Modelos de Clasificacion

Daniela Cuesta - Paola Peralta Flores

Se realizan cinco tipo de clasificadores.

```
# Cargamos librerias
library(ggplot2)
library(ggpubr)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(glmnet) ## regresiones logistIcas
```

```
Loading required package: Matrix

Loaded glmnet 4.1-7
```

```
library(caret) ### bayes y knn
```

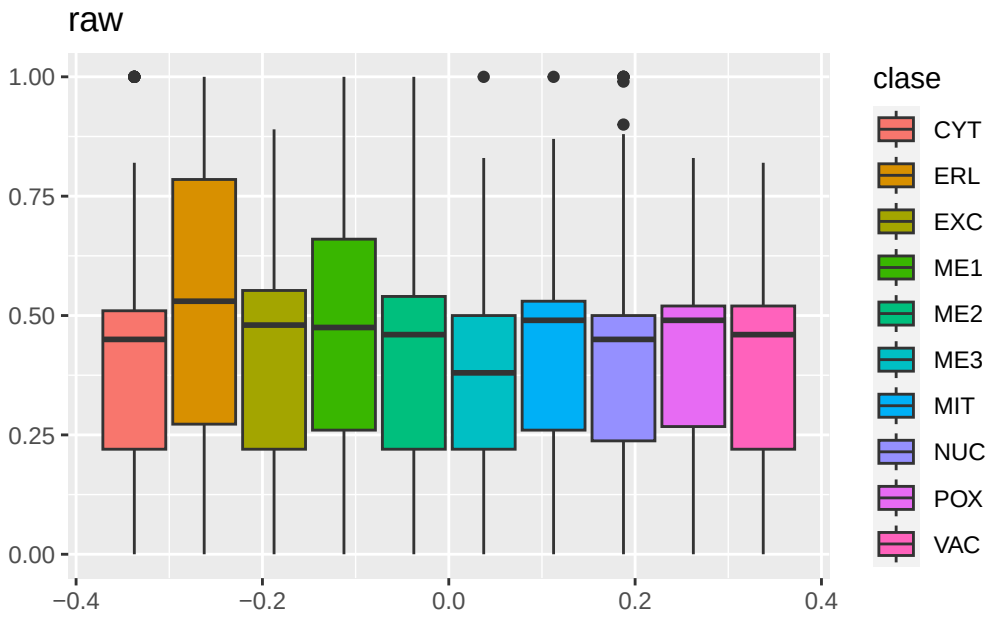
```
Loading required package: lattice
```

```r
library(e1071) ## bayes
```

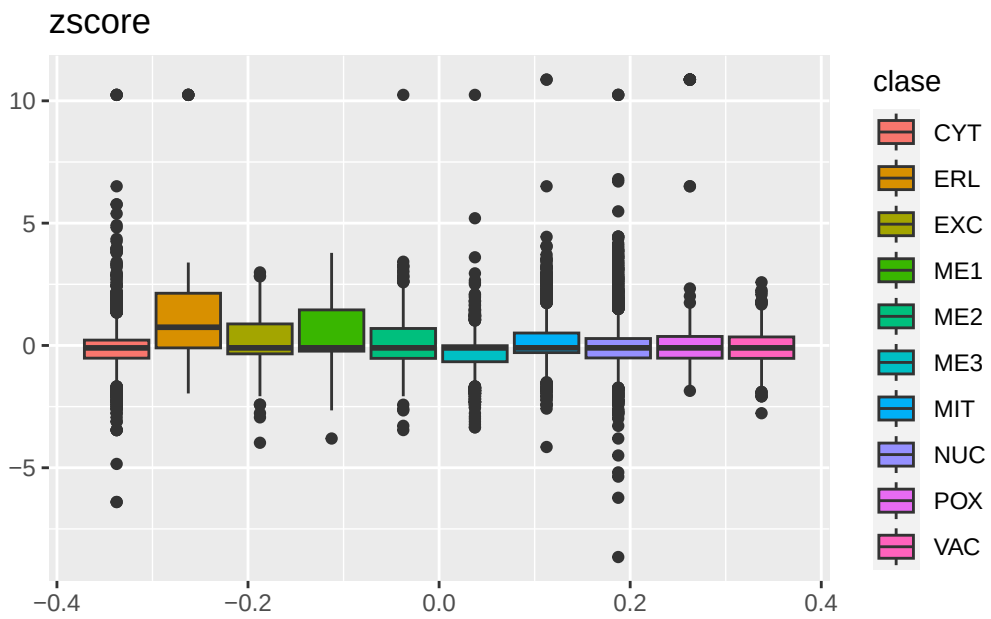
```r
# Quitamos la primera columna
datos <- read.table("./yeast.data",header = F)[,-1]

# Funciones de transformacion
min.max.mean <- function(X) apply(X,2,function(x) (x-mean(x))/(max(x)-min(x)))
min.max.median <- function(X) apply(X,2,function(x) (x-median(x))/(max(x)-min(x)))
min.max <- function(X) apply(X,2,function(x) (x-min(x))/(max(x)-min(x)))
zscore <- function(X) apply(X,2,function(x) (x-mean(x))/sd(x))
l2 <- function(X) apply(X,2,function(x) x/sqrt(sum(x^2)))

# Particion de datos datos <- as.data.frame(datos)
datos.numericos <- datos[, which(unlist(lapply(datos, is.numeric)))]
clase <- datos$V10 <- as.factor(datos$V10)
colnames(datos.numericos) <- paste0("Var", rep(1:8))

# Procedemos a crear una lista con todas las transformaciones
datos.lista <- list(
  raw = bind_cols(datos.numericos,clase=clase),
  zscore = bind_cols(zscore(datos.numericos),
                     clase = clase),
  l2 = bind_cols(l2(datos.numericos), clase = clase),
  media = bind_cols(min.max.mean(datos.numericos), clase =
                      clase),
  mediana = bind_cols(min.max.median(datos.numericos), clase =
                        clase),
  min_max = bind_cols(min.max(datos.numericos),
  clase = clase))

# Al ser demasiadas variables, podemos realizar un melt
lista_graficos <- vector("list",length=length(datos.lista))
datos.melt <- lapply(datos.lista,reshape2::melt)
```

```
Using clase as id variables
Using clase as id variables
Using clase as id variables
Using clase as id variables
Using clase as id variables
Using clase as id variables
```

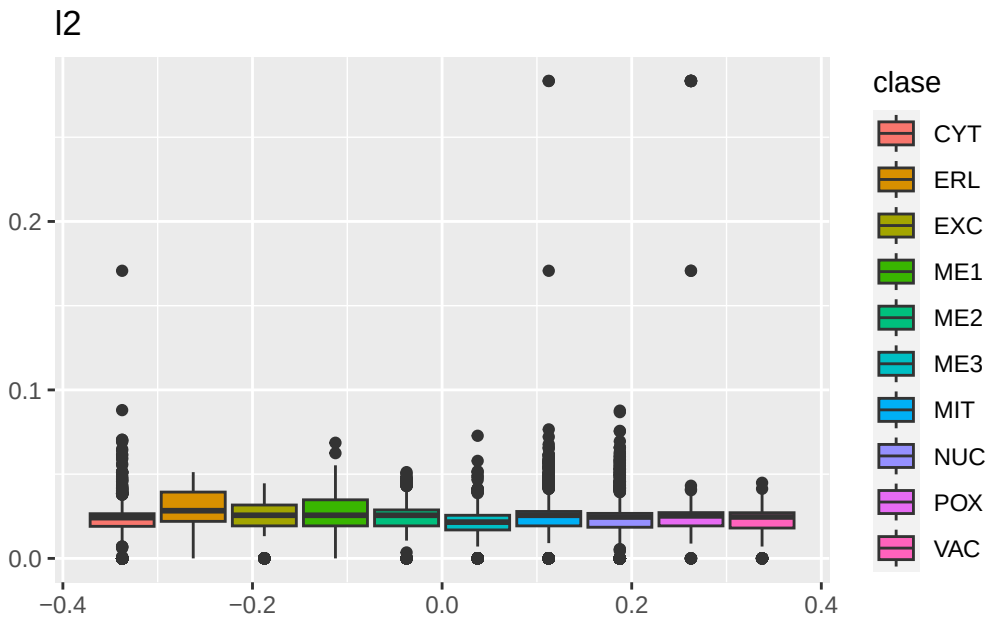
```
# Graficos
for(l in 1:length(datos.melt)){

  X <- datos.melt[[l]]
  nombre <- names(datos.melt)[l]
  lista_graficos[[l]] <- ggplot(X,aes(y=value,fill=clase))+geom_boxplot()+
    ggtitle(nombre)+xlab("")+ylab("")

}

names(lista_graficos) <- paste0("plot",1:length(datos.lista))

lista_graficos$plot1
```



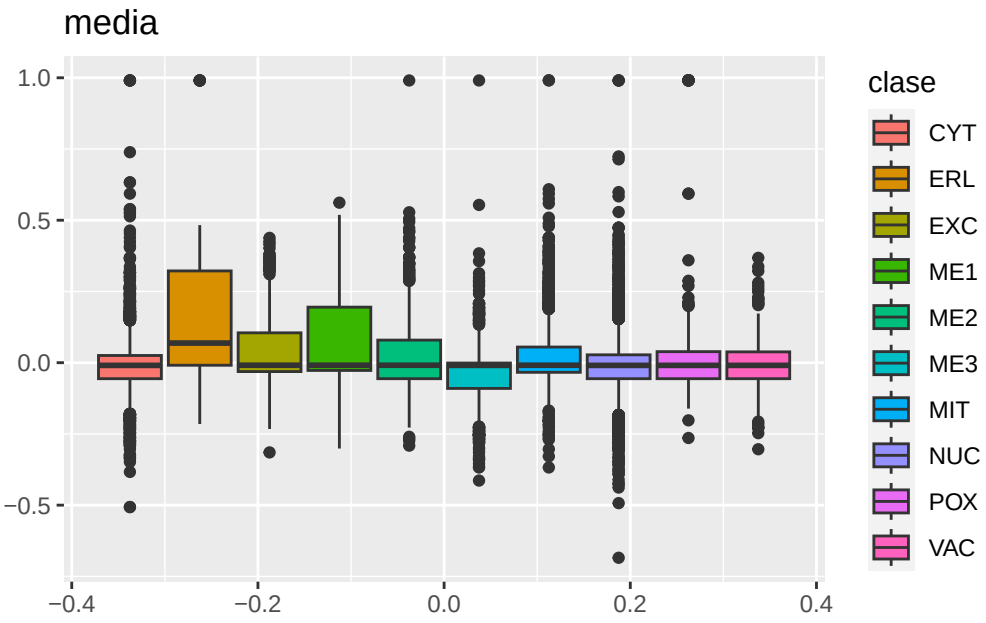

```
lista_graficos$plot2
```



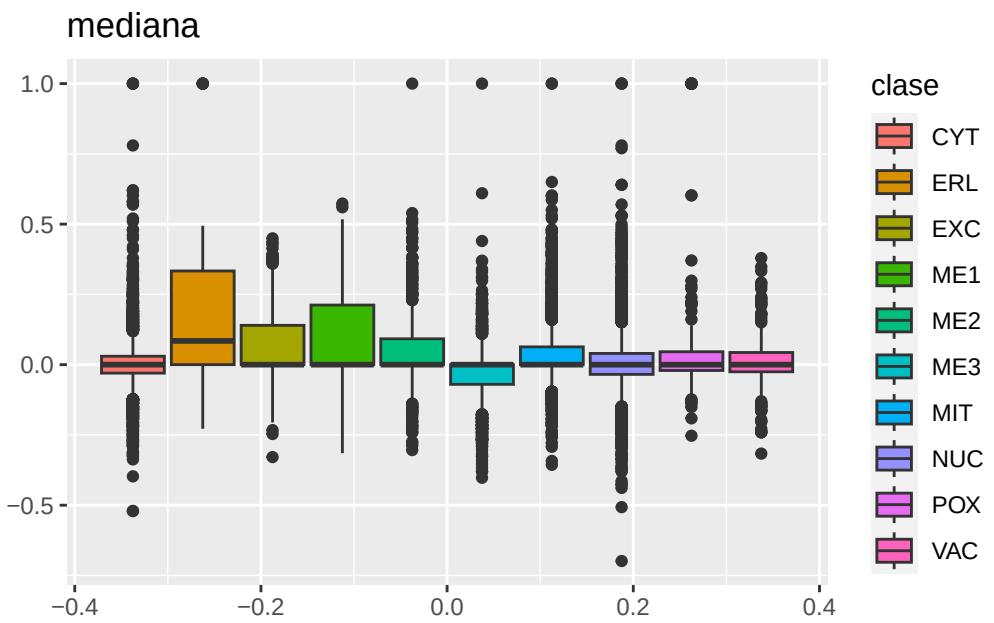

```
lista_graficos$plot3
```



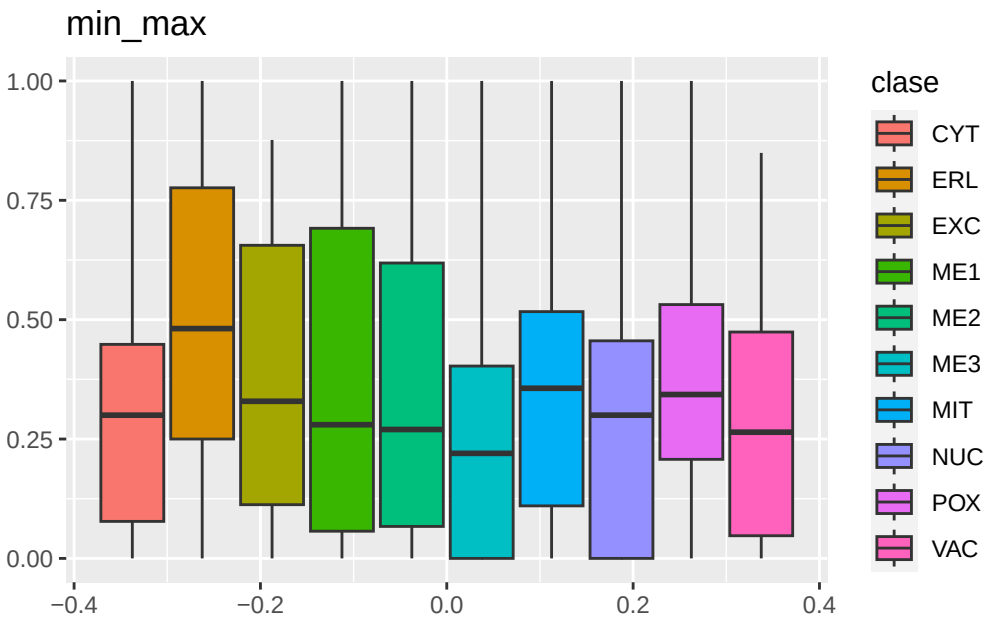
```
lista_graficos$plot4
```




```
lista_graficos$plot5
```




```
lista_graficos$plot6
```



```r
# Fijamos la semilla y la muestra
set.seed(123456789)
trControl <- trainControl(method = 'cv', number = 10)
n  <- nrow(datos)
idx <- sample(1:n,size=n*0.7,replace=F)
lambda_seq <- seq(0.01, 1, by = 0.01)

# Para conjunto de datos podemos realizar el split
entrenamiento <- lapply(datos.lista, function(x) x[idx,])
test <- lapply(datos.lista, function(x) x[-idx,])
```

REGRESION LOGISTICA LINEAL

```r
# Semilla
set.seed(123456789)

# Se establece una funcion de entrenamiento
myfnlog <- function(x) train(clase ~ ., data = x, method = "multinom",
                             trControl = trControl, trace = F)

# Se aplica la funcion creada a los datos de entrenamiento
logistica.lista <- lapply(entrenamiento,myfnlog)
logisita.pred <- vector("list",length = length(datos.lista))

# Se crea un vector de la misma longitud de datos.lista
for(l in 1:length(datos.lista)){
  logisita.pred[[l]] <- predict(logistica.lista[[l]],test[[l]])
}
names(logisita.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))

# Se asignan valres a las posiciones del vector.pred
for(l in 1:length(datos.lista)){
  accuracy[l] <- confusionMatrix(test$raw$clase,logisita.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
accuracy_logis<-accuracy
accuracy_logis
```

```
      raw    zscore        l2     media   mediana   min_max
0.5919283 0.5919283 0.5964126 0.5919283 0.5941704 0.5919283
```

RIDGE

```r
# Esta línea establece la semilla para la generación de números aleatorios.
# Al establecer una semilla, se asegura que los resultados sean reproducibles.
# Al ejecutar el mismo código con la misma semilla, se obtendrán los mismos resultados.
set.seed(123456789)

# La función ridge sigue una estructura similar a la función lasso
# En este caso se utiliza alpha = 0 para especificar el método de regularización Ridge.
# Los parámetros (clase ~ ., data, trControl, tuneGrid y trace) se mantienen similares.
ridge <- function(x) train(clase ~ ., data = x, method = "glmnet",
                           trControl = trControl,tuneGrid = expand.grid
                           (alpha=0,lambda=lambda_seq), trace = F)

# Se aplica la función ridge a cada elemento de la lista entrenamiento.
# lapply aplica una función a cada elemento de una lista
# lapply devuelve una nueva lista con los resultados.
ridge.lista <- lapply(entrenamiento,ridge)
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
```

```
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```r
ridge.pred <- vector("list",length = length(datos.lista))

# Se crea un vector llamado ridge.pred de la misma longitud que la lista datos.lista
#y se utilizan los modelos entrenados (ridge.lista) para hacer predicciones en los
#datos de prueba (test).
# Se asignan los nombres de datos.lista a los elementos de ridge.pred.
# Finalmente, se crea un vector llamado accuracy de tipo numérico con una longitud
#igual a la longitud de datos.lista.
for(l in 1:length( datos.lista)){
  ridge.pred[[l]] <- predict(ridge.lista[[l]],test[[l]])
}
names(ridge.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))

# Se utiliza un bucle for para calcular la exactitud de las predicciones.
# El valor de exactitud se guarda en el vector accuracy en la posición correspondiente.
# Finalmente, se crea un vector "accuracy_ridge" que almacena los valores de exactitud.
for(l in 1:length(datos.lista)){
  accuracy[l] <- confusionMatrix(test$raw$clase,ridge.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
accuracy_ridge <- accuracy
accuracy_ridge
```

```
      raw    zscore        l2     media   mediana   min_max
0.5919283 0.5941704 0.5941704 0.5941704 0.5941704 0.5941704
```

LASSO

```r
# Esta línea establece la semilla para la generación de números aleatorios.
# Al establecer una semilla, se asegura que los resultados sean reproducibles
# Al ejecutar el mismo código con la misma semilla, se obtendrán los mismos resultados.
set.seed(123456789)

# La función lasso sigue una estructura similar a la función ridge.
# Se utiliza alpha = 1 para especificar el uso del método Lasso en lugar de Ridge.
# Los parámetros (clase ~ ., data, trControl, tuneGrid y trace) se mantienen similares.
lasso <- function(x) train(clase ~ ., data = x, method = "glmnet",
```

```r
                              trControl = trControl,tuneGrid = expand.grid
                              (alpha=1,lambda=lambda_seq), trace = F)

# Se aplica la función lasso a cada elemento de la lista entrenamiento utilizando lapply.
# El resultado es una lista llamada lasso.lista que contiene los modelos entrenados.
# Se crea un vector llamado lasso.pred de tipo lista con una longitud igual a la
#longitud de datos.lista.
# Este vector se utilizará para almacenar los resultados de las predicciones
#realizadas por los modelos Lasso.
lasso.lista <- lapply(entrenamiento,lasso)
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
```

```
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```r
lasso.pred <- vector("list",length = length(datos.lista))
```

```r
# Se utiliza la función predict para obtener las predicciones.
# Las predicciones se almacenan en el vector lasso.pred en la posición correspondiente.
# Se crea un vector llamado accuracy de tipo numérico con una longitud igual a la
#de datos.lista.
for(l in 1:length( datos.lista)){
  lasso.pred[[l]] <- predict(lasso.lista[[l]],test[[l]])
}
names(lasso.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))

# Se utiliza un bucle for para calcular la exactitud de las predicciones realizadas
#por los modelos datos.lista en los datos de prueba test.
# El valor de exactitud se guarda en el vector accuracy en la posición correspondiente.
# Finalmente, se crea un vector accuracy_lasso que almacena los valores de exactitud.
for(l in 1:length(datos.lista)){
  accuracy[l] <- confusionMatrix(test$raw$clase,lasso.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
accuracy_lasso <- accuracy
accuracy_lasso
```

```
      raw    zscore        l2     media   mediana   min_max
0.5919283 0.5919283 0.5919283 0.5919283 0.5919283 0.5919283
```

KNN

```r
# Esta línea establece la semilla para la generación de números aleatorios.
# Al establecer una semilla, se asegura que los resultados sean reproducibles.
# Al ejecutar el mismo código con la misma semilla, se obtendrán los mismos resultados.
set.seed(123456789)

# La función myfnknn usa la función train del paquete caret para entrenar un modelo k-NN.
# El parámetro method se establece en "knn" para indicar el algoritmo k-NN.
# El parámetro tuneLength indica el número de valores que se probarán en la búsqueda.
# En este caso, se establece en 20, es decir, se probarán 20 valores diferentes para
#el parámetro k en el modelo k-NN.
knn <- function(x) train(clase ~ ., data = x, method = "knn",
                         trControl = trControl, tuneLength = 20)

# Se aplica la función knn a cada elemento de la lista entrenamiento utilizando lapply.
# Resulta una lista que contiene los modelos entrenados utilizando el algoritmo k-NN.
```

```r
# A continuación, se crea un vector llamado KNN.pred de tipo lista con una longitud
#igual a la longitud de datos.lista.
# Este vector almacenará los resultados de predicciones realizadas por los modelos k-NN.
knn.lista <- lapply(entrenamiento,knn)
knn.pred <- vector("list",length = length(datos.lista))

# Se utiliza un bucle for para realizar las predicciones las predicciones se almacenan
#en el vector knn.pred en la posición correspondiente.
for(l in 1:length( datos.lista)){
  knn.pred[[l]] <- predict(knn.lista[[l]],test[[l]])
}

# Se asignan los nombres de datos.lista a los elementos de knn.pred utilizando la
#función names(knn.pred) <- names(datos.lista).
# Se crea un vector llamado accuracy de tipo numérico con una longitud igual a la
#longitud de datos.lista.
names(knn.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))

# Se utiliza un bucle for para calcular la exactitud de las predicciones realizadas
#por los modelos knn.pred en los datos de prueba test.
# Para cada modelo y conjunto de datos, se calcula la matriz de confusión y
#se extrae el valor de la exactitud general.
# El valor de exactitud se guarda en el vector accuracy en la posición correspondiente.
# Finalmente, se asignan los nombres de datos.lista a los elementos de accuracy.
# Además, se crea un vector llamado accuracy_knn que almacena los valores de exactitud.
for(l in 1:length(datos.lista)){
  accuracy[l] <- confusionMatrix(test$raw$clase,knn.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
accuracy_knn <- accuracy
accuracy_knn
```

```
      raw    zscore        l2     media   mediana   min_max
0.5852018 0.5717489 0.5986547 0.5941704 0.5874439 0.6031390
```

NAIVE BAYES

```r
#Esta línea establece la semilla para la generación de números aleatorios.
# Al establecer una semilla, se asegura que los resultados sean reproducibles.
# Al ejecutar el mismo código con la misma semilla, se obtendrán los mismos resultados.
```

```r
set.seed(123456789)

# Se aplica la función naive a cada elemento de la lista entrenamiento utilizando lapply.
# Resulta en una lista llamada naive.lista.
# Se crea un vector llamado naive.pred de tipo lista con una longitud igual a la
#longitud de datos.lista. Este vector
#se utilizará para almacenar los resultados de las predicciones realizadas por los
#modelos de Naive Bayes.
naive <- function(x) train(clase ~ ., data = x, method = "naive_bayes",
                           trControl = trControl, tuneLength = 20)
naive.lista <- lapply(entrenamiento,naive)
naive.pred <- vector("list",length = length(datos.lista))

# Se utiliza un bucle for para realizar las predicciones.
# Para cada modelo y conjunto de datos, se utiliza "predict" para obtener las predicciones
# Las predicciones se almacenan en el vector naive.pred en la posición correspondiente.
for(l in 1:length( datos.lista)){
  naive.pred[[l]] <- predict(naive.lista[[l]],test[[l]])
}

# En el código adicional que has proporcionado, se asignan los nombres de datos.lista
#a los elementos de naive.pred
# Se crea un vector llamado accuracy de tipo numérico con una longitud igual a la
#longitud de datos.lista.
names(naive.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))

# A continuación, se utiliza un bucle for para calcular la exactitud de las predicciones
#realizadas por los modelos naive.pred.
# Se utiliza la función confusionMatrix y luego se extrae el valor de la exactitud general
# El valor de exactitud se guarda en el vector accuracy en la posición correspondiente.
# Finalmente, se asignan los nombres de datos.lista a los elementos de accuracy.
# Además, se crea el vector accuracy_bayes que almacena los valores de exactitud.
for(l in 1:length(datos.lista)){
  accuracy[l] <- confusionMatrix(test$raw$clase,naive.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
accuracy_bayes <- accuracy
accuracy_bayes
```

```
    raw    zscore        l2     media   mediana   min_max
```

```
0.4080717 0.5336323 0.4596413 0.5336323 0.4125561 0.4125561
```

MATRIZ ACCURACY

```r
# Se genera una matriz de 5 Filas x 6 Columnas.
# La matriz se llena con valores por defecto.
matrizaccuracy <- matrix(nrow = 5, ncol = 6)

# Se asigna los valores a la matriz creada anteriormente.
matrizaccuracy[1, ] <- accuracy_logis
matrizaccuracy[2, ] <- accuracy_ridge
matrizaccuracy[3, ] <- accuracy_lasso
matrizaccuracy[4, ] <- accuracy_knn
matrizaccuracy[5, ] <- accuracy_bayes

# Se define el nombre para las variables de cada columna y cada fila.
filas <- c("LOGISTICA", "RIDGE", "LASSO", "KNN", "BAYES")
columnas <- c("raw", "zscore", "l2", "media", "mediana", "min_max")

# Se asignan los nombres definidos en el paso anterior
rownames(matrizaccuracy) <- filas
colnames(matrizaccuracy) <- columnas

# Se muestra la matriz desarrollada completamente
print(matrizaccuracy)
```

```
               raw    zscore        l2     media   mediana   min_max
LOGISTICA 0.5919283 0.5919283 0.5964126 0.5919283 0.5941704 0.5919283
RIDGE     0.5919283 0.5941704 0.5941704 0.5941704 0.5941704 0.5941704
LASSO     0.5919283 0.5919283 0.5919283 0.5919283 0.5919283 0.5919283
KNN       0.5852018 0.5717489 0.5986547 0.5941704 0.5874439 0.6031390
BAYES     0.4080717 0.5336323 0.4596413 0.5336323 0.4125561 0.4125561
```

```r
# Encontrar el número máximo y su posición en la matriz.
maxi <- which.max(matrizaccuracy)
fila_max <- row(matrizaccuracy)[maxi]
columna_max <- col(matrizaccuracy)[maxi]

# Encontrar el nombre de la fila y columna del numero maximo.
nombre_fila <- rownames(matrizaccuracy)[fila_max]
nombre_columna <- colnames(matrizaccuracy)[columna_max]
```

```r
# Imprimir el número máximo y su posición.
cat("El procentaje mayor de los cinco metodos trabajados es: ", matrizaccuracy[fila_max, c
```

```
El procentaje mayor de los cinco metodos trabajados es:  0.603139 %
```

```r
cat("Con el metodo", nombre_fila, "y la transformacion", nombre_columna, "\n")
```

```
Con el metodo KNN y la transformacion min_max
```