Trabajo Práctico 7B: *Machine Learning*

Parte I

Participar del siguiente desafío de Kaggle para la predicción de inclinación peligrosa del arbolado público de Mendoza:

`https://www.kaggle.com/competitions/arbolado-publico-mendoza-2025`

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

1. Bajar el archivo `arbolado-mendoza-dataset.csv`, el mismo se encuentra en formato csv.

   a) Crear un nuevo archivo con el nombre `arbolado-mendoza-dataset-validation.csv` que contenga un 20 % de los datos, seleccionados de manera aleatoria, siguiendo una distribución uniforme.

   b) Crear un nuevo archivo con el nombre `arbolado-mendoza-dataset-train.csv` que contenga el 80 % restante de los datos.

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 las preguntas anteriores 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 `circ_tronco_cm`. Probar con diferentes números de bins.

   b) Repetir el ejercicio (3*a*) separando por la clase de la variable `inclinacion_peligrosa`.

   c) 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 ejercicio (3*a*) para seleccionar los puntos de corte para cada categoría. Guardar el nuevo dataframe con el nombre `arbolado-mendoza-dataset-circ_tronco_cm-train.csv`.

4. Implementar un clasificador aleatorio, considerando lo siguiente:

   a) Implementar una función que dado un conjunto de observaciones (en forma de dataframe) 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 dataframe y aplicarle la función `random_classifier`.

d) A partir de la columna recientemente generada, y la columna con la clase que indica inclinación peligrosa, calcular utilizando lenguaje `R` (`dplyr`):

i) Número de árboles CON inclinación peligrosa que fueron correctamente predichos como peligrosos por el modelo/algoritmo (*True Positive*).

ii) Número de árboles SIN inclinación peligrosa que fueron correctamente predichos como no peligrosos por el modelo (*True Negative*).

iii) Número de árboles SIN inclinación peligrosa que fueron incorrectamente predichos como peligrosos según el modelo (*False Positives*).

iv) Número de árboles CON inclinación peligrosa que fueron incorrectamente predichos como no peligrosos según el modelo (*False Negatives*).

v) A partir de lo anterior, armar la matriz de confusión correspondiente, similar a la siguiente:

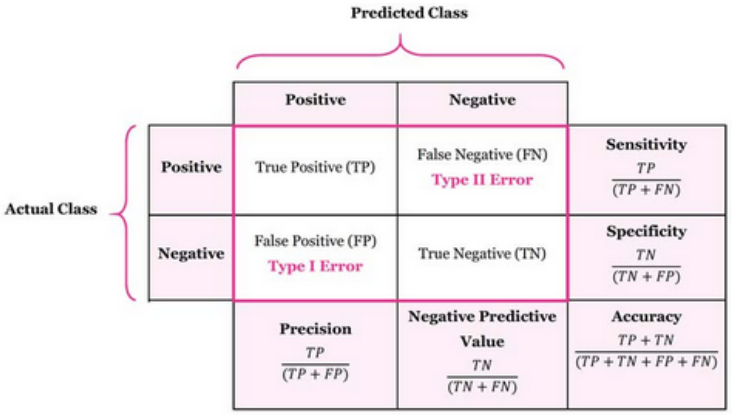
n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

5. Implementar un clasificador por clase mayoritaria:

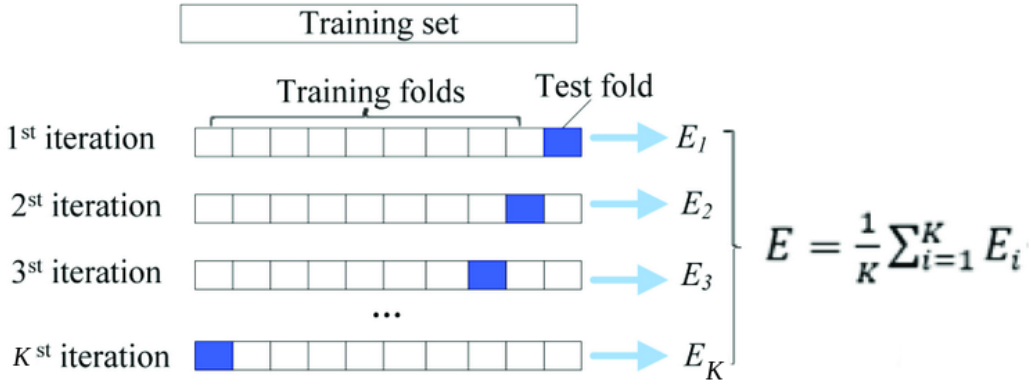
a) Implementar una función de nombre `biggerclass_classifier`, que reciba como parámetro el dataframe generado con anterioridad y genere una nueva columna de nombre `prediction_class` en donde se asigne siempre la clase mayoritaria. La función deberá devolver el dataframe original junto a la nueva columna generada.

b) Repetir los ejercicios (4*c*) y (4*d*), pero aplicando la función `biggerclass_classifier`.

6. A partir de una matriz de confusión es posible calcular distintas métricas que 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 ejercicios (4) y (5).



7. Implementar la técnica de validación cruzada ($k-$*folds Cross validation*):
La validación cruzada es una técnica para estimar el error de generalización de un algoritmo/-modelo de *machine learning*. La técnica consiste en separar el conjunto de datos en k partes (normalmente denominadas *folds*), previo realizar una mezcla aleatoria. Luego, en la primera iteración se utiliza k_1 como conjunto de testeo, y el resto de las partes se utilizan para entrenar E_1. El proceso se repite por k iteraciones utilizando en cada una diferentes conjuntos de entrenamiento y test, como se ejemplifica en la siguiente 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` contenga los índices del dataframe que fueron seleccionados para el primer *fold*, `Fold2` para el segundo, y así con los demás.

   b) Crear una función de nombre `cross_validation()` que reciba como parámetro un dataframe y un número de *folds* y entrene un modelo de árbol de decisión (utilizar el 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.
   A continuación se presenta un 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')
```

8. Forma de entrega:

   a) Dentro de la carpeta `tp7-ml` del repositorio `ia-uncuyo-2025`, crear una carpeta `data` y colocarle los archivos:

   - `arbolado-mendoza-dataset-train.csv`
   - `arbolado-mendoza-dataset-validation.csv`
   - `arbolado-mendoza-dataset-circ_tronco_cm-train.csv`

b) Dentro de la carpeta `tp7-ml`, crear una nueva carpeta `code`, y dentro de ésta una nueva carpeta `eda-clasif-cv` que incluya el código utilizado.

c) Colocar un archivo con el nombre `tp7B-eda.md` que contenga:

i) Las respuestas a las preguntas del ejercicio (2) junto a sus correspondientes visualizaciones.

ii) Las visualizaciones (histogramas) generadas y los criterios de corte seleccionados en el ejercicio (3).

d) Colocar un archivo con el nombre `tp7B-clasificadores.md` que contenga:

i) La matriz de confusión para el clasificador aleatorio y las métricas correspondientes (tabla).

ii) La matriz de confusión para el clasificador por clase mayoritaria y las métricas correspondientes (tabla).

e) Colocar en un archivo con el nombre `tp7B-cv.md`:

i) El código (en un bloque de código) de las funciones `create_folds()` y `cross_validation()`.

ii) Una tabla con los resultados (media y desviación estándar de las métricas seleccionadas) de aplicar el clasificador árbol de decisión en los distintos conjuntos.

Parte II

Dado el desafío de Kaggle:

1. Proponer una estrategia para predecir la peligrosidad del arbolado público de Mendoza con un resultado superior al 0,69 en la métrica AUC.

2. Forma de entrega:

a) Dentro de la carpeta `tp7-ml` del repositorio `ia-uncuyo-2025`, colocar un archivo con el nombre `tp7B-reporte-arbolado.md` que contenga:

i) Descripción del proceso de preprocesamiento (si es que lo hubo), como por ejemplo: ¿se eliminaron variables?, ¿se crearon nuevas?, ¿se normalizaron valores (0,1)?, entre otras.

ii) Resultados obtenidos sobre el conjunto de validación.

iii) Resultados obtenidos en Kaggle.

iv) Descripción detallada del algoritmo propuesto.

b) Dentro de la carpeta `tp7-ml/code` crear una nueva carpeta `desafio` donde se incluya TODO el código utilizado para el envío.

Parte III (Opcional)

Teniendo en cuenta el pseudo-código provisto en AIMA sobre árboles de decisión:

1. Implementar un algoritmo para construir un árbol de decisión de acuerdo al pseudo-código provisto en AIMA. El mismo se puede implementar en `python`. Considerar lo siguiente:

a) El algoritmo sólo deberá considerar variables discretas.

b) Se deberá comprobar su correcto funcionamiento de manera empírica sobre el dataset `tennis.csv`[1].

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

3. Forma de entrega:

[1] `https://raw.githubusercontent.com/sjwhitworth/golearn/master/examples/datasets/tennis.csv`

a) Dentro de la carpeta `tp7-ml` del repositorio `ia-uncuyo-2025`, colocar un archivo con el nombre `tp7B-reporte-id3.md` que contenga:

   i) Resultados sobre la evaluación sobre `tennis.csv`.

   ii) Información sobre las estrategias para datos de tipo real.

b) Dentro de la carpeta `tp7-ml/code` crear una nueva carpeta `id3` donde se incluya el código utilizado para la implementación del árbol de decisión.