Avance 1: Exploración y Análisis de Datos (EDA)

1. Se carga el dataset

- Se importó el archivo (base_datos_restaurantes_USA_v2.csv) utilizando la ruta al archivo y se verificó su correcta lectura mediante *df.head()* para visualizar las primeras filas del DataFrame y entender su estructura.

2. Se hace una exploración de la estructura del dataset

- Número de filas y de columnas.

- Tipos de datos por columna (strings, bools, enteros, etc.)

3. Se realiza una copia del dataftame y *df* pasó a llamarse *df_sandiego*. Más adelante, habíendo corregido registros y eliminado columnas, se generó copia del último dataframe y *df_sandiego* pasó a llamarse *df_limpio*.

4. Se identifican y se hace un tratamiento de valores faltantes

Se utilizó: *df.isnull().sum()* para identificar valores nulos.

Al recibir columnas con valores nulos, de acuerdo a cada una, se evaluó si se les debía imputar un valor determinado y conservar dicha columna, si estas debían ser eliminadas o si serían conservadas sin modificaciones.

5. Detección de duplicados

Se buscaron registros duplicados usando: *df.duplicated().sum()*. No se encontraron valores duplicados.

Detección de outliers

Inspección manual de valores extremos. Se analizó si los outliers eran errores o casos especiales válidos.

6. Resultados e interpretación

Durante la exploración del dataset se detectaron:

- Columnas con registros nulos relevantes.
- Columnas redundantes o irrelevantes.
- Las visualizaciones iniciales permitieron observar patrones básicos y primeras tendencias del dataset.
- Se identificaron inconsistencias al cruzar datos entre columnas.

Avance 2: Conexión y Extracción de Datos de la API de Yelp

1. Se define la api_url, la api_key (para esta se tiene en cuenta la seguridad ya que es una llave) y los headers de autenticación.

2. Búsqueda Inicial: se realiza una búsqueda con term='restaurants' y location='San Diego'.

3. Paginación: se implementa un bucle *while* (se prefiere antes que el *for*) para poder iterar, aumentando el parámetro offset en 50 registros por paso. Esto permite extraer hasta 200 registros, superando el límite de 50 por llamada. Siendo 200 el límite definido también por Yelp.

4. Normalización JSON: Se utiliza *pd.json_normalize* para normalizar (o aplanar) la estructura JSON de la respuesta que vayamos a recibir. Se extrae la lista de negocios usando *data['businesses']*.

5. Se normalizan las categorías anidadas mediante *record_path=['categories']* mientras se mantienen los metadatos clave (rating, price, name, etc.).

6. Se agrupan las categorías utilizando *groupby* y *.join*.

7. Los dataframes obtenidos en cada iteración (en total son 4 iteraciones de 50 cada uno), se concatenan usando *pd.concat*. Se agrupan los restaurantes por nombre para consolidar todas sus categorías en una sola columna *string*, evitando que se repitan. Esto se logra aplicando además el comando *drop_duplicates*.

8. Se guarda el dataframe final (*df_final*), exportándolo a un archivo .csv llamado ***yelp_restaurants_sandiego.csv***.

Output del Avance:

El resultado de este proceso es el archivo **yelp_restaurants_sandiego.csv**, que contiene datos de restaurantes en la ciudad de San Diego, incluyendo:

- name
- rating
- review_count
- price (original y otro estandarizado)
- categories (listado consolidado de categorías del restaurante)

Pasos para emular y ejecutar localmente:

- Instalar las librerías: pandas - numpy – matplotlib.pyplot – seaborn – requests
- Reemplazar el *placeholder* 'api_key_valida_aqui' en la variable api_key con una clave válida de Yelp.
- Ejecutar las *notebook*:
    - **Avance_EDA.ipynb**
    - **Avance_API_Yelp.ipynb**
    - **Extra_Credits_Web_Scraping.ipynb**

Extra Credits: Web Scraping

Se le pidió a la IA (ChatGPT) que generara un script Python para hacer web scraping tomando la precaución de usar user agent para evitar bloqueo al buscar en wikipedia, en la página de San Diego, California sitios históricos para visitar en dicha ciudad.

El script que generó la IA siguió los siguientes pasos:

1. Importar las librerías necesarias.
2. Usar la URL de la página de Wikipedia de San Diego.
3. Configurar el User-Agent para evitar bloqueos.
4. Parsear el contenido HTML de la página.
5. Incoporar palabras clave que indiquen lugares históricos o monumentos en San Diego.
6. Buscar las listas ‘ul’ y ‘ol’ cuya función es agrupar elementos relacionados en la página.
7. Dentro de cada lista (ul y ol) se buscaron los elementos enlistados ‘li’.
8. Eliminar datos en blanco y duplicados.
9. Guardar en un dataframe.
10. Guardar y exportar el df en un .csv.
11. Finalmente, imprimir la leyenda “Generado por ChatGPT”.