Problem Statement:

We have used Cars dataset from kaggle with features including make, model, year, engine, and other properties of the car used to predict its price.

Importing the necessary libraries

```
%pip install seaborn
%pip install matplotlib
%pip install scikit-learn
```

```
Requirement already satisfied: seaborn in c:\users\laksh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\laksh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.25 in c:\users\laksh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from seaborn) (2.1.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\laksh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from seaborn) (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\laksh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\laksh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\laksh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.56.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\laksh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in c:\users\laksh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from matplotlib!=3.6.1,>=3.1-
```

```
>seaborn) (24.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from matplotlib!=3.6.1,>=3.1-
>seaborn) (10.1.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\
laksh\appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from matplotlib!=3.6.1,>=3.1-
>seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from matplotlib!=3.6.1,>=3.1-
>seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\laksh\appdata\
local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\
localcache\local-packages\python311\site-packages (from pandas>=0.25-
>seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.1 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from pandas>=0.25->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\laksh\appdata\
local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\
localcache\local-packages\python311\site-packages (from python-
dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.17.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: C:\Users\laksh\AppData\Local\Microsoft\
WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\
python.exe -m pip install --upgrade pip

Requirement already satisfied: matplotlib in c:\users\laksh\appdata\
local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\
localcache\local-packages\python311\site-packages (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\laksh\appdata\
local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\
localcache\local-packages\python311\site-packages (from matplotlib)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
```

```
packages\python311\site-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.20 in c:\users\laksh\appdata\
local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\
localcache\local-packages\python311\site-packages (from matplotlib)
(1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from matplotlib) (10.1.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\
laksh\appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\laksh\appdata\
local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\
localcache\local-packages\python311\site-packages (from python-
dateutil>=2.7->matplotlib) (1.17.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: C:\Users\laksh\AppData\Local\Microsoft\
WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\
python.exe -m pip install --upgrade pip

Requirement already satisfied: scikit-learn in c:\users\laksh\appdata\
local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\
localcache\local-packages\python311\site-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\laksh\appdata\
local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\
localcache\local-packages\python311\site-packages (from scikit-learn)
(1.15.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\laksh\
appdata\local\packages\
```

```
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\laksh\
appdata\local\packages\
pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-
packages\python311\site-packages (from scikit-learn) (3.6.0)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: C:\Users\laksh\AppData\Local\Microsoft\
WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\
python.exe -m pip install --upgrade pip
```

```
import pandas as pd
import numpy as np
import seaborn as sns #visualisation
import matplotlib.pyplot as plt #visualisation
%matplotlib inline
sns.set(color_codes=True)
from scipy import stats
import warnings
warnings.filterwarnings("ignore")
```

Load the dataset into dataframe

```
## load the csv file
df =pd.read_csv('Cars_data.csv')

## print the head of the dataframe
print(df.head())
```

```
  Make       Model  Year             Engine Fuel Type  Engine HP  \
0  BMW  1 Series M  2011  premium unleaded (required)      335.0
1  BMW    1 Series  2011  premium unleaded (required)      300.0
2  BMW    1 Series  2011  premium unleaded (required)      300.0
3  BMW    1 Series  2011  premium unleaded (required)      230.0
4  BMW    1 Series  2011  premium unleaded (required)      230.0

   Engine Cylinders Transmission Type     Driven_Wheels  Number of
Doors  \
0               6.0            MANUAL  rear wheel drive
2.0
1               6.0            MANUAL  rear wheel drive
2.0
2               6.0            MANUAL  rear wheel drive
2.0
3               6.0            MANUAL  rear wheel drive
2.0
4               6.0            MANUAL  rear wheel drive
2.0
```

```
                       Market Category Vehicle Size Vehicle Style  \
0  Factory Tuner,Luxury,High-Performance      Compact         Coupe
1                     Luxury,Performance      Compact   Convertible
2                Luxury,High-Performance      Compact         Coupe
3                     Luxury,Performance      Compact         Coupe
4                                 Luxury      Compact   Convertible

   highway MPG  city mpg  Popularity   MSRP
0           26        19        3916  46135
1           28        19        3916  40650
2           28        20        3916  36350
3           28        18        3916  29450
4           28        18        3916  34500
```

Now we observe the each features present in the dataset.

`Make:` The Make feature is the company name of the Car. `Model:` The Model feature is the model or different version of Car models. `Year:` The year describes the model has been launched. `Engine Fuel Type:` It defines the Fuel type of the car model. `Engine HP:` It's say the Horsepower that refers to the power an engine produces. `Engine Cylinders:` It define the nos of cylinders in present in the engine. `Transmission Type:` It is the type of feature that describe about the car transmission type i.e Mannual or automatic. `Driven_Wheels:` The type of wheel drive. `No of doors:` It defined nos of doors present in the car. `Market Category:` This features tells about the type of car or which category the car belongs. `Vehicle Size:` It's say about the about car size. `Vehicle Style:` The feature is all about the style that belongs to car. `highway MPG:` The average a car will get while driving on an open stretch of road without stopping or starting, typically at a higher speed. `city mpg:` City MPG refers to driving with occasional stopping and braking. `Popularity:` It can refered to rating of that car or popularity of car. `MSRP:` The price of that car.

`Check the datatypes`

```
# Get the datatypes of each columns number of records in each column.
df.dtypes
```

```
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
```

`Dropping irrevalent columns`

If we consider all columns present in the dataset then unneccessary columns will impact on the model's accuracy. Not all the columns are important to us in the given dataframe, and hence we would drop the columns that are irrevalent to us. It would reflect our model's accucary so we need to drop them. Otherwise it will affect our model.

The list cols_to_drop contains the names of the cols that are irrevalent, drop all these cols from the dataframe.

`cols_to_drop = ["Engine Fuel Type", "Market Category", "Vehicle Style", "Popularity", "Number of Doors", "Vehicle Size"]`

These features are not neccessary to obtain the model's accucary. It does not contain any relevant information in the dataset.

```
# initialise cols_to_drop
col_to_drop = ["Engine Fuel Type", "Market Category", "Vehicle Style",
"Popularity", "Number of Doors", "Vehicle Size"]

# drop the irrevalent cols and print the head of the dataframe
df = df.drop(columns=col_to_drop, axis=1 )

# print df head
print(df.head())

  Make       Model  Year  Engine HP  Engine Cylinders Transmission
Type  \
0  BMW  1 Series M  2011      335.0               6.0
MANUAL
1  BMW    1 Series  2011      300.0               6.0
MANUAL
2  BMW    1 Series  2011      300.0               6.0
MANUAL
3  BMW    1 Series  2011      230.0               6.0
MANUAL
4  BMW    1 Series  2011      230.0               6.0
MANUAL

      Driven_Wheels  highway MPG  city mpg   MSRP
0  rear wheel drive           26        19  46135
1  rear wheel drive           28        19  40650
2  rear wheel drive           28        20  36350
3  rear wheel drive           28        18  29450
4  rear wheel drive           28        18  34500

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11914 entries, 0 to 11913
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Make               11914 non-null  object 
 1   Model              11914 non-null  object 
 2   Year               11914 non-null  int64  
 3   Engine HP          11845 non-null  float64
 4   Engine Cylinders   11884 non-null  float64
 5   Transmission Type  11914 non-null  object 
 6   Driven_Wheels      11914 non-null  object 
 7   highway MPG        11914 non-null  int64  
 8   city mpg           11914 non-null  int64  
 9   MSRP               11914 non-null  int64  
dtypes: float64(2), int64(4), object(4)
memory usage: 930.9+ KB
```

`Renaming the columns`

Now, Its time for renaming the feature to useful feature name. It will help to use them in model training purpose.

We have already dropped the unneccesary columns, and now we are left with useful columns. One extra thing that we would do is to rename the columns such that the name clearly represents the essence of the column.

The given dict represents (in key value pair) the previous name, and the new name for the dataframe columns

```
# rename cols
rename_cols ={
    "Make": "Brand",
    "Model": "CarModel",
    "Year": "ManufactureYear",
    "Engine HP": "Horsepower",
    "Engine Cylinders": "Cylinders",
    "Transmission Type": "Transmission",
    "Driven_Wheels": "DriveType",
    "highway MPG": "HighwayMileage",
    "city mpg": "CityMileage",
    "MSRP": "Price"
}

# use a pandas function to rename the current columns -
df = df.rename(rename_cols, axis = 1)

# Print the head of the dataframe
print(df.head())
```

```
  Brand     CarModel  ManufactureYear  Horsepower  Cylinders
Transmission  \
0   BMW  1 Series M             2011       335.0        6.0
MANUAL
1   BMW     1 Series             2011       300.0        6.0
MANUAL
2   BMW     1 Series             2011       300.0        6.0
MANUAL
3   BMW     1 Series             2011       230.0        6.0
MANUAL
4   BMW     1 Series             2011       230.0        6.0
MANUAL

          DriveType  HighwayMileage  CityMileage  Price
0  rear wheel drive              26           19  46135
1  rear wheel drive              28           19  40650
2  rear wheel drive              28           20  36350
3  rear wheel drive              28           18  29450
4  rear wheel drive              28           18  34500
```

`Dropping the duplicate rows`

There are many rows in the dataframe which are duplicate, and hence they are just repeating the information. Its better if we remove these rows as they don't add any value to the dataframe.

For given data, we would like to see how many rows were duplicates. For this, we will count the number of rows, remove the dublicated rows, and again count the number of rows.

```python
# number of rows before removing duplicated rows
before_remove_duplicated_rows = len(df)
print(before_remove_duplicated_rows)
```

```
11914
```

```python
# drop the duplicated rows
df = df.drop_duplicates()

# print head of df
print(df.head())
```

```
  Brand     CarModel  ManufactureYear  Horsepower  Cylinders
Transmission  \
0   BMW  1 Series M             2011       335.0        6.0
MANUAL
1   BMW     1 Series             2011       300.0        6.0
MANUAL
2   BMW     1 Series             2011       300.0        6.0
MANUAL
3   BMW     1 Series             2011       230.0        6.0
MANUAL
```

```
4   BMW    1 Series              2011       230.0        6.0
MANUAL

          DriveType  HighwayMileage  CityMileage  Price
0  rear wheel drive              26           19  46135
1  rear wheel drive              28           19  40650
2  rear wheel drive              28           20  36350
3  rear wheel drive              28           18  29450
4  rear wheel drive              28           18  34500

# Count Number of rows after deleting duplicated rows
after_remove_duplicates = len(df)
print(after_remove_duplicates)

10925
```

`Dropping the null or missing values`

Missing values are usually represented in the form of Nan or null or None in the dataset.

Finding whether we have null values in the data is by using the isnull() function.

There are many values which are missing, in pandas dataframe these values are reffered to as np.nan. We want to deal with these values beause we can't use nan values to train models. Either we can remove them to apply some strategy to replace them with other values.

To keep things simple we will be dropping nan values

```
# check for nan values in each columns
missing_values = df.isnull().sum()
print(missing_values)

Brand              0
CarModel           0
ManufactureYear    0
Horsepower        69
Cylinders         30
Transmission       0
DriveType          0
HighwayMileage     0
CityMileage        0
Price              0
dtype: int64
```

As we can see that the HP and Cylinders have null values of 69 and 30. As these null values will impact on models' accuracy. So to avoid the impact we will drop the these values. As these values are small camparing with dataset that will not impact any major affect on model accuracy so we will drop the values.

```
# drop missing values
df = df.dropna()


# Make sure that missing values are removed
# check number of nan values in each col again
non_missing_values = df.isnull().sum()
print(non_missing_values)
```

```
Brand              0
CarModel           0
ManufactureYear    0
Horsepower         0
Cylinders          0
Transmission       0
DriveType          0
HighwayMileage     0
CityMileage        0
Price              0
dtype: int64
```

```
#Describe statistics of df
statistics = df.describe(include='all')
print(statistics)
```

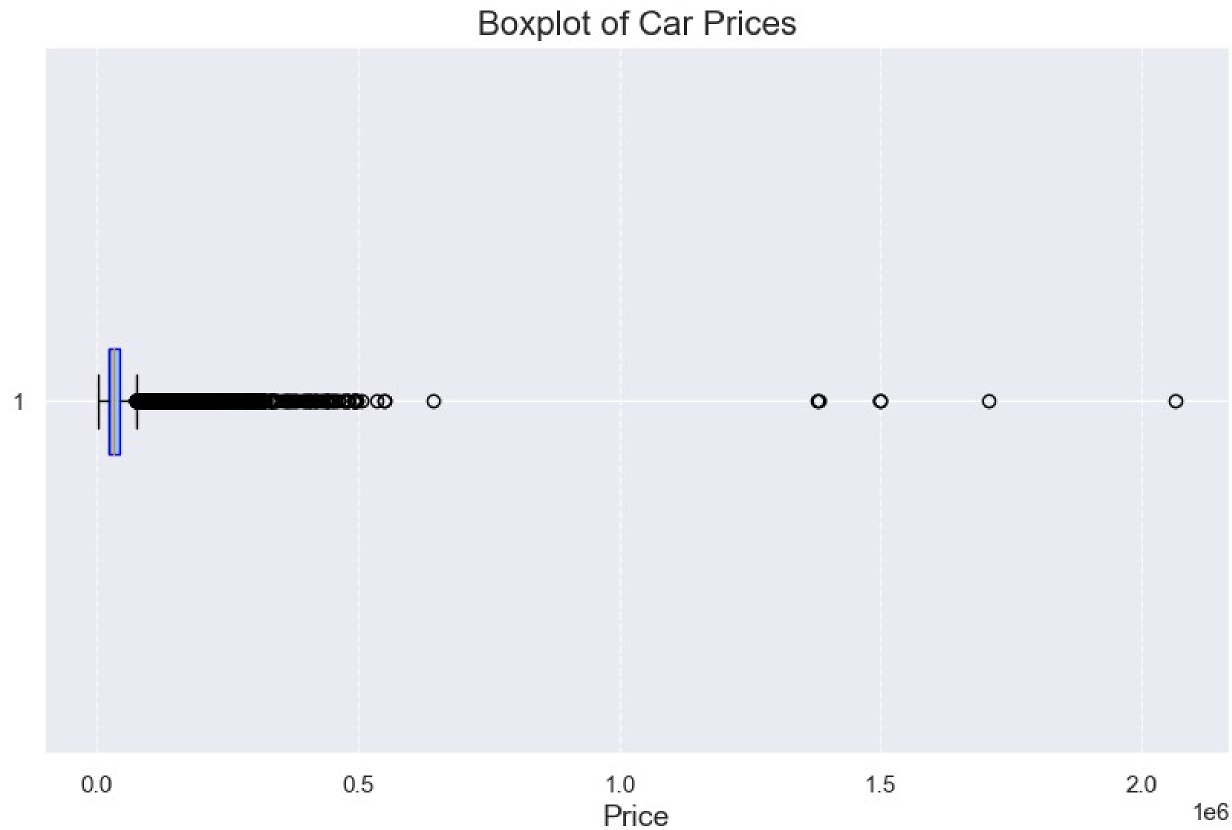
```
            Brand        CarModel  ManufactureYear    Horsepower  \
count       10827           10827     10827.000000  10827.000000
unique         47             904              NaN           NaN
top     Chevrolet  Silverado 1500              NaN           NaN
freq         1043             156              NaN           NaN
mean          NaN             NaN      2010.896370    254.553062
std           NaN             NaN         7.029534    109.841537
min           NaN             NaN      1990.000000     55.000000
25%           NaN             NaN      2007.000000    173.000000
50%           NaN             NaN      2015.000000    240.000000
75%           NaN             NaN      2016.000000    303.000000
max           NaN             NaN      2017.000000   1001.000000

           Cylinders Transmission          DriveType
HighwayMileage  \
count   10827.000000        10827              10827    10827.000000

unique           NaN            5                  4             NaN

top              NaN    AUTOMATIC  front wheel drive             NaN

freq             NaN         7750               4168             NaN

mean        5.691604          NaN                NaN       26.308119

std         1.768551          NaN                NaN        7.504652
```

```
min          0.000000          NaN                 NaN     12.000000
25%          4.000000          NaN                 NaN     22.000000
50%          6.000000          NaN                 NaN     25.000000
75%          6.000000          NaN                 NaN     30.000000
max         16.000000          NaN                 NaN    354.000000

         CityMileage         Price
count   10827.000000  1.082700e+04
unique           NaN           NaN
top              NaN           NaN
freq             NaN           NaN
mean       19.327607  4.249325e+04
std         6.643567  6.229451e+04
min         7.000000  2.000000e+03
25%        16.000000  2.197250e+04
50%        18.000000  3.084500e+04
75%        22.000000  4.330000e+04
max       137.000000  2.065902e+06
```

`Removing outliers`

Sometimes a dataset can contain extreme values that are outside the range of what is expected and unlike the other data. These are called outliers and often machine learning modeling and model skill in general can be improved by understanding and even removing these outlier values.

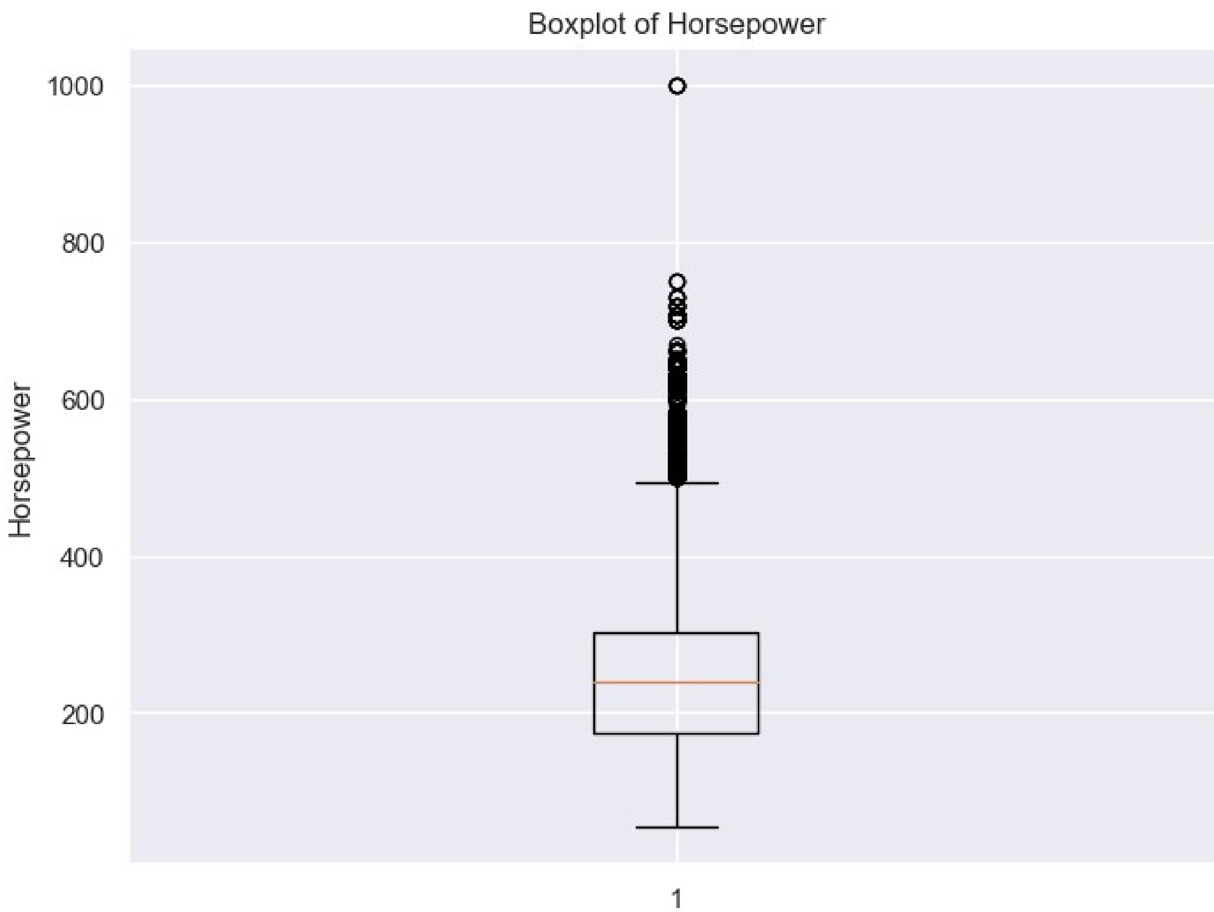
```
## Plot a boxplot for 'Price' column in dataset.
plt.figure(figsize=(10, 6))
plt.boxplot(df['Price'], vert=False, patch_artist=True,
boxprops=dict(facecolor='skyblue', color='blue'))
plt.title('Boxplot of Car Prices', fontsize=16)
plt.xlabel('Price', fontsize=14)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```




**`Observation:`**

Here as you see that we got some values near to 1.5 and 2.0 . So these values are called outliers. Because there are away from the normal values. Now we have detect the outliers of the feature of Price. Similarly we will checking of anothers features.

```
## PLot a boxplot for 'HP' columns in dataset
plt.figure(figsize=(8, 6))
plt.boxplot(df['Horsepower'])
plt.title('Boxplot of Horsepower')
plt.ylabel('Horsepower')
plt.show()
```




**Observation:**

Here boxplots show the proper distribution of of 25 percentile and 75 percentile of the feature of HP.

```
Q1 = np.percentile(df['Horsepower'], 25)
Q3 = np.percentile(df['Horsepower'], 75)
median = np.median(df['Horsepower'])

plt.figure(figsize=(8, 6))

sns.boxplot(y=df['Horsepower'])
```

```
<Axes: ylabel='Horsepower'>
```


print all the columns which are of int or float datatype in df.

Hint: Use loc with condition

```
# print all the columns which are of int or float datatype in df.
num_columns = df.select_dtypes(include=['int', 'float']).columns

# Print the column names
print("Columns with int or float datatype:")
for col in num_columns:
    print(col)

Columns with int or float datatype:
ManufactureYear
Horsepower
Cylinders
HighwayMileage
CityMileage
Price
```

`Save the column names of the above output in variable list named 'l'`

```
# save column names of the above output in variable list
l=['ManufactureYear','Horsepower','Cylinders','HighwayMileage','CitytM
ileage','Price']
```

**`Outliers removal techniques - IQR Method`**

**Here comes cool Fact for you!**

IQR is the first quartile subtracted from the third quartile; these quartiles can be clearly seen on a box plot on the data.

- Calculate IQR and give a suitable threshold to remove the outliers and save this new dataframe into df2.

Let us help you to decide threshold: Outliers in this case are defined as the observations that are below (Q1 − 1.5x IQR) or above (Q3 + 1.5x IQR)

```
## define Q1 and Q2
Q1 = np.percentile(df['Horsepower'], 25)
Q2 = np.percentile(df['Horsepower'], 75)


# # define IQR (interquantile range)
IQR = Q3 - Q1

# # define df2 after removing outliers
df2 = df[(df['Horsepower'] >= (Q1 - 1.5 * IQR)) & (df['Horsepower'] <=
(Q3 + 1.5 * IQR))]

# find the shape of df & df2
df.shape
df2.shape


(10332, 10)

# find unique values and there counts in each column in df using value
counts function.

for i in df.columns:
    print ("--------------- %s ----------------" % i)
     # code
    print (df[i].value_counts())
    print("\n")

--------------- Brand ----------------
Brand
Chevrolet        1043
Ford              798
Toyota            651
Volkswagen        563
Nissan            540
Dodge             513
GMC               475
Honda             429
Cadillac          396
```

```
Mazda              392
Mercedes-Benz      340
Suzuki             338
Infiniti           326
BMW                324
Audi               320
Hyundai            254
Acura              246
Volvo              241
Subaru             229
Kia                219
Mitsubishi         202
Lexus              201
Chrysler           185
Buick              184
Pontiac            163
Lincoln            152
Porsche            134
Land Rover         126
Oldsmobile         111
Saab               101
Aston Martin        91
Bentley             74
Ferrari             69
Plymouth            62
Scion               60
FIAT                58
Maserati            55
Lamborghini         52
Rolls-Royce         31
Lotus               28
HUMMER              17
Maybach             16
McLaren              5
Alfa Romeo           5
Genesis              3
Bugatti              3
Spyker               2
Name: count, dtype: int64


---------------- CarModel ----------------
CarModel
Silverado 1500     156
F-150              126
Sierra 1500         90
Tundra              78
Frontier            76
                  ...
```

```
M4 GTS              1
LFA                 1
Horizon             1
GS F                1
Zephyr              1
Name: count, Length: 904, dtype: int64


--------------- ManufactureYear ----------------
ManufactureYear
2015    2029
2016    2022
2017    1580
2014     530
2012     350
2009     349
2007     332
2013     320
2008     316
2011     278
2010     272
2003     233
2004     230
2005     205
2002     203
2006     194
2001     168
1997     148
1998     143
1993     135
2000     114
1999     111
1994     109
1992     104
1995     103
1996      98
1991      84
1990      67
Name: count, dtype: int64


--------------- Horsepower ----------------
Horsepower
200.0    373
170.0    255
240.0    248
285.0    246
210.0    243
        ...
557.0      1
```

```
361.0       1
456.0       1
661.0       1
151.0       1
Name: count, Length: 355, dtype: int64


--------------- Cylinders ----------------
Cylinders
4.0     4227
6.0     4215
8.0     1889
12.0     228
5.0      159
10.0      65
3.0       28
0.0       13
16.0       3
Name: count, dtype: int64


--------------- Transmission ----------------
Transmission
AUTOMATIC           7750
MANUAL              2498
AUTOMATED_MANUAL     553
DIRECT_DRIVE          15
UNKNOWN               11
Name: count, dtype: int64


--------------- DriveType ----------------
DriveType
front wheel drive    4168
rear wheel drive     3120
all wheel drive      2281
four wheel drive     1258
Name: count, dtype: int64


--------------- HighwayMileage ----------------
HighwayMileage
24     822
23     758
26     725
22     686
25     685
28     651
27     555
30     499
```

```
21      488
19      488
31      488
20      469
29      425
18      345
17      340
33      329
32      292
34      270
16      199
35      199
36      191
37      166
38      130
15      116
40      109
39      107
41       65
42       46
14       37
43       21
46       21
44       21
48       16
45       14
13       13
50       10
47        7
109       6
12        5
53        5
82        3
111       3
354       1
106       1
Name: count, dtype: int64


---------------- CityMileage ----------------
CityMileage
17     1154
16     1014
15      949
18      938
19      793
20      742
14      603
22      571
```

```
21       551
13       537
23       425
25       392
24       372
12       282
27       243
26       207
11       187
28       160
30       127
31       116
29        98
10        76
9         33
32        21
34        20
36        20
40        19
44        18
42        17
41        17
35        15
33        13
53        13
43        13
54        10
8          9
37         8
39         6
51         6
50         6
128        6
49         4
137        3
85         3
55         3
47         2
58         2
129        1
7          1
38         1
Name: count, dtype: int64


--------------- Price ----------------
Price
2000     599
29995     18
```

```
25995     16
20995     15
27995     15
         ...
66347      1
62860      1
48936      1
68996      1
50920      1
Name: count, Length: 6014, dtype: int64
```

`Visualising Univariate Distributions`

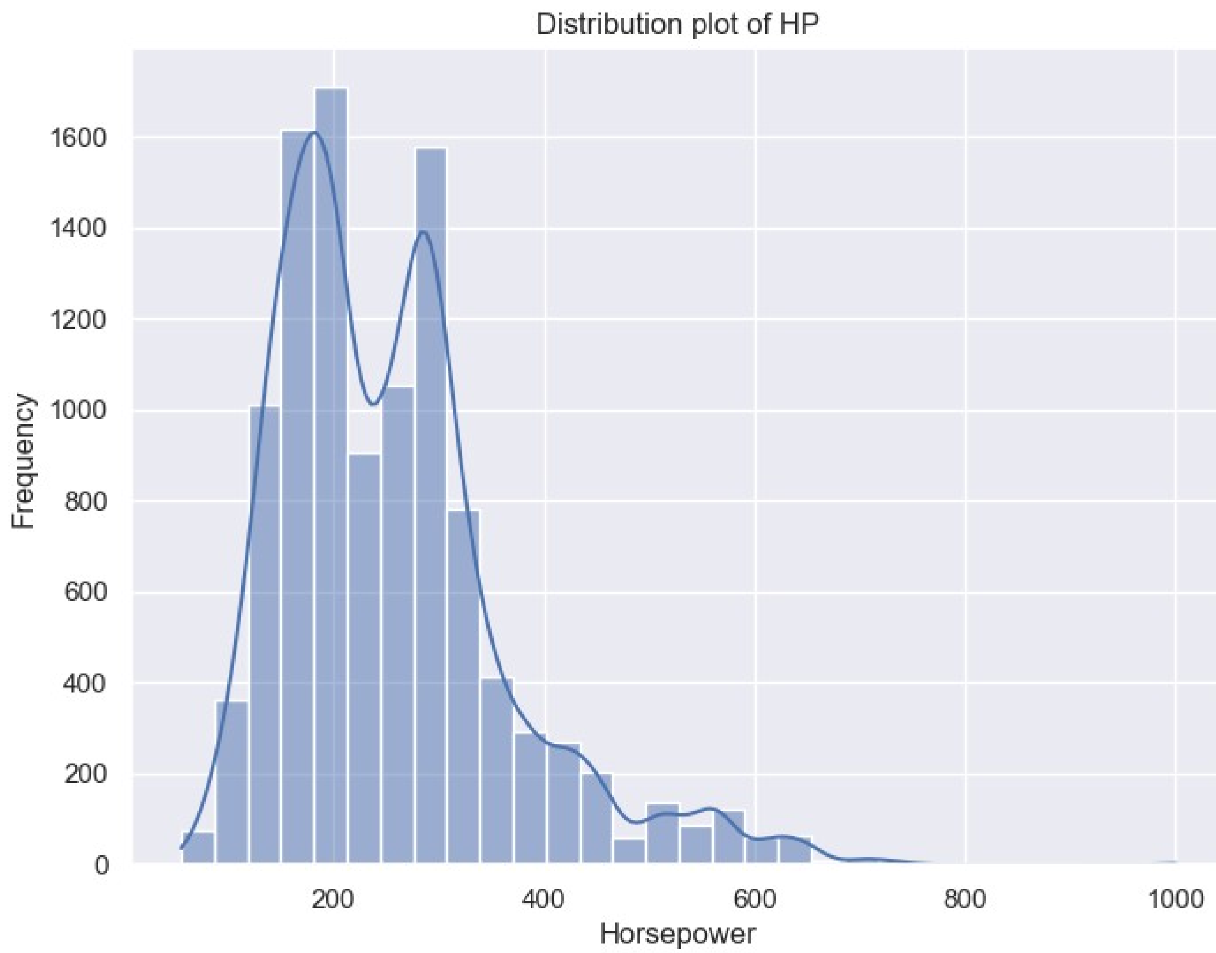
We will use seaborn library to visualize eye catchy univariate plots.

Do you know? you have just now already explored one univariate plot. guess which one? Yeah its box plot.

`Histogram & Density Plots`

Histograms and density plots show the frequency of a numeric variable along the y-axis, and the value along the x-axis. The `sns.distplot()` function plots a density curve. Notice that this is aesthetically better than vanilla `matplotlib`.

```
#ploting distplot for variable HP
plt.figure(figsize=(8,6))
sns.histplot(df['Horsepower'], kde=True, bins=30)  # kde=True adds the smooth density curve
plt.title('Distribution plot of HP')
plt.xlabel('Horsepower')
plt.ylabel('Frequency')
plt.show()
```




**`Observation:`**

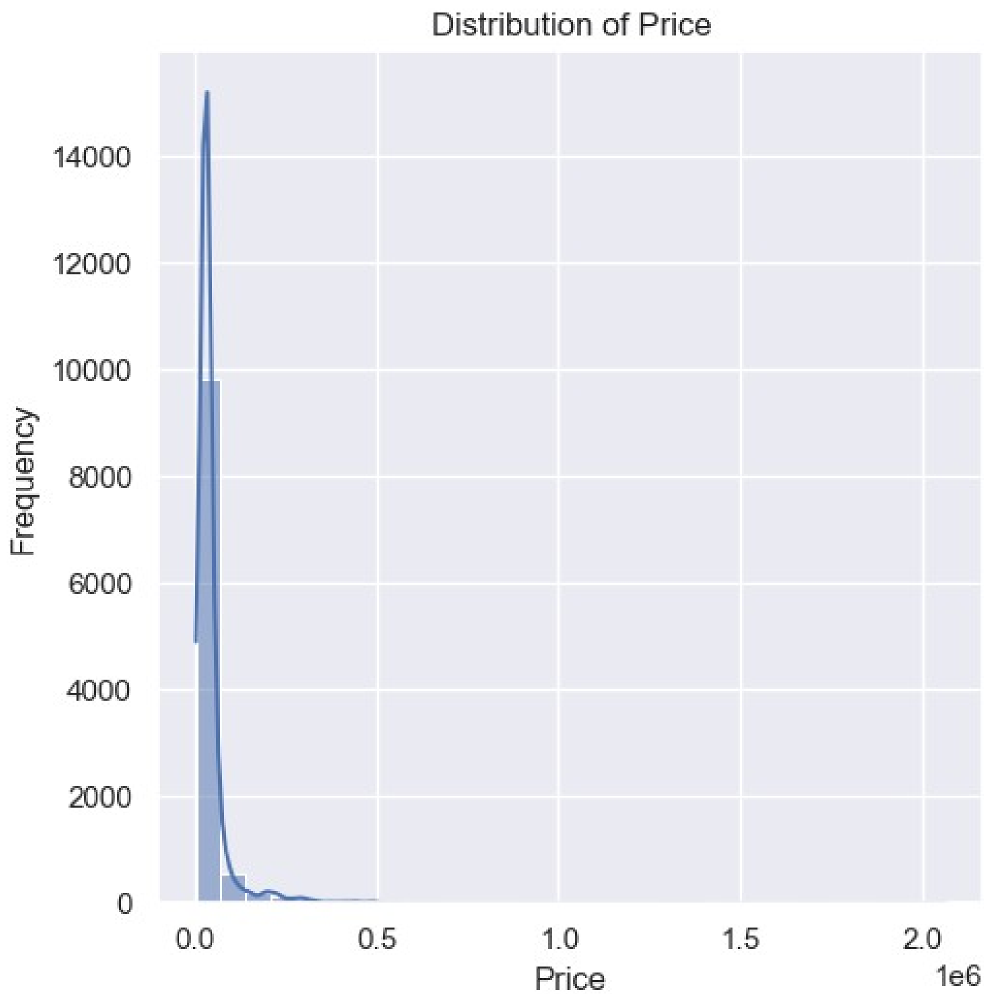
We plot the Histogram of feature HP with help of distplot in seaborn. In this graph we can see that there is max values near at 200. similary we have also the 2nd highest value near 400 and so on.  It represents the overall distribution of continuous data variables.

Since seaborn uses matplotlib behind the scenes, the usual matplotlib functions work well with seaborn. For example, you can use subplots to plot multiple univariate distributions.

- Hint: use matplotlib subplot function

```
# plot all the columns present in list l together using subplot of
dimention (2,3).


c=0
plt.figure(figsize=(15,10))
for i in l:
     # code here
    c += 1
plt.subplot(2, 3, c)
sns.histplot(df[i], kde=True, bins=30)
```

```
plt.title(f'Distribution of {i}')
plt.xlabel(i)
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```

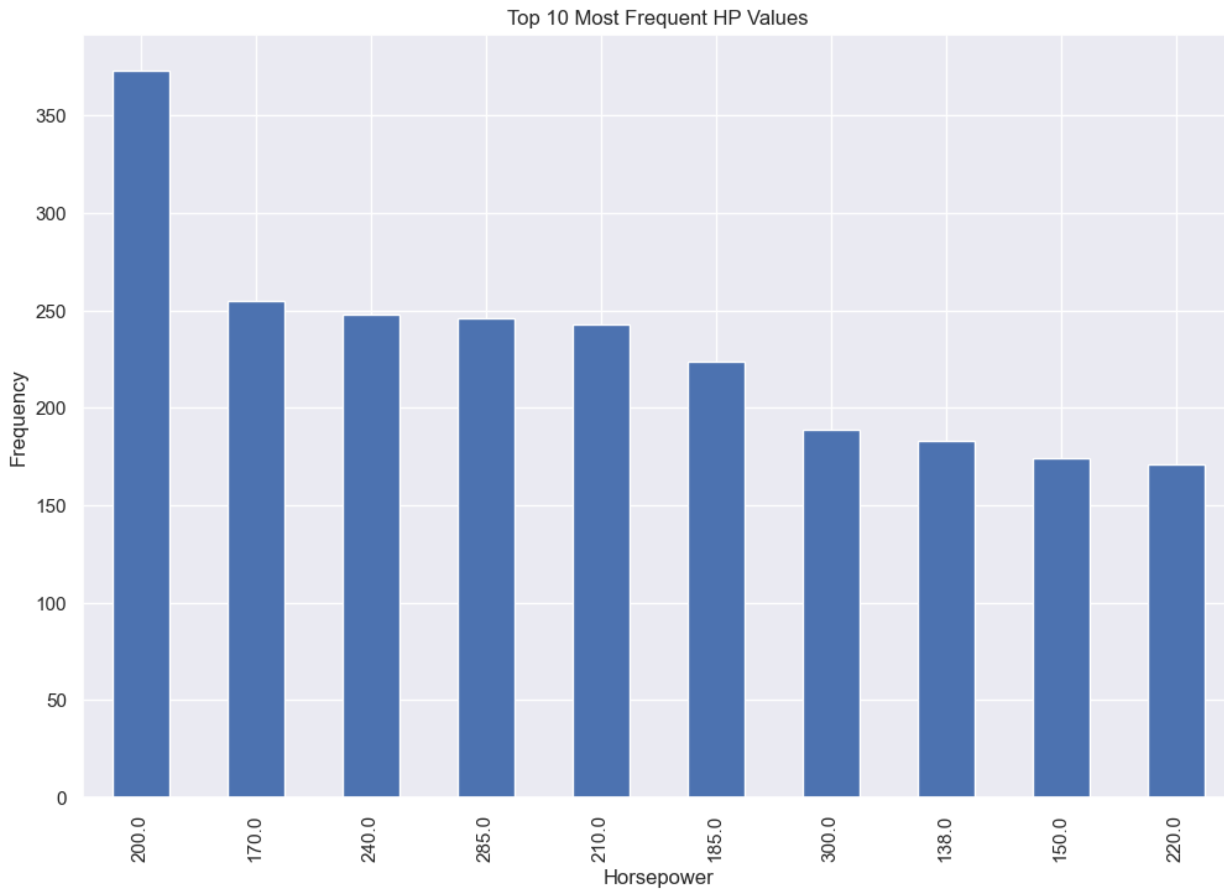



`Bar Chart Plots`

Plot a histogram depicting the make in X axis and number of cars in y axis.

```
# plt.figure(figsize = (12,8))

# use nlargest and then .plot to get bar plot like below output
# Plot Title, X & Y label
plt.figure(figsize=(12, 8))

# Example: Get top 10 values from 'HP' column and plot bar chart
top_hp = df['Horsepower'].value_counts().nlargest(10)
```

```
top_hp.plot(kind='bar')

plt.title('Top 10 Most Frequent HP Values')
plt.xlabel('Horsepower')
plt.ylabel('Frequency')

plt.show()
```




**`Observation:`**

In this plot we can see that we have plot the bar plot with the cars model and nos. of cars.

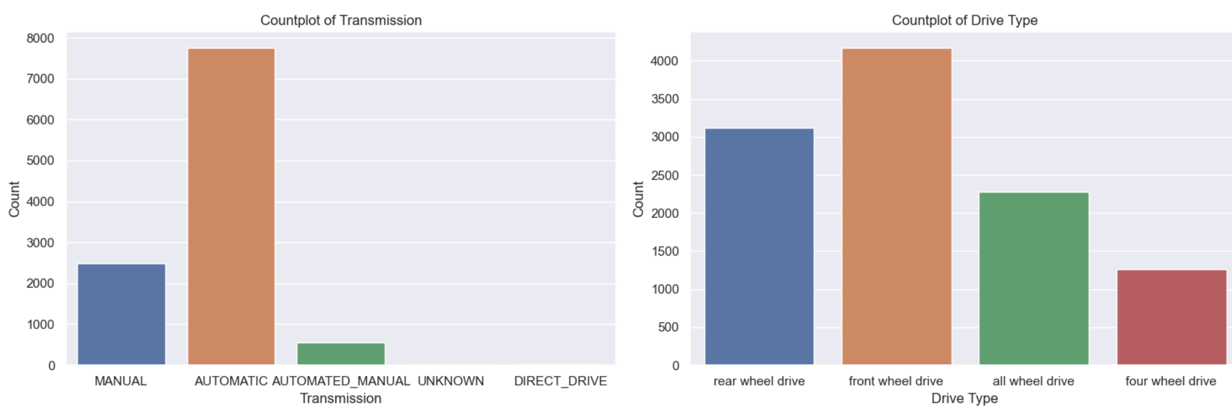
`Count Plot`

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable.

Plot a countplot for a variable Transmission vertically with hue as Drive mode

```
print(df.columns)
```

```
Index(['Brand', 'CarModel', 'ManufactureYear', 'Horsepower', 'Cylinders',
       'Transmission', 'DriveType', 'HighwayMileage', 'CityMileage', 'Price'],
      dtype='object')

plt.figure(figsize=(15,5))

# plot countplot on transmission and drive mode
plt.subplot(1, 2, 1)
sns.countplot(data=df, x='Transmission')
plt.title('Countplot of Transmission')
plt.xlabel('Transmission')
plt.ylabel('Count')

# Countplot for drive_mode
plt.subplot(1, 2, 2)
sns.countplot(data=df, x='DriveType')
plt.title('Countplot of Drive Type')
plt.xlabel('Drive Type')
plt.ylabel('Count')

plt.tight_layout()
plt.show()
```




**Observation:**

In this count plot, We have plot the feature of Transmission with help of hue. We can see that the the nos of count and the transmission type and automated manual is plotted. Drive mode as been given with help of hue.

`Visualising Bivariate Distributions`

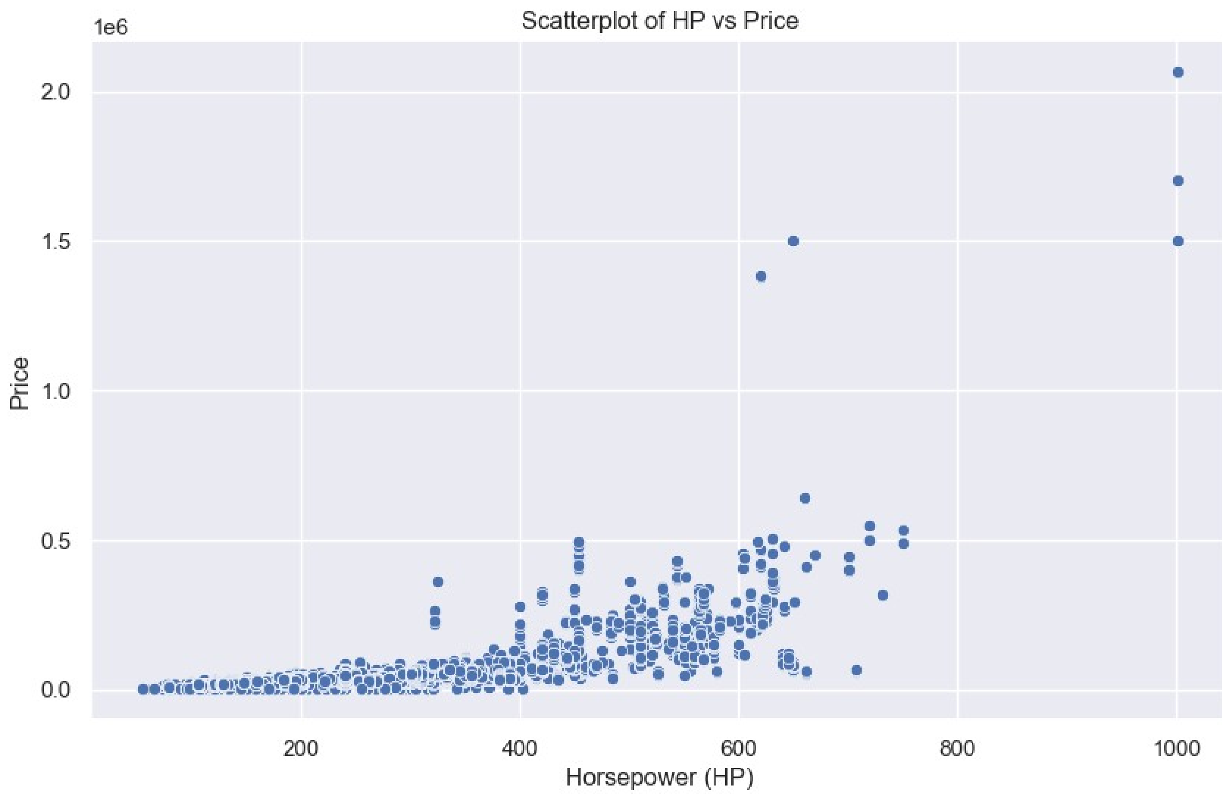
Bivariate distributions are simply two univariate distributions plotted on x and y axes respectively. They help you observe the relationship between the two variables.

`Scatter Plots`

Scatterplots are used to find the correlation between two continuos variables.

Using scatterplot find the correlation between 'HP' and 'Price' column of the data.

```
## Your code here -
fig, ax = plt.subplots(figsize=(10,6))

# plot scatterplot on hp and price
sns.scatterplot(data=df, x='Horsepower', y='Price', ax=ax)

# Add title and labels
ax.set_title('Scatterplot of HP vs Price')
ax.set_xlabel('Horsepower (HP)')
ax.set_ylabel('Price')

plt.show()
```




**`Observation:`**

It is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data. We have plot the scatter plot with x axis as HP and y axis as Price. The data points between the features should be same either wise it give errors.

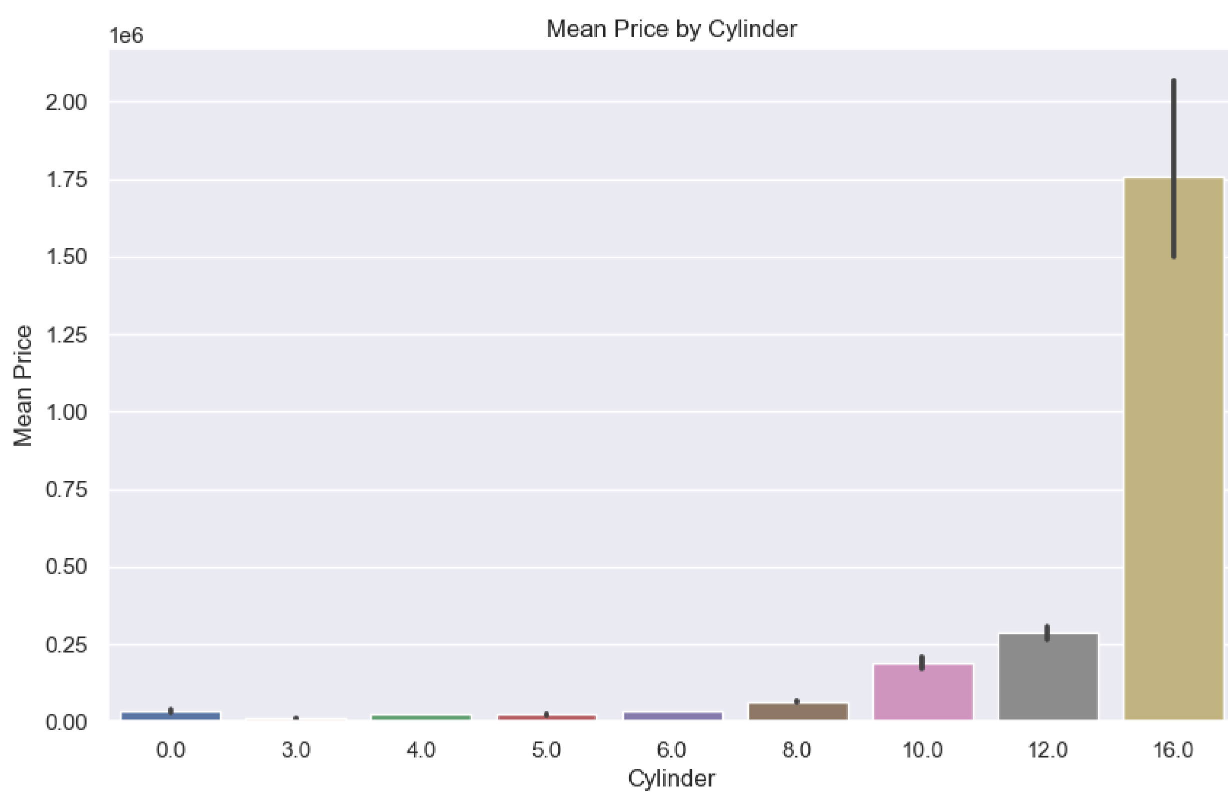
Plotting Aggregated Values across Categories

Bar Plots - Mean, Median and Count Plots

Bar plots are used to **display aggregated values** of a variable, rather than entire distributions. This is especially useful when you have a lot of data which is difficult to visualise in a single figure.

For example, say you want to visualise and *compare the Price across Cylinders*. The `sns.barplot()` function can be used to do that.

```
# bar plot with default statistic=mean between Cylinder and Price
plt.figure(figsize=(10, 6))

# Create a bar plot with default statistic=mean
sns.barplot(data=df, x='Cylinders', y='Price')

# Add title and labels
plt.title('Mean Price by Cylinder')
plt.xlabel('Cylinder')
plt.ylabel('Mean Price')

plt.show()
```



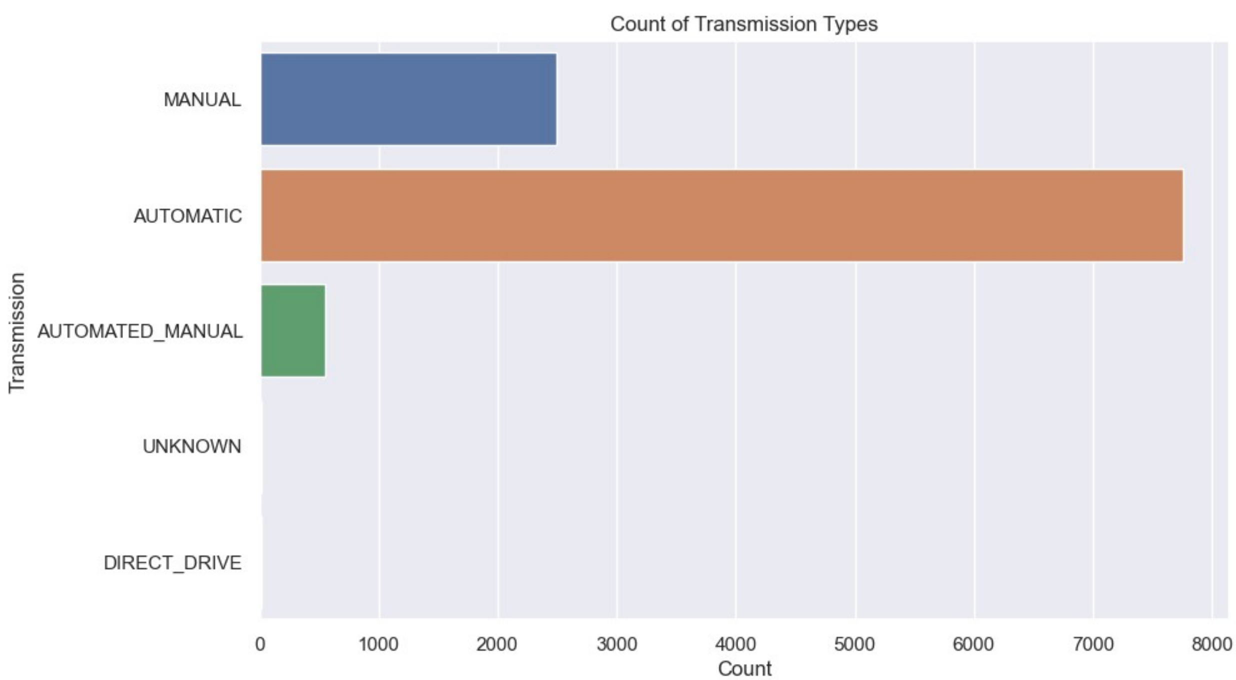
**`Observation:`**

By default, seaborn plots the mean value across categories, though you can plot the count, median, sum etc. Also, barplot computes and shows the confidence interval of the mean as well.

`When you want to visualise having a large number of categories, it is helpful to plot the categories across the y-axis.`

`Let's now drill down into Transmission sub categories.`

```
# Plotting categorical variable Transmission across the y-axis
plt.figure(figsize=(10, 6))

# Plotting countplot with Transmission on the y-axis
sns.countplot(data=df, y='Transmission')

# Add title and labels
plt.title('Count of Transmission Types')
plt.xlabel('Count')
plt.ylabel('Transmission')

plt.show()
```




These plots looks beutiful isn't it? In Data Analyst life such charts are there unavoidable friend.:)

`Multivariate Plots`

`Heatmaps`

A heat map is a two-dimensional representation of information with the help of colors. Heat maps can help the user visualize simple or complex information

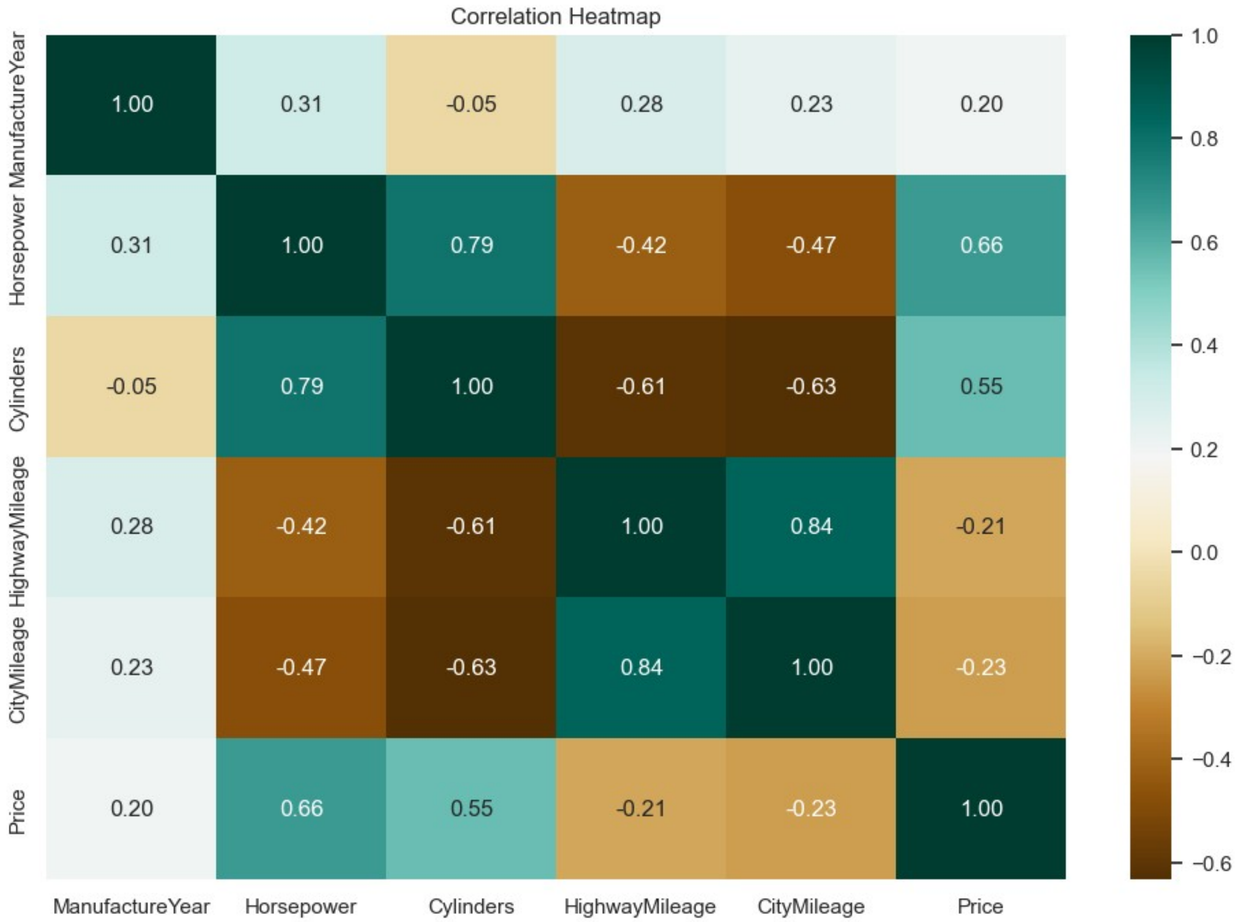
Using heatmaps plot the correlation between the features present in the dataset.

```
#find the correlation of features of the data
numeric_df = df.select_dtypes(include=['int64', 'float64'])

# Calculate the correlation matrix
corr = numeric_df.corr()

# Print the correlation matrix
print(corr)
```

```
                 ManufactureYear  Horsepower  Cylinders
HighwayMileage  \
ManufactureYear         1.000000    0.314971  -0.050598
0.284237
Horsepower              0.314971    1.000000   0.788007        -
0.420281
Cylinders              -0.050598    0.788007   1.000000        -
0.611576
HighwayMileage          0.284237   -0.420281  -0.611576
1.000000
CityMileage             0.234135   -0.473551  -0.632407
0.841229
Price                   0.196789    0.659835   0.554740        -
0.209150

                 CityMileage     Price
ManufactureYear     0.234135  0.196789
Horsepower         -0.473551  0.659835
Cylinders          -0.632407  0.554740
HighwayMileage      0.841229 -0.209150
CityMileage         1.000000 -0.234050
Price              -0.234050  1.000000
```

```
# Using the correlated df, plot the heatmap
# set cmap = 'BrBG', annot = True - to get the same graph as shown
below
# set size of graph = (12,8)

corr = df.select_dtypes(include=['int64', 'float64']).corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr, annot=True, cmap='BrBG', fmt=".2f")
```

```python
# Add title
plt.title("Correlation Heatmap")

plt.show()
```




**`Observation:`**

A heatmap contains values representing various shades of the same colour for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different colour can also be used.

The above heatmap plot shows correlation between various variables in the colored scale of -1 to 1.