TFM - base de datosMD

Federico José Saviotti Brunacci

7/10/2021

# Librerías

# Visualizar base de datos y preprocesado

getwd()

[1] "C:/Users/Federico/Desktop/MASTER/TFM/datos"

setwd("C:/Users/Federico/Desktop/MASTER/TFM/datos/")

Lectura de la base de datos

BD <- read.csv("Indian Liver Patient Dataset (ILPD).csv", header = FALSE)  
colnames(BD) <- c("Edad", "Género", "TBil", "DBil", "Alkphos", "Sgpt", "Sgot", "TP", "Albúmina", "Ratio", "Condicion")  
  
summary(BD)

Edad Género TBil DBil   
 Min. : 4.00 Length:583 Min. : 0.400 Min. : 0.100   
 1st Qu.:33.00 Class :character 1st Qu.: 0.800 1st Qu.: 0.200   
 Median :45.00 Mode :character Median : 1.000 Median : 0.300   
 Mean :44.75 Mean : 3.299 Mean : 1.486   
 3rd Qu.:58.00 3rd Qu.: 2.600 3rd Qu.: 1.300   
 Max. :90.00 Max. :75.000 Max. :19.700   
   
 Alkphos Sgpt Sgot TP   
 Min. : 63.0 Min. : 10.00 Min. : 10.0 Min. :2.700   
 1st Qu.: 175.5 1st Qu.: 23.00 1st Qu.: 25.0 1st Qu.:5.800   
 Median : 208.0 Median : 35.00 Median : 42.0 Median :6.600   
 Mean : 290.6 Mean : 80.71 Mean : 109.9 Mean :6.483   
 3rd Qu.: 298.0 3rd Qu.: 60.50 3rd Qu.: 87.0 3rd Qu.:7.200   
 Max. :2110.0 Max. :2000.00 Max. :4929.0 Max. :9.600   
   
 Albúmina Ratio Condicion   
 Min. :0.900 Min. :0.3000 Min. :1.000   
 1st Qu.:2.600 1st Qu.:0.7000 1st Qu.:1.000   
 Median :3.100 Median :0.9300 Median :1.000   
 Mean :3.142 Mean :0.9471 Mean :1.286   
 3rd Qu.:3.800 3rd Qu.:1.1000 3rd Qu.:2.000   
 Max. :5.500 Max. :2.8000 Max. :2.000   
 NA's :4

## Descripción del conjunto de datos

str(BD)

'data.frame': 583 obs. of 11 variables:  
 $ Edad : int 65 62 62 58 72 46 26 29 17 55 ...  
 $ Género : chr "Female" "Male" "Male" "Male" ...  
 $ TBil : num 0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7 ...  
 $ DBil : num 0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2 ...  
 $ Alkphos : int 187 699 490 182 195 208 154 202 202 290 ...  
 $ Sgpt : int 16 64 60 14 27 19 16 14 22 53 ...  
 $ Sgot : int 18 100 68 20 59 14 12 11 19 58 ...  
 $ TP : num 6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 ...  
 $ Albúmina : num 3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.4 ...  
 $ Ratio : num 0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 ...  
 $ Condicion: int 1 1 1 1 1 1 1 1 2 1 ...

summary(BD)

Edad Género TBil DBil   
 Min. : 4.00 Length:583 Min. : 0.400 Min. : 0.100   
 1st Qu.:33.00 Class :character 1st Qu.: 0.800 1st Qu.: 0.200   
 Median :45.00 Mode :character Median : 1.000 Median : 0.300   
 Mean :44.75 Mean : 3.299 Mean : 1.486   
 3rd Qu.:58.00 3rd Qu.: 2.600 3rd Qu.: 1.300   
 Max. :90.00 Max. :75.000 Max. :19.700   
   
 Alkphos Sgpt Sgot TP   
 Min. : 63.0 Min. : 10.00 Min. : 10.0 Min. :2.700   
 1st Qu.: 175.5 1st Qu.: 23.00 1st Qu.: 25.0 1st Qu.:5.800   
 Median : 208.0 Median : 35.00 Median : 42.0 Median :6.600   
 Mean : 290.6 Mean : 80.71 Mean : 109.9 Mean :6.483   
 3rd Qu.: 298.0 3rd Qu.: 60.50 3rd Qu.: 87.0 3rd Qu.:7.200   
 Max. :2110.0 Max. :2000.00 Max. :4929.0 Max. :9.600   
   
 Albúmina Ratio Condicion   
 Min. :0.900 Min. :0.3000 Min. :1.000   
 1st Qu.:2.600 1st Qu.:0.7000 1st Qu.:1.000   
 Median :3.100 Median :0.9300 Median :1.000   
 Mean :3.142 Mean :0.9471 Mean :1.286   
 3rd Qu.:3.800 3rd Qu.:1.1000 3rd Qu.:2.000   
 Max. :5.500 Max. :2.8000 Max. :2.000   
 NA's :4

### Reconversión de variables

BD$Condicion <- as.factor(BD$Condicion)  
BD$Género <- as.factor(BD$Género)  
  
summary(BD)

Edad Género TBil DBil   
 Min. : 4.00 Female:142 Min. : 0.400 Min. : 0.100   
 1st Qu.:33.00 Male :441 1st Qu.: 0.800 1st Qu.: 0.200   
 Median :45.00 Median : 1.000 Median : 0.300   
 Mean :44.75 Mean : 3.299 Mean : 1.486   
 3rd Qu.:58.00 3rd Qu.: 2.600 3rd Qu.: 1.300   
 Max. :90.00 Max. :75.000 Max. :19.700   
   
 Alkphos Sgpt Sgot TP   
 Min. : 63.0 Min. : 10.00 Min. : 10.0 Min. :2.700   
 1st Qu.: 175.5 1st Qu.: 23.00 1st Qu.: 25.0 1st Qu.:5.800   
 Median : 208.0 Median : 35.00 Median : 42.0 Median :6.600   
 Mean : 290.6 Mean : 80.71 Mean : 109.9 Mean :6.483   
 3rd Qu.: 298.0 3rd Qu.: 60.50 3rd Qu.: 87.0 3rd Qu.:7.200   
 Max. :2110.0 Max. :2000.00 Max. :4929.0 Max. :9.600   
   
 Albúmina Ratio Condicion  
 Min. :0.900 Min. :0.3000 1:416   
 1st Qu.:2.600 1st Qu.:0.7000 2:167   
 Median :3.100 Median :0.9300   
 Mean :3.142 Mean :0.9471   
 3rd Qu.:3.800 3rd Qu.:1.1000   
 Max. :5.500 Max. :2.8000   
 NA's :4

#El ratio tiene 4 valores vacíos, por tanto estas observaciones deberían eliminarse.  
#Lo mismo ocurre con los valores repetidos  
  
BD.0 <- na.omit(BD)  
BD.0 <- BD.0[!duplicated(BD.0), ]

La base de datos posee:

* 566 pacientes (observaciones)
* 404 con hígado enfermo y 162 con hígado sano
* 11 variables
* Las Variables:

Variables numéricas: *Edad*: Edad en años *TBil*: Bilirrubina total (mg/dL) *DBil*: Bilirrubina directa (mg/dL) *Alkphos*: Fosfatasa alcalina (U/L) *Sgpt*: Aminotransferasa alanina *Sgot*: Aminotransferasa aspartato *TP*: proteínas totales *Albúmina*. *Ratio*: Ratio albúmina y Ratio globulina

Variables factoriales: *Género*: Female - Mujer, Male - Hombre, variable factorial. *Condicion*: 1 - pacientes con problemas hepáticos, 2 - pacientes sin problemas hepáticos, variable factorial.

* Variable respuesta: Condicion
* Variables predictoras: Edad, género, TBil, DBil, Alkphos, Sgpt, Sgot, TP, Albúmina y Ratio.

## Análisis exploratorio

table(BD.0$Condicion)

1 2   
404 162

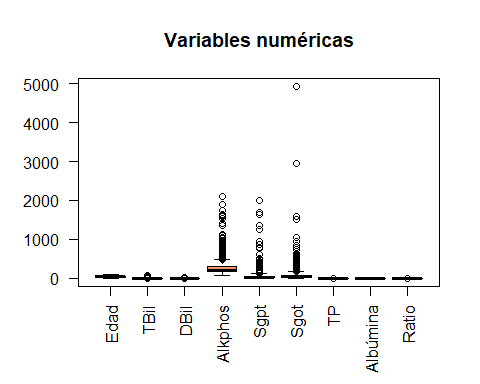
* *Los datos no están balanceados respecto a la condición del paciente*

Rangos de variables:

summary(BD.0)

Edad Género TBil DBil   
 Min. : 4.00 Female:138 Min. : 0.400 Min. : 0.100   
 1st Qu.:33.00 Male :428 1st Qu.: 0.800 1st Qu.: 0.200   
 Median :45.00 Median : 1.000 Median : 0.300   
 Mean :44.89 Mean : 3.339 Mean : 1.506   
 3rd Qu.:58.00 3rd Qu.: 2.600 3rd Qu.: 1.300   
 Max. :90.00 Max. :75.000 Max. :19.700   
 Alkphos Sgpt Sgot TP   
 Min. : 63.0 Min. : 10.00 Min. : 10.0 Min. :2.700   
 1st Qu.: 176.0 1st Qu.: 23.00 1st Qu.: 25.0 1st Qu.:5.800   
 Median : 208.0 Median : 35.00 Median : 41.0 Median :6.600   
 Mean : 292.6 Mean : 80.14 Mean : 109.9 Mean :6.495   
 3rd Qu.: 298.0 3rd Qu.: 60.75 3rd Qu.: 87.0 3rd Qu.:7.200   
 Max. :2110.0 Max. :2000.00 Max. :4929.0 Max. :9.600   
 Albúmina Ratio Condicion  
 Min. :0.900 Min. :0.300 1:404   
 1st Qu.:2.600 1st Qu.:0.700 2:162   
 Median :3.100 Median :0.950   
 Mean :3.146 Mean :0.948   
 3rd Qu.:3.800 3rd Qu.:1.100   
 Max. :5.500 Max. :2.800

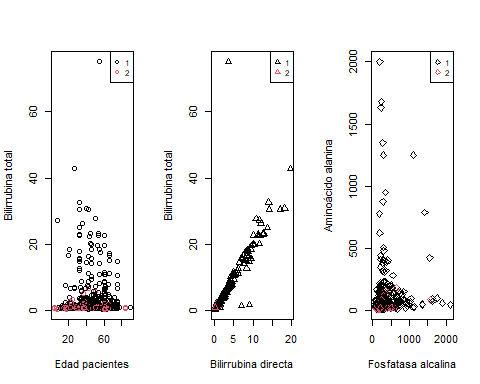
boxplot(BD.0[-c(2,11)], las=2, col="lightsalmon2", main="Variables numéricas")



* El rango de las variables numéricas difiere enormemente en función del parámetro observado. También se observa que es debido a valores atípicos (outliers), por tanto, se puede aplicar normalización sobre los datos.

### Visualización de la variable respuesta

#Se ven los datos no balanceados y solapados  
  
par(mfrow=c(1,3))  
plot(BD.0$Edad, BD.0$TBil, col=BD.0$Condicion,   
 xlab= "Edad pacientes", ylab= "Bilirrubina total")  
legend('topright', legend = levels(BD.0$Condicion), col = 1:2, cex = 0.8, pch = 1)  
  
plot(BD.0$DBil, BD.0$TBil, col=BD.0$Condicion,   
 xlab= "Bilirrubina directa", ylab= "Bilirrubina total",  
 pch = 2)  
legend('topright', legend = levels(BD.0$Condicion), col = 1:2, cex = 0.8, pch = 2)  
  
plot(BD.0$Alkphos, BD.0$Sgpt, col=BD.0$Condicion,   
 xlab= "Fosfatasa alcalina", ylab= "Aminoácido alanina",  
 pch = 5)  
legend('topright', legend = levels(BD.0$Condicion), col = 1:2, cex = 0.8, pch = 5)



par(mfrow=c(1,1))

Los gráficos muestran la comparación entre diversas variables numéricas del modelo y el color de sus valores denota la clase a la que pertenece cada observación.

Claramente hay un desbalanceo entre las 2 clases y un solapamiento entre las observaciones que dificultarán la clasificación, por tanto habrá que tratar de arreglar la base de datos para que los algoritmos den una predicción mejor.

#### Submuestreo aleatorio

Mediante el uso de la función downSample se puede aplicar un submuestreo que iguala el número de observaciones de la clase mayoritaria al de la minoritaria eliminando observaciones de manera aleatoria. Se ha aplicado una semilla concreta para evitar favorecer la reproducibilidad.

set.seed(123)  
BD\_DS <- downSample(BD.0[,-c(11)], BD.0$Condicion, yname = "Condicion")  
str(BD\_DS)

'data.frame': 324 obs. of 11 variables:  
 $ Edad : int 50 38 45 48 60 42 13 26 52 50 ...  
 $ Género : Factor w/ 2 levels "Female","Male": 2 2 2 1 2 1 1 1 2 1 ...  
 $ TBil : num 0.9 1.8 2.2 0.8 8.9 0.8 0.7 0.7 0.8 27.7 ...  
 $ DBil : num 0.3 0.8 0.8 0.2 4 0.2 0.2 0.2 0.2 10.8 ...  
 $ Alkphos : int 901 342 209 142 950 168 350 144 245 380 ...  
 $ Sgpt : int 23 168 25 26 33 25 17 36 48 39 ...  
 $ Sgot : int 17 441 20 25 32 18 24 33 49 348 ...  
 $ TP : num 6.2 7.6 8 6 6.8 6.2 7.4 8.2 6.4 7.1 ...  
 $ Albúmina : num 3.5 4.4 4 2.6 3.1 3.1 4 4.3 3.2 2.3 ...  
 $ Ratio : num 1.2 1.3 1 0.7 0.8 1 1.1 1.1 1 0.4 ...  
 $ Condicion: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...

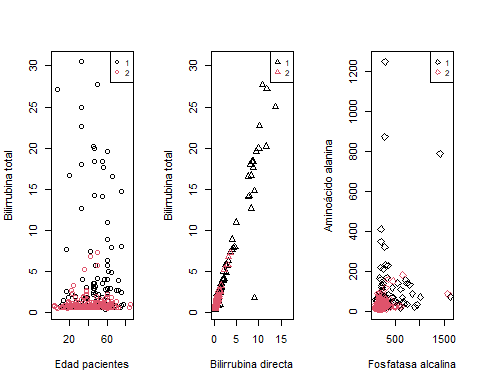
table(BD\_DS$Condicion)

1 2   
162 162

Se puede ver que ahora hay el mismo número de observaciones de la clase 1 y de la 2.

* *Representación gráfica de esta nueva base de datos*

par(mfrow=c(1,3))  
plot(BD\_DS$Edad, BD\_DS$TBil, col=BD\_DS$Condicion,   
 xlab= "Edad pacientes", ylab= "Bilirrubina total")  
legend('topright', legend = levels(BD\_DS$Condicion), col = 1:2, cex = 0.8, pch = 1)  
  
plot(BD\_DS$DBil, BD\_DS$TBil, col=BD\_DS$Condicion,   
 xlab= "Bilirrubina directa", ylab= "Bilirrubina total",  
 pch = 2)  
legend('topright', legend = levels(BD\_DS$Condicion), col = 1:2, cex = 0.8, pch = 2)  
  
plot(BD\_DS$Alkphos, BD\_DS$Sgpt, col=BD\_DS$Condicion,   
 xlab= "Fosfatasa alcalina", ylab= "Aminoácido alanina",  
 pch = 5)  
legend('topright', legend = levels(BD\_DS$Condicion), col = 1:2, cex = 0.8, pch = 5)



par(mfrow=c(1,1))

Como se puede ver, ahora hay un menor número de observaciones de la clase 1.

#### Submuestreo con Edited Nearest Neighbor (ENN)

Este método, a diferencia del anterior, no iguala las observaciones, sino que utiliza el algoritmo de K vecinos cercanos (KNN) para eliminar valores de la clase mayoritaria que se encuentren muy cerca de la clase minoritaria. El paquete empleado es el de UBL y la función ENNClassif.

set.seed(123)  
ENN\_DAT <- ENNClassif(Condicion ~. , dat= BD.0, k= 5, dist= "HEOM", p= 2, Cl= 1)  
  
BD\_ENN <- ENN\_DAT[[1]]  
  
str(BD\_ENN)

'data.frame': 480 obs. of 11 variables:  
 $ Edad : int 62 62 58 72 46 26 29 17 55 57 ...  
 $ Género : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 1 1 2 2 2 ...  
 $ TBil : num 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7 0.6 ...  
 $ DBil : num 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2 0.1 ...  
 $ Alkphos : int 699 490 182 195 208 154 202 202 290 210 ...  
 $ Sgpt : int 64 60 14 27 19 16 14 22 53 51 ...  
 $ Sgot : int 100 68 20 59 14 12 11 19 58 59 ...  
 $ TP : num 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 5.9 ...  
 $ Albúmina : num 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.4 2.7 ...  
 $ Ratio : num 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 0.8 ...  
 $ Condicion: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 2 1 1 ...  
 - attr(\*, "na.action")= 'omit' Named int [1:4] 210 242 254 313  
 ..- attr(\*, "names")= chr [1:4] "210" "242" "254" "313"

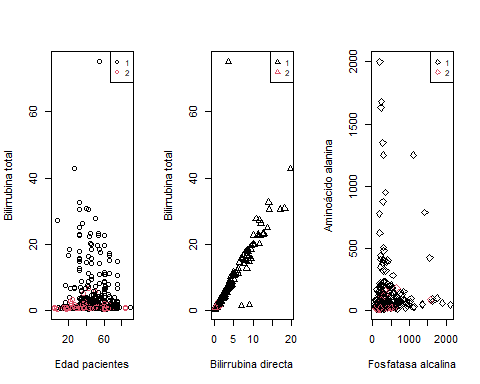
table(BD\_ENN$Condicion)

1 2   
318 162

En este caso concreto se aplica para los 5 vecinos más cercanos utilizando una distancia que permite trabajar con variables tanto numéricas como factoriales.

Las clases no quedan balanceadas, sin embargo como se puede vislumbrar en los gráficos hay un clareamiento en los datos.

par(mfrow=c(1,3))  
plot(BD\_ENN$Edad, BD\_ENN$TBil, col=BD\_ENN$Condicion,   
 xlab= "Edad pacientes", ylab= "Bilirrubina total")  
legend('topright', legend = levels(BD\_ENN$Condicion), col = 1:2, cex = 0.8, pch = 1)  
  
plot(BD\_ENN$DBil, BD\_ENN$TBil, col=BD\_ENN$Condicion,   
 xlab= "Bilirrubina directa", ylab= "Bilirrubina total",  
 pch = 2)  
legend('topright', legend = levels(BD\_ENN$Condicion), col = 1:2, cex = 0.8, pch = 2)  
  
plot(BD\_ENN$Alkphos, BD\_ENN$Sgpt, col=BD\_ENN$Condicion,   
 xlab= "Fosfatasa alcalina", ylab= "Aminoácido alanina",  
 pch = 5)  
legend('topright', legend = levels(BD\_ENN$Condicion), col = 1:2, cex = 0.8, pch = 5)



par(mfrow=c(1,1))

#### ENN y submuestreo aleatorio

Se aplica la primera técnica a la ya pulida en la 2da para ver si aumenta su poder predictor.

set.seed(123)  
BD\_ENN\_DS <- downSample(BD\_ENN[,-c(11)], BD\_ENN$Condicion, yname = "Condicion")  
  
str(BD\_ENN\_DS)

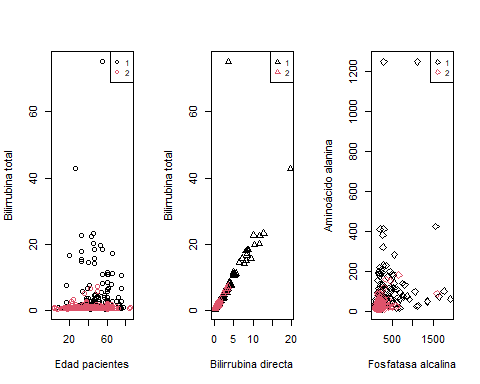
'data.frame': 324 obs. of 11 variables:  
 $ Edad : int 75 51 48 52 40 51 73 25 40 40 ...  
 $ Género : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...  
 $ TBil : num 6.7 2.2 0.7 2.7 3.6 4 1.9 0.8 0.6 14.5 ...  
 $ DBil : num 3.6 1 0.2 1.4 1.8 2.5 0.7 0.1 0.1 6.4 ...  
 $ Alkphos : int 458 610 326 251 285 275 1750 130 98 358 ...  
 $ Sgpt : int 198 17 29 20 50 382 102 23 35 50 ...  
 $ Sgot : int 143 28 17 40 60 330 141 42 31 75 ...  
 $ TP : num 6.2 7.3 8.7 6 7 7.5 5.5 8 6 5.7 ...  
 $ Albúmina : num 3.2 2.6 5.5 1.7 2.9 4 2 4 3.2 2.1 ...  
 $ Ratio : num 1 0.55 1.7 0.39 0.7 1.1 0.5 1 1.1 0.5 ...  
 $ Condicion: Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...

table(BD\_ENN\_DS$Condicion)

1 2   
162 162

Vuelve a igualarse el número de observaciones

par(mfrow=c(1,3))  
plot(BD\_ENN\_DS$Edad, BD\_ENN\_DS$TBil, col=BD\_ENN\_DS$Condicion,   
 xlab= "Edad pacientes", ylab= "Bilirrubina total")  
legend('topright', legend = levels(BD\_ENN\_DS$Condicion), col = 1:2, cex = 0.8, pch = 1)  
  
plot(BD\_ENN\_DS$DBil, BD\_ENN\_DS$TBil, col=BD\_ENN\_DS$Condicion,   
 xlab= "Bilirrubina directa", ylab= "Bilirrubina total",  
 pch = 2)  
legend('topright', legend = levels(BD\_ENN\_DS$Condicion), col = 1:2, cex = 0.8, pch = 2)  
  
plot(BD\_ENN\_DS$Alkphos, BD\_ENN\_DS$Sgpt, col=BD\_ENN\_DS$Condicion,   
 xlab= "Fosfatasa alcalina", ylab= "Aminoácido alanina",  
 pch = 5)  
legend('topright', legend = levels(BD\_ENN\_DS$Condicion), col = 1:2, cex = 0.8, pch = 5)



par(mfrow=c(1,1))

Visualmente también se vuelven a ver las observaciones con mayor claridad.

# Algoritmos

## Separación de base de datos - Entrenamiento y prueba

Se crea un grupo de entrenamiento y otro de evaluación para cada una de las bases de datos.

set.seed(123)  
random\_ids <- order(runif(nrow(BD\_DS)))  
  
#DOWNSAMPLE ALEATORIO  
BD\_Ptrain1 <- BD\_DS[random\_ids[1:round(length(random\_ids)\*0.67)],]  
BD\_Ptest1 <- BD\_DS[-random\_ids[1:round(length(random\_ids)\*0.67)],]  
  
  
#ENN UNDERSAMPLE  
set.seed(123)  
random\_ids2 <- order(runif(nrow(BD\_ENN)))  
  
BD\_Ptrain2 <- BD\_ENN[random\_ids2[1:round(length(random\_ids2)\*0.67)],]  
BD\_Ptest2 <- BD\_ENN[-random\_ids2[1:round(length(random\_ids2)\*0.67)],]  
  
  
#ENN + DOWNSAMPLE ALEATORIO  
BD\_Ptrain3 <- BD\_ENN\_DS[random\_ids[1:round(length(random\_ids)\*0.67)],]  
BD\_Ptest3 <- BD\_ENN\_DS[-random\_ids[1:round(length(random\_ids)\*0.67)],]

El paquete “caret” posee herramientas para el preprocesado.

Se ejecutará cada algoritmo cambiando los valores de preprocesado, teniendo por un lado el modo de escalado y centralización, por otro el de transformar en rango contenido entre 0 y 1 las variables numéricas.

El método de preprocesado incluye “center” y “scale” en los cuales se sustraen las medias de los datos predictores y se divide por su desviación estándar. El método “range” transforma los valores numéricos aplicando el valor mínimo y máximo de cada varible.

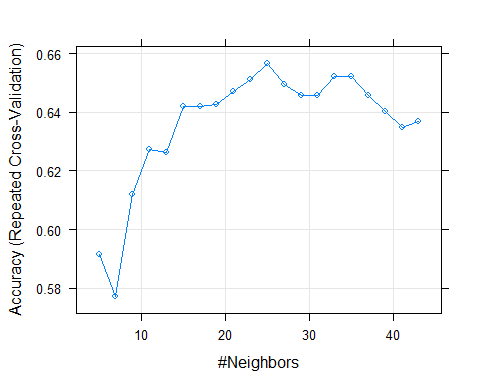
## Knn - entrenamiento

Se aplica 10-fold crossvalidation, que hace el remuestreo de los datos para su correcto entrenamiento.

ctrl <- trainControl(method="repeatedcv",number=10,repeats = 5)  
  
#KNN  
  
 #Downsample  
set.seed(1234567)  
knn\_1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "knn",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
knn\_1

k-Nearest Neighbors   
  
217 samples  
 10 predictor  
 2 classes: '1', '2'   
  
No pre-processing  
Resampling: Cross-Validated (10 fold, repeated 5 times)   
Summary of sample sizes: 195, 196, 195, 196, 195, 195, ...   
Resampling results across tuning parameters:  
  
 k Accuracy Kappa   
 5 0.5915584 0.1840333  
 7 0.5770130 0.1553025  
 9 0.6119481 0.2261121  
 11 0.6274459 0.2578975  
 13 0.6262338 0.2548470  
 15 0.6419481 0.2859747  
 17 0.6418615 0.2860730  
 19 0.6427706 0.2880489  
 21 0.6472727 0.2968181  
 23 0.6512121 0.3043656  
 25 0.6568398 0.3154149  
 27 0.6494372 0.3012372  
 29 0.6458874 0.2942339  
 31 0.6458009 0.2941373  
 33 0.6522078 0.3070006  
 35 0.6522944 0.3072640  
 37 0.6459307 0.2947064  
 39 0.6403463 0.2835848  
 41 0.6348485 0.2728958  
 43 0.6367965 0.2767997  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 25.

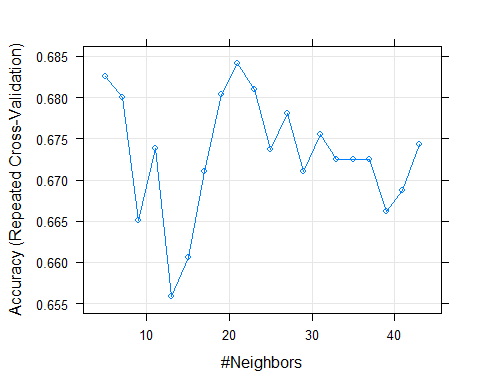
plot(knn\_1)



# ENN  
set.seed(1234567)  
knn\_2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "knn",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
knn\_2

k-Nearest Neighbors   
  
322 samples  
 10 predictor  
 2 classes: '1', '2'   
  
No pre-processing  
Resampling: Cross-Validated (10 fold, repeated 5 times)   
Summary of sample sizes: 290, 290, 290, 290, 290, 290, ...   
Resampling results across tuning parameters:  
  
 k Accuracy Kappa   
 5 0.6825910 0.2806716  
 7 0.6800721 0.2791239  
 9 0.6651442 0.2400612  
 11 0.6738032 0.2708349  
 13 0.6558071 0.2358602  
 15 0.6606354 0.2477561  
 17 0.6710924 0.2754947  
 19 0.6803311 0.2938416  
 21 0.6841984 0.3018588  
 23 0.6810343 0.2950725  
 25 0.6736883 0.2790702  
 27 0.6780254 0.2915979  
 29 0.6710887 0.2745771  
 31 0.6755645 0.2874796  
 33 0.6724762 0.2802297  
 35 0.6724774 0.2762230  
 37 0.6724963 0.2759704  
 39 0.6662262 0.2671549  
 41 0.6687072 0.2695663  
 43 0.6742742 0.2810394  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 21.

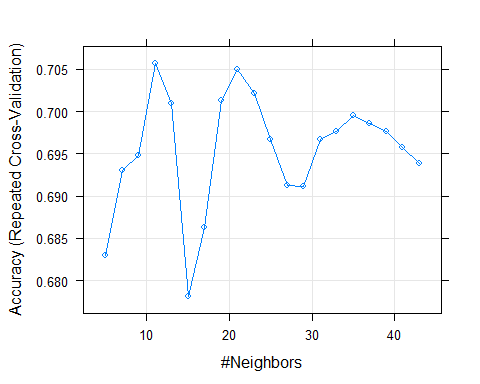
plot(knn\_2)



# ENN + downsample  
set.seed(1234567)  
knn\_3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "knn",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
knn\_3

k-Nearest Neighbors   
  
217 samples  
 10 predictor  
 2 classes: '1', '2'   
  
No pre-processing  
Resampling: Cross-Validated (10 fold, repeated 5 times)   
Summary of sample sizes: 195, 196, 195, 196, 195, 195, ...   
Resampling results across tuning parameters:  
  
 k Accuracy Kappa   
 5 0.6829870 0.3675847  
 7 0.6929870 0.3878787  
 9 0.6947619 0.3919659  
 11 0.7057576 0.4139845  
 13 0.7009524 0.4049851  
 15 0.6780519 0.3593059  
 17 0.6862771 0.3759381  
 19 0.7012987 0.4060335  
 21 0.7049351 0.4130379  
 23 0.7021645 0.4074845  
 25 0.6966667 0.3963425  
 27 0.6912121 0.3854179  
 29 0.6911688 0.3855883  
 31 0.6967100 0.3967265  
 33 0.6976623 0.3985271  
 35 0.6995238 0.4022615  
 37 0.6985714 0.4003453  
 39 0.6976623 0.3985271  
 41 0.6957576 0.3949104  
 43 0.6938961 0.3912916  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 11.

plot(knn\_3)



Para cada modelo se van eligiendo los valores que presentan mayor precisión a la hora de entrenarlo.

## Knn - evaluación

Se dará la evaluación con valores reales desconocidos de los que se sabe su verdadera clasificación, las bases de datos de prueba.

knn\_1pred <- predict(knn\_1, newdata = BD\_Ptest1 )  
CM\_knn1 <- confusionMatrix(knn\_1pred, BD\_Ptest1$Condicion)  
  
knn\_2pred <- predict(knn\_2, newdata = BD\_Ptest2 )  
CM\_knn2 <- confusionMatrix(knn\_2pred, BD\_Ptest2$Condicion)  
  
knn\_3pred <- predict(knn\_3, newdata = BD\_Ptest3 )  
CM\_knn3 <- confusionMatrix(knn\_3pred, BD\_Ptest3$Condicion)

Se almacenan los datos en las distintas matrices de confusión empleando la función confusionMatrix que tiene tanto la precisión real del modelo, como su sensibilidad y especificidad.

Para almacenar los valores de interés para los modelos procesados, hay que eliminar el icono de la almohadilla #, de tal forma se aplicará el preprocesado correspondiente en los datos del algoritmo.

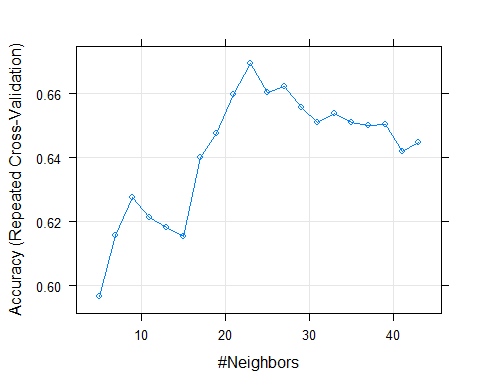
#Sin preprocesar  
precNO <- c(CM\_knn1$overall["Accuracy"], CM\_knn2$overall["Accuracy"], CM\_knn3$overall["Accuracy"])  
senNO <- c(CM\_knn1$byClass["Sensitivity"], CM\_knn2$byClass["Sensitivity"], CM\_knn3$byClass["Sensitivity"])  
speNO <- c(CM\_knn1$byClass["Specificity"], CM\_knn2$byClass["Specificity"], CM\_knn3$byClass["Specificity"])

### KNN - modelos preprocesados

#KNN - preprocesado con center y scale  
  
 #Downsample  
set.seed(1234567)  
knn\_1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "knn",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
knn\_1

k-Nearest Neighbors   
  
217 samples  
 10 predictor  
 2 classes: '1', '2'   
  
Pre-processing: centered (10), scaled (10)   
Resampling: Cross-Validated (10 fold, repeated 5 times)   
Summary of sample sizes: 195, 196, 195, 196, 195, 195, ...   
Resampling results across tuning parameters:  
  
 k Accuracy Kappa   
 5 0.5965801 0.1945450  
 7 0.6158442 0.2323075  
 9 0.6277056 0.2564418  
 11 0.6213853 0.2437805  
 13 0.6182251 0.2374968  
 15 0.6154545 0.2323886  
 17 0.6402597 0.2825925  
 19 0.6475758 0.2968398  
 21 0.6596537 0.3214379  
 23 0.6696104 0.3407745  
 25 0.6603463 0.3225886  
 27 0.6622078 0.3261229  
 29 0.6555844 0.3133590  
 31 0.6509524 0.3044264  
 33 0.6537662 0.3098549  
 35 0.6509957 0.3044260  
 37 0.6502597 0.3028858  
 39 0.6503896 0.3031355  
 41 0.6419913 0.2867685  
 43 0.6446753 0.2921404  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 23.

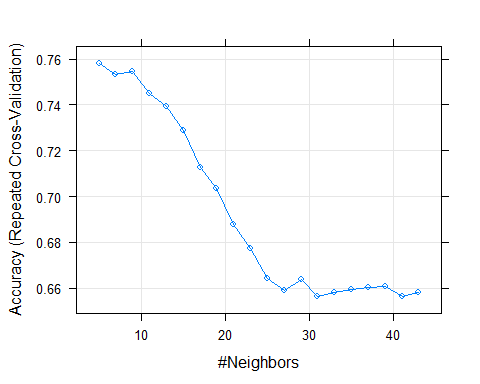
plot(knn\_1)



# ENN  
set.seed(1234567)  
knn\_2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "knn",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
knn\_2

k-Nearest Neighbors   
  
322 samples  
 10 predictor  
 2 classes: '1', '2'   
  
Pre-processing: centered (10), scaled (10)   
Resampling: Cross-Validated (10 fold, repeated 5 times)   
Summary of sample sizes: 290, 290, 290, 290, 290, 290, ...   
Resampling results across tuning parameters:  
  
 k Accuracy Kappa   
 5 0.7583040 0.4575299  
 7 0.7532295 0.4467238  
 9 0.7544000 0.4446778  
 11 0.7450037 0.4184342  
 13 0.7393371 0.4078800  
 15 0.7289241 0.3789114  
 17 0.7128030 0.3405670  
 19 0.7034445 0.3127984  
 21 0.6879735 0.2707403  
 23 0.6773851 0.2418535  
 25 0.6642742 0.1949620  
 27 0.6588019 0.1738742  
 29 0.6636871 0.1789271  
 31 0.6562427 0.1514979  
 33 0.6580633 0.1495010  
 35 0.6593133 0.1564853  
 37 0.6605217 0.1595524  
 39 0.6605761 0.1570645  
 41 0.6561999 0.1419958  
 43 0.6581140 0.1463795  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 5.

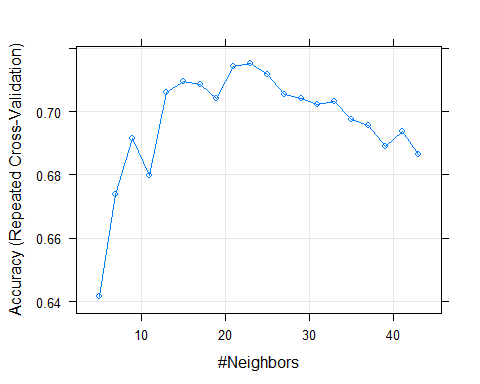
plot(knn\_2)



# ENN + downsample  
set.seed(1234567)  
knn\_3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "knn",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
knn\_3

k-Nearest Neighbors   
  
217 samples  
 10 predictor  
 2 classes: '1', '2'   
  
Pre-processing: centered (10), scaled (10)   
Resampling: Cross-Validated (10 fold, repeated 5 times)   
Summary of sample sizes: 195, 196, 195, 196, 195, 195, ...   
Resampling results across tuning parameters:  
  
 k Accuracy Kappa   
 5 0.6415584 0.2856425  
 7 0.6739394 0.3498077  
 9 0.6916017 0.3852283  
 11 0.6796970 0.3616132  
 13 0.7061472 0.4143208  
 15 0.7095238 0.4213011  
 17 0.7086580 0.4195696  
 19 0.7040260 0.4104982  
 21 0.7142424 0.4308932  
 23 0.7152814 0.4328909  
 25 0.7115152 0.4256539  
 27 0.7052381 0.4133246  
 29 0.7040693 0.4112511  
 31 0.7022511 0.4073272  
 33 0.7031602 0.4090964  
 35 0.6975758 0.3981238  
 37 0.6956710 0.3944891  
 39 0.6891342 0.3815163  
 41 0.6938095 0.3906194  
 43 0.6864069 0.3759434  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 23.

plot(knn\_3)

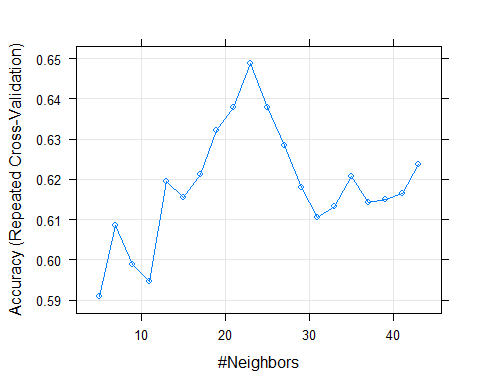


knn\_1pred <- predict(knn\_1, newdata = BD\_Ptest1 )  
CM\_knn1 <- confusionMatrix(knn\_1pred, BD\_Ptest1$Condicion)  
  
knn\_2pred <- predict(knn\_2, newdata = BD\_Ptest2 )  
CM\_knn2 <- confusionMatrix(knn\_2pred, BD\_Ptest2$Condicion)  
  
knn\_3pred <- predict(knn\_3, newdata = BD\_Ptest3 )  
CM\_knn3 <- confusionMatrix(knn\_3pred, BD\_Ptest3$Condicion)  
  
#Preprocesado center + scale   
precCS <- c(CM\_knn1$overall["Accuracy"], CM\_knn2$overall["Accuracy"], CM\_knn3$overall["Accuracy"])  
senCS <- c(CM\_knn1$byClass["Sensitivity"], CM\_knn2$byClass["Sensitivity"], CM\_knn3$byClass["Sensitivity"])  
speCS <- c(CM\_knn1$byClass["Specificity"], CM\_knn2$byClass["Specificity"], CM\_knn3$byClass["Specificity"])

#KNN - procesado con range  
  
 #Downsample  
set.seed(1234567)  
knn\_1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "knn",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneLength = 20)  
knn\_1

k-Nearest Neighbors   
  
217 samples  
 10 predictor  
 2 classes: '1', '2'   
  
Pre-processing: re-scaling to [0, 1] (10)   
Resampling: Cross-Validated (10 fold, repeated 5 times)   
Summary of sample sizes: 195, 196, 195, 196, 195, 195, ...   
Resampling results across tuning parameters:  
  
 k Accuracy Kappa   
 5 0.5908225 0.1835156  
 7 0.6084848 0.2185880  
 9 0.5989177 0.1987761  
 11 0.5947619 0.1910347  
 13 0.6195238 0.2399136  
 15 0.6156277 0.2334459  
 17 0.6212554 0.2443505  
 19 0.6322078 0.2663308  
 21 0.6379221 0.2776286  
 23 0.6488745 0.2992424  
 25 0.6377489 0.2767413  
 27 0.6283550 0.2587975  
 29 0.6180087 0.2382297  
 31 0.6105628 0.2234927  
 33 0.6132468 0.2291321  
 35 0.6206494 0.2437358  
 37 0.6142857 0.2310266  
 39 0.6151082 0.2327372  
 41 0.6164502 0.2356033  
 43 0.6236797 0.2498399  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 23.

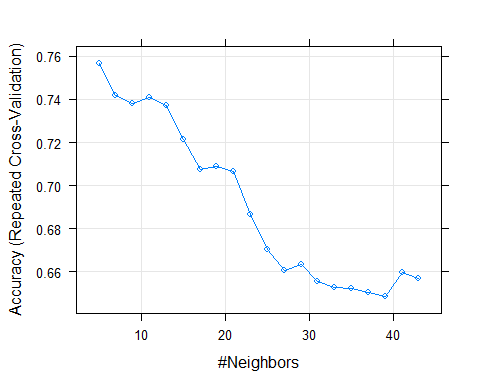
plot(knn\_1)



# ENN  
set.seed(1234567)  
knn\_2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "knn",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneLength = 20)  
knn\_2

k-Nearest Neighbors   
  
322 samples  
 10 predictor  
 2 classes: '1', '2'   
  
Pre-processing: re-scaling to [0, 1] (10)   
Resampling: Cross-Validated (10 fold, repeated 5 times)   
Summary of sample sizes: 290, 290, 290, 290, 290, 290, ...   
Resampling results across tuning parameters:  
  
 k Accuracy Kappa   
 5 0.7568609 0.45191919  
 7 0.7419972 0.41228739  
 9 0.7382081 0.40147889  
 11 0.7407460 0.40729089  
 13 0.7370326 0.38262351  
 15 0.7215426 0.34053559  
 17 0.7076906 0.30271812  
 19 0.7090756 0.29323061  
 21 0.7065530 0.28168612  
 23 0.6864522 0.22611761  
 25 0.6705065 0.17278294  
 27 0.6606971 0.13746911  
 29 0.6632160 0.14006277  
 31 0.6556769 0.11219935  
 33 0.6527224 0.09848389  
 35 0.6520986 0.09242502  
 37 0.6502212 0.08348899  
 39 0.6483663 0.07268537  
 41 0.6595228 0.10681998  
 43 0.6571377 0.09854735  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 5.

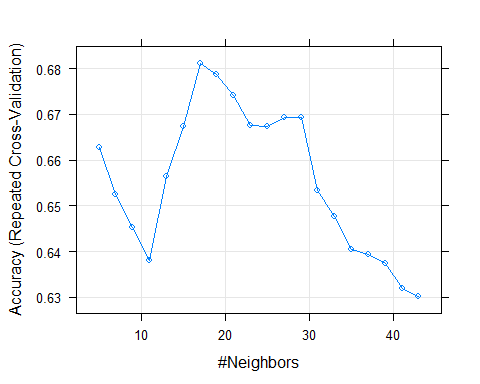
plot(knn\_2)



# ENN + downsample  
set.seed(1234567)  
knn\_3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "knn",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneLength = 20)  
knn\_3

k-Nearest Neighbors   
  
217 samples  
 10 predictor  
 2 classes: '1', '2'   
  
Pre-processing: re-scaling to [0, 1] (10)   
Resampling: Cross-Validated (10 fold, repeated 5 times)   
Summary of sample sizes: 195, 196, 195, 196, 195, 195, ...   
Resampling results across tuning parameters:  
  
 k Accuracy Kappa   
 5 0.6626840 0.3273965  
 7 0.6524675 0.3074367  
 9 0.6452814 0.2933150  
 11 0.6379654 0.2787192  
 13 0.6563203 0.3155067  
 15 0.6673160 0.3368721  
 17 0.6812121 0.3642500  
 19 0.6786147 0.3588723  
 21 0.6740260 0.3495483  
 23 0.6674892 0.3365198  
 25 0.6674459 0.3363827  
 27 0.6693074 0.3400010  
 29 0.6692208 0.3399957  
 31 0.6533333 0.3090911  
 33 0.6477056 0.2981811  
 35 0.6404329 0.2837398  
 37 0.6393939 0.2820388  
 39 0.6374892 0.2782380  
 41 0.6320346 0.2677346  
 43 0.6300866 0.2639883  
  
Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 17.

plot(knn\_3)



knn\_1pred <- predict(knn\_1, newdata = BD\_Ptest1 )  
CM\_knn1 <- confusionMatrix(knn\_1pred, BD\_Ptest1$Condicion)  
  
knn\_2pred <- predict(knn\_2, newdata = BD\_Ptest2 )  
CM\_knn2 <- confusionMatrix(knn\_2pred, BD\_Ptest2$Condicion)  
  
knn\_3pred <- predict(knn\_3, newdata = BD\_Ptest3 )  
CM\_knn3 <- confusionMatrix(knn\_3pred, BD\_Ptest3$Condicion)  
  
#Preprocesado range  
precR <- c(CM\_knn1$overall["Accuracy"], CM\_knn2$overall["Accuracy"], CM\_knn3$overall["Accuracy"])  
senR <- c(CM\_knn1$byClass["Sensitivity"], CM\_knn2$byClass["Sensitivity"], CM\_knn3$byClass["Sensitivity"])  
speR <- c(CM\_knn1$byClass["Specificity"], CM\_knn2$byClass["Specificity"], CM\_knn3$byClass["Specificity"])

## KNN - Tabla de datos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algoritmo | KNN - Precisión |  |  |  |  |
|  | Submuestreo | ENN | ENN y submuestreo |  |  |
| Sin procesar | 0.6542056 | 0.721519 | 0.6728972 |  |  |
| Procesado (Center y scale) | 0.588785 | 0.6962025 | 0.682243 |  |  |
| Procesado (Range) | 0.6168224 | 0.7088608 | 0.6728972 |  |  |

## Naive bayes - entrenamiento

Los datos y el preprocesado es el mismo que con el algoritmo knn.

#Naive Bayes - Sin preprocesar  
  
 # Downsample  
set.seed(1234567)  
NB1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "naive\_bayes",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
  
  
  
 # ENN  
set.seed(1234567)  
NB2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "naive\_bayes",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
  
  
  
 # ENN + downsample  
set.seed(1234567)  
NB3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "naive\_bayes",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)

El algoritmo basado en Naive Bayes generado emplea el kernel dado que mejora la precisión del modelo y ajusta su límite a 1.

## Naive bayes - evaluación

Se dará la evaluación con valores reales desconocidos de los que se sabe su verdadera clasificación, las bases de datos de prueba.

NB\_1 <- predict(NB1, newdata = BD\_Ptest1)  
CMNB1 <- confusionMatrix(NB\_1, BD\_Ptest1$Condicion)  
  
NB\_2 <- predict(NB2, newdata = BD\_Ptest2)  
CMNB2 <- confusionMatrix(NB\_2, BD\_Ptest2$Condicion)  
  
NB\_3 <- predict(NB3, newdata = BD\_Ptest3)  
CMNB3 <- confusionMatrix(NB\_3, BD\_Ptest3$Condicion)  
  
#Sin preprocesar  
precNO\_NB <- c(CMNB1$overall["Accuracy"], CMNB2$overall["Accuracy"], CMNB3$overall["Accuracy"])  
senNO\_NB <- c(CMNB1$byClass["Sensitivity"], CMNB2$byClass["Sensitivity"], CMNB3$byClass["Sensitivity"])  
speNO\_NB <- c(CMNB1$byClass["Specificity"], CMNB2$byClass["Specificity"], CMNB3$byClass["Specificity"])

Se almacenan los datos en las distintas matrices de confusión empleando la función confusionMatrix que tiene tanto la precisión real del modelo, como su sensibilidad y especificidad.

Para almacenar los valores de interés para los modelos procesados, hay que eliminar el icono de la almohadilla #, de tal forma se aplicará el preprocesado correspondiente en los datos del algoritmo.

### Naive Bayes - modelos preprocesados

#Naive Bayes - preprocesado con center y scale  
  
 # Downsample  
set.seed(1234567)  
NB1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "naive\_bayes",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
  
  
  
 # ENN  
set.seed(1234567)  
NB2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "naive\_bayes",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)  
  
  
  
 # ENN + downsample  
set.seed(1234567)  
NB3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "naive\_bayes",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneLength = 20)

NB\_1 <- predict(NB1, newdata = BD\_Ptest1)  
CMNB1 <- confusionMatrix(NB\_1, BD\_Ptest1$Condicion)  
  
NB\_2 <- predict(NB2, newdata = BD\_Ptest2)  
CMNB2 <- confusionMatrix(NB\_2, BD\_Ptest2$Condicion)  
  
NB\_3 <- predict(NB3, newdata = BD\_Ptest3)  
CMNB3 <- confusionMatrix(NB\_3, BD\_Ptest3$Condicion)  
  
#preprocesado center + scale   
precCS\_NB <- c(CMNB1$overall["Accuracy"], CMNB2$overall["Accuracy"], CMNB3$overall["Accuracy"])  
senCS\_NB <- c(CMNB1$byClass["Sensitivity"], CMNB2$byClass["Sensitivity"], CMNB3$byClass["Sensitivity"])  
speCS\_NB <- c(CMNB1$byClass["Specificity"], CMNB2$byClass["Specificity"], CMNB3$byClass["Specificity"])

#Naive Bayes - preprocesado con range  
  
 # Downsample  
set.seed(1234567)  
NB1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "naive\_bayes",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneLength = 20)  
  
  
  
 # ENN  
set.seed(1234567)  
NB2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "naive\_bayes",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneLength = 20)  
  
  
  
 # ENN + downsample  
set.seed(1234567)  
NB3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "naive\_bayes",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneLength = 20)

NB\_1 <- predict(NB1, newdata = BD\_Ptest1)  
CMNB1 <- confusionMatrix(NB\_1, BD\_Ptest1$Condicion)  
  
NB\_2 <- predict(NB2, newdata = BD\_Ptest2)  
CMNB2 <- confusionMatrix(NB\_2, BD\_Ptest2$Condicion)  
  
NB\_3 <- predict(NB3, newdata = BD\_Ptest3)  
CMNB3 <- confusionMatrix(NB\_3, BD\_Ptest3$Condicion)  
  
#preprocesado range  
precR\_NB <- c(CMNB1$overall["Accuracy"], CMNB2$overall["Accuracy"], CMNB3$overall["Accuracy"])  
senR\_NB <- c(CMNB1$byClass["Sensitivity"], CMNB2$byClass["Sensitivity"], CMNB3$byClass["Sensitivity"])  
speR\_NB <- c(CMNB1$byClass["Specificity"], CMNB2$byClass["Specificity"], CMNB3$byClass["Specificity"])

## NAIVE BAYES - Tabla de datos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algoritmo | NB - Precisión |  |  |  |  |
|  | Submuestreo | ENN | ENN y submuestreo |  |  |
| Sin procesar | 0.635514 | 0.7594937 | 0.6728972 |  |  |
| Procesado (Center y scale) | 0.635514 | 0.7594937 | 0.6728972 |  |  |
| Procesado (Range) | 0.635514 | 0.7658228 | 0.6728972 |  |  |

## ANN

Los datos y el preprocesado es el mismo que con el algoritmo knn.

## ANN - entrenamiento

Para entrenar el modelo uno debe seleccionar las variables de la base de datos que conformarán las predictoras y la variable respuesta. En este caso la variable respuesta muestra presencia de tener un problema hepático en pacientes (variable Condicion).

#ANN - Sin preprocesar  
  
 # Downsample  
set.seed(1234567)  
ANN1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "nnet",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(size= c(1,3,5,7,9,10), decay = seq(from = 0.1, to = 0.5, by = 0.1))  
 )  
  
  
 # ENN   
set.seed(1234567)  
ANN2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "nnet",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(size= c(1,3,5,7,9,10), decay = seq(from = 0.1, to = 0.5, by = 0.1))  
 )  
  
  
 # ENN + Downsample  
set.seed(1234567)  
ANN3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "nnet",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(size= c(1,3,5,7,9,10), decay = seq(from = 0.1, to = 0.5, by = 0.1))  
 )

## ANN - evaluación

ANN\_1 <- predict(ANN1, newdata = BD\_Ptest1 )  
CMANN1 <- confusionMatrix(ANN\_1, BD\_Ptest1$Condicion)  
  
ANN\_2 <- predict(ANN2, newdata = BD\_Ptest2 )  
CMANN2 <- confusionMatrix(ANN\_2, BD\_Ptest2$Condicion)  
  
ANN\_3 <- predict(ANN3, newdata = BD\_Ptest3 )  
CMANN3 <- confusionMatrix(ANN\_3, BD\_Ptest3$Condicion)  
  
#Sin preprocesar  
precNO\_ANN <- c(CMANN1$overall["Accuracy"], CMANN2$overall["Accuracy"], CMANN3$overall["Accuracy"])  
senNO\_ANN <- c(CMANN1$byClass["Sensitivity"], CMANN2$byClass["Sensitivity"], CMANN3$byClass["Sensitivity"])  
speNO\_ANN <- c(CMANN1$byClass["Specificity"], CMANN2$byClass["Specificity"], CMANN3$byClass["Specificity"])

Empleando la función de confusionMatrix se puede vislumbrar la comparación entre los valores reales y los predichos por la red neuronal. Se accede a los valores de interés y estos son almacenados en distintos vectores específicos.

### ANN - Modelos preprocesados

#ANN - preprocesado con center y scale  
  
 # Downsample  
set.seed(1234567)  
ANN1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "nnet",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(size= c(1,3,5,7,9,10), decay = seq(from = 0.1, to = 0.5, by = 0.1))  
 )  
  
  
 # ENN   
set.seed(1234567)  
ANN2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "nnet",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(size= c(1,3,5,7,9,10), decay = seq(from = 0.1, to = 0.5, by = 0.1))  
 )  
  
  
 # ENN + Downsample  
set.seed(1234567)  
ANN3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "nnet",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(size= c(1,3,5,7,9,10), decay = seq(from = 0.1, to = 0.5, by = 0.1))  
 )

ANN\_1 <- predict(ANN1, newdata = BD\_Ptest1 )  
CMANN1 <- confusionMatrix(ANN\_1, BD\_Ptest1$Condicion)  
  
ANN\_2 <- predict(ANN2, newdata = BD\_Ptest2 )  
CMANN2 <- confusionMatrix(ANN\_2, BD\_Ptest2$Condicion)  
  
ANN\_3 <- predict(ANN3, newdata = BD\_Ptest3 )  
CMANN3 <- confusionMatrix(ANN\_3, BD\_Ptest3$Condicion)  
  
#preprocesado con center y scale  
preCS\_ANN <- c(CMANN1$overall["Accuracy"], CMANN2$overall["Accuracy"], CMANN3$overall["Accuracy"])  
senCS\_ANN <- c(CMANN1$byClass["Sensitivity"], CMANN2$byClass["Sensitivity"], CMANN3$byClass["Sensitivity"])  
speCS\_ANN <- c(CMANN1$byClass["Specificity"], CMANN2$byClass["Specificity"], CMANN3$byClass["Specificity"])

#ANN - preprocesado con range  
  
 # Downsample  
set.seed(1234567)  
ANN1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "nnet",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneGrid = expand.grid(size= c(1,3,5,7,9,10), decay = seq(from = 0.1, to = 0.5, by = 0.1))  
 )  
  
  
 # ENN   
set.seed(1234567)  
ANN2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "nnet",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneGrid = expand.grid(size= c(1,3,5,7,9,10), decay = seq(from = 0.1, to = 0.5, by = 0.1))  
 )  
  
  
 # ENN + Downsample  
set.seed(1234567)  
ANN3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "nnet",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneGrid = expand.grid(size= c(1,3,5,7,9,10), decay = seq(from = 0.1, to = 0.5, by = 0.1))  
 )

ANN\_1 <- predict(ANN1, newdata = BD\_Ptest1 )  
CMANN1 <- confusionMatrix(ANN\_1, BD\_Ptest1$Condicion)  
  
ANN\_2 <- predict(ANN2, newdata = BD\_Ptest2 )  
CMANN2 <- confusionMatrix(ANN\_2, BD\_Ptest2$Condicion)  
  
ANN\_3 <- predict(ANN3, newdata = BD\_Ptest3 )  
CMANN3 <- confusionMatrix(ANN\_3, BD\_Ptest3$Condicion)  
  
#Preprocesado con range  
preR\_ANN <- c(CMANN1$overall["Accuracy"], CMANN2$overall["Accuracy"], CMANN3$overall["Accuracy"])  
senR\_ANN <- c(CMANN1$byClass["Sensitivity"], CMANN2$byClass["Sensitivity"], CMANN3$byClass["Sensitivity"])  
speR\_ANN <- c(CMANN1$byClass["Specificity"], CMANN2$byClass["Specificity"], CMANN3$byClass["Specificity"])

## ANN - Tabla de datos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algoritmo | ANN - Precisión |  |  |  |  |
|  | Submuestreo | ENN | ENN y submuestreo |  |  |
| Sin procesar | 0.635514 | 0.721519 | 0.6728972 |  |  |
| Procesado (Center y scale) | 0.6542056 | 0.7594937 | 0.7383178 |  |  |
| Procesado (Range) | 0.6168224 | 0.7151899 | 0.7102804 |  |  |

## SVM

Los datos y el preprocesado es el mismo que con el algoritmo knn.

## SVM - entrenamiento

#SVM - Sin preprocesar  
  
# Downsample  
set.seed(1234567)  
SVM1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "svmLinear",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(C= c(2^(2:9))))  
  
  
# ENN  
set.seed(1234567)  
SVM2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "svmLinear",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(C= c(2^(2:9))))  
  
  
# ENN + downsample  
set.seed(1234567)  
SVM3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "svmLinear",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(C= c(2^(2:9))))

## SVM - evaluación

SVM\_1 <- predict(SVM1, newdata = BD\_Ptest1)  
CMSVM1 <- confusionMatrix(SVM\_1, BD\_Ptest1$Condicion)  
  
SVM\_2 <- predict(SVM2, newdata = BD\_Ptest2)  
CMSVM2 <- confusionMatrix(SVM\_2, BD\_Ptest2$Condicion)  
  
SVM\_3 <- predict(SVM3, newdata = BD\_Ptest3)  
CMSVM3 <- confusionMatrix(SVM\_3, BD\_Ptest3$Condicion)  
  
#Sin preprocesar  
precNO\_SVML <- c(CMSVM1$overall["Accuracy"], CMSVM2$overall["Accuracy"], CMSVM3$overall["Accuracy"])  
senNO\_SVML <- c(CMSVM1$byClass["Sensitivity"], CMSVM2$byClass["Sensitivity"], CMSVM3$byClass["Sensitivity"])  
speNO\_SVML <- c(CMSVM1$byClass["Specificity"], CMSVM2$byClass["Specificity"], CMSVM3$byClass["Specificity"])

### SVM - Modelos preprocesados

# SVM - Preprocesado con center y scale   
  
# Downsample  
set.seed(1234567)  
SVM1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "svmLinear",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(C= c(2^(2:9))))  
  
  
# ENN  
set.seed(1234567)  
SVM2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "svmLinear",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(C= c(2^(2:9))))  
  
  
# ENN + downsample  
set.seed(1234567)  
SVM3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "svmLinear",  
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 tuneGrid = expand.grid(C= c(2^(2:9))))

SVM\_1 <- predict(SVM1, newdata = BD\_Ptest1)  
CMSVM1 <- confusionMatrix(SVM\_1, BD\_Ptest1$Condicion)  
  
SVM\_2 <- predict(SVM2, newdata = BD\_Ptest2)  
CMSVM2 <- confusionMatrix(SVM\_2, BD\_Ptest2$Condicion)  
  
SVM\_3 <- predict(SVM3, newdata = BD\_Ptest3)  
CMSVM3 <- confusionMatrix(SVM\_3, BD\_Ptest3$Condicion)  
  
#preprocesado center + scale   
precCS\_SVML <- c(CMSVM1$overall["Accuracy"], CMSVM2$overall["Accuracy"], CMSVM3$overall["Accuracy"])  
senCS\_SVML <- c(CMSVM1$byClass["Sensitivity"], CMSVM2$byClass["Sensitivity"], CMSVM3$byClass["Sensitivity"])  
speCS\_SVML <- c(CMSVM1$byClass["Specificity"], CMSVM2$byClass["Specificity"], CMSVM3$byClass["Specificity"])

#SVM - Preprocesado con range  
  
# Downsample  
set.seed(1234567)  
SVM1 <- train(Condicion ~ ., data = BD\_Ptrain1, method = "svmLinear",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneGrid = expand.grid(C= c(2^(2:9))))  
  
  
# ENN  
set.seed(1234567)  
SVM2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "svmLinear",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneGrid = expand.grid(C= c(2^(2:9))))  
  
  
# ENN + downsample  
set.seed(1234567)  
SVM3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "svmLinear",  
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 tuneGrid = expand.grid(C= c(2^(2:9))))

El modelo generado del “support vector machine” emplea un kernel lineal, que implica que los vectores utilizados para la clasificación son líneas rectas.

SVM\_1 <- predict(SVM1, newdata = BD\_Ptest1)  
CMSVM1 <- confusionMatrix(SVM\_1, BD\_Ptest1$Condicion)  
  
SVM\_2 <- predict(SVM2, newdata = BD\_Ptest2)  
CMSVM2 <- confusionMatrix(SVM\_2, BD\_Ptest2$Condicion)  
  
SVM\_3 <- predict(SVM3, newdata = BD\_Ptest3)  
CMSVM3 <- confusionMatrix(SVM\_3, BD\_Ptest3$Condicion)  
  
#preprocesado range  
precR\_SVML <- c(CMSVM1$overall["Accuracy"], CMSVM2$overall["Accuracy"], CMSVM3$overall["Accuracy"])  
senR\_SVML <- c(CMSVM1$byClass["Sensitivity"], CMSVM2$byClass["Sensitivity"], CMSVM3$byClass["Sensitivity"])  
speR\_SVML <- c(CMSVM1$byClass["Specificity"], CMSVM2$byClass["Specificity"], CMSVM3$byClass["Specificity"])

## SVM - Tabla de datos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algoritmo | SVM - Precisión |  |  |  |  |
|  | Submuestreo | ENN | ENN y submuestreo |  |  |
| Sin procesar | 0.6168224 | 0.721519 | 0.7663551 |  |  |
| Procesado (Center y scale) | 0.6168224 | 0.721519 | 0.7663551 |  |  |
| Procesado (Range) | 0.6168224 | 0.7151899 | 0.7663551 |  |  |

## Random Forest - entrenamiento

#RANDOM FOREST - Sin preprocesar  
 #Dowsample  
set.seed(1234567)  
RF1 <- train(Condicion ~., data=BD\_Ptrain1, method='rf',   
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 metric='Accuracy',   
 tuneLength = 9)  
  
  
 #ENN  
set.seed(1234567)  
RF2 <- train(Condicion ~., data=BD\_Ptrain2, method='rf',   
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 metric='Accuracy',   
 tuneLength = 9)  
  
  
 #ENN + Downsample  
set.seed(1234567)  
RF3 <- train(Condicion ~., data=BD\_Ptrain3, method='rf',   
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 #preProcess = "range",  
 metric='Accuracy',   
 tuneLength = 9)

RF\_1 <- predict(RF1, newdata = BD\_Ptest1 )  
CM\_RF1 <- confusionMatrix(RF\_1, BD\_Ptest1$Condicion)  
  
RF\_2 <- predict(RF2, newdata = BD\_Ptest2 )  
CM\_RF2 <- confusionMatrix(RF\_2, BD\_Ptest2$Condicion)  
  
RF\_3 <- predict(RF3, newdata = BD\_Ptest3 )  
CM\_RF3 <- confusionMatrix(RF\_3, BD\_Ptest3$Condicion)  
  
  
#Sin preprocesar  
precNO\_rf <- c(CM\_RF1$overall["Accuracy"], CM\_RF2$overall["Accuracy"], CM\_RF3$overall["Accuracy"])  
senNO\_rf <- c(CM\_RF1$byClass["Sensitivity"], CM\_RF2$byClass["Sensitivity"], CM\_RF3$byClass["Sensitivity"])  
speNO\_rf <- c(CM\_RF1$byClass["Specificity"], CM\_RF2$byClass["Specificity"], CM\_RF3$byClass["Specificity"])  
  
#Variables importantes  
VIMPRF1NO <- varImp(RF1)  
VIMPRF2NO <- varImp(RF2)  
VIMPRF3NO <- varImp(RF3)

### Random Forest - Modelos preprocesados

#RANDOM FOREST - Preprocesado con center y scale  
 #Dowsample  
set.seed(1234567)  
RF1 <- train(Condicion ~., data=BD\_Ptrain1, method='rf',   
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 metric='Accuracy',   
 tuneLength = 9)  
  
  
  
 #ENN  
set.seed(1234567)  
RF2 <- train(Condicion ~., data=BD\_Ptrain2, method='rf',   
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 metric='Accuracy',   
 tuneLength = 9)  
  
  
 #ENN + Downsample  
set.seed(1234567)  
RF3 <- train(Condicion ~., data=BD\_Ptrain3, method='rf',   
 trControl = ctrl,  
 preProcess = c("center","scale"),  
 #preProcess = "range",  
 metric='Accuracy',   
 tuneLength = 9)

RF\_1 <- predict(RF1, newdata = BD\_Ptest1 )  
CM\_RF1 <- confusionMatrix(RF\_1, BD\_Ptest1$Condicion)  
  
RF\_2 <- predict(RF2, newdata = BD\_Ptest2 )  
CM\_RF2 <- confusionMatrix(RF\_2, BD\_Ptest2$Condicion)  
  
RF\_3 <- predict(RF3, newdata = BD\_Ptest3 )  
CM\_RF3 <- confusionMatrix(RF\_3, BD\_Ptest3$Condicion)  
  
#Preprocesado center y scale  
precCS\_rf <- c(CM\_RF1$overall["Accuracy"], CM\_RF2$overall["Accuracy"], CM\_RF3$overall["Accuracy"])  
senCS\_rf <- c(CM\_RF1$byClass["Sensitivity"], CM\_RF2$byClass["Sensitivity"], CM\_RF3$byClass["Sensitivity"])  
speCS\_rf <- c(CM\_RF1$byClass["Specificity"], CM\_RF2$byClass["Specificity"], CM\_RF3$byClass["Specificity"])  
  
#Variables importantes  
VIMPRF1CS <- varImp(RF1)  
VIMPRF2CS <- varImp(RF2)  
VIMPRF3CS <- varImp(RF3)

#RANDOM FOREST - Preprocesado con range  
 #Dowsample  
set.seed(1234567)  
RF1 <- train(Condicion ~., data=BD\_Ptrain1, method='rf',   
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 metric='Accuracy',   
 tuneLength = 9)  
  
  
  
 #ENN  
set.seed(1234567)  
RF2 <- train(Condicion ~., data=BD\_Ptrain2, method='rf',   
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 metric='Accuracy',   
 tuneLength = 9)  
  
  
 #ENN + Downsample  
set.seed(1234567)  
RF3 <- train(Condicion ~., data=BD\_Ptrain3, method='rf',   
 trControl = ctrl,  
 #preProcess = c("center","scale"),  
 preProcess = "range",  
 metric='Accuracy',   
 tuneLength = 9)

RF\_1 <- predict(RF1, newdata = BD\_Ptest1 )  
CM\_RF1 <- confusionMatrix(RF\_1, BD\_Ptest1$Condicion)  
  
RF\_2 <- predict(RF2, newdata = BD\_Ptest2 )  
CM\_RF2 <- confusionMatrix(RF\_2, BD\_Ptest2$Condicion)  
  
RF\_3 <- predict(RF3, newdata = BD\_Ptest3 )  
CM\_RF3 <- confusionMatrix(RF\_3, BD\_Ptest3$Condicion)  
  
#Preprocesado range  
precR\_rf <- c(CM\_RF1$overall["Accuracy"], CM\_RF2$overall["Accuracy"], CM\_RF3$overall["Accuracy"])  
senR\_rf <- c(CM\_RF1$byClass["Sensitivity"], CM\_RF2$byClass["Sensitivity"], CM\_RF3$byClass["Sensitivity"])  
speR\_rf <- c(CM\_RF1$byClass["Specificity"], CM\_RF2$byClass["Specificity"], CM\_RF3$byClass["Specificity"])  
  
#Variables importantes  
VIMPRF1R <- varImp(RF1)  
VIMPRF2R <- varImp(RF2)  
VIMPRF3R <- varImp(RF3)

## RF - Tabla de datos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algoritmo | SVM - Precisión |  |  |  |  |
|  | Submuestreo | ENN | ENN y submuestreo |  |  |
| Sin procesar | 0.6635514 | 0.9113924 | 0.7476636 |  |  |
| Procesado (Center y scale) | 0.6448598 | 0.7025316 | 0.7476636 |  |  |
| Procesado (Range) | 0.6542056 | 0.9113924 | 0.7383178 |  |  |

# Modelos elegidos:

SVM - ENN + Submuestreo y Preprocesado range

Naive Bayes - ENN y Preprocesado range

ANN - ENN y Preprocesado center + scale

# SVM - ENN + Submuestreo y Preprocesado range  
#set.seed(1234567)  
#SVM3 <- train(Condicion ~ ., data = BD\_Ptrain3, method = "svmLinear",  
# trControl = ctrl,  
# #preProcess = c("center","scale"),  
# preProcess = "range",  
# tuneGrid = expand.grid(C= c(2)))  
#SVM3  
  
#Coste 2  
  
#SVM\_PRED <- predict(SVM3, newdata = BD\_Ptest3 )  
#CM\_SVMDEF <- confusionMatrix(SVM\_PRED, BD\_Ptest3$Condicion)  
  
# Naive Bayes - ENN y Preprocesado range  
#set.seed(1234567)  
#NB2 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "naive\_bayes",  
# trControl = ctrl,  
# #preProcess = c("center","scale"),  
# preProcess = "range",  
# tuneLength = 20)  
#NB2  
  
#Laplace 0, uso de kernel = TRUE y ajuste constante a 1.  
  
#NB\_PRED <- predict(NB2, newdata = BD\_Ptest2 )  
#CM\_NBDEF <- confusionMatrix(NB\_PRED, BD\_Ptest2$Condicion)  
  
  
# ANN - ENN y center + scale  
#set.seed(1234567)  
#ANN3 <- train(Condicion ~ ., data = BD\_Ptrain2, method = "nnet",  
# trControl = ctrl,  
# preProcess = c("center","scale"),  
# #preProcess = "range",  
# tuneGrid = expand.grid(size= 7, decay = 0.1)  
# )  
#ANN3  
  
#nodos 7 y decaimiento 0,1  
  
#ANN\_PRED <- predict(ANN3, newdata = BD\_Ptest2 )  
#CM\_ANNDEF <- confusionMatrix(ANN\_PRED, BD\_Ptest2$Condicion)

*TABLA base de datos*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variable | Máximo | Mínimo | Media | Desviación estándar | Factores |
| Edad | 90 | 4 | 44.8869258 | 16.2748932 |  |
| Género | - | - | - | - | Female: 138, Male: 428 |
| TBil | 75 | 0.4 | 3.3388693 | 6.2867278 |  |
| DBil | 19.7 | 0.1 | 1.5058304 | 2.841485 |  |
| Alkphos | 2110 | 63 | 292.5671378 | 245.936559 |  |
| Spgt | 2000 | 10 | 80.1431095 | 182.0448812 |  |
| Sgot | 4929 | 10 | 109.8922261 | 291.8418969 |  |
| TP | 9.6 | 2.7 | 6.4948763 | 1.0875117 |  |
| Albúmina | 5.5 | 0.9 | 3.145583 | 0.7957453 |  |
| Ratio | 2.8 | 0.3 | 0.9480035 | 0.3196354 |  |