EXPLORATORY DATA ANALYSIS

BY

SURESH KUMAR

WHAT IS EXPLORATORY DATA ANALYSIS

- Exploratory Data Analysis (EDA) is an essential step in the data analysis process that involves examining and summarizing datasets to better understand their underlying structure, patterns, and relationships. The primary goal of EDA is to gain insights into the data, identify potential issues, and develop a deeper understanding of the data's characteristics.

What data are we exploring ?

- I am very facsinate about cars, I got a very beautiful data-set of Toyoto from NPTEL. To give a piece of brief information about the data set this data contains more of 1436 rows and more than 11columns which contains features of the car such as Engine Fuel Type, Engine Size, HP, Doors, KM, CC and many more. So in this tutorial, we will explore the data and make it ready for modeling.

Let's get started !!!

1. Importing the Required Libraries

- import pandas as pd
- import numpy as np
- import matplotlib.pyplot as plt
- import seaborn as sns

Loading the Data into the Data Frame.

- df = pd.read_csv("F:/carsdata/Toyota (1).csv")

"Importing data into a pandas DataFrame is a crucial step in Exploratory Data Analysis (EDA), especially when dealing with comma-separated values. Fortunately, pandas provides a straightforward way to read CSV files into a DataFrame, making it easy to load and manipulate the data."

To display the top 5 rows

- cars=df.copy()
- cars.head(5)
- Out[5]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic
0	0	13500	23.0	46986	Diesel	90	1.0	0
1	1	13750	23.0	72937	Diesel	90	1.0	0
3	3	14950	26.0	48000	Diesel	90	0.0	0
4	4	13750	30.0	38500	Diesel	90	0.0	0
5	5	12950	32.0	61000	Diesel	90	0.0	0

To display the bottom 5 rows

- cars.tail(5)
- Out[7]:
- Unnamed: 0 Price Age KM FuelType HP MetColor Automatic
- 1425 1425 7950 80.0 ?? Petrol 86 1.0 0
- 1429 1429 8950 78.0 24000 Petrol 86 1.0 1
- 1430 1430 8450 80.0 23000 Petrol 86 0.0 0
- 1432 1432 10845 72.0 ?? Petrol 86 0.0 0
- 1435 1435 6950 76.0 1 Petrol 110 0.0 0

Checking the types of data

- cars.dtypes
- Out[8]:
- Unnamed: 0 int64
- Price int64
- Age float64
- KM object
- FuelType object
- HP object
- MetColor float64
- Automatic int64
- dtype: object

Dropping irrelevant columns

- cars=cars.drop(['Automatic'], axis=1)

- cars.head(5)
- Out[11]:
- Unnamed: 0 Price Age KM FuelType HP MetColor
- 0 0 13500 23.0 46986 Diesel 90 1.0
- 1 1 13750 23.0 72937 Diesel 90 1.0
- 3 3 14950 26.0 48000 Diesel 90 0.0
- 4 4 13750 30.0 38500 Diesel 90 0.0
- 5 5 12950 32.0 61000 Diesel 90 0.0

Renaming the column names

- cars=cars.rename(columns={"Price":"cost"})

- cars.head(5)
- Out[17]:
- Unnamed: 0 cost Age KM FuelType HP MetColor
- 0 0 13500 23.0 46986 Diesel 90 1.0
- 1 1 13750 23.0 72937 Diesel 90 1.0
- 3 3 14950 26.0 48000 Diesel 90 0.0
- 4 4 13750 30.0 38500 Diesel 90 0.0
- 5 5 12950 32.0 61000 Diesel 90 0.0

Total number of rows and columns and # Rows containing duplicate data

- cars.shape
- Out[18]: (1111, 7)
- duplicate_rows_cars = cars[cars.duplicated()]
- print("number of duplicate rows: ", duplicate_rows_cars.shape)
- number of duplicate rows: (0, 7)

Used to count the number of rows before removing the data

- cars.count()
- Out[22]:
- Unnamed: 0 1111
- cost 1111
- Age 1111
- KM 1111
- FuelType 1111
- HP 1111
- MetColor 1111
- dtype: int64

Dropping the duplicates

- cars=cars.drop_duplicates()

- cars.head(5)
- Out[25]:
- Unnamed: 0 cost Age KM FuelType HP MetColor
- 0 0 13500 23.0 46986 Diesel 90 1.0
- 1 1 13750 23.0 72937 Diesel 90 1.0
- 3 3 14950 26.0 48000 Diesel 90 0.0
- 4 4 13750 30.0 38500 Diesel 90 0.0
- 5 5 12950 32.0 61000 Diesel 90 0.0

Counting the number of rows after removing duplicates.

- cars.count()
- Out[26]:
- Unnamed: 0 1111
- cost 1111
- Age 1111
- KM 1111
- FuelType 1111
- HP 1111
- MetColor 1111
- dtype: int64

Finding the null values..

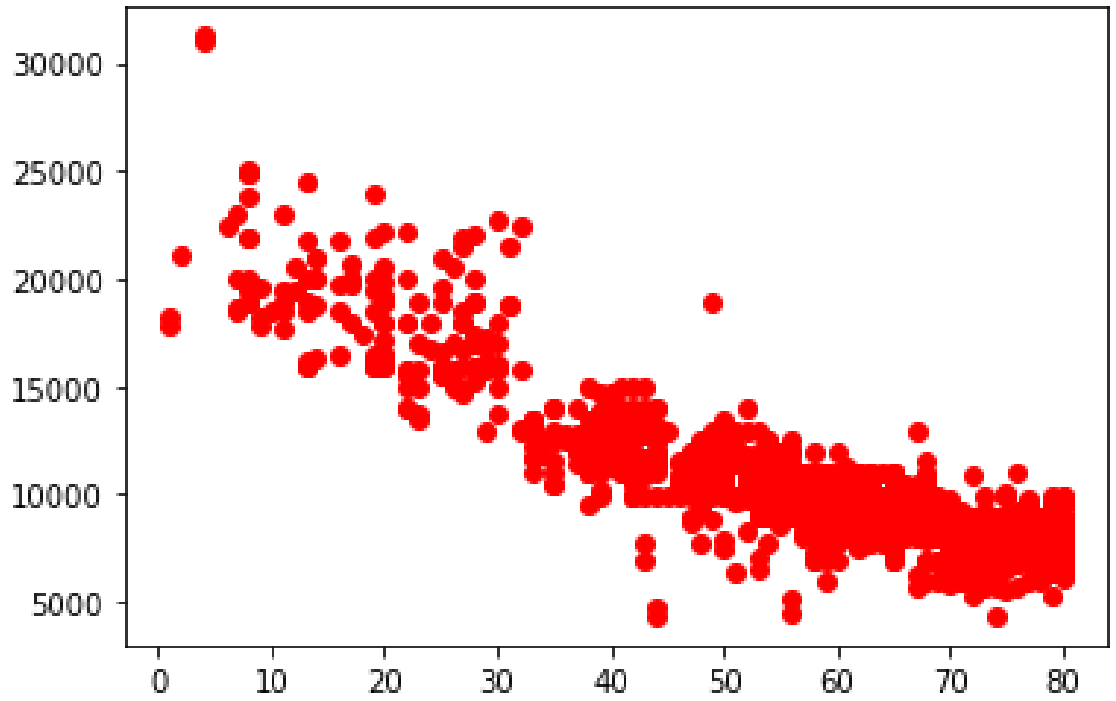
- print(cars.isnull().sum())
- Unnamed: 0   0
- cost   0
- Age   0
- KM   0
- FuelType   0
- HP   0
- MetColor   0
- dtype: int64

Dropping the missing values.

- cars = cars.dropna()
- cars.count()
- Out[28]:
- Unnamed: 0   1111
- cost   1111
- Age   1111
- KM   1111
- FuelType   1111
- HP   1111
- MetColor   1111
- dtype: int64

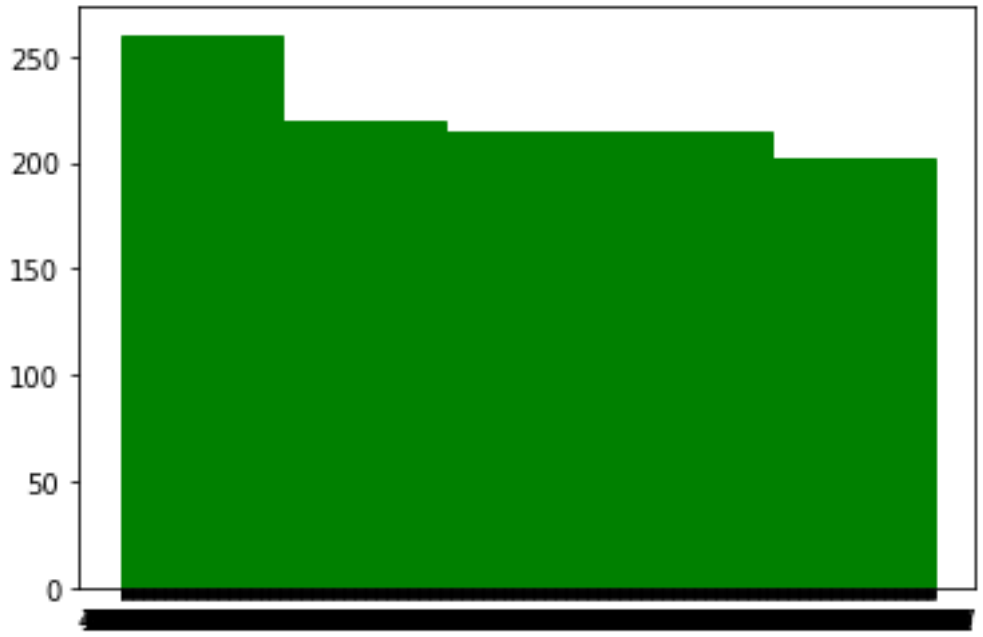
Representation of Scatter Plot [CARS(age vs cost)]

- plt.scatter(cars['Age'], cars['cost'],c='red')
- Out[31]: <matplotlib.collections.PathCollection at 0x22fbb7ce890>



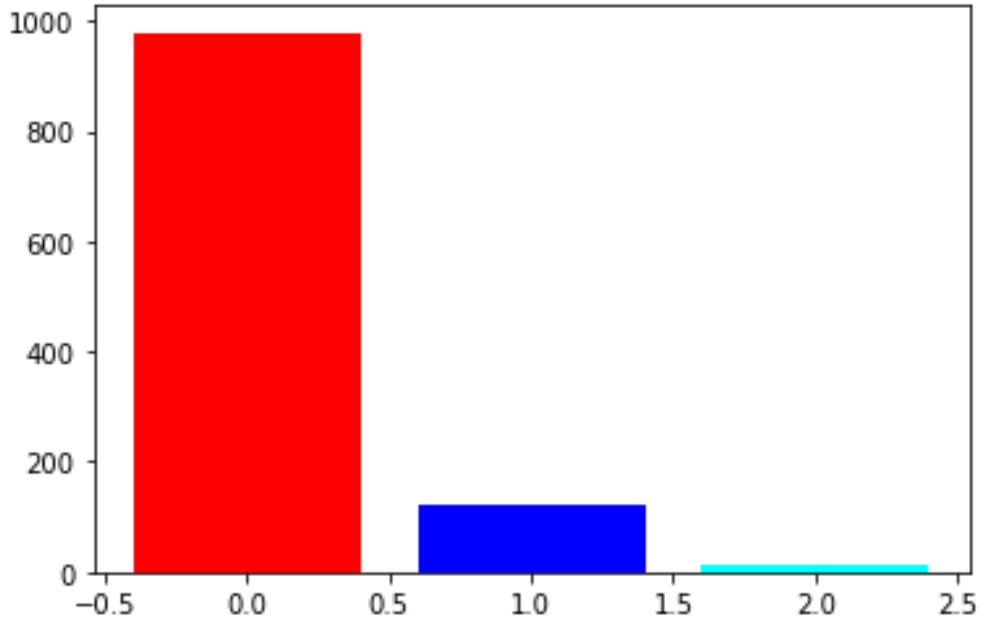
Representation Histogram (KM vs Frequency)

- plt.title('histogram of kilometer')
- plt.xlabel('kilometer')
- plt.ylabel('frequency')
- plt.show()
- plt.hist(cars['KM'],color='green',edgecolor='green',bins=5)
- Out[32]:
- (array([260., 220., 215., 214., 202.]),
- array([0., 198., 396., 594., 792., 990.]),
- <BarContainer object of 5 artists>)



Representation Bargraph

- counts=[979,120,12]
- FuelType=('petrol','diesel','cng')
- index=np.arange(len(FuelType))
- plt.bar(index,counts,color=['red','blue','cyan'])
- Out[33]: <BarContainer object of 3 artists>



- *This is the Exploratory Data Analysis About Toyota cars.*

THANK YOU

BY

DONKADA SURESH KUMAR