Redes Neuronales

Daniela Cuesta - Paola Peralta Flores - Mariuxi Tenesaca

El código lee un conjunto de datos desde un archivo llamado "wdbc.data" utilizando la función **`read.table()`**.

```r
dataset <- read.table(file = "wdbc.data", header = FALSE, sep = ",")
head(dataset)
```

```
        V1 V2    V3    V4     V5     V6      V7      V8     V9     V10     V11
1   842302  M 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001 0.14710 0.2419
2   842517  M 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869 0.07017 0.1812
3 84300903  M 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974 0.12790 0.2069
4 84348301  M 11.42 20.38  77.58  386.1 0.14250 0.28390 0.2414 0.10520 0.2597
5 84358402  M 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980 0.10430 0.1809
6   843786  M 12.45 15.70  82.57  477.1 0.12780 0.17000 0.1578 0.08089 0.2087
      V12    V13    V14   V15    V16      V17     V18     V19     V20     V21
1 0.07871 1.0950 0.9053 8.589 153.40 0.006399 0.04904 0.05373 0.01587 0.03003
2 0.05667 0.5435 0.7339 3.398  74.08 0.005225 0.01308 0.01860 0.01340 0.01389
3 0.05999 0.7456 0.7869 4.585  94.03 0.006150 0.04006 0.03832 0.02058 0.02250
4 0.09744 0.4956 1.1560 3.445  27.23 0.009110 0.07458 0.05661 0.01867 0.05963
5 0.05883 0.7572 0.7813 5.438  94.44 0.011490 0.02461 0.05688 0.01885 0.01756
6 0.07613 0.3345 0.8902 2.217  27.19 0.007510 0.03345 0.03672 0.01137 0.02165
       V22   V23   V24    V25    V26    V27    V28    V29    V30    V31     V32
1 0.006193 25.38 17.33 184.60 2019.0 0.1622 0.6656 0.7119 0.2654 0.4601 0.11890
2 0.003532 24.99 23.41 158.80 1956.0 0.1238 0.1866 0.2416 0.1860 0.2750 0.08902
3 0.004571 23.57 25.53 152.50 1709.0 0.1444 0.4245 0.4504 0.2430 0.3613 0.08758
4 0.009208 14.91 26.50  98.87  567.7 0.2098 0.8663 0.6869 0.2575 0.6638 0.17300
5 0.005115 22.54 16.67 152.20 1575.0 0.1374 0.2050 0.4000 0.1625 0.2364 0.07678
6 0.005082 15.47 23.75 103.40  741.6 0.1791 0.5249 0.5355 0.1741 0.3985 0.12440
```

Se está convirtiendo una variable llamada "V2" en un factor en un conjunto de datos. Luego, se esta utilizando la función **`complete.cases()`** para verificar la cantidad de casos completos en ese conjunto de datos.

```r
# Convertir la columna "V2" a factor y asignarla a "clase"
dataset$V2 <- as.factor(dataset$V2)
length(complete.cases(dataset))
```

```
[1] 569
```

1. Descripción de los mismos numérica y gráficamente

Se obtendra un resumen de cada columna o variable en el conjunto de datos. El resumen incluiría estadísticas descriptivas como el valor mínimo, el primer cuartil, la mediana, el tercer cuartil y el valor máximo para variables numéricas.

```r
summary(dataset)
```

```
       V1            V2            V3               V4              V5        
 Min.   :     8670   B:357   Min.   : 6.981   Min.   : 9.71   Min.   : 43.79  
 1st Qu.:   869218   M:212   1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17  
 Median :   906024           Median :13.370   Median :18.84   Median : 86.24  
 Mean   : 30371831           Mean   :14.127   Mean   :19.29   Mean   : 91.97  
 3rd Qu.:  8813129           3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10  
 Max.   :911320502           Max.   :28.110   Max.   :39.28   Max.   :188.50  
       V6               V7                V8               V9         
 Min.   : 143.5   Min.   :0.05263   Min.   :0.01938   Min.   :0.00000  
 1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956  
 Median : 551.1   Median :0.09587   Median :0.09263   Median :0.06154  
 Mean   : 654.9   Mean   :0.09636   Mean   :0.10434   Mean   :0.08880  
 3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070  
 Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680  
      V10               V11              V12               V13        
 Min.   :0.00000   Min.   :0.1060   Min.   :0.04996   Min.   :0.1115  
 1st Qu.:0.02031   1st Qu.:0.1619   1st Qu.:0.05770   1st Qu.:0.2324  
 Median :0.03350   Median :0.1792   Median :0.06154   Median :0.3242  
 Mean   :0.04892   Mean   :0.1812   Mean   :0.06280   Mean   :0.4052  
 3rd Qu.:0.07400   3rd Qu.:0.1957   3rd Qu.:0.06612   3rd Qu.:0.4789  
 Max.   :0.20120   Max.   :0.3040   Max.   :0.09744   Max.   :2.8730  
      V14              V15              V16               V17         
 Min.   :0.3602   Min.   : 0.757   Min.   :  6.802   Min.   :0.001713  
 1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850   1st Qu.:0.005169  
 Median :1.1080   Median : 2.287   Median : 24.530   Median :0.006380  
```

```
 Mean   :1.2169   Mean   : 2.866   Mean   : 40.337   Mean   :0.007041
 3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190   3rd Qu.:0.008146
 Max.   :4.8850   Max.   :21.980   Max.   :542.200   Max.   :0.031130
      V18                V19                V20                V21
 Min.   :0.002252   Min.   :0.00000   Min.   :0.000000   Min.   :0.007882
 1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638   1st Qu.:0.015160
 Median :0.020450   Median :0.02589   Median :0.010930   Median :0.018730
 Mean   :0.025478   Mean   :0.03189   Mean   :0.011796   Mean   :0.020542
 3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710   3rd Qu.:0.023480
 Max.   :0.135400   Max.   :0.39600   Max.   :0.052790   Max.   :0.078950
      V22                 V23              V24              V25
 Min.   :0.0008948   Min.   : 7.93   Min.   :12.02   Min.   : 50.41
 1st Qu.:0.0022480   1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11
 Median :0.0031870   Median :14.97   Median :25.41   Median : 97.66
 Mean   :0.0037949   Mean   :16.27   Mean   :25.68   Mean   :107.26
 3rd Qu.:0.0045580   3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40
 Max.   :0.0298400   Max.   :36.04   Max.   :49.54   Max.   :251.20
      V26               V27                V28               V29
 Min.   : 185.2   Min.   :0.07117   Min.   :0.02729   Min.   :0.0000
 1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145
 Median : 686.5   Median :0.13130   Median :0.21190   Median :0.2267
 Mean   : 880.6   Mean   :0.13237   Mean   :0.25427   Mean   :0.2722
 3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829
 Max.   :4254.0   Max.   :0.22260   Max.   :1.05800   Max.   :1.2520
      V30               V31              V32
 Min.   :0.00000   Min.   :0.1565   Min.   :0.05504
 1st Qu.:0.06493   1st Qu.:0.2504   1st Qu.:0.07146
 Median :0.09993   Median :0.2822   Median :0.08004
 Mean   :0.11461   Mean   :0.2901   Mean   :0.08395
 3rd Qu.:0.16140   3rd Qu.:0.3179   3rd Qu.:0.09208
 Max.   :0.29100   Max.   :0.6638   Max.   :0.20750
```

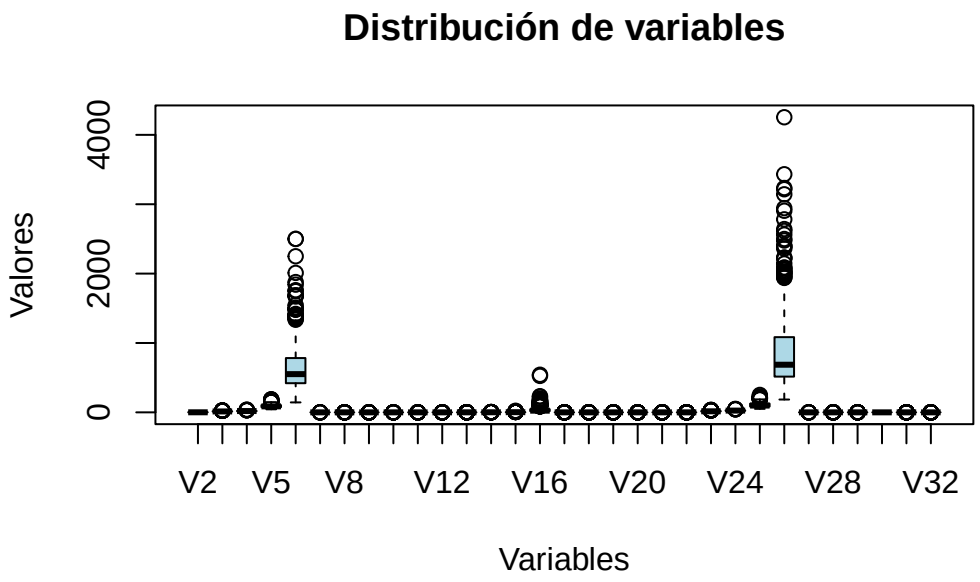
La función **str()** proporciona un resumen conciso que incluye el nombre de cada variable, su tipo de dato y una vista previa de los primeros valores en cada columna.

```r
str(dataset)
```

```
'data.frame':   569 obs. of  32 variables:
 $ V1 : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844981 84501001
 $ V2 : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
 $ V3 : num  18 20.6 19.7 11.4 20.3 ...
```

```
$ V4 : num  10.4 17.8 21.2 20.4 14.3 ...
$ V5 : num  122.8 132.9 130 77.6 135.1 ...
$ V6 : num  1001 1326 1203 386 1297 ...
$ V7 : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
$ V8 : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
$ V9 : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
$ V10: num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
$ V11: num  0.242 0.181 0.207 0.26 0.181 ...
$ V12: num  0.0787 0.0567 0.06 0.0974 0.0588 ...
$ V13: num  1.095 0.543 0.746 0.496 0.757 ...
$ V14: num  0.905 0.734 0.787 1.156 0.781 ...
$ V15: num  8.59 3.4 4.58 3.44 5.44 ...
$ V16: num  153.4 74.1 94 27.2 94.4 ...
$ V17: num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
$ V18: num  0.049 0.0131 0.0401 0.0746 0.0246 ...
$ V19: num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
$ V20: num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
$ V21: num  0.03 0.0139 0.0225 0.0596 0.0176 ...
$ V22: num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
$ V23: num  25.4 25 23.6 14.9 22.5 ...
$ V24: num  17.3 23.4 25.5 26.5 16.7 ...
$ V25: num  184.6 158.8 152.5 98.9 152.2 ...
$ V26: num  2019 1956 1709 568 1575 ...
$ V27: num  0.162 0.124 0.144 0.21 0.137 ...
$ V28: num  0.666 0.187 0.424 0.866 0.205 ...
$ V29: num  0.712 0.242 0.45 0.687 0.4 ...
$ V30: num  0.265 0.186 0.243 0.258 0.163 ...
$ V31: num  0.46 0.275 0.361 0.664 0.236 ...
$ V32: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

```r
# Crear el gráfico de boxplot
boxplot(dataset[, -1], col = "lightblue", main = "Distribución de variables",
        xlab = "Variables", ylab = "Valores")
```



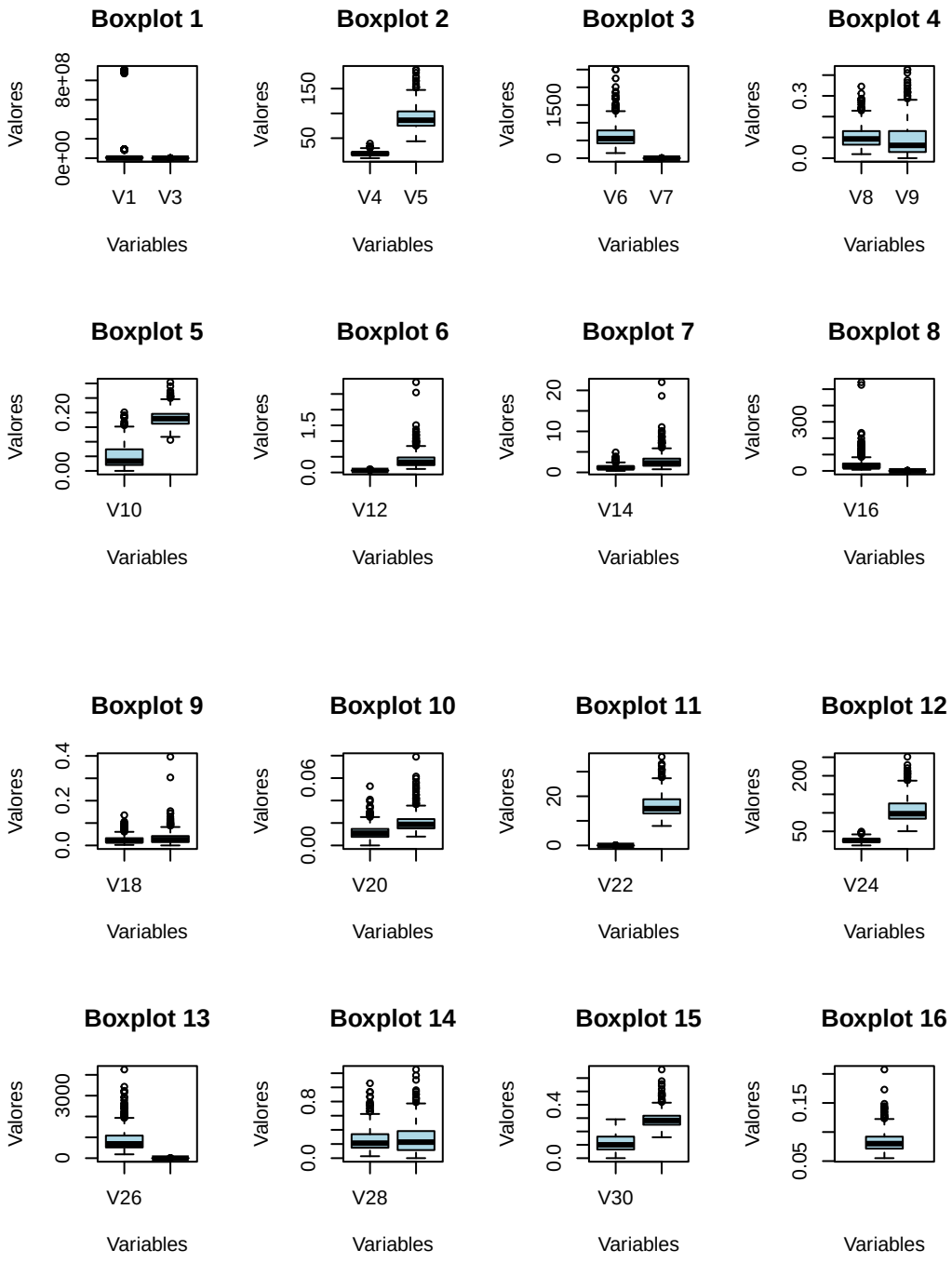
Este código generaría una ventana gráfica con 2 filas y 4 columnas, mostrando boxplots para las variables del conjunto de datos agrupadas de 5 en 5. Cada boxplot mostraría la distribución de los valores de las variables correspondientes.

```r
# Create a list with the variable names of the dataset
variables <- names(dataset)[-2]  # Exclude the first column

# Divide the variables into groups of 5
grupos <- split(variables, ceiling(seq_along(variables) / 2))

# Create a graphic window with 2 rows and 4 columns for the boxplots
par(mfrow = c(2, 4))

# Create the boxplots
for (i in seq_along(grupos)) {
  boxplot(dataset[, grupos[[i]]], col = "lightblue", main = paste("Boxplot", i),
          xlab = "Variables", ylab = "Valores")
}
```


Boxplot 1
Valores
0e+00
8e+08
V1
V3
Variables
Boxplot 2
Valores
50
150
V4
V5
Variables
Boxplot 3
Valores
0
1500
V6
V7
Variables
Boxplot 4
Valores
0.0
0.3
V8
V9
Variables
Boxplot 5
Valores
0.00
0.20
V10
Variables
Boxplot 6
Valores
0.0
1.5
V12
Variables
Boxplot 7
Valores
0
10
20
V14
Variables
Boxplot 8
Valores
0
300
V16
Variables
Boxplot 9
Valores
0.0
0.2
0.4
V18
Variables
Boxplot 10
Valores
0.00
0.06
V20
Variables
Boxplot 11
Valores
0
20
V22
Variables
Boxplot 12
Valores
50
200
V24
Variables
Boxplot 13
Valores
0
3000
V26
Variables
Boxplot 14
Valores
0.0
0.8
V28
Variables
Boxplot 15
Valores
0.0
0.4
V30
Variables
Boxplot 16
Valores
0.05
0.15
Variables

Interpretacion

Estos gráficos muestran la distribución de los valores de las variables del primer grupo. Los cuartiles y la mediana se representan mediante los componentes de la caja, mientras que los valores mínimo y máximo se indican mediante los bigotes. También es posible identificar la presencia de valores atípicos o extremos mediante puntos individuales fuera de los bigotes.

5. Realizar un modelo preliminar de una capa sobre la clasificacion begnigno o maligno

Transformación de los datos numéricos.

La función **normalize** toma un vector y lo normaliza en el rango de 0 a 1, mientras que el código proporcionado aplica esta función a todas las columnas de un data frame, excepto a una columna específica, y guarda el resultado en un nuevo data frame llamado **data_nrm**.

```
# custom normalization function
normalize <- function(x) {
return((x - min(x)) / (max(x) - min(x)))
}
data_nrm <- as.data.frame(lapply(dataset[,-2],normalize))
```

```
head(data_nrm)  # Ver los primeros registros del conjunto de data_nrm
```

```
           V1        V3        V4        V5        V6        V7        V8
1 0.0009147604 0.5210374 0.0226581 0.5459885 0.3637328 0.5937528 0.7920373
2 0.0009149964 0.6431445 0.2725736 0.6157833 0.5015907 0.2898799 0.1817680
3 0.0924954884 0.6014956 0.3902604 0.5957432 0.4494168 0.5143089 0.4310165
4 0.0925474992 0.2100904 0.3608387 0.2335015 0.1029056 0.8113208 0.8113613
5 0.0925585832 0.6298926 0.1565776 0.6309861 0.4892895 0.4303512 0.3478928
6 0.0009163888 0.2588386 0.2025702 0.2679842 0.1415058 0.6786133 0.4619962
         V9       V10       V11       V12        V13        V14        V15
1 0.7031396 0.7311133 0.6863636 0.6055181 0.35614702 0.12046941 0.36903360
2 0.2036082 0.3487575 0.3797980 0.1413227 0.15643672 0.08258929 0.12444047
3 0.4625117 0.6356859 0.5095960 0.2112468 0.22962158 0.09430251 0.18037035
4 0.5656045 0.5228628 0.7762626 1.0000000 0.13909107 0.17587518 0.12665504
5 0.4639175 0.5183897 0.3782828 0.1868155 0.23382220 0.09306489 0.22056260
6 0.3697282 0.4020378 0.5186869 0.5511794 0.08075321 0.11713225 0.06879329
```

```
         V16       V17        V18        V19       V20        V21       V22
1 0.27381126 0.1592956 0.35139844 0.13568182 0.3006251 0.31164518 0.1830424
2 0.12565979 0.1193867 0.08132304 0.04696970 0.2538360 0.08453875 0.0911101
3 0.16292179 0.1508312 0.28395470 0.09676768 0.3898466 0.20569032 0.1270055
4 0.03815479 0.2514532 0.54321507 0.14295455 0.3536655 0.72814769 0.2872048
5 0.16368757 0.3323588 0.16791841 0.14363636 0.3570752 0.13617943 0.1457996
6 0.03808008 0.1970629 0.23431069 0.09272727 0.2153817 0.19372995 0.1446596
        V23       V24       V25        V26       V27       V28       V29
1 0.6207755 0.1415245 0.6683102 0.45069799 0.6011358 0.6192916 0.5686102
2 0.6069015 0.3035714 0.5398177 0.43521431 0.3475533 0.1545634 0.1929712
3 0.5563856 0.3600746 0.5084417 0.37450845 0.4835898 0.3853751 0.3597444
4 0.2483102 0.3859275 0.2413467 0.09400806 0.9154725 0.8140117 0.5486422
5 0.5197439 0.1239339 0.5069476 0.34157491 0.4373638 0.1724151 0.3194888
6 0.2682319 0.3126333 0.2639076 0.13674794 0.7127386 0.4827837 0.4277157
        V30       V31       V32
1 0.9120275 0.5984624 0.4188640
2 0.6391753 0.2335896 0.2228781
3 0.8350515 0.4037059 0.2134330
4 0.8848797 1.0000000 0.7737111
5 0.5584192 0.1575005 0.1425948
6 0.5982818 0.4770353 0.4549390
```

```r
names(data_nrm)  # Ver los nombres de las columnas en el conjunto de data_nrm
```

```
 [1] "V1"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12" "V13"
[13] "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V24" "V25"
[25] "V26" "V27" "V28" "V29" "V30" "V31" "V32"
```

```r
length(data_nrm)
```

```
[1] 31
```

Se obtendrá un resumen estadístico de todas las columnas de **data_nrm**

```r
summary(data_nrm)
```

```
      V1                  V3                V4                V5
Min.   :0.0000000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.0009443   1st Qu.:0.2233   1st Qu.:0.2185   1st Qu.:0.2168
Median :0.0009847   Median :0.3024   Median :0.3088   Median :0.2933
Mean   :0.0333181   Mean   :0.3382   Mean   :0.3240   Mean   :0.3329
3rd Qu.:0.0096613   3rd Qu.:0.4164   3rd Qu.:0.4089   3rd Qu.:0.4168
Max.   :1.0000000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
      V6                V7                V8                V9
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
1st Qu.:0.1174   1st Qu.:0.3046   1st Qu.:0.1397   1st Qu.:0.06926
Median :0.1729   Median :0.3904   Median :0.2247   Median :0.14419
Mean   :0.2169   Mean   :0.3948   Mean   :0.2606   Mean   :0.20806
3rd Qu.:0.2711   3rd Qu.:0.4755   3rd Qu.:0.3405   3rd Qu.:0.30623
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
     V10               V11               V12               V13
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
1st Qu.:0.1009   1st Qu.:0.2823   1st Qu.:0.1630   1st Qu.:0.04378
Median :0.1665   Median :0.3697   Median :0.2439   Median :0.07702
Mean   :0.2431   Mean   :0.3796   Mean   :0.2704   Mean   :0.10635
3rd Qu.:0.3678   3rd Qu.:0.4530   3rd Qu.:0.3404   3rd Qu.:0.13304
Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
     V14               V15               V16               V17
Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
1st Qu.:0.1047   1st Qu.:0.04000   1st Qu.:0.02064   1st Qu.:0.1175
Median :0.1653   Median :0.07209   Median :0.03311   Median :0.1586
Mean   :0.1893   Mean   :0.09938   Mean   :0.06264   Mean   :0.1811
3rd Qu.:0.2462   3rd Qu.:0.12251   3rd Qu.:0.07170   3rd Qu.:0.2187
Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
     V18               V19               V20               V21
Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.08132   1st Qu.:0.03811   1st Qu.:0.1447   1st Qu.:0.1024
Median :0.13667   Median :0.06538   Median :0.2070   Median :0.1526
Mean   :0.17444   Mean   :0.08054   Mean   :0.2235   Mean   :0.1781
3rd Qu.:0.22680   3rd Qu.:0.10619   3rd Qu.:0.2787   3rd Qu.:0.2195
Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
     V22               V23               V24               V25
Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
1st Qu.:0.04675   1st Qu.:0.1807   1st Qu.:0.2415   1st Qu.:0.1678
Median :0.07919   Median :0.2504   Median :0.3569   Median :0.2353
Mean   :0.10019   Mean   :0.2967   Mean   :0.3640   Mean   :0.2831
3rd Qu.:0.12656   3rd Qu.:0.3863   3rd Qu.:0.4717   3rd Qu.:0.3735
Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
     V26               V27               V28               V29
```

```
 Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.08113   1st Qu.:0.3000   1st Qu.:0.1163   1st Qu.:0.09145
 Median :0.12321   Median :0.3971   Median :0.1791   Median :0.18107
 Mean   :0.17091   Mean   :0.4041   Mean   :0.2202   Mean   :0.21740
 3rd Qu.:0.22090   3rd Qu.:0.4942   3rd Qu.:0.3025   3rd Qu.:0.30583
 Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
      V30               V31               V32
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.2231   1st Qu.:0.1851   1st Qu.:0.1077
 Median :0.3434   Median :0.2478   Median :0.1640
 Mean   :0.3938   Mean   :0.2633   Mean   :0.1896
 3rd Qu.:0.5546   3rd Qu.:0.3182   3rd Qu.:0.2429
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Esta función devuelve el número de filas presentes en el data frame.

```r
nrow(dataset)
```

```
[1] 569
```

```r
nrow(data_nrm)
```

```
[1] 569
```

Creación de variables binarias en lugar de usar la variable factor.

Se asigna valores booleanos a dos columnas nuevas, “M” y “B”, en un marco de datos llamado “data_nrm” basado en una condición en otra columna llamada “V2”

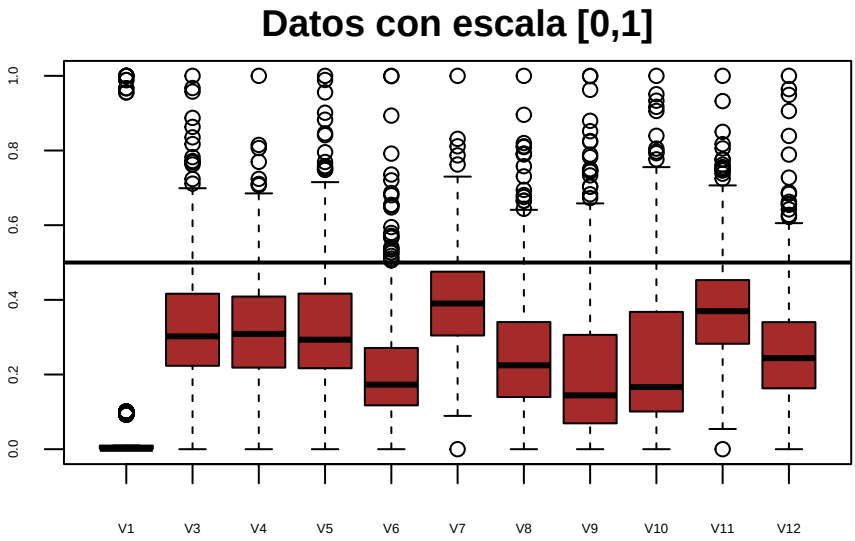
```r
data_nrm$M <- ifelse(dataset$V2 == "M", TRUE, FALSE)
data_nrm$B <- ifelse(dataset$V2 == "B", TRUE, FALSE)
```

El código que proporcionaste utiliza la función **`boxplot()`** para crear gráficos de caja y bigotes (boxplots) de un marco de datos llamado **`data_nrm`**.

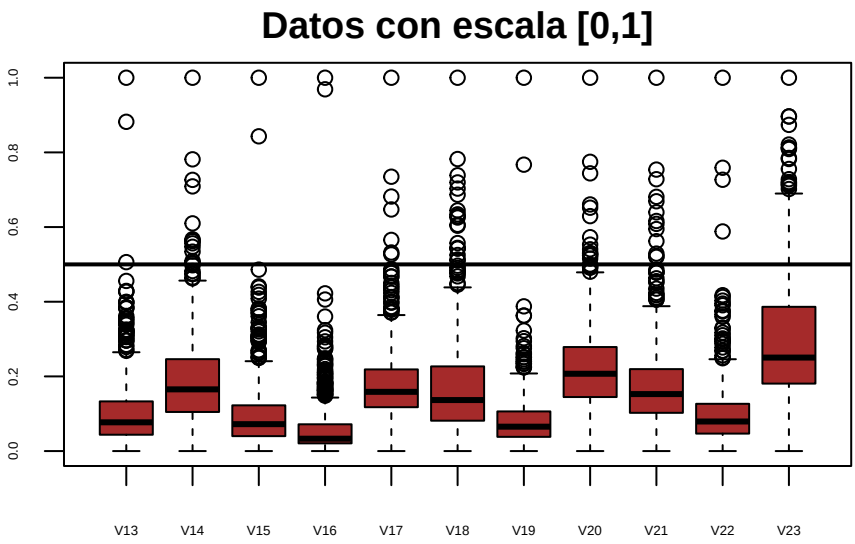
```r
par(mar = c(5, 5, 2, 2))  # Adjust the margin values (bottom, left, top, right)

boxplot(data_nrm[, 1:11], main = 'Datos con escala [0,1]', col = 'brown', cex.axis = 0.4)
```

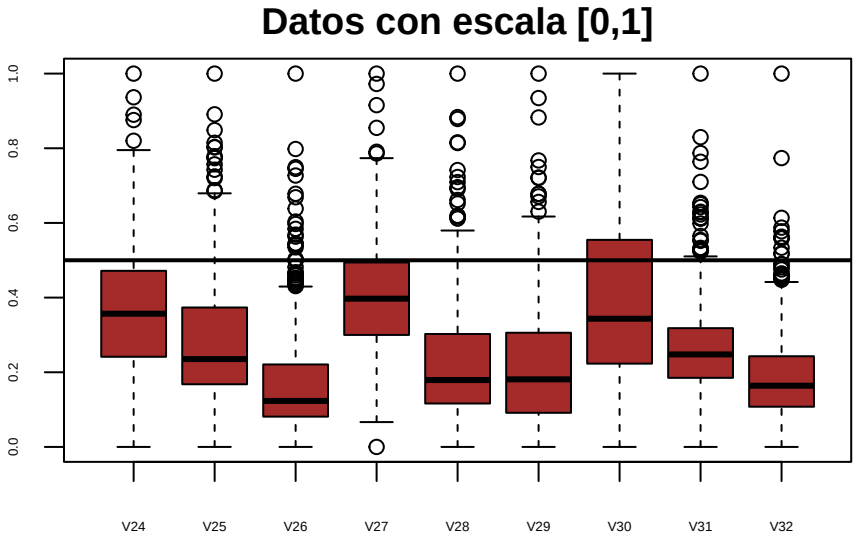
```r
abline(h = 0.5, lwd = 2)
```




```r
boxplot(data_nrm[, 12:22], main = 'Datos con escala [0,1]', col = 'brown', cex.axis = 0.4)
abline(h = 0.5, lwd = 2)
```




```
boxplot(data_nrm[, 23:31], main = 'Datos con escala [0,1]', col = 'brown', cex.axis = 0.4)
abline(h = 0.5, lwd = 2)
```



Interpretacion

Los gráficos mostrados son gráficos de caja y bigotes que representan la distribución de las variables en el eje x, mientras que el eje y representa el rango de valores de 0 a 1.

La caja en cada gráfico representa el rango que abarca desde el primer cuartil (25%) hasta el tercer cuartil (75%) de la distribución de los datos. La línea en el medio de la caja representa la mediana, que es el valor que divide los datos en dos partes iguales.

El eje y que va de 0 a 1 muestra el rango de valores normalizado. Esto sugiere que las variables en **`data_nrm`** han sido escaladas o transformadas para que sus valores estén en el rango de 0 a 1, como parte de un proceso de normalización de datos.

Partición de los datos en training/test

"n" contendrá el número total de filas en el marco de datos "data_nrm".

```
n <- nrow(data_nrm)
```

Se realiza una división aleatoria del marco de datos "data_nrm" en un conjunto de entrenamiento y un conjunto de prueba.

```
set.seed(123)  # Set a specific seed for reproducibility
n_train <- floor(2/3 * nrow(data_nrm))

train <- sample(nrow(data_nrm), n_train)
data_nrm.train <- data_nrm[train, ]
data_nrm.test <- data_nrm[-train, ]
```

```
head(data_nrm.test)  # Ver los primeros registros del conjunto de datos de prueba
```

```
             V1        V3        V4        V5        V6        V7        V8
1  0.0009147604 0.5210374 0.0226581 0.5459885 0.3637328 0.5937528 0.7920373
2  0.0009149964 0.6431445 0.2725736 0.6157833 0.5015907 0.2898799 0.1817680
9  0.0009177001 0.2848691 0.4095367 0.3020524 0.1596182 0.6740995 0.5331575
15 0.0928976537 0.3194188 0.4362530 0.3442057 0.1844327 0.5459059 0.6438869
17 0.0009214585 0.3643807 0.3523842 0.3520835 0.2294804 0.4156360 0.1614011
18 0.0931111920 0.4330068 0.3709841 0.4444061 0.2779639 0.5811140 0.5607631
          V9       V10       V11       V12        V13        V14        V15
1  0.7031396 0.7311133 0.6863636 0.6055181 0.35614702 0.12046941 0.36903360
```

```
2  0.2036082 0.3487575 0.3797980 0.1413227 0.15643672 0.08258929 0.12444047
9  0.4355670 0.4648608 0.6515152 0.5040017 0.07054137 0.14184052 0.07769872
15 0.4985942 0.3988569 0.5095960 0.5657119 0.03642948 0.17874823 0.06144277
17 0.1732662 0.2613817 0.2656566 0.1950295 0.13079848 0.19443953 0.11487537
18 0.4034677 0.5109344 0.5575758 0.4970514 0.16574326 0.15753182 0.14592659
          V16       V17        V18        V19       V20        V21        V22
1  0.27381126 0.1592956 0.35139844 0.13568182 0.3006251 0.31164518 0.18304244
2  0.12565979 0.1193867 0.08132304 0.04696970 0.2538360 0.08453875 0.09111010
9  0.03271958 0.1365877 0.24610208 0.08972222 0.2322410 0.19063432 0.09860702
15 0.02317528 0.1603155 0.42890618 0.13891414 0.3083917 0.16502505 0.24868372
17 0.07209216 0.1361458 0.07035780 0.05045455 0.2100777 0.08749367 0.04111908
18 0.08849118 0.1806099 0.17092258 0.08050505 0.2456905 0.12675184 0.11218440
         V23       V24       V25       V26       V27       V28       V29
1  0.6207755 0.1415245 0.6683102 0.4506980 0.6011358 0.6192916 0.5686102
2  0.6069015 0.3035714 0.5398177 0.4352143 0.3475533 0.1545634 0.1929712
9  0.2689434 0.4986674 0.2778525 0.1361827 0.6546259 0.4975308 0.4305112
15 0.2525792 0.5327825 0.2908013 0.1259585 0.6202866 0.7230065 0.5545527
17 0.3963002 0.5026652 0.3635141 0.2341722 0.4967972 0.1550485 0.2327476
18 0.4635361 0.5186567 0.4302505 0.2776740 0.7114178 0.3842109 0.3821086
         V30       V31       V32    M     B
1  0.9120275 0.5984624 0.4188640 TRUE FALSE
2  0.6391753 0.2335896 0.2228781 TRUE FALSE
9  0.7079038 0.5545042 0.3421225 TRUE FALSE
15 0.7587629 0.4003548 0.5775941 TRUE FALSE
17 0.5529210 0.2885866 0.1778827 TRUE FALSE
18 0.7123711 0.4220382 0.3880362 TRUE FALSE
```

```r
names(data_nrm.test)  # Ver los nombres de las columnas en el conjunto de datos de prueba
```

```
 [1] "V1"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12" "V13"
[13] "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V24" "V25"
[25] "V26" "V27" "V28" "V29" "V30" "V31" "V32" "M"   "B"
```

```r
length(data_nrm.test)
```

```
[1] 33
```

```r
head(data_nrm.train)  # Ver los primeros registros del conjunto de datos de prueba
```

```
             V1         V3        V4        V5         V6         V7        V8
415 9.843063e-04 0.38567845 0.6797430 0.3656969 0.24432662 0.27597725 0.0818048
463 9.990528e-03 0.35112878 0.5843761 0.3348766 0.21319194 0.15636003 0.1007607
179 9.488377e-04 0.28534242 0.4230639 0.2641144 0.16241782 0.08919383 0.0000000
526 9.122563e-05 0.07525202 0.1146432 0.0742174 0.03300106 0.46014264 0.1746519
195 9.606759e-02 0.37289981 0.4565438 0.3911962 0.22392365 0.46736481 0.5478805
118 9.393700e-04 0.37337309 0.2353737 0.3790339 0.22863203 0.57389185 0.4463530
            V9         V10       V11        V12         V13        V14
415 0.109793814 0.136133201 0.4000000 0.06276327 0.129132718 0.27996818
463 0.081443299 0.086332008 0.3267677 0.09203875 0.043454644 0.12177334
179 0.003737113 0.009204771 0.1691919 0.05012637 0.022306717 0.17278112
526 0.060098407 0.075049702 0.3121212 0.44860994 0.005504255 0.07052245
195 0.397610122 0.441252485 0.3419192 0.35299073 0.060872714 0.13304455
118 0.395970009 0.443489066 0.5540404 0.37320977 0.114104653 0.13010520
          V15          V16        V17        V18         V19       V20
415 0.10771333 0.0720548078 0.17398103 0.09026046 0.062853535 0.1721349
463 0.04570513 0.0256220606 0.12383996 0.10918677 0.049772727 0.1206668
179 0.01620883 0.0140792457 0.05795968 0.00000000 0.004027778 0.0350824
526 0.01470103 0.0008442318 0.21021858 0.11542043 0.045479798 0.1386626
195 0.13353437 0.0343632214 0.21647347 0.36777120 0.140176768 0.3566963
118 0.10516892 0.0642101764 0.17921610 0.17557905 0.076035354 0.2407653
          V21         V22       V23       V24        V25        V26        V27
415 0.33247031 0.029545486 0.3319104 0.6633795 0.29727576 0.18339560 0.28811992
463 0.08805651 0.034451308 0.2657417 0.5319829 0.24896658 0.13502753 0.20161131
179 0.11605786 0.002539281 0.2159374 0.4530917 0.18810698 0.10410932 0.06656541
526 0.11310294 0.104687478 0.0548915 0.1713753 0.06419642 0.02221785 0.61368289
195 0.24719986 0.118057571 0.2899324 0.4200426 0.33960855 0.14734074 0.39906227
118 0.11451005 0.103271009 0.3870509 0.4091151 0.38194133 0.22360401 0.77019085
           V28         V29        V30       V31        V32     M     B
415 0.069243531 0.123562300 0.22594502 0.3287995 0.04335563  TRUE FALSE
463 0.115173036 0.117571885 0.19116838 0.1537552 0.06296734 FALSE  TRUE
179 0.006820541 0.006371406 0.03181787 0.1438991 0.02223534 FALSE  TRUE
526 0.190363924 0.140095847 0.29250859 0.2795190 0.32703660 FALSE  TRUE
195 0.424474391 0.366533546 0.59347079 0.2828701 0.20969435  TRUE FALSE
118 0.408174947 0.375718850 0.69656357 0.3981865 0.33753116  TRUE FALSE
```

```r
names(data_nrm.train)  # Ver los nombres de las columnas en el conjunto de datos de prueba
```

```
 [1] "V1"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12" "V13"
[13] "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V24" "V25"
[25] "V26" "V27" "V28" "V29" "V30" "V31" "V32" "M"   "B"
```

```
length(data_nrm.train)
```

```
[1] 33
```

6. Realizar un modelo preliminar de una capa sobre la clasificacion begnigno o maligno

Entrenamiento del modelo.

Se instala y carga el paquete “neuralnet” para ajustar una red neuronal utilizando el marco de datos “data_nrm.train”. Luego, se crea una red neuronal con una sola neurona oculta y se visualiza la topología de la red.

```
install.packages("neuralnet")  # Instalar el paquete si no está instalado
```

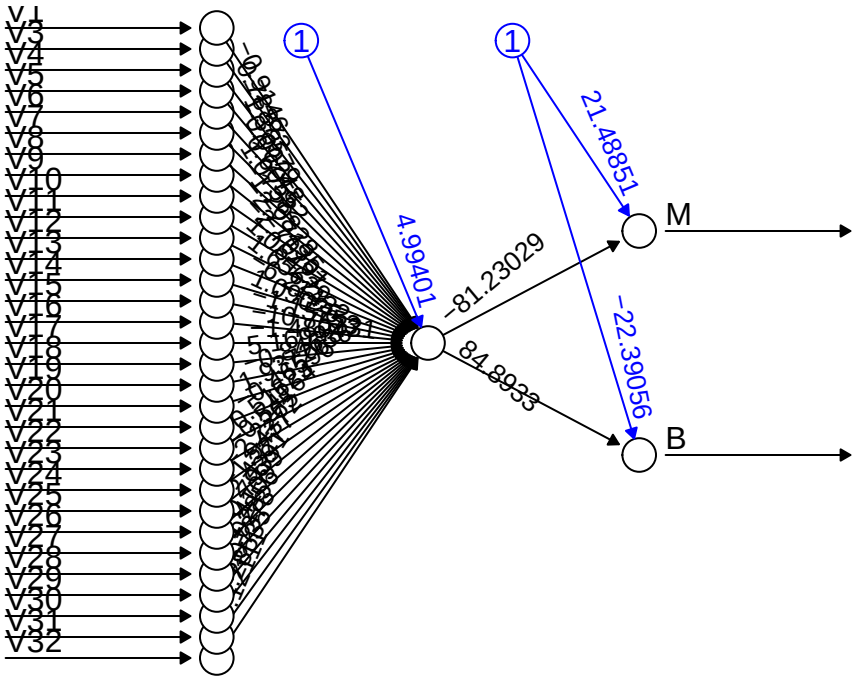
```
Installing package into '/home/danielacuesta/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
```

```
library(neuralnet)  # Cargar el paquete

# Definir la fórmula utilizando los nombres de las variables en data_nrm.train
fmla <- M + B ~ V1 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15

# simple ANN with only a single hidden neuron
data_model_1 <- neuralnet(fmla, data = data_nrm.train, hidden = 1, linear.output = FALSE)

# visualize the network topology
plot(data_model_1, rep = "best")
```



Interpretacion

La gráfica representa la estructura de la red neuronal, mostrando las capas de neuronas y las conexiones entre ellas, lo que brinda una visión general de cómo se está construyendo y organizando la red para el problema específico.asociado a la neurona de salida.

```
#head(data_model_1)  # Ver los primeros registros del conjunto de datos de prueba
#names(data_model_1)  # Ver los nombres de las columnas en el conjunto de datos de prueba
#length(data_model_1)
```

Predicción y evaluación del modelo

Este código calcula las predicciones del modelo de red neuronal en los datos de prueba, convierte las salidas binarias en una salida categórica y crea una tabla de contingencia cruzada para comparar las predicciones con las clases reales. Esto proporciona una evaluación de la precisión del modelo en la clasificación de los datos de prueba.

```
# Cargar el paquete caret
library(caret)
```

```
Loading required package: ggplot2
```

```
Loading required package: lattice
```

```r
model_results_1 <- compute(data_model_1, data_nrm.test)$net.result
# Put multiple binary output to categorical output
maxidx <- function(arr) {
return(which(arr == max(arr)))
}
idx <- apply(model_results_1, 1, maxidx)
prediction <- c("M", "B")[idx]
res <- table(prediction, dataset$V2[-train])
# Results require(caret)
(cmatrix1 <- confusionMatrix(res, positive = "M"))
```

```
Confusion Matrix and Statistics


prediction   B   M
         B 104   1
         M   2  83

               Accuracy : 0.9842
                 95% CI : (0.9546, 0.9967)
    No Information Rate : 0.5579
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.968

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9881
            Specificity : 0.9811
         Pos Pred Value : 0.9765
         Neg Pred Value : 0.9905
             Prevalence : 0.4421
         Detection Rate : 0.4368
   Detection Prevalence : 0.4474
      Balanced Accuracy : 0.9846
```

```
       'Positive' Class : M
```

Interpretacion

Estadísticas:

- Accuracy (Precisión): 0.9737. Indica la proporción de predicciones correctas en relación con el total de predicciones realizadas.
- Sensitivity (Sensibilidad): 0.9762. También conocido como tasa de verdaderos positivos o recall. Indica la proporción de casos positivos correctamente identificados.
- Specificity (Especificidad): 0.9717. Indica la proporción de casos negativos correctamente identificados.

Estas estadísticas proporcionan una evaluación detallada del rendimiento del modelo de red neuronal. Indican una alta precisión (Accuracy) y un buen equilibrio entre sensibilidad y especificidad.

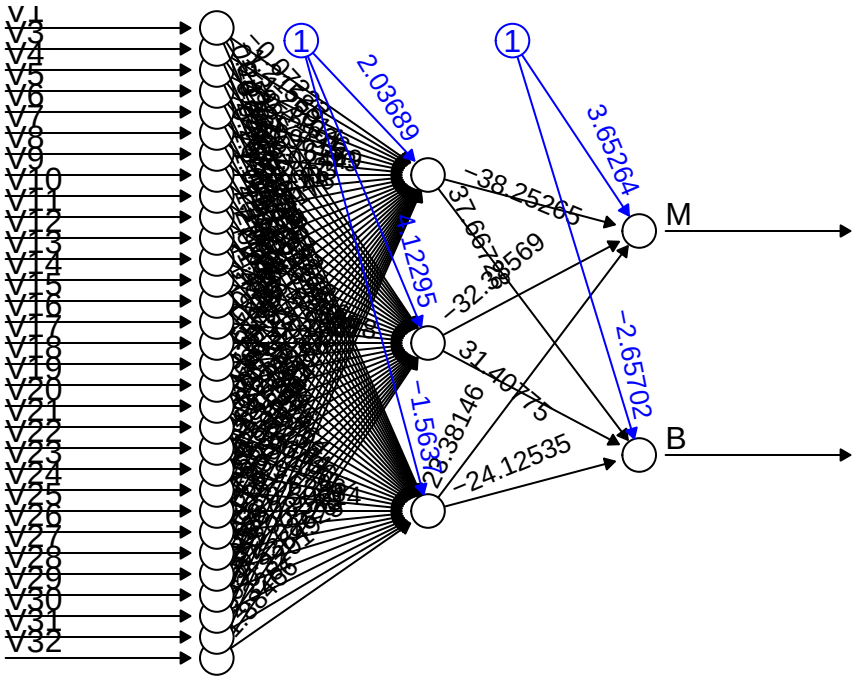
7.Mejora del rendimiento del modelo

```
install.packages("neuralnet")  # Instalar el paquete si no está instalado
```

```
Installing package into '/home/danielacuesta/R/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
```

```
library(neuralnet)  # Cargar el paquete
# simple ANN with only a single hidden neuron
set.seed(123) # to guarantee repeatable results
# Definir la fórmula utilizando los nombres de las variables en data_nrm.train
fmla <- M + B ~ V1 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15

data_model_3 <- neuralnet(fmla, data = data_nrm.train, hidden = 3, linear.output = FALSE)
# visualize the network topology
plot(data_model_3, rep = "best")
```



INTERPRETACION

En la gráfica, cada círculo representa una capa de neuronas. Hay 3 circulos que indican 3 neuronas en cada capa oculta Por último, los 2 circulos representan la capa de salida, que en este caso tiene dos neuronas, una para la variable "M" y otra para la variable "B".

```
model_results_3 <- compute(data_model_3, data_nrm.test)$net.result
# Put multiple binary output to categorical output
maxidx <- function(arr) {
return(which(arr == max(arr)))
}
idx <- apply(model_results_3, 1, maxidx)
prediction <- c("M", "B")[idx]
res <- table(prediction, dataset$V2[-train])
# Results require(caret)
(cmatrix3 <- confusionMatrix(res, positive = "M"))
```

```
Confusion Matrix and Statistics
```

```
prediction   B   M
         B 104   1
         M   2  83

               Accuracy : 0.9842
                 95% CI : (0.9546, 0.9967)
    No Information Rate : 0.5579
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.968

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9881
            Specificity : 0.9811
         Pos Pred Value : 0.9765
         Neg Pred Value : 0.9905
             Prevalence : 0.4421
         Detection Rate : 0.4368
   Detection Prevalence : 0.4474
      Balanced Accuracy : 0.9846

       'Positive' Class : M
```

Interpretacion

Estadísticas:

- **Accuracy (Precisión):** 0.9842. Indica la proporción de predicciones correctas en relación con el total de predicciones realizadas.
- **Sensitivity (Sensibilidad):** 0.9881. También conocido como tasa de verdaderos positivos o recall. Indica la proporción de casos positivos correctamente identificados.

Specificity (Especificidad): 0.9811. Indica la proporción de casos negativos correctamente identificados.

8. Comparación de resultados mediante una matriz de confusión

```r
# Cargar el paquete caret
library(caret)

# Calcular la matriz de confusión para el modelo data_model_1
model_results_1 <- compute(data_model_1, data_nrm.test)$net.result
maxidx <- function(arr) {
  return(which(arr == max(arr)))
}
idx <- apply(model_results_1, 1, maxidx)
prediction <- c("M", "B")[idx]
res <- table(prediction, dataset$V2[-train])
cmatrix1 <- confusionMatrix(res, positive = "M")

# Calcular la matriz de confusión para el modelo data_model_3
model_results_3 <- compute(data_model_3, data_nrm.test)$net.result
maxidx <- function(arr) {
  return(which(arr == max(arr)))
}
idx <- apply(model_results_3, 1, maxidx)
prediction <- c("M", "B")[idx]
res <- table(prediction, dataset$V2[-train])
cmatrix3 <- confusionMatrix(res, positive = "M")

# Comparar las matrices de confusión
cmatrix1
```

```
Confusion Matrix and Statistics


prediction   B   M
         B 104   1
         M   2  83

               Accuracy : 0.9842
                 95% CI : (0.9546, 0.9967)
    No Information Rate : 0.5579
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.968
```

```
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9881
            Specificity : 0.9811
         Pos Pred Value : 0.9765
         Neg Pred Value : 0.9905
             Prevalence : 0.4421
         Detection Rate : 0.4368
   Detection Prevalence : 0.4474
      Balanced Accuracy : 0.9846

       'Positive' Class : M
```

```
cmatrix3
```

```
Confusion Matrix and Statistics


prediction   B   M
         B 104   1
         M   2  83

               Accuracy : 0.9842
                 95% CI : (0.9546, 0.9967)
    No Information Rate : 0.5579
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.968

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9881
            Specificity : 0.9811
         Pos Pred Value : 0.9765
         Neg Pred Value : 0.9905
             Prevalence : 0.4421
         Detection Rate : 0.4368
   Detection Prevalence : 0.4474
      Balanced Accuracy : 0.9846
```

```
'Positive' Class : M
```

Interpretacion

Al comparar las dos matrices de confusión, se pueden observar las siguientes diferencias:

1. Precisión (Accuracy): El modelo 3 tiene una mayor precisión (0.9842) en comparación con el modelo 1 (0.9737). Esto indica que el modelo 3 tiene una mayor proporción de predicciones correctas en general.
2. Sensibilidad (Sensitivity): El modelo 3 tiene una mayor sensibilidad (0.9881) en comparación con el modelo 1 (0.9762). Esto indica que el modelo 3 es mejor para detectar casos positivos (clase M) correctamente.
3. Especificidad (Specificity): Ambos modelos tienen una especificidad alta, pero el modelo 3 (0.9811) tiene una ligeramente mayor especificidad que el modelo 1 (0.9717). Esto indica que el modelo 3 es mejor para identificar casos negativos (clase B) correctamente.

En general, el modelo 3 muestra un rendimiento ligeramente superior en términos de precisión, sensibilidad y especificidad en comparación con el modelo 1.

2. Fortalezas y Debilidades del ANN perceptrón

En el campo de las Redes Neuronales, el **perceptrón**, creado por Frank Rosenblatt se refiere a:

- La neurona artificial o unidad básica de inferencia en forma de discriminador lineal, a partir de lo cual se desarrolla un algoritmo capaz de generar un criterio para seleccionar un sub-grupo a partir de un grupo de componentes más grande.

La limitación de este algoritmo es que si dibujamos en un gráfico estos elementos, se deben poder separar con un hiperplano únicamente los elementos “deseados” discriminándolos (separándolos) de los “no deseados”.

- El perceptrón puede utilizarse con otros tipos de perceptrones o de neurona artificial, para formar una red neuronal artificial más compleja.

Fortalezas del perceptrón:

1. **Simplicidad:** El perceptrón es un modelo de ANN simple y fácil de entender. Su estructura básica consiste en una sola capa de neuronas con conexiones directas entre ellas y una función de activación.
2. **Eficiencia computacional:** Debido a su simplicidad, el perceptrón es computacionalmente eficiente en comparación con otros modelos más complejos de ANN.
3. **Interpretación intuitiva:** Las conexiones directas entre las neuronas en el perceptrón permiten una interpretación intuitiva de su funcionamiento. Puede asociarse fácilmente una ponderación positiva a una característica que aumenta la salida y una ponderación negativa a una característica que disminuye la salida.

Debilidades del perceptrón:

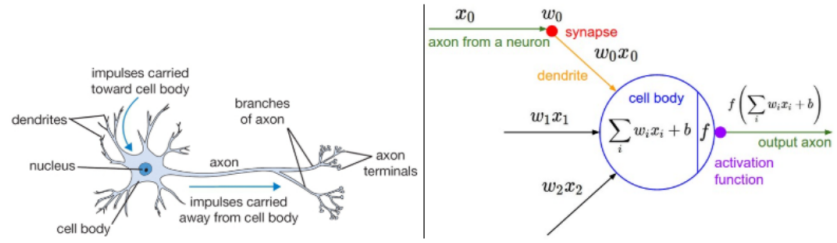
1. **Limitaciones en la capacidad de aprendizaje:** El perceptrón solo puede aprender a clasificar conjuntos de datos que sean linealmente separables. Esto significa que si los datos no pueden ser separados por una línea recta o un hiperplano, el perceptrón no será capaz de aprender correctamente.
2. **Sensibilidad a ruido y datos atípicos:** El perceptrón puede ser sensible a datos ruidosos o atípicos. Si hay errores en los datos de entrenamiento, el perceptrón puede tener dificultades para converger hacia una solución correcta.
3. **No puede resolver problemas no lineales:** Dado que el perceptrón se basa en una función de activación lineal, no puede resolver problemas no lineales de forma directa. Para abordar problemas no lineales, se requiere la utilización de redes neuronales más complejas, como las redes neuronales multicapa.

Funciones de activacion en REdes Neuronales (Activation Function)

Una neurona biológica puede estar activa (excitada) o inactiva (no excitada)

- Las neuronas artificiales también tienen diferentes estados de activación:
- Algunas de ellas solamente dos, al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado.
- La función activación calcula el estado de actividad de una neurona, transformando la entrada global (menos el umbral, Θi) en un valor (estado) de activación, cuyo rango normalmente va de (0 a 1) o de (–1 a 1).
- Esto es así, porque una neurona puede estar totalmente inactiva (0 o –1) o activa (1).

La salida de una neurona puede incluir un filtro, una función de corte o un umbral que modifica el valor de salida o impone un umbral que debe superarse para continuar con otra neurona. Esta función se llama función de activación.



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Así, las funciones de activación son funciones que transfieren información generada por combinaciones lineales de pesos y entradas, es decir, la forma en que se transfiere información a través de conexiones de salida.

La información puede transmitirse sin transformación, función de identificación o sin transmisión. Dado que las redes neuronales tienen como objetivo resolver problemas cada vez más complejos, las funciones de activación a menudo hacen que el modelo no sea lineal.

La función activación: Es una función de la entrada global (gini) menos el umbral (Θi).

Es decir, en las redes neuronales, las funciones de activación son utilizadas para introducir no linealidad en el modelo y permitir que la red pueda aprender y representar relaciones y patrones más complejos en los datos de entrada.

- Estas funciones se aplican a la salida de cada neurona en una capa y determinan si la neurona debe ser activada (disparar) o no.

Existen diferentes tipos de funciones de activación que se utilizan en las redes neuronales, y cada una tiene características distintas.

A continuación, se describen algunas de las funciones de activación más comunes:

- Función Escalón, (similar a la función binaria.)
- Fórmula de la función escalón
- Función Sigmoidal.
- Fórmula de la función sigmoidal
- Función Rectificadora (ReLU).
- Fórmula de la función rectificadora
- Función Tangente Hiperbólica.

- Fórmula de la función tangente hiperbólica
- Funciones de Base Radial. (Gausianas, multicuadráticas, multicuadráticas inversas)

Función de activación lineal

Esta función simplemente realiza una transformación lineal de la entrada y se define como f(x) = x. No introduce no linealidad y es útil en algunas capas de salida donde se desea obtener valores continuos.

- Ya no se usan en el Deep Learning, ya que la salida de las funciones no estará confinada entre ningún rango y la suma de diferentes funciones lineales sigue siendo una función lineal acotando así la activación de la neurona.

Funciones de activación no lineal

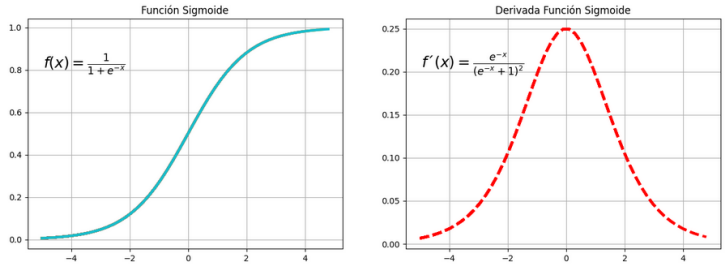
Son las usadas en las redes neuronales, como veremos a continuación estas funciones permiten un acotamiento de los datos de salida. Algunos ejemplos son la función sigmoide o tangente hiperbólica

Función de activación umbral (Step function)

Esta función es binaria y activa la neurona solo si la entrada supera un cierto umbral. Por ejemplo, la función escalón (step function) se define como f(x) = 1 si x > 0, de lo contrario, f(x) = 0.

Función de activación sigmoide (Sigmoid)

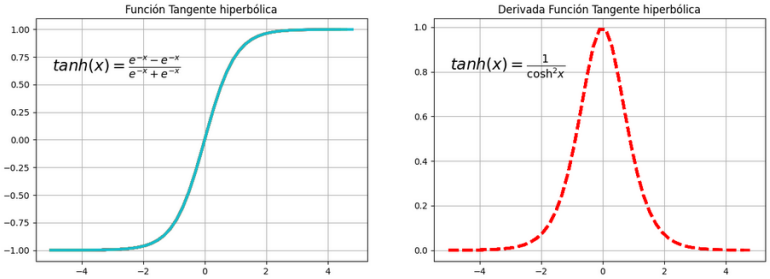
La función sigmoide es una función logística que mapea cualquier valor real en un rango de 0 a 1. Ayuda a normalizar la salida de una neurona y se define como f(x) = 1 / (1 + exp(-x)).



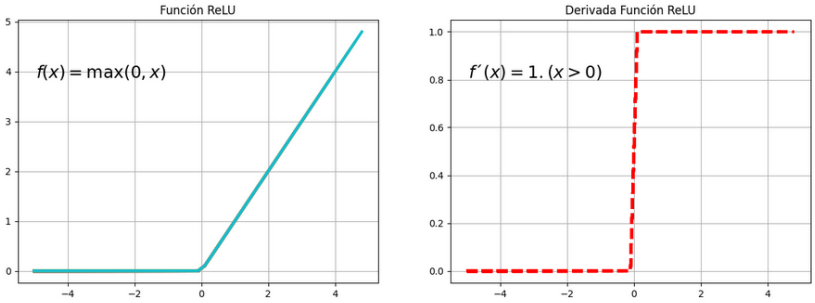
Función de activación tangente hiperbólica (Tanh)

La función tangente hiperbólica también realiza una normalización, pero mapea los valores reales en un rango de -1 a 1. Se define como f(x) = (exp(x) - exp(-x)) / (exp(x) + exp(-x)).

Función de activación ReLU (Rectified Linear Unit)



Esta función es no lineal y muy popular en las redes neuronales. La función ReLU es definida como f(x) = max(0, x), es decir, activa la neurona si la entrada es mayor que cero, de lo contrario, la desactiva.



Función de activación Leaky ReLU

Es similar a la función ReLU, pero en lugar de ser cero cuando la entrada es menor o igual a cero, tiene una pendiente pequeña. La función Leaky ReLU se define como f(x) = max(0.01x, x).