CAR PRICE PREDICTION

Car Price is one of the key factor for car owner. Some cars with good brand names seems to be of good value in the market and some may have not that much value in sale market. Its a problem for the car owner or company that some cars are more in demand and some are not.

Scrapping

I have scrapped the data from cars.com of different location.

In [2]:
```
#### First scrap the data from OLX,CARS24,CARDEKHO ETC
#### SO IMPORT BASIC LIBRARIES TO SCRAP DATA
```

In [1]:
```
import selenium
import pandas as pd
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
import time
```

In [46]:
```
### connect to the web driver
driver=webdriver.Chrome(r'D:\flip Robo\chromedriver.exe')
```

In [ ]:
```

```

In [3]:
```
##### Open the URL for new delhi location
URL = 'https://www.cars24.com/buy-used-cars-new-delhi/?itm_source=Cars24Website&itm_medium=sticky_header'
driver.get(URL)
time.sleep(2)
```

In [ ]:
```
#columns are
#Brand, model, variant, manufacturing year, driven kilometers, fuel, number of owners, location and
#at last target variable Price of the car
```

Store data in different columns

[16]:
```
Brand = []
Model = []
Variant = []
Manufacturing = []
Kilometers = []
Fuel = []
Location = []
Price = []
Owner = []
for i in url:
    driver.get(i)
    try:

        manufacturing = driver.find_element_by_xpath('/html/body/div[1]/div[2]/div/div/div/div[1]/div[2]/h2').text[0:4]
        brand = driver.find_element_by_xpath('/html/body/div[1]/div[2]/div/div/div/div[1]/div[2]/h2').text[5:11]
        model = driver.find_element_by_xpath('/html/body/div[1]/div[2]/div/div/div/div[1]/div[2]/h2').text[12:16]
        variant = driver.find_element_by_xpath('/html/body/div[1]/div[2]/div/div/div/div[1]/div[2]/h2').text[17:23]
        kilometer =driver.find_element_by_xpath('/html/body/div[1]/div[3]/div[3]/div/div/div/div/ul/li[3]/strong').text
        fuel = driver.find_element_by_xpath('/html/body/div[1]/div[2]/div/div/div/div[1]/div[2]/ul/li[3]').text
        location = driver.find_element_by_xpath('/html/body/div[1]/div[3]/div[4]/div/div/div/div[1]/p/span').text
        price = driver.find_element_by_xpath('/html/body/div[1]/div[2]/div/div/div/div[1]/div[3]/strong').text[10:18]
        owner = driver.find_element_by_xpath('/html/body/div[1]/div[2]/div/div/div/div[1]/div[2]/ul/li[5]').text
        Brand.append(brand)
        Model.append(model)
        Variant.append(variant)
        Kilometers.append(kilometer)
        Fuel.append(fuel)
        Location.append(location)
        Manufacturing.append(manufacturing)
        Owner.append(owner)
        Price.append(price)
    except NoSuchElementException:
```

Final CSV file

```
]: car_csv
```

]:

	Manufacturing	Brand	Model	Variant	Kilometers	Fuel	Location	Owner	Price
0	2020	Maruti	S PR	SSO VX	3,176 km	Petrol	DELHI	1st Owner	4,02,199
1	2013	Maruti	Swif	ZDI M	24,022 km	Diesel	DELHI	1st Owner	3,96,599
2	2020	Maruti	S PR	SSO VX	7,556 km	Petrol	DELHI	1st Owner	4,01,899
3	2015	Maruti	Alto	800 LX	9,217 km	Petrol	DELHI	1st Owner	2,58,599
4	2017	Maruti	Alto	K10 VX	11,691 km	Petrol	DELHI	1st Owner	3,45,699
...	...	...	...	...	...	...	...	...	...
715	2015	Hyunda	Eon	SPORTZ	15,346 km	Petrol	CHENNAI	3rd Owner	3,01,299
716	2012	Maruti	Wago	R 1.0	38,553 km	Petrol	CHENNAI	1st Owner	3,15,199
717	2018	Tata T	ago	Z 1.2	12,997 km	Petrol	CHENNAI	1st Owner	5,28,899
718	2017	Maruti	Alto	800 VX	21,239 km	Petrol	CHENNAI	1st Owner	3,20,999
719	2017	Maruti	Wago	R 1.0	18,782 km	Petrol	CHENNAI	2nd Owner	4,19,599

720 rows × 9 columns

Import the libraries and load the data

In [125]:
```
### import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from scipy import stats
from sklearn.preprocessing import StandardScaler
```

In [126]:
```
### load the data sets
Cars_df = pd.read_csv('cars.csv')
```

In [127]:
```
Cars_df
```

Out[127]:

	Unnamed: 0	Manufacturing	Brand	Model	Variant	Kilometers	Fuel	Location	Owner	Price
0	0	2020	Maruti	S PR	SSO VX	3,176 km	Petrol	DELHI	1st Owner	4,02,199
1	1	2013	Maruti	Swif	ZDI M	24,022 km	Diesel	DELHI	1st Owner	3,96,599
2	2	2020	Maruti	S PR	SSO VX	7,556 km	Petrol	DELHI	1st Owner	4,01,899
3	3	2015	Maruti	Alto	800 LX	9,217 km	Petrol	DELHI	1st Owner	2,58,599
4	4	2017	Maruti	Alto	K10 VX	11,691 km	Petrol	DELHI	1st Owner	3,45,699
5	5	2015	Honda	maze	1.2 VX	27,407 km	Petrol	DELHI	1st Owner	4,53,399
6	6	2017	Maruti	Alto	800 LX	8,501 km	Petrol	DELHI	1st Owner	2,93,799
7	7	2015	Maruti	Alto	800 LX	17,637 km	Petrol	DELHI	1st Owner	2,78,799
8	8	2014	Maruti	Alto	800 VX	12,535 km	Petrol	DELHI	1st Owner	2,93,599
9	9	2018	Renaul	Kwi	RXL M	8,220 km	Petrol	DELHI	1st Owner	3,14,199
10	10	2013	Maruti	Alto	800 LX	10,111 km	Petrol	DELHI	1st Owner	2,27,899
11	11	2010	Maruti	Alto	K10 VX	41,190 km	Petrol	DELHI	1st Owner	1,80,599

EDA and Pre-processing

EDA is the most important part of data as it shows the columns ,num of null values , value counts ,datatypes, statistical summary etc.

EDA AND PREPROCESSING OF DATA

In [128]:
```
#### number of columns
df.columns
```

Out[128]:
```
Index(['Unnamed: 0', 'Manufacturing', 'Brand', 'Model', 'Variant',
       'Kilometers', 'Fuel', 'Location', 'Owner', 'Price'],
      dtype='object')
```

In [129]:
```
#### data types
Cars_df.dtypes
```

Out[129]:
```
Unnamed: 0         int64
Manufacturing     object
Brand             object
Model             object
Variant           object
Kilometers        object
Fuel              object
Location          object
Owner             object
Price             object
dtype: object
```

In [130]:
```
#### km and price are numeric columns showing in object types so it can be fixed
Cars_df['Kilometers'].replace('km','', regex=True, inplace=True)
```

In [131]:
```
Cars_df.replace(',','', regex=True, inplace=True)
```

In [132]:
```
Cars_df
```

Pre-processing

Through pre-processing techniques cleanliness of data is done.

In [149]:
```
#### check null columns
Cars_df.isnull().sum()
```

Out[149]:
```
Unnamed: 0        0
Manufacturing     0
Brand             0
Model             0
Variant           0
Kilometers        0
Fuel              0
Location          0
Owner             0
Price             0
dtype: int64
```

In [139]:
```
### many columns found blank in all rows so i replace with nan to drop the nan columns directly
import numpy as np
Cars_df.replace('--','None', regex=True, inplace=True)
```

In [144]:
```
## fill NaN/None value with zero in km and price
Cars_df['Kilometers'].replace('None',0, regex=True, inplace=True)
Cars_df['Price'].replace('None',0, regex=True, inplace=True)
```

In [146]:
```
Cars_df['Price']= Cars_df['Price'].fillna(0)
```

In [148]:
```
Cars_df['Location']= Cars_df['Location'].fillna('None')
```

In [155]:
```
#### now replace ₹ sign from price if any one have
Cars_df.replace('₹','', regex=True, inplace=True)
```

In [163]:
```
### dropunamed columns which no use its unique and also level is unamed
Cars_df.drop(['Unnamed: 0'],axis=1,inplace=True)
```

Separate the data into categorical and continuous columns and its help to keep original data safe and any mistake not lose of original one and also I can use encoding techniques as per columns

In [159]:
```
##### find the catagorical value
columns = [columns for columns in Cars_df.columns if Cars_df[columns].dtypes=='object']
columns
```

Out[159]:
```
['Manufacturing', 'Brand', 'Model', 'Variant', 'Fuel', 'Location', 'Owner']
```

In [160]:
```
Cars_df_cat = Cars_df[columns]
```

In [161]:
```
Cars_df_cat
```

Out[161]:

	Manufacturing	Brand	Model	Variant	Fuel	Location	Owner
0	2020	Maruti	S PR	SSO VX	Petrol	DELHI	1st Owner
1	2013	Maruti	Swif	ZDI M	Diesel	DELHI	1st Owner
2	2020	Maruti	S PR	SSO VX	Petrol	DELHI	1st Owner
3	2015	Maruti	Alto	800 LX	Petrol	DELHI	1st Owner
4	2017	Maruti	Alto	K10 VX	Petrol	DELHI	1st Owner
5	2015	Honda	maze	1.2 VX	Petrol	DELHI	1st Owner
6	2017	Maruti	Alto	800 LX	Petrol	DELHI	1st Owner
7	2015	Maruti	Alto	800 LX	Petrol	DELHI	1st Owner
8	2014	Maruti	Alto	800 VX	Petrol	DELHI	1st Owner
9	2018	Renaul	Kwi	RXL M	Petrol	DELHI	1st Owner
10	2013	Maruti	Alto	800 LX	Petrol	DELHI	1st Owner
11	2010	Maruti	Alto	K10 VX	Petrol	DELHI	1st Owner

In [164]:
```
##### find the continous columns
count_col = [count_col for count_col in Cars_df.columns if Cars_df[count_col].dtypes!='object']
count_col
```

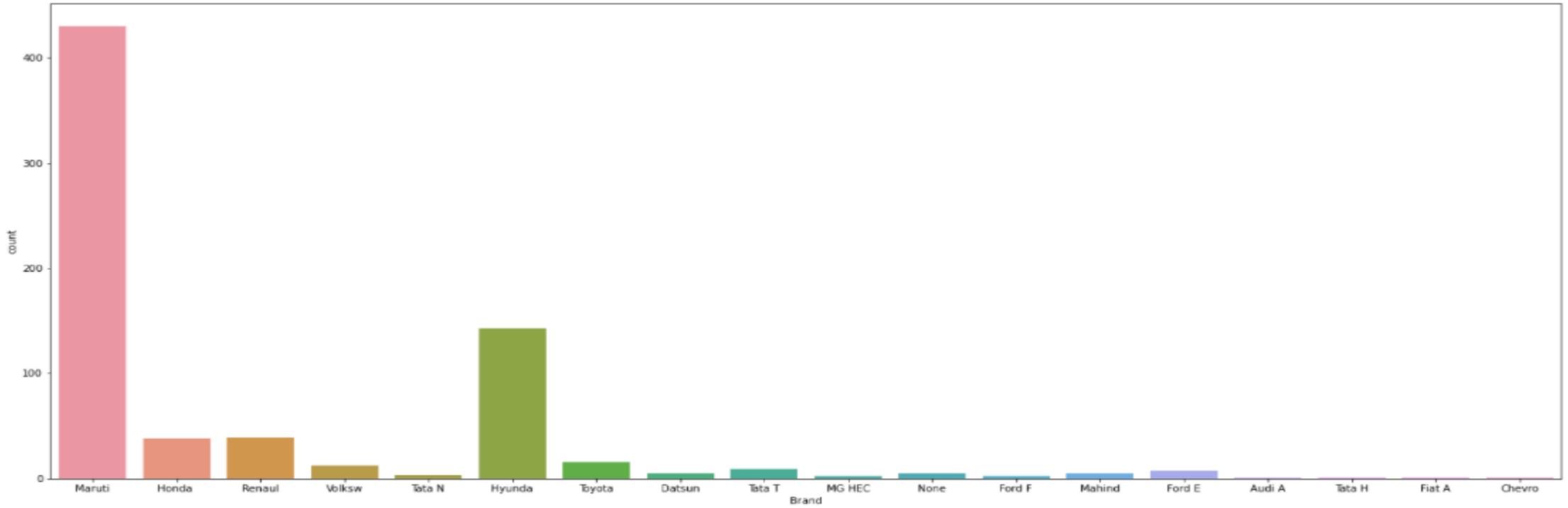
Out[164]:
```
['Kilometers', 'Price']
```

Visualization of data

In [167]:
```
Cars_df.columns
```

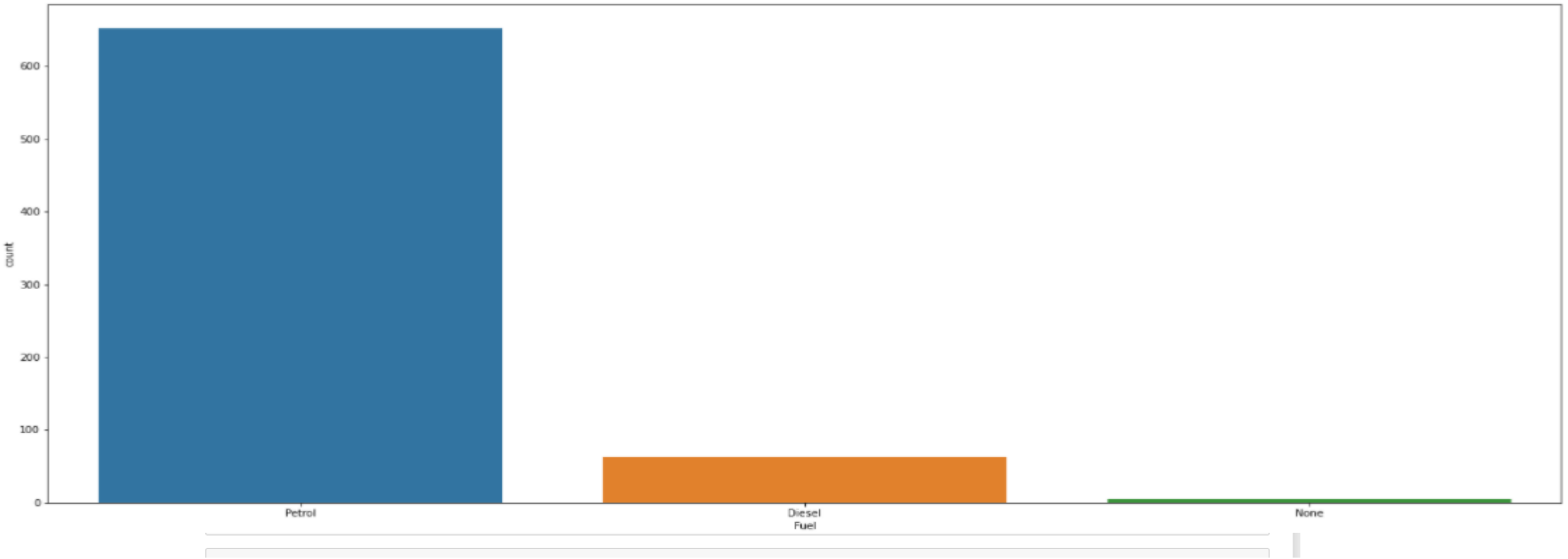
Out[167]:
```
Index(['Manufacturing', 'Brand', 'Model', 'Variant', 'Kilometers', 'Fuel',
       'Location', 'Owner', 'Price'],
      dtype='object')
```

In [176]:
```
##### Visalization
#### Bar Plot
plt.figure(figsize=(25,10))
sns.countplot(x=Cars_df['Brand'])
plt.show()
```



Maruti is highest selling used car
and mostly petrol type car is on sale as per chart

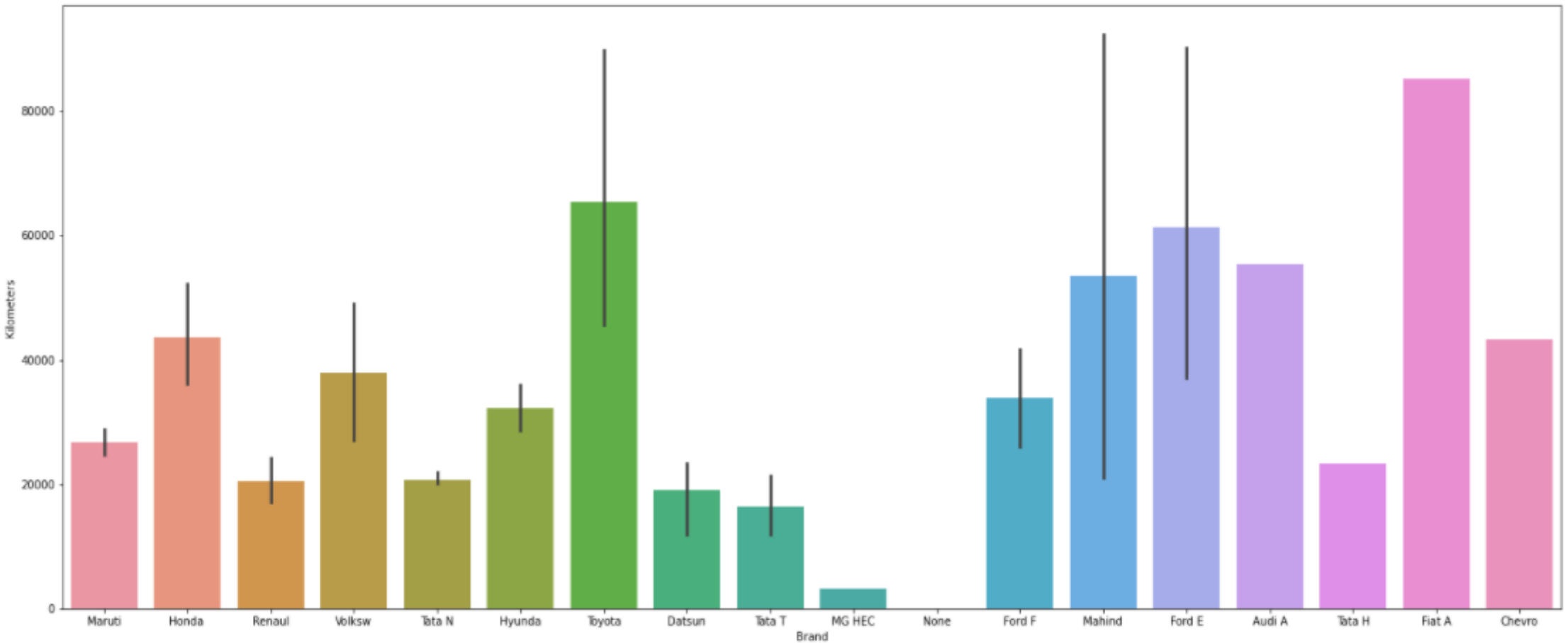
In [178]:
```
#### Bar Plot
plt.figure(figsize=(25,10))
sns.countplot(x=Cars_df['Fuel'])
plt.show()
```



In [ ]:
```
### mostly petrol engine cars are seems to be in all location to sell
```

In [180]:
```
plt.figure(figsize=(25,10))
sns.barplot(x=Cars_df['Brand'],y=Cars_df['Kilometers'])
plt.show
```

Out[180]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Model Building

In [209]:
```
#### Model building
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingRegressor,RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
rf=RandomForestRegressor()
dtc = DecisionTreeRegressor()
lr=LinearRegression()
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

In [210]:
```
y = Cars_df['Price']
x = Cars_df_new
```

In [211]:
```
x.shape
```

Out[211]: (720, 8)

In [212]:
```
y.shape
```

Out[212]: (720,)

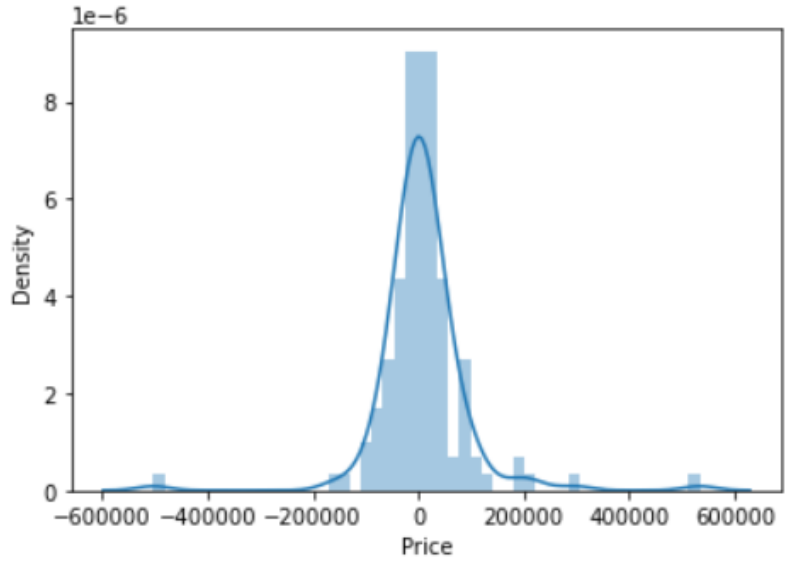
Random State

[219]:
```
### find the best accuracy
maxAcc=0
maxRs=0
for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=i)
    lr.fit(x_train,y_train)
    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
   # print(f"At Random State {i},the tarining accuracy is :- ",{r2_score(y_train,pred_train)})
   # print(f"At Random State {i},the Test accuracy is :- ",{r2_score(y_test,pred_test)})
    accu = r2_score(y_test,pred_test)
    if accu>maxAcc:
        maxAcc=accu
        maxRs=i
print("Best accuracy -",maxAcc,'Best Random state = ',maxRs)
```

```
Best accuracy - 0.4381943345303301 Best Random state =  80
```

Best Model

```
Prediction
r2 Score is : {} 0.6315734737010505
Cross Validation Score: [0.528079   0.62870466 0.79865651 0.49111379 0.68635612]
MAE: 47793.187811448406
MSE: 7388445127.846478
RMSE: 85956.06510215832
```



Hyper parameter tunning increase the accuracy

In [230]:
```
from sklearn.model_selection import GridSearchCV
```

In [231]:
```
parameters = {
    "n_estimators":[5,50,250,500],
    "max_depth":[1,3,5,7,9],
    "learning_rate":[0.01,0.1,1,10,100]
}
```

In [232]:
```
GCV=GridSearchCV(GradientBoostingRegressor(),parameters,cv=5)
GCV.fit(x_train,y_train)

GCV.best_params_
```

Out[232]:
```
{'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 500}
```

In [238]:
```
Final_model=GradientBoostingRegressor(learning_rate=0.1,max_depth=2,n_estimators=500)
Final_model.fit(x_train,y_train)
pred=Final_model.predict(x_test)
accuracy = r2_score(y_test,pred)
print(accuracy*100)
```

```
64.12066318624335
```

In [ ]:
```
#### accuracy has been increased
```

THANK YOU