```
# TFM: Análisis predictivo de incidentes navales en EEUU, 2002 - 2015
# Anexo 7. Funciones de R
# Oscar Antón, 2023, UNED



# Funciones basadas en la autoevaluación de Minería de Datos II: FUNCIONES_MODULO_7.R


#-------------------------------------------------------------#

# 1. Guardado de archivos

# Función para exportar archivo y descripción
loggedsave <- function (archivo, destino) {

  # Verifica si la carpeta local existe, y si no, la crea
  if (!dir.exists(destino)) {
    dir.create(destino, recursive = TRUE)
  }

  # Exportación de datos
  saveRDS(archivo, paste0(destino, "/", deparse(substitute(archivo)), ".rds"))

  # Log del proceso y descripción
  sink(paste0(destino, "/", "dataframesDescription.txt"), append = TRUE)
  cat(strftime(Sys.time(), format = "%Y-%m-%d | %H:%M:%S"), " | ",
      "Nombre: ", deparse(substitute(archivo)), " | ",
      "Dimensiones: ", dim(archivo), " | ",
      "Peso en memoria: ", object.size(archivo) / (1024^2) , " Megas ", "|",
      "Tiempo computado : ", toc(), "|",
      "\n")
  sink()
}

#-------------------------------------------------------------#

# 2. Representación de mapa con leaflet




#-------------------------------------------------------------#

# 3. Reducción de variabilidad en variables factoriales

lump_factorials <- function(variable) {
  fct_lump(variable, prop = 0.008, other_level = "other value")
}

#-------------------------------------------------------------#

# 4. Modelos machine learning


# Funciones que se utilizan para la salida de los resultados


grafico_metricas <- function(modelo){

  g1<- plot(modelo, main="Métrica ROC")

  g2<- plot(modelo, metric="Accuracy", main="Métrica Accuracy")
  g3<- plot(modelo, metric="Kappa", main="Métrica Kappa")

  print(g1)
  grid.arrange(g2, g3, ncol=2)

}
```

```
# Resultados


resultados <-function(modelo, texto){
  resultados <- modelo$results %>% arrange(desc(modelo$metric))
  head(resultados, 12) %>%
    knitr::kable("html",  caption =paste("RESULTADOS DEL MODELO", texto))%>%
    kableExtra::kable_styling(bootstrap_options = c("striped", "hover"),
                              full_width = F, font_size = 14)
}



# Mejor modelo


mejor_modelo <-function(modelo){
  print("El mejor módelo es el que muestra los siguientes hiperparámetros:")

  modelo$bestTune%>%
    knitr::kable("html")%>%
    kableExtra::kable_styling(bootstrap_options = c("striped", "hover"),
                              full_width = F, font_size = 12)


}

# Curva ROC

curvas_ROC<- function(modelo, texto, train, test){

  # Establezco el segundo nivel(si responde a la oferta), como referencia
  clase <- modelo$levels[2]

  # Valores predichos y ROC para entrenamiento y validación
  pred_train <- predict(modelo, train , type="prob")
  curvaROC_train <- roc(train[ , c(length(train))],
                        pred_train[,clase])

  pred_test <- predict(modelo, test, type="prob")
  curvaROC_test <- roc(test[ , c(length(test))],pred_test[,clase])

  # Salidas

  plot(curvaROC_test,axes=TRUE, col="navyblue",
       auc.polygon=TRUE,grid.col=c("orangered", "aliceblue"),
       max.auc.polygon=TRUE, grid=c(0.1, 0.2),
       main=paste("Curvas ROC del modelo",texto),
       print.auc=TRUE, auc.polygon.col="aquamarine")
  plot(curvaROC_train, col="orangered", add=TRUE)
  legend("bottomright", legend = c("Test", "Validación"), col = c("navyblue",
"orangered"), lwd = 2)


  print(paste(c("ROC del modelo con el fichero de test:"), auc(curvaROC_test)))

}



# Importancia de las variables

importancia_var<-function(modelo, texto){


  VarImp <- varImp(modelo)
```

```
  ggplot(VarImp, top = 10)+
    geom_bar(stat = "identity", fill = "#00a99d") +
    geom_text(aes(label=round(Importance,2)), color = "#ffffff", hjust=1.2, size=3) +
    labs(title="Importancia de las variables",
         subtitle = paste("Modelo", texto), x="Variables", y="Importancia") +
    theme_minimal()

}


# Importancia media de las variables para modelos con multiclase

importancia_var_overall <- function(modelo, texto){

  importancia_overall <- rowMeans(varImp(modelo)$importance)
  df <- data.frame(Variables = substr(names(importancia_overall),1,20) , Importance =
as.numeric(importancia_overall)) %>% head(10)

  ggplot(df, aes(x = reorder(Variables, Importance), y = Importance)) +
    geom_bar(stat = "identity", fill = "#00a99d") +
    geom_text(aes(label=round(Importance,2)), color = "#ffffff", hjust=1.2, size=3) +
    scale_y_continuous(limits = c(0, 100)) +
    labs(title="Importancia de las variables", subtitle = paste("Modelo", texto),
x="Variables", y="Importancia") +
    theme_minimal() +
    coord_flip()
}



# Validación - Predicción

validation <- function(modelo, texto, train, test){
  pred_train  <- predict(modelo, train, type="raw")
  pred_test  <- predict(modelo, test, type="raw")

  print(paste("Modelo", texto, "- Tabla de confusión para los datos de entrenamiento"))
  print(confusionMatrix(data = pred_train, reference = train$y))

  print(paste("Modelo", texto, "- Tabla de confusión para los datos de validación"))
  confusionMatrix(data = pred_test, reference = test$y)
}


# Resumen del modelo, Mostrando datos del entrenamiento y de la validacion

resumen <-  function(modelo,train, test){

  clase <- modelo$levels[2]

  # Entrenamiento

  pred_train_prob <- predict(modelo, train, type="prob")
  curvaROC_train <- roc(train[ , c(length(train))],pred_train_prob[,clase])
  AUC_train <- round(auc(curvaROC_train),digits=3)

  pred_train <- predict(modelo, train, type="raw")
  confusion_train <- confusionMatrix(data = pred_train, reference = train$y)
  Accuracy_train=round(c(confusion_train[["overall"]][["Accuracy"]]), digits=3)
  AciertosClSI_train=round(c(confusion_train[["byClass"]][["Pos Pred Value"]]),
                           digits=3)
  AciertosClNO_train=round(c(confusion_train [["byClass"]][["Neg Pred Value"]]),
                           digits=3)
  Kappa_train=round(c(confusion_train[["overall"]][["Kappa"]]), digits=3)
  Sensitivity_train=round(c(confusion_train[["byClass"]][["Sensitivity"]]),
                          digits=3)
  Specificity_train=round(c(confusion_train[["byClass"]][["Specificity"]]),
                          digits=3)
```

```
  train <- c(AUC_train, Accuracy_train, AciertosClSI_train, AciertosClNO_train,
             Kappa_train, Sensitivity_train, Specificity_train)

  # Validación
  pred_test_prob <- predict(modelo, test, type="prob")
  curvaROC_test <- roc(test[ , c(length(test))],pred_test_prob[,clase])
  AUC_test <- round(auc(curvaROC_test), digits=3)

  pred_test <- predict(modelo, test, type="raw")
  confusion_test <- confusionMatrix(data = pred_test, reference = test$y)
  Accuracy_test=round(c(confusion_test[["overall"]][["Accuracy"]]), digits=3)
  AciertosClSI_test=round(c(confusion_test[["byClass"]][["Pos Pred Value"]]),
                          digits=3)
  AciertosClNO_test=round(c(confusion_test [["byClass"]][["Neg Pred Value"]]),
                          digits=3)
  Kappa_test=round(c(confusion_test[["overall"]][["Kappa"]]), digits=3)
  Sensitivity_test=round(c(confusion_test[["byClass"]][["Sensitivity"]]),
                         digits=3)
  Specificity_test=round(c(confusion_test[["byClass"]][["Specificity"]]),
                         digits=3)

  test <- c(AUC_test, Accuracy_test, AciertosClSI_test, AciertosClNO_test,
            Kappa_test, Sensitivity_test, Specificity_test)

  resumen <- rbind(train,test)
  colnames(resumen) <- c("AUC", "Accuracy", "Aciertos Clase SI",
                         "Aciertos Clase NO","Kappa", "Sensitivity",
                         "Specificity")
  rownames(resumen) <- c("Datos Entrenamiento", "Datos Validación")
  resumen
}


# Resumen si la variable objetivo es multiclase (más de 2 niveles)
resumen_multiclass <-  function(modelo,train, test){

  # Entrenamiento

  pred_train_prob <- predict(modelo, train, type="prob")
  curvaROC_train <- multiclass.roc(train[ , c(length(train))],pred_train_prob)
  AUC_train <- round(auc(curvaROC_train),digits=3)

  pred_train <- predict(modelo, train, type="raw")
  confusion_train <- confusionMatrix(data = pred_train, reference = train$y)
  Accuracy_train=round(c(confusion_train[["overall"]][["Accuracy"]]), digits=3)

  Kappa_train=round(c(confusion_train[["overall"]][["Kappa"]]), digits=3)

  # Puede ser la media si el conjunto de datos esta balanceado
  Sensitivity_train=round(mean(confusion_train[["byClass"]][ ,
"Sensitivity"]),digits=3)
  Specificity_train=round(mean(confusion_train[["byClass"]][ ,
"Specificity"]),digits=3)

  train <- c(AUC_train, Accuracy_train, Kappa_train, Sensitivity_train,
Specificity_train)

  # Validación
  pred_test_prob <- predict(modelo, test, type="prob")
  curvaROC_test <- multiclass.roc(test[ , c(length(test))],pred_test_prob)
  AUC_test <- round(auc(curvaROC_test), digits=3)

  pred_test <- predict(modelo, test, type="raw")
  confusion_test <- confusionMatrix(data = pred_test, reference = test$y)
  Accuracy_test=round(c(confusion_test[["overall"]][["Accuracy"]]), digits=3)

  Kappa_test=round(c(confusion_test[["overall"]][["Kappa"]]), digits=3)

  # Puede ser la media si el conjunto de datos esta balanceado
```

```
  Sensitivity_test=round(mean(confusion_test[["byClass"]][ , "Sensitivity"]), digits=3)
  Specificity_test=round(mean(confusion_test[["byClass"]][ , "Specificity"]), digits=3)

  test <- c(AUC_test, Accuracy_test, Kappa_test, Sensitivity_test, Specificity_test)

  resumen <- rbind(train,test)
  colnames(resumen) <- c("AUC", "Accuracy", "Kappa", "Sensitivity", "Specificity")
  rownames(resumen) <- c("Datos Entrenamiento", "Datos Validación")
  resumen
}


# Matriz de confusión para modelos entrenados con Keras TensorFlow
keras_confusion <- function(model, x_test, y_test){
  # Calculamos la matriz de predicciones de los datos de test
  predicciones <- predict(model, as.matrix(x_test))

  # Pasamos los datos a lista
  Y_test <- apply(y_test, 1, function(row) which(row == 1))
  # Pasamos los datos a una lista con el máximo de cada linea
  predicciones <- apply(predicciones, 1, which.max)

  # Creamos la matriz de confusión como un dataframe a partir de una tabla
  matriz_conf <- as.data.frame(table(predicciones, Y_test))

  # Visualizado con un gráfico tiles de ggplot
  ggplot(matriz_conf, aes(x = predicciones, y = Y_test, fill = Freq)) +
    geom_tile(colour = "white") +
    geom_text(aes(label=Freq), colour = "white") +
    scale_fill_continuous(trans = 'reverse') +
    labs(title = paste(deparse(substitute(model)), ": Matriz de confusión"), x =
"Predicción", y = "Referencia")
}



# Resumen si la variable objetivo es multiclase (más de 2 niveles) modelos entrenados
con Keras TensorFlow
keras_resumen_multiclass <-  function(modelo, x_train, x_test, y_train, y_test){
  # Entrenamiento
  predicciones <- predict(modelo, as.matrix(x_train))
  predicciones <- apply(predicciones, 1, which.max)
  Y_train <- max.col(y_train)

  curvaROC_train <- multiclass.roc(Y_train, predicciones)
  AUC_train <- round(auc(curvaROC_train),digits=3)

  confusion_train <- confusionMatrix(as.factor(Y_train), as.factor(predicciones))

  Accuracy_train <- round(c(confusion_train[["overall"]][["Accuracy"]]), digits=3)

  Kappa_train <- round(c(confusion_train[["overall"]][["Kappa"]]), digits=3)

  Sensitivity_train <- round(mean(confusion_train[["byClass"]][ , "Sensitivity"]),
digits=3)
  Specificity_train <- round(mean(confusion_train[["byClass"]][ , "Specificity"]),
digits=3)

  train <- c(AUC_train, Accuracy_train, Kappa_train, Sensitivity_train,
Specificity_train)

  # Validación
  predicciones <- predict(modelo, as.matrix(x_test))
  predicciones <- apply(predicciones, 1, which.max)
  Y_test <- max.col(y_test)

  curvaROC_test <- multiclass.roc(Y_test, predicciones)
  AUC_test <- round(auc(curvaROC_test),digits=3)
```

```
  confusion_test <- confusionMatrix(as.factor(Y_test), as.factor(predicciones))

  Accuracy_test <- round(c(confusion_test[["overall"]][["Accuracy"]]), digits=3)

  Kappa_test <- round(c(confusion_test[["overall"]][["Kappa"]]), digits=3)

  Sensitivity_test <- round(mean(confusion_test[["byClass"]][ , "Sensitivity"]),
digits=3)
  Specificity_test <- round(mean(confusion_test[["byClass"]][ , "Specificity"]),
digits=3)

  test <- c(AUC_test, Accuracy_test, Kappa_test, Sensitivity_test, Specificity_test)

  resumen <- rbind(train,test)
  colnames(resumen) <- c("AUC", "Accuracy", "Kappa", "Sensitivity", "Specificity")
  rownames(resumen) <- c("Datos Entrenamiento", "Datos Validación")
  resumen
}
```