codigoTFG

Jorge de AndrÃfÆ'Ã†â€™Ãfâ€šÃ‚Â©s

12 de enero de 2019

Lo primero que tengo que hacer es importar el dataset que he creado

```
dataset <- read.csv("C:/Users/jorge/Desktop/Documentos Clase/Universidad/4ºCarrera/1er Cuatrimestre/Int
```

Ahora lo que hago es pasarlo a una matriz, quitando tanto el nombre (que no me interesa) como la etiqueta (que no la necesito por ahora)

```
matriz.pacientes.etiquetas <- dataset[, -1]
matriz.pacientes.datos <- matriz.pacientes.etiquetas[, -25]
```

Análisis Exploratorio

Primero compruebo que todos los datos tienen un tipo correcto.

```
sapply(matriz.pacientes.datos, class)
```

```
##              edad               sex rel_ctxo_rel_mala   rel_ctxo_trauma
##         "integer"         "integer"         "integer"         "integer"
##    rel_ctxo_buena           ed_perm           ed_norm           ed_estr
##         "integer"         "integer"         "integer"         "integer"
##          resil_ba          resil_me          resil_al           pen_dic
##         "integer"         "integer"         "integer"         "integer"
##            gen_ex              etiq           fil_men           max_min
##         "integer"         "integer"         "integer"         "integer"
##          conc_arb          pseu_res               deb           raz_emo
##         "integer"         "integer"         "integer"         "integer"
##             inhib             asert             agres             impuls
##         "integer"         "integer"         "integer"         "integer"
```

Veo la media de la edad de los pacientes y el rango en el que se mueve

```
mean(matriz.pacientes.datos[, 1])
```

```
## [1] 26.46269
```

```
range(matriz.pacientes.datos[, 1])
```

```
## [1] 13 52
```

Finalmente, veo un resúmen de cada columna

```
summary(matriz.pacientes.datos)
```

```
##       edad            sex        rel_ctxo_rel_mala rel_ctxo_trauma
##  Min.   :13.00   Min.   :0.000   Min.   :0.0000    Min.   :0.0000
##  1st Qu.:19.50   1st Qu.:0.000   1st Qu.:0.0000    1st Qu.:0.0000
##  Median :25.00   Median :0.000   Median :0.0000    Median :0.0000
##  Mean   :26.46   Mean   :0.209   Mean   :0.1343    Mean   :0.3582
##  3rd Qu.:30.50   3rd Qu.:0.000   3rd Qu.:0.0000    3rd Qu.:1.0000
##  Max.   :52.00   Max.   :1.000   Max.   :1.0000    Max.   :1.0000
##  rel_ctxo_buena      ed_perm          ed_norm          ed_estr
```

```
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.0000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.5075   Mean   :0.2836   Mean   :0.4925   Mean   :0.2239
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##     resil_ba         resil_me         resil_al          pen_dic
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:1.0000
##  Median :1.0000   Median :0.0000   Median :0.00000   Median :1.0000
##  Mean   :0.5672   Mean   :0.4179   Mean   :0.01493   Mean   :0.8955
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##      gen_ex            etiq           fil_men         max_min
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:1.0000   1st Qu.:0.5000   1st Qu.:1.000   1st Qu.:1.0000
##  Median :1.0000   Median :1.0000   Median :1.000   Median :1.0000
##  Mean   :0.9552   Mean   :0.7463   Mean   :0.791   Mean   :0.9701
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000
##     conc_arb         pseu_res           deb            raz_emo
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:1.000
##  Median :1.0000   Median :1.0000   Median :1.0000   Median :1.000
##  Mean   :0.9851   Mean   :0.5075   Mean   :0.9403   Mean   :0.791
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000
##      inhib            asert            agres           impuls
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :1.0000   Median :0.0000   Median :0.000   Median :1.0000
##  Mean   :0.6567   Mean   :0.1343   Mean   :0.209   Mean   :0.6119
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000   Max.   :1.0000
```

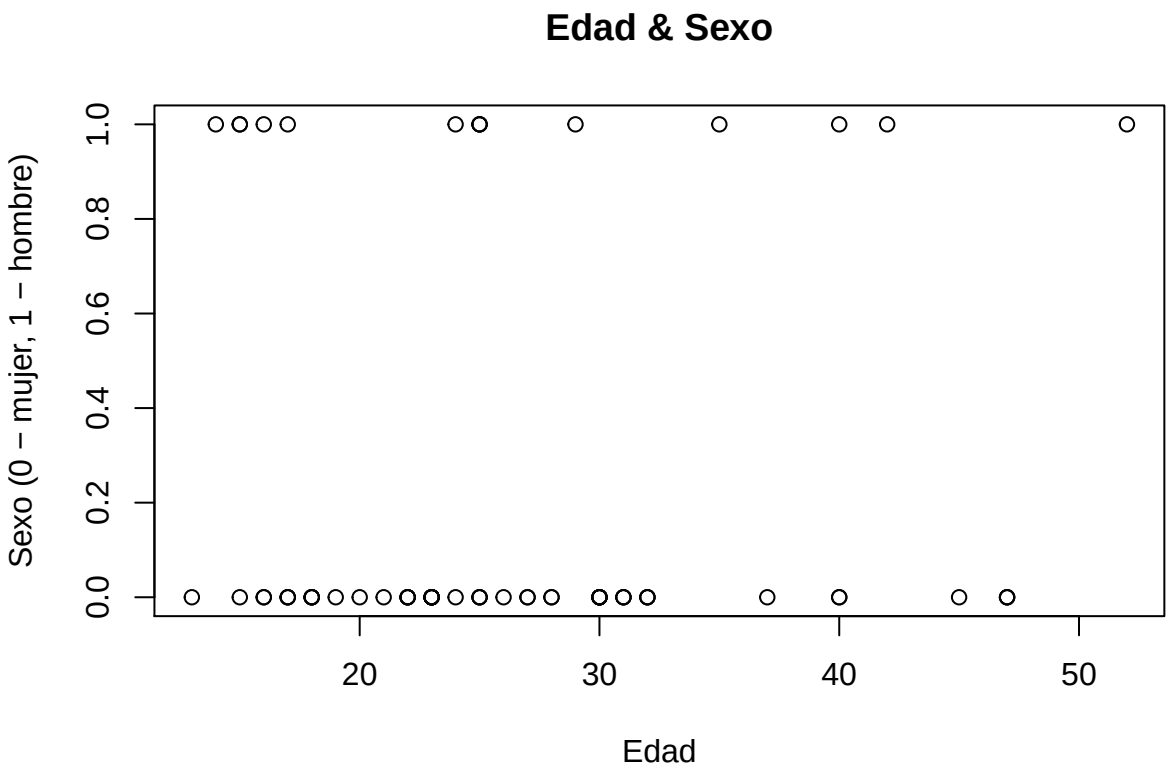
Como se puede ver, los datos de los pacientes están muy distanciados, y además su media es muy alta. Así, la media de la edad difiere enormemente del resto de valores de la matriz. Debido a ello, debemos de hacer un preprocesado de los datos del problema.

Antes que este preprocesado,voy a hacer la visualización de algunas relaciones entre variables, de tal manera que podamos ver gráficamente algunos aspectos interesantes:

Visualización de Datos

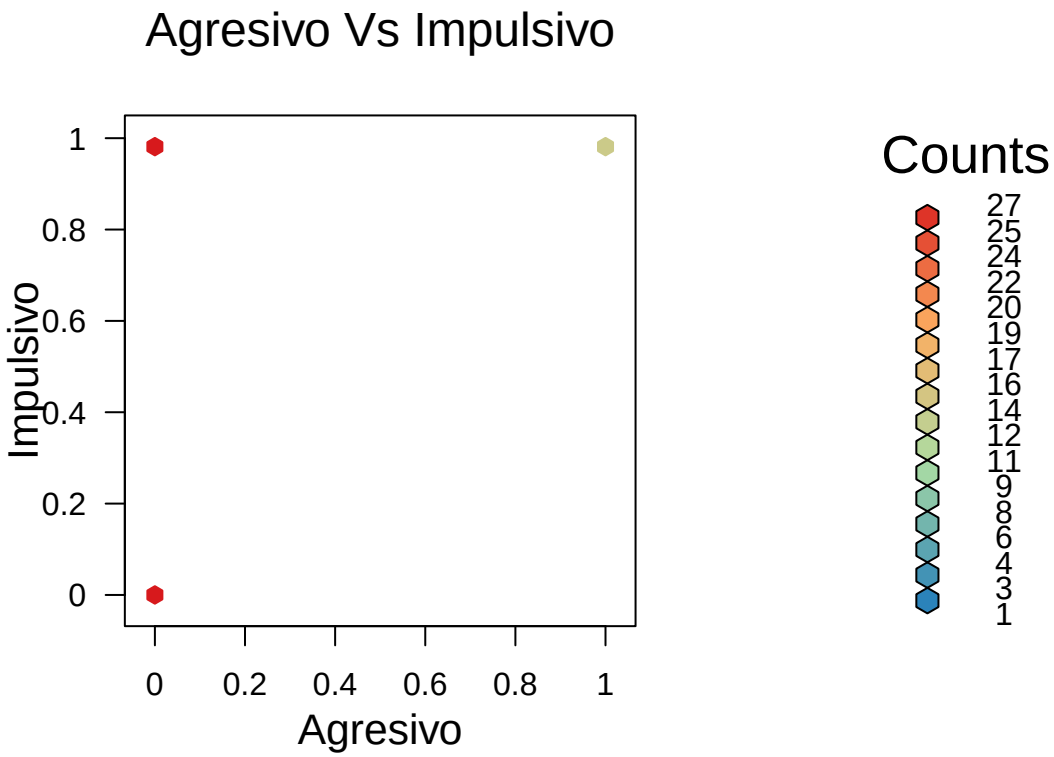
Ahora voy a sacar un plot para ver la relación entre la edad y el sexo de las personas que están en consulta

```
plot(matriz.pacientes.datos[,1], matriz.pacientes.datos[,2], xlab="Edad", ylab="Sexo (0 - mujer, 1 - hor
```



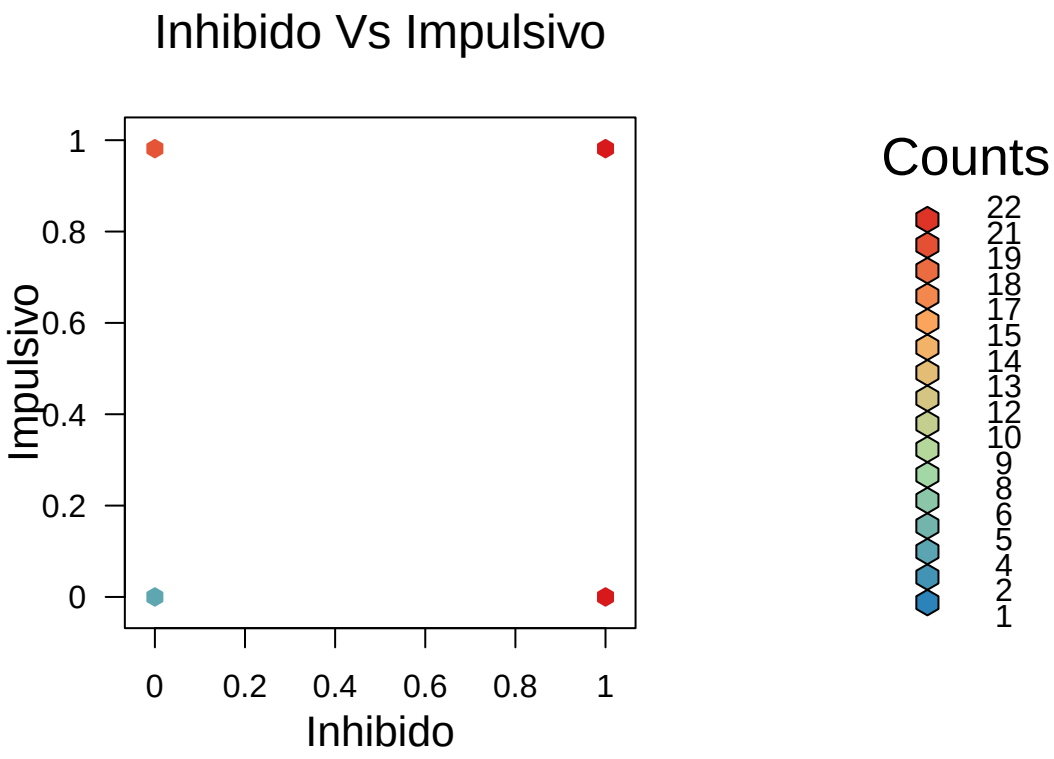
Otro plot para ver la correlación entre ser agresivo y ser impulsivo

```
#install.packages("hexbin")
#install.packages("RColorBrewer")

library(hexbin)
library(RColorBrewer)

rf <- colorRampPalette(rev(brewer.pal(4,'Spectral')))
df <- data.frame(matriz.pacientes.datos[, 23], matriz.pacientes.datos[, 24])
h <- hexbin(df)
plot(h, colramp=rf, xlab="Agresivo", ylab="Impulsivo", main="Agresivo Vs Impulsivo")
```



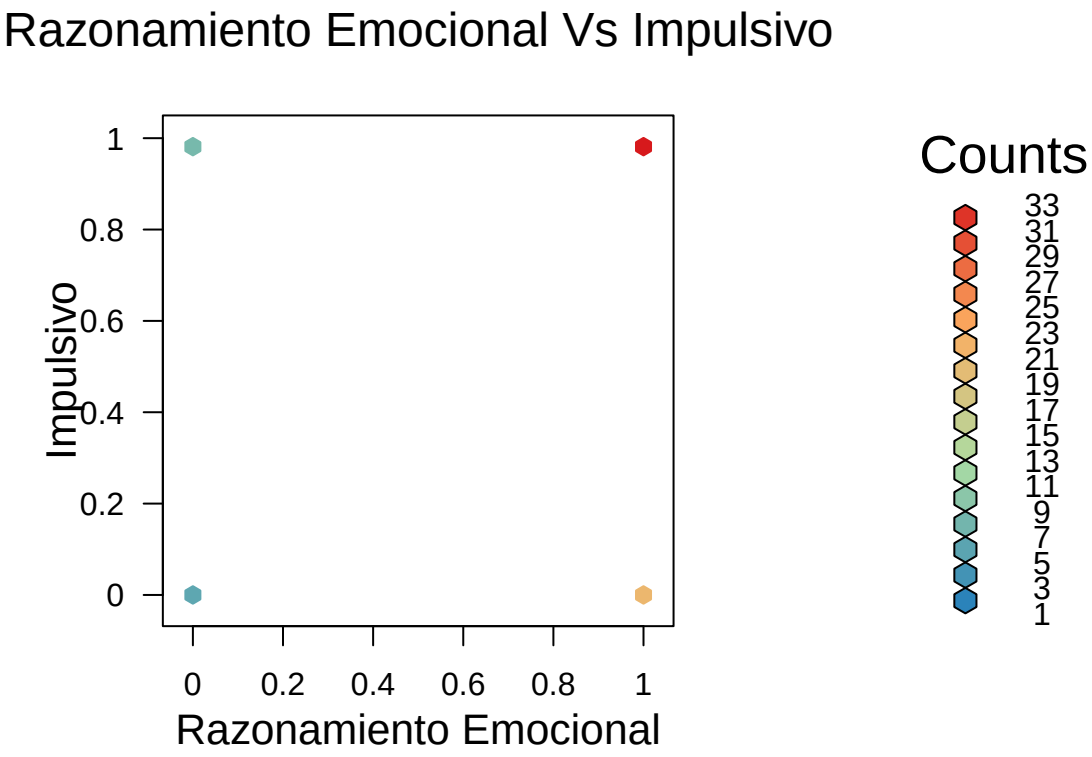
Otro plot similar para ver la relación de ser inhibido e impulsivo

```
df <- data.frame(matriz.pacientes.datos[, 21], matriz.pacientes.datos[, 24])
h <- hexbin(df)
plot(h, colramp=rf, xlab="Inhibido", ylab="Impulsivo", main="Inhibido Vs Impulsivo")
```



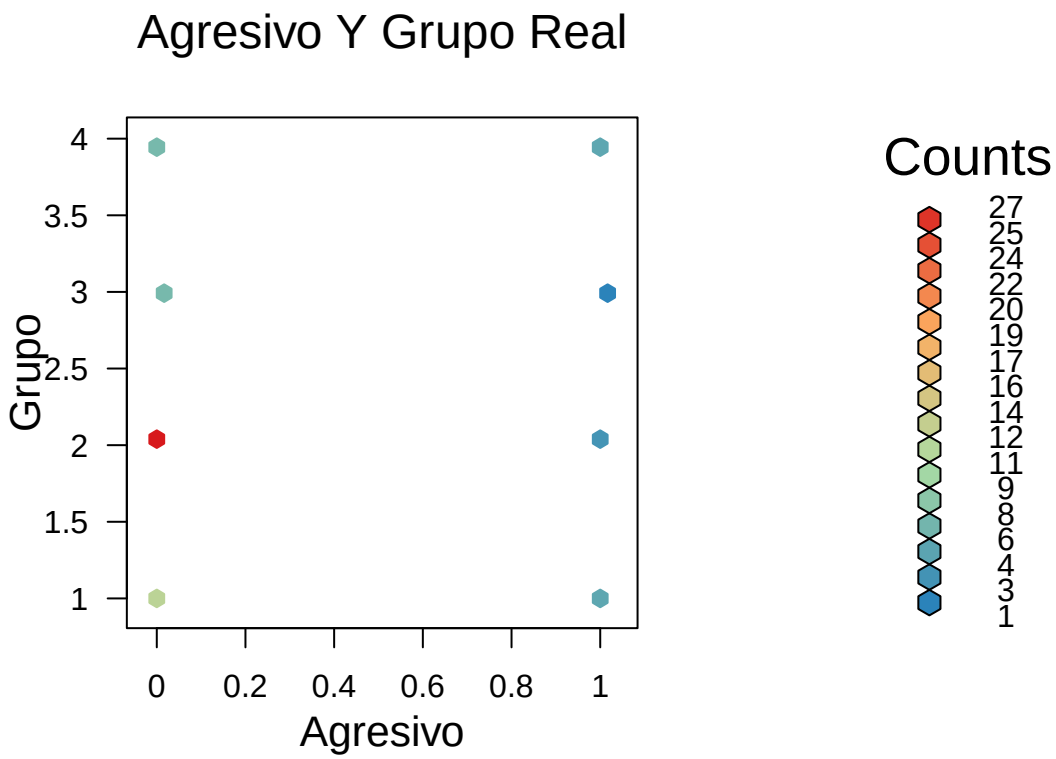
Voy a ver la relación entre el razonamiento emocional (actuar según tus sentimientos) y la impulsividad

```
df <- data.frame(matriz.pacientes.datos[, 20], matriz.pacientes.datos[, 24])
h <- hexbin(df)
plot(h, colramp=rf, xlab="Razonamiento Emocional", ylab="Impulsivo", main="Razonamiento Emocional Vs Imp
```



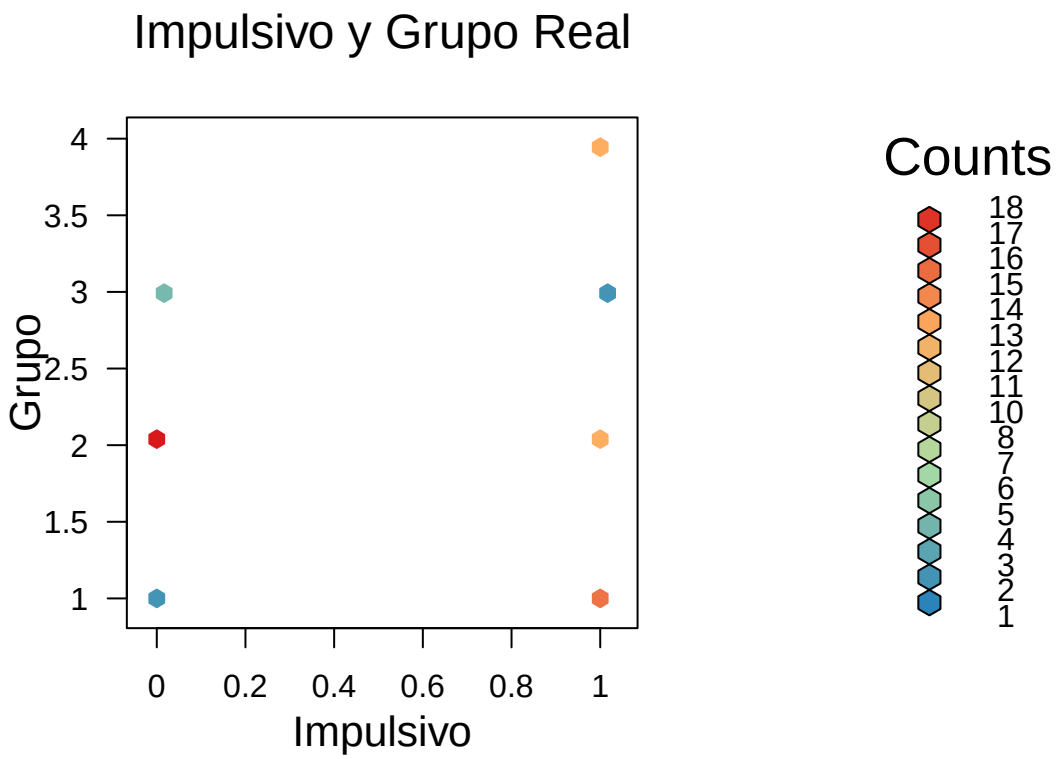
Ahora quiero sacar una relación entre ser agresivo y ver el grupo en el que están

```
rf <- colorRampPalette(rev(brewer.pal(4,'Spectral')))
df <- data.frame(matriz.pacientes.datos[, 23], matriz.pacientes.etiquetas[, 25])
h <- hexbin(df)
plot(h, colramp=rf, xlab="Agresivo", ylab="Grupo", main="Agresivo Y Grupo Real")
```



Voy a hacer lo mismo con la impulsividad

```
rf <- colorRampPalette(rev(brewer.pal(4,'Spectral')))
df <- data.frame(matriz.pacientes.datos[, 24], matriz.pacientes.etiquetas[, 25])
h <- hexbin(df)
plot(h, colramp=rf, xlab="Impulsivo", ylab="Grupo", main="Impulsivo y Grupo Real")
```



De estas gráficas estamos obteniendo información realmente interesante antes de la predicción de los datos. He preferido hacer gráficas en 2D porque las gráficas en 3D son mucho más difíciles de interpretar que estas bonitas gráficas en 2D

Vamos a ver la correlación que tienen mis variables

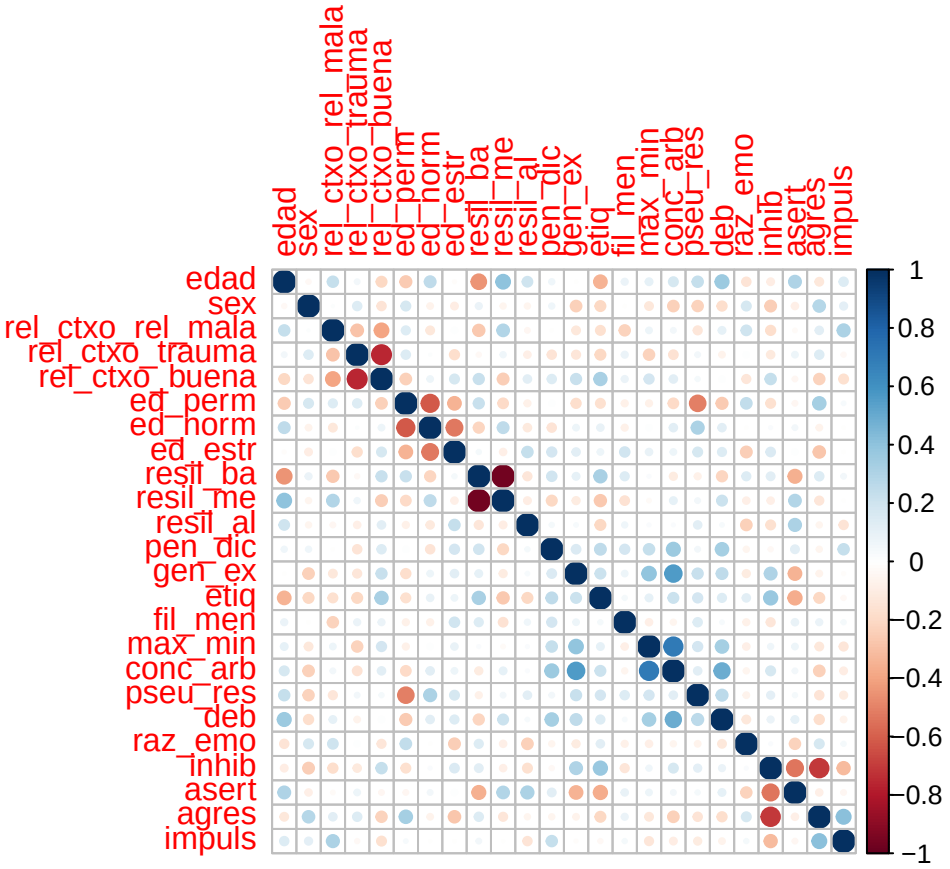
```
res <- cor(matriz.pacientes.datos[, 1:24], method = "spearman") # Por mi tipo de datos, hacemos la corr
options(width = 100)
res.round <- round(res, 2)
```

Como saca una tabla enorme, lo que voy a hacer es usar una librería que me da una función para sacar de una forma bonita las correlaciones entre las variables.

```
#install.packages("corrplot")
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot(res.round, method="circle")
```



Como podemos ver, por ejemplo, resiliencia baja y media tienen una correlación de -1, ya que si hay una no hay la otra y viceversa. Esto pasa igual con las relaciones entre contexto, ya que buena - trauma, trauma - mala, mala - buena tienen que ser inversas.

Como he comentado antes, Lo que voy a hacer ahora es un centrado y escalado de los datos de la matriz. De esta manera, la red neuronal no tendrá ningún valor que destaque especialmente y con ello no dará de inicio más peso a unos valores que a otros, ya que no lo buscamos.

Modelos de Inteligencia Artificial supervisados

Lo primero que hacemos es importar la librería caret

```
#install.packages("caret")
library(caret)
```

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

```
## Loading required package: ggplot2
```

Ahora hacemos un centrado y escalado de los datos, ya que la edad no sigue el rango del resto de valores, y distorsionaría la predicción

```
preObjeto <- preProcess(matriz.pacientes.datos, method=c("center", "scale"))  # Quiero hacer un centrad
matriz.pacientes.datos.centscal <- predict(preObjeto, matriz.pacientes.datos) # Obtengo los valores en
```

Ahora vamos a importar la librería nnet, que nos sirve para hacer perceptrones

```
#install.packages("nnet")
library(nnet)
```

Ahora lo que hago es coger un conjunto muy grande de los datos para hacer el entrenamiento

```
conjuntoEntrenamiento <- sample(1:67, 55)
```

1 NEURONA

Lo que voy a hacer ahora es entrenar la red neuronal con diferente cantidad de neuronas,y voy a ir comparando el resultado...

SIN SOFTMAX

```
pacientes.1neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz
```

```
## # weights:  33
## initial  value 56.636634
## iter  10 value 36.723735
## iter  20 value 35.636351
## iter  30 value 34.714303
## iter  40 value 34.701207
## iter  50 value 34.674997
## iter  60 value 34.539873
## iter  70 value 33.559397
## iter  80 value 33.549741
## iter  90 value 33.547230
## iter 100 value 33.546402
## final  value 33.546402
## stopped after 100 iterations
```

Lo voy a entrenar también con el SOFTMAX = true. Esto optimiza la verosimilitud, no el error cuadrático medio...

CON SOFTMAX

```
pacientes.1neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz
```

```
## # weights:  33
## initial  value 82.753522
## iter  10 value 51.290525
## iter  20 value 42.878241
## iter  30 value 40.800581
## iter  40 value 39.902762
## iter  50 value 39.377553
## iter  60 value 39.009673
## iter  70 value 38.589781
## iter  80 value 38.515379
## iter  90 value 38.500037
## iter 100 value 38.497546
## final  value 38.497546
## stopped after 100 iterations
```

Una vez que lo tengo entrenado, lo que voy a hacer es calcular el error tanto en el entrenamiento como en el test de cada uno

```
pacientes.prediccion.1neu <- predict( pacientes.1neu, matriz.pacientes.datos.centscal[conjuntoEntrenami
pacientes.prediccion.1neu # Vemos las probabilidades de pertenencia de cada valor
```

```
##                1          2          3            4
## 51 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 48 0.000000e+00 0.95645617 0.04354383 0.000000e+00
```

```
## 30 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 37 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 3  0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 24 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 47 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 61 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 53 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 34 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 1  4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 27 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 58 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 54 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 4  4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 42 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 60 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 12 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 29 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 23 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 13 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 46 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 28 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 59 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 43 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 6  4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 21 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 9  0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 32 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 45 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 17 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 25 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 26 1.084338e-06 0.24991990 0.75005080 2.822044e-05
## 55 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 20 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 65 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 15 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 40 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 10 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 5  0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 63 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 14 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 33 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 19 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 66 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 52 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 50 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 56 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 41 6.352772e-13 0.25032695 0.74967305 4.698397e-10
## 38 4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 18 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 31 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 39 0.000000e+00 0.95645617 0.04354383 0.000000e+00
## 7  4.482454e-01 0.04303722 0.12944387 3.792736e-01
## 57 1.095511e-06 0.24991956 0.75005091 2.844314e-05
```

Ahora que los tengo todos entrenados, Determinamos cual es la máxima, es decir, la clase a la que hay que asignar los objetos

```
pacientes.prediccion.1neu.class <- apply( pacientes.prediccion.1neu, MARGIN=1, FUN='which.is.max')
pacientes.prediccion.1neu.class
```

```
## 51 48 30 37  3 24 47 61 53 34  1 27 58 54  4 42 60 12 29 23 13 46 28 59 43  6 21  9 32 45 17 25 26
##  1  2  2  2  2  1  1  1  2  2  1  1  1  1  1  1  2  1  2  1  2  1  1  1  1  1  2  2  1  2  1  1  3
## 55 20 65 15 40 10  5 63 14 33 19 66 52 50 56 41 38 18 31 39  7 57
##  1  2  1  1  1  2  2  2  1  2  2  2  1  2  1  3  1  2  2  2  1  3
```

Lo visualizo en forma de tabla para ir viendo el error

```
table( pacientes.prediccion.1neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] )  # Lo v
```

```
##
## pacientes.prediccion.1neu.class  1  2  3  4
##                               1 13  2  3 11
##                               2  0 22  1  0
##                               3  0  0  3  0
```

Calculo el acierto

```
sum( diag( table( pacientes.prediccion.1neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25]
```

```
## [1] 0.6909091
```

TEST

```
pacientes.prediccion.test.1neu <- predict( pacientes.1neu, matriz.pacientes.datos.centscal[-conjuntoEnt
pacientes.prediccion.test.1neu
```

```
##            1         2         3         4
## 2  0.0000000 0.95645617 0.04354383 0.0000000
## 8  0.0000000 0.95645617 0.04354383 0.0000000
## 11 0.0000000 0.95645617 0.04354383 0.0000000
## 16 0.0000000 0.95645617 0.04354383 0.0000000
## 22 0.4482454 0.04303722 0.12944387 0.3792736
## 35 0.0000000 0.95645617 0.04354383 0.0000000
## 36 0.0000000 0.95645617 0.04354383 0.0000000
## 44 0.4482454 0.04303722 0.12944387 0.3792736
## 49 0.4482454 0.04303722 0.12944387 0.3792736
## 62 0.0000000 0.95645617 0.04354383 0.0000000
## 64 0.4482454 0.04303722 0.12944387 0.3792736
## 67 0.4482454 0.04303722 0.12944387 0.3792736
```

```
pacientes.prediccion.test.1neu.class <- apply( pacientes.prediccion.test.1neu, MARGIN=1, FUN='which.is.
pacientes.prediccion.test.1neu.class
```

```
##  2  8 11 16 22 35 36 44 49 62 64 67
##  2  2  2  2  1  2  2  1  1  2  1  1
```

```
table( pacientes.prediccion.test.1neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )
```

```
##
## pacientes.prediccion.test.1neu.class 1 2 3 4
##                                    1 2 2 1 0
##                                    2 2 4 0 1
```

```r
sum( diag( table( pacientes.prediccion.test.1neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamien
```

```
## [1] 0.5
```

2 NEURONAS

A partir de ahora voy a hacer exactamente lo mismo, por lo que haré chunks más grandes para evitar una sobrecarga de chunks, y reduciré la cantidad de comentarios, ya que serán redundantes

SIN SOFTMAX

```r
pacientes.2neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz
```

```
## # weights:  62
## initial  value 59.897480
## iter  10 value 35.958026
## iter  20 value 35.430056
## iter  30 value 35.222350
## iter  40 value 35.221236
## iter  50 value 35.217286
## iter  60 value 34.995585
## iter  70 value 32.727458
## iter  80 value 31.057841
## iter  90 value 30.898357
## iter 100 value 30.656757
## final  value 30.656757
## stopped after 100 iterations
```

CON SOFTMAX

```r
pacientes.2neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz
```

```
## # weights:  62
## initial  value 81.775080
## iter  10 value 44.390633
## iter  20 value 26.831479
## iter  30 value 22.004062
## iter  40 value 21.655630
## iter  50 value 21.531382
## iter  60 value 21.508478
## iter  70 value 21.506043
## iter  80 value 21.505202
## final  value 21.505110
## converged
```

```r
pacientes.prediccion.2neu <- predict( pacientes.2neu, matriz.pacientes.datos.centscal[conjuntoEntrenami
pacientes.prediccion.2neu # Vemos las probabilidades de pertenencia de cada valor
```

```
##             1           2         3         4
## 51 0.06666483 1.333416e-01 0.06666713 0.7333264
## 48 0.00000000 1.000000e+00 0.00000000 0.0000000
## 30 0.00000000 1.000000e+00 0.00000000 0.0000000
## 37 0.00000000 1.000000e+00 0.00000000 0.0000000
## 3  0.00000000 1.000000e+00 0.00000000 0.0000000
## 24 0.06666483 1.333416e-01 0.06666713 0.7333264
## 47 0.06666483 1.333416e-01 0.06666713 0.7333264
## 61 0.06666483 1.333416e-01 0.06666713 0.7333264
## 53 0.00000000 1.000000e+00 0.00000000 0.0000000
```

```
## 34 0.00000000 1.000000e+00 0.00000000 0.0000000
## 1  0.13332233 2.000104e-01 0.66666724 0.0000000
## 27 0.06666483 1.333416e-01 0.06666713 0.7333264
## 58 0.91666885 0.000000e+00 0.08333115 0.0000000
## 54 0.16670533 8.939212e-15 0.83329467 0.0000000
## 4  0.06666483 1.333416e-01 0.06666713 0.7333264
## 42 0.91666885 0.000000e+00 0.08333115 0.0000000
## 60 0.00000000 1.000000e+00 0.00000000 0.0000000
## 12 0.06666483 1.333416e-01 0.06666713 0.7333264
## 29 0.00000000 1.000000e+00 0.00000000 0.0000000
## 23 0.91666885 0.000000e+00 0.08333115 0.0000000
## 13 0.00000000 1.000000e+00 0.00000000 0.0000000
## 46 0.06666483 1.333416e-01 0.06666713 0.7333264
## 28 0.06666483 1.333416e-01 0.06666713 0.7333264
## 59 0.06666483 1.333416e-01 0.06666713 0.7333264
## 43 0.91666885 0.000000e+00 0.08333115 0.0000000
## 6  0.06666483 1.333416e-01 0.06666713 0.7333264
## 21 0.00000000 1.000000e+00 0.00000000 0.0000000
## 9  0.00000000 1.000000e+00 0.00000000 0.0000000
## 32 0.06666483 1.333416e-01 0.06666713 0.7333264
## 45 0.00000000 1.000000e+00 0.00000000 0.0000000
## 17 0.06666483 1.333416e-01 0.06666713 0.7333264
## 25 0.91666885 0.000000e+00 0.08333115 0.0000000
## 26 0.16667067 7.666820e-06 0.83332166 0.0000000
## 55 0.91666885 0.000000e+00 0.08333115 0.0000000
## 20 0.00000000 1.000000e+00 0.00000000 0.0000000
## 65 0.06666483 1.333416e-01 0.06666713 0.7333264
## 15 0.91666885 0.000000e+00 0.08333115 0.0000000
## 40 0.91666885 0.000000e+00 0.08333115 0.0000000
## 10 0.00000000 1.000000e+00 0.00000000 0.0000000
## 5  0.00000000 1.000000e+00 0.00000000 0.0000000
## 63 0.13332233 2.000104e-01 0.66666724 0.0000000
## 14 0.91666885 0.000000e+00 0.08333115 0.0000000
## 33 0.00000000 1.000000e+00 0.00000000 0.0000000
## 19 0.00000000 1.000000e+00 0.00000000 0.0000000
## 66 0.00000000 1.000000e+00 0.00000000 0.0000000
## 52 0.91666885 0.000000e+00 0.08333115 0.0000000
## 50 0.13332233 2.000104e-01 0.66666724 0.0000000
## 56 0.91666885 0.000000e+00 0.08333115 0.0000000
## 41 0.13332233 2.000104e-01 0.66666724 0.0000000
## 38 0.91666885 0.000000e+00 0.08333115 0.0000000
## 18 0.00000000 1.000000e+00 0.00000000 0.0000000
## 31 0.00000000 1.000000e+00 0.00000000 0.0000000
## 39 0.00000000 1.000000e+00 0.00000000 0.0000000
## 7  0.06666483 1.333416e-01 0.06666713 0.7333264
## 57 0.13332233 2.000104e-01 0.66666724 0.0000000
```

```
pacientes.prediccion.2neu.class <- apply( pacientes.prediccion.2neu, MARGIN=1, FUN='which.is.max')
pacientes.prediccion.2neu.class
```

```
## 51 48 30 37  3 24 47 61 53 34  1 27 58 54  4 42 60 12 29 23 13 46 28 59 43  6 21  9 32 45 17 25 26
##  4  2  2  2  2  4  4  4  2  2  3  4  1  3  4  1  2  4  2  1  2  4  4  4  1  4  2  2  4  2  4  1  3
## 55 20 65 15 40 10  5 63 14 33 19 66 52 50 56 41 38 18 31 39  7 57
##  1  2  4  1  1  2  2  3  1  2  2  2  1  3  1  3  1  2  2  2  4  3
```

```
table( pacientes.prediccion.2neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] )  # Lo v
```

```
##
## pacientes.prediccion.2neu.class  1  2  3  4
##                               1 11  0  1  0
##                               2  0 21  0  0
##                               3  1  1  5  0
##                               4  1  2  1 11
```

```
sum( diag( table( pacientes.prediccion.2neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25]
```

```
## [1] 0.8727273
```

TEST

```
pacientes.prediccion.test.2neu <- predict( pacientes.2neu, matriz.pacientes.datos.centscal[-conjuntoEnt
pacientes.prediccion.test.2neu
```

```
##             1         2          3         4
## 2  0.13332233 0.2000104 0.66666724 0.0000000
## 8  0.00000000 1.0000000 0.00000000 0.0000000
## 11 0.00000000 1.0000000 0.00000000 0.0000000
## 16 0.91666885 0.0000000 0.08333115 0.0000000
## 22 0.91666885 0.0000000 0.08333115 0.0000000
## 35 0.06666483 0.1333416 0.06666713 0.7333264
## 36 0.13332233 0.2000104 0.66666724 0.0000000
## 44 0.06666483 0.1333416 0.06666713 0.7333264
## 49 0.06666483 0.1333416 0.06666713 0.7333264
## 62 0.00000000 1.0000000 0.00000000 0.0000000
## 64 0.06666483 0.1333416 0.06666713 0.7333264
## 67 0.91666885 0.0000000 0.08333115 0.0000000
```

```
pacientes.prediccion.test.2neu.class <- apply( pacientes.prediccion.test.2neu, MARGIN=1, FUN='which.is.
pacientes.prediccion.test.2neu.class
```

```
##  2  8 11 16 22 35 36 44 49 62 64 67
##  3  2  2  1  1  4  3  4  4  2  4  1
```

```
table( pacientes.prediccion.test.2neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )
```

```
##
## pacientes.prediccion.test.2neu.class 1 2 3 4
##                                    1 1 0 1 1
##                                    2 2 1 0 0
##                                    3 0 2 0 0
##                                    4 1 3 0 0
```

```
sum( diag( table( pacientes.prediccion.test.2neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamien
```

```
## [1] 0.1666667
```

3 NEURONAS

SIN SOFTMAX

```
pacientes.3neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz
```

```
## # weights:  91
## initial  value 51.602123
## iter  10 value 34.735817
```

```
## iter  20 value 28.784708
## iter  30 value 24.809894
## iter  40 value 24.142779
## iter  50 value 24.096732
## iter  60 value 22.883953
## iter  70 value 22.845636
## iter  80 value 22.836067
## iter  90 value 22.828587
## iter 100 value 22.822595
## final  value 22.822595
## stopped after 100 iterations
```

CON SOFTMAX

```
pacientes.3neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz
```

```
## # weights:  91
## initial  value 87.112903
## iter  10 value 44.074851
## iter  20 value 24.688519
## iter  30 value 15.956247
## iter  40 value 15.006105
## iter  50 value 14.996652
## final  value 14.996563
## converged
```

```
pacientes.prediccion.3neu <- predict( pacientes.3neu, matriz.pacientes.datos.centscal[conjuntoEntrenami
pacientes.prediccion.3neu # Vemos las probabilidades de pertenencia de cada valor
```

```
##                1            2            3            4
## 51 3.437001e-15 5.277100e-16 6.751910e-32 1.000000e+00
## 48 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 30 2.764602e-12 1.000000e+00 4.604276e-12 1.394159e-16
## 37 4.108873e-12 4.999989e-01 5.000011e-01 1.428420e-28
## 3  1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 24 4.779891e-17 4.389356e-17 5.978533e-33 1.000000e+00
## 47 9.267421e-34 2.079466e-08 1.093352e-23 1.000000e+00
## 61 6.250071e-01 1.249988e-01 1.249691e-01 1.250249e-01
## 53 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 34 4.108873e-12 4.999989e-01 5.000011e-01 1.428420e-28
## 1  1.000000e+00 4.541863e-15 1.514132e-15 4.867077e-34
## 27 4.775623e-17 4.387076e-17 5.975506e-33 1.000000e+00
## 58 6.250110e-01 1.249979e-01 1.250079e-01 1.249832e-01
## 54 4.665772e-26 2.500063e-01 7.499937e-01 6.664262e-10
## 4  4.775623e-17 4.387076e-17 5.975506e-33 1.000000e+00
## 42 1.000000e+00 4.720237e-15 1.575719e-15 5.336528e-34
## 60 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 12 1.000000e+00 2.086239e-14 9.472371e-31 5.096528e-17
## 29 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 23 1.000000e+00 4.541863e-15 1.514132e-15 4.867077e-34
## 13 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 46 4.665772e-26 2.500063e-01 7.499937e-01 6.664262e-10
## 28 6.685982e-34 1.645594e-08 6.723482e-24 1.000000e+00
## 59 5.458243e-17 4.988164e-17 7.695351e-33 1.000000e+00
## 43 6.250110e-01 1.249979e-01 1.250079e-01 1.249832e-01
## 6  4.777193e-17 4.388463e-17 5.979227e-33 1.000000e+00
```

```
## 21 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 9  1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 32 4.665772e-26 2.500063e-01 7.499937e-01 6.664262e-10
## 45 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 17 6.685982e-34 1.645594e-08 6.723482e-24 1.000000e+00
## 25 1.000000e+00 2.086239e-14 9.472371e-31 5.096528e-17
## 26 4.665772e-26 2.500063e-01 7.499937e-01 6.664262e-10
## 55 1.000000e+00 4.541863e-15 1.514132e-15 4.867077e-34
## 20 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 65 6.250110e-01 1.249979e-01 1.250079e-01 1.249832e-01
## 15 1.000000e+00 4.542587e-15 1.514378e-15 4.867592e-34
## 40 1.000000e+00 2.086239e-14 9.472371e-31 5.096528e-17
## 10 1.795735e-12 1.000000e+00 1.489188e-16 5.939876e-12
## 5  1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 63 4.108873e-12 4.999989e-01 5.000011e-01 1.428420e-28
## 14 6.250110e-01 1.249979e-01 1.250079e-01 1.249832e-01
## 33 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 19 1.791127e-12 1.000000e+00 1.400271e-16 6.329132e-12
## 66 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 52 6.250110e-01 1.249979e-01 1.250079e-01 1.249832e-01
## 50 4.108873e-12 4.999989e-01 5.000011e-01 1.428420e-28
## 56 6.250110e-01 1.249979e-01 1.250079e-01 1.249832e-01
## 41 4.108873e-12 4.999989e-01 5.000011e-01 1.428420e-28
## 38 6.250110e-01 1.249979e-01 1.250079e-01 1.249832e-01
## 18 5.041962e-15 1.000000e+00 5.225108e-16 7.560495e-09
## 31 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 39 1.788974e-12 1.000000e+00 1.361970e-16 6.513141e-12
## 7  1.913480e-31 3.771182e-06 3.000597e-19 9.999962e-01
## 57 4.108873e-12 4.999989e-01 5.000011e-01 1.428420e-28
```

```
pacientes.prediccion.3neu.class <- apply( pacientes.prediccion.3neu, MARGIN=1, FUN='which.is.max')
pacientes.prediccion.3neu.class
```

```
## 51 48 30 37  3 24 47 61 53 34  1 27 58 54  4 42 60 12 29 23 13 46 28 59 43  6 21  9 32 45 17 25 26
##  4  2  2  3  2  4  4  1  2  3  1  4  1  3  4  1  2  1  2  1  2  3  4  4  1  4  2  2  3  2  4  1  3
## 55 20 65 15 40 10  5 63 14 33 19 66 52 50 56 41 38 18 31 39  7 57
##  1  2  1  1  1  2  2  3  1  2  2  2  1  3  1  3  1  2  2  2  4  3
```

```
table( pacientes.prediccion.3neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] )  # Lo v
```

```
##
## pacientes.prediccion.3neu.class  1  2  3  4
##                               1 13  1  1  1
##                               2  0 19  0  0
##                               3  0  4  6  0
##                               4  0  0  0 10
```

```
sum( diag( table( pacientes.prediccion.3neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25]
```

```
## [1] 0.8727273
```

TEST

```
pacientes.prediccion.test.3neu <- predict( pacientes.3neu, matriz.pacientes.datos.centscal[-conjuntoEnt
pacientes.prediccion.test.3neu
```

```
##              1           2           3           4
```

```
## 2  1.000000e+00 2.086239e-14 9.472371e-31 5.096528e-17
## 8  4.108873e-12 4.999989e-01 5.000011e-01 1.428420e-28
## 11 1.000000e+00 2.086239e-14 9.472371e-31 5.096528e-17
## 16 5.582275e-26 2.506290e-01 7.493710e-01 7.091390e-10
## 22 1.000000e+00 4.542587e-15 1.514378e-15 4.867592e-34
## 35 4.665772e-26 2.500063e-01 7.499937e-01 6.664262e-10
## 36 4.108873e-12 4.999989e-01 5.000011e-01 1.428420e-28
## 44 4.775623e-17 4.387076e-17 5.975506e-33 1.000000e+00
## 49 4.904268e-17 4.500598e-17 6.283926e-33 1.000000e+00
## 62 1.789055e-12 1.000000e+00 1.361968e-16 6.512728e-12
## 64 6.685982e-34 1.645594e-08 6.723482e-24 1.000000e+00
## 67 4.665772e-26 2.500063e-01 7.499937e-01 6.664262e-10
```

```
pacientes.prediccion.test.3neu.class <- apply( pacientes.prediccion.test.3neu, MARGIN=1, FUN='which.is.m
pacientes.prediccion.test.3neu.class
```

```
##  2  8 11 16 22 35 36 44 49 62 64 67
##  1  3  1  3  1  3  3  4  4  2  4  3
```

```
table( pacientes.prediccion.test.3neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )
```

```
##
## pacientes.prediccion.test.3neu.class 1 2 3 4
##                                    1 0 2 1 0
##                                    2 1 0 0 0
##                                    3 2 2 0 1
##                                    4 1 2 0 0
```

```
sum( diag( table( pacientes.prediccion.test.3neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamient
```

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

3 NEURONAS

Con Decay

SIN SOFTMAX

```
pacientes.3neu.decay <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( m
```

```
## # weights:  91
## initial  value 75.484406
## iter  10 value 32.670795
## iter  20 value 31.078702
## iter  30 value 30.531362
## iter  40 value 30.469562
## iter  50 value 30.467211
## iter  60 value 30.459850
## iter  70 value 30.401437
## iter  80 value 30.398366
## iter  90 value 30.398248
## iter  90 value 30.398248
## iter  90 value 30.398248
## final  value 30.398248
## converged
```

CON SOFTMAX

```r
pacientes.3neu.decay <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( r
```

```
## # weights:  91
## initial  value 78.094861
## iter  10 value 44.393486
## iter  20 value 30.038797
## iter  30 value 26.557067
## iter  40 value 23.995853
## iter  50 value 20.053173
## iter  60 value 18.756549
## iter  70 value 18.581193
## iter  80 value 18.517356
## iter  90 value 18.503952
## iter 100 value 18.500928
## final  value 18.500928
## stopped after 100 iterations
```

```r
pacientes.prediccion.3neu.decay <- predict( pacientes.3neu.decay, matriz.pacientes.datos.centscal[conju
pacientes.prediccion.3neu.decay # Vemos las probabilidades de pertenencia de cada valor
```

```
##                1            2            3            4
## 51 2.235612e-03 0.2848125657 0.0032016577 7.097502e-01
## 48 2.430427e-03 0.8728131125 0.0060824856 1.186740e-01
## 30 4.574570e-02 0.8923814687 0.0493253343 1.254749e-02
## 37 6.931612e-03 0.7864082262 0.2049069050 1.753256e-03
## 3  1.945193e-02 0.9430937527 0.0282058272 9.248489e-03
## 24 1.557333e-01 0.0027934619 0.0124290861 8.290441e-01
## 47 4.252251e-03 0.2480584763 0.0284125666 7.192767e-01
## 61 2.460642e-04 0.4754876515 0.1452237059 3.790426e-01
## 53 7.648510e-02 0.8452366964 0.0581768890 2.010131e-02
## 34 5.305634e-03 0.9621385952 0.0290373121 3.518459e-03
## 1  9.035620e-01 0.0161762222 0.0505988020 2.966299e-02
## 27 4.137039e-02 0.0008307268 0.0021764004 9.556225e-01
## 58 9.446621e-01 0.0034070822 0.0492060596 2.724718e-03
## 54 6.521892e-02 0.0019056912 0.9327935723 8.181897e-05
## 4  5.583853e-03 0.0115430988 0.0058754560 9.769976e-01
## 42 9.691623e-01 0.0030221764 0.0228416013 4.973925e-03
## 60 1.460222e-03 0.9493045232 0.0408177214 8.417533e-03
## 12 8.725989e-01 0.0080933237 0.0252658362 9.404196e-02
## 29 5.433164e-03 0.9759283674 0.0138145470 4.823921e-03
## 23 9.677738e-01 0.0032824961 0.0257880151 3.155662e-03
## 13 2.084577e-02 0.9402215194 0.0299174227 9.015288e-03
## 46 7.326527e-02 0.8487480777 0.0563828868 2.160376e-02
## 28 4.533013e-03 0.0244579372 0.0138979105 9.571111e-01
## 59 6.625908e-02 0.0009815984 0.0020061523 9.307532e-01
## 43 8.799779e-01 0.0057888856 0.0416533844 7.257982e-02
## 6  1.267553e-02 0.0019331381 0.0008938594 9.844975e-01
## 21 1.906603e-03 0.9617262240 0.0221583118 1.420886e-02
## 9  3.796228e-03 0.9716990081 0.0119018311 1.260293e-02
## 32 2.750763e-02 0.0423025592 0.8012574814 1.289323e-01
## 45 5.334157e-03 0.9760669113 0.0138634283 4.735504e-03
## 17 3.165829e-04 0.2562337680 0.0007910875 7.426586e-01
## 25 9.177405e-01 0.0077337668 0.0392244041 3.530131e-02
## 26 6.369703e-02 0.0018767537 0.9343464775 7.973479e-05
```

```
## 55 9.732575e-01 0.0024936470 0.0213877826 2.861105e-03
## 20 2.013345e-02 0.9409977014 0.0300385543 8.830294e-03
## 65 3.675979e-02 0.0097032844 0.0227592027 9.307777e-01
## 15 7.474957e-01 0.1196402300 0.1160204916 1.684359e-02
## 40 9.707451e-01 0.0027748050 0.0214999764 4.980152e-03
## 10 1.244550e-02 0.8690446626 0.0175400653 1.009698e-01
## 5  6.086983e-03 0.9694394893 0.0198601355 4.613392e-03
## 63 9.599397e-05 0.3417453459 0.6581277248 3.093533e-05
## 14 9.004135e-01 0.0052144761 0.0855555322 8.816477e-03
## 33 3.600276e-03 0.9681897722 0.0253839041 2.826048e-03
## 19 3.786060e-03 0.9180755579 0.0752393697 2.899012e-03
## 66 3.061365e-03 0.9577289695 0.0370433267 2.166339e-03
## 52 8.585199e-01 0.0434129058 0.0635709195 3.449623e-02
## 50 4.744661e-02 0.8923489022 0.0458017696 1.440272e-02
## 56 8.182112e-01 0.0072931915 0.0224436807 1.520519e-01
## 41 3.763897e-01 0.4769594741 0.1137738334 3.287703e-02
## 38 4.092923e-02 0.0027777558 0.9547301066 1.562906e-03
## 18 2.184564e-03 0.8755414068 0.0056060677 1.166680e-01
## 31 5.112084e-03 0.9751825900 0.0130134126 6.691914e-03
## 39 5.427990e-03 0.9749125827 0.0148429188 4.816508e-03
## 7  8.452284e-02 0.0485873273 0.0136770270 8.532128e-01
## 57 9.472870e-04 0.0436052942 0.9554137807 3.363812e-05
```

```
pacientes.prediccion.3neu.class.decay <- apply( pacientes.prediccion.3neu.decay, MARGIN=1, FUN='which.is
pacientes.prediccion.3neu.class.decay
```

```
## 51 48 30 37  3 24 47 61 53 34  1 27 58 54  4 42 60 12 29 23 13 46 28 59 43  6 21  9 32 45 17 25 26
##  4  2  2  2  2  4  4  2  2  2  1  4  1  3  4  1  2  1  2  1  2  2  4  4  1  4  2  2  3  2  4  1  3
## 55 20 65 15 40 10  5 63 14 33 19 66 52 50 56 41 38 18 31 39  7 57
##  1  2  4  1  1  2  2  3  1  2  2  2  1  2  1  2  3  2  2  2  4  3
```

```
table( pacientes.prediccion.3neu.class.decay, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] )
```

```
##
## pacientes.prediccion.3neu.class.decay  1  2  3  4
##                                     1 13  0  0  0
##                                     2  0 24  1  0
##                                     3  0  0  6  0
##                                     4  0  0  0 11
```

```
sum( diag( table( pacientes.prediccion.3neu.class.decay, matriz.pacientes.etiquetas[conjuntoEntrenamien
```

```
## [1] 0.9818182
```

TEST

```
pacientes.prediccion.test.3neu.decay <- predict( pacientes.3neu.decay, matriz.pacientes.datos.centscal[
pacientes.prediccion.test.3neu.decay
```

```
##               1           2           3           4
## 2  9.729745e-01 0.002545471 0.0216032341 2.876772e-03
## 8  5.409637e-03 0.952821605 0.0385913484 3.177410e-03
## 11 1.154883e-04 0.105675519 0.0002738739 8.939351e-01
## 16 2.002798e-02 0.002411291 0.0033430703 9.742177e-01
## 22 7.474957e-01 0.119640230 0.1160204916 1.684359e-02
## 35 6.929340e-03 0.001849886 0.9912033743 1.740059e-05
## 36 9.550086e-01 0.006306039 0.0343313482 4.354055e-03
```

```
## 44 6.238588e-01 0.006887186 0.0447878336 3.244661e-01
## 49 1.538406e-04 0.383553541 0.0126956913 6.035969e-01
## 62 6.309078e-04 0.427992140 0.0015912989 5.697857e-01
## 64 3.803885e-05 0.688472149 0.1903011771 1.211886e-01
## 67 9.354648e-01 0.009667761 0.0338334322 2.103399e-02
```

```
pacientes.prediccion.test.3neu.class.decay <- apply( pacientes.prediccion.test.3neu.decay, MARGIN=1, FU
pacientes.prediccion.test.3neu.class.decay
```

```
##  2  8 11 16 22 35 36 44 49 62 64 67
##  1  2  4  4  1  3  1  1  4  4  2  1
```

```
table( pacientes.prediccion.test.3neu.class.decay , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 1
```

```
##
## pacientes.prediccion.test.3neu.class.decay 1 2 3 4
##                                            1 1 3 1 0
##                                            2 1 1 0 0
##                                            3 0 1 0 0
##                                            4 2 1 0 1
```

```
sum( diag( table( pacientes.prediccion.test.3neu.class.decay, matriz.pacientes.etiquetas[-conjuntoEntrer
```

```
## [1] 0.25
```

5 NEURONAS

SIN SOFTMAX

```
pacientes.5neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz
```

```
## # weights:  149
## initial  value 48.318552
## iter  10 value 29.860073
## iter  20 value 22.850310
## iter  30 value 21.973296
## iter  40 value 21.541953
## iter  50 value 21.462933
## iter  60 value 21.218882
## iter  70 value 17.814244
## iter  80 value 16.119374
## iter  90 value 15.946398
## iter 100 value 14.885578
## final  value 14.885578
## stopped after 100 iterations
```

CON SOFTMAX

```
pacientes.5neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz
```

```
## # weights:  149
## initial  value 81.673888
## iter  10 value 24.060598
## iter  20 value 10.634271
## iter  30 value 4.124032
## iter  40 value 1.651327
## iter  50 value 1.386893
## iter  60 value 1.386305
## iter  70 value 1.386298
```

```
## final  value 1.386296
## converged
```

```
pacientes.prediccion.5neu <- predict( pacientes.5neu, matriz.pacientes.datos.centscal[conjuntoEntrenami
pacientes.prediccion.5neu # Vemos las probabilidades de pertenencia de cada valor
```

```
##                1             2             3             4
## 51  1.707806e-55  1.636642e-79 8.168345e-152  1.000000e+00
## 48 1.044236e-124  1.000000e+00  3.435749e-28  1.921334e-91
## 30  2.306771e-11  1.000000e+00  3.143227e-09  2.147959e-24
## 37 4.962740e-109  1.000000e+00  1.191306e-22  7.154387e-80
## 3  5.617144e-107  1.000000e+00  2.310428e-41  3.162828e-45
## 24  5.226303e-27  2.694179e-33  1.999566e-92  1.000000e+00
## 47  1.819177e-48  2.052287e-81 1.298515e-150  1.000000e+00
## 61  8.343496e-65  1.000000e+00  2.791183e-11  2.027561e-14
## 53 5.412979e-133  1.000000e+00  1.093131e-19  2.225631e-74
## 34 1.044357e-124  1.000000e+00  3.435815e-28  1.921465e-91
## 1   9.999999e-01  1.528502e-32  6.467995e-08  7.348302e-45
## 27  1.708290e-55  1.636513e-79 8.168694e-152  1.000000e+00
## 58  1.000000e+00 4.009072e-151  6.072202e-93  1.707478e-82
## 54  1.208930e-07  1.383545e-94  9.999999e-01  6.735026e-88
## 4   4.855087e-57  2.839988e-77 3.297639e-150  1.000000e+00
## 42  1.000000e+00 4.009534e-151  6.070664e-93  1.707874e-82
## 60 1.044236e-124  1.000000e+00  3.435749e-28  1.921334e-91
## 12  1.000000e+00 2.888815e-116 7.019136e-158  1.142728e-19
## 29 5.412979e-133  1.000000e+00  1.093131e-19  2.225631e-74
## 23  1.000000e+00  2.036428e-77  2.770632e-45  3.441089e-63
## 13 1.044236e-124  1.000000e+00  3.435749e-28  1.921334e-91
## 46  5.978589e-20  4.999847e-01  5.000151e-01  1.244036e-07
## 28  5.226303e-27  2.694179e-33  1.999566e-92  1.000000e+00
## 59  1.708404e-55  1.636929e-79 8.174336e-152  1.000000e+00
## 43  1.000000e+00 4.009072e-151  6.072202e-93  1.707478e-82
## 6   3.579157e-43  1.254411e-10  3.226450e-25  1.000000e+00
## 21 2.542787e-220  1.000000e+00  2.525446e-60 2.829126e-112
## 9  2.542787e-220  1.000000e+00  2.525446e-60 2.829126e-112
## 32  3.823973e-09  1.963167e-07  9.999998e-01  5.087267e-15
## 45 2.542787e-220  1.000000e+00  2.525446e-60 2.829126e-112
## 17  8.474182e-75  4.125135e-25 1.849379e-123  1.000000e+00
## 25  9.999999e-01  1.104920e-23  1.586391e-29  6.649637e-08
## 26 1.737431e-117  1.750221e-97  1.000000e+00 3.765046e-165
## 55  1.000000e+00 5.877315e-144  3.476114e-88  6.873160e-77
## 20  2.306771e-11  1.000000e+00  3.143227e-09  2.147959e-24
## 65  4.488977e-08  9.161515e-28  1.373503e-07  9.999998e-01
## 15  1.000000e+00  8.534347e-64  2.986336e-46  4.620413e-33
## 40  1.000000e+00 2.333599e-120 1.584492e-160  9.683291e-21
## 10 5.617144e-107  1.000000e+00  2.310428e-41  3.162828e-45
## 5  1.044236e-124  1.000000e+00  3.435749e-28  1.921334e-91
## 63 7.930875e-165  1.304788e-71  1.000000e+00 2.573103e-193
## 14  1.000000e+00 4.009072e-151  6.072202e-93  1.707478e-82
## 33 1.044236e-124  1.000000e+00  3.435749e-28  1.921334e-91
## 19 5.298318e-107  1.000000e+00  2.287953e-41  3.055368e-45
## 66 2.544783e-220  1.000000e+00  2.523425e-60 2.831832e-112
## 52  1.000000e+00 3.570228e-129  2.270815e-78  2.510092e-65
## 50 5.412979e-133  1.000000e+00  1.093131e-19  2.225631e-74
## 56  9.999999e-01  1.646390e-55  1.832955e-29  1.032508e-07
```

```
## 41  5.978589e-20  4.999847e-01  5.000151e-01  1.244036e-07
## 38  3.388688e-07  9.112436e-38  9.999997e-01  1.343287e-37
## 18  3.778674e-68  1.000000e+00  6.643083e-86  1.469217e-22
## 31 1.044236e-124  1.000000e+00  3.435749e-28  1.921334e-91
## 39 2.542787e-220  1.000000e+00  2.525446e-60 2.829126e-112
## 7  8.642833e-102  1.829687e-12 9.981664e-104  1.000000e+00
## 57 5.686926e-104  1.872542e-78  1.000000e+00 1.046343e-139
```

```
pacientes.prediccion.5neu.class <- apply( pacientes.prediccion.5neu, MARGIN=1, FUN='which.is.max')
pacientes.prediccion.5neu.class
```

```
## 51 48 30 37  3 24 47 61 53 34  1 27 58 54  4 42 60 12 29 23 13 46 28 59 43  6 21  9 32 45 17 25 26
##  4  2  2  2  2  4  4  2  2  2  1  4  1  3  4  1  2  1  2  1  2  3  4  4  1  4  2  2  3  2  4  1  3
## 55 20 65 15 40 10  5 63 14 33 19 66 52 50 56 41 38 18 31 39  7 57
##  1  2  4  1  1  2  2  3  1  2  2  2  1  2  1  3  3  2  2  2  4  3
```

```
table( pacientes.prediccion.5neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] )  # Lo v
```

```
##
## pacientes.prediccion.5neu.class  1  2  3  4
##                               1 13  0  0  0
##                               2  0 23  0  0
##                               3  0  1  7  0
##                               4  0  0  0 11
```

```
sum( diag( table( pacientes.prediccion.5neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25]
```

```
## [1] 0.9818182
```

TEST

```
pacientes.prediccion.test.5neu <- predict( pacientes.5neu, matriz.pacientes.datos.centscal[-conjuntoEnt
pacientes.prediccion.test.5neu
```

```
##                1             2             3             4
## 2   2.306771e-11  1.000000e+00  3.143227e-09  2.147959e-24
## 8  1.044236e-124  1.000000e+00  3.435749e-28  1.921334e-91
## 11 9.299379e-164  1.000000e+00 4.900422e-118  2.168396e-43
## 16  8.737860e-23 1.281129e-116  1.000000e+00 7.118984e-117
## 22  1.000000e+00  8.534347e-64  2.986336e-46  4.620413e-33
## 35 3.618952e-117  1.172167e-97  1.000000e+00 5.826460e-165
## 36  2.987226e-39  3.522385e-45  1.000000e+00  1.559491e-85
## 44  5.226303e-27  2.694179e-33  1.999566e-92  1.000000e+00
## 49  2.259420e-66  8.217415e-08  9.999999e-01  4.609703e-27
## 62 2.542787e-220  1.000000e+00  2.525446e-60 2.829126e-112
## 64 8.642833e-102  1.829687e-12 9.981664e-104  1.000000e+00
## 67  1.281486e-24  4.515794e-28  1.000000e+00  2.728910e-51
```

```
pacientes.prediccion.test.5neu.class <- apply( pacientes.prediccion.test.5neu, MARGIN=1, FUN='which.is.
pacientes.prediccion.test.5neu.class
```

```
##  2  8 11 16 22 35 36 44 49 62 64 67
##  2  2  2  3  1  3  3  4  3  2  4  3
```

```
table( pacientes.prediccion.test.5neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )
```

```
##
## pacientes.prediccion.test.5neu.class 1 2 3 4
##                                    1 0 0 1 0
```

```
##                                   2 2 2 0 0
##                                   3 2 2 0 1
##                                   4 0 2 0 0
```

```
sum( diag( table( pacientes.prediccion.test.5neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamien
```

```
## [1] 0.1666667
```

5 NEURONAS

CON DECAY

SIN SOFTMAX

```
pacientes.5neu.decay <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( m
```

```
## # weights:  149
## initial  value 61.584998
## iter  10 value 29.962380
## iter  20 value 23.586173
## iter  30 value 22.692875
## iter  40 value 22.348898
## iter  50 value 22.319200
## iter  60 value 22.316104
## final  value 22.316090
## converged
```

CON SOFTMAX

```
pacientes.5neu.decay <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( m
```

```
## # weights:  149
## initial  value 83.962973
## iter  10 value 33.975927
## iter  20 value 21.627897
## iter  30 value 20.729540
## iter  40 value 19.809326
## iter  50 value 19.123560
## iter  60 value 19.014613
## iter  70 value 18.956170
## iter  80 value 18.954756
## final  value 18.954722
## converged
```

```
pacientes.prediccion.5neu.decay <- predict( pacientes.5neu.decay, matriz.pacientes.datos.centscal[conju
pacientes.prediccion.5neu.decay # Vemos las probabilidades de pertenencia de cada valor
```

```
##              1           2           3           4
## 51 0.217630452 0.215056760 0.0038768618 5.634359e-01
## 48 0.006944402 0.959472383 0.0019591779 3.162404e-02
## 30 0.018729595 0.928544004 0.0501788688 2.547532e-03
## 37 0.044276941 0.943005007 0.0122890187 4.290333e-04
## 3  0.003154654 0.976397321 0.0031710929 1.727693e-02
## 24 0.085008871 0.016289581 0.0386461147 8.600554e-01
## 47 0.036064286 0.091733269 0.0926692824 7.795332e-01
## 61 0.221297621 0.577022307 0.0005930202 2.010871e-01
## 53 0.027401434 0.902743786 0.0095514078 6.030337e-02
## 34 0.047481832 0.938752902 0.0134001479 3.651181e-04
## 1  0.353040063 0.382062855 0.2616725565 3.224526e-03
```

```
## 27 0.007377586 0.002147741 0.0078853895 9.825893e-01
## 58 0.914734553 0.010417526 0.0308255611 4.402236e-02
## 54 0.026158141 0.018296706 0.9544165392 1.128614e-03
## 4  0.050626636 0.028428046 0.0102490727 9.106962e-01
## 42 0.911094094 0.033279875 0.0049327734 5.069326e-02
## 60 0.026443130 0.967503427 0.0044792473 1.574196e-03
## 12 0.647606386 0.273078804 0.0167104785 6.260433e-02
## 29 0.011318913 0.977000910 0.0028617070 8.818470e-03
## 23 0.991773204 0.003625700 0.0045788824 2.221324e-05
## 13 0.031882025 0.961584229 0.0061002749 4.334715e-04
## 46 0.023804282 0.893680892 0.0097845156 7.273031e-02
## 28 0.002443935 0.019466731 0.0418356872 9.362536e-01
## 59 0.007644483 0.004962353 0.0036942343 9.836989e-01
## 43 0.939160170 0.003996015 0.0416642326 1.517958e-02
## 6  0.015085343 0.051503083 0.0334529918 8.999586e-01
## 21 0.001938087 0.982261687 0.0012870569 1.451317e-02
## 9  0.003900097 0.962667343 0.0078137907 2.561877e-02
## 32 0.019262232 0.005162034 0.7056774334 2.698983e-01
## 45 0.007246396 0.975727878 0.0052478976 1.177783e-02
## 17 0.001028300 0.150424232 0.0589682641 7.895792e-01
## 25 0.964876401 0.007270332 0.0122055312 1.564774e-02
## 26 0.036919519 0.001271759 0.9577808934 4.027829e-03
## 55 0.887421703 0.003585659 0.0961130272 1.287961e-02
## 20 0.015289512 0.982853043 0.0010657052 7.917404e-04
## 65 0.039371032 0.008163250 0.0374884558 9.149773e-01
## 15 0.688955658 0.204581008 0.0697029353 3.676040e-02
## 40 0.910461105 0.013989889 0.0384226080 3.712640e-02
## 10 0.017163899 0.873458926 0.0077833802 1.015938e-01
## 5  0.012551783 0.971930982 0.0121738168 3.343419e-03
## 63 0.098360747 0.141403252 0.7591192367 1.116765e-03
## 14 0.946157758 0.003109072 0.0365531093 1.418006e-02
## 33 0.028013706 0.941076644 0.0303819297 5.277199e-04
## 19 0.003663627 0.971461795 0.0033916626 2.148292e-02
## 66 0.015492591 0.960818077 0.0198472603 3.842072e-03
## 52 0.840797678 0.052201343 0.0123316901 9.466929e-02
## 50 0.312481378 0.590515828 0.0923566839 4.646110e-03
## 56 0.938111691 0.014719340 0.0060448142 4.112415e-02
## 41 0.077185245 0.110947522 0.7139957150 9.787152e-02
## 38 0.221768726 0.000066958 0.7672201169 1.094420e-02
## 18 0.001543756 0.996383472 0.0001035794 1.969192e-03
## 31 0.004176261 0.995115037 0.0000772733 6.314289e-04
## 39 0.005057288 0.969753599 0.0044563526 2.073276e-02
## 7  0.051439769 0.034490571 0.0152794033 8.987903e-01
## 57 0.116934672 0.031289539 0.8515497279 2.260613e-04
```

```
pacientes.prediccion.5neu.decay.class <- apply( pacientes.prediccion.5neu.decay, MARGIN=1, FUN='which.is
pacientes.prediccion.5neu.decay.class
```

```
## 51 48 30 37  3 24 47 61 53 34  1 27 58 54  4 42 60 12 29 23 13 46 28 59 43  6 21  9 32 45 17 25 26
##  4  2  2  2  2  4  4  2  2  2  2  4  1  3  4  1  2  1  2  1  2  2  4  4  1  4  2  2  3  2  4  1  3
## 55 20 65 15 40 10  5 63 14 33 19 66 52 50 56 41 38 18 31 39  7 57
##  1  2  4  1  1  2  2  3  1  2  2  2  1  2  1  3  3  2  2  2  4  3
```

```
table( pacientes.prediccion.5neu.decay.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] )
```

```
##
## pacientes.prediccion.5neu.decay.class  1  2  3  4
##                                    1 12  0  0  0
##                                    2  1 24  0  0
##                                    3  0  0  7  0
##                                    4  0  0  0 11
```

```
sum( diag( table( pacientes.prediccion.5neu.decay.class, matriz.pacientes.etiquetas[conjuntoEntrenamien
```

```
## [1] 0.9818182
```

TEST

```
pacientes.prediccion.test.decay.5neu <- predict( pacientes.5neu.decay, matriz.pacientes.datos.centscal[
pacientes.prediccion.test.decay.5neu
```

```
##              1           2           3           4
## 2  0.775581325 0.220232869 0.0017791335 0.0024066731
## 8  0.045811152 0.941658328 0.0121858387 0.0003446818
## 11 0.113653280 0.474170752 0.0043132124 0.4078627562
## 16 0.535190741 0.060372131 0.2324167068 0.1720204213
## 22 0.688955658 0.204581008 0.0697029353 0.0367603989
## 35 0.018442039 0.050241905 0.8630416665 0.0682743895
## 36 0.616323895 0.290764519 0.0926945117 0.0002170742
## 44 0.132332753 0.007202813 0.4369239263 0.4235405075
## 49 0.112197913 0.298114284 0.0008499258 0.5888378769
## 62 0.001540155 0.975243361 0.0009282543 0.0222882288
## 64 0.001612660 0.069448407 0.0264174203 0.9025215124
## 67 0.193951389 0.049076568 0.5833198873 0.1736521563
```

```
pacientes.prediccion.test.decay.5neu.class <- apply( pacientes.prediccion.test.decay.5neu, MARGIN=1, FU
pacientes.prediccion.test.decay.5neu.class
```

```
##  2  8 11 16 22 35 36 44 49 62 64 67
##  1  2  2  1  1  3  1  3  4  2  4  3
```

```
table( pacientes.prediccion.test.decay.5neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 1
```

```
##
## pacientes.prediccion.test.decay.5neu.class 1 2 3 4
##                                          1 0 2 1 1
##                                          2 2 1 0 0
##                                          3 1 2 0 0
##                                          4 1 1 0 0
```

```
sum( diag( table( pacientes.prediccion.test.decay.5neu.class, matriz.pacientes.etiquetas[-conjuntoEntrer
```

```
## [1] 0.08333333
```

Obtención de Resultados de Perceptrón

Importo los datos:

```
dataset.resultados <- read.csv2("C:/Users/jorge/Desktop/Documentos Clase/Universidad/4ºCarrera/1er Cuati
```

Ahora voy a sacar un gráfico donde comparo los resultados.

```r
#install.packages("plotly")
library("plotly")
```

```
##
## Attaching package: 'plotly'
## The following object is masked from 'package:ggplot2':
##
##     last_plot
## The following object is masked from 'package:stats':
##
##     filter
## The following object is masked from 'package:graphics':
##
##     layout
```

```r
tipos = dataset.resultados[, 1]
real = dataset.resultados[, 2]
practico = dataset.resultados[, 3]

p<- plot_ly(dataset.resultados, x = ~tipos, y = ~real, type = 'bar', name = 'Real') %>% add_trace(y = ~

p
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

```r
#Mostramos el gráfico interactivo
```

KNN

```r
#install.packages("class")
library("class")

# Para hacer la predicción con knn, voy a coger los grupos de una manera distinta:

conjuntoEntrenamiento = matriz.pacientes.datos.centscal[1:55, 1:24]
conjuntoTest = matriz.pacientes.datos.centscal[56:67, 1:24]
# Utilizo por supuesto la matriz de centrado y escalado

etiquetasEntrenamiento = matriz.pacientes.etiquetas[1:55, 25]
etiquetasTest = matriz.pacientes.etiquetas[56:67, 25]
```

```r
conjuntoEntrenamiento
conjuntoTest
etiquetasEntrenamiento
etiquetasTest
```

Para K = 8...

```r
# K = 8

prediccion.knn.8 <- knn(train = conjuntoEntrenamiento, test = conjuntoTest, cl = etiquetasEntrenamiento
prediccion.knn.8
```

```
##  [1] 1 2 2 1 2 1 2 2 2 1 2 2
```

```
## Levels: 1 2 3 4
```

Sacamos crosstable:

```
#install.packages("gmodels")
library("gmodels")
```

```
CrossTable(x = etiquetasTest , y = prediccion.knn.8, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  12
##
##
##                | prediccion.knn.8
## etiquetasTest |         1 |         2 | Row Total |
## --------------|-----------|-----------|-----------|
##             1 |         1 |         3 |         4 |
##               |     0.250 |     0.750 |     0.333 |
##               |     0.250 |     0.375 |           |
##               |     0.083 |     0.250 |           |
## --------------|-----------|-----------|-----------|
##             2 |         1 |         3 |         4 |
##               |     0.250 |     0.750 |     0.333 |
##               |     0.250 |     0.375 |           |
##               |     0.083 |     0.250 |           |
## --------------|-----------|-----------|-----------|
##             3 |         0 |         2 |         2 |
##               |     0.000 |     1.000 |     0.167 |
##               |     0.000 |     0.250 |           |
##               |     0.000 |     0.167 |           |
## --------------|-----------|-----------|-----------|
##             4 |         2 |         0 |         2 |
##               |     1.000 |     0.000 |     0.167 |
##               |     0.500 |     0.000 |           |
##               |     0.167 |     0.000 |           |
## --------------|-----------|-----------|-----------|
##  Column Total |         4 |         8 |        12 |
##               |     0.333 |     0.667 |           |
## --------------|-----------|-----------|-----------|
##
##
```

Para K = 6

```
# K = 6
```

```r
prediccion.knn.6 <- knn(train = conjuntoEntrenamiento, test = conjuntoTest, cl = etiquetasEntrenamiento
prediccion.knn.6
```

```
##  [1] 1 2 2 1 2 1 2 2 2 1 2 2
## Levels: 1 2 3 4
```

```r
CrossTable(x = etiquetasTest , y = prediccion.knn.6, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  12
##
##
##                | prediccion.knn.6
## etiquetasTest |         1 |         2 | Row Total |
## --------------|-----------|-----------|-----------|
##             1 |         1 |         3 |         4 |
##               |     0.250 |     0.750 |     0.333 |
##               |     0.250 |     0.375 |           |
##               |     0.083 |     0.250 |           |
## --------------|-----------|-----------|-----------|
##             2 |         1 |         3 |         4 |
##               |     0.250 |     0.750 |     0.333 |
##               |     0.250 |     0.375 |           |
##               |     0.083 |     0.250 |           |
## --------------|-----------|-----------|-----------|
##             3 |         0 |         2 |         2 |
##               |     0.000 |     1.000 |     0.167 |
##               |     0.000 |     0.250 |           |
##               |     0.000 |     0.167 |           |
## --------------|-----------|-----------|-----------|
##             4 |         2 |         0 |         2 |
##               |     1.000 |     0.000 |     0.167 |
##               |     0.500 |     0.000 |           |
##               |     0.167 |     0.000 |           |
## --------------|-----------|-----------|-----------|
##  Column Total |         4 |         8 |        12 |
##               |     0.333 |     0.667 |           |
## --------------|-----------|-----------|-----------|
##
##
```

Para k = 10

```r
# K = 10
```

```
prediccion.knn.10 <- knn(train = conjuntoEntrenamiento, test = conjuntoTest, cl = etiquetasEntrenamiento
prediccion.knn.10
```

```
##  [1] 1 2 2 1 2 1 2 2 4 1 2 2
## Levels: 1 2 3 4
```

```
CrossTable(x = etiquetasTest , y = prediccion.knn.10, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  12
##
##
##                | prediccion.knn.10
## etiquetasTest |         1 |         2 |         4 | Row Total |
## --------------|-----------|-----------|-----------|-----------|
##             1 |         1 |         3 |         0 |         4 |
##               |     0.250 |     0.750 |     0.000 |     0.333 |
##               |     0.250 |     0.429 |     0.000 |           |
##               |     0.083 |     0.250 |     0.000 |           |
## --------------|-----------|-----------|-----------|-----------|
##             2 |         1 |         2 |         1 |         4 |
##               |     0.250 |     0.500 |     0.250 |     0.333 |
##               |     0.250 |     0.286 |     1.000 |           |
##               |     0.083 |     0.167 |     0.083 |           |
## --------------|-----------|-----------|-----------|-----------|
##             3 |         0 |         2 |         0 |         2 |
##               |     0.000 |     1.000 |     0.000 |     0.167 |
##               |     0.000 |     0.286 |     0.000 |           |
##               |     0.000 |     0.167 |     0.000 |           |
## --------------|-----------|-----------|-----------|-----------|
##             4 |         2 |         0 |         0 |         2 |
##               |     1.000 |     0.000 |     0.000 |     0.167 |
##               |     0.500 |     0.000 |     0.000 |           |
##               |     0.167 |     0.000 |     0.000 |           |
## --------------|-----------|-----------|-----------|-----------|
##  Column Total |         4 |         7 |         1 |        12 |
##               |     0.333 |     0.583 |     0.083 |           |
## --------------|-----------|-----------|-----------|-----------|
##
##
```

Como se puede observar, la mejor predicción la hemos hecho con K = 8

Random Forest

Ahora voy a implementar una solución mediante Random Forest:

Lo primero que hacemos es importar el paquete de Random Forest

```
#install.packages("randomForest")
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.2
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Una vez instalado e importado, lo que tengo que hacer es crear el Random Forest, y ejecutarlo...

```
model <- randomForest(grupo ~ ., data = dataset[2:26], importance = TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer unique values. Are you
## sure you want to do regression?
```

```
model
```

```
##
## Call:
##  randomForest(formula = grupo ~ ., data = dataset[2:26], importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 1.315377
##                     % Var explained: -26.55
```

Ahora lo voy a hacer con 10 fold X Validation:

```
result <- rfcv(dataset[2:26], dataset$grupo, cv.fold=10)
```

```
## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?
```

```
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?
```

```
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?
```

```
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?
```

```
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?
```

```
## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?
```

```
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
```

```
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or
## fewer unique values. Are you sure you want to do regression?
result

## $n.var
## [1] 25 12  6  3  1
##
## $error.cv
##           25           12            6            3            1
## 3.112446e-01 2.227883e-01 1.575984e-01 2.482962e-01 3.522388e-06
##
## $predicted
## $predicted$`25`
##  [1] 1.967048 1.905633 2.005329 3.280100 1.903424 3.474869 2.764486 1.589926 2.071252 2.037490
## [11] 2.078429 2.012467 1.817522 2.262862 1.931722 3.316509 2.828005 1.796239 2.065219 2.028365
## [21] 2.186924 2.713544 1.515740 2.882112 1.689901 2.833359 3.443325 3.057927 2.207481 2.018743
## [31] 1.855771 3.333664 1.865047 1.845703 2.075057 1.801829 1.959792 2.927078 1.900692 1.396010
## [41] 2.731887 1.441329 1.470389 1.933091 1.948533 1.920256 3.287975 1.984159 1.795112 2.012932
## [51] 2.864326 1.568090 1.989112 2.952422 1.852968 1.585895 2.809879 1.675684 3.008480 2.131276
## [61] 1.863086 1.795348 2.703935 2.236784 3.049072 1.985911 1.721845
##
## $predicted$`12`
##  [1] 1.824742 1.948653 2.003032 3.453962 1.927750 3.618671 3.037030 1.510172 2.087948 2.011466
## [11] 2.021743 1.702170 1.906264 2.115551 1.746643 3.413158 2.889324 2.000222 2.020324 1.991375
## [21] 2.108648 2.772111 1.551192 2.731215 1.490866 2.817975 3.721914 3.087761 2.149823 1.960785
## [31] 1.800437 3.295616 1.839795 1.856312 1.989177 1.884366 2.023721 2.966985 1.998390 1.403873
## [41] 2.803019 1.363609 1.415907 1.864036 1.945294 1.920927 3.441420 2.061682 1.657582 2.063549
## [51] 3.113406 1.494480 2.143767 2.873342 1.638609 1.345186 2.842398 1.641233 3.284709 2.053197
## [61] 1.833923 1.719750 2.791375 2.067947 3.352351 1.974174 1.705471
##
## $predicted$`6`
##  [1] 1.587314 1.930347 1.992559 3.612336 1.937350 3.748767 3.322148 1.440330 2.004623 1.928768
## [11] 1.961051 1.686625 1.902031 2.089464 1.739571 3.443246 2.983041 1.906911 2.024820 2.020913
## [21] 2.004623 2.699701 1.465046 2.768577 1.415562 2.897264 3.703338 3.460793 1.897061 2.007129
## [31] 1.825398 3.134131 1.999207 1.955697 2.023207 1.895634 1.964184 3.055692 2.044360 1.232690
## [41] 2.777112 1.327819 1.247297 1.852929 2.004623 2.016821 3.480499 2.023709 1.335248 1.990733
## [51] 3.359114 1.513956 1.894510 2.870243 1.524013 1.344203 2.863673 1.537556 3.424529 1.986865
## [61] 1.921134 1.674951 2.892843 1.938388 3.676537 1.823713 1.523903
```

```
##
## $predicted$`3`
##  [1] 1.682126 1.869038 2.069927 3.383949 1.935628 3.383949 3.004554 1.633047 2.008166 2.060431
## [11] 1.977034 1.755870 1.869038 1.737358 1.770671 3.338571 3.051559 1.952709 2.001278 1.977034
## [21] 2.008166 2.590660 1.676767 2.562213 1.419402 2.704634 3.321051 3.053990 1.910890 2.055358
## [31] 1.906921 2.967284 2.064443 2.004972 1.910890 1.906921 1.935628 2.749014 2.082037 1.535323
## [41] 2.615711 1.535323 1.508498 2.043596 2.008166 1.890025 3.355060 2.001278 1.528970 2.001278
## [51] 3.100863 1.682126 2.107786 2.999342 1.740290 1.608470 2.811937 1.740290 3.293795 2.036674
## [61] 1.997109 1.765482 2.803576 2.051618 3.312996 1.997109 1.648607
##
## $predicted$`1`
##  [1] 1.000 2.000 2.000 4.000 2.000 4.000 4.000 1.000 2.000 2.000 2.000 1.000 2.000 1.000 1.000 4.000
## [17] 4.000 2.000 2.000 2.000 2.000 2.998 1.000 4.000 1.000 2.996 4.000 4.000 2.000 2.000 2.000 2.990
## [33] 2.000 2.000 2.000 2.000 2.000 2.996 2.000 1.000 3.000 1.000 1.000 2.000 2.000 2.000 4.000 2.000
## [49] 1.000 2.000 4.000 1.000 2.000 3.000 1.000 1.000 2.990 1.000 4.000 2.000 2.000 1.000 3.000 2.000
## [65] 4.000 2.000 1.000
```

Podemos ver el error, bajo la variable $error.cv, y podemos ver las predicciones que se han hecho para cada una de las n.var.

SVM de Kernel Lineal

Lo bueno que tiene SVM es que es muy robusto frente a la dimensión, por lo que deberíamos de obtener a priori buenos resultados con este método.

Lo primero que hay que hacer es importar la librería...

```
#install.packages("e1071")
library("e1071")
```

```
## Warning: package 'e1071' was built under R version 3.5.2
```

Con este método no necesito tener un conjunto de entrenaminento y otro de test, por lo que sigo adelante.

Ahora que hemos instalado la librería, vamos a crear el SVM:

```
modelo_svm <- svm(grupo ~ ., data=dataset[2:26], kernel="linear")
summary(modelo_svm)
```

```
##
## Call:
## svm(formula = grupo ~ ., data = dataset[2:26], kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.04166667
##     epsilon:  0.1
##
##
## Number of Support Vectors:  64
```

Ahora que tenemos creado este primer modelo, toca predecir:

```
prediccion <- predict(modelo_svm,dataset[,2:25])
prediccion
```

```
##        1        2        3        4        5        6        7        8        9       10       11
## 2.710785 1.167141 2.103294 2.272653 1.921987 2.951614 2.215446 1.903501 2.343303 2.364138 1.624035
##       12       13       14       15       16       17       18       19       20       21       22
## 1.264225 1.134063 3.102210 1.472399 3.897730 2.237491 1.897509 2.142401 1.897167 2.651879 1.472399
##       23       24       25       26       27       28       29       30       31       32       33
## 1.102637 1.498541 1.595378 2.586326 3.896878 2.912379 1.897073 2.064304 1.392893 2.897015 2.102379
##       34       35       36       37       38       39       40       41       42       43       44
## 1.897257 2.712875 1.764464 1.925354 2.268604 1.897013 1.237698 1.952647 1.724457 2.010257 2.102934
##       45       46       47       48       49       50       51       52       53       54       55
## 2.102424 2.102300 2.865796 2.103005 2.080632 2.155359 1.844763 2.016170 1.897797 2.896991 2.171207
##       56       57       58       59       60       61       62       63       64       65       66
## 2.075540 2.088427 1.103007 2.142982 2.187305 2.102484 2.155098 2.270547 2.102423 3.897624 2.103058
##       67
## 2.053638
```

El problema que tenemos con estas predicciones es que están siendo contínuas, y no discretas, por lo que las voy a discretizar redondeando:

```
prediccion <- round(prediccion, digits = 0)
prediccion
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
##  3  1  2  2  2  3  2  2  2  2  2  1  1  3  1  4  2  2  2  2  3  1  1  1  2  3  4  3  2  2  1  3  2
## 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
##  2  3  2  2  2  2  1  2  2  2  2  2  2  3  2  2  2  2  2  2  3  2  2  2  1  2  2  2  2  2  2  4  2
## 67
##  2
```

Ahora que hemos predicho, tenemos que sacar la matriz de confusión:

```
matriz.conf <- table(prediccion, dataset[,26])
matriz.conf
```

```
##
## prediccion  1  2  3  4
##          1  5  3  1  1
##          2 10 25  4  5
##          3  2  2  3  3
##          4  0  0  0  3
```

```
sum(diag(matriz.conf))/67
```

```
## [1] 0.5373134
```

Obtenemos un porcentaje de acierto medio, pero esto es sin tener en cuenta que los pacientes del grupo 2 pueden pertenecer al 1, lo cual suma alrededor de 10 pacientes más, por lo que obtendríamos valores mucho más altos que rondarían el 65-70% de acierto.

SVM de Kernel RBF

```
modelo_svm.radial <- svm(grupo ~ ., data=dataset[2:26], kernel="radial")
summary(modelo_svm.radial)
```

```
##
## Call:
## svm(formula = grupo ~ ., data = dataset[2:26], kernel = "radial")
##
##
```

```
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.04166667
##     epsilon:  0.1
##
##
## Number of Support Vectors:  60
```

Ahora que tenemos creado este primer modelo, toca predecir:

```
prediccion.radial <- predict(modelo_svm.radial,dataset[,2:25])
prediccion.radial
```

```
##        1        2        3        4        5        6        7        8        9       10       11
## 1.549973 1.897491 2.102992 2.931869 1.958610 2.933492 2.460079 1.968849 2.102839 2.102609 2.102858
##       12       13       14       15       16       17       18       19       20       21       22
## 1.834806 1.897158 1.792309 2.081453 3.345387 2.644334 1.896957 2.102818 2.101751 2.102659 2.081453
##       23       24       25       26       27       28       29       30       31       32       33
## 1.102789 2.171444 1.688974 2.247323 3.098435 2.310026 2.102645 2.090946 1.897066 2.757770 2.102934
##       34       35       36       37       38       39       40       41       42       43       44
## 1.935061 2.103190 1.897447 2.078152 2.427070 1.897235 1.396473 2.724358 1.577114 1.558528 1.911562
##       45       46       47       48       49       50       51       52       53       54       55
## 2.102722 2.093837 2.558446 2.102649 1.964595 2.102263 1.784159 1.755280 2.102747 2.897155 1.865630
##       56       57       58       59       60       61       62       63       64       65       66
## 1.404453 2.647560 1.341993 2.806776 2.103103 2.102609 1.945673 2.749015 2.102654 3.127086 2.102353
##       67
## 1.482761
```

El problema que tenemos con estas predicciones es que están siendo contínuas, y no discretas, por lo que las voy a discretizar redondeando:

```
prediccion.radial <- round(prediccion.radial, digits = 0)
prediccion.radial
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
##  2  2  2  3  2  3  2  2  2  2  2  2  2  2  2  3  3  2  2  2  2  2  1  2  2  2  3  2  2  2  2  3  2
## 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
##  2  2  2  2  2  2  1  3  2  2  2  2  2  3  2  2  2  2  2  2  3  2  1  3  1  3  2  2  2  3  2  3  2
## 67
##  1
```

Ahora que hemos predicho, tenemos que sacar la matriz de confusión:

```
matriz.conf.radial <- table(prediccion.radial, dataset[,26])
matriz.conf.radial
```

```
##
## prediccion.radial  1  2  3  4
##                1  5  0  0  0
##                2 12 30  3  4
##                3  0  0  5  8
```

```
sum(diag(matriz.conf.radial))/67
```

```
## [1] 0.5970149
```

Obtenemos un acierto del 60%, al que hay que sumar otros 12 pacientes. ### Si añadimos estos pacientes,

nos encontramos con un acierto del 77% Por lo tanto, SVM de Kernel radial es una buena técnica para la predicción en este problema.

Ahora pasamos a los modelos de inteligencia artificial no supervisados:

Modelos de inteligencia artificial no supervisados

El primer modelo de inteligencia artificial no supervisado que voy a usar es un modelo de clustering llamado Dendrograma.

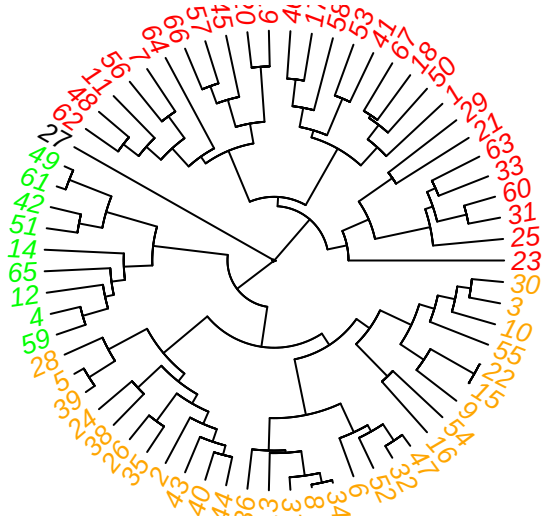
Dendrograma

Para esto, lo que voy a hacer es dividirlo en 4 clusters, coincidiendo con los 4 grupos de trastornos que tengo.

```
#install.packages("ape")
library(ape)
```

```
## Warning: package 'ape' was built under R version 3.5.2
```

```
dd <- dist(scale(dataset[,2:25]), method = "euclidean") #Nos basamos en la distancia euclídea
hier.clust <- hclust(dd, method = "ward.D2")
colores.dendrograma = c("red", "orange", "green", "black")
cluster.4 = cutree(hier.clust, 4)
plot(as.phylo(hier.clust), type = "fan", tip.color = colores.dendrograma[cluster.4], label.offset = 0.3
```

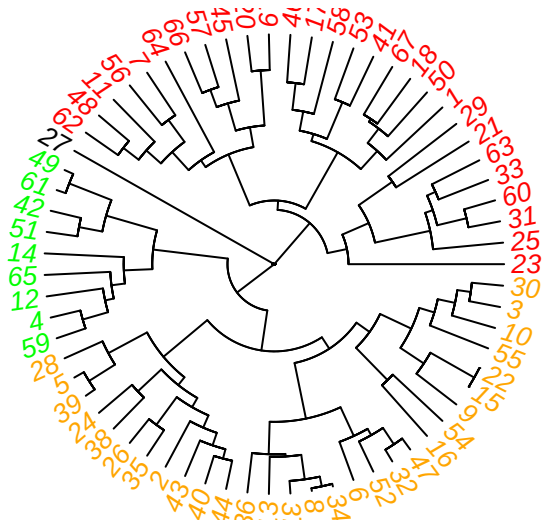


Como vemos, estamos obteniendo el indentificador de cada paciente en el dendrograma, donde los pacientes

que mas se parecen estarán más juntos, mientras que los que menos se parecen estarán más separados. Es interesante analizar como los pacientes verdes y los naranjas surgen de la misma salida del centro, cosa que no ocurre con los rojos y los negros, lo cual quiere decir que algo tienen en común estos dos tipos de casos.

Ahora voy a hacer el mismo dendrograma pero con el DataSet de centrado y escalado, de tal manera que veamos a ver si hay diferencias:

```
dd <- dist(scale(matriz.pacientes.datos.centscal), method = "euclidean") #Nos basamos en la distancia e
hier.clust <- hclust(dd, method = "ward.D2")
colores.dendrograma = c("red", "orange", "green", "black")
cluster.4 = cutree(hier.clust, 4)
plot(as.phylo(hier.clust), type = "fan", tip.color = colores.dendrograma[cluster.4], label.offset = 0.3
```



Si lo comparamos, vemos que hemos obtenido exactamente el mismo resultado, por lo que en este caso el centrado y escalado no es necesario.

Vamos a analizar algunos pacientes aleatoriamente para ver si ha acertado, o al menos si se ha aproximado:

Paciente 1: Analizado - Rojo ——— Real: 1 Paciente 2: Analizado - Naranja —— Real: 2 Paciente 3: Analizado - Naranja —— Real: 2 Paciente 4: Analizado - Verde ——— Real: 4 Paciente 5: Analizado - Naranja —— Real: 2 Paciente 27: Analizado - Negro ——- Real: 4

Es decir, a la vista de estos resultados, podemos concluir que el grupo 1 es el de los pacientes en rojo, el grupo 2 es el de los pacientes en naranja, y luego entre el grupo 3 y el grupo 4 hay dudas, pero teniendo varios pacientes tanto del grupo 3 como del grupo 4 en nuestro DataSet parece ser que algo de error ha cometido.

K-Means

El algoritmo KMeans en principio no es el algoritmo más adecuado para este trabajo, ya que se basa en círculos para la clasificación de los individuos, cuando en principio en mis datos esto no es así. De todas formas, voy a clasificar a los pacientes siguiendo este algoritmo para comprobar la eficacia que tiene sobre mi problema:

```
set.seed(76964057) #Seed para reproducibilidad
k <-kmeans(dataset[,2:25], centers=4) #Creo 4 clusters
k$centers #Muestro los centros
```

```
##       edad       sex rel_ctxo_rel_mala rel_ctxo_trauma rel_ctxo_buena    ed_perm   ed_norm
## 1 16.61111 0.2777778        0.05555556       0.3888889      0.5555556 0.61111111 0.1666667
## 2 23.82609 0.1739130        0.08695652       0.3043478      0.6086957 0.08695652 0.6956522
## 3 30.94118 0.1176471        0.17647059       0.3529412      0.4705882 0.11764706 0.6470588
## 4 44.44444 0.3333333        0.33333333       0.4444444      0.2222222 0.44444444 0.3333333
##     ed_estr  resil_ba   resil_me  resil_al   pen_dic    gen_ex      etiq   fil_men   max_min
## 1 0.2222222 0.9444444 0.05555556 0.0000000 0.8333333 0.9444444 0.7777778 0.7222222 0.9444444
## 2 0.2173913 0.4782609 0.52173913 0.0000000 0.9130435 0.9565217 1.0000000 0.7826087 0.9565217
## 3 0.2352941 0.4705882 0.52941176 0.0000000 0.9411765 1.0000000 0.5882353 0.8235294 1.0000000
## 4 0.2222222 0.2222222 0.66666667 0.1111111 0.8888889 0.8888889 0.3333333 0.8888889 1.0000000
##    conc_arb  pseu_res       deb   raz_emo     inhib      asert     agres    impuls
## 1 0.9444444 0.2222222 0.7777778 0.8333333 0.6666667 0.00000000 0.3333333 0.5000000
## 2 1.0000000 0.6956522 1.0000000 0.9130435 0.7826087 0.08695652 0.1304348 0.5652174
## 3 1.0000000 0.6470588 1.0000000 0.7058824 0.5294118 0.23529412 0.2352941 0.7647059
## 4 1.0000000 0.3333333 1.0000000 0.5555556 0.5555556 0.33333333 0.1111111 0.6666667
```

```
table(k$cluster) #Número de puntos en cada cluster
```

```
##
##  1  2  3  4
## 18 23 17  9
```

Interpretando estos resultados, obtenemos:

El cluster 1 destaca por sexo más hacia masculino que otros, una relación contexto ciertamente buena, una educación permisiva, una resiliencia baja, maximización y minimización, razonamiento emocional, cierta inhibición y poca agresividad.

El cluster 2 destaca por una edad mayor, es el cluster con mejor relación con el contexto, y suelen tener las personas de este cluster una educación normal. Destaca por una resiliencia media, pensamiento dicotómico, generalización excesiva, etiquetado, conclusiones arbitrarias, deberías, razonamiento emocional e inhibición.

El cluster número 3 destaca por tener una edad aún más elevada, más ratio de personas del sexo femenino que ningún otro cluster, y tienen una relación con el contexto bastante variable. La educación de estas personas es principalmente normal, con una resiliencia que puede ser tanto baja como media. Destacan por el pensamiento dicotómico, generalización excesiva, poco etiquetado, maximización y minimización, filtro mental, conclusiones arbitrarias, pseudoresponsabilidad, deberías, y suelen ser bastante inhibidos e impulsivos.

Finalmente, el cluster 4 destaca por ser el que tiene la edad más elevada y el ratio de sexo más masculino. La relación con el contexto de estos individuos clasificados en este grupo es principalmente de trauma, aunque también hay buenas y malas. La educación de estos individuos es principalmente permisiva, y la resiliciencia tiende a media. Destacan por la poca etiquetación que hacen, pero un gran fitro mental, conclusiones arbitrarias, poca pseudo-responsabilidad, muchos deberías, poco razonamiento emocional, y son principalmente inhibidos e impulsivos.

Finalmente, podemos ver como ha introducido a 18 individuos en el cluster 1, 23 en el cluster 2, 17 en el cluster 3 y 9 en el cluster 4.