codigoTFG

Jorge de Andrés

12 de enero de 2019

Borramos el environment lo primero para siempre tenerlo limpio en la ejecución:

rm(list=ls())

Importamos todas las librerías que vamos a ir necesitando:

#install.packages("ggplot2")  
#install.packages("caret")  
#install.packages("plyr")  
#install.packages("wordcloud")  
#install.packages("hexbin")  
#install.packages("RColorBrewer")  
#install.packages("corrplot")  
#install.packages("FactoMineR")  
#devtools::install\_github("kassambara/factoextra")  
#install.packages("factoextra")  
#install.packages("nnet")  
#install.packages("plotly")  
#install.packages("class")  
#install.packages("gmodels")  
#install.packages("randomForest")  
#install.packages("e1071")  
#install.packages("ape")  
#install.packages("cluster")  
#install.packages("fpc")  
#install.packages("RWeka")  
  
library("ggplot2")  
library("caret")

## Loading required package: lattice

library("plyr")  
library("wordcloud")

## Loading required package: RColorBrewer

library("hexbin")  
library("RColorBrewer")  
library("corrplot")

## corrplot 0.84 loaded

library("FactoMineR")  
library("factoextra")

## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ

library("nnet")  
library("plotly")

##   
## Attaching package: 'plotly'

## The following objects are masked from 'package:plyr':  
##   
## arrange, mutate, rename, summarise

## 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

library("class")  
library("gmodels")  
library("randomForest")

## 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

library("e1071")  
library("ape")  
library("cluster")  
library("fpc")  
library("RWeka")

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

dataset <- read.csv("Datos/datos.txt", header = TRUE)

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])

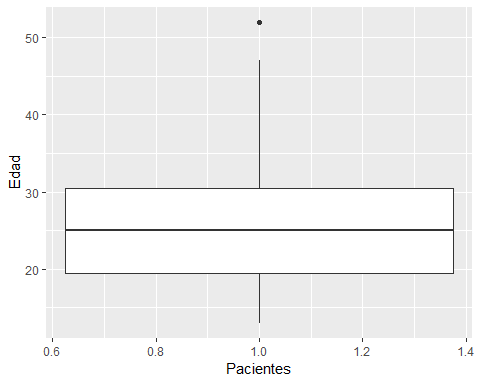
## [1] 26.46269

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

## [1] 13 52

Voy a ver estos datos gráficamente:

qplot(1, matriz.pacientes.datos[, 1], xlab = "Pacientes", ylab = "Edad", geom="boxplot")

 Pasamos el qplot a PDF:

pdf("Imágenes Obtenidas/boxplotEdadPacientes.pdf")  
  
qplot(1, matriz.pacientes.datos[, 1], xlab = "Pacientes", ylab = "Edad", geom="boxplot")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

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.

# Preparación de los datos

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.

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 centrado y escalado  
matriz.pacientes.datos.centscal <- predict(preObjeto, matriz.pacientes.datos) # Obtengo los valores en la matriz centscal

Después del preprocesado, aunque con los datos no preprocesados, voy a hacer la visualización de algunas relaciones entre variables, de tal manera que podamos ver gráficamente algunos aspectos interesantes:

## Visualización de Datos

Para empezar voy a sacar una nube de palabras para mostrar los nombres más comúnes en los datos facilitados:

# Lo primero que tengo que hacer es contar la frecuencia de los nombres  
  
dataNombres <- ddply(dataset,.(nom),nrow)  
dataNombres <- dataNombres[order(dataNombres$V1, decreasing = TRUE), ]

Una vez que tengo los nombres contados y ordenados, es el momento de crear la WordCloud

set.seed(9999) # Para el mantenimiento del mismo patrón  
  
wordcloud(words = dataNombres$nom, freq = dataNombres$V1, min.freq = 1, random.order=FALSE, rot.per=0.5, colors=c("Orange","Purple","Pink", "Red", "Yellow", "Green", "Blue", "Black"))

## Warning in wordcloud(words = dataNombres$nom, freq = dataNombres$V1,  
## min.freq = 1, : Yolanda could not be fit on page. It will not be plotted.



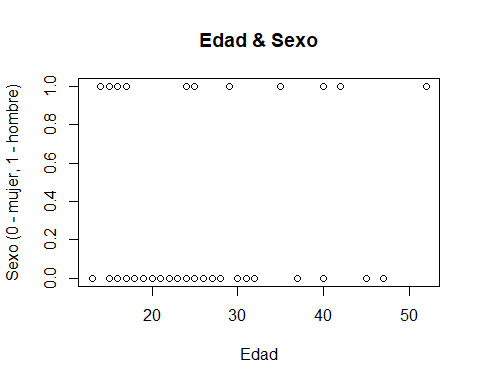
Lo pasamos a PDF:

set.seed(9999)  
  
pdf("Imágenes Obtenidas/wordcloudNombresPacientes.pdf")  
  
wordcloud(words = dataNombres$nom, freq = dataNombres$V1, min.freq = 1, random.order=FALSE, rot.per=0.5, colors=c("Orange","Purple","Pink", "Red", "Yellow", "Green", "Blue", "Black"))  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

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 - hombre)", main="Edad & Sexo")



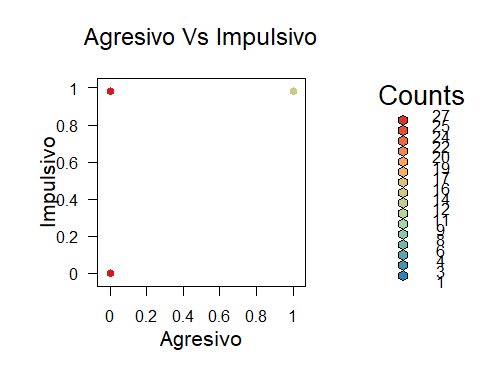
Lo pasamos a PDF:

pdf("Imágenes Obtenidas/GráficoEdad-Sexo.pdf")  
  
plot(matriz.pacientes.datos[,1], matriz.pacientes.datos[,2], xlab="Edad", ylab="Sexo (0 - mujer, 1 - hombre)", main="Edad & Sexo")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

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

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")



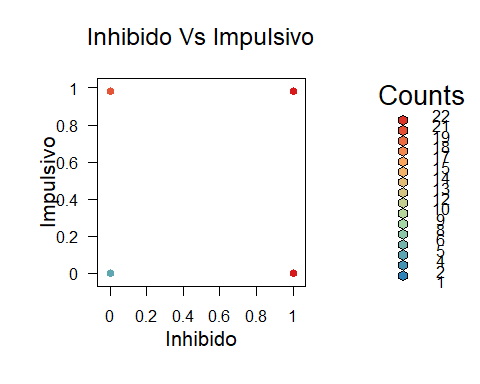
Lo pasamos a PDF:

pdf("Imágenes Obtenidas/GraficoAgresivoVsImpulsivo.pdf")  
  
plot(h, colramp=rf, xlab="Agresivo", ylab="Impulsivo", main="Agresivo Vs Impulsivo")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

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")



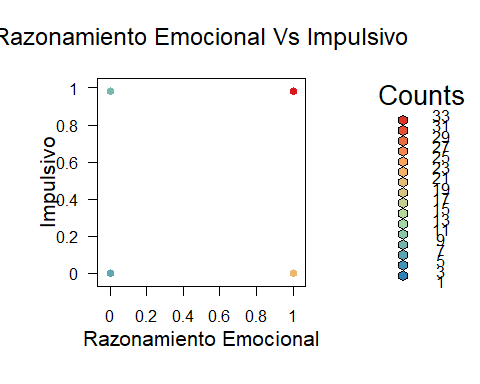
Lo guardo en PDF:

pdf("Imágenes Obtenidas/GraficoInhibidoVsImpulsivo.pdf")  
  
plot(h, colramp=rf, xlab="Inhibido", ylab="Impulsivo", main="Inhibido Vs Impulsivo")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

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 Impulsivo")



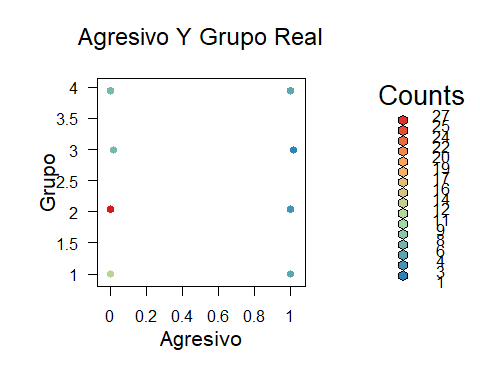
Lo guardo en PDF:

pdf("Imágenes Obtenidas/GraficoRazonamientoEmocionalVsImpulsivo.pdf")  
  
plot(h, colramp=rf, xlab="Razonamiento Emocional", ylab="Impulsivo", main="Razonamiento Emocional Vs Impulsivo")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

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")



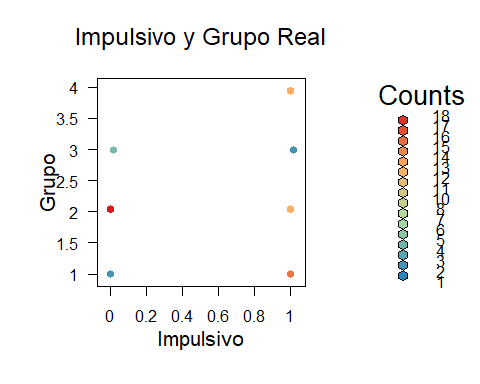
Lo guardo en PDF:

pdf("Imágenes Obtenidas/GraficoAgresivoVsGrupo.pdf")  
  
plot(h, colramp=rf, xlab="Agresivo", ylab="Grupo", main="Agresivo Y Grupo Real")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

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")



Lo guardo en PDF:

pdf("Imágenes Obtenidas/GraficoImpulsivoVsGrupo.pdf")  
  
plot(h, colramp=rf, xlab="Impulsivo", ylab="Grupo", main="Impulsivo y Grupo Real")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

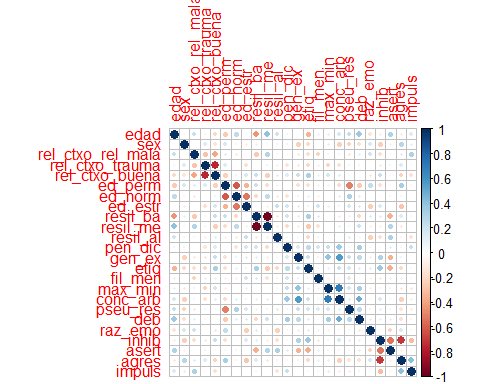
De estas gráficas estamos obteniendo información realmente interesante antes de la predicción de los datos. He preferido hacer gráficas en 2D porque las gráficas en 3D son mucho más difíciles de interpretar que estas bonitas gráficas en 2D

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 correlación por spearman  
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.

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



Guardamos la matriz de correlación en PDF para tener mejor visualización:

pdf("Imágenes Obtenidas/Corrplot.pdf")  
  
corrplot(res.round, method="circle")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

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.

Ahora voy a sacar un PCA para ver la importancia de las variables:

Para los cálculos, uso la matriz con el centrado y escalado ya hechos

resultado.pca <- PCA(matriz.pacientes.datos.centscal, graph = FALSE)  
  
#Con la siguiente línea podemos ver que podemos hacer con esto calculado  
print(resultado.pca)

## \*\*Results for the Principal Component Analysis (PCA)\*\*  
## The analysis was performed on 67 individuals, described by 24 variables  
## \*The results are available in the following objects:  
##   
## name description   
## 1 "$eig" "eigenvalues"   
## 2 "$var" "results for the variables"   
## 3 "$var$coord" "coord. for the variables"   
## 4 "$var$cor" "correlations variables - dimensions"  
## 5 "$var$cos2" "cos2 for the variables"   
## 6 "$var$contrib" "contributions of the variables"   
## 7 "$ind" "results for the individuals"   
## 8 "$ind$coord" "coord. for the individuals"   
## 9 "$ind$cos2" "cos2 for the individuals"   
## 10 "$ind$contrib" "contributions of the individuals"   
## 11 "$call" "summary statistics"   
## 12 "$call$centre" "mean of the variables"   
## 13 "$call$ecart.type" "standard error of the variables"   
## 14 "$call$row.w" "weights for the individuals"   
## 15 "$call$col.w" "weights for the variables"

Nos interesa ver los eigenvalues, que son los que presentarán la cantidad de varianza que aportan las variables:

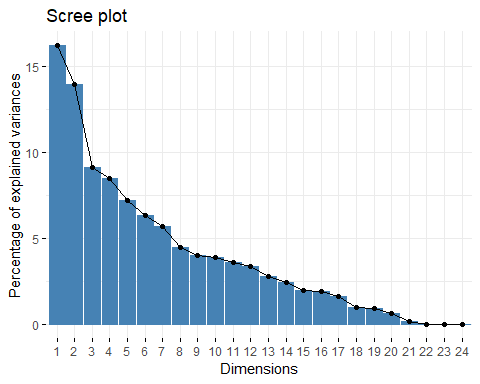
eigenvalues.PCA <- resultado.pca$eig  
eigenvalues.PCA

## eigenvalue percentage of variance cumulative percentage of variance  
## comp 1 3.896946e+00 1.623727e+01 16.23727  
## comp 2 3.348839e+00 1.395349e+01 30.19077  
## comp 3 2.189584e+00 9.123265e+00 39.31403  
## comp 4 2.044520e+00 8.518834e+00 47.83287  
## comp 5 1.737900e+00 7.241252e+00 55.07412  
## comp 6 1.521215e+00 6.338397e+00 61.41252  
## comp 7 1.374042e+00 5.725176e+00 67.13769  
## comp 8 1.079722e+00 4.498843e+00 71.63653  
## comp 9 9.591848e-01 3.996603e+00 75.63314  
## comp 10 9.311536e-01 3.879807e+00 79.51294  
## comp 11 8.644377e-01 3.601824e+00 83.11477  
## comp 12 8.099267e-01 3.374695e+00 86.48946  
## comp 13 6.658121e-01 2.774217e+00 89.26368  
## comp 14 5.935233e-01 2.473014e+00 91.73669  
## comp 15 4.698651e-01 1.957771e+00 93.69447  
## comp 16 4.632196e-01 1.930082e+00 95.62455  
## comp 17 3.922638e-01 1.634433e+00 97.25898  
## comp 18 2.445767e-01 1.019069e+00 98.27805  
## comp 19 2.251255e-01 9.380229e-01 99.21607  
## comp 20 1.497768e-01 6.240699e-01 99.84014  
## comp 21 3.836592e-02 1.598580e-01 100.00000  
## comp 22 9.366318e-32 3.902633e-31 100.00000  
## comp 23 8.328156e-32 3.470065e-31 100.00000  
## comp 24 3.135473e-32 1.306447e-31 100.00000

Como se puede comprobar, de las 24 variables (componentes) que tenemos, la mitad de la varianza la conseguimos con aproximadamente 5 variables. También se puede ver que a parti de las 17 variables prácticamente no hay un aumento de la varianza. En el caso de un problema grande, sería interesante la eliminación de algunas de las variables, para dejar un dataset más pequeño con el que poder trabajar. En nuestro caso, nuestro problema es pequeño, y además las variables están escogidas a mano, por lo que no haré una reducción del dataset.

Ahora, para completar este apartado de PCA, lo que voy a hacer es sacar la gráfica de la varianza acumulada con los valores anteriores:

plotPCA <- fviz\_screeplot(resultado.pca, ncp=24)  
plot(plotPCA)



Obtengo este gráfico en PDF para tener una mejor visualización:

pdf("Imágenes Obtenidas/GraficoEigenvalues.pdf")  
  
plot(plotPCA)  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

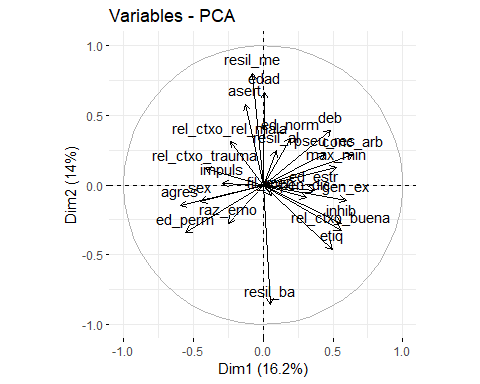
Ahora voy a sacar un “Factor Map” de las variables. Esto lo puedo hacer gracias a las coordenadas que me da una de las variables tras hacer el PCA. Así, voy primero a ver la tabla y luego voy a sacar el mapa:

resultado.pca$var$coord

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5  
## edad 0.007991017 0.66451493 0.19586260 0.007841951 0.02805228  
## sex -0.447913174 -0.11533971 0.10245651 0.057192465 0.11492883  
## rel\_ctxo\_rel\_mala -0.234645431 0.31242667 0.16869825 0.565844840 -0.35019764  
## rel\_ctxo\_trauma -0.417381244 0.11742230 -0.21001522 -0.136376092 0.34458474  
## rel\_ctxo\_buena 0.560339731 -0.32571613 0.08634947 -0.255162421 -0.09161091  
## ed\_perm -0.553915390 -0.34063941 0.23874352 0.241008740 -0.33289199  
## ed\_norm 0.195717913 0.33691768 -0.54551596 0.069259334 0.57202966  
## ed\_estr 0.364218109 -0.03574847 0.39611368 -0.343671891 -0.32610946  
## resil\_ba 0.051112365 -0.85796492 0.20216746 -0.039061871 0.21241196  
## resil\_me -0.074743641 0.80191325 -0.29969255 0.153867444 -0.18199709  
## resil\_al 0.095173049 0.24393919 0.39293737 -0.466258490 -0.12766293  
## pen\_dic 0.311964031 -0.08886036 0.58511514 0.186141485 0.18431008  
## gen\_ex 0.595148670 -0.10912103 0.08286185 0.282531851 0.16255400  
## etiq 0.499365039 -0.45912251 -0.17281954 0.173075672 0.14741419  
## fil\_men 0.059354773 -0.07552104 0.31916448 -0.355626217 0.39559535  
## max\_min 0.524773891 0.12325520 0.31096225 0.378637566 -0.01643986  
## conc\_arb 0.645068936 0.21765964 0.22093906 0.466271365 0.04542650  
## pseu\_res 0.443972323 0.22314014 -0.11950533 -0.176808668 0.43526671  
## deb 0.484524206 0.38834362 0.18502988 0.268386661 0.11400205  
## raz\_emo -0.251049993 -0.27394959 -0.20413968 0.433531845 0.02071493  
## inhib 0.563528317 -0.27772651 -0.48974596 0.018053816 -0.35529325  
## asert -0.126327074 0.57800397 0.34437650 -0.343315766 0.04021108  
## agres -0.591302668 -0.14500954 0.29219301 0.239210208 0.43081535  
## impuls -0.289816690 0.01403726 0.32193103 0.339678159 0.27914883

Como se puede ver, me está poniendo mis 24 variables en 5 dimensiones, con unas coordenadas concretas. Ahora, lo que voy a hacer, es representarlo. Con esta representación podré sacar algunas conclusiones:

fviz\_pca\_var(resultado.pca)



Exporto a PDF este gráfico de coordenadas:

pdf("Imágenes Obtenidas/GraficoVectoresVariablesPCA.pdf")  
  
fviz\_pca\_var(resultado.pca)  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

Con esto puedo sacar conclusiones al igual que con el gran gráfico de correlaciones de variables, solo que esta representación está intencionada para más de 2 dimensiones.

Puedo ver algunas de las conclusiones fáciles que saqué anteriormente, como que resiliencia media es contraria a baja, o que la relación con el contexto de trauma y mala son contrarias a buena.

Otras relaciones también puedo ver, como que los deberías y el razonamiento emocional parecen ser ciertamente contrarios, o que el filtro mental no depende de prácticamente nada ya que está en todo el centro.

También es importante ver como, mediante dos componentes principales (dos dimensiones), solo estoy explicando un del 30,2% del total, lo que es muy poco. Por unirlo con los gráficos anteriores, estas dos componentes que se han elegido como x e y son las dos variables que más varianza (y por lo tanto, explicación) tenían en el gráfico de barras anterior.

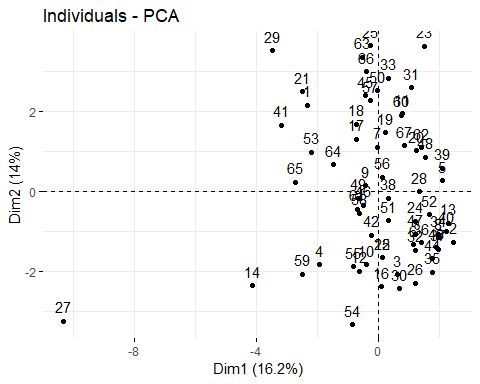
Ahora mi siguiente paso es sacar un gráfico de los individuos, para ver donde están colocados en este sistema:

head(resultado.pca$ind$coord) # Solo saco los primeros para no ocupar demasiado espacio

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5  
## 1 -2.3243690 2.147815 -1.1849618 2.4481512 -0.7586328  
## 2 2.4647257 -1.262473 0.2217190 -1.1784100 -1.4473760  
## 3 0.6387125 -2.080331 -0.1818521 0.7676582 -2.0265412  
## 4 -1.9384395 -1.832160 1.4628618 1.1820858 1.1852182  
## 5 2.0986406 0.262897 -0.2150152 -0.7686587 -1.2663434  
## 6 1.1578332 -1.323444 -0.8453683 0.9774806 -0.1661987

Ahora, tras ver que todos mis individuos tienen unas ciertas coordenadas, vamos a representarlos gráficamente:

fviz\_pca\_ind(resultado.pca)



Exporto a PDF las coordenadas de los individuos:

pdf("Imágenes Obtenidas/GraficoIndividuosPCA.pdf")  
  
fviz\_pca\_ind(resultado.pca)  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

Se puede ver que la mayoría de los pacientes están en torno al centro, mientras que tenemos un outlayer, que es el número 27.

# Modelos de Inteligencia Artificial supervisados

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

set.seed(1)  
  
dataframe.resultados.1neu <- data.frame(Ent\_1Neu = numeric(),  
 Test\_1Neu = numeric())  
  
for(i in 1:20)  
{  
 pacientes.1neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=1)  
  
 #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[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.1neu) # Vemos las probabilidades de pertenencia de cada valor  
   
 # 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  
   
 # Lo visualizo en forma de tabla para ir viendo el error  
   
 table( pacientes.prediccion.1neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
 #Calculo el acierto  
   
 acierto.ent.teorico.1neu <- sum( diag( table( pacientes.prediccion.1neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 #TEST  
   
 pacientes.prediccion.test.1neu <- predict( pacientes.1neu, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.1neu  
   
 pacientes.prediccion.test.1neu.class <- apply( pacientes.prediccion.test.1neu, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.1neu.class  
   
 table( pacientes.prediccion.test.1neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.test.teorico.1neu <- sum( diag( table( pacientes.prediccion.test.1neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
 dataframe.pasada <- data.frame(Ent\_1Neu = acierto.ent.teorico.1neu,  
 Test\_1Neu = acierto.test.teorico.1neu)  
   
 dataframe.resultados.1neu <- rbind(dataframe.resultados.1neu, dataframe.pasada)  
  
}

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

set.seed(1)  
  
dataframe.resultados.1neu.soft <- data.frame(Ent\_1Neu\_soft = numeric(),  
 Test\_1Neu\_soft = numeric())  
  
for(i in 1:20)  
{  
 pacientes.1neu.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=1, softmax = T )  
  
 #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.softmax <- predict( pacientes.1neu.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.1neu.softmax) # Vemos las probabilidades de pertenencia de cada valor  
   
 # 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.softmax <- apply( pacientes.prediccion.1neu.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.1neu.class.softmax  
   
 # Lo visualizo en forma de tabla para ir viendo el error  
   
 table( pacientes.prediccion.1neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
 #Calculo el acierto  
   
 acierto.ent.teorico.1neu.soft <- sum( diag( table( pacientes.prediccion.1neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 #TEST  
   
 pacientes.prediccion.test.1neu.softmax <- predict( pacientes.1neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.1neu.softmax  
   
 pacientes.prediccion.test.1neu.class.softmax <- apply( pacientes.prediccion.test.1neu.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.1neu.class.softmax  
   
 table( pacientes.prediccion.test.1neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.test.teorico.1neu.soft <- sum( diag( table( pacientes.prediccion.test.1neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
 dataframe.pasada <- data.frame(Ent\_1Neu\_soft = acierto.ent.teorico.1neu.soft,  
 Test\_1Neu\_soft = acierto.test.teorico.1neu.soft)  
   
 dataframe.resultados.1neu.soft <- rbind(dataframe.resultados.1neu.soft ,dataframe.pasada)  
   
}

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

set.seed(1)  
  
dataframe.resultados.2neu <- data.frame(Ent\_2Neu = numeric(),  
 Test\_2Neu = numeric())  
  
for(i in 1:20)  
{  
  
 pacientes.2neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=2 )  
   
 pacientes.prediccion.2neu <- predict( pacientes.2neu, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.2neu) # Vemos las probabilidades de pertenencia de cada valor  
   
 pacientes.prediccion.2neu.class <- apply( pacientes.prediccion.2neu, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.2neu.class  
   
   
 table( pacientes.prediccion.2neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
 acierto.teorico.entrenamiento.2neu <- sum( diag( table( pacientes.prediccion.2neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 # TEST  
   
 pacientes.prediccion.test.2neu <- predict( pacientes.2neu, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.2neu  
   
 pacientes.prediccion.test.2neu.class <- apply( pacientes.prediccion.test.2neu, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.2neu.class  
   
 table( pacientes.prediccion.test.2neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.2neu <- sum( diag( table( pacientes.prediccion.test.2neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_2Neu = acierto.teorico.entrenamiento.2neu,  
 Test\_2neu = acierto.teorico.test.2neu)  
   
 dataframe.resultados.2neu <- rbind(dataframe.resultados.2neu, dataframe.pasada)  
   
   
}

CON SOFTMAX

set.seed(1)  
  
dataframe.resultados.2neu.soft <- data.frame(Ent\_2Neu\_soft = numeric(),  
 Test\_2Neu\_soft = numeric())  
  
for(i in 1:20)  
{  
  
 pacientes.2neu.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=2, softmax = T )  
   
 pacientes.prediccion.test.2neu.softmax <- predict( pacientes.2neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.test.2neu.softmax)  
   
 pacientes.prediccion.test.2neu.class.softmax <- apply( pacientes.prediccion.test.2neu.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.2neu.class.softmax  
   
 table( pacientes.prediccion.test.2neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.ent.2neu.softmax <- sum( diag( table( pacientes.prediccion.test.2neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
 # TEST  
   
 pacientes.prediccion.test.2neu.softmax <- predict( pacientes.2neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.2neu.softmax  
   
 pacientes.prediccion.test.2neu.class.softmax <- apply( pacientes.prediccion.test.2neu.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.2neu.class.softmax  
   
 table( pacientes.prediccion.test.2neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.2neu.softmax <- sum(diag(table(pacientes.prediccion.test.2neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_2Neu\_soft = acierto.teorico.ent.2neu.softmax,  
 Test\_2neu\_soft = acierto.teorico.test.2neu.softmax)  
   
 dataframe.resultados.2neu.soft <- rbind(dataframe.resultados.2neu.soft, dataframe.pasada)  
}

3 NEURONAS

SIN SOFTMAX

set.seed(1)  
  
dataframe.resultados.3neu <- data.frame(Ent\_3Neu = numeric(),  
 Test\_3Neu = numeric())  
  
for(i in 1:20)  
{  
  
 pacientes.3neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=3)  
   
 pacientes.prediccion.3neu <- predict( pacientes.3neu, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.3neu) # Vemos las probabilidades de pertenencia de cada valor  
   
   
 pacientes.prediccion.3neu.class <- apply( pacientes.prediccion.3neu, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.3neu.class  
   
   
 table( pacientes.prediccion.3neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
   
 acierto.teorico.entrenamiento.3neu <- sum( diag( table( pacientes.prediccion.3neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 # TEST  
   
 pacientes.prediccion.test.3neu <- predict( pacientes.3neu, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.3neu  
   
 pacientes.prediccion.test.3neu.class <- apply( pacientes.prediccion.test.3neu, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.3neu.class  
   
 table( pacientes.prediccion.test.3neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.3neu <- sum( diag( table( pacientes.prediccion.test.3neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_3Neu = acierto.teorico.entrenamiento.3neu,  
 Test\_3neu = acierto.teorico.test.3neu)  
   
 dataframe.resultados.3neu <- rbind(dataframe.resultados.3neu, dataframe.pasada)  
}

CON SOFTMAX

set.seed(1)  
  
dataframe.resultados.3neu.soft <- data.frame(Ent\_3Neu\_soft = numeric(),  
 Test\_3Neu\_soft = numeric())  
  
for(i in 1:20)  
{  
  
 pacientes.3neu.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=3, softmax = T)  
   
 pacientes.prediccion.3neu.softmax <- predict( pacientes.3neu.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.3neu.softmax) # Vemos las probabilidades de pertenencia de cada valor  
   
   
 pacientes.prediccion.3neu.class.softmax <- apply(pacientes.prediccion.3neu.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.3neu.class.softmax  
   
   
 table( pacientes.prediccion.3neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
   
 acierto.teorico.ent.3neu.softmax <- sum( diag( table( pacientes.prediccion.3neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 #TEST  
   
 pacientes.prediccion.test.3neu.softmax <- predict( pacientes.3neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.3neu.softmax  
   
 pacientes.prediccion.test.3neu.class.softmax <- apply( pacientes.prediccion.test.3neu.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.3neu.class.softmax  
   
 table( pacientes.prediccion.test.3neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.3neu.softmax <- sum( diag( table( pacientes.prediccion.test.3neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_3Neu\_soft = acierto.teorico.ent.3neu.softmax,  
 Test\_3neu\_soft = acierto.teorico.test.3neu.softmax)  
   
 dataframe.resultados.3neu.soft <- rbind(dataframe.resultados.3neu.soft, dataframe.pasada)  
}

3 NEURONAS

Con Decay

SIN SOFTMAX

set.seed(1)  
  
dataframe.resultados.3neu.decay <- data.frame(Ent\_3Neu\_decay = numeric(),  
 Test\_3Neu\_decay = numeric())  
  
for(i in 1:20)  
{  
  
 pacientes.3neu.decay <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=3, decay = 0.2)  
   
 pacientes.prediccion.3neu.decay <- predict( pacientes.3neu.decay, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.3neu.decay) # Vemos las probabilidades de pertenencia de cada valor  
   
   
 pacientes.prediccion.3neu.class.decay <- apply( pacientes.prediccion.3neu.decay, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.3neu.class.decay  
   
   
 table( pacientes.prediccion.3neu.class.decay, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
   
 acierto.teorico.ent.3neu.decay <- sum( diag( table( pacientes.prediccion.3neu.class.decay, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 #TEST  
   
 pacientes.prediccion.test.3neu.decay <- predict( pacientes.3neu.decay, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.3neu.decay  
   
 pacientes.prediccion.test.3neu.class.decay <- apply( pacientes.prediccion.test.3neu.decay, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.3neu.class.decay  
   
 table( pacientes.prediccion.test.3neu.class.decay , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.3neu.decay <- sum( diag( table( pacientes.prediccion.test.3neu.class.decay, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_3Neu\_decay = acierto.teorico.ent.3neu.decay,  
 Test\_3neu\_decay = acierto.teorico.test.3neu.decay)  
   
 dataframe.resultados.3neu.decay <- rbind(dataframe.resultados.3neu.decay, dataframe.pasada)  
   
}

CON SOFTMAX

set.seed(1)  
  
dataframe.resultados.3neu.decay.softmax <- data.frame(Ent\_3Neu\_decay\_sf = numeric(),  
 Test\_3Neu\_decay\_sf = numeric())  
  
for(i in 1:20)  
{  
  
 pacientes.3neu.decay.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=3, softmax = T, decay = 0.03)  
   
 pacientes.prediccion.3neu.decay.softmax <- predict( pacientes.3neu.decay.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.3neu.decay.softmax) # Vemos las probabilidades de pertenencia de cada valor  
   
   
 pacientes.prediccion.3neu.class.decay.softmax <- apply( pacientes.prediccion.3neu.decay.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.3neu.class.decay.softmax  
   
   
 table( pacientes.prediccion.3neu.class.decay.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
   
 acierto.teorico.ent.3neu.decay.sf <- sum( diag( table( pacientes.prediccion.3neu.class.decay.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 # TEST  
   
 pacientes.prediccion.test.3neu.decay.softmax <- predict( pacientes.3neu.decay.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.3neu.decay.softmax  
   
 pacientes.prediccion.test.3neu.class.decay.softmax <- apply( pacientes.prediccion.test.3neu.decay.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.3neu.class.decay.softmax  
   
 table( pacientes.prediccion.test.3neu.class.decay.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.3neu.decay.sf <- sum( diag( table( pacientes.prediccion.test.3neu.class.decay.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_3Neu\_decay\_sf = acierto.teorico.ent.3neu.decay.sf,  
 Test\_3neu\_decay\_sf = acierto.teorico.test.3neu.decay.sf)  
   
 dataframe.resultados.3neu.decay.softmax <- rbind(dataframe.resultados.3neu.decay.softmax, dataframe.pasada)  
}

5 NEURONAS

SIN SOFTMAX

set.seed(1)  
  
dataframe.resultados.5neu <- data.frame(Ent\_5Neu = numeric(),  
 Test\_5Neu = numeric())  
  
for(i in 1:20)  
{  
  
 pacientes.5neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=5 )  
   
 pacientes.prediccion.5neu <- predict( pacientes.5neu, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.5neu) # Vemos las probabilidades de pertenencia de cada valor  
   
 pacientes.prediccion.5neu.class <- apply( pacientes.prediccion.5neu, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.5neu.class  
   
 table( pacientes.prediccion.5neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
 acierto.teorico.entrenamiento.5neu <- sum( diag( table( pacientes.prediccion.5neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 #TEST  
   
 pacientes.prediccion.test.5neu <- predict( pacientes.5neu, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.5neu  
   
 pacientes.prediccion.test.5neu.class <- apply( pacientes.prediccion.test.5neu, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.5neu.class  
   
 table( pacientes.prediccion.test.5neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.5neu <- sum( diag( table( pacientes.prediccion.test.5neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_5Neu = acierto.teorico.entrenamiento.5neu,  
 Test\_5neu = acierto.teorico.test.5neu)  
   
 dataframe.resultados.5neu <- rbind(dataframe.resultados.5neu, dataframe.pasada)  
   
}

CON SOFTMAX

set.seed(1)  
  
dataframe.resultados.5neu.soft <- data.frame(Ent\_5Neu\_soft = numeric(),  
 Test\_5Neu\_soft = numeric())  
  
for(i in 1:20)  
{  
  
 pacientes.5neu.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=5, softmax = T )  
   
 pacientes.prediccion.5neu.softmax <- predict( pacientes.5neu.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.5neu.softmax) # Vemos las probabilidades de pertenencia de cada valor  
   
 pacientes.prediccion.5neu.class.softmax <- apply( pacientes.prediccion.5neu.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.5neu.class.softmax  
   
 table( pacientes.prediccion.5neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
 acierto.teorico.ent.5neu.softmax <- sum( diag( table( pacientes.prediccion.5neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 # TEST  
   
 pacientes.prediccion.test.5neu.softmax <- predict( pacientes.5neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.5neu.softmax  
   
 pacientes.prediccion.test.5neu.class.softmax <- apply( pacientes.prediccion.test.5neu.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.5neu.class.softmax  
   
 table( pacientes.prediccion.test.5neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.5neu.softmax <- sum( diag( table( pacientes.prediccion.test.5neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_5Neu\_soft = acierto.teorico.ent.5neu.softmax,  
 Test\_5neu\_soft = acierto.teorico.test.5neu.softmax)  
   
 dataframe.resultados.5neu.soft <- rbind(dataframe.resultados.5neu.soft, dataframe.pasada)  
}

5 NEURONAS

CON DECAY

SIN SOFTMAX

set.seed(1)  
  
dataframe.resultados.5neu.decay <- data.frame(Ent\_5Neu\_decay = numeric(),  
 Test\_5Neu\_decay = numeric())  
  
for(i in 1:20)  
{  
  
 pacientes.5neu.decay <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=5, decay=0.1)  
   
 pacientes.prediccion.5neu.decay <- predict( pacientes.5neu.decay, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.5neu.decay) # Vemos las probabilidades de pertenencia de cada valor  
   
 pacientes.prediccion.5neu.decay.class <- apply( pacientes.prediccion.5neu.decay, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.5neu.decay.class  
   
 table( pacientes.prediccion.5neu.decay.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
 acierto.teorico.ent.5neu.decay <- sum( diag( table( pacientes.prediccion.5neu.decay.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 # TEST  
   
 pacientes.prediccion.test.decay.5neu <- predict( pacientes.5neu.decay, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.decay.5neu  
   
 pacientes.prediccion.test.decay.5neu.class <- apply( pacientes.prediccion.test.decay.5neu, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.decay.5neu.class  
   
 table( pacientes.prediccion.test.decay.5neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.5neu.decay <- sum( diag( table( pacientes.prediccion.test.decay.5neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_5Neu\_decay = acierto.teorico.ent.5neu.decay,  
 Test\_5neu\_decay = acierto.teorico.test.5neu.decay)  
   
 dataframe.resultados.5neu.decay <- rbind(dataframe.resultados.5neu.decay, dataframe.pasada)  
   
}

CON SOFTMAX

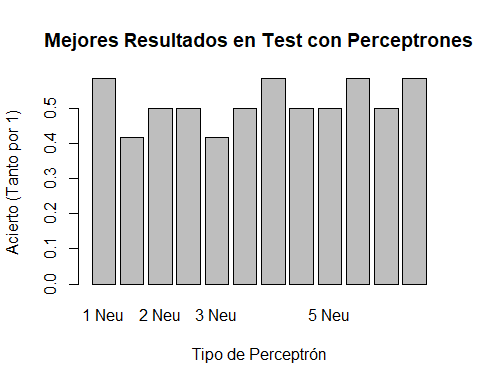
set.seed(1)  
  
dataframe.resultados.5neu.decay.softmax <- data.frame(Ent\_5Neu\_decay\_sf = numeric(),  
 Test\_5Neu\_decay\_sf = numeric())  
  
for(i in 1:20)  
{  
  
  
 pacientes.5neu.decay.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=5, softmax = T, decay = 0.05)  
   
 pacientes.prediccion.5neu.decay.softmax <- predict( pacientes.5neu.decay.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
 head(pacientes.prediccion.5neu.decay.softmax) # Vemos las probabilidades de pertenencia de cada valor  
   
 pacientes.prediccion.5neu.decay.class.softmax <- apply( pacientes.prediccion.5neu.decay.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.5neu.decay.class.softmax  
   
 table( pacientes.prediccion.5neu.decay.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.  
   
 acierto.teorico.ent.5neu.decay.sf <- sum( diag( table( pacientes.prediccion.5neu.decay.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto  
   
 # TEST  
   
 pacientes.prediccion.test.decay.5neu.softmax <- predict( pacientes.5neu.decay.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
 pacientes.prediccion.test.decay.5neu.softmax  
   
 pacientes.prediccion.test.decay.5neu.class.softmax <- apply( pacientes.prediccion.test.decay.5neu.softmax, MARGIN=1, FUN='which.is.max')  
 pacientes.prediccion.test.decay.5neu.class.softmax  
   
 table( pacientes.prediccion.test.decay.5neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )  
 acierto.teorico.test.5neu.decay.sf <- sum( diag( table( pacientes.prediccion.test.decay.5neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12  
   
   
 dataframe.pasada <- data.frame(Ent\_5Neu\_decay\_sf = acierto.teorico.ent.5neu.decay.sf,  
 Test\_5neu\_decay\_sf = acierto.teorico.test.5neu.decay.sf)  
   
 dataframe.resultados.5neu.decay.softmax <- rbind(dataframe.resultados.5neu.decay.softmax, dataframe.pasada)  
   
}

Ahora lo que hay que hacer es unir todos los resultados:

dataframe.resultados.perceptron <- cbind(dataframe.resultados.1neu,  
 dataframe.resultados.1neu.soft,  
 dataframe.resultados.2neu,  
 dataframe.resultados.2neu.soft,  
 dataframe.resultados.3neu,  
 dataframe.resultados.3neu.soft,  
 dataframe.resultados.3neu.decay,  
 dataframe.resultados.3neu.decay.softmax,  
 dataframe.resultados.5neu,  
 dataframe.resultados.5neu.soft,  
 dataframe.resultados.5neu.decay,  
 dataframe.resultados.5neu.decay.softmax)  
  
remove(dataframe.resultados.1neu)  
remove(dataframe.resultados.1neu.soft)  
remove(dataframe.resultados.2neu)  
remove(dataframe.resultados.2neu.soft)  
remove(dataframe.resultados.3neu)  
remove(dataframe.resultados.3neu.soft)  
remove(dataframe.resultados.3neu.decay)  
remove(dataframe.resultados.3neu.decay.softmax)  
remove(dataframe.resultados.5neu)  
remove(dataframe.resultados.5neu.soft)  
remove(dataframe.resultados.5neu.decay)  
remove(dataframe.resultados.5neu.decay.softmax)

Ahora visualizamos los mejores resultados de cada entrenamiento:

# Obtenemos los máximos de cada columna de test y guardamos:  
  
max.1neu <- max(dataframe.resultados.perceptron[, 2])  
max.1neu.s <- max(dataframe.resultados.perceptron[, 4])  
max.2neu <- max(dataframe.resultados.perceptron[, 6])  
max.2neu.s <- max(dataframe.resultados.perceptron[, 8])  
max.3neu <- max(dataframe.resultados.perceptron[, 10])  
max.3neu.s <- max(dataframe.resultados.perceptron[, 12])  
max.3neu.d <- max(dataframe.resultados.perceptron[, 14])  
max.3neu.d.s <- max(dataframe.resultados.perceptron[, 16])  
max.5neu <- max(dataframe.resultados.perceptron[, 18])  
max.5neu.s <- max(dataframe.resultados.perceptron[, 20])  
max.5neu.d <- max(dataframe.resultados.perceptron[, 22])  
max.5neu.d.s <- max(dataframe.resultados.perceptron[, 24])  
  
array.maximos.perceptron <- c(max.1neu,   
 max.1neu.s,   
 max.2neu,  
 max.2neu.s,  
 max.3neu,  
 max.3neu.s,  
 max.3neu.d,   
 max.3neu.d.s,   
 max.5neu,   
 max.5neu.s,   
 max.5neu.d,   
 max.5neu.d.s)  
  
barplot(array.maximos.perceptron,  
 main = "Mejores Resultados en Test con Perceptrones",  
 xlab = "Tipo de Perceptrón",  
 ylab = "Acierto (Tanto por 1)",  
 names.arg = c("1 Neu", "1 Neu Soft",   
 "2 Neu", "2 Neu Soft",   
 "3 Neu", "3 Neu Soft", "3 Neu Decay", "3 Neu Soft Decay",   
 "5 Neu", "5 Neu Soft", "5 Neu Decay", "5 Neu Soft Decay")  
 )



Exporto a PDF este barplot:

pdf("Imágenes Obtenidas/BarplotResultadosPerceptron.pdf")  
  
barplot(array.maximos.perceptron,  
 main = "Mejores Resultados en Test con Perceptrones",  
 xlab = "Tipo de Perceptrón",  
 ylab = "Acierto (Tanto por 1)",  
 names.arg = c("1 Neu", "1 Neu Soft",   
 "2 Neu", "2 Neu Soft",   
 "3 Neu", "3 Neu Soft", "3 Neu Decay", "3 Neu Soft Decay",   
 "5 Neu", "5 Neu Soft", "5 Neu Decay", "5 Neu Soft Decay")  
 )  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

# Obtención de Resultados de Perceptrón

Importo los datos:

dataset.resultados <- read.csv2("Datos/Resultados.txt")

Ahora voy a sacar un gráfico interactivo donde comparo los resultados.

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 = ~practico, name = 'Práctico') %>% layout(yaxis = list(title = 'Porcentaje'), barmode = 'group')  
  
p

#Mostramos el gráfico interactivo

Ahora que hemos sacado los resultados obtenidos con el perceptrón multicapa, vamos con otras técnicas supervisadas:

## KNN

Hacemos nuevos conjuntos:

# 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]

Si quisiéramos mostrar los conjuntos de entrenamiento y de test…

conjuntoEntrenamiento  
conjuntoTest  
etiquetasEntrenamiento  
etiquetasTest

Comenzamos las pruebas. Como sabemos, normalmente el mejor valor de K para KNN suele ser el valor que más se acerque a la raíz cuadrada del total de los valores. Por eso, empezaremos por K = 8:

### Para K = 8…

set.seed(2)  
  
prediccion.knn.8 <- knn(train = conjuntoEntrenamiento, test = conjuntoTest, cl = etiquetasEntrenamiento, k = 8)  
prediccion.knn.8

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

Sacamos crosstable:

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

set.seed(2)  
  
prediccion.knn.6 <- knn(train = conjuntoEntrenamiento, test = conjuntoTest, cl = etiquetasEntrenamiento, k = 6)  
prediccion.knn.6

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

Obtenemos la crosstable:

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 | 4 | Row Total |   
## --------------|-----------|-----------|-----------|-----------|  
## 1 | 0 | 3 | 1 | 4 |   
## | 0.000 | 0.750 | 0.250 | 0.333 |   
## | 0.000 | 0.375 | 1.000 | |   
## | 0.000 | 0.250 | 0.083 | |   
## --------------|-----------|-----------|-----------|-----------|  
## 2 | 1 | 3 | 0 | 4 |   
## | 0.250 | 0.750 | 0.000 | 0.333 |   
## | 0.333 | 0.375 | 0.000 | |   
## | 0.083 | 0.250 | 0.000 | |   
## --------------|-----------|-----------|-----------|-----------|  
## 3 | 0 | 2 | 0 | 2 |   
## | 0.000 | 1.000 | 0.000 | 0.167 |   
## | 0.000 | 0.250 | 0.000 | |   
## | 0.000 | 0.167 | 0.000 | |   
## --------------|-----------|-----------|-----------|-----------|  
## 4 | 2 | 0 | 0 | 2 |   
## | 1.000 | 0.000 | 0.000 | 0.167 |   
## | 0.667 | 0.000 | 0.000 | |   
## | 0.167 | 0.000 | 0.000 | |   
## --------------|-----------|-----------|-----------|-----------|  
## Column Total | 3 | 8 | 1 | 12 |   
## | 0.250 | 0.667 | 0.083 | |   
## --------------|-----------|-----------|-----------|-----------|  
##   
##

### Para k = 10

set.seed(2)  
  
prediccion.knn.10 <- knn(train = conjuntoEntrenamiento, test = conjuntoTest, cl = etiquetasEntrenamiento, k = 10)  
prediccion.knn.10

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

Obtenemos la crosstable:

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:

set.seed(3) #Pongo una seed para reproducibilidad

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

model <- randomForest(as.factor(dataset[, 26]) ~ ., data = dataset[, 2:25], importance = TRUE, ntree = 300)  
model

##   
## Call:  
## randomForest(formula = as.factor(dataset[, 26]) ~ ., data = dataset[, 2:25], importance = TRUE, ntree = 300)   
## Type of random forest: classification  
## Number of trees: 300  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 50.75%  
## Confusion matrix:  
## 1 2 3 4 class.error  
## 1 5 9 1 2 0.7058824  
## 2 4 26 0 0 0.1333333  
## 3 2 5 0 1 1.0000000  
## 4 7 3 0 2 0.8333333

Aquí podemos ver la matriz de confusión, de la que obtenemos también el fallo por clases.

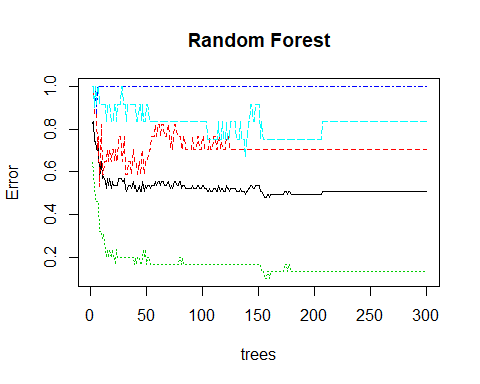
El Out-Of-Bag es un método de estimación de error que se usa en algunos algoritmos como Random Forest, y usa el modelo de Bagging para hacer muestras de submuestras usadas para el entrenamiento. El OOB es el error de predidcción medio de cada una de las muestras de entrenamiento.

Bagging es un meta-algoritmo usado para aumentar la estabilidad y precisión de algoritmos de Machine Learning de clasificación y regresión.

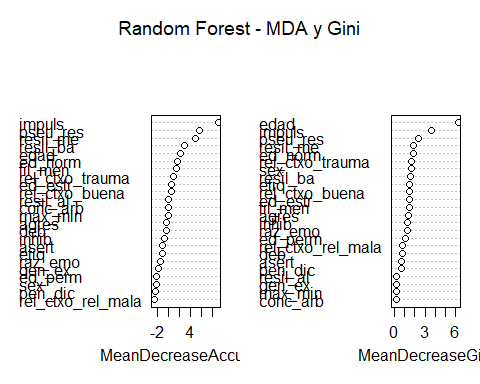
Vemos que el valor OOB (Out-Of-the-Bag) es una valor muy alto, de alrededor del 50%, por lo que nuestro modelo no está prediciendo bien.

Ahora obtenemos el número de árboles que necesitamos realmente, y la importancia de las variables en este modelo:

plot(model, main="Random Forest")



varImpPlot(model, main = "Random Forest - MDA y Gini") # Gracias a importance = true



Exporto a PDF los gráficos obtenidos:

pdf("Imágenes Obtenidas/ResultadosRandomForest.pdf")  
  
plot(model, main="Random Forest")  
varImpPlot(model, main = "Random Forest - MDA y Gini")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

Vamos a interpretar estos datos del modelo:

* MeanDecreaseAccuracy se refiere al decremento de la exactitud del modelo si se permutan los valores en cada característica. En otras palabras, MDA nos muestra la media de valores que se clasificarían mal si se quitara esa característica de la predicción. Por ello, eliminar elementos coo la impulsividad o la pseudo-resiliencia sería fatal para el modelo, mientras que la eliminación de relación con el contexto mala, pensamiento dicotómico o generalización excesiva parece ser que mejorarían el sistema.
* MeanDecreaseGini se refiere a la medida de la ganancia media de pureza mediante la división de cierta variable. Cuanto más importante sea la variable, habrá un mayor descenso en el Gini. La importancia de este está íntimamente relacionada a la función de decisión local, que Random Forest usa para seleccionar cual es la mejor separación. Debido a esto, estamos viendo que la edad, la impulsividad y la pseudo-responsabilidad son valores que tienen un gran descenso en Gini, lo cual significa que Random Forest ha determinado que eran de gran importancia.

Un elemento que me gustaría destacar es que el modelo está bastante regularizado y es bastante robusto, ya que casi todas las variables son bastante resistentes a las permutaciones, y por lo tanto no tienen un peso extremo a la hora de tomar decisiones, a excepción de la impulsividad.

Ahora, para probar a ver si hay diferencia, en vez del dataset normal voy a usar el dataset centrado y escalado, para ver si el OOB desciende:

# primero añado el grupo a la matriz, creando una nueva  
  
matriz.pacientes.datos.centscal.grupo <- cbind(matriz.pacientes.datos.centscal, dataset$grupo)  
colnames(matriz.pacientes.datos.centscal.grupo)[25] <- "grupo"

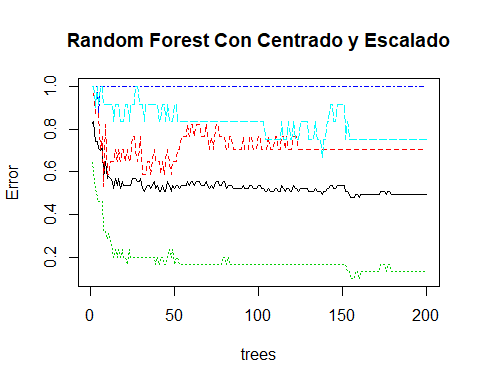
Ahora entrenamos el nuevo modelo:

set.seed(3)  
  
model\_centscal <- randomForest(as.factor(grupo) ~ ., data = matriz.pacientes.datos.centscal.grupo, importance = TRUE, ntree = 200)  
model\_centscal

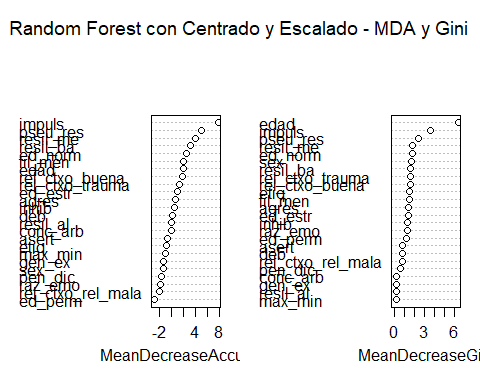
##   
## Call:  
## randomForest(formula = as.factor(grupo) ~ ., data = matriz.pacientes.datos.centscal.grupo, importance = TRUE, ntree = 200)   
## Type of random forest: classification  
## Number of trees: 200  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 49.25%  
## Confusion matrix:  
## 1 2 3 4 class.error  
## 1 5 9 1 2 0.7058824  
## 2 4 26 0 0 0.1333333  
## 3 2 5 0 1 1.0000000  
## 4 6 3 0 3 0.7500000

Podemos ver que obtenemos lo mismo. Vamos a ver si también obtenemos lo mismo en los grafos de importancia:

plot(model\_centscal, main="Random Forest Con Centrado y Escalado")



varImpPlot(model\_centscal, main="Random Forest con Centrado y Escalado - MDA y Gini") # Gracias a importance = true



Exporto a PDF estos resultados:

# Lo exporto de nuevo a PDF  
  
pdf("Imágenes Obtenidas/ResultadosRandomForestCentScal.pdf")  
  
plot(model\_centscal, main="Random Forest Con Centrado y Escalado")  
varImpPlot(model\_centscal, main="Random Forest con Centrado y Escalado - MDA y Gini")  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

En este caso, el MDA me dice que el sexo es la gran lacra de este modelo, mientras que la impulsividad sigue siendo el factor más determinante para acertar. Del Gini obtengo resultados similares.

Ahora, ya que hemos visto que cambiando el dataset no mejoramos el modelo, lo que vamos a hacer es, con el primer dataset, quitar las columnas que en el MDA tengan un valor negativo, de tal manera que la precisión del modelo debería de aumentar.

# Las columnas que son una mayor lacra son las siguientes: Relación Contexto Mala, Pensamiento dicotómico, generalización excesiva, sexo, razonamiento emocional, deberías, asertivo, relación-contexto buena, etiquetado, relación contexto trauma.  
  
dataset.randomforest <- dataset[ , c(-1, -3, -4, -5, -6, -13, -14, -15, -20, -23)]  
head(dataset.randomforest)

## edad ed\_perm ed\_norm ed\_estr resil\_ba resil\_me resil\_al fil\_men max\_min conc\_arb pseu\_res raz\_emo  
## 1 20 0 1 0 0 1 0 0 1 1 0 1  
## 2 19 0 0 1 1 0 0 0 1 1 1 0  
## 3 23 1 0 0 1 0 0 0 1 1 0 1  
## 4 16 1 0 0 1 0 0 1 1 1 0 1  
## 5 23 0 0 1 0 1 0 1 1 1 1 1  
## 6 22 0 1 0 1 0 0 0 1 1 0 1  
## inhib agres impuls grupo  
## 1 0 1 1 1  
## 2 1 0 0 2  
## 3 1 0 0 2  
## 4 0 1 1 4  
## 5 1 0 0 2  
## 6 1 0 1 4

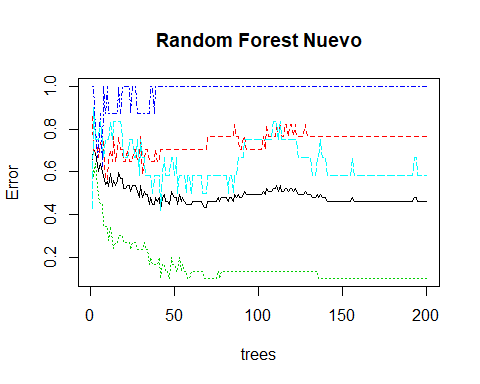
Ahora que tenemos el dataset creado, vamos a hacer la predicción de nuevo:

set.seed(3)  
  
model\_new <- randomForest(as.factor(grupo) ~ ., data = dataset.randomforest, importance = TRUE, ntree = 200)  
model\_new

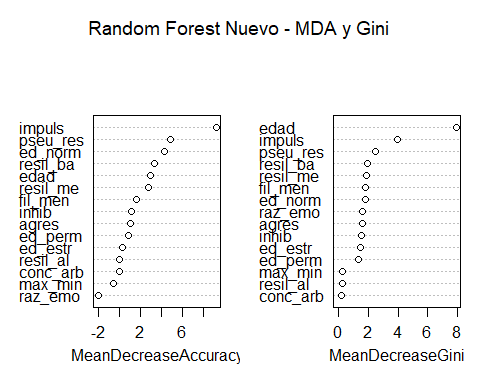
##   
## Call:  
## randomForest(formula = as.factor(grupo) ~ ., data = dataset.randomforest, importance = TRUE, ntree = 200)   
## Type of random forest: classification  
## Number of trees: 200  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 46.27%  
## Confusion matrix:  
## 1 2 3 4 class.error  
## 1 4 9 0 4 0.7647059  
## 2 2 27 0 1 0.1000000  
## 3 2 5 0 1 1.0000000  
## 4 4 3 0 5 0.5833333

Vamos a ver ahora ls gráficos…

plot(model\_new, main="Random Forest Nuevo")



varImpPlot(model\_new, main = "Random Forest Nuevo - MDA y Gini")



Exporto a PDF estos resultados del Random Forest corregido:

pdf("Imágenes Obtenidas/ResultadosRandomForestMejorado.pdf")  
  
plot(model\_new, main="Random Forest Nuevo")  
varImpPlot(model\_new, main = "Random Forest Nuevo - MDA y Gini")   
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

Como se puede observar, ahora hay nuevas variables que “lastran” el resultado del Random Forest. De todas maneras, hemos conseguido una bajada del 50% de media en OOB Error al 45% de media en OOB Error (entrenando), lo cual es una disminución menos notable de lo esperado, pero es una disminución.

Ahora lo voy a hacer con 10 fold X Validation:

set.seed(4)  
  
result <- rfcv(dataset[, 2:26], as.factor(dataset$grupo), cv.fold=10)  
head(result)

## $n.var  
## [1] 25 12 6 3 1  
##   
## $error.cv  
## 25 12 6 3 1   
## 0.1940299 0.1492537 0.1194030 0.1343284 0.0000000   
##   
## $predicted  
## $predicted$`25`  
## [1] 2 2 2 4 2 4 2 2 2 2 2 1 2 1 2 4 2 2 2 2 2 2 1 4 1 2 4 4 2 2 2 4 2 2 2 2 2 4 2 1 2 1 1 2 2 2 4 2  
## [49] 1 2 4 1 2 2 1 1 2 1 4 2 1 1 3 2 4 2 1  
## Levels: 1 2 3 4  
##   
## $predicted$`12`  
## [1] 1 2 2 4 2 4 2 1 2 2 2 1 2 1 1 4 4 2 2 2 2 2 1 4 1 2 4 4 2 2 2 4 2 2 2 2 2 4 2 1 2 1 1 2 2 2 4 2  
## [49] 1 2 4 1 2 3 1 1 2 1 4 2 1 2 2 2 4 2 1  
## Levels: 1 2 3 4  
##   
## $predicted$`6`  
## [1] 1 2 2 4 2 4 2 1 2 2 2 1 2 1 1 4 2 2 2 2 2 2 1 4 1 3 4 4 2 2 2 4 2 2 2 2 2 4 2 1 2 1 1 2 2 2 4 2  
## [49] 1 2 4 1 2 3 1 1 2 1 4 2 1 1 3 2 4 2 1  
## Levels: 1 2 3 4  
##   
## $predicted$`3`  
## [1] 1 2 2 4 2 4 2 1 2 2 2 1 2 1 1 4 4 2 2 2 2 2 1 4 1 2 4 4 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 2 2 2 4 2  
## [49] 1 2 4 1 2 2 1 1 2 1 4 2 2 1 2 2 4 2 1  
## Levels: 1 2 3 4  
##   
## $predicted$`1`  
## [1] 1 2 2 4 2 4 4 1 2 2 2 1 2 1 1 4 4 2 2 2 2 3 1 4 1 3 4 4 2 2 2 3 2 2 2 2 2 3 2 1 3 1 1 2 2 2 4 2  
## [49] 1 2 4 1 2 3 1 1 3 1 4 2 2 1 3 2 4 2 1  
## Levels: 1 2 3 4

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.

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:

set.seed(5)  
  
modelo.svm <- svm(matriz.pacientes.datos.centscal, as.factor(dataset[, 26]), kernel = "linear") # Al poner los grupos como factor, estoy consiguiendo que no sean continuos para el modelo, sino "discretos", ya que los factor no son valores que puedan ser continuos. Con esto consigo una clasificación.  
summary(modelo.svm)

##   
## Call:  
## svm.default(x = matriz.pacientes.datos.centscal, y = as.factor(dataset[, 26]), kernel = "linear")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
## gamma: 0.04166667   
##   
## Number of Support Vectors: 58  
##   
## ( 16 22 12 8 )  
##   
##   
## Number of Classes: 4   
##   
## Levels:   
## 1 2 3 4

Como vemos en el resumen, tenemos una C-Classification (necesitamos clasificar), con kernel lineal, y 58 vectores soporte.

Ahora que tenemos creado este primer modelo, toca predecir:

set.seed(5)  
  
prediccion <- predict(modelo.svm, matriz.pacientes.datos.centscal)  
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   
## 1 2 2 4 2 4 4 2 2 2 2 1 2 1 3 4 2 2 2 2 2 3 1 1 1 3 4 4 2 2 2 1 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 3 2 1 3 1 1 1 2 2 2 2 4 2 1 1 2 3 1 1 3 1 4 2 4 2 3 2 4 2   
## 67   
## 1   
## Levels: 1 2 3 4

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 13 1 1 2  
## 2 2 28 0 2  
## 3 1 0 7 0  
## 4 1 1 0 8

sum(diag(matriz.conf))/67

## [1] 0.8358209

## SVM de Kernel RBF

set.seed(5)  
  
modelo\_svm.radial <- svm(matriz.pacientes.datos.centscal, as.factor(dataset[, 26]), kernel="radial")  
summary(modelo\_svm.radial)

##   
## Call:  
## svm.default(x = matriz.pacientes.datos.centscal, y = as.factor(dataset[, 26]), kernel = "radial")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.04166667   
##   
## Number of Support Vectors: 66  
##   
## ( 17 29 12 8 )  
##   
##   
## Number of Classes: 4   
##   
## Levels:   
## 1 2 3 4

Aquí tenemos una C-Classification (necesaria para clasificar), con Kernel esta vez radial y 66 vectores soporte.

Ahora que tenemos creado este primer modelo, toca predecir:

set.seed(5)  
  
prediccion.radial <- predict(modelo\_svm.radial, matriz.pacientes.datos.centscal)  
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   
## 1 2 2 4 2 4 2 2 2 2 2 1 2 1 2 4 2 2 2 2 2 2 1 1 1 2 4 2 2 2 2 2 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 1 2 1 2 1 1 2 2 2 2 2 1 2 1 1 2 3 1 1 2 1 4 2 1 2 2 2 4 2   
## 67   
## 2   
## Levels: 1 2 3 4

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 13 1 1 2  
## 2 4 29 6 4  
## 3 0 0 1 0  
## 4 0 0 0 6

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

## [1] 0.7313433

### Como vemos, nos movemos en valores superiores al 75% de acierto

Por lo tanto, SVM 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.

set.seed(6)  
  
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") # Creamos los colores con los que queremos el cluster  
cluster.4 <- cutree(hier.clust, 4) # Cluster jerárquico de 4...

Exporto a PDF el dendrograma obtenido:

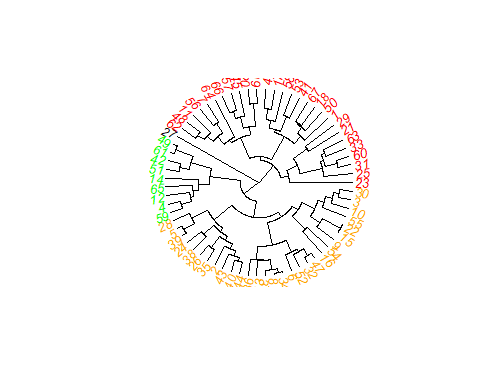
pdf("Imágenes Obtenidas/dendrograma\_pacientes.pdf")  
  
plot(as.phylo(hier.clust), type = "fan", tip.color = colores.dendrograma[cluster.4], label.offset = 0.3, cex = 0.8) #Lo pintamos  
  
dev.off()

## png   
## 2

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:

set.seed(6)  
  
dd <- dist(scale(matriz.pacientes.datos.centscal), method = "euclidean")  
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, cex = 0.8)



Lo exportamos a PDF tras la creación:

pdf("Imágenes Obtenidas/dendrograma\_pacientes\_centscal.pdf")  
  
plot(as.phylo(hier.clust), type = "fan", tip.color = colores.dendrograma[cluster.4], label.offset = 0.3, cex = 0.8)  
  
dev.off()

## png   
## 2

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

## 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:

Hacemos el clustering y vemos algunos resultados:

set.seed(7)  
  
datos.kmeans <- matriz.pacientes.datos # Sin la clasificación dentro del dataset  
  
clusters <- kmeans(datos.kmeans, centers=4)  
clusters$centers

## edad sex rel\_ctxo\_rel\_mala rel\_ctxo\_trauma rel\_ctxo\_buena ed\_perm ed\_norm  
## 1 23.82609 0.1739130 0.08695652 0.3043478 0.6086957 0.08695652 0.6956522  
## 2 16.61111 0.2777778 0.05555556 0.3888889 0.5555556 0.61111111 0.1666667  
## 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 0.2173913 0.4782609 0.52173913 0.0000000 0.9130435 0.9565217 1.0000000 0.7826087 0.9565217  
## 2 0.2222222 0.9444444 0.05555556 0.0000000 0.8333333 0.9444444 0.7777778 0.7222222 0.9444444  
## 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 1.0000000 0.6956522 1.0000000 0.9130435 0.7826087 0.08695652 0.1304348 0.5652174  
## 2 0.9444444 0.2222222 0.7777778 0.8333333 0.6666667 0.00000000 0.3333333 0.5000000  
## 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

prediccion.kmeans <- clusters$cluster  
prediccion.kmeans

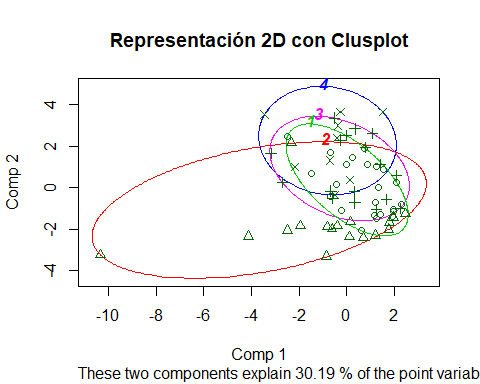
## [1] 2 2 1 2 1 1 1 1 1 2 3 2 1 2 2 2 4 1 1 1 1 2 4 1 4 2 2 1 4 2 3 1 3 1 2 1 2 3 3 3 3 1 1 2 4 4 3 1  
## [49] 3 3 3 3 4 2 2 4 3 3 2 1 1 3 3 1 3 4 1

Interpretando estos resultados, obtenemos:

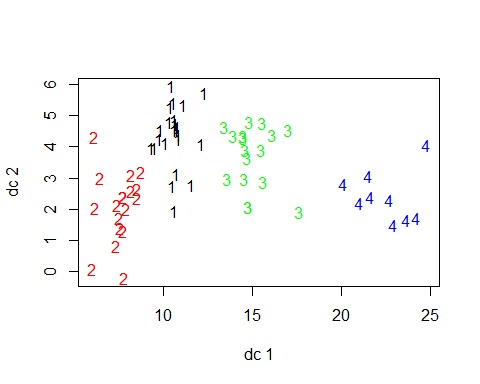
* El cluster 1 destaca por sexo más hacia masculino que otros, una relación contexto ciertamente buena, una educación permisiva, una resiliencia baja, maximización y minimización, razonamiento emocional, cierta inhibición y poca agresividad.
* El cluster 2 destaca por una edad mayor, es el cluster con 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.

Ahora sacamos la gráfica para poder ver como los ha clasificado sobre dos componentes principales artificiales:

# Representado sobre las dos componentes principales que más explicación nos dan de las variables  
  
clusplot(datos.kmeans, clusters$cluster, color = TRUE, main = "Representación 2D con Clusplot", labels = 4, xlab = "Comp 1", ylab = "Comp 2")



# Ahora la siguiente representación será con componentes discriminantes, que son las dos dimensiones sobre las que la representación de datos es más linealmente separable respecto a la predicción de grupos que ha hecho KMeans  
  
plotcluster(datos.kmeans, clusters$cluster)



Exportamos a PDF los gráficos obtenidos del KMeans:

pdf("Imágenes Obtenidas/ResultadosKmeans.pdf")  
  
clusplot(datos.kmeans, clusters$cluster, color = TRUE, main = "Representación 2D con Clusplot", labels = 4, xlab = "Comp 1", ylab = "Comp 2")   
plotcluster(datos.kmeans, clusters$cluster)  
  
dev.off

## function (which = dev.cur())   
## {  
## if (which == 1)   
## stop("cannot shut down device 1 (the null device)")  
## .External(C\_devoff, as.integer(which))  
## dev.cur()  
## }  
## <bytecode: 0x0000000015716200>  
## <environment: namespace:grDevices>

## Minería de Reglas - Rules

Para esto usaremos la librería RWeka, que es la implementación en R de Weka. Como matriz de inicio, usaremos la matriz.pacientes.datos, que está ya limpia.

### Reglas de Sexo

# Lo primero que tengo que hacer es convertir el sexo en un factor si no lo es  
  
matriz.pacientes.datos.rm = matriz.pacientes.datos  
  
matriz.pacientes.datos.rm$sex <- as.factor(matriz.pacientes.datos.rm$sex)  
  
part\_sex <- PART(sex~., data = matriz.pacientes.datos.rm)  
oneR\_sex <- OneR(sex~., data = matriz.pacientes.datos.rm)  
jrip\_sex <- JRip(sex~., data = matriz.pacientes.datos.rm)  
  
print("Reglas sexo PART")

## [1] "Reglas sexo PART"

part\_sex

## PART decision list  
## ------------------  
##   
## agres <= 0 AND  
## pen\_dic > 0 AND  
## rel\_ctxo\_trauma <= 0: 0 (34.0/4.0)  
##   
## gen\_ex > 0 AND  
## pen\_dic > 0 AND  
## ed\_estr <= 0 AND  
## edad <= 32 AND  
## impuls > 0 AND  
## raz\_emo > 0 AND  
## edad > 15: 0 (9.0/2.0)  
##   
## gen\_ex > 0 AND  
## agres <= 0 AND  
## ed\_norm <= 0: 0 (8.0)  
##   
## gen\_ex > 0 AND  
## pen\_dic > 0 AND  
## pseu\_res <= 0: 1 (6.0/1.0)  
##   
## resil\_ba <= 0: 0 (8.0/1.0)  
##   
## : 1 (2.0)  
##   
## Number of Rules : 6

print("Regla sexo OneR")

## [1] "Regla sexo OneR"

oneR\_sex

## edad:  
## < 49.5 -> 0  
## >= 49.5 -> 1  
## (54/67 instances correct)

print("Reglas sexo JRip")

## [1] "Reglas sexo JRip"

jrip\_sex

## JRIP rules:  
## ===========  
##   
## => sex=0 (67.0/14.0)  
##   
## Number of Rules : 1

Como podemos ver, del sexo no podemos sacar ninguna regla demasiado fiable, por lo que no podemos predecirlo demasiado bien.

### Reglas de impulsividad

# Lo primero que tengo que hacer es convertir el sexo en un factor si no lo es  
  
matriz.pacientes.datos.rm = matriz.pacientes.datos  
  
matriz.pacientes.datos.rm$impuls <- as.factor(matriz.pacientes.datos.rm$impuls)  
  
part\_imp <- PART(impuls~., data = matriz.pacientes.datos.rm)  
oneR\_imp <- OneR(impuls~., data = matriz.pacientes.datos.rm)  
jrip\_imp <- JRip(impuls~., data = matriz.pacientes.datos.rm)  
  
print("Reglas impulsividad PART")

## [1] "Reglas impulsividad PART"

part\_imp

## PART decision list  
## ------------------  
##   
## agres > 0: 1 (14.0)  
##   
## rel\_ctxo\_rel\_mala <= 0 AND  
## pen\_dic > 0 AND  
## ed\_perm > 0: 0 (7.0/1.0)  
##   
## pen\_dic > 0 AND  
## rel\_ctxo\_rel\_mala > 0: 1 (6.0)  
##   
## pen\_dic <= 0: 0 (5.0)  
##   
## sex > 0 AND  
## pseu\_res <= 0: 0 (3.0)  
##   
## rel\_ctxo\_trauma <= 0 AND  
## etiq > 0 AND  
## inhib > 0 AND  
## pseu\_res <= 0: 1 (8.0/1.0)  
##   
## rel\_ctxo\_trauma > 0: 1 (8.0/1.0)  
##   
## etiq > 0 AND  
## inhib > 0 AND  
## edad <= 26: 0 (7.0)  
##   
## etiq > 0: 1 (7.0/1.0)  
##   
## : 0 (2.0)  
##   
## Number of Rules : 10

print("Regla impulsividad OneR")

## [1] "Regla impulsividad OneR"

oneR\_imp

## edad:  
## < 22.5 -> 1  
## < 25.5 -> 0  
## < 46.0 -> 1  
## >= 46.0 -> 0  
## (46/67 instances correct)

print("Reglas impulsividad JRip")

## [1] "Reglas impulsividad JRip"

jrip\_imp

## JRIP rules:  
## ===========  
##   
## (agres <= 0) and (edad <= 25) => impuls=0 (29.0/10.0)  
## => impuls=1 (38.0/7.0)  
##   
## Number of Rules : 2

De la impulsividad vemos que es un poco más predecible, pero tampoco obtenemos numerosas reglas generales.

### Reglas de agresividad

# Lo primero que tengo que hacer es convertir el sexo en un factor si no lo es  
  
matriz.pacientes.datos.rm = matriz.pacientes.datos  
  
matriz.pacientes.datos.rm$agres <- as.factor(matriz.pacientes.datos.rm$agres)  
  
part\_agr <- PART(agres~., data = matriz.pacientes.datos.rm)  
oneR\_agr <- OneR(agres~., data = matriz.pacientes.datos.rm)  
jrip\_agr <- JRip(agres~., data = matriz.pacientes.datos.rm)  
  
print("Reglas agresividad PART")

## [1] "Reglas agresividad PART"

part\_agr

## PART decision list  
## ------------------  
##   
## inhib > 0: 0 (44.0)  
##   
## asert <= 0: 1 (14.0/1.0)  
##   
## : 0 (9.0/1.0)  
##   
## Number of Rules : 3

print("Regla Agresividad OneR")

## [1] "Regla Agresividad OneR"

oneR\_agr

## inhib:  
## < 0.5 -> 1  
## >= 0.5 -> 0  
## (58/67 instances correct)

print("Reglas agresividad JRip")

## [1] "Reglas agresividad JRip"

jrip\_agr

## JRIP rules:  
## ===========  
##   
## (inhib <= 0) and (asert <= 0) => agres=1 (14.0/1.0)  
## => agres=0 (53.0/1.0)  
##   
## Number of Rules : 2

### Reglas de grupo

Las más importantes:

# Lo primero que tengo que hacer es convertir el sexo en un factor si no lo es  
  
matriz.pacientes.datos.rm = matriz.pacientes.datos  
matriz.pacientes.datos.rm <- cbind(matriz.pacientes.datos.rm, dataset$grupo)  
  
matriz.pacientes.datos.rm$`dataset$grupo` <- as.factor(matriz.pacientes.datos.rm$`dataset$grupo`)  
  
part\_gru <- PART(`dataset$grupo`~., data = matriz.pacientes.datos.rm)  
oneR\_gru <- OneR(`dataset$grupo`~., data = matriz.pacientes.datos.rm)  
jrip\_gru <- JRip(`dataset$grupo`~., data = matriz.pacientes.datos.rm)  
  
print("Reglas grupo PART")

## [1] "Reglas grupo PART"

part\_gru

## PART decision list  
## ------------------  
##   
## impuls <= 0 AND  
## fil\_men > 0 AND  
## pen\_dic > 0 AND  
## resil\_me <= 0 AND  
## rel\_ctxo\_trauma <= 0: 2 (9.0/3.0)  
##   
## impuls <= 0 AND  
## fil\_men <= 0: 2 (6.0)  
##   
## impuls <= 0 AND  
## resil\_ba <= 0 AND  
## pseu\_res <= 0: 2 (4.0)  
##   
## impuls <= 0 AND  
## ed\_estr <= 0: 3 (5.0/1.0)  
##   
## impuls > 0 AND  
## deb > 0 AND  
## resil\_ba <= 0 AND  
## ed\_estr <= 0 AND  
## sex <= 0 AND  
## ed\_perm <= 0 AND  
## pseu\_res > 0: 2 (6.0)  
##   
## impuls > 0 AND  
## gen\_ex > 0 AND  
## etiq > 0 AND  
## fil\_men > 0 AND  
## rel\_ctxo\_rel\_mala <= 0 AND  
## pseu\_res > 0 AND  
## rel\_ctxo\_trauma <= 0: 1 (5.0/1.0)  
##   
## impuls > 0 AND  
## gen\_ex > 0 AND  
## etiq > 0 AND  
## fil\_men <= 0: 4 (3.0)  
##   
## impuls > 0 AND  
## fil\_men <= 0: 1 (3.0/1.0)  
##   
## impuls <= 0: 2 (2.0)  
##   
## rel\_ctxo\_rel\_mala > 0 AND  
## edad <= 37: 1 (2.0)  
##   
## rel\_ctxo\_rel\_mala <= 0 AND  
## max\_min > 0 AND  
## raz\_emo > 0 AND  
## inhib > 0: 4 (6.0/3.0)  
##   
## rel\_ctxo\_rel\_mala <= 0 AND  
## rel\_ctxo\_trauma > 0 AND  
## ed\_norm <= 0: 4 (7.0/2.0)  
##   
## rel\_ctxo\_rel\_mala <= 0 AND  
## raz\_emo > 0: 2 (4.0/1.0)  
##   
## rel\_ctxo\_rel\_mala <= 0: 1 (3.0/1.0)  
##   
## : 2 (2.0)  
##   
## Number of Rules : 15

print("Regla grupo OneR")

## [1] "Regla grupo OneR"

oneR\_gru

## agres:  
## < 0.5 -> 2  
## >= 0.5 -> 1  
## (32/67 instances correct)

print("Reglas grupo JRip")

## [1] "Reglas grupo JRip"

jrip\_gru

## JRIP rules:  
## ===========  
##   
## (edad >= 37) and (pseu\_res >= 1) => dataset$grupo=1 (4.0/0.0)  
## => dataset$grupo=2 (63.0/33.0)  
##   
## Number of Rules : 2