- **IMPORTANTE:** La elección de un dataset adecuado es parte de la evaluación.

2.

Entregables:

- Un **único Notebook de Jupyter/Colab (.ipynb)** con el nombre ProyectoFinal_Apellido_Nombre.ipynb.
- El notebook debe estar claramente estructurado, con celdas de Markdown para explicar cada sección, y debe ser completamente ejecutable de principio a fin.

Desglose del Proyecto y Tareas:

El notebook debe seguir esta estructura, demostrando una clara separación de responsabilidades.

Parte 1: Selección, Exploración y Limpieza de Datos (2 puntos)

- **1.1. Selección y Justificación:**
  - En una celda de Markdown, presenta el dataset que elegiste. Incluye el enlace a Kaggle.
  - Describe brevemente el problema de negocio y define cuál es tu variable objetivo (target).
- **1.2. Carga e Inspección Inicial:**
  - Carga el dataset en un DataFrame de Pandas.
  - Usa .head(), .info(), y .describe() para una primera inspección.
  - Comenta tus hallazgos iniciales en una celda de Markdown.
- **1.3. Limpieza de Datos:**
  - Identifica y maneja los valores faltantes (NaN). Justifica tu estrategia (¿eliminar?, ¿rellenar?).
  - Asegúrate de que los tipos de datos de las columnas sean los correctos.
- **1.4. Análisis Exploratorio de Datos (EDA):**
  - Crea al menos **dos visualizaciones** relevantes (usando Matplotlib o Seaborn) que te ayuden a entender la relación entre las variables y tu objetivo.
  - Acompaña cada visualización con una celda de Markdown explicando qué observas.

Parte 2: Aplicando Programación Orientada a Objetos (POO) para el Pipeline (4 puntos)

En esta parte, encapsularás el flujo de trabajo en una clase para crear código modular.

- **2.1. Creación de la Clase DataPipeline:**
  - Define una clase llamada DataPipeline.
  - **__init__(self):** El constructor debe inicializar atributos como el modelo (self.model = None).
  - **Método preprocess(self, df):**
    - Este método debe tomar un DataFrame y aplicar el preprocesamiento.
    - **Feature Engineering (Uso Funcional):** Usa el método .apply() con una función lambda para crear al menos una nueva columna a partir de una existente.
    - Maneja las variables categóricas (puedes usar pd.get_dummies()).
    - Devuelve los DataFrames X (features) e y (target) listos para el modelo.
  - **Método train(self, X, y, model_instance):**
    - Recibe los datos de entrenamiento y una instancia de un modelo de Scikit-learn.
    - Divide los datos en entrenamiento y prueba (train_test_split).
    - Entrena el modelo y lo almacena en self.model.
    - Imprime una confirmación de que el modelo fue entrenado.
  - **Método evaluate(self, X, y):**
    - Recibe los datos de prueba.
    - Usa el self.model para hacer predicciones.
    - Calcula y muestra al menos dos métricas de evaluación relevantes.
- **2.2. Demostración de la Clase en el Notebook:**
  - Crea una instancia de tu clase DataPipeline.
  - Llama secuencialmente a los métodos preprocess, train, y evaluate para ejecutar todo el flujo de trabajo.

Parte 3: Calidad y Reproducibilidad del Software (4 puntos)

Aquí simularás buenas prácticas de ingeniería de software dentro de tu notebook.

- **3.1. Testing con pytest (Simulado en Colab):**
  - **Crear un archivo de prueba:** Usa %%writefile test_mi_pipeline.py en una celda para crear un archivo de prueba.
  - **Escribir la prueba:** Dentro de ese archivo, escribe al menos **una prueba unitaria** para una parte de tu lógica. Un buen candidato es