USO DE EDA PARA PROCESAMIENTO DE DATOS

El análisis de datos exploratorios o (EDA) consiste en comprender los conjuntos de datos resumiendo sus características principales y, a menudo, representándolos visualmente. Este paso es muy importante especialmente cuando llegamos a modelar los datos para aplicar el aprendizaje automático. El trazado en EDA consta de histogramas, diagramas de caja, diagramas de dispersión y muchos más. A menudo lleva mucho tiempo explorar los datos. A través del proceso de EDA, podemos pedir definir el planteamiento o definición del problema en nuestro conjunto de datos, lo cual es muy importante.

¿Cómo realizar un análisis de datos exploratorios?

Esta es una de esas preguntas cuya respuesta todo el mundo está interesado. Bueno, la respuesta es que depende del conjunto de datos con el que estés trabajando. No existe un método único o métodos comunes para realizar EDA, mientras que en este tutorial puede comprender algunos métodos y gráficos comunes que se usarían en el proceso EDA.

¿Qué datos estamos explorando hoy?

Como soy un gran admirador de los autos, obtuve un conjunto de datos de autos muy hermoso de Kaggle. El conjunto de datos se puede descargar desde aquí. Para brindar una breve información sobre el conjunto de datos, estos datos contienen más de 10, 000 filas y más de 10 columnas que contienen características del automóvil, como tipo de combustible del motor, HP del motor, tipo de transmisión, MPG en carretera, MPG en ciudad y muchos más. más. Entonces, en este tutorial, exploraremos los datos y los prepararemos para el modelado.

https://www.kaggle.com/datasets/CooperUnion/cardataset

1. Importando las bibliotecas requeridas para EDA

A continuación se muestran las bibliotecas que se utilizan para realizar EDA (análisis de datos exploratorios) en este tutorial.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(color_codes=True)
```

2. Cargando los datos en el marco de datos.

Cargar los datos en el marco de datos de pandas es sin duda uno de los pasos más importantes en EDA, ya que podemos ver que el valor del conjunto de datos está separado por comas. Entonces, todo lo que tenemos que hacer es leer el CSV en un marco de datos y el marco de datos de pandas hará el trabajo por nosotros.

Para obtener o cargar el conjunto de datos en el cuaderno, todo lo que hice fue un paso trivial. En Google Colab, en el lado izquierdo del cuaderno, encontrará un > (símbolo mayor que). Al hacer clic en eso encontrarás una pestaña con tres opciones, solo debes seleccionar Archivos. Luego podrá cargar fácilmente su archivo con la ayuda de la opción Cargar. No es necesario montarlo en Google Drive ni utilizar bibliotecas específicas, simplemente cargue el conjunto de datos y su trabajo estará listo. Una cosa para recordar en este paso es que los archivos cargados se eliminarán cuando se recicle este tiempo de ejecución. Así es como introduce los datos en el cuaderno.

[10]:
```
df = pd.read_csv("data.csv")
# To display the top 5 rows
df.head(4)
```

[10]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSR
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	4613
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	4065
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	3635
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	2945

[7]:
```
df.tail(5)
```

[7]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16
11913	Lincoln	Zephyr	2006	regular unleaded	221.0	6.0	AUTOMATIC	front wheel drive	4.0	Luxury	Midsize	Sedan	26	17

3. Comprobación de los tipos de datos

Aquí verificamos los tipos de datos porque a veces el MSRP o el precio del automóvil se almacenan como una cadena, si en ese caso tenemos que convertir esa cadena a datos enteros, solo entonces podemos trazar los datos a través de un gráfico. Aquí, en este caso, los datos ya están en formato entero, así que no hay de qué preocuparse.

```
[11]: df.dtypes

[11]: Make                 object
      Model                object
      Year                  int64
      Engine Fuel Type     object
      Engine HP           float64
      Engine Cylinders    float64
      Transmission Type    object
      Driven_Wheels        object
      Number of Doors     float64
      Market Category      object
      Vehicle Size         object
      Vehicle Style        object
      highway MPG           int64
      city mpg              int64
      Popularity            int64
      MSRP                  int64
      dtype: object
```

4. Eliminar columnas irrelevantes

Este paso ciertamente es necesario en cada EDA porque a veces habrá muchas columnas que nunca usaremos, en tales casos eliminarlas es la única solución. En este caso, las columnas como Tipo de combustible del motor, Categoría de mercado, Estilo de vehículo, Popularidad, Número de puertas y Tamaño del vehículo no tienen ningún sentido para mí, así que las eliminé para este caso.

```
[12]: df = df.drop(['Engine Fuel Type', 'Market Category', 'Vehicle Style', 'Popularity', 'Number of Doors', 'Vehicle Size'], axis=1)
      df.head(5)
```

[12]:

	Make	Model	Year	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	highway MPG	city mpg	MSRP
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

5. Cambiar el nombre de las columnas

En este caso, la mayoría de los nombres de las columnas son muy confusos de leer, así que simplemente modifiqué los nombres de las columnas. Este es un buen enfoque, mejora la legibilidad del conjunto de datos.

```
[13]: df = df.rename(columns={"Engine HP": "HP", "Engine Cylinders": "Cylinders", "Transmission Type": "Transmission", "Driven_Wheels": "Drive Mode"
      df.head(5)
```

[13]:

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

6. Eliminar las filas duplicadas

Esto suele ser algo útil porque un conjunto de datos enorme, como en este caso, contiene más de 10.000 filas, a menudo tiene algunos datos duplicados que pueden resultar perturbadores, por lo que aquí elimino todos los valores duplicados del conjunto de datos. Por ejemplo, antes de eliminar tenía 11914 filas de datos, pero después de eliminar los duplicados, 10925 datos, lo que significa que tenía 989 datos duplicados.

```
[14]: df.shape
```

```
[14]: (11914, 10)
```

```
[15]: duplicate_rows_df = df[df.duplicated()]
print("number of duplicate rows: ", duplicate_rows_df.shape)
```

```
number of duplicate rows:  (989, 10)
```

Ahora eliminemos los datos duplicados porque está bien eliminarlos.

```
[16]: df.count()      # Used to count the number of rows
```

```
[16]: Make            11914
Model           11914
Year            11914
HP              11845
Cylinders       11884
Transmission    11914
Drive Mode      11914
MPG-H           11914
MPG-C           11914
Price           11914
dtype: int64
```

Como se ve arriba, hay 11914 filas y estamos eliminando 989 filas de datos duplicados.

```
[17]: df = df.drop_duplicates()
df.head(5)
```

[17]:

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

7. Eliminar los valores faltantes o nulos.

Esto es muy similar al paso anterior, pero aquí todos los valores faltantes se detectan y se eliminan más adelante. Ahora bien, este no es un buen enfoque para hacerlo, porque muchas personas simplemente reemplazan los valores faltantes con la media o el promedio de esa columna, pero en este caso, simplemente eliminé esos valores faltantes. Esto se debe a que faltan casi 100 valores en comparación con 10.000 valores. Este es un número pequeño y es insignificante, por lo que simplemente eliminé esos valores.

```
[19]: print(df.isnull().sum())
```

```
Make            0
Model           0
Year            0
HP             69
Cylinders      30
Transmission    0
Drive Mode      0
MPG-H           0
MPG-C           0
Price           0
dtype: int64
```

Esta es la razón por la que en el paso anterior, al contar tanto los cilindros como los caballos de fuerza (HP), se obtuvieron 10856 y 10895 en 10925 filas.

```
[20]: df = df.dropna()    # Dropping the missing values.
      df.count()

[20]: Make           10827
      Model          10827
      Year           10827
      HP             10827
      Cylinders      10827
      Transmission   10827
      Drive Mode     10827
      MPG-H          10827
      MPG-C          10827
      Price          10827
      dtype: int64
```

Collapse Output

Esta es la razón por la que en el paso anterior, al contar tanto los cilindros como los caballos de fuerza (HP), se obtuvieron 10856 y 10895 en 10925 filas.

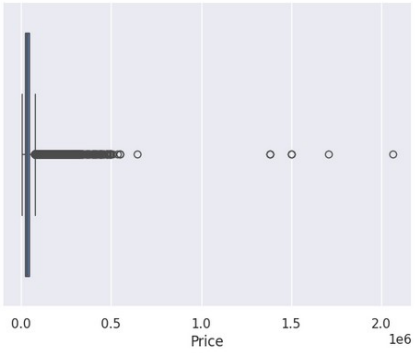
```
[21]: print(df.isnull().sum())   # After dropping the values

Make           0
Model          0
Year           0
HP             0
Cylinders      0
Transmission   0
Drive Mode     0
MPG-H          0
MPG-C          0
Price          0
```

Collapse Output

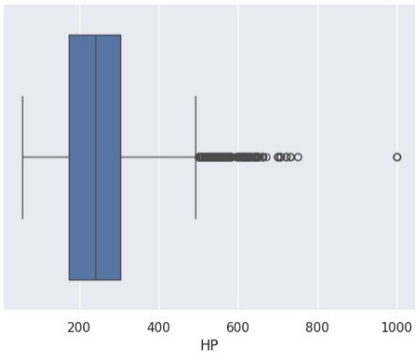
8. Detección de valores atípicos

Un valor atípico es un punto o conjunto de puntos que son diferentes de otros puntos. A veces pueden ser muy altos o muy bajos. A menudo es una buena idea detectar y eliminar los valores atípicos. Porque los valores atípicos son una de las principales razones por las que el resultado es un modelo menos preciso. Por eso es una buena idea eliminarlos. La detección y eliminación de valores atípicos que voy a realizar se llama técnica de puntuación IQR. A menudo, los valores atípicos se pueden ver con visualizaciones que utilizan un diagrama de caja. A continuación se muestra el diagrama de caja de MSRP, cilindros, caballos de fuerza y tamaño del motor. Aquí en todas las tramas, puede encontrar que algunos puntos están fuera del cuadro y no son más que valores atípicos. La técnica de encontrar y eliminar valores atípicos que estoy realizando en esta tarea se basa en un tutorial de ciencia de datos.

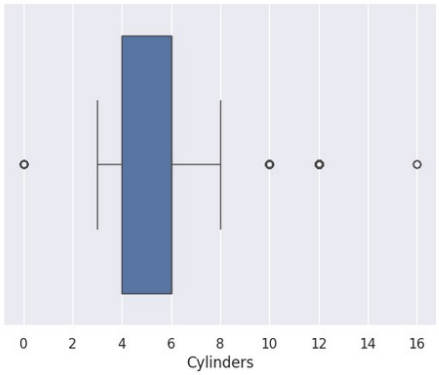
```
[22]: sns.boxplot(x=df['Price'])

[22]: <Axes: xlabel='Price'>
```



```
[23]: sns.boxplot(x=df['HP'])
[23]: <Axes: xlabel='HP'>
```



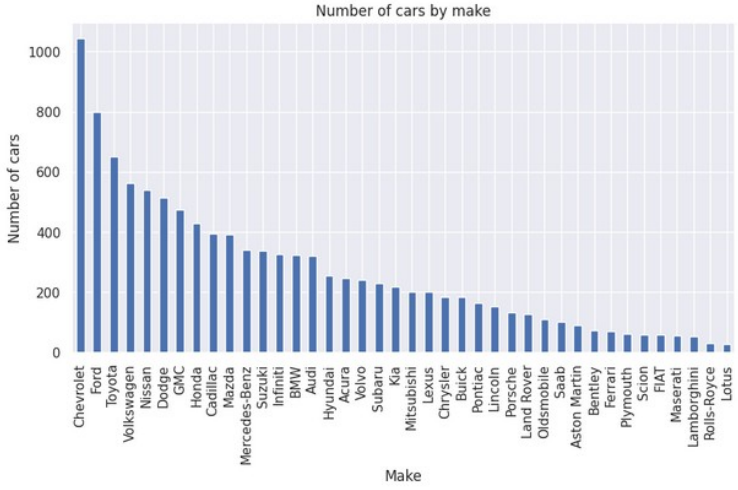

```
[26]: sns.boxplot(x=df['Cylinders'])
[26]: <Axes: xlabel='Cylinders'>
```



9. Trazar diferentes características entre sí (dispersión) y frecuencia

El histograma se refiere a la frecuencia de aparición de variables en un intervalo. En este caso, existen principalmente 10 tipos diferentes de empresas fabricantes de automóviles, pero a menudo es importante saber quién tiene la mayor cantidad de automóviles. Hacer este histograma es una de las soluciones triviales que nos permite saber el número total de automóviles fabricados por una empresa diferente.

```
[41]: df.Make.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make');
```



10. Conclusión

El análisis exploratorio de datos (EDA) desempeña un papel fundamental en el procesamiento de datos al proporcionar una visión detallada de la estructura, distribución y relaciones entre las variables en un conjunto de datos. En el contexto de este proyecto sobre fabricantes de automóviles, el EDA ha permitido comprender la diversidad de empresas presentes y la distribución de la cantidad de automóviles fabricados por cada una. Esto es crucial para identificar patrones, anomalías y tendencias significativas en los datos, lo que a su vez facilita la toma de decisiones informadas en la implementación de técnicas de inteligencia artificial. Además, al trazar características entre sí y examinar la frecuencia de variables, el EDA proporciona una base sólida para el preprocesamiento de datos, incluyendo la detección y manejo de valores atípicos, la selección de características relevantes y la normalización de datos, contribuyendo así a mejorar la calidad y eficacia de los modelos de aprendizaje automático que se desarrollarán posteriormente.

En el siguiente enlace de GitHub pueden encontrar la practica desarrollada para que puedan realizar el ejercicio.