El objetivo de esta segunda parte consiste en participar del desafío de Kaggle para la predicción de la inclinación peligrosa del arbolado público de Mendoza.
https://www.kaggle.com/competitions/arbolado-publico-mendoza-2024

PARTE A)

Antes de realizar un primer envío se deben realizar las siguiente actividades

1. Bajar el archivo **arbolado-mza-dataset.csv**, el mismo se encuentra en formato CSV (comma separated values).
    a. Seleccionar de manera uniformemente aleatoria el 20% del conjunto de datos y crear un nuevo archivo con el nombre de **arbolado-mendoza-dataset-validation.csv** y el 80% restante con el nombre de **arbolado-mendoza-dataset-train.csv**
2. A partir del archivo **arbolado-mendoza-dataset-train.csv** responder las siguientes preguntas:
    a. Cual es la distribución de las clase `inclinacion_peligrosa?`
    b. ¿Se puede considerar alguna sección más peligrosa que otra?
    c. ¿Se puede considerar alguna especie más peligrosa que otra?

    IMPORTANTE: para responder cada una de estas preguntas se deberá generar una visualización/gráfico que justifique la respuesta.

3. A partir del archivo **arbolado-mendoza-dataset-train.csv,**
    a. ~~Generar un histograma de frecuencia para la variable: `altura`. Probar con diferentes números de bins~~. #no va
    b. Generar un histograma de frecuencia para la variable `circ_tronco_cm`. Probar con diferentes números de bins.
    c. Repetir el punto b) pero separando por la clase de la variable `inclinación_peligrosa?`
    d. Crear una nueva variable categórica de nombre `circ_tronco_cm_cat` a partir `circ_tronco_cm`, en donde puedan asignarse solo 4 posibles valores [muy `alto`, `alto`, `medio`, `bajo`]. Utilizar la información del punto b. para

seleccionar los puntos de corte para cada categoría. Guardar el nuevo dataframe bajo el nombre de **arbolado-mendoza-dataset-circ_tronco_cm-train.csv**

4. Clasificador aleatorio:
   a. Implementar una función que dado un conjunto de observaciones (data.frame) genere una nueva columna de nombre `prediction_prob` con un valor aleatorio entre 0 y 1.
   b. Implementar una función de nombre `random_classifier,` que reciba como parámetro el dataframe generado con anterioridad y a partir de la columna `predictions_prob` genere una nueva columna `prediction_class` bajo el siguiente criterio:

      `If predictions_prob > 0.5 then prediction_class=1 else prediction_class=0`

      La función deberá devolver el dataframe original junto a la nueva columna generada.

   c. Cargar el archivo **arbolado-mendoza-dataset-validation.csv** como un data.frame y aplicarle la función `random_classifier`
   d. A partir de la columna recientemente generada y la columna con la clase (inclinación peligrosa) calcular utilizando lenguaje R (dplyr) el número de:
      i. Número de árboles CON inclinación peligrosa que fueron correctamente predicho como peligrosos por el modelo/algoritmo. (True Positive)
      ii. Número de árboles SIN inclinación peligrosa que fueron correctamente predicho como no peligrosos por el modelo. (True Negative)
      iii. Número de árboles SIN inclinación peligrosa que fueron incorrectamente predicho como peligrosos según el modelo. (False Positives)
      iv. Número de árboles CON inclinación peligrosa que fueron incorrectamente predicho como no peligrosos según el modelo. (False Negatives)

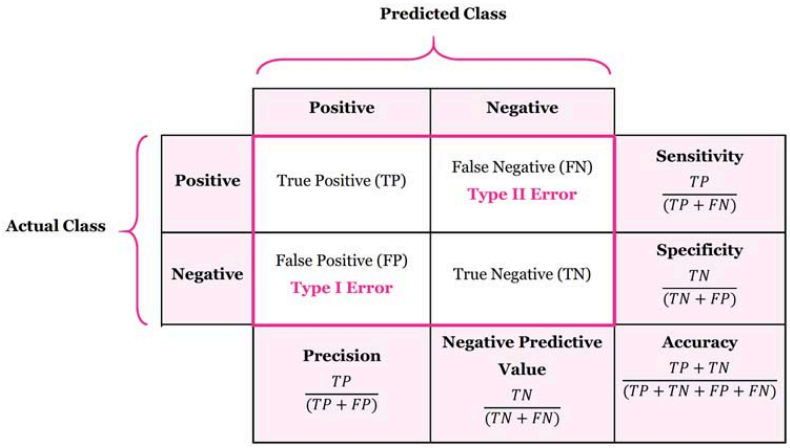
      El resultado es una tabla que se conoce como matriz de confusión.

n = 165	Predicted: No	Predicted: Yes
Actual: No	50	10
Actual: Yes	5	100

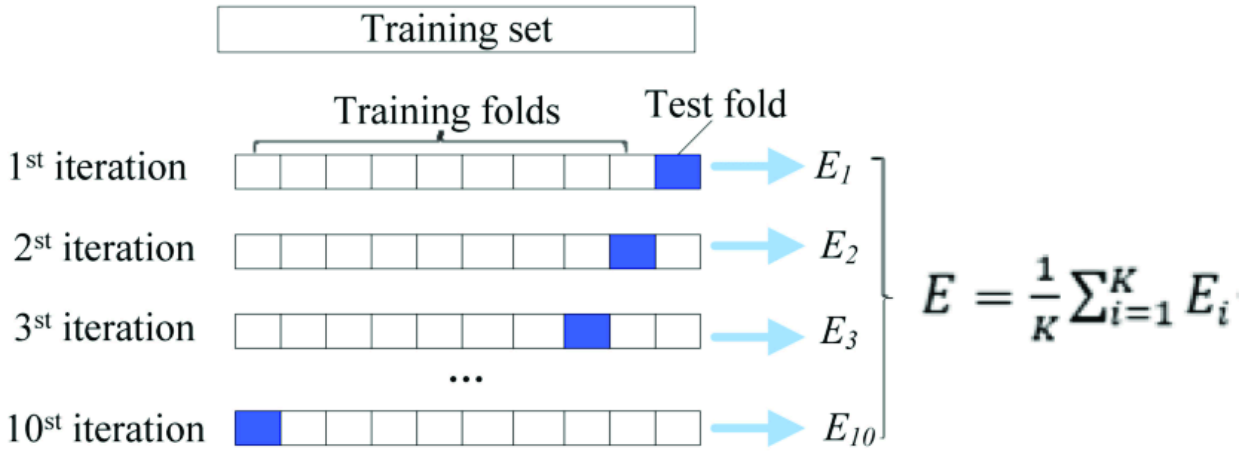
5. Clasificador por clase mayoritaria:
   a. Implementar una función de nombre `biggerclass_classifier`, que reciba como parámetro el dataframe generado con anterioridad y genere una nueva columna de nombre `prediction_class` en donde se asigne siempre de la clase mayoritaria

      La función deberá devolver el dataframe original junto a la nueva columna generada.
   b. Repetir los puntos 4.c y 4.d pero aplicando la nueva función `biggerclass_classifier`

6. A partir de una matriz de confusión es posible calcular distintas métricas que nos permiten determinar la calidad del modelo de clasificación.
   Utilizar la siguiente imagen como guía crear funciones para calcular: **Accuracy, Precision, Sensitivity, Specificity** y calcularlas para las matrices de confusión generadas en los puntos 4 y 5.



7. Validación cruzada (Cross validation) (k-folds):
   La validación cruzada es una técnica para estimar el error de generalización de un algoritmo/modelo de machine learning. La técnica consiste en (previo realizar una mezcla aleatoria) separar el conjunto de datos en k partes (normalmente denominadas ***folds***). Luego en la primera iteración se utilizan k-1 partes para entrenar E_1 y se utiliza la restante para test. El proceso se repite por k iteraciones utilizando en cada una diferentes conjuntos de entrenamiento y test. (Ver figura)



a. Crear una función de nombre `create_folds()` que reciba como parámetro un dataframe y la cantidad de folds y devuelva una lista de R con la siguiente estructura:
`list(Fold1=c(...), Fold2=c(..),... Fold10=c())`
Donde Fold1 va a contender los índices del dataframe que fueron seleccionados para el primer fold, y así con los demás.

b. Crear una función de nombre `cross_validation()` que reciba como parámetro un data frame y un número de folds y entrene un modelo de árbol de decisión (utilizar paquete `rpart`) para cada uno de los posibles conjuntos de entrenamiento y calcule las métricas: **Accuracy, Precision, Sensitivity, Specificity** para cada uno de los posibles conjuntos de tests. Devolver media y desviación estándar

Ejemplo de uso de rpart

```
# seleccionamos la clase y las variables que nos interesan
library(rpart)
train_formula<-formula(inclinacion_peligrosa~altura+
      Circ_tronco_cm+
      Lat+long+
      Seccion+
      especie)
# generamos el modelo
tree_model<-rpart(train_formula,data=data_train)
# obtenemos la predicción
p<-predict(tree_model,data_val,type='class')
```

2. Forma de entrega:

I. Dentro del repositorio en github con el nombre ia-uncuyo-2024 crear una carpeta con el nombre **tp7-ml**.
II. Colocar dentro de una carpeta **data/** los archivos:
    A. **arbolado-mendoza-dataset-train.csv**
    B. **arbolado-mendoza-dataset-validation.csv**
    C. **arbolado-mendoza-dataset-circ_tronco_cm-train.csv**
III. Colocar un archivo con el nombre **tp7-eda.md** que contenga:
    A. Las respuestas a las preguntas del punto 2 junto a sus correspondientes visualizaciones.
    B. Las visualizaciones(histogramas) generadas en el punto 3
    C. Los criterios de corte seleccionados en el punto 3.
IV. Colocar un archivo con el nombre **tp7-clasificadores.md** que contenga:
    A. La matriz de confusión para el clasificador aleatorio y las métricas correspondientes. (tabla)
    B. La matriz de confusión para el clasificador por clase mayoritaria y las métricas correspondientes. (tabla)
V. Colocar en un archivo con el nombre **tp7-cv.md:**
    A. El código (en un bloque de código) de las funciones `create_folds()` y `cross_validation()`
    B. Una tabla con los resultados (media y desviación estándar de las métricas seleccionadas) de aplicar el clasificador un árbol de decisión en los distintos conjuntos
VI. Dentro de la carpeta (**tp7-ml/**) crear una nueva carpeta **code/.** Dentro de esta, crear una nueva carpeta **intro/** donde se va a incluir el código utilizado en la parte A.

PARTE B)

1. Proponer una estrategia para predecir la peligrosidad del arbolado público de Mendoza con un resultado superior al 0.69 en la métrica AUC en kaggle.

2. Forma de entrega:

I. Colocar un archivo con el nombre **tp7-reporte-arbolado.md** que contenga:

A. Descripción del proceso de preprocesamiento (si es que lo hubiera) Como por ejemplo:
    1. Se eliminaron variables?
    2. Se crearon nuevas?
    3. Se normalizaron valores (0,1)
    4. otras...
B. Resultados obtenidos sobre el conjunto de validación
C. Resultados obtenidos en Kaggle
D. Descripción detallada del algoritmo propuesto

II. Dentro de la carpeta (**tp7-ml/code**) crear una nueva carpeta **desafio/** donde se va a incluir TODO el código utilizado para el envío.

PARTE C)

1. Implementar un algoritmo para construir un árbol de decisión de acuerdo al pseudo-código provisto en AIMA (Se puede implementar en Python).
    a. El algoritmo sólo deberá considerar variables discretas.
    b. Se deberá comprobar su correcto funcionamiento de manera empírica sobre el dataset tennis.csv

2. Investigar sobre las estrategias de los árboles de decisión para datos de tipo **real** y elaborar un breve resumen.

2. Forma de entrega:

I. Colocar un archivo con el nombre **tp7-id3.md** que contenga:
    A. Resultados sobre la evaluación sobre tennis.csv
    B. información sobre las estrategias para datos de tipo real
II. Dentro de la carpeta (**tp7-ml/code**) crear una nueva carpeta **id3/** donde se va a incluir el código utilizado para la implementación del árbol de decisión.