Taller 1.

1. Para cualquier número $n > 1$ se define la siguiente operación:
   a. Si el número es par, se divide entre 2.
   b. Si el número es impar, se multiplica por 3 y se suma 1.
   c. Si es 1 termina.

   Se define la **secuencia** de $\boldsymbol{n}$ como la lista de resultados de cada operación hasta llegar a 1. Por ejemplo, la secuencia de 6 es: 6, 3, 10, 5, 16, 8, 4, 2, 1. En este punto solo se pueden utilizar las funcionalidades básicas de Python.

   a) Cree una función que reciba un entero n y retorne una lista con la secuencia de n.
   b) Cree una función que reciba un entero m, retorne el valor n tal que su secuencia sea la más larga para los valores menores a m y retorne el largo de la secuencia. Utilice el punto a).
   c) Similar al punto b), cree una función que reciba un entero m y retorne un diccionario donde las llaves sean enteros y los valores correspondan al largo de las secuencias de cada entero.
   d) Intenté modificar el punto a) y c) para crear una función que utiliza los resultados calculados previamente y retorna un diccionario donde las llaves sean enteros y los valores correspondan al largo de las secuencias de cada entero. Esto debería permitirles crear una función mucho más rápida.
   e) Prueben las funciones del punto c) y d) con diferentes potencias de 10. (No pasen de 10^6 o 10^7)

2. En este punto solo se pueden utilizar las operaciones básicas de numpy. Los vectores son arrays en numpy. Pueden asumir que los vectores Y solo toman dos valores.

   a. Cree una función *euclidean_distance* que calcule la distancia euclidiana en R^n entre dos vectores x, z.
   b. Cree una función *nearest_neighbor* que, dado un vector x y una matriz X_train , retorna el vector de los datos de entrenamiento X_train que minimiza la distancia euclidiana a x. Debe utilizar la función *euclidean_distance*.
   c. Cree una función *k_nearest_neighbors* que, dado un vector x, una matriz X_train y un entero k, retorna los k vectores más cercanos de los datos de entrenamiento X. Debe usar alguna de las funciones creadas previamente.
   d. Cree una función *predict_class* que, dado vector x, una matriz X_train, un vector Y_train, y un entero k, haga la predicción de cuál debe ser su clase (binaria) según el algoritmo de kNN. Debe utilizar la función *k_nearest_neighbors*.
   e. Cree una función *predict_test_set* que, dado una matriz X_test, una matriz X_train, un vector Y_train, y un entero k haga la predicción de los vectores en los datos de evaluación X_test y retorne sus predicciones en un vector Y_pred. Debe utilizar la función *predict_class*.

f. Cree una función *calculate_metrics* que dado un vector Y_test y un vector Y_pred. Calcule las siguientes métricas:
    - Accuracy
    - Precision
    - Recall
    - F1 Score
    - Specifity

   El retorno de la función debe ser un diccionario donde las llaves son las métricas y los valores son los resultados.

3. Recrear el experimento en el libro Elements of Statistical Learning 7.10.2 The Wrong and Right Way to Do Cross-validation.

4. Descargar el archivo classification_datasaet.csv.
    a. Entrenar un modelo que optimice la predicción de ambas clases.
    b. Entrenar un modelo que optimice la predicción de la clase 0.
    c. Entrenar un modelo que optimice la predicción de la clase 1.
    d. Comentar los resultados y las métricas utilizadas para cada caso.

Consideraciones adicionales:

- El trabajo se debe hacer en grupos de entre 2 y 4 personas.
- La idea es trabajar los puntos en conjunto.
- Para la entrega se espera un script de Python o un Jupyter Notebook y un **pequeño** documento para las preguntas 3 y 4d.
- La fecha límite de entrega es el jueves 22 de febrero de 2023 a las 11:59 pm
- Algunos puntos permiten utilizar únicamente ciertas funcionalidades, utilizar funciones por fuera de lo permitido puede invalidar el literal.
- Pueden crear más funciones si lo consideran necesario.