Limpiar y asegurar dataset listo para modelado

Continuación del análisis de inundaciones

Este notebook da continuidad al trabajo iniciado en **AA_final_Entrega_2.ipynb**,
donde se realizaron los primeros pasos de procesamiento y se guardó el dataset en la ruta:

`/content/drive/MyDrive/Aprendizaje Automático/Para parcial/Data/dataset_modelo_inundaciones.csv`

En este archivo retomamos ese dataset ya preparado para:

- Profundizar en el análisis de los modelos de aprendizaje automático.
- Generar visualizaciones comparativas (matriz de confusión, curvas ROC/PR).
- Justificar la elección del modelo más adecuado.
- Presentar las conclusiones finales del proyecto.

De esta manera, este notebook integra los ajustes realizados y completa la síntesis estratégica del análisis de inundaciones.

```
# Preparamos entorno
import pandas as pd
import numpy as np

# Ruta del CSV generado en Entrega 2
ruta_csv = "/content/drive/MyDrive/Aprendizaje Automático/Para parcial/Data/dataset_modelo_inundaciones.csv"

# Cargar CSV
df = pd.read_csv(ruta_csv)
print("Cargado:", df.shape)
```

```
Cargado: (2746, 16)
```

```
# Mostrar primeras filas
display(df.head(6))

# Estructura y tipos de datos
print("Shape:", df.shape)
display(df.info())

# Resumen numérico rápido
display(df.describe().round(2))
```

	id	id_poligon	se_inunda_	con_que_fr	provinicia	departamen	localidad	distancia_cuerpo_agua	distancia_curso_agua
0	1	17325.0	1	SOLAMENTE CUANDO DILUVIA (UNA O DOS VECES POR ...	Formosa	Formosa	Formosa	1197.382158	222.458286
1	2	17322.0	1	CADA VEZ QUE LLUEVE FUERTE (MUCHAS VECES POR AÑO)	Formosa	Formosa	Formosa	261.021717	62.385068
2	3	18578.0	2	SOLAMENTE CUANDO DILUVIA (UNA O DOS VECES POR ...	Formosa	Formosa	Formosa	1762.778477	988.503585
3	4	12001.0	1	SOLAMENTE CUANDO DILUVIA (UNA O DOS VECES POR ...	Formosa	Formosa	Villa del Carmen	945.972415	1460.188798
4	5	12000.0	1	SOLAMENTE CUANDO DILUVIA (UNA O DOS VECES POR ...	Formosa	Formosa	Villa del Carmen	1108.288575	1622.064382
5	6	11931.0	1	SOLAMENTE CUANDO DILUVIA (UNA O DOS VECES POR ...	Formosa	Formosa	Villa del Carmen	643.093629	891.469630

```
Shape: (2746, 16)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2746 entries, 0 to 2745
Data columns (total 16 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   id                           2746 non-null   int64
 1   id_poligon                   2741 non-null   float64
 2   se_inunda_                   2746 non-null   int64
 3   con_que_fr                   2746 non-null   object
 4   provinicia                   2746 non-null   object
 5   departamen                   2746 non-null   object
 6   localidad                    2746 non-null   object
 7   distancia_cuerpo_agua        2746 non-null   float64
 8   distancia_curso_agua         2746 non-null   float64
 9   microbasurales_cercanos      2746 non-null   int64
 10  se_inunda_binaria            2746 non-null   int64
 11  frecuencia_num               2746 non-null   int64
 12  frecuencia_codificada_Alta   2746 non-null   bool
 13  frecuencia_codificada_Baja   2746 non-null   bool
 14  frecuencia_codificada_Media  2746 non-null   bool
 15  frecuencia_codificada_Otro   2746 non-null   bool
dtypes: bool(4), float64(3), int64(5), object(4)
memory usage: 268.3+ KB
None
```

	id	id_poligon	se_inunda_	distancia_cuerpo_agua	distancia_curso_agua	microbasurales_cercanos	se_inunda_binar
count	2746.00	2741.00	2746.0	2746.00	2746.00	2746.00	274
mean	1373.50	17169.24	1.5	72396.49	6352.60	2.39	
std	792.85	5768.72	0.5	119056.78	22806.00	1.83	
min	1.00	9.00	1.0	0.00	0.07	1.00	
25%	687.25	14145.00	1.0	5157.33	145.50	1.00	
50%	1373.50	18274.00	2.0	14620.79	691.92	2.00	
75%	2059.75	21552.00	2.0	76227.61	2229.25	3.00	
max	2746.00	25307.00	2.0	650630.87	263048.46	21.00	

Cargamos el CSV generado en la Entrega 2 y verificamos su estructura con .info() y .head().

El dataset contiene 2746 filas y 16 columnas, sin valores nulos en las columnas predictoras clave.

```
# Función para codificar frecuencia (usa "keywords", evita variantes textuales)
def codificar_frecuencia(valor):
    s = str(valor).lower()
    if any(k in s for k in ["cada vez", "muchas veces", "varias veces", "con que me lleve fuerte", "fuerte"]):
        return "Alta"
    if any(k in s for k in ["ocasional", "frecuente", "algunas veces"]):
        return "Media"
    if any(k in s for k in ["solamente", "una o dos", "una vez", "rara vez", "diluv"]):
        return "Baja"
    if any(k in s for k in ["ns", "nc", "otro", "desconoc"]):
        return "Otro"
    return "Otro"

# Asegurar existencia de columna fuente y crear/normalizar 'frecuencia_codificada'
col_freq_src = None
for cand in ["frecuencia_codificada", "con_que_fr", "con_que_fr "]:
    if cand in df.columns:
        col_freq_src = cand
        break
if col_freq_src is None:
    raise KeyError("No encuentro columna origen de frecuencia en el DataFrame. Revisar nombres de columnas.")
print("Fuente frecuencia encontrada:", col_freq_src)

# Si la columna ya existe y tiene valores largos, la normalizamos; si no, la creamos desde el original
# Preferimos aplicar la función sobre la columna textual original (con_que_fr) si existe
if "con_que_fr" in df.columns:
    df["frecuencia_codificada"] = df["con_que_fr"].apply(codificar_frecuencia)
else:
    df["frecuencia_codificada"] = df[col_freq_src].apply(codificar_frecuencia)

print("Distribución frecuencia_codificada:\n", df["frecuencia_codificada"].value_counts())
```

```
Fuente frecuencia encontrada: con_que_fr
Distribución frecuencia_codificada:
 frecuencia_codificada
Alta     1491
Media     638
Baja      572
Otro       45
Name: count, dtype: int64
```

Codificación de la frecuencia de inundación

Se definió y aplicó una función de normalización que agrupa las respuestas textuales originales en cuatro categorías limpias: `Alta`, `Media`, `Baja` y `Otro`. La función opera sobre palabras clave (por ejemplo, "cada vez", "ocasionalmente", "solamente", "ns/nc") para evitar depender de redacciones exactas y cubrir variantes ortográficas.

Resultado esperado: una columna nueva y homogénea `frecuencia_codificada` lista para análisis y modelado, con conteos por categoría que permiten comprobar que no quedan valores residuales con frases largas ni ambigüedades. Esta transformación facilita la creación de variables dummy y la inclusión de la frecuencia como predictor en los modelos supervisados.

```
# Comprobaciones mínimas para la variable objetivo
if "se_inunda_binaria" not in df.columns:
    raise KeyError("No se encontró la columna 'se_inunda_binaria' en el DataFrame. Revisa nombres de columnas.")

print("Tipo de dato:", df["se_inunda_binaria"].dtype)
print("\nDistribución (valor_counts):")
print(df["se_inunda_binaria"].value_counts(dropna=False))
```

```
Tipo de dato: int64

Distribución (valor_counts):
se_inunda_binaria
2    1374
1    1372
Name: count, dtype: int64
```

Se verificó que la variable objetivo `se_inunda_binaria` existe en el DataFrame, muestra su tipo de dato y la distribución de clases. Sirve como control rápido antes de avanzar al preprocesado o al modelado para asegurarse de que el objetivo está presente, en el formato esperado y sin valores faltantes.

```
# Forzar tipos numéricos en distancias y conteos
for col in ["distancia_cuerpo_agua", "distancia_curso_agua", "microbasurales_cercanos", "frecuencia_num"]:
```

```
        if col in df.columns:
            df[col] = pd.to_numeric(df[col], errors="coerce")

    # Eliminar columnas no necesarias para el CSV final (geometría pesada, buffers)
    cols_to_drop = [c for c in ["WKT", "geometry", "buffer_500m"] if c in df.columns]
    if cols_to_drop:
        df = df.drop(columns=cols_to_drop)
        print("Se eliminaron columnas:", cols_to_drop)
```

Forzar la conversión detecta y normaliza valores atípicos o mal tipados (p. ej. comas decimales como string, textos) y los marca como NaN para tratarlos de forma controlada.

Eliminar columnas geométricas reduce el peso del CSV y evita problemas al serializar objetos complejos (geometrías Shapely / WKT) que no se usan directamente en el modelado.

```
# Eliminar filas con NA en objetivo y en frecuencia codificada
antes = df.shape[0]
df = df.dropna(subset=["se_inunda_binaria", "frecuencia_codificada"])
print(f"Filas removidas por NA objetivo/frecuencia: {antes - df.shape[0]}")

# Eliminar dummies antiguas si existen (buscamos prefijo seguido de '_')
old_dummies = [c for c in df.columns if c.startswith("frecuencia_codificada_")]
if old_dummies:
    df = df.drop(columns=old_dummies)
    print("Eliminadas dummies antiguas:", old_dummies)

# Crear dummies limpias (mantener todas las categorías)
df = pd.get_dummies(df, columns=["frecuencia_codificada"], prefix="frecuencia_codificada", drop_first=False)
print("Dummies creadas:", [c for c in df.columns if c.startswith("frecuencia_codificada_")])
```

```
Filas removidas por NA objetivo/frecuencia: 0
Eliminadas dummies antiguas: ['frecuencia_codificada_Alta', 'frecuencia_codificada_Baja', 'frecuencia_codificada_Media', 'fr
Dummies creadas: ['frecuencia_codificada_Alta', 'frecuencia_codificada_Baja', 'frecuencia_codificada_Media', 'frecuencia_coc
```

Se eliminaron filas con valores faltantes en el objetivo y en la variable de frecuencia (no se removió ninguna en esta corrida). Se suprimieron dummies previas generadas en ejecuciones anteriores para evitar duplicados y luego se crearon dummies limpias y consistentes a partir de la columna `frecuencia_codificada`.

El resultado garantiza una codificación one-hot reproducible y lista para modelado: cada categoría aparece como columna booleana y las filas mantienen integridad (una sola categoría activa por fila).

```
# Verificaciones finales
print("\nDtypes finales:\n", df.dtypes)
print("\nNulos por columna:\n", df.isnull().sum())
print("\nCols de frecuencia (dummy):", [c for c in df.columns if c.startswith("frecuencia_codificada_")])
print("\nShape final:", df.shape)

# Guardar CSV final
ruta_base = "/content/drive/MyDrive/Aprendizaje Automático/Para parcial"
nombre_salida = "dataset_modelo_inundaciones.csv"
ruta_salida = os.path.join(ruta_base, nombre_salida)

df.to_csv(ruta_salida, index=False)
print("Guardado en:", ruta_salida)
```

```
Dtypes finales:
 id                            int64
id_poligon                  float64
se_inunda_                    int64
con_que_fr                   object
provinicia                   object
departamen                   object
localidad                    object
distancia_cuerpo_agua       float64
distancia_curso_agua        float64
microbasurales_cercanos       int64
se_inunda_binaria             int64
frecuencia_num                int64
frecuencia_codificada_Alta     bool
frecuencia_codificada_Baja     bool
frecuencia_codificada_Media    bool
frecuencia_codificada_Otro     bool
dtype: object

Nulos por columna:
 id                          0
id_poligon                  5
```