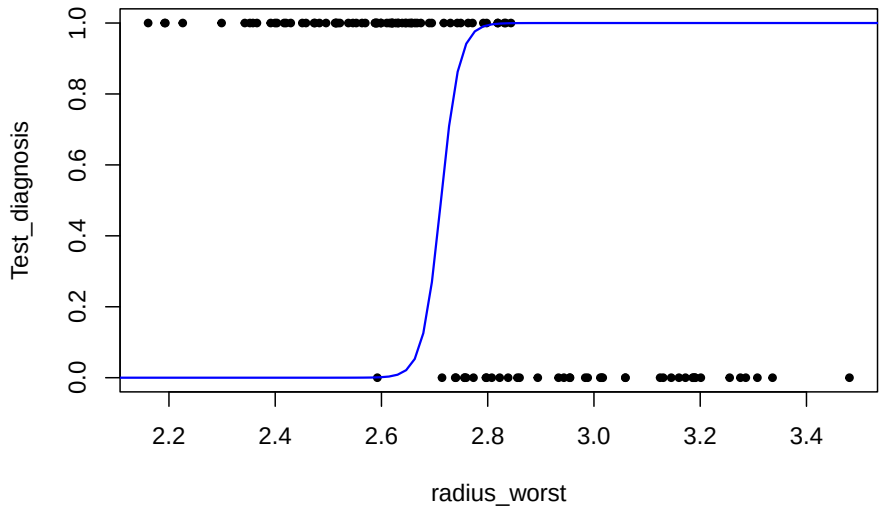
```
result <- coords(roc.out23, x = "best")
result
```

```
##   threshold specificity sensitivity
## 1 0.4588467   0.9534884   0.9859155
```

As we can notice, the value of threshold returned by the function is slightly different but at the same time the values of specificity and sensitivity don't change, so we can keep our threshold values as the best one.

```
wdbc_test$diagnosis <- ifelse(wdbc_test$diagnosis == "B", 1, 0)

plot(wdbc_test$radius_worst, wdbc_test$diagnosis, pch=20,xlab='radius_worst',
     ylab="Test_diagnosis")


# function to compute the inverse of the logit
inv.logit <- function(beta0, beta1,beta2,beta3,beta4,beta5,beta6,beta7,
                      x1,x2,x3,x4,x5,x6,x7) {
  y <- exp(beta0+beta1*(x1)+beta2*(x2)+beta3*(x3)+beta4*(x4)+beta5*(x5)+beta6*(x6)
```

```
            +beta7*(x7))
  return(y/(1+y))
}
x1 <- seq(-3, -2, length=100)#smoothness_mean
x2<- seq(-2.2, 1.1, length=100) # radius_SE
x3 <- seq(0, 0.4, length=100) # concavity_SE
x4 <- seq(-7.1, -3.52, length=100)#fractal_dim_SE
x5 <- seq(2, 3.6, length=100)#radius_worst
x6 <- seq(2.4,4, length=100)# texture_worst
x7 <- seq(0, 0.26, length=100)# concave_pts_worst

beta.hat <- coefficients(lr_model_23)
beta0.hat <- beta.hat[1]
beta1.hat <- beta.hat[2]
beta2.hat <- beta.hat[3]
beta3.hat <- beta.hat[4]
beta4.hat <- beta.hat[5]
beta5.hat <- beta.hat[6]
beta6.hat <- beta.hat[7]
beta7.hat <- beta.hat[8]

y <- inv.logit(beta0.hat, beta1.hat, beta2.hat, beta3.hat,beta4.hat,beta5.hat,
               beta6.hat,beta7.hat, x1,x2,x3,x4,x5,x6,x7)
lines(x5, y, col="blue", lwd=1.5)
```



In this plot we can see how the logistic regression manages to correctly classify the target points of the column *area_worst*, took from the test set.

K-Nearest Neighbors

The second model we decided to use is K-Nearest Neighbors (KNN).
KNN is a model used to resolve classification problems. The idea behind this model is that we choose the k nearest neighbors to the new point and select the label that belongs to the majority of the neighbors.

Since the choice of k is crucial, we decided to test different values and choose the one that produced the best result. The choice was made by calculating the error on the train set and then selecting the k that produced the smallest error. In our case the value of k was 6.

```
wdbc_train_vif <- wdbc_train_k[-c(1, 3, 4, 6, 7, 10, 12, 13, 14, 16, 18, 23, 24, 25, 26,
                                  27, 28, 30)]
wdbc_test_vif <- wdbc_test_k[-c(1, 3, 4, 6, 7, 10, 12, 13, 14, 16, 18, 23, 24, 25, 26,
                                27, 28, 30)]

kmax <- 50
err <- rep(0,kmax)
for (l in 1:kmax){
  knn_predictor <- knn(wdbc_train_vif, wdbc_test_vif, wdbc$diagnosis[sample], k=l)
  err[l] <- mean(knn_predictor != wdbc$diagnosis[-sample])
}
k <- which.min(err)
k
```

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

In the implemented model we decided not to use all the features but the one remained after the selection done using VIF. In this way, we are removing collinearity that might influence negatively the classifier.

```
knn_predictor <- knn(wdbc_train_vif, wdbc_test_vif, wdbc$diagnosis[sample], k)
err_k <- mean(knn_predictor != wdbc$diagnosis[-sample])
err_k
```

```
## [1] 0.09649123
```

Also in this case we calculated the confusion matrix.

```
CM <- table(knn_predictor, wdbc$diagnosis[-sample])
CM <- addmargins(CM, margin = c(1, 2))
CM
```

```
##
## knn_predictor   B   M Sum
##           B    68   8  76
##           M     3  35  38
##           Sum  71  43 114
```

```
precision <- 68/76
precision
```

```
## [1] 0.8947368
```

```
sensitivity <- 68/71
sensitivity
```

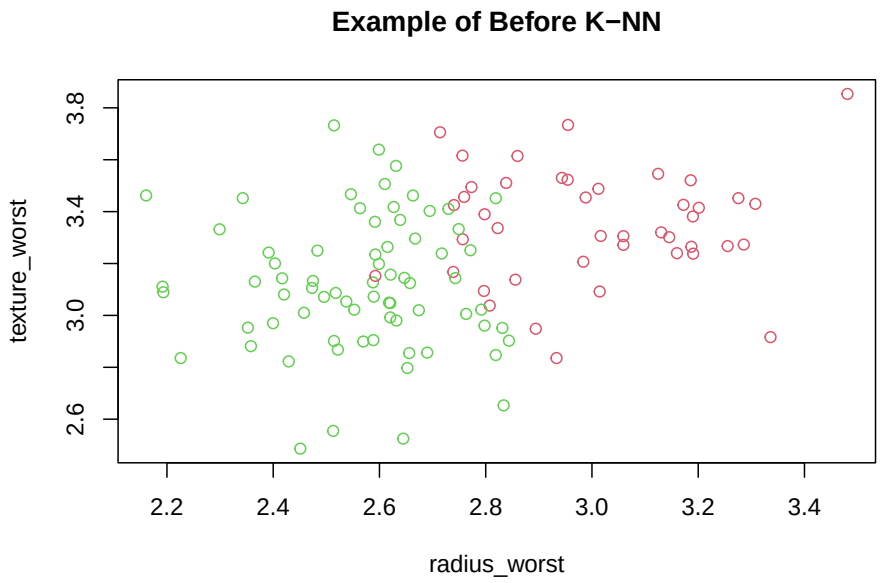
```
## [1] 0.9577465
```

```
specificity <- 35/43
specificity
```

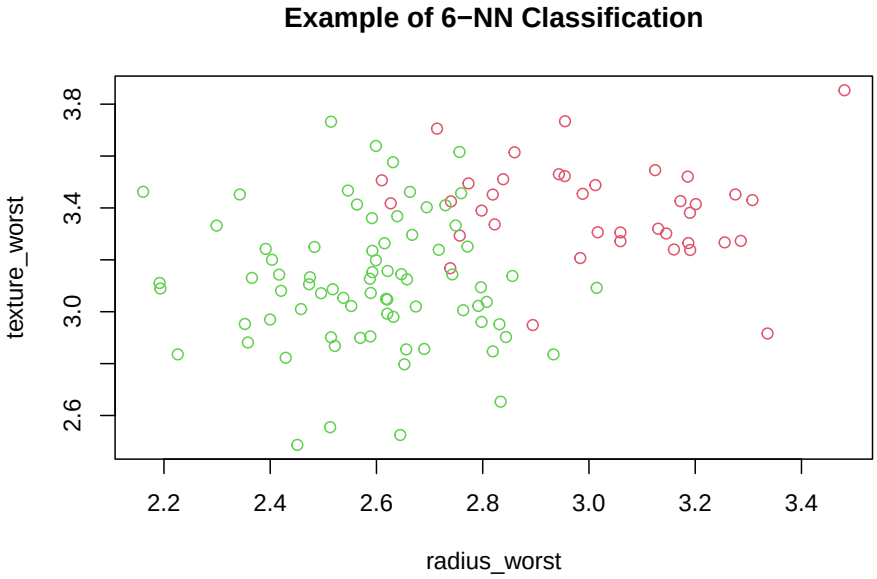
```
## [1] 0.8139535
```

If we look at the Confusion Matrix, we immediately notice that the model does some error when classifying the benign tumor.
Since we are dealing with tumors, we cannot let the model predict a tumor as benign while it is malignant. So, when we will have to compare all the models, we will have to consider this result.

```
##### B & M PLOT ####

B <-wdbc_df$diagnosis[-sample] =="B"
plot(wdbc_df$radius_worst[-sample], wdbc_df$texture_worst[-sample], col =B+2 ,
     xlab="radius_worst",ylab="texture_worst",main="Example of Before K-NN")
legend(20, 0.30, legend=c("Benign", "Malign"),
       col = c(2,3), pch = 1, cex = 0.8,
       title = "Data types", text.font=4)
```




```
BB <- knn_predictor =="B"
plot(wdbc_df$radius_worst[-sample], wdbc_df$texture_worst[-sample],
     col = BB+2,xlab="radius_worst",ylab="texture_worst",
     main="Example of 6-NN Classification")
legend(20, 0.30, legend=c("Benign", "Malign"),
       col = c(2,3), pch = 1, cex = 0.8,
       title = "Data types", text.font=4)
```



In the first plot, we can see how the values of the feature *radius_mean* are plotted according to their true value.
While, in the second plot, we can see how points are classified by the model. It is evident that the model is doing some errors and this confirms the values that we obtain in the confusion matrix.

Bayes Classifier

Bayes Classifier is a probabilistic model used for classification problems. It is based on the Bayes Theorem. In practice, we predict the probability of an event to happen given that some events occurred.
As before, we build the model using the features coming from the VIF selection.

```
bayes_cl <- naiveBayes(diagnosis ~. -area_worst - radius_mean - perimeter_mean -
                   perimeter_worst-area_SE - concavity_mean - area_mean -
                   fractal_dim_worst - compactness_worst - concavity_worst -
                   texture_SE - perimeter_SE - fractal_dim_mean - concave_pts_SE
                 - symmetry_worst - compactness_mean - smoothness_worst - compactness_SE,
                 data = wdbc_train)

# predict
predictions <- predict(bayes_cl, newdat = wdbc_test)

CM <- table(wdbc_test$diagnosis, predictions)
CM <- addmargins(CM, margin = c(1, 2))
CM
```

```
##      predictions
##         B   M Sum
##   B    69   2  71
##   M     3  40  43
##   Sum  72  42 114
```

```
precision <- 69/71
precision
```

```
## [1] 0.971831
```

```
err <- 7 / 114
err
```

```
## [1] 0.06140351
```

```
sensitivity <- 69/72
sensitivity
```

```
## [1] 0.9583333
```

```
specificity <- 40/42
specificity
```

```
## [1] 0.952381
```

Linear Discriminant Analysis (LDA)

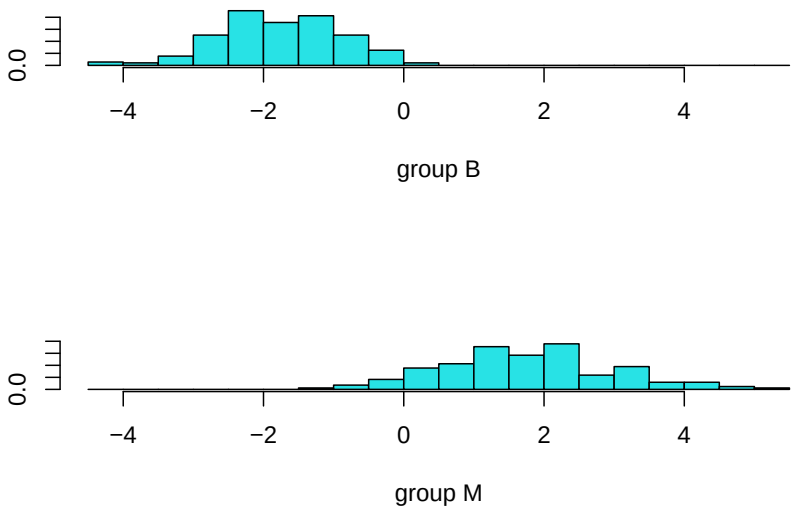
As a fourth model we used the Linear Discriminant Analysis (LDA) to do prediction.
Also in this case we tried the model with the features coming from the selection made with VIF.

```
lda_model <- lda(diagnosis ~.-area_worst - radius_mean - perimeter_mean -
                   perimeter_worst-area_SE - concavity_mean - area_mean -
                   fractal_dim_worst - compactness_worst - concavity_worst -
                   texture_SE - perimeter_SE - fractal_dim_mean - concave_pts_SE
                 - symmetry_worst - compactness_mean - smoothness_worst - compactness_SE,
                    data = wdbc_train)
lda_model
```

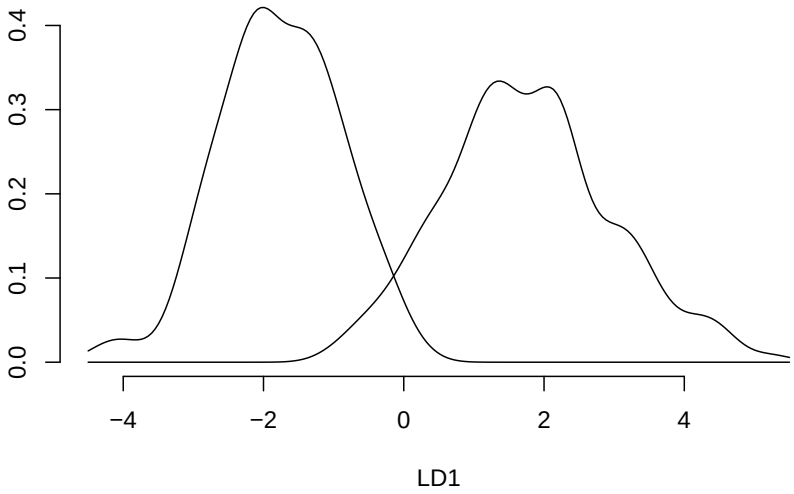
```
## Call:
## lda(diagnosis ~ . - area_worst - radius_mean - perimeter_mean -
##     perimeter_worst - area_SE - concavity_mean - area_mean -
##     fractal_dim_worst - compactness_worst - concavity_worst -
##     texture_SE - perimeter_SE - fractal_dim_mean - concave_pts_SE -
##     symmetry_worst - compactness_mean - smoothness_worst - compactness_SE,
##     data = wdbc_train)
##
## Prior probabilities of groups:
##         B         M
## 0.6285714 0.3714286
##
## Group means:
##   texture_mean smoothness_mean concave_pts_mean symmetry_mean  radius_SE
## B     2.861709       -2.392849       0.02521844     -1.762361 -1.3343108
## M     3.063533       -2.284704       0.08404616     -1.657907 -0.6263332
##   smoothness_SE concavity_SE symmetry_SE fractal_dim_SE radius_worst
## B     -5.022221   0.02468876   -3.941404      -5.816084     2.585837
## M     -5.049640   0.04052123   -3.986387      -5.636562     3.038655
##   texture_worst concave_pts_worst
## B      3.134965        0.07144043
## M      3.364480        0.16707627
##
## Coefficients of linear discriminants:
##                            LD1
## texture_mean       0.34535767
## smoothness_mean   -0.16054822
## concave_pts_mean  -0.64741100
```

```
## symmetry_mean      1.10080941
## radius_SE          0.57397426
## smoothness_SE      0.45644163
## concavity_SE      -6.95341580
## symmetry_SE       -0.01774347
## fractal_dim_SE    -0.13822009
## radius_worst       2.65811356
## texture_worst      1.32828108
## concave_pts_worst 17.10773724
```

```
plot(lda_model)
```




```
plot(lda_model, type="density")
```



In the first plot we can see that the coefficients of the linear discriminant are distributed using a histogram. The two groups are well separated and overlap just a bit, which is more visible if we look at the second plot.

As before we computed the Confusion Matrix, the error and the value of the precision

```
lda_pred<-predict(lda_model,wdbc_test,type = "response")

lda_class <- lda_pred$class
CM <-table(lda_class, wdbc_test$diagnosis)
CM <- addmargins(CM, margin = c(1, 2))
CM
```

```
##
## lda_class   B   M Sum
##       B    70   4  74
##       M     1  39  40
##       Sum  71  43 114
```

```
err<-mean(lda_class!=wdbc_test$diagnosis)
err
```

```
## [1] 0.04385965
```

```
precision <- 70/74
precision
```

```
## [1] 0.9459459
```

```
sensitivity <- 70/71
sensitivity
```

```
## [1] 0.9859155
```

```
specificity <- 39/43
specificity
```

```
## [1] 0.9069767
```

In this case, we obtain good results in classification and fothe the precision.

Quadratic Discriminant Analysis (QDA)

The last model we decided to implement is the Quadratic Discriminant Analysis (QDA).
As for LDA, we used the features coming from the selection made with VIF.

```
qda_model <- qda(diagnosis ~.,  data = wdbc_train)
qda_model
```

```
## Call:
## qda(diagnosis ~ ., data = wdbc_train)
##
## Prior probabilities of groups:
##         B         M
## 0.6285714 0.3714286
##
## Group means:
##   radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## B    2.488953     2.861709       4.349197  6.098867       -2.392849
## M    2.852195     3.063533       4.739089  6.835952       -2.284704
##   compactness_mean concavity_mean concave_pts_mean symmetry_mean
## B        -2.606204     0.04407655       0.02521844     -1.762361
## M        -2.005713     0.14740064       0.08404616     -1.657907
##   fractal_dim_mean  radius_SE texture_SE perimeter_SE  area_SE smoothness_SE
## B        -2.772684 -1.3343108 0.09207004    0.6259797 2.971765     -5.022221
## M        -2.782249 -0.6263332 0.11436657    1.3291390 4.066259     -5.049640
##   compactness_SE concavity_SE concave_pts_SE symmetry_SE fractal_dim_SE
## B      -4.063820   0.02468876    0.009671938   -3.941404      -5.816084
## M      -3.584061   0.04052123    0.014939670   -3.986387      -5.636562
##   radius_worst texture_worst perimeter_worst area_worst smoothness_worst
## B     2.585837      3.134965        4.458142   6.288434        -2.092382
## M     3.038655      3.364480        4.937717   7.193494        -1.941392
##   compactness_worst concavity_worst concave_pts_worst symmetry_worst
## B         -1.818705       0.1480151        0.07144043      -1.320701
## M         -1.089816       0.3624660        0.16707627      -1.156465
##   fractal_dim_worst
## B         -2.546581
## M         -2.423889
```

```
qda_pred<-predict(qda_model,wdbc_test,type = "response")

qda_class <- qda_pred$class
CM <- table(qda_class,wdbc_test$diagnosis)
CM <- addmargins(CM, margin = c(1, 2))
CM
```

```
##
## qda_class   B   M Sum
##       B    68   6  74
##       M     3  37  40
##       Sum  71  43 114
```

```
err<-mean(qda_class!=wdbc_test$diagnosis)
err
```

```
## [1] 0.07894737
```

```
precision <- 68/74
precision
```

```
## [1] 0.9189189
```

```
sensitivity <- 68/71
sensitivity
```

```
## [1] 0.9577465
```

```
specificity <- 37/43
specificity
```

```
## [1] 0.8604651
```

As for KNN, also in this case the model is doing some errors when classifying the benign tumor.

Final considerations

After implementing five models, we had to decide which one was the more accurate.
We have to remember that we are dealing with a tumor classification problem, so we want that our model is able to correctly classify all the tumors. If it makes some mistakes we prefer to have these belonging to the II Type error. In essence we prefer that our model classifies a tumor as Malignant while in reality it is benignant.
Let's see all the results we obtained in a table

	LOGISTIC REGRESSION	KNN	BAYES CLASSIFIER	LDA	QDA
ERROR	0.02631579	0.09649123	0.02816901	0.04385965	0.07894737
PRECISION	0.9722222	0.8947368	0.971831	0.9459459	0.9189189
SENSITIVITY	0.9859155	0.9577465	0.9583333	0.9859155	0.9577465
SENSIBILITY	0.9534884	0.8139535	0.952381	0.9069767	0.8604651

Looking at the value of the error we can see that the two models that have the smallest error are Logistic Regression and Bayes Classifier. This implies that they have also the highest value of precision.
As said before, we want that our models are able to correctly classifies the tumors, this implies that we want to have the values of sensitivity and sensibility as closed to one as possible. Looking at sensitivity, the two best models are Logistic Regression and LDA, while looking at the sensibility the two best models are Logistic Regression and Bayes Classifier.
Since Logistic regression is the model that has all the best values, we can conclude that, in this case, it is the best model that we can use to try to predict the nature of a tumor.