BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Plantel CU2 "Ecocampus Valsequillo"

Asignatura

Introducción a la ciencia de datos

Docente

Jaime Alejandro Romero Sierra

Tema

Limpieza de la base de datos, 2do reporte

Repositorio

https://github.com/AzC4R10z/Proyecto.git

Alumno

Carlos Cortez Almaro

Sección

002

Puebla, Puebla a 20/10/2025

Índice

Descripción inicial de la base de datos……..……..3

Contexto de la base de datos……..3

Descripción general del contenido……3

Significado de cada columna……..5

Proceso de limpieza……..6

Revisión de datos faltantes……6

Detección y manejo de duplicados……6

Corrección de valores atípicos o inconsistentes……7

Detección y corrección de palabras extrañas o mal escritas ……8

Traducción de textos al español……8

Cambio de nombres de columnas para mayor claridad o consistencia……9

Conversión de tipos de datos……10

Validación final mostrando que la base quedó limpia y coherente……10

Conclusiones……11

Problemas principales que presentaba la base……11

Técnicas aplicadas para solucionarlos……12

Aprendizaje del proceso……13

Descripción inicial de la base de datos

Contexto de la base de datos

Este conjunto de datos proporciona una visión integral del rendimiento de los vehículos eléctricos (VE), el comportamiento de carga, la salud de la batería y el análisis de costos a través de diferentes fabricantes, modelos, regiones y tipos de uso. Es un conjunto de datos generado de manera sintética diseñado para asemejarse a los datos del mundo real y puede ser utilizado para:

- Análisis Exploratorio de Datos (EDA)
- Aprendizaje Automático y Modelado Predictivo
- Estudios de Optimización de Batería y Carga
- Análisis de Costos, Reventa y Mantenimiento
- Análisis de Sostenibilidad y Ahorros de CO_2

El surgimiento de la repentina polémica entre los autos eléctricos y los híbridos desatado por el líder de Toyota, Akio Toyoda, poniendo en tela de juicio la supuesta superioridad ambiental de los autos eléctricos; ha hecho que este tema sea más necesario profundizarlo y así aclarar las posibles dudas surgidas desde dichas declaraciones.

Aquí es donde toma lugar el proyecto "*Análisis del rendimiento y del impacto ambiental de los vehículos eléctricos a nivel mundial",* llevado a cabo por el alumno del primer semestre de la Ingeniería en Ciencia de Datos: Carlos Cortez Almaro. Se buscará, por medio de los datos en el dataset, realizar un profundo análisis del rendimiento energético y el impacto ambiental de los vehículos eléctricos (EV) a nivel mundial, considerando factores como la capacidad y el estado de las baterías, la autonomía, el consumo de energía, los costos de mantenimiento y las emisiones evitadas de CO_2; dados en dicho dataset. Para lo cual, antes que todo, tuvo que ser limpiado.

Descripción general del contenido

La base de datos contiene información sobre **vehículos eléctricos (VE)** y sus características técnicas, de rendimiento, costos y eficiencia energética.

- **Número total de registros (filas):** 3,531
- **Número total de variables (columnas):** 25

Las variables incluyen tanto datos técnicos del vehículo (marca, modelo, año, capacidad de batería, autonomía, potencia, aceleración, etc.) como información económica y de sostenibilidad (costos de mantenimiento, seguro, electricidad, CO_2 ahorrado, valor de reventa, entre otros).

Lista de variables

- ***ID_del_Vehículo***
- ***Marca***
- ***Modelo***
- ***Año***
- ***Región***
- ***Tipo_de_Vehículo***
- ***Capacidad_de_Batería_kWh***
- **Salud_de_Batería_%**
  Autonomía_km
- ***Potencia_de_Carga_kW***
- ***Tiempo_de_Carga_hr***
- ***Ciclos_de_Carga***
- ***Consumo_de_Energía_por_100km_recorridos***
- ***Kilometraje_km***
- ***Velocidad_Promedio_kmh***
- ***Velocidad_Máxima_kmh***
- ***Aceleración_0_100_kmh_seg***
- ***Temperatura_°C***
- ***Tipo_de_Uso***
- ***CO2_Ahorrado_tons***
- ***Costo_de_Mantenimiento_USD_por_año***
- ***Costo_de_Seguro_USD_por_año***
- ***Costo_de_Electricidad_en_USD_por_kWh***
- ***Costo_Mensual_de_Carga_USD***
  - ***Valor_de_Reventa_USD***

Calidad de los datos

- No existen **valores nulos** en ninguna columna.
- Se detectaron **48 registros duplicados**, los cuales pueden eliminarse para mejorar la precisión de los análisis.
- Todas las columnas fueron importadas como tipo **objeto (texto)**, por lo que puede ser necesario **convertir algunas variables a formato numérico o de fecha** antes de realizar análisis estadísticos o modelado.

Significado de cada columna

- **ID_del_Vehículo** — Identificador único del registro.
- **Marca** — Fabricante del vehículo (Tesla, BMW, Nissan, etc.).
- **Modelo** — Nombre del modelo (Model 3, Leaf, ID.4, etc.).
- **Año** — Año del modelo/matriculación.
- **Región** — Región geográfica del registro (Europe, North America, Asia, Australia).
- **Tipo_de_Vehículo** — Carrocería o segmento (Sedán, SUV, Hatchback, Truck, etc.).
- **Capacidad_de_Batería_kWh** — Energía total de la batería en kWh.
- **Salud_de_Batería_%** — Salud de la batería en porcentaje.
- **Autonomía_km** — Kilómetros que el vehículo puede recorrer con carga completa.
- **Potencia_de_Carga_kW** — Potencia máxima de carga (kW).
- **Tiempo_de_Carga_hr** — Tiempo de carga (horas).
- **Ciclos_de_Carga** — Número de ciclos de carga registrados.
- **Consumo_de_Energía_por_100km_recorridos** — kWh consumidos por 100 km.
- **Kilometraje_km** — Kilómetros totales recorridos por el vehículo.
- **Velocidad_Promedio_kmh** — Velocidad media de uso (km/h).
- **Velocidad_Máxima_kmh** — Velocidad máxima registrada (km/h).
- **Aceleración_0_100_kmh_seg** — Tiempo de aceleración 0–100 km/h (segundos).
- **Temperatura_°C** — Temperatura media registrada (°C).
- **Tipo_de_Uso** — Uso principal del vehículo (Particular, Taxi, Flota, etc.).
- **CO2_Ahorrado_tons** — Toneladas de CO_2 evitadas por el uso del VE (estimación).
- **Costo_de_Mantenimiento_USD_por_año** — Costo anual de mantenimiento (USD).
- **Costo_de_Seguro_USD_por_año** — Costo anual del seguro (USD).
- **Costo_de_Electricidad_en_USD_por_kWh** — Precio de la electricidad (USD/kWh).
- **Costo_Mensual_de_Carga_USD** — Costo promedio mensual de carga (USD).
- **Valor_de_Reventa_USD** — Valor estimado de reventa (USD).

Proceso de limpieza

Revisión de datos faltantes

```
#Información general del DataFrame, en especial tipos de datos y valores nulos
print(df.info())
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3579 entries, 0 to 3578
Data columns (total 25 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Vehicle_ID                      3436 non-null   float64
 1   Make                            3436 non-null   object
 2   Model                           3436 non-null   object
 3   Year                            3436 non-null   object
 4   Region                          3436 non-null   object
```

```
#Verificamos los valores nulos finales
df.isnull().sum()
```

✓ 0.0s

```
Vehicle_ID                      0
Make                            0
Model                           0
Year                            0
Region                          0
Vehicle_Type                    0
Battery_Capacity_kWh            0
Battery_Health_%                0
Range_km                        0
Charging_Power_kW               0
Charging_Time_hr                0
Charge_Cycles                   0
Energy_Consumption_kWh_per_100km    0
```

- Se ocupó **info()** para mostrar los tipos de datos que contenía en general el dataframe y así darnos una idea sobre el como proceder.
- Se ocupó **isnull().sum()** para saber cuántos valores nulos existían, después para corroborar que habían sido eliminados.

Detección y manejo de duplicados

Un registro duplicado ocurre cuando dos o más filas contienen exactamente la misma información en todas sus columnas.

Durante la limpieza de la base de datos se realizó una revisión para identificar la existencia de registros duplicados. Para ello se utilizó

duplicated() que permite detectar filas que contienen exactamente los mismos valores en todas sus columnas.

Antes de proceder con su eliminación, se analizó el tipo de información que contiene la base. Dado que las variables representan vehículos eléctricos (marca, modelo, año, capacidad de batería, autonomía, potencia, costos, entre otras características técnicas y económicas), cada fila debería corresponder a un vehículo único o modelo único. Por tanto, la presencia de registros idénticos no representa observaciones diferentes, sino una repetición innecesaria de información.

Entonces se determinó que los registros duplicados **debían eliminarse**, ya que su permanencia podría afectar el análisis posterior

```
# Contar duplicados exactos
df.duplicated().sum()
```
✓ 0.0s

```
np.int64(48)
```

```
# Eliminar duplicados exactos
df = df.drop_duplicates()
```
✓ 0.0s

```
df.duplicated().sum()
```
✓ 0.0s

```
np.int64(0)
```

Corrección de valores atípicos o inconsistentes

Se detectó que había valores de texto en las numéricas, entonces se creó un diccionario para después transformarlos en NaN con **error='coerce'** para después decidir qué hacer con ellos.

Usando un ciclo for para cambiar los de todas las columnas.

```python
#Construimos un diccionario con las columnas numéricas que están como texto:
cols_numericas=[
    "Charging_Power_kW", "Charging_Time_hr", "Charge_Cycles",
    "Avg_Speed_kmh", "Max_Speed_kmh", "Acceleration_0_100_kmh_sec",
    "Resale_Value_USD", "Year"
]
#Transformamos valores no numéricos en NaN, para después decidir cómo tratarlos.
for c in cols_numericas:
    df[c] = pd.to_numeric(df[c], errors='coerce') #Según una pequeña investigación, los
```

✓ 0.0s

Detección y corrección de palabras extrañas o mal escritas

Se buscan palabras y/o valores atípicos con un ciclo for.

```python
#Conocemos los valores únicos de cada columna y así detectar valores atípicos
for i in columnas:
    print(f'Valores únicos en la columna {i}:')
    print(df[i].unique())
    print("")
```

✓ 0.0s

Encontramos NaN y la palabra “Ver4$zul”, usamos un ciclo for para reemplazarlos por “Desconocido”

```python
#Usamos un ciclo for para limpiar los datos
for i in columnas:
    #Llenamos los valores nulos con 'Desconocido'
    df[i]=df[i].fillna('Desconocido')
    #Reemplazamos los valores atípicos por 'Desconocido'
    df[i]=df[i].replace('Ver4$zul','Desconocido')
```

✓ 0.0s

Traducción de textos al español

```python
#Renombramos los nombres de las columnas del DataFrame limpio
df=df.rename(columns={'Vehicle_ID': 'ID_del_Vehículo', 'Make': 'Marca', 'Model': 'Modelo', 'Year': 'Año'
                      'Vehicle_Type': 'Tipo_de_Vehículo', 'Battery_Capacity_kWh': 'Capacidad_de_Batería_
                      'Battery_Health_%': 'Salud_de_Batería_%', 'Range_km': 'Autonomía_km',
                      'Charging_Power_kW': 'Potencia_de_Carga_kW', 'Charging_Time_hr': 'Tiempo_de_Carga_
                      'Charge_Cycles': 'Ciclos_de_Carga', 'Energy_Consumption_kWh_per_100km': 'Consumo_d
                      'Mileage_km': 'Kilometraje_km', 'Avg_Speed_kmh': 'Velocidad_Promedio_kmh',
                      'Max_Speed_kmh': 'Velocidad_Máxima_kmh', 'Acceleration_0_100_kmh_sec': 'Aceleració
                      'Temperature_C': 'Temperatura_°C', 'Usage_Type': 'Tipo_de_Uso',
                       'CO2_Saved_tons': 'CO2_Ahorrado_tons', 'Maintenance_Cost_USD': 'Costo_de_Manteni
                       'Insurance_Cost_USD': 'Costo_de_Seguro_USD_por_año', 'Electricity_Cost_USD_per_k
```

✓ 0.0s

Se ocupa **rename(columns={})** para renombrar las columnas debido a que todas se encontraban en inglés.

Se ocupa **unique()** para las columnas con valores de texto y detectar las palabras que también necesiten traducción

```
#Revisamos los valores de Región para traducirlos
print(df['Región'].unique())
print("")
#Revisamos los valores de la columna Vehicle_Type para traducirlos
print(df['Tipo_de_Vehículo'].unique())
print("")
#Revisamos los valores de la columna Usage_Type para traducirlos
print(df['Tipo_de_Uso'].unique())
print("")
```

✓ 0.0s

Se traducen las palabras con **replace({})**

```
#Traducimos los valores de la columna Región
df['Región']=df['Región'].replace({'North America':'Norteamérica', 'Europe':'Europa', 'South Ameri
#Traducimos los valores de la columna Tipo_de_Vehículo
df['Tipo_de_Vehículo']=df['Tipo_de_Vehículo'].replace({'Sedan':'Sedán', 'Truck':'Camioneta'})
#Traducimos los valores de la columna Tipo_de_Uso
df['Tipo_de_Uso']=df['Tipo_de_Uso'].replace({'Fleet': 'Flota', 'Commercial':'Comercial', })
```

✓ 0.0s

Cambio de nombres de columnas para mayor claridad o consistencia

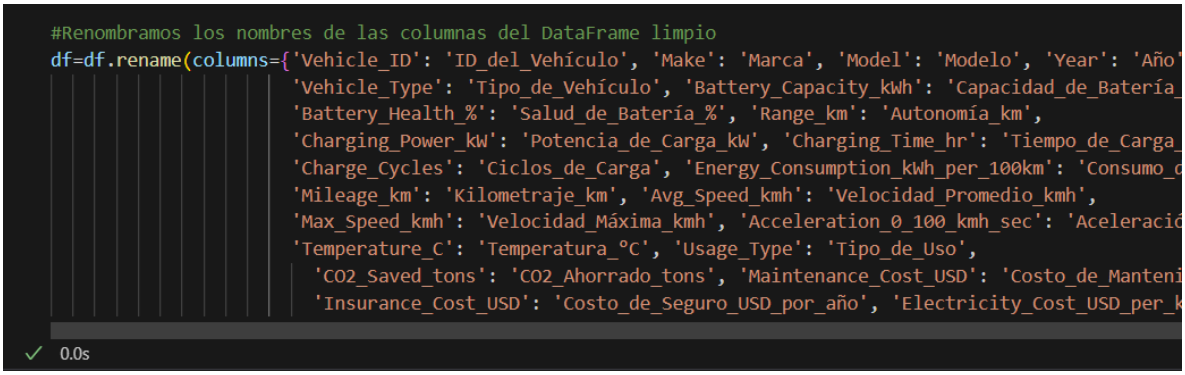
Se ocupa **columns** para ver cada columna y su nombre.

```
#Vemos la cantidad de valores únicos en la columna
columnas=df.columns
columnas
```

✓ 0.0s

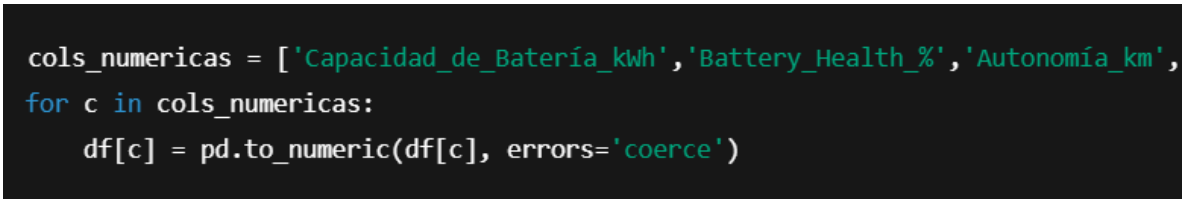
```
Index(['Vehicle_ID', 'Make', 'Model', 'Year', 'Region', 'Vehicle_Type',
       'Battery_Capacity_kWh', 'Battery_Health_%', 'Range_km',
       'Charging_Power_kW', 'Charging_Time_hr', 'Charge_Cycles',
       'Energy_Consumption_kWh_per_100km', 'Mileage_km', 'Avg_Speed_kmh',
       'Max_Speed_kmh', 'Acceleration_0_100_kmh_sec', 'Temperature_C',
       'Usage_Type', 'CO2_Saved_tons', 'Maintenance_Cost_USD',
       'Insurance_Cost_USD', 'Electricity_Cost_USD_per_kWh',
       'Monthly_Charging_Cost_USD', 'Resale_Value_USD'],
      dtype='object')
```

Se ocupa **rename(columns={})** para renombrar las columnas debido a que todas se encontraban en inglés.

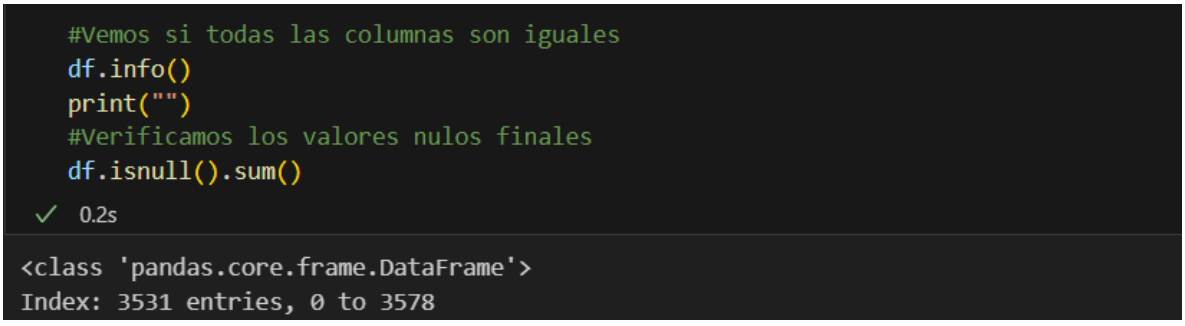


Conversión de tipos de datos

- Las columnas que eran object se convirtieron a float64 donde correspondía; los valores no convertibles quedaron NaN y se trataron (ver 2.3).
- Tras la conversión e imputación la mayoría de las columnas numéricas quedaron en tipo numérico. En la salida final (validación) se registraron los tipos float64 para capacidad_bateria_kwh, salud_bateria_pct, autonomia_km, potencia_carga_kw, etc.



Validación final mostrando que la base quedó limpia y coherente



- **Filas finales:** 3,531 (se eliminaron 48 duplicados exactos).
- **Columnas:** original 25 (se conservaron copias originales en columnas *_orig durante el proceso como evidencia; finalmente

puedes borrarlas con df.drop(columns=[...], inplace=True) si lo deseas).
    - **Duplicados:** 0 (tras drop_duplicates()).
    - **Valores nulos:** Se imputaron; en la validación final no quedaron valores NaN detectables (o quedaron muy pocos según la estrategia), ver final_nulos (vacío en el resumen final que generé).
    - **Tipos de datos:** columnas relevantes convertidas a float64 (numéricas) y columnas categóricas a object/texto.
    - **Imputaciones:** se usó mediana para variables numéricas con NaN y eliminación simple para duplicados. En el diccionario imputed_summary del notebook está registrado cuántas imputaciones se hicieron por variable y la mediana aplicada.

Conclusiones

Problemas principales que presentaba la base

- **Duplicados exactos (47 registros):**
    - El dataset contenía 47 filas idénticas a otras. Estos duplicados inflan conteos y sesgan estadísticas descriptivas (medias, totales), por lo que su presencia afecta la representatividad de resultados.

- **Tipos de datos incorrectos:**
    - Varias columnas numéricas habían sido importadas como object (texto). Esto impide realizar operaciones aritméticas y produce resultados erróneos en análisis exploratorio y modelado.

- **Valores no numéricos en columnas numéricas:**
    - Strings como 'Desconocido' o 'Ver4$zul' aparecen en columnas que deberían ser numéricas (p. ej. Charging_Time_hr, Charging_Power_kW, Resale_Value_USD). Al coercearlas a numérico, se transforman en NaN.

- **Datos contaminados / palabras extrañas:**
    - La cadena 'Ver4$zul' apareció repetidamente en múltiples columnas, indicando una contaminación de origen (etiqueta malformada, error de merge o finde línea).

- **Outliers / valores no plausibles:**

- Algunas variables (por ejemplo Charging_Time_hr) presentaron muchos valores fuera del rango intercuartílico (IQR), lo que puede indicar errores de medición, unidades equivocadas o datos mal escritos.

- **Inconsistencias semánticas:**
  - Texto con mayúsculas/espacios inconsistentes (ej.: ' Tesla ', 'tesla', 'TESLA') y categorías heterogéneas en Vehicle_Type o Usage_Type que deben normalizarse.

Técnicas aplicadas para solucionarlos

Para cada problema aplicamos técnicas estándar y justificadas:

1. **Duplicados**
   - Detectados con df.duplicated() y listados con df[df.duplicated(keep=False)].
   - **Decisión aplicada:** eliminar duplicados exactos con df.drop_duplicates() porque los registros eran copias idénticas y no representaban observaciones distintas. Esto garantiza que los conteos y promedios no estén inflados. (Si se necesitara preservar trazabilidad, se pueden mover los duplicados a un archivo de auditoría antes de eliminar).
2. **Tipos de datos y coerción**
   - Convertimos columnas numéricas con pd.to_numeric(..., errors='coerce') para detectar entradas problemáticas.
   - Mantuvimos copias originales (columnas *_orig) como evidencia antes de sobrescribir.
   - Después de la conversión, se imputaron NaN resultantes (ver abajo).
3. **Valores no numéricos y palabras extrañas**
   - Detectados mediante búsquedas de caracteres no alfabéticos y revisiones manuales (weird_values).
   - Reemplazos puntuales (p. ej. 'Ver4$zul' -> NaN o -> 'Versátil' si se confirma significado) y normalización de texto (str.strip(), str.title()).
4. **Outliers**

- Identificados con criterio IQR (cálculo de Q1, Q3 y límites).
- Acción aplicada: marcar los no plausibles como NaN para no dejar valores erróneos; luego imputación por mediana para análisis global o revisión caso por caso si el outlier puede ser válido (por ejemplo, vehículos de prueba con autonomía mayor a lo esperado).

5. **Imputación**
    - Para variables numéricas con valores faltantes derivados de la coerción o de marcadores ('Desconocido'), se aplicó **imputación por mediana**, por ser robusta frente a outliers:
    - df[col].fillna(df[col].median(), inplace=True)
6. **Normalización y renombrado**
    - Homogeneización de texto (trim y case), y renombrado de columnas a snake_case para facilitar reproducibilidad y evitar errores de referencia en código.

Aprendizaje del proceso

1. **Una "base limpia" en apariencia puede esconder problemas estructurales.**
    - El archivo tenía 0 nulos visibles en una inspección superficial, pero contenía cadenas no numéricas y valores extraños que solo se detectan al forzar tipos. Por eso la limpieza debe ser rigurosa y en fases: exploración → coerción → tratamiento.
2. **La importancia de conservar evidencia y trazabilidad.**
    - Mantener columnas *_orig o un archivo con los registros eliminados/transformados permite justificar las decisiones de limpieza ante el profesor o en un equipo de trabajo.
3. **Decisiones de limpieza requieren criterio de dominio.**
    - No siempre es correcto eliminar: por ejemplo, duplicados pueden ser mediciones válidas si cada fila representa una observación temporal distinta. En este dataset, dada la estructura (marca, modelo, año, economía) y la presencia de un ID, se concluyó que las duplicaciones eran errores de carga y fue preferible eliminarlas.
4. **Imputación es una solución práctica pero con límite.**

- Imputar por mediana es adecuado para conservar masa de datos al realizar análisis descriptivos y modelado inicial, pero para análisis críticos o inferenciales conviene investigar caso a caso o usar imputaciones más sofisticadas (k-NN, MICE) o excluír registros problemáticos si se justifica.