**📌 Propuesta de Proyecto: Análisis de Accidentes de Tránsito**

**Hipótesis nula sugerida:**  
*“El número de accidentes de tráfico no varía entre días laborables y fines de semana.”*

**📁 Paso 1: Dataset**

Conseguir un dataset real con accidentes de tránsito que no esté completamente limpio.

Archivo obtenido de la base de datos de Kaggle

**📊 Dataset: Accidentes de Tráfico en Madrid (2019–2023)**

Este conjunto de datos contiene información detallada sobre accidentes de tráfico en Madrid desde 2019 hasta 2023. Incluye campos como:

* Número de expediente
* Fecha y hora del accidente
* Localización
* Distrito
* Tipo de accidente
* Condiciones meteorológicas
* Número de víctimas

Se puedes acceder y descargar el dataset desde: ([Kaggle](https://www.kaggle.com/datasets/jairoordezpacheco/accidentes-de-trfico-de-madrid-espaa-2019-a-2023?utm_source=chatgpt.com" \o "Accidentes de Tráfico de Madrid españa 2019 a 2023 - Kaggle))

**🧪 Hipótesis nula a contrastar**

*"El número de accidentes de tráfico no varía entre días laborables y fines de semana."*

**🛠️ Pasos para desarrollar el proyecto**

1. **Carga y exploración de datos:**
   * Utilizamos pandas para cargar el archivo CSV.
   * Exploramos las primeras filas y las columnas disponibles.
   * Identificamos valores nulos o inconsistencias.
2. **Limpieza y transformación:**
   * Convertimos las columnas de fecha y hora al tipo de dato datetime.
   * Creamos una nueva columna que indique el día de la semana.
   * Clasificamos los días como "laborables" o "fines de semana".
3. **Análisis estadístico:**
   * Agrupamos los datos por tipo de día y calculamos el número medio de accidentes.
   * Utilizo bucles y condicionales para contar los accidentes por día.
   * Defino y utilizo al menos una función para automatizar tareas repetitivas. Bucle (While).
4. **Visualización:**
   * Crea gráficos de barras o líneas que muestren la distribución de accidentes por día.
   * Utilizamos Power Bi para las visualizaciones.
5. **Conclusiones:**
   * Interpretamos los resultados y determinamos si se rechaza o no la hipótesis nula.
   * Posibles razones para los patrones observados.

**Código en Python**

**import pandas as pd**

**# Cargar datos**

**df = pd.read\_csv("datos\_madrid.csv", encoding="latin1")**

**# Limpiar la hora**

**df['hora\_limpia'] = df['hora'].str.replace('a.Êm.', 'AM', regex=False)**

**df['hora\_limpia'] = df['hora\_limpia'].str.replace('p.Êm.', 'PM', regex=False)**

**# Crear columna datetime combinada**

**df['fecha\_hora'] = pd.to\_datetime(df['fecha'] + ' ' + df['hora\_limpia'], errors='coerce', dayfirst=True)**

**# Extraer día de la semana**

**df['dia\_semana'] = df['fecha\_hora'].dt.day\_name()**

**# Clasificar días como laborables o fin de semana**

**def clasificar\_dia(dia):**

**if dia in ['Saturday', 'Sunday']:**

**return 'fin de semana'**

**else:**

**return 'laborable'**

**df['tipo\_dia'] = df['dia\_semana'].apply(clasificar\_dia)**

**# Mostrar verificación**

**print(df[['fecha', 'hora', 'hora\_limpia', 'fecha\_hora', 'dia\_semana', 'tipo\_dia']].head())**

**📥 Paso 1: Movemos el archivo CSV al proyecto**

1. **Ubicamos el archivo descargado (llamado datos\_madrid.csv).**
2. **Moverlo manualmente a la carpeta donde tengo el archivo Trabajo\_Accidentes.py**

**🐍 Paso 2: Cargamos el archivo en el código Python**

**Una vez que el archivo .csv esté en la misma carpeta, uso este código en el archivo .py:**

**import pandas as pd**

**# Cargar datos**

**df = pd.read\_csv("datos\_madrid.csv", encoding="latin1")**

**print(df.head())**

**# Verificar que se cargue correctamente**

**▶️ Paso 3: Ejecutar desde la terminal**

**En la terminal de Visual Studio Code (abajo), escribimos:**

**python Trabajo\_Accidentes.py**

**Y veremos impreso en la terminal las primeras filas del dataset !!!!.**

**🔄 Paso 4: Limpieza y transformación**

**🎯 Objetivo:**

* **Convertir fecha + hora en datetime.**
* **Crear columna dia\_semana (ej: Monday, Tuesday...).**
* **Clasificar días como "laborable" o "fin de semana".**

**# Limpiar la hora**

**df['hora\_limpia'] = df['hora'].str.replace('a.Êm.', 'AM', regex=False)**

**df['hora\_limpia'] = df['hora\_limpia'].str.replace('p.Êm.', 'PM', regex=False)**

**# Crear columna datetime combinada**

**df['fecha\_hora'] = pd.to\_datetime(df['fecha'] + ' ' + df['hora\_limpia'], errors='coerce', dayfirst=True)**

**# Crear columna con día de la semana**

**df['dia\_semana'] = df['fecha\_hora'].dt.day\_name()**

**# Función para clasificar tipo de día**

**def clasificar\_dia(dia):**

**if dia in ['Saturday', 'Sunday']:**

**return 'fin de semana'**

**else:**

**return 'laborable'**

**# Aplicar función a nueva columna**

**df['tipo\_dia'] = df['dia\_semana'].apply(clasificar\_dia)**

**# Verificar columnas nuevas**

**print(df[['fecha\_hora', 'dia\_semana', 'tipo\_dia']].head())**

**Por defecto, day\_name() devuelve los nombres en inglés, pero lo vamos a traducir fácilmente.**

**Un diccionario**

**dias\_ingles\_espanol = {**

**'Monday': 'Lunes',**

**'Tuesday': 'Martes',**

**'Wednesday': 'Miércoles',**

**'Thursday': 'Jueves',**

**'Friday': 'Viernes',**

**'Saturday': 'Sábado',**

**'Sunday': 'Domingo'**

**}**

**df['dia\_semana'] = df['fecha\_hora'].dt.day\_name().map(dias\_ingles\_espanol)**

1. **df['dia\_semana']**
   * **Es la columna de tu DataFrame que contiene el nombre del día de la semana (por ejemplo, 'Lunes', 'Martes', etc.).**
2. **.apply(clasificar\_dia)**
   * **.apply() es un método de pandas que permite aplicar una función a cada elemento de una columna.**
   * **En este caso, la función que se aplica es clasificar\_dia, que definimos anteriormente.**
3. **df['tipo\_dia'] = ...**
   * **El resultado de aplicar la función se guarda en una nueva columna llamada 'tipo\_dia'.**

**¿Qué hace esto?:**

* **Sirve para clasificar cada día de la semana en una categoría, por ejemplo:**
  + **Si el día es 'Sábado' o 'Domingo', la función podría devolver 'Fin de semana'.**
  + **Si es cualquier otro día, podría devolver 'Laborable'.**
* **Así, en la columna 'tipo\_dia' tendrás etiquetas como 'Laborable' o 'Fin de semana' para cada fila.**

**📊 Paso 5: Análisis estadístico**

**🎯 Objetivo:**

* **Calcular la media de accidentes por tipo de día.**
* **Usar bucles y condicionales.**
* **Crear al menos una función propia.**

**accidentes\_por\_dia = df.groupby('tipo\_dia').size()**

**print("\nAccidentes por tipo de día:")**

**print(accidentes\_por\_dia)**

1. **df.groupby('tipo\_dia')**
   * **Agrupa el DataFrame df según los valores de la columna 'tipo\_dia'.**
   * **Es decir, crea grupos para cada categoría de tipo de día (por ejemplo, 'Laborable' y 'Fin de semana').**
2. **.size()**

* **Cuenta cuantas filas(accidentes) hay en cada Grupo.**
* **El resultado es una Serie de Pandas donde el índice son los tipos de día y el valor es el número de accidentes en cada tipo de día.**

1. **Accidentes\_por\_día=…**

* **Esto guardará el resultado en la variable accidentes\_por\_día.**