**ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO**

## INGENIERIA BIOMÉDICA

# PROYECTO MINERIA DE DATOS EN BIOMEDICINA
# PRACTICA 2: MACHINE LEARNING

**Desirée Rivera Rodríguez**

**Curso 2022-2023**

# 1. INTRODUCCIÓN

En el siguiente proyecto de Machine Learning se desarrollará un modelo para predecir si un paciente tiene cáncer de mama o no a través de los datos de una biopsia. Para ello se implementará algoritmos de aprendizaje tanto supervisados (KNN, SVM, Naive Bayes y Arboles de Decisión) como no supervisado (K-Medias) para poder determinarla contaremos con un conjunto de datos para analizar, limpiar y entrenar el modelo y testear cuales obtienen mejores resultados.

Para conseguir que uno o más algoritmos tengan mejores resultados en sus predicciones, se someterán los datos a un análisis estadístico y tratamientos a los datos con la finalidad de adecuarlos para que los algoritmos hagan su trabajo de manera más efectiva.

La base de datos que ha sido elegida trata de la detección del cáncer de mama en los hospitales de Winsconsin, Estados Unidos. Estos datos se han obtenido a través de un diagnóstico hecho por biopsia. En este método se examinan pequeñas cantidades de tejido del tumor para obtener las características de las células individuales y propiedades contextuales importantes gracias a la cuales se ha logrado un diagnóstico exitoso.

A partir de este proceso, 10 características son modeladas numéricamente.

- **Radio**: el radio de un núcleo individual es medido promediando la longitud de la línea del segmento radial definido por centroide del límite y los puntos individuales del límite.
- **Perímetro**: es la distancia total entre los puntos del límite constituyen el perímetro nuclear.
- **Área**: se mide contando el número de pixeles en el interior de los límites y se añade la mitad de los píxeles del perímetro.
- **Compacidad**: el perímetro y el área son combinados para dar una medida de la compacidad del núcleo de la célula.
- **Suavidad**: es la diferencia entre la longitud de una línea radial y la longitud media de las líneas alrededor de esta.
- **Concavidad**: irregularidades de forma en el núcleo de la célula
- **Puntos cóncavos**: número de la concavidad del contorno.
- **Simetría**: se obtiene encontrando la línea más larga que pase por el centro de la célula.
- **Dimensión fractal**: es una característica de forma, es decir, a mayor valor corresponde a un menor contorno y por tanto a una mayor probabilidad malignidad.

- **Textura:** El valor medio, extremo (el más largo) y el error standard de cada característica son computados para cada imagen. Los valores extremos son más intuitivamente útiles para el problema en cuestión, ya que solo algunas células malignas pueden ocurrir en un ejemplo dado.

A partir de estas variables, se quiere predecir la variable de diagnosis, es decir si el tumor es benigno o maligno. Para ello se entrenarán algoritmos tanto supervisado como no supervisado que puedan identificar las dos clases

# 2. INSERTAMOS EL DATASET

Una vez obtenida esta base de datos lo primero que hacemos es el procesamiento de datos para ello cargamos los datos a utilizar:

```
#Procesamiento de datos
setwd("C:/Users/desir/OneDrive/Documentos/1.Ingenieria biomedica/3º/Mineria/Entrega_Algoritmos")
dat <- read.csv("data.csv")
```

# 3. ESTADÍSTICA DESCRIPTIVA

Procedemos a estudiar nuestros datos para ello visualizamos los datos de varias maneras diferentes.

La primera será ver la estructura de los datos, es decir que tipos de datos tenemos, el tamaño de la base de datos y la cantidad de variables.

```
> str(dat)
'data.frame':   569 obs. of  33 variables:
 $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844981 84501001 ...
 $ diagnosis              : chr  "M" "M" "M" "M" ...
 $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
 $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
 $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
 $ area_mean              : num  1001 1326 1203 386 1297 ...
 $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
 $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
 $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
 $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
 $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
 $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
 $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
 $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
 $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
 $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
 $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
 $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
 $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
 $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
 $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
 $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
 $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
 $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
 $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
 $ area_worst             : num  2019 1956 1709 568 1575 ...
 $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
 $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
 $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
 $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
 $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
 $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
 $ x                      : logi  NA NA NA NA NA NA ...
>
```

De estas informaciones podemos decir que nuestra base tiene 569 muestras de tejido de pacientes distintos con 33 variables cada uno. Siendo 30 variables cuantitativas y dos

cualitativas. De estas 33 variables podemos ver que la primera variable es el ID la cual no nos aporta mucha información así que la eliminaremos. Asimismo, vemos que la variable 33 sus datos son NA, los cuales tampoco en aportan mucha información por lo tanto también lo eliminamos.

```
dat <- dat[,-1]
dat <- dat[,-32]
```

Como se puede comprobar los datos se pueden dividir en tres partes: la media, el error estándar y el "peor valor" de las diez variables que se han mencionado anteriormente. Para seguir con el análisis descriptivo y observar los resultados con más claridad se dividirá en las tres categorías.

```
features_mean <- dat[,2:11]
features_se <- dat[,12:21]
features_worst <- dat[,22:31]
```

Ahora miramos las estadísticas generales de las características del conjunto de datos. Estos serán la media, mediana, desviación típica, curtosis, rango, los cuartiles entre otros. Primero lo hacemos con la primera categoría:

```
> summary(features_worst)
  radius_worst   texture_worst   perimeter_worst    area_worst     smoothness_worst  compactness_worst concavity_worst  concave.points_worst
 Min.   : 7.93   Min.   :12.02   Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729   Min.   :0.0000   Min.   :0.00000
 1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145   1st Qu.:0.06493
 Median :14.97   Median :25.41   Median : 97.66   Median : 686.5   Median :0.13130   Median :0.21190   Median :0.2267   Median :0.09993
 Mean   :16.27   Mean   :25.68   Mean   :107.26   Mean   : 880.6   Mean   :0.13237   Mean   :0.25427   Mean   :0.2722   Mean   :0.11461
 3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829   3rd Qu.:0.16140
 Max.   :36.04   Max.   :49.54   Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800   Max.   :1.2520   Max.   :0.29100
 symmetry_worst   fractal_dimension_worst
 Min.   :0.1565   Min.   :0.05504
 1st Qu.:0.2504   1st Qu.:0.07146
 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
```

```
> describe(features_mean)
                        vars   n   mean     sd median trimmed    mad    min     max   range skew kurtosis   se
radius_mean                1 569  14.13   3.52  13.37   13.82   2.82   6.98   28.11   21.13 0.94     0.81 0.15
texture_mean               2 569  19.29   4.30  18.84   19.04   4.17   9.71   39.28   29.57 0.65     0.73 0.18
perimeter_mean             3 569  91.97  24.30  86.24   89.74  18.84  43.79  188.50  144.71 0.99     0.94 1.02
area_mean                  4 569 654.89 351.91 551.10  606.13 227.28 143.50 2501.00 2357.50 1.64     3.59 14.75
smoothness_mean            5 569   0.10   0.01   0.10    0.10   0.01   0.05    0.16    0.11 0.45     0.82 0.00
compactness_mean           6 569   0.10   0.05   0.09    0.10   0.05   0.02    0.35    0.33 1.18     1.61 0.00
concavity_mean             7 569   0.09   0.08   0.06    0.08   0.06   0.00    0.43    0.43 1.39     1.95 0.00
concave.points_mean        8 569   0.05   0.04   0.03    0.04   0.03   0.00    0.20    0.20 1.17     1.03 0.00
symmetry_mean              9 569   0.18   0.03   0.18    0.18   0.03   0.11    0.30    0.20 0.72     1.25 0.00
fractal_dimension_mean    10 569   0.06   0.01   0.06    0.06   0.01   0.05    0.10    0.05 1.30     2.95 0.00
```

Ahora de la segunda categoría:

```
> summary(features_se)
   radius_se        texture_se      perimeter_se       area_se
 Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
 1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850
 Median :0.3242   Median :1.1080   Median : 2.287   Median : 24.530
 Mean   :0.4052   Mean   :1.2169   Mean   : 2.866   Mean   : 40.337
 3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190
 Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200
 smoothness_se      compactness_se     concavity_se     concave.points_se
 Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.   :0.000000
 1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638
 Median :0.006380   Median :0.020450   Median :0.02589   Median :0.010930
 Mean   :0.007041   Mean   :0.025478   Mean   :0.03189   Mean   :0.011796
 3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710
 Max.   :0.031130   Max.   :0.135400   Max.   :0.39600   Max.   :0.052790
  symmetry_se       fractal_dimension_se
 Min.   :0.007882   Min.   :0.0008948
 1st Qu.:0.015160   1st Qu.:0.0022480
 Median :0.018730   Median :0.0031870
 Mean   :0.020542   Mean   :0.0037949
 3rd Qu.:0.023480   3rd Qu.:0.0045580
 Max.   :0.078950   Max.   :0.0298400
```

```
> describe(features_se)
                    vars   n  mean    sd median trimmed   mad  min    max  range skew kurtosis   se
radius_se              1 569  0.41  0.28   0.32    0.36  0.16 0.11   2.87   2.76 3.07    17.45 0.01
texture_se             2 569  1.22  0.55   1.11    1.16  0.47 0.36   4.88   4.52 1.64     5.26 0.02
perimeter_se           3 569  2.87  2.02   2.29    2.51  1.14 0.76  21.98  21.22 3.43    21.12 0.08
area_se                4 569 40.34 45.49  24.53   31.69 13.63 6.80 542.20 535.40 5.42    48.59 1.91
smoothness_se          5 569  0.01  0.00   0.01    0.01  0.00 0.00   0.03   0.03 2.30    10.32 0.00
compactness_se         6 569  0.03  0.02   0.02    0.02  0.01 0.00   0.14   0.13 1.89     5.02 0.00
concavity_se           7 569  0.03  0.03   0.03    0.03  0.02 0.00   0.40   0.40 5.08    48.24 0.00
concave.points_se      8 569  0.01  0.01   0.01    0.01  0.01 0.00   0.05   0.05 1.44     5.04 0.00
symmetry_se            9 569  0.02  0.01   0.02    0.02  0.01 0.01   0.08   0.07 2.18     7.78 0.00
fractal_dimension_se  10 569  0.00  0.00   0.00    0.00  0.00 0.00   0.03   0.03 3.90    25.94 0.00
```

Y por último la tercera categoría:

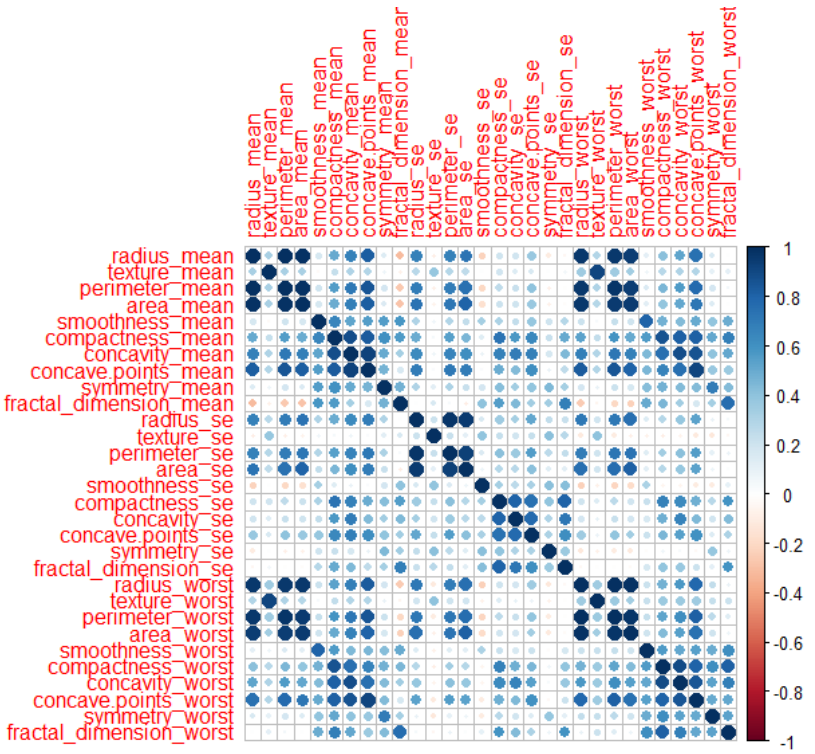
```
> describe(features_worst)
                     vars   n   mean     sd median trimmed    mad    min     max   range skew kurtosis   se
radius_worst            1 569  16.27   4.83  14.97   15.73   3.65   7.93   36.04   28.11 1.10     0.91 0.20
texture_worst           2 569  25.68   6.15  25.41   25.39   6.42  12.02   49.54   37.52 0.50     0.20 0.26
perimeter_worst         3 569 107.26  33.60  97.66  103.42  25.01  50.41  251.20  200.79 1.12     1.04 1.41
area_worst              4 569 880.58 569.36 686.50  788.02 319.65 185.20 4254.00 4068.80 1.85     4.32 23.87
smoothness_worst        5 569   0.13   0.02   0.13    0.13   0.02   0.07    0.22    0.15 0.41     0.49 0.00
compactness_worst       6 569   0.25   0.16   0.21    0.23   0.13   0.03    1.06    1.03 1.47     2.98 0.01
concavity_worst         7 569   0.27   0.21   0.23    0.25   0.20   0.00    1.25    1.25 1.14     1.57 0.01
concave.points_worst    8 569   0.11   0.07   0.10    0.11   0.07   0.00    0.29    0.29 0.49    -0.55 0.00
symmetry_worst          9 569   0.29   0.06   0.28    0.28   0.05   0.16    0.66    0.51 1.43     4.37 0.00
fractal_dimension_worst 10 569  0.08   0.02   0.08    0.08   0.01   0.06    0.21    0.15 1.65     5.16 0.00
```

```
> summary(features_worst)
  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.points_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_dimension_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
```

Teniendo estos datos, si nos fijamos en el valor máximo y mínimo de cada una de las variables nos damos cuenta de que cada una de las variables tienen escalas diferentes lo cual puede producir imprecisiones. Por consiguiente, hay que tenerlo en consideración para normalizar los datos al aplicar algunos algoritmos de aprendizaje.

Pasamos a ver la correlación entre las variables para ello primero quitamos nuestra variable predictora:

```
datos_cancer_correlacion = dat[,-1]
corrplot(cor(datos_cancer_correlacion))
```

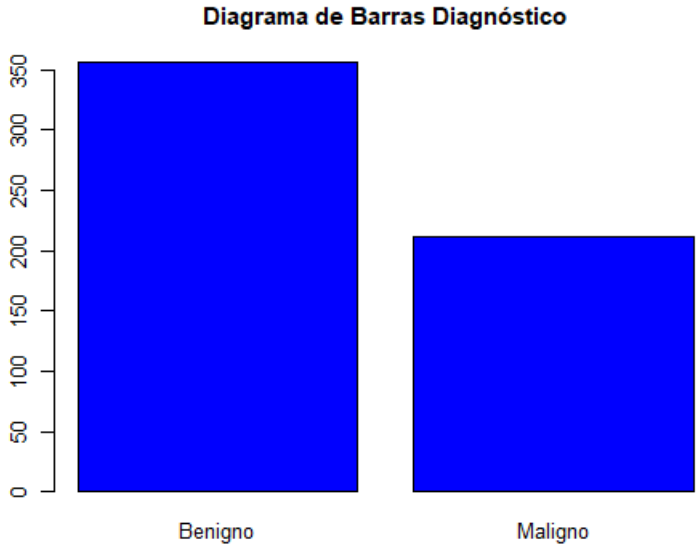


Como podemos observar la misma variable en las tres categorías tiene unos valores de correlación alto como era de esperar. Asimismo, en las tres categorías podemos ver que el radio, el perímetro y el área tiene una alta correlación en las tres categorías. De la misma forma, podemos visualizar que la concavidad, compacidad y los puntos cóncavos tienen también una alta correlación.

Otra visualización interesante es ver los datos estadísticos de la variable predictora, es decir de diagnósticos. Además, convertimos la variable de predictora en factor para que tenga etiquetas.

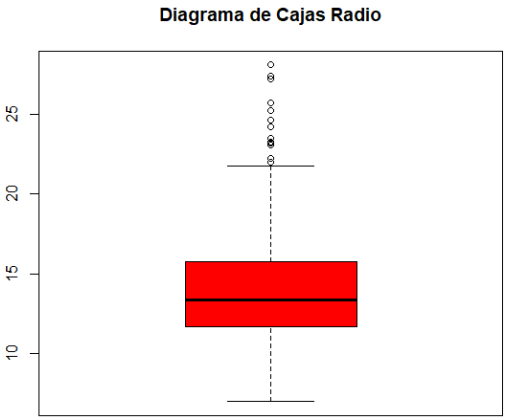
```
#Convertimos en factor la variable a predecir
dat$diagnosis <- factor(dat$diagnosis, levels= c("B","M"), labels = c("Benigno","Maligno"))
#Vemos la proporción de los posibles valores de nuestra variable a predecir
table(dat$diagnosis)
prop.table(table(dat$diagnosis))
barplot(table(na.omit(dat$diagnosis)), main= "Diagrama de Barras Diagnóstico", col= "blue")
```

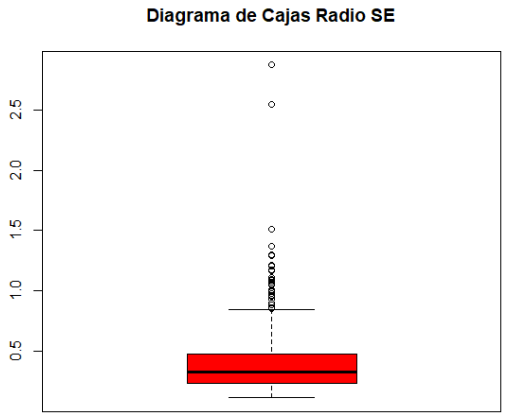
```
> table(dat$diagnosis)

Benigno Maligno
    357     212
> prop.table(table(dat$diagnosis))

  Benigno   Maligno
0.6274165 0.3725835
```

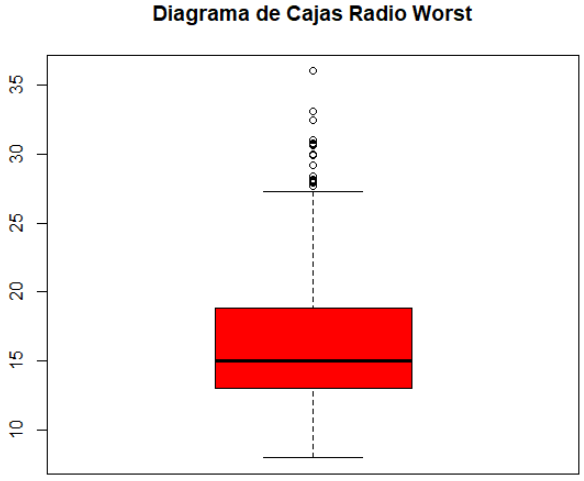



Por último, se muestra de la dispersión de los datos y datos atípicos por medio de un gráfico de caja.

```r
boxplot(dat$radius_mean,main= "Diagrama de Cajas Radio", col= "red")
boxplot(dat$texture_mean,main= "Diagrama de Cajas Textura", col= "red")
boxplot(dat$perimeter_mean,main= "Diagrama de Cajas Perimetro", col= "red")
boxplot(dat$area_mean,main= "Diagrama de Cajas Area", col= "red")
boxplot(dat$smoothness_mean,main= "Diagrama de Cajas Smoothness", col= "red")
boxplot(dat$compactness_mean,main= "Diagrama de Cajas Compactness", col= "red")
boxplot(dat$concavity_mean,main= "Diagrama de Cajas Concavity", col= "red")
boxplot(dat$concave.points_mean,main= "Diagrama de Cajas Concave Points", col= "red")
boxplot(dat$symmetry_mean,main= "Diagrama de Cajas simetria", col= "red")
boxplot(dat$fractal_dimension_mean,main= "Diagrama de Cajas Dimension Fractal", col= "red")

boxplot(dat$radius_se,main= "Diagrama de Cajas Radio SE", col= "red")
boxplot(dat$texture_se,main= "Diagrama de Cajas Textura SE", col= "red")
boxplot(dat$perimeter_se,main= "Diagrama de Cajas Perimetro SE", col= "red")
boxplot(dat$area_se,main= "Diagrama de Cajas Area SE", col= "red")
boxplot(dat$smoothness_se,main= "Diagrama de Cajas Smoothness SE", col= "red")
boxplot(dat$compactness_se,main= "Diagrama de Cajas Compactness SE", col= "red")
boxplot(dat$concavity_se,main= "Diagrama de Cajas Concavity SE", col= "red")
boxplot(dat$concave.points_se,main= "Diagrama de Cajas Concave Points SE", col= "red")
boxplot(dat$symmetry_se,main= "Diagrama de Cajas Simetria SE", col= "red")
boxplot(dat$fractal_dimension_se,main= "Diagrama de Cajas Dimension Fractal SE", col= "red")


boxplot(dat$radius_worst,main= "Diagrama de Cajas Radio Worst", col= "red")
boxplot(dat$texture_worst,main= "Diagrama de Cajas Textura Worst", col= "red")
boxplot(dat$perimeter_worst,main= "Diagrama de Cajas Perimetro Worst", col= "red")
boxplot(dat$area_worst,main= "Diagrama de Cajas Area Worst", col= "red")
boxplot(dat$smoothness_worst,main= "Diagrama de Cajas Smoothness Worst", col= "red")
boxplot(dat$compactness_worst,main= "Diagrama de Cajas Compactness Worst", col= "red")
boxplot(dat$concavity_worst,main= "Diagrama de Cajas Concavity Worst", col= "red")
boxplot(dat$concave.points_worst,main= "Diagrama de Cajas Concave Points Worst", col= "red")
boxplot(dat$symmetry_worst,main= "Diagrama de Cajas Simetria Worst", col= "red")
boxplot(dat$fractal_dimension_worst,main= "Diagrama de Cajas Dimension Fractal Worst", col= "red")
```



Aquí pongo un ejemplo de la variable radio en sus tres categorías, pero al ejecutar todo el código podemos ver que las 30 variables que vamos a utilizar para predecir la enfermedad tienen outliers que pueden influir.

# 4. ALGORITMOS DE APRENDIZAJE SUPERVISADO.

Primero aplicaremos algoritmos de aprendizaje supervisado los cuales trabajan con datos etiquetados intentado encontrar una función que con unas variables de entrada se les asigne una etiqueta de salida adecuada. En nuestro caso usaremos estos algoritmos con la finalidad de diagnosticar una enfermedad, por lo tanto, nos encontramos ante un problema de clasificación. En este apartado solo nos centraremos en las medidas de rendimiento de cada uno de los algoritmos elegidos.

- **KNN:**

  Es un método el cual busca en las observaciones más cercanas a la que se está tratando de predecir y clasifica el punto de interés basado en la mayoría de los datos que le rodean.

  Primero calcula la distancia entre la observación que se quiere clasificar y el resto de observaciones que pertenecen a la muestra de entrenamiento. Para ello la importancia de normalizar. En segundo lugar, se selecciona los k elementos más cercanos, es decir con menor distancia. Por último, se mira la clase a la que pertenece según la cantidad de k más cercana

  En primer lugar, se probó con un k =21. Al hacer nuestra validación de nuestro algoritmo vemos que tiene una alta precisión con un nivel de Kappa cercano a 1 lo que significa los resultados obtenidos no se han dado por “casualidad”.

  De la misma manera tendremos la medida F que es la combinación de las medidas de precisión y sensibilidad. Con esta medida nos será más fácil comparar los distintos algoritmos. En este caso es muy alta.

```
> confusionMatrix(df_test_labels,dat_test_pred)
Confusion Matrix and Statistics

          Reference
Prediction Benigno Maligno
   Benigno       97       1
   Maligno        8      65

               Accuracy : 0.9474
                 95% CI : (0.9024, 0.9757)
    No Information Rate : 0.614
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8911

 Mcnemar's Test P-Value : 0.0455

            Sensitivity : 0.9238
            Specificity : 0.9848
         Pos Pred Value : 0.9898
         Neg Pred Value : 0.8904
             Prevalence : 0.6140
         Detection Rate : 0.5673
   Detection Prevalence : 0.5731
      Balanced Accuracy : 0.9543

       'Positive' Class : Benigno
```

```
> precision(data = df_test_labels, reference = dat_test_pred, relevant = "Benigno")
[1] 0.9897959
> recall(data = df_test_labels, reference = dat_test_pred, relevant = "Benigno")
[1] 0.9238095
> F_meas(data = df_test_labels, reference = dat_test_pred, relevant = "Benigno")
[1] 0.955665
```

Para ver si se podía mejorar el algoritmo hicimos un bucle y nos percatamos que k=7 es el que mejores valores de las medidas de rendimiento tiene dado que acierta más verdaderos negativos o errores subiendo tanto sus valores de especificidad como sensibilidad. Asimismo, se obtiene mejores resultados en la medida F.

```
> confusionMatrix(df_test_labels,dat_test_pred)
Confusion Matrix and Statistics

          Reference
Prediction Benigno Maligno
   Benigno       97       1
   Maligno        5      68

               Accuracy : 0.9649
                 95% CI : (0.9252, 0.987)
    No Information Rate : 0.5965
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9278

 Mcnemar's Test P-Value : 0.2207

            Sensitivity : 0.9510
            Specificity : 0.9855
         Pos Pred Value : 0.9898
         Neg Pred Value : 0.9315
             Prevalence : 0.5965
         Detection Rate : 0.5673
   Detection Prevalence : 0.5731
      Balanced Accuracy : 0.9682

       'Positive' Class : Benigno
```

```
> precision(data = df_test_labels, reference = dat_test_pred, relevant = "Benigno")
[1] 0.9897959
> recall(data = df_test_labels, reference = dat_test_pred, relevant = "Benigno")
[1] 0.9509804
> F_meas(data = df_test_labels, reference = dat_test_pred, relevant = "Benigno")
[1] 0.97
```

Por último, probamos si usando la tipificación en lugar de la normalización se mejora los resultados con el mismo k vecinos más cercanos.

```
> precision(data = dat_test_labels, reference = dat_test_pred, relevant = "Benigno")
[1] 0.9693878
> recall(data = dat_test_labels, reference = dat_test_pred, relevant = "Benigno")
[1] 0.95
> F_meas(data = dat_test_labels, reference = dat_test_pred, relevant = "Benigno")
[1] 0.959596
> confusionMatrix(dat_test_labels,dat_test_pred)
Confusion Matrix and Statistics

          Reference
Prediction Benigno Maligno
   Benigno      95       3
   Maligno       5      68

               Accuracy : 0.9532
                 95% CI : (0.9099, 0.9796)
    No Information Rate : 0.5848
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9041

 Mcnemar's Test P-Value : 0.7237

            Sensitivity : 0.9500
            Specificity : 0.9577
         Pos Pred Value : 0.9694
         Neg Pred Value : 0.9315
             Prevalence : 0.5848
         Detection Rate : 0.5556
   Detection Prevalence : 0.5731
      Balanced Accuracy : 0.9539

       'Positive' Class : Benigno
```

Viendo las medidas de rendimientos vemos que es un muy buen modelo, pero nos quedamos con el modelo normalizado con k=7 al darnos mejores medidas de rendimiento.

- **SVM:**
  Las Máquinas de Vector Soporte se basa en el concepto de hiperplano. En este modelo se busca de un hiperplano que separe de forma óptima a los puntos de una clase de la de otra, que eventualmente han podido ser previamente proyectados a un espacio de dimensionalidad superior. La característica fundamental es buscar el hiperplano que tenga la máxima distancia (margen) con los puntos que estén más cerca de él mismo. De esta forma, los puntos del vector que son etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.

Primero probamos con el parámetro de Kernel ‘vanilladot’ y analizamos las siguientes medidas de rendimiento

```
> confusionMatrix(df_test$diagnosis, df_predictions)
Confusion Matrix and Statistics

          Reference
Prediction Benigno Maligno
   Benigno      97       1
   Maligno       2      71

               Accuracy : 0.9825
                 95% CI : (0.9496, 0.9964)
    No Information Rate : 0.5789
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9641

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9798
            Specificity : 0.9861
         Pos Pred Value : 0.9898
         Neg Pred Value : 0.9726
             Prevalence : 0.5789
         Detection Rate : 0.5673
   Detection Prevalence : 0.5731
      Balanced Accuracy : 0.9830

       'Positive' Class : Benigno

> precision(data = df_test$diagnosis, reference = df_predictions, relevant = "Benigno")
[1] 0.9897959
> recall(data = df_test$diagnosis, reference = df_predictions, relevant = "Benigno")
[1] 0.979798
> F_meas(data = df_test$diagnosis, reference = df_predictions, relevant = "Benigno")
[1] 0.9847716
```

AL hacer nuestra validación podemos observar que tiene una precisión muy cercana a 1 por lo que se puede decir que predice si una persona tiene cáncer o no de manera casi segura. Con el valor del coeficiente de Kappa se puede demostrar que estas predicciones no se dan por “casualidad”. Junto con la medida F y la precisión podemos decir que es un modelo muy bueno.

De la misma forma vamos a probar si mejora aun más el modelo cambiando el parámetro de Kernel a rbfdot

```
> confusionMatrix(df_test$diagnosis, df_predictions_rbf)
Confusion Matrix and Statistics

          Reference
Prediction Benigno Maligno
   Benigno      98       0
   Maligno       3      70

               Accuracy : 0.9825
                 95% CI : (0.9496, 0.9964)
    No Information Rate : 0.5906
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.964

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9703
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9589
             Prevalence : 0.5906
         Detection Rate : 0.5731
   Detection Prevalence : 0.5731
      Balanced Accuracy : 0.9851

       'Positive' Class : Benigno

> precision(data = df_test$diagnosis, reference = df_predictions_rbf, relevant = "Benigno")
[1] 1
> recall(data = df_test$diagnosis, reference = df_predictions_rbf, relevant = "Benigno")
[1] 0.970297
> F_meas(data = df_test$diagnosis, reference = df_predictions_rbf, relevant = "Benigno")
[1] 0.9849246
```

Aquí vemos que tanto la accuracy como el coeficiente Kappa es igual. No obstante, podemos ver que en este modelo no hay ningún falso positivo conllevando que la especificidad y la precisión sea del valor máximo. Asimismo, la medida F es un pelín más alta que en el anterior por lo que nos quedamos de momento con este modelo.

Por último, vamos a comprobar cambiando otra vez el parámetro de Kernel a "tanhdot".

```
> confusionMatrix(df_test$diagnosis, df_predictions_t)
Confusion Matrix and Statistics

          Reference
Prediction Benigno Maligno
   Benigno      90       8
   Maligno       6      67

               Accuracy : 0.9181
                 95% CI : (0.8664, 0.9545)
    No Information Rate : 0.5614
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8333

 Mcnemar's Test P-Value : 0.7893

            Sensitivity : 0.9375
            Specificity : 0.8933
         Pos Pred Value : 0.9184
         Neg Pred Value : 0.9178
             Prevalence : 0.5614
         Detection Rate : 0.5263
   Detection Prevalence : 0.5731
      Balanced Accuracy : 0.9154

       'Positive' Class : Benigno

> precision(data = df_test$diagnosis, reference = df_predictions_t, relevant = "Benigno")
[1] 0.9183673
> recall(data = df_test$diagnosis, reference = df_predictions_t, relevant = "Benigno")
[1] 0.9375
> F_meas(data = df_test$diagnosis, reference = df_predictions_t, relevant = "Benigno")
[1] 0.9278351
```

Viendo los resultados de la medida de rendimiento son buenos, pero no tanto como el del modelo que hemos usado con el parámetro de Kernel rdfdot.

- **Arboles de Decisión:**

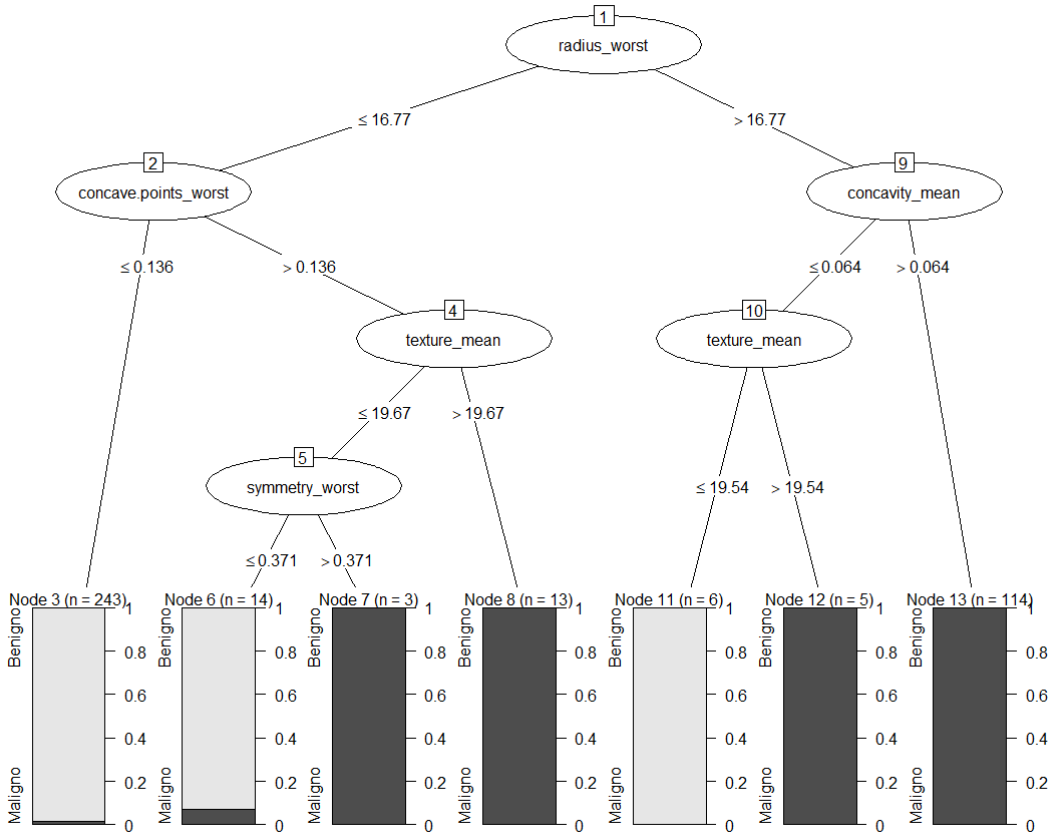
Es un gran modelo para darle un toque explicativo a las decisiones que se toman para predecir si es en nuestro caso Benigno o Maligno. Aplicando el modelo vemos su rendimiento:

```
> confusionMatrix(df_test$diagnosis, df_pred)
Confusion Matrix and Statistics

          Reference
Prediction Benigno Maligno
   Benigno      94       5
   Maligno       5      67

               Accuracy : 0.9415
                 95% CI : (0.8951, 0.9716)
    No Information Rate : 0.5789
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8801

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9495
            Specificity : 0.9306
         Pos Pred Value : 0.9495
         Neg Pred Value : 0.9306
             Prevalence : 0.5789
         Detection Rate : 0.5497
   Detection Prevalence : 0.5789
      Balanced Accuracy : 0.9400

       'Positive' Class : Benigno

> precision(data = df_test$diagnosis, reference = df_pred, relevant = "Benigno")
[1] 0.9494949
> recall(data = df_test$diagnosis, reference = df_pred, relevant = "Benigno")
[1] 0.9494949
> F_meas(data = df_test$diagnosis, reference = df_pred, relevant = "Benigno")
[1] 0.9494949
```

Podemos apreciar que tiene valores muy buenos en las medidas de rendimiento, pero vamos a ver si se puede mejorar el modelo usando la técnica de poda. Para ello utilizamos el concepto de Boosting para mejorar el modelo. En este agregaremos unos parámetros que será el número de árboles separados. Estos árboles juntaran su fortalezas y debilidades para mejorar el modelo. Probamos con 10, 20, 30 árboles y con las medidas de rendimiento vemos que el mejor modelo es el que utiliza 20 trials:

```
> confusionMatrix(df_test$diagnosis, df_pred20)
Confusion Matrix and Statistics

          Reference
Prediction Benigno Maligno
   Benigno      96       3
   Maligno       3      69

               Accuracy : 0.9649
                 95% CI : (0.9252, 0.987)
    No Information Rate : 0.5789
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.928

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9697
            Specificity : 0.9583
         Pos Pred Value : 0.9697
         Neg Pred Value : 0.9583
             Prevalence : 0.5789
         Detection Rate : 0.5614
   Detection Prevalence : 0.5789
      Balanced Accuracy : 0.9640

       'Positive' Class : Benigno

> precision(data = df_test$diagnosis, reference = df_pred20, relevant = "Benigno")
[1] 0.969697
> recall(data = df_test$diagnosis, reference = df_pred20, relevant = "Benigno")
[1] 0.969697
> F_meas(data = df_test$diagnosis, reference = df_pred20, relevant = "Benigno")
[1] 0.969697
```

Dibujamos su árbol de decisión para ver las decisiones que toma para elegir una clase u otra.



- **Naives Bayes:**

Naïve Bayes es un algoritmo de aprendizaje automático basado en el teorema de Bayes. Usa las probabilidades condicionales de las variables independientes para poder clasificar cada observación de la base de datos. Procedemos a ver las medidas de rendimiento.

```
> confusionMatrix(cod_test, dat_test_pred)
Confusion Matrix and Statistics

          Reference
Prediction Benigno Maligno
   Benigno      94       4
   Maligno       7      66

               Accuracy : 0.9357
                 95% CI : (0.8878, 0.9675)
    No Information Rate : 0.5906
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8678

 Mcnemar's Test P-Value : 0.5465

            Sensitivity : 0.9307
            Specificity : 0.9429
         Pos Pred Value : 0.9592
         Neg Pred Value : 0.9041
             Prevalence : 0.5906
         Detection Rate : 0.5497
   Detection Prevalence : 0.5731
      Balanced Accuracy : 0.9368

       'Positive' Class : Benigno

> precision(data = cod_test, reference = dat_test_pred, relevant = "Benigno")
[1] 0.9591837
> recall(data = cod_test, reference = dat_test_pred, relevant = "Benigno")
[1] 0.9306931
> F_meas(data = cod_test, reference = dat_test_pred, relevant = "Benigno")
[1] 0.9447236
```

Podemos comentar que es un modelo sencillo y muy efectivo. No obstante, para este modelo comparándolo con otros no es el mejor aunque tendría una gran utilidad.

- **Comparación**

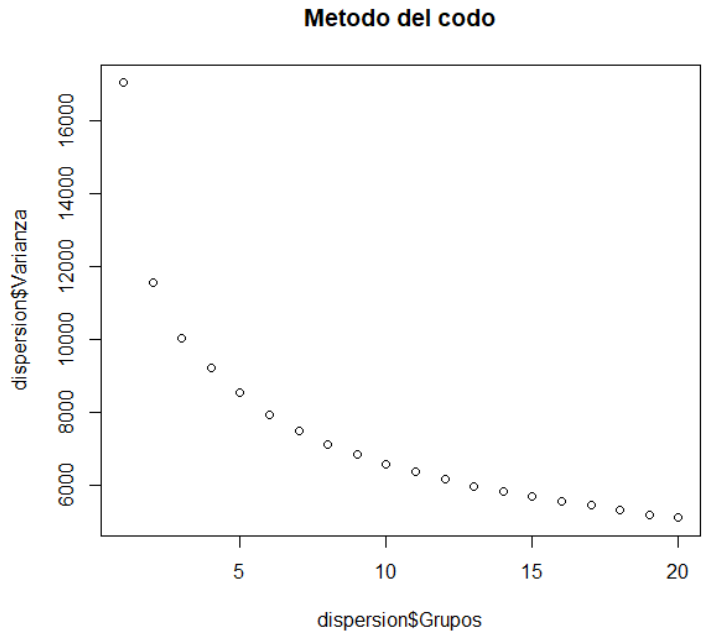
Comparando todos los modelos podemos destacar que los 4 son muy buenos algoritmo de aprendizaje para predecir el cancer de mama con los datos de la base proporcionada. Pero si nos tuviéramos que quedar con uno y analizando las medidas de rendimiento sería el SVM con un parámetro Kernel de rbfdot.

# 5. ALGORITMOS DE APRENDIZAJE SUPERVISADO.

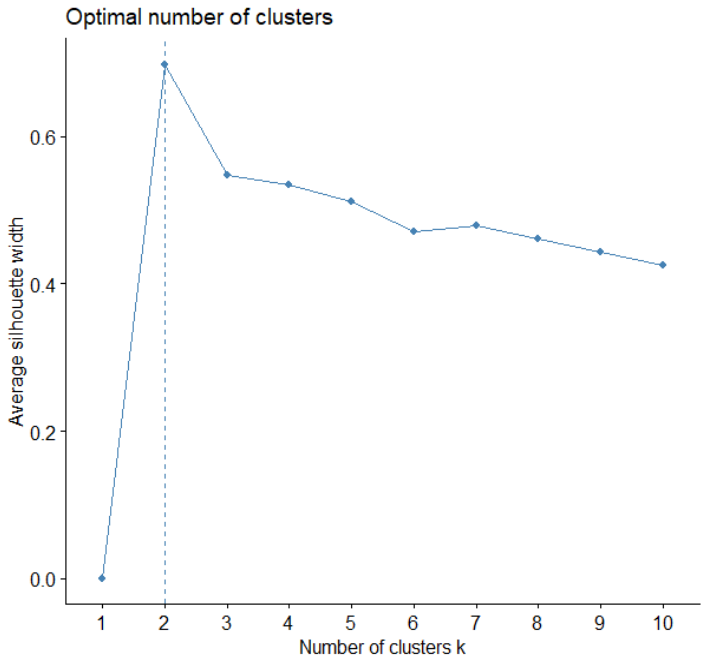
Utilizamos el K-Means que es un tipo de aprendizaje no supervisado, que se utiliza cuando se tienen datos no etiquetados, es decir, datos sin categorías o grupos definidos. El objetivo de este algoritmo es encontrar grupos en los datos. Los puntos de datos se agrupan según la similitud de características.

En este caso tenemos que suponer que la variable categórica diagnóstico no nos la dan entonces a partir de las otras variables vamos a ver si el algoritmo reconoce los dos grupos que tenemos en la variable diagnosis.

Se aplican unos centroides aleatorios y con un bucle vamos a ver la inercia intra total en la dispersión para escoger nuestros números de clusters. Al aplicar el método del codo no se ve muy bien:




Por esa razón vamos a usar el método de 'silhouette' que es más preciso. En este se puede ver que hay dos grupos que se pueden corresponder a nuestras etiquetas que hemos usado en los algoritmos supervisados:




Por último, vemos nuestros clusters: