Ejercicio de Árbol de regresión.

Una **inmobiliaria** quiere construir un modelo de **Machine Learning** que le permita **estimar el precio de una casa** en función de algunas características. Para esto, te entrega un archivo con **1000 registros históricos** (`casas_sucias.csv`), pero los datos tienen **errores y valores inconsistentes** que deben ser limpiados antes de usarse.

El archivo contiene las siguientes columnas:

- **superficie**: tamaño en metros cuadrados. Algunos valores vienen como texto (ej. `"120m2"`, `"?"`).
- **habitaciones**: número de habitaciones. Existen errores como `"tres"`, valores faltantes (`NaN`) y números fuera de rango.
- **antiguedad**: años desde la construcción. Hay valores negativos (errores de digitación), la palabra `"nueva"`, y celdas vacías.
- **ubicacion**: tipo de ubicación (`"urbano"` o `"rural"`). Existen errores tipográficos (`"urbnaa"`, `"ruralll"`) y valores nulos.
- **precio**: valor en pesos colombianos. Algunos registros tienen valores extremos como `0` o `9999999` que parecen errores.

🎯 Tu misión

1. **Comprensión del negocio (CRISP-DM)**
   Explica por qué un modelo de predicción de precios es útil para la inmobiliaria.
2. **Comprensión de los datos**
   Carga el archivo `casas_sucias.csv` y analiza:
   - Número de filas y columnas
   - Tipos de datos
   - Valores faltantes y erróneos
3. **Preparación de los datos (Limpieza)**

- Corregir la columna `superficie` (eliminar `"m2"`, reemplazar `"?"` por valores estimados).
- Convertir la columna `habitaciones` a números, corrigiendo `"tres"` y rellenando nulos.
- Transformar la columna `antiguedad`: `"nueva"` → `0`, valores negativos a positivos o reemplazo lógico.
- Normalizar la columna `ubicacion` para que solo tenga `"urbano"` y `"rural"`.
- Detectar y tratar precios anómalos (cercanos a `0` o `9999999`).

4. **Modelado con Árbol de Regresión**
   - Separar variables predictoras y la variable objetivo (`precio`).
   - Dividir el dataset en entrenamiento y prueba.
   - Entrenar un **árbol de regresión** con scikit-learn.
   - Evaluar el modelo con **MAE** y **R² Score**.
5. **Guardar el modelo**
   - Exportar el modelo entrenado en un archivo `.pkl` usando `joblib`.
6. **Despliegue en Render**
   - Construir una **API con FastAPI** que permita enviar características de una casa (`superficie`, `habitaciones`, `antiguedad`, `ubicacion`) y devolver el **precio estimado**.
   - Preparar `requirements.txt` y `main.py`.
   - Subir el proyecto a GitHub y desplegarlo en Render.