ACTIVIDAD SESIÓN DISEÑO DE REDES NEURONALES EN PYTHON

El objetivo de esta actividad es que los estudiantes diseñen y entrenen una red neuronal para resolver un problema de clasificación utilizando un conjunto de datos realista. Para ello, vamos a trabajar con el **dataset de "Titanic"** disponible en **Kaggle**, el cual es ampliamente utilizado para problemas de clasificación binaria.

En este caso, el problema consiste en predecir si un pasajero sobrevivió al hundimiento del Titanic o no, basándonos en características como la edad, el sexo, la clase del boleto, entre otras. Este es un problema de clasificación binaria, donde las dos clases son: **sobrevivió (1)** o **no sobrevivió (0)**.

INSTRUCCIONES

1. **Realiza la descarga del dataset "Titanic" desde Kaggle (0.5 puntos):**
    - Dirígete a Kaggle Titanic dataset y regístrate/inicia sesión si aún no lo has hecho.
    - Descarga el archivo train.csv, que contiene los datos de entrenamiento, que utilizaremos para crear y entrenar el modelo.
    - Una vez descargado, guarda el archivo train.csv en el directorio de trabajo de tu proyecto.
2. **Realiza la carga de datos en un DataFrame (0.5 puntos):**
    - Usa la librería **Pandas** para cargar el archivo train.csv en un DataFrame y explora las primeras filas para ver qué datos contiene.
    - Usa df.head() para ver las primeras filas del dataset.
3. **Realiza el análisis exploratorio de los datos (EDA) (1 punto):**
    - Inspecciona las columnas y busca posibles valores nulos.
    - Analiza las estadísticas descriptivas de las variables numéricas y el balance de clases en la variable objetivo (columna Survived).
4. **Realiza el preprocesamiento de los datos (2 puntos):**
    - Elimina las columnas que no son necesarias para el modelo (como Name, Ticket, Cabin).

- Imputa los valores nulos de las columnas numéricas (por ejemplo, usando la media o mediana).
- Realiza **One-Hot Encoding** para las variables categóricas como Sex y Embarked, transformándolas en variables numéricas.
- Normaliza las variables numéricas (por ejemplo, utilizando **StandardScaler** para columnas como Age y Fare).

5. **Realiza la división de los datos en entrenamiento y prueba (0.5 puntos):**
   - Usa la función train_test_split de **Scikit-learn** para dividir los datos en un conjunto de entrenamiento (80%) y prueba (20%).
6. **Realiza el diseño de la red neuronal (2 puntos):**
   - Crea una red neuronal utilizando **Keras**. Asegúrate de incluir:
     - Capa de entrada que coincida con el número de características de entrada.
     - Una o más capas ocultas con activación **ReLU**.
     - Capa de salida con activación **Sigmoid** (para clasificación binaria).
   - Compila el modelo con el optimizador **Adam** y la función de pérdida **binary_crossentropy**.
7. **Realiza el entrenamiento del modelo (1 punto):**
   - Entrena el modelo usando el conjunto de datos de entrenamiento y valida el rendimiento utilizando el conjunto de prueba.
   - Utiliza al menos 50 épocas y un tamaño de batch de 32.
8. **Realiza la evaluación del modelo (1 punto):**
   - Evalúa el modelo usando el conjunto de prueba y calcula la precisión (accuracy) en ese conjunto.
   - Asegúrate de imprimir los resultados de la precisión en el conjunto de prueba.
9. **Realiza la visualización del rendimiento del modelo (1 punto):**
   - Grafica la precisión y la pérdida durante el entrenamiento usando **Matplotlib**.

- Compara la precisión en el conjunto de entrenamiento y de validación para ver si hay overfitting o underfitting.

10. **Realiza la predicción sobre los datos de prueba (0.5 puntos):**
    - Usa el modelo entrenado para realizar predicciones sobre los datos de prueba.
    - Imprime las primeras predicciones y compáralas con las etiquetas reales de la columna Survived.

INSTRUCCIONES ADICIONALES:

- Puntos totales = 10 puntos.
- Comprimir el archivo en formato .zip o .rar.
- Subir el archivo a la plataforma.