Contexto

InsightReach es una empresa de marketing digital especializada en campañas personalizadas para negocios locales. Con el crecimiento de su base de clientes y la expansión a nuevos mercados, la empresa busca optimizar su estrategia de segmentación para mejorar la efectividad de sus campañas.

Para ello, se ha diseñado un reto técnico dirigido a candidatos para el rol de Científico de Datos Junior, cuyo objetivo es demostrar su capacidad para integrar múltiples fuentes de datos, analizarlas y generar insights accionables.

Proceso Análisis BD (Avance_1_EDA.ipynb)

1. Objetivos del análisis BD clientes

- Explorar y conocer el comportamiento de los ciudadanos de 10 ciudades de Estados Unidos respecto al consumo y preferencias en los diferentes restaurantes de las mismas. Con este análisis se busca responder a preguntas como:
    - ¿En qué ciudad se gasta más dinero en consumo de restaurantes?
    - ¿Quiénes gastan más en restaurantes, los hombre o las mujeres?
    - ¿Cuánto gastan en promedio los ciudadanos según su estrato socioeconómico?
    - ¿Cuál es la relación entre la frecuencia de visitas a restaurantes y el promedio de gasto en cada una de ellas?
    - ¿Cómo se distribuyen las preferencias alimenticias según la ciudad de residencia?
    - Entre muchas otras.

2. Carga y Descripción Inicial de los Datos

- Carga de librerías: Se importan pandas, numpy, matplotlib, seaborn y openpyxl para análisis y visualización.

- Carga del dataset: Se lee el archivo base_datos_restaurantes_USA_v2.csv en un DataFrame llamado df.

- Descripción de columnas: Se documentan las variables del dataset, que incluyen datos demográficos, hábitos de consumo, preferencias alimenticias, membresía premium, ingresos, etc.

3. Limpieza y Preprocesamiento

a. Revisión y eliminación de columnas innecesarias

Se identifican y eliminan columnas que no aportan valor analítico, como id_persona, nombre, apellido, telefono_contacto y correo_electronico.

b. Detección y tratamiento de valores nulos

- Se analiza el porcentaje de valores nulos por variable.

- Se imputan valores nulos en variables numéricas usando KNNImputer (vecinos más cercanos).

- Para la variable categórica preferencias_alimenticias, los nulos se reemplazan por "Otro".

c. Detección y tratamiento de outliers

Se identifican valores imposibles en variables como edad y frecuencia_visita y se reemplazan por NaN para su posterior imputación.

Se ajustan los rangos de edad (18-80 años) y frecuencia de visita (0-15 veces al mes).

d. Conversión de tipos de datos

Se ajustan los tipos de datos de las variables numéricas a int o float según corresponda.

4. Análisis Exploratorio de Datos (EDA)

a. Estadísticas descriptivas y visualización

Se generan histogramas y boxplots para analizar la distribución y detectar outliers en variables numéricas.

Se exploran las variables categóricas con conteos y gráficos de barras.

b. Análisis por segmentos

Por ciudad: Se analiza el gasto promedio mensual y la distribución de preferencias alimenticias y estratos socioeconómicos por ciudad.

Por género y estrato: Se estudia el gasto promedio y la distribución de membresías premium.

Clientes de alto gasto: Se identifican y describen los clientes que más gastan (top 10%).

c. Relación entre variables

- Se exploran relaciones entre frecuencia de visita, gasto, ingresos y membresía premium.

- Se analiza el consumo de licor por grupos de edad.

5. Integración con Datos Externos (Yelp)

- Se filtran los clientes de una ciudad específica (ejemplo: Miami).

- Se carga la base de restaurantes de Yelp (yelp_restaurants.csv).

- Se calcula un score ponderado para los restaurantes considerando rating y cantidad de reviews.

6. Recomendaciones Personalizadas

a. Agrupación de categorías de restaurantes por preferencias alimenticias

Se crea un diccionario que asocia cada preferencia alimenticia (Vegetariano, Mariscos, Carnes, Vegano, etc.) con las categorías de restaurantes relevantes de Yelp.

b. Función de recomendación por preferencia

Se implementa la función recomendar_por_preferencia, que filtra restaurantes según las categorías asociadas a la preferencia del cliente y su score.

c. Función de recomendación por estrato socioeconómico

Se implementa la función recomendar_por_estrato, que filtra restaurantes según el rango de precios adecuado para cada estrato y su score.

7. Visualización de Resultados

Se presentan tablas y gráficos que muestran:

- Distribución de clientes por ciudad, género, estrato y preferencias.
- Gasto promedio por segmento.
- Recomendaciones de restaurantes personalizadas.
- Entre otros

8. Conclusiones

El análisis permite:

- Identificar segmentos de clientes valiosos para estrategias de marketing.

- Detectar tendencias de consumo y preferencias alimenticias.
- Integrar datos externos para enriquecer la oferta de recomendaciones.
- Proponer estrategias de fidelización y personalización basadas en datos.

Proceso API Yelp (Avance_2_API_Yelp.ipynb)

1. Objetivo

El propósito de este notebook es obtener, limpiar y analizar información de restaurantes de una ciudad específica (ejemplo: Miami) utilizando la API pública de Yelp. El resultado es un archivo enriquecido (yelp_restaurants.csv) que puede ser utilizado para análisis posteriores y recomendaciones personalizadas.

2. Carga de Librerías

Se importan las siguientes librerías:

- pandas y numpy para manipulación de datos.
- requests para consumir la API de Yelp.
- matplotlib y seaborn para visualización.
- openpyxl para manejo de archivos Excel.

3. Configuración de la API de Yelp

- Se definen las credenciales de acceso (apikey) y la URL base de la API.

- Se establece la ciudad de interés (por defecto, "Miami").

4. Consulta Inicial a la API

- Se realiza una primera consulta para obtener información básica de restaurantes en la ciudad seleccionada.

- Se exploran los datos recibidos y se verifica la cantidad total de restaurantes disponibles en la API para esa ciudad.

5. Extracción de Todos los Resultados

- La API de Yelp limita la cantidad de resultados por consulta a 50.

- Se implementa un ciclo que utiliza el parámetro offset para paginar los resultados y obtener hasta 200 restaurantes (límite de la API gratuita).

- Cada consulta devuelve un lote de restaurantes, que se normaliza y almacena en un DataFrame.

- Todos los DataFrames se concatenan para formar un único DataFrame con todos los restaurantes extraídos.

6. Limpieza y Transformación de Datos

- Se convierten los conteos de reviews a tipo entero.
- Se imputan valores nulos en la columna de precios con "No especificado".
- Se transforman los símbolos de precio ($, $$, etc.) a categorías descriptivas ("Bajo", "Medio", "Alto", "Muy Alto").
- Se eliminan duplicados basados en el identificador del restaurante.
- Se genera una copia del DataFrame para análisis por categoría.

7. Análisis Exploratorio

- Se calcula el rating promedio y el promedio de reviews por ciudad.
- Se identifican los restaurantes con más reviews.
- Se calcula un score ponderado para cada restaurante, considerando tanto el rating como la cantidad de reviews (método similar al de IMDb).
- Se listan los mejores restaurantes según este score.

8. Visualización

- Se grafica la cantidad de restaurantes por ciudad y nivel de precio usando un gráfico de barras apiladas.

9. Exportación de Datos

- El DataFrame final, con todas las categorías y atributos relevantes, se exporta a un archivo CSV llamado yelp_restaurants.csv para su uso en análisis posteriores.

10. Conclusión

Este proceso permite:

- Obtener una muestra representativa de restaurantes de una ciudad desde la API de Yelp.
- Limpiar y transformar los datos para facilitar su análisis.
- Calcular métricas relevantes para identificar los mejores restaurantes.
- Generar un archivo listo para integrarse con otras fuentes de datos y sistemas de recomendación.

Proceso Web Scraping y Automatización en Wikipedia para Ciudades de EE. UU. (Avance_3_WebScraping.ipynb)

1. Objetivo

El propósito de este notebook es hacer un pedido a la IA (chatgpt o copilot) para que esta genere un código que permita hacer web scraping de la sección "Cultura" de la página de Wikipedia de cualquier ciudad de Estados Unidos, utilizando BeautifulSoup. Esto se debe hacer por medio del siguiente prompt:

“Necesito que me des un scraping en python, utilizando beautiful soup de la página de wikipedia de cualquier ciudad de Estados Unidos, por lo que necesito que ese sea un parámetro en la url y que le pueda pasar una variable con el nombre de la ciudad que quiero scrapear. Necesito que el scraping busque el h2 con id = "Cultura" (no span, H2, ya que ahora funciona así), normalmente estos h2 se encuentran dentro de un div, por lo cual se debe encontrar el parent de dicho h2 para que luego saque todo el contenido (sin imágenes) de esa sección, incluyendo h3 y párrafos. Estos h3 y párrafos, suelen encontrarse al mismo nivel del div parent del h2, NO dentro de él. Ten en cuenta que los h3 también vienen dentro de divs, pero los párrafos no y que si se encuentra otro div con un h2 dentro es porque está iniciando una sección diferente. Si no se encuentra el h2 con el id = 'Cultura', necesito que me arroje un mensaje de que no fue encontrada la sección en la página. Si la ciudad proporcionada no se encuentra en wikipedia, quiero que imprima un mensaje sugiriendo buscar cómo aparece la ciudad en la url de wikipedia. Por último, quiero que los h3 se distingan de los párrafos, que se pueda evidenciar fácilmente como título y que los párrafos tengan algunos saltos de línea y se vean organizados”

2. Scraping de la sección "Cultura" con BeautifulSoup generado por IA

a. Descripción del proceso

- Se define una función scrape_cultura_section(city_name) que recibe el nombre de la ciudad como parámetro.

- Se construye la URL de Wikipedia en español para la ciudad deseada.

- Se realiza una petición HTTP a la página.

- Si la página no existe, se muestra un mensaje sugiriendo verificar el nombre en la URL de Wikipedia.

- Se busca el elemento <h2> con id="Cultura". Si no existe, se informa que no se encontró la sección.

- Se identifica el div padre de ese <h2>.

- Se recorre el contenido hermano de ese div hasta encontrar otro div con un <h2>, lo que indica el fin de la sección.

- e extraen los títulos <h3> (dentro de divs) y los párrafos <p>, distinguiéndolos claramente en la salida.

- Si no se encuentra contenido relevante, se informa al usuario.

b. Características adicionales

- El nombre de la ciudad es un parámetro, permitiendo reutilizar la función para cualquier ciudad.

- Los títulos <h3> se muestran destacados y los párrafos organizados con saltos de línea.

- El código es robusto ante cambios en la estructura de la página y errores de conexión.

3. Automatización de Wikipedia con Selenium (Puntos Extra)

a. Descripción del proceso

- Se define la función explorar_wikipedia(ciudad_buscar, idioma_deseado).
- Se abre la página de Wikipedia de una ciudad (por defecto, Miami).

- Se utiliza el buscador interno de Wikipedia para buscar otra ciudad.

- Se intenta cambiar el tema de la página a "Oscuro" (o "Claro" si ya está en oscuro).

- Se abre el menú de idiomas y se selecciona el idioma deseado (por ejemplo, "Português").

- Se espera unos segundos para visualizar los cambios y luego se cierra el navegador.

b. Características adicionales

- Uso de Selenium WebDriver y WebDriver Manager para automatizar la descarga y gestión del driver de Chrome.

- El nombre de la ciudad y el idioma son parámetros de la función.

- El código maneja excepciones si no se puede cambiar el tema o el idioma, mostrando mensajes informativos.

4. Conclusión

Este notebook permite:

- Extraer de forma flexible y organizada la información de la sección "Cultura" de cualquier ciudad de EE. UU. en Wikipedia.
- Automatizar la navegación, búsqueda, cambio de tema y cambio de idioma en Wikipedia usando Selenium.
- Proveer mensajes claros al usuario en caso de errores o ausencia de información.