In [1]:
```python
import pandas as pd
import numpy as np                  # For mathematical calculations
import seaborn as sns               # For data visualization
import matplotlib.pyplot as plt
import seaborn as sn                # For plotting graphs
import io
%matplotlib inline
import warnings                     # To ignore any warnings
warnings.filterwarnings("ignore")
```

In [2]:
```python
filepath2 = r"C:\Users\91623\OneDrive\Desktop\projects\car prices\car.csv"
df= pd.read_csv(filepath2)
print(df)
```

```
        Brand                              Model  Year  Selling_Price  \
0      Maruti                      Maruti 800 AC  2007          60000
1      Maruti           Maruti Wagon R LXI Minor  2007         135000
2     Hyundai               Hyundai Verna 1.6 SX  2012         600000
3      Datsun             Datsun RediGO T Option  2017         250000
4       Honda             Honda Amaze VX i-DTEC  2014         450000
...       ...                                ...   ...            ...
4335  Hyundai  Hyundai i20 Magna 1.4 CRDi (Diesel)  2014         409999
4336  Hyundai           Hyundai i20 Magna 1.4 CRDi  2014         409999
4337   Maruti                  Maruti 800 AC BSIII  2009         110000
4338  Hyundai     Hyundai Creta 1.6 CRDi SX Option  2016         865000
4339  Renault                     Renault KWID RXT  2016         225000

      KM_Driven    Fuel Seller_Type Transmission         Owner
0         70000  Petrol  Individual       Manual   First Owner
1         50000  Petrol  Individual       Manual   First Owner
2        100000  Diesel  Individual       Manual   First Owner
3         46000  Petrol  Individual       Manual   First Owner
4        141000  Diesel  Individual       Manual  Second Owner
...         ...     ...         ...          ...           ...
4335      80000  Diesel  Individual       Manual  Second Owner
4336      80000  Diesel  Individual       Manual  Second Owner
4337      83000  Petrol  Individual       Manual  Second Owner
4338      90000  Diesel  Individual       Manual   First Owner
4339      40000  Petrol  Individual       Manual   First Owner

[4340 rows x 9 columns]
```

In [3]:
```python
df.head()
```

Out[3]:

	Brand	Model	Year	Selling_Price	KM_Driven	Fuel	Seller_Type	Transmission	Owner
0	Maruti	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Brand          4340 non-null   object
 1   Model          4340 non-null   object
 2   Year           4340 non-null   int64
 3   Selling_Price  4340 non-null   int64
 4   KM_Driven      4340 non-null   int64
 5   Fuel           4340 non-null   object
 6   Seller_Type    4340 non-null   object
 7   Transmission   4340 non-null   object
 8   Owner          4340 non-null   object
dtypes: int64(3), object(6)
memory usage: 305.3+ KB
```

In [5]:
```python
df.describe()
```

Out[5]:

	Year	Selling_Price	KM_Driven
count	4340.000000	4.340000e+03	4340.000000
mean	2013.090783	5.041273e+05	66215.777419
std	4.215344	5.785487e+05	46644.102194
min	1992.000000	2.000000e+04	1.000000
25%	2011.000000	2.087498e+05	35000.000000
50%	2014.000000	3.500000e+05	60000.000000
75%	2016.000000	6.000000e+05	90000.000000
max	2020.000000	8.900000e+06	806599.000000

In [6]:
```python
df.columns = df.columns.str.lower()
```

In [7]:
```python
unique = [feature for feature in df.columns if len(df[feature].unique())>0 and len(df[feature].unique())<100]
for feature in unique:
    print("{} has {} unique values : {} {}".format(feature,len(df[feature].unique()),df[feature].unique(),"\n"))
```

```
brand has 29 unique values : ['Maruti' 'Hyundai' 'Datsun' 'Honda' 'Tata' 'Chevrolet' 'Toyota' 'Jaguar'
 'Mercedes-Benz' 'Audi' 'Skoda' 'Jeep' 'BMW' 'Mahindra' 'Ford' 'Nissan'
 'Renault' 'Fiat' 'Volkswagen' 'Volvo' 'Mitsubishi' 'Land' 'Daewoo' 'MG'
 'Force' 'Isuzu' 'OpelCorsa' 'Ambassador' 'Kia']

year has 27 unique values : [2007 2012 2017 2014 2016 2015 2018 2019 2013 2011 2010 2009 2006 1996
 2005 2008 2004 1998 2003 2002 2020 2000 1999 2001 1995 1997 1992]

fuel has 5 unique values : ['Petrol' 'Diesel' 'CNG' 'LPG' 'Electric']

seller_type has 3 unique values : ['Individual' 'Dealer' 'Trustmark Dealer']

transmission has 2 unique values : ['Manual' 'Automatic']

owner has 5 unique values : ['First Owner' 'Second Owner' 'Fourth & Above Owner' 'Third Owner'
 'Test Drive Car']
```

In [8]:
```python
df[['brand']].value_counts()
```

Out[8]:
```
brand
Maruti            1280
Hyundai            821
Mahindra           365
Tata               361
Honda              252
Ford               238
Toyota             206
Chevrolet          188
Renault            146
Volkswagen         107
Skoda               68
Nissan              64
Audi                60
BMW                 39
Fiat                37
Datsun              37
Mercedes-Benz       35
Mitsubishi           6
Jaguar               6
Land                 5
Volvo                4
Ambassador           4
Jeep                 3
MG                   2
OpelCorsa            2
Force                1
Isuzu                1
Daewoo               1
Kia                  1
dtype: int64
```

In [9]:
```python
df[['fuel']].value_counts()
```

Out[9]:
```
fuel
Diesel      2153
Petrol      2123
CNG           40
LPG           23
Electric       1
dtype: int64
```

In [10]:
```python
df[['owner']].value_counts()
```

Out[10]:
```
owner
First Owner             2832
Second Owner            1106
Third Owner              304
Fourth & Above Owner      81
Test Drive Car            17
dtype: int64
```

In [11]:
```python
df[['seller_type']].value_counts()
```

Out[11]:
```
seller_type
Individual          3244
Dealer               994
Trustmark Dealer     102
dtype: int64
```

In [12]:
```python
df[['transmission']].value_counts()
```

Out[12]:
```
transmission
Manual         3892
Automatic       448
dtype: int64
```

In [13]:
```python
df.columns
```

Out[13]:
```
Index(['brand', 'model', 'year', 'selling_price', 'km_driven', 'fuel',
       'seller_type', 'transmission', 'owner'],
      dtype='object')
```

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

Out[14]:
```
(4340, 9)
```

In [15]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [16]:
```python
df1=df
```

In [17]:
```python
list1=['fuel','seller_type','transmission','owner']
for i in list1:
    df1[i]=le.fit_transform(df1[i])

df1.head()
```

Out[17]:

	brand	model	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti	Maruti 800 AC	2007	60000	70000	4	1	1	0
1	Maruti	Maruti Wagon R LXI Minor	2007	135000	50000	4	1	1	0
2	Hyundai	Hyundai Verna 1.6 SX	2012	600000	100000	1	1	1	0
3	Datsun	Datsun RediGO T Option	2017	250000	46000	4	1	1	0
4	Honda	Honda Amaze VX i-DTEC	2014	450000	141000	1	1	1	2

In [18]:
```python
unique = [feature for feature in df1.columns if len(df1[feature].unique())>0 and len(df1[feature].unique())<100
for feature in unique:
    print("{} has {} unique values : {} {}".format(feature,len(df1[feature].unique()),df1[feature].unique(),"\n
```

```
brand has 29 unique values : ['Maruti' 'Hyundai' 'Datsun' 'Honda' 'Tata' 'Chevrolet' 'Toyota' 'Jaguar'
 'Mercedes-Benz' 'Audi' 'Skoda' 'Jeep' 'BMW' 'Mahindra' 'Ford' 'Nissan'
 'Renault' 'Fiat' 'Volkswagen' 'Volvo' 'Mitsubishi' 'Land' 'Daewoo' 'MG'
 'Force' 'Isuzu' 'OpelCorsa' 'Ambassador' 'Kia']

year has 27 unique values : [2007 2012 2017 2014 2016 2015 2018 2019 2013 2011 2010 2009 2006 1996
 2005 2008 2004 1998 2003 2002 2020 2000 1999 2001 1995 1997 1992]

fuel has 5 unique values : [4 1 0 3 2]

seller_type has 3 unique values : [1 0 2]

transmission has 2 unique values : [1 0]

owner has 5 unique values : [0 2 1 4 3]
```

In [ ]:

In [ ]:

In [19]:
```python
sns.scatterplot(data =df1['selling_price'])
```

Out[19]:
```
<AxesSubplot:ylabel='selling_price'>
```

In [20]:
```python
y=df['selling_price']
```

In [21]:
```python
y.shape
```

Out[21]:
```
(4340,)
```

In [22]:
```python
x=df1[["year","km_driven","fuel","transmission","owner"]]
```

In [23]:
```python
x.shape
```

Out[23]:
```
(4340, 5)
```

In [24]:
```python
x
```

Out[24]:

	year	km_driven	fuel	transmission	owner
0	2007	70000	4	1	0
1	2007	50000	4	1	0
2	2012	100000	1	1	0
3	2017	46000	4	1	0
4	2014	141000	1	1	2
...	...	...	...	...	...
4335	2014	80000	1	1	2
4336	2014	80000	1	1	2
4337	2009	83000	4	1	2
4338	2016	90000	1	1	0
4339	2016	40000	4	1	0

4340 rows × 5 columns

In [ ]:

In [ ]:

In [25]:
```python
from sklearn.model_selection import train_test_split
```

In [26]:
```python
x_train,x_test,y_train,y_test =train_test_split(x,y, test_size=0.3,random_state=2529)
```

In [27]:
```python
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[27]:
```
((3038, 5), (1302, 5), (3038,), (1302,))
```

linear regration

In [28]:
```python
from sklearn.linear_model import LinearRegression
```

In [29]:
```python
lr=LinearRegression()
```

In [30]:
```python
lr.fit(x_train,y_train)
```

Out[30]:
```
▾ LinearRegression
LinearRegression()
```

In [31]:
```python
y_pred = lr.predict(x_test)
```

In [32]:
```python
from sklearn.metrics import mean_squared_error , mean_absolute_error , r2_score
```

In [33]:
```python
mean_squared_error(y_test,y_pred)
```

Out[33]:
```
193206294338.9267
```

In [34]:
```python
mean_absolute_error(y_test,y_pred)
```

Out[34]:
```
233874.30747087934
```

In [35]:
```python
lin_reg=r2_score (y_test,y_pred)*100
lin_reg
```

Out[35]:
```
40.766878661465576
```

In [36]:
```python
y_test
```

Out[36]:
```
2613     300000
1311    1300000
2321     340000
1454     350000
4235     215000
          ...
249      500000
265      160000
1534     575000
848      250000
635      800000
Name: selling_price, Length: 1302, dtype: int64
```

In [37]:
```python
y_pred
```

Out[37]:
```
array([502128.0177722 , 709308.34602509, 506480.61003758, ...,
       694356.54476541, 380279.02445321, 792207.99351607])
```

random forest

In [38]:
```python
from sklearn.ensemble import RandomForestRegressor
```

In [39]:
```python
rfr=RandomForestRegressor()
rfr
```

Out[39]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [40]:
```python
rfr.fit(x_train,y_train)
```

Out[40]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [41]:
```python
rfr_pred=rfr.predict(x_test)
```

In [42]:
```python
rfr_score=r2_score(y_test,rfr_pred)*100
print("Accuracy score for LR :",rfr_score)
```

```
Accuracy score for LR : 53.45936561634799
```

Lasso

In [43]:
```python
from sklearn.linear_model import Lasso
```

In [44]:
```python
la=Lasso()
la
```

Out[44]:
```
▾ Lasso
Lasso()
```

In [45]:
```python
la.fit(x_train,y_train)
```

Out[45]:
```
▾ Lasso
Lasso()
```

In [46]:
```python
la_pred=la.predict(x_test)
```

In [47]:
```python
la_score=r2_score(y_test,la_pred)*100
print("Accuracy score for LR :",la_score)
```

```
Accuracy score for LR : 40.76690412586098
```

vector

In [48]:
```python
from sklearn.svm import SVR
```

In [49]:
```python
svr=SVR()
svr.fit(x_train,y_train)
```

Out[49]:
```
▾ SVR
SVR()
```

In [50]:
```python
svr_pred=svr.predict(x_test)
```

In [51]:
```python
from sklearn.metrics import r2_score

svr_score=r2_score(y_test,svr_pred)*100
print("Accuracy score for LR :",svr_score)
```

```
Accuracy score for LR : -4.245104766302887
```

In [52]:
```python
from sklearn.linear_model import Ridge
```

In [53]:
```python
rr=Ridge()
rr
```

Out[53]:
```
▾ Ridge
Ridge()
```

In [54]:
```python
rr.fit(x_train,y_train)
```

Out[54]:
```
▾ Ridge
Ridge()
```

In [55]:
```python
rr_pred=rr.predict(x_test)
```

In [56]:
```python
from sklearn.metrics import r2_score
rr_score=r2_score(y_test,rr_pred)*100
print("Accuracy score for RidgeR :",rr_score)
```

```
Accuracy score for RidgeR : 40.77093794036354
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: