```
conjuntos.descriptores <- function (archivo=" S-M training set.csv ",
conjuntos=1000,descrip.conjunto=200, semilla=125) {

is.installed <- function(mypkg) { is.element(mypkg, installed.packages()[,1]) }


if (is.installed("data.table") == FALSE) {install.packages("data.table")}


if (is.installed("caret") == FALSE) {install.packages("caret")}


library(data.table)


library(caret)


set.seed(semilla)


df <- as.data.frame(fread(input = archivo, check.names = TRUE))


df <- df[ , apply(df, 2, function(x) !any(is.na(x)))]


df <- df[ , -nearZeroVar(df)]


n <- conjuntos


j <- c(1:ncol(df))


k <- j[-c(1,2)]


dflist <-list()


for(i in 1:n) {
```

```
        dflist[[i]] <- df[ , c(2,sample(k,descrip.conjunto)) ]

        }

dflist

 }

lista.conjuntos2 <- conjuntos.descriptores(archivo="S-M training set.csv")




forward.stepwise.testF.lm <- function(tabla.conjunto , punto.corte=0.05 , steps=6 ,
punto.corte.vif = 2) {

is.installed <- function(mypkg) { is.element(mypkg, installed.packages()[,1]) }

if ( is.installed("car") == FALSE) {install.packages("car")}
library(car)

df <- as.data.frame(tabla.conjunto)

null <- lm(clase ~1,data = df, na.action = na.omit)

full <- lm(clase~.,data = df, na.action = na.omit)

repeat {

paso <- add1(null,scope=formula(full),test="F", na.action = na.omit)
```

```
menor.probabilidad <- which.min(paso$`Pr(>F)`)

elegido <- noquote(row.names(paso)[menor.probabilidad])

formulacion <- paste(".~. +", elegido)

if ( paso$`Pr(>F)`[menor.probabilidad] >= punto.corte ) {

break
}

null <- update(null, formula. = formulacion)

if (length(null$coefficients) == 3){

test.vif <- data.frame(vif(null))

if (test.vif[nrow(test.vif),1] >= punto.corte.vif) {

formulacion2 <- paste(".~. -",paste(noquote(rownames(test.vif)[nrow(test.vif)]) ,collapse =
" - "))

null<-update(null, formula. = formulacion2)

full <- update(full, formula. = formulacion2)

}
}

if (length(null$coefficients) > 3){
```

```
test.vif <- data.frame(vif(null))

if (any(test.vif[,1] >= punto.corte.vif)) {

formulacion2 <- paste(".~. -",paste(noquote(rownames(test.vif)[nrow(test.vif)]) ,collapse =
" - "))

null<-update(null, formula. = formulacion2)

full <- update(full, formula. = formulacion2)

}
}

test <- drop1(null, test = "F")

if (any(test$` Pr(>F)` >= punto.corte, na.rm = TRUE)) {

formulacion3<- paste(".~. -",paste(noquote(rownames(test)[which(test$` Pr(>F)` >=
punto.corte)]) ,collapse = " - "))

null<-update(null, formula. = formulacion3)

full<-update(full, formula. = formulacion3)

}

if (length(null$coefficients) >= steps + 1){

break
}
```

```
if (length(full$coefficients) == length(null$coefficients)){

break
}

}

result <-null

result

}

lista.modelos<- lapply(lista.conjuntos2, forward.stepwise.testF.lm,
punto.corte=0.05,steps=6 ,punto.corte.vif = 2)



AUC.curvas.ROC <- function ( x = lista.modelos ) {

is.installed <- function(mypkg) { is.element(mypkg, installed.packages()[,1]) }

if ( is.installed("pROC") == FALSE) {install.packages("pROC")}

library(pROC)

lista.predicciones <- lapply( x , predict)

lista.auc.roc.numeric<-list()

for (i in 1:length(lista.predicciones)) {
```

```
p <-lista.conjuntos2[[1]]

q<-factor(p[,"clase"])

lista.auc.roc.numeric[[i]]<- roc(predictor = lista.predicciones[[i]],response = q, direction =
"<" , auc = TRUE)$auc[[1]]

lista.auc.roc.numeric

}

models.ranking.by.auc.roc<-order(unlist(lista.auc.roc.numeric), decreasing = TRUE)

matrix.ranking <- cbind(modelo = models.ranking.by.auc.roc , AUC =
lista.auc.roc.numeric[models.ranking.by.auc.roc])

tabla.ranking <-as.data.frame(matrix.ranking)

tabla.ranking<-data.frame(apply(X=tabla.ranking, MARGIN = 2, FUN = unlist))

tabla.ranking

}

tabla.AUC.ordenadas <- AUC.curvas.ROC( x= lista.modelos)




k.fold.CV.lm <- function ( tabla.conjunto  , punto.corte.optimizado = 0.5 , folds = 10 ,
repeticiones = 1  , punto.corte = 0.05 , steps = 6 , punto.corte.vif = 2) {

is.installed <- function(mypkg) { is.element(mypkg, installed.packages()[,1]) }
```

```
if (is.installed("data.table") == FALSE) {install.packages("data.table")}

if (is.installed("plyr") == FALSE) {install.packages("plyr")}

if (is.installed("caret") == FALSE) {install.packages("caret")}

library(data.table)

library(plyr)

library(caret)

df <- tabla.conjunto

pb <- txtProgressBar(min = 0, max = repeticiones, style = 3)

acc <- matrix(nrow = folds , ncol = repeticiones)

for(j in 1:repeticiones) {

set.seed(j)

index <- createFolds( y = factor(df$clase) , k = folds , list = TRUE , returnTrain = FALSE)

for(i in 1:folds) {

train <- df[-index[[i]], ]

test <- df[index[[i]], ]
```

```
model <- forward.stepwise.testF.lm( tabla.conjunto = train  , punto.corte = punto.corte , steps = steps , punto.corte.vif = punto.corte.vif )

results_prob <- predict(object = model, newdata = test,type='response')

results <- ifelse(test = results_prob > punto.corte.optimizado, yes = 1 ,no = 0 )

answers <- test$clase

misClasificError <- mean(answers != results)

acc[i,j] <- 1-misClasificError

}

setTxtProgressBar(pb, j)

}

porcentaje.buenas.clasificaciones <- mean(colMeans(acc))

sd <- sd(as.vector(acc))

close(pb)

resultado <- list ("Accuracy total" ,   porcentaje.buenas.clasificaciones , "desviacion estandar" , sd , "Accuracy de cada fold" , as.vector(acc) )

resultado

}
```

```
lista.porcentajes.buenas.clasificaciones.k.fold.CV.lm <- lapply(lista.conjuntos2 ,
k.fold.CV.lm , punto.corte.optimizado = 0.5 , folds = 10 , repeticiones = 5 ,
punto.corte=0.05 , steps=6 , punto.corte.vif = 2)


tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm <- matrix(ncol =
length(lista.porcentajes.buenas.clasificaciones.k.fold.CV.lm[[1]][[6]]) + 3  , nrow =
length(lista.porcentajes.buenas.clasificaciones.k.fold.CV.lm))


for ( i in 1:length(lista.porcentajes.buenas.clasificaciones.k.fold.CV.lm)) {


tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm[i,1] <- i


tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm[i,2] <-
lista.porcentajes.buenas.clasificaciones.k.fold.CV.lm[[i]][[2]]


tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm[i,3] <-
lista.porcentajes.buenas.clasificaciones.k.fold.CV.lm[[i]][[4]]


tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm[i,4:ncol(tabla.porcentajes.buenas.
clasificaciones.k.fold.CV.lm)] <-
lista.porcentajes.buenas.clasificaciones.k.fold.CV.lm[[i]][[6]]


}


tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm <-
as.data.frame(tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm)


folds <- sprintf("Accuracy fold
%d",seq(1:length(lista.porcentajes.buenas.clasificaciones.k.fold.CV.lm[[1]][[6]])))


colnames(tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm) <- c("modelo" ,
"Accuracy" , "estandar deviation" , folds)
```

```
tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm <-
tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm[order(tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm$Accuracy , decreasing = TRUE),]

View(tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm)

library(openxlsx)

write.xlsx(x= tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm, file= "Resultado k fold CV version 2 - lm.xlsx" , colNames= TRUE, keepNA=TRUE)




AUC.curvas.ROC.test.set.lm <- function ( test.set= " S-M test set.csv ", modelos = lista.modelos ) {

is.installed <- function(mypkg) { is.element(mypkg, installed.packages()[,1])

if ( is.installed("pROC") == FALSE) {install.packages("pROC")}

if (is.installed("data.table") == FALSE) {install.packages("data.table")}

library(data.table)

library(pROC)

df.test.set <- as.data.frame(fread(input = test.set, check.names = TRUE))

lista.predicciones.test <- lapply(X = modelos , FUN = predict, newdata = df.test.set, type = "response" )

lista.auc.roc.numeric <- list()
```

```
for (i in 1:length(lista.predicciones.test)) {

q<-factor(df.test.set[,"clase"])

lista.auc.roc.numeric[[i]]<- roc(predictor = lista.predicciones.test[[i]] , response = q , direction = "<" , auc = TRUE)$auc[[1]]

lista.auc.roc.numeric

}

models.ranking.by.auc.roc<-order(unlist(lista.auc.roc.numeric), decreasing = TRUE)

matrix.ranking <- cbind(modelo = models.ranking.by.auc.roc,AUC = lista.auc.roc.numeric[models.ranking.by.auc.roc])

tabla.ranking <-as.data.frame(matrix.ranking)

tabla.ranking<-data.frame(apply(X=tabla.ranking, MARGIN = 2, FUN = unlist))

tabla.ranking

}

tabla.AUC.ordenadas.test.set <- AUC.curvas.ROC.test.set.lm( test.set= " S-M test set.csv ", modelos = lista.modelos)

save(lista.modelos,tabla.AUC.ordenadas, tabla.AUC.ordenadas.test.set , tabla.porcentajes.buenas.clasificaciones.k.fold.CV.lm, file = "modelos_lineales.RData")

load("modelos_lineales.RData")
```