What is Exploratory Data Analysis?

How to Perform EDA Using Python?

Step 1: Import Python Libraries

Step 2: Reading Dataset

Step 3: Data Reduction

Step 4: Feature Engineering

Step 5: Creating Features

Step 6: Data Cleaning/Wrangling

Step 7: EDA Exploratory Data Analysis

Step 8: Statistics Summary

Step 9: EDA Univariate Analysis

Step 10: Data Transformation

Step 12: EDA Bivariate Analysis

Step 13: EDA Multivariate Analysis

Step 14: Impute Missing values

1. Univariate Analysis

Univariate analysis focuses on studying one variable to understand its characteristics. It helps to describe data and find patterns within a single feature. Various common methods like histograms are used to show data distribution, box plots to detect

outliers and understand data spread and bar charts for categorical data. Summary statistics like mean, median, mode, variance and standard deviation helps in describing the central tendency and spread of the data

Mean → Average value

Median → Middle value

Mode → Most frequent value

Variance → How far values are from the mean (squared)

Standard deviation → Average distance from the mean (easier to understand)

Step 1: Import Python Libraries

The first step involved in ML using python is understanding and playing around with our data using libraries. Here is the link to the dataset.

Import all libraries required for our analysis, such as those for data loading, statistical analysis, visualizations, data transformations, and merging and joining.

Pandas and Numpy have been used for Data Manipulation and numerical Calculations

Matplotlib and Seaborn have been used for Data visualizations.

In [10]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#to ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

Step 2: Reading Dataset

The Pandas library offers a wide range of possibilities for loading data into the pandas DataFrame from files like .json, .csv, .xlsx, .sql, image file extensions etc.

Most of the data are available in a tabular format of CSV files. It is trendy and easy to access. Using the read_csv() function, data can be converted to a pandas DataFrame.

The data to predict used car price is being used as an example. In this dataset, we are trying to analyze the used car's price and how EDA focuses on identifying the factors influencing the car price. We have stored the data in the DataFrame data.

In [14]:
```
data = pd.read_csv("used_cars_data.csv")
```

Analyzing the Data

Before we make any inferences, we listen to our data by examining all variables in the data.

The main goal of data understanding is to gain general insights about the data, which covers the number of rows and columns, values in the data, datatypes, and Missing

values in the dataset.

shape – shape will display the number of observations(rows) and features(columns) in the dataset

There are 7253 observations and 14 variables in our dataset

head() will display the top 5 observations of the dataset

In [19]:
```
data.shape
```

Out[19]: (7253, 14)

In [21]:
```
data.head()
```

Out[21]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seat
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	5.
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.

In [25]:
```
data.tail()
```

Out[25]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	First	20.54 kmpl	1598 CC	103.6 bhp
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	First	17.21 kmpl	1197 CC	103.6 bhp
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	First	23.08 kmpl	1461 CC	63.1 bhp
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	Third	17.2 kmpl	1197 CC	103.6 bhp
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	First	10.0 kmpl	2148 CC	170 bhp

info() helps to understand the data type and information about data, including the number of records in each column, data having null or not null, Data type, the memory usage of the dataset

In [28]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   S.No.              7253 non-null   int64  
 1   Name               7253 non-null   object 
 2   Location           7253 non-null   object 
 3   Year               7253 non-null   int64  
 4   Kilometers_Driven  7253 non-null   int64  
 5   Fuel_Type          7253 non-null   object 
 6   Transmission       7253 non-null   object 
 7   Owner_Type         7253 non-null   object 
 8   Mileage            7251 non-null   object 
 9   Engine             7207 non-null   object 
 10  Power              7207 non-null   object 
 11  Seats              7200 non-null   float64
 12  New_Price          1006 non-null   object 
 13  Price              6019 non-null   float64
dtypes: float64(2), int64(3), object(9)
memory usage: 793.4+ KB
```

Check for Duplication

nunique() based on several unique values in each column and the data description, we can identify the continuous and categorical columns in the data. Duplicated data can be handled or removed based on further analysis

In [31]:
```
data.nunique()
```

Out[31]:
```
S.No.                7253
Name                 2041
Location               11
Year                   23
Kilometers_Driven    3660
Fuel_Type               5
Transmission            2
Owner_Type              4
Mileage               450
Engine                150
Power                 386
Seats                   9
New_Price             625
Price                1373
dtype: int64
```

Missing Values Calculation

isnull() is widely been in all pre-processing steps to identify null values in the data

In our example, data.isnull().sum() is used to get the number of missing records in each column

In [34]:
```
data.isnull().sum()
```

Out[34]:
```
S.No.                  0
Name                   0
Location               0
Year                   0
Kilometers_Driven      0
Fuel_Type              0
Transmission           0
Owner_Type             0
Mileage                2
Engine                46
Power                 46
Seats                 53
New_Price           6247
Price               1234
dtype: int64
```

In [36]:
```
# The below code helps to calculate the
#percentage of missing values in each column
(data.isnull().sum()/(len(data)))*100
```

Out[36]:
```
S.No.                0.000000
Name                 0.000000
Location             0.000000
Year                 0.000000
Kilometers_Driven    0.000000
Fuel_Type            0.000000
Transmission         0.000000
Owner_Type           0.000000
Mileage              0.027575
Engine               0.634220
Power                0.634220
Seats                0.730732
New_Price           86.129877
Price               17.013650
dtype: float64
```

Step 3: Data Reduction

Some columns or variables can be dropped if they do not add value to our analysis.

In our dataset, the column S.No have only ID values, assuming they don't have any predictive power to predict the dependent variable.

In [39]:
```
# Remove S.No. column from data
data = data.drop(['S.No.'], axis = 1)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               7253 non-null   object 
 1   Location           7253 non-null   object 
 2   Year               7253 non-null   int64  
 3   Kilometers_Driven  7253 non-null   int64  
 4   Fuel_Type          7253 non-null   object 
 5   Transmission       7253 non-null   object 
 6   Owner_Type         7253 non-null   object 
 7   Mileage            7251 non-null   object 
 8   Engine             7207 non-null   object 
 9   Power              7207 non-null   object 
 10  Seats              7200 non-null   float64
 11  New_Price          1006 non-null   object 
 12  Price              6019 non-null   float64
dtypes: float64(2), int64(2), object(9)
memory usage: 736.8+ KB
```

Step 4: Feature Engineering

Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling. The main goal of Feature

engineering is to create meaningful data from raw data.

Step 5: Creating Features

We will play around with the variables Year and Name in our dataset. If we see the sample data, the column “Year” shows the manufacturing year of the car.

It would be difficult to find the car’s age if it is in year format as the Age of the car is a contributing factor to Car Price.

Introducing a new column, “Car_Age” to know the age of the car

In [43]:
```
from datetime import date
date.today().year
data['Car_Age']=date.today().year-data['Year']
data.head()
```

Out[43]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	5.0	8.6
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.0	

In [45]:
```
data['Brand'] = data.Name.str.split().str.get(0)
```

In [47]:
```
data['Model'] = data.Name.str.split().str.get(1) + data.Name.str.split().str.get(2)
```

In [49]:
```
data[['Name','Brand','Model']]
```

Out[49]:

	Name	Brand	Model
0	Maruti Wagon R LXI CNG	Maruti	WagonR
1	Hyundai Creta 1.6 CRDi SX Option	Hyundai	Creta1.6
2	Honda Jazz V	Honda	JazzV
3	Maruti Ertiga VDI	Maruti	ErtigaVDI
4	Audi A4 New 2.0 TDI Multitronic	Audi	A4New
...	...	...	...
7248	Volkswagen Vento Diesel Trendline	Volkswagen	VentoDiesel
7249	Volkswagen Polo GT TSI	Volkswagen	PoloGT
7250	Nissan Micra Diesel XV	Nissan	MicraDiesel
7251	Volkswagen Polo GT TSI	Volkswagen	PoloGT
7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Mercedes-Benz	E-Class2009-2013

7253 rows × 3 columns

Step 6: Data Cleaning/Wrangling

Some names of the variables are not relevant and not easy to understand. Some data may have data entry errors, and some variables may need data type conversion. We need to fix this issue in the data.

In the example, The brand name ‘Isuzu’ ‘ISUZU’ and ‘Mini’ and ‘Land’ looks incorrect. This needs to be corrected

In [52]:
```
print(data.Brand.unique()) # unique() = what values,
#nunique() = how many values
print(data.Brand.nunique())
```

```
['Maruti' 'Hyundai' 'Honda' 'Audi' 'Nissan' 'Toyota' 'Volkswagen' 'Tata'
 'Land' 'Mitsubishi' 'Renault' 'Mercedes-Benz' 'BMW' 'Mahindra' 'Ford'
 'Porsche' 'Datsun' 'Jaguar' 'Volvo' 'Chevrolet' 'Skoda' 'Mini' 'Fiat'
 'Jeep' 'Smart' 'Ambassador' 'Isuzu' 'ISUZU' 'Force' 'Bentley'
 'Lamborghini' 'Hindustan' 'OpelCorsa']
33
```

In [54]:
```
searchfor = ['Isuzu' ,'ISUZU','Mini','Land']
data[data.Brand.str.contains('|'.join(searchfor))].head(5)
```

Out[54]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats
13	Land Rover Range Rover 2.2L Pure	Delhi	2014	72000	Diesel	Automatic	First	12.7 kmpl	2179 CC	187.7 bhp	5.0
14	Land Rover Freelander 2 TD4 SE	Pune	2012	85000	Diesel	Automatic	Second	0.0 kmpl	2179 CC	115 bhp	5.0
176	Mini Countryman Cooper D	Jaipur	2017	8525	Diesel	Automatic	Second	16.6 kmpl	1998 CC	112 bhp	5.0
191	Land Rover Range Rover 2.2L Dynamic	Coimbatore	2018	36091	Diesel	Automatic	First	12.7 kmpl	2179 CC	187.7 bhp	5.0
228	Mini Cooper Convertible S	Kochi	2017	26327	Petrol	Automatic	First	16.82 kmpl	1998 CC	189.08 bhp	4.0

Step 7: EDA Exploratory Data Analysis

Exploratory Data Analysis refers to the crucial process of performing initial investigations on data to discover patterns to check assumptions with the help of summary statistics and graphical representations.

EDA can be leveraged to check for outliers, patterns, and trends in the given data.

EDA helps to find meaningful patterns in data.

EDA provides in-depth insights into the data sets to solve our business problems.

EDA gives a clue to impute missing values in the dataset

Step 8: Statistics Summary

The information gives a quick and simple description of the data.

Can include Count, Mean, Standard Deviation, median, mode, minimum value, maximum value, range, standard deviation, etc.

Statistics summary gives a high-level idea to identify whether the data has any outliers, data entry error, distribution of data such as the data is normally distributed or left/right skewed

In python, this can be achieved using describe()

describe()– Provide a statistics summary of data belonging to numerical datatype such as int, float

In [58]:
```
data.describe().T
```

Out[58]:

	count	mean	std	min	25%	50%	75%	max
Year	7253.0	2013.365366	3.254421	1996.00	2011.0	2014.00	2016.00	2019.0
Kilometers_Driven	7253.0	58699.063146	84427.720583	171.00	34000.0	53416.00	73000.00	6500000.0
Seats	7200.0	5.279722	0.811660	0.00	5.0	5.00	5.00	10.0
Price	6019.0	9.479468	11.187917	0.44	3.5	5.64	9.95	160.0
Car_Age	7253.0	12.634634	3.254421	7.00	10.0	12.00	15.00	30.0

From the statistics summary, we can infer the below findings :

Years range from 1996- 2019 and has a high in a range which shows used cars contain both latest models and old model cars.

On average of Kilometers-driven in Used cars are ~58k KM. The range shows a huge difference between min and max as max values show 650000 KM shows the evidence of an outlier. This record can be removed.

Min value of Mileage shows 0 cars won't be sold with 0 mileage. This sounds like a data entry issue.

It looks like Engine and Power have outliers, and the data is right-skewed.

The average number of seats in a car is 5. car seat is an important feature in price contribution.

The max price of a used car is 160k which is quite weird, such a high price for used cars. There may be an outlier or data entry issue.

describe(include='all') provides a statistics summary of all data, include object, category etc

In [ ]: