In [1]:
```
import pandas as pd
import numpy as np
import re
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

```
C:\Users\Vignesh\anaconda3\lib\site-packages\pandas\core\computation\expre
ssions.py:21: UserWarning: Pandas requires version '2.8.0' or newer of 'nu
mexpr' (version '2.7.3' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
C:\Users\Vignesh\anaconda3\lib\site-packages\pandas\core\arrays\masked.py:
62: UserWarning: Pandas requires version '1.3.4' or newer of 'bottleneck'
(version '1.3.2' currently installed).
  from pandas.core import (
```

In [2]:
```
# pip install --upgrade scikit-learn
```

In [3]:

```
df_train = pd.read_csv(r'C:\Users\Vignesh\Desktop\Sem_3\Research\archiv
df_train.head(20)
```

Out[3]:

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	O
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	
5	5	Hyundai EON LPG Era Plus Option	Hyderabad	2012	75000	LPG	Manual	
6	6	Nissan Micra Diesel XV	Jaipur	2013	86999	Diesel	Manual	
7	7	Toyota Innova Crysta 2.8 GX AT 8S	Mumbai	2016	36000	Diesel	Automatic	
8	8	Volkswagen Vento Diesel Comfortline	Pune	2013	64430	Diesel	Manual	
9	9	Tata Indica Vista Quadrajet LS	Chennai	2012	65932	Diesel	Manual	
10	10	Maruti Ciaz Zeta	Kochi	2018	25692	Petrol	Manual	
11	11	Honda City 1.5 V AT Sunroof	Kolkata	2012	60000	Petrol	Automatic	
12	12	Maruti Swift VDI BSIV	Jaipur	2015	64424	Diesel	Manual	
13	13	Land Rover Range Rover 2.2L Pure	Delhi	2014	72000	Diesel	Automatic	
14	14	Land Rover Freelander 2 TD4 SE	Pune	2012	85000	Diesel	Automatic	
15	15	Mitsubishi Pajero Sport 4X4	Delhi	2014	110000	Diesel	Manual	
16	16	Honda Amaze S i-Dtech	Kochi	2016	58950	Diesel	Manual	
17	17	Maruti Swift DDiS VDI	Jaipur	2017	25000	Diesel	Manual	

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	O
18	18	Renault Duster 85PS Diesel RxL Plus	Kochi	2014	77469	Diesel	Manual	
19	19	Mercedes-Benz New C-Class C 220 CDI BE Avantgare	Bangalore	2014	78500	Diesel	Automatic	

In [4]:

```
df_test = pd.read_csv(r"C:\Users\Vignesh\Desktop\SEM_3\Research\archive
df_test.head(20)
```

Out[4]:

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
0	0	Maruti Alto K10 LXI CNG	Delhi	2014	40929	CNG	Manual	
1	1	Maruti Alto 800 2016-2019 LXI	Coimbatore	2013	54493	Petrol	Manual	
2	2	Toyota Innova Crysta Touring Sport 2.4 MT	Mumbai	2017	34000	Diesel	Manual	
3	3	Toyota Etios Liva GD	Hyderabad	2012	139000	Diesel	Manual	
4	4	Hyundai i20 Magna	Mumbai	2014	29000	Petrol	Manual	
5	5	Mahindra XUV500 W8 2WD	Coimbatore	2016	85609	Diesel	Manual	
6	6	Toyota Fortuner 4x2 AT TRD Sportivo	Pune	2015	59000	Diesel	Automatic	
7	7	Hyundai EON Era Plus	Jaipur	2013	65000	Petrol	Manual	
8	8	Honda City 1.5 S MT	Mumbai	2011	66000	Petrol	Manual	
9	9	Mahindra XUV500 W6 2WD	Coimbatore	2015	54684	Diesel	Manual	
10	10	Audi Q5 2008-2012 2.0 TDI	Mumbai	2012	78000	Diesel	Automatic	
11	11	Hyundai Grand i10 Magna	Jaipur	2016	21000	Petrol	Manual	
12	12	Toyota Corolla H5	Chennai	2007	90000	Petrol	Manual	
13	13	Maruti Swift Vdi BSIII	Coimbatore	2008	87628	Diesel	Manual	
14	14	Nissan Terrano XL	Mumbai	2014	45000	Petrol	Manual	
15	15	BMW X1 sDrive20d	Coimbatore	2013	30788	Diesel	Automatic	

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
16	16	BMW 3 Series GT 320d Luxury Line	Hyderabad	2015	39524	Diesel	Automatic	
17	17	Ford Ikon 1.4 TDCi DuraTorq	Chennai	2009	140000	Diesel	Manual	
18	18	Maruti Swift AMT ZXI	Kochi	2019	15409	Petrol	Automatic	
19	19	Maruti Swift Dzire VXi	Jaipur	2015	36502	Petrol	Manual	

In [5]:
```
1 df_train.columns
```

Out[5]:
```
Index(['Unnamed: 0', 'Name', 'Location', 'Year', 'Kilometers_Driven',
       'Fuel_Type', 'Transmission', 'Owner_Type', 'Mileage', 'Engine', 'Po
wer',
       'Seats', 'New_Price', 'Price'],
      dtype='object')
```

In [6]:
```
1 df_test.columns
```

Out[6]:
```
Index(['Unnamed: 0', 'Name', 'Location', 'Year', 'Kilometers_Driven',
       'Fuel_Type', 'Transmission', 'Owner_Type', 'Mileage', 'Engine', 'Po
wer',
       'Seats', 'New_Price'],
      dtype='object')
```

In [7]:
```
1 df_train.shape
```

Out[7]:
```
(6019, 14)
```

In [8]:
```
1 df_test.shape
```

Out[8]:
```
(1234, 13)
```

In [9]:
```
1 df_train.isnull().sum()
```

Out[9]:
```
Unnamed: 0             0
Name                   0
Location               0
Year                   0
Kilometers_Driven      0
Fuel_Type              0
Transmission           0
Owner_Type             0
Mileage                2
Engine                36
Power                 36
Seats                 42
New_Price           5195
Price                  0
dtype: int64
```

In [10]:
```
1 df_test.isnull().sum()
```

Out[10]:
```
Unnamed: 0             0
Name                   0
Location               0
Year                   0
Kilometers_Driven      0
Fuel_Type              0
Transmission           0
Owner_Type             0
Mileage                0
Engine                10
Power                 10
Seats                 11
New_Price           1052
dtype: int64
```

In [11]:
```
1 df_train.drop(['New_Price','Unnamed: 0'],axis = 1, inplace= True)
2 df_test.drop(['New_Price','Unnamed: 0'],axis= 1,inplace =True)
```

In [12]:
```
1 # null_valued_rows = df_train['Mileage'].isnull()
2 # rows_with_null = df_train[null_valued_rows]
3
4 # df_train.drop([4446,4904],inplace = True)
5
6 # df_train.isnull().sum()
```

In [13]:
```
1 df_train['Engine'] = df_train['Engine'].str.replace(r'\s*CC', '', regex
2 df_train['Power'] = df_train['Power'].str.replace(r'\s*bhp', '', regex=
3 df_train['Mileage'] = df_train['Mileage'].str.replace(r'\s*(km/kg|kmpl)
4
5 df_test['Engine']  = df_test['Engine'].str.replace(r'\s*CC', '', regex=
6 df_test['Power'] = df_test['Power'].str.replace(r'\s*bhp', '', regex=Tr
7 df_test['Mileage'] = df_test['Mileage'].str.replace(r'\s*(km/kg|kmpl)',
```

In [14]:
```
df_train['Engine'] = pd.to_numeric(df_train['Engine'], errors='coerce',
df_train['Mileage'] = pd.to_numeric(df_train['Mileage'], errors='coerce
df_train['Power'] = pd.to_numeric(df_train['Power'], errors='coerce', c
df_train['Seats'] = pd.to_numeric(df_train['Seats'], errors='coerce', c

df_test['Engine'] = pd.to_numeric(df_test['Engine'], errors='coerce', c
df_test['Mileage'] = pd.to_numeric(df_test['Mileage'], errors='coerce',
df_test['Power'] = pd.to_numeric(df_test['Power'], errors='coerce', dow
df_test['Seats'] = pd.to_numeric(df_test['Seats'], errors='coerce', dow
```

In [15]:
```
df_train['Engine'].fillna(df_train['Engine'].mean(),inplace = True)
df_train['Mileage'].fillna(df_train['Mileage'].mean(),inplace = True)
df_train['Power'].fillna(df_train['Power'].mean(),inplace = True)
df_train['Seats'].fillna(df_train['Seats'].mean(), inplace=True)

df_test['Engine'].fillna(df_test['Engine'].mean(),inplace = True)
df_test['Mileage'].fillna(df_test['Mileage'].mean(),inplace = True)
df_test['Power'].fillna(df_test['Power'].mean(),inplace = True)
df_test['Seats'].fillna(df_test['Seats'].mean(), inplace=True)
```

In [16]:
```
1 df_train.head(10)
```

Out[16]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	M
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	
5	Hyundai EON LPG Era Plus Option	Hyderabad	2012	75000	LPG	Manual	First	
6	Nissan Micra Diesel XV	Jaipur	2013	86999	Diesel	Manual	First	
7	Toyota Innova Crysta 2.8 GX AT 8S	Mumbai	2016	36000	Diesel	Automatic	First	
8	Volkswagen Vento Diesel Comfortline	Pune	2013	64430	Diesel	Manual	First	
9	Tata Indica Vista Quadrajet LS	Chennai	2012	65932	Diesel	Manual	Second	

In [17]:
```
1 df_test.head(10)
```

Out[17]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mil
0	Maruti Alto K10 LXI CNG	Delhi	2014	40929	CNG	Manual	First	3
1	Maruti Alto 800 2016-2019 LXI	Coimbatore	2013	54493	Petrol	Manual	Second	2
2	Toyota Innova Crysta Touring Sport 2.4 MT	Mumbai	2017	34000	Diesel	Manual	First	1
3	Toyota Etios Liva GD	Hyderabad	2012	139000	Diesel	Manual	First	2
4	Hyundai i20 Magna	Mumbai	2014	29000	Petrol	Manual	First	1
5	Mahindra XUV500 W8 2WD	Coimbatore	2016	85609	Diesel	Manual	Second	1
6	Toyota Fortuner 4x2 AT TRD Sportivo	Pune	2015	59000	Diesel	Automatic	First	1
7	Hyundai EON Era Plus	Jaipur	2013	65000	Petrol	Manual	First	2
8	Honda City 1.5 S MT	Mumbai	2011	66000	Petrol	Manual	Second	1
9	Mahindra XUV500 W6 2WD	Coimbatore	2015	54684	Diesel	Manual	First	1

In [18]:
```
1 df_train.isnull().sum()
```

Out[18]:
```
Name                0
Location            0
Year                0
Kilometers_Driven   0
Fuel_Type           0
Transmission        0
Owner_Type          0
Mileage             0
Engine              0
Power               0
Seats               0
Price               0
dtype: int64
```

In [19]:
```
1 df_test.isnull().sum()
```

Out[19]:
```
Name                0
Location            0
Year                0
Kilometers_Driven   0
Fuel_Type           0
Transmission        0
Owner_Type          0
Mileage             0
Engine              0
Power               0
Seats               0
dtype: int64
```

In [20]:
```
1 print(df_train['Name'][0].split()[0])
```

```
Maruti
```

In [21]:
```
1 df_train['Brands'] = df_train['Name'].apply(lambda x: x.split()[0])
2
3 df_test['Brands'] = df_test['Name'].apply(lambda x: x.split()[0])
```

In [22]:
```
1 df_train.head(20)
```

Out[22]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second
5	Hyundai EON LPG Era Plus Option	Hyderabad	2012	75000	LPG	Manual	First
6	Nissan Micra Diesel XV	Jaipur	2013	86999	Diesel	Manual	First
7	Toyota Innova Crysta 2.8 GX AT 8S	Mumbai	2016	36000	Diesel	Automatic	First
8	Volkswagen Vento Diesel Comfortline	Pune	2013	64430	Diesel	Manual	First
9	Tata Indica Vista Quadrajet LS	Chennai	2012	65932	Diesel	Manual	Second
10	Maruti Ciaz Zeta	Kochi	2018	25692	Petrol	Manual	First
11	Honda City 1.5 V AT Sunroof	Kolkata	2012	60000	Petrol	Automatic	First
12	Maruti Swift VDI BSIV	Jaipur	2015	64424	Diesel	Manual	First
13	Land Rover Range Rover 2.2L Pure	Delhi	2014	72000	Diesel	Automatic	First
14	Land Rover Freelander 2 TD4 SE	Pune	2012	85000	Diesel	Automatic	Second
15	Mitsubishi Pajero Sport 4X4	Delhi	2014	110000	Diesel	Manual	First
16	Honda Amaze S i-Dtech	Kochi	2016	58950	Diesel	Manual	First
17	Maruti Swift DDiS VDI	Jaipur	2017	25000	Diesel	Manual	First

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type
18	Renault Duster 85PS Diesel RxL Plus	Kochi	2014	77469	Diesel	Manual	First
19	Mercedes-Benz New C-Class C 220 CDI BE Avantgare	Bangalore	2014	78500	Diesel	Automatic	First

In [23]:
```
1 df_train['Brands'].value_counts()
```

Out[23]:
```
Brands
Maruti           1211
Hyundai          1107
Honda             608
Toyota            411
Mercedes-Benz     318
Volkswagen        315
Ford              300
Mahindra          272
BMW               267
Audi              236
Tata              186
Skoda             173
Renault           145
Chevrolet         121
Nissan             91
Land               60
Jaguar             40
Fiat               28
Mitsubishi         27
Mini               26
Volvo              21
Porsche            18
Jeep               15
Datsun             13
Force               3
ISUZU               2
Smart               1
Ambassador          1
Isuzu               1
Bentley             1
Lamborghini         1
Name: count, dtype: int64
```

In [24]:
```
1 df_test.head(20)
```

Out[24]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	M
0	Maruti Alto K10 LXI CNG	Delhi	2014	40929	CNG	Manual	First	
1	Maruti Alto 800 2016-2019 LXI	Coimbatore	2013	54493	Petrol	Manual	Second	
2	Toyota Innova Crysta Touring Sport 2.4 MT	Mumbai	2017	34000	Diesel	Manual	First	
3	Toyota Etios Liva GD	Hyderabad	2012	139000	Diesel	Manual	First	
4	Hyundai i20 Magna	Mumbai	2014	29000	Petrol	Manual	First	
5	Mahindra XUV500 W8 2WD	Coimbatore	2016	85609	Diesel	Manual	Second	
6	Toyota Fortuner 4x2 AT TRD Sportivo	Pune	2015	59000	Diesel	Automatic	First	
7	Hyundai EON Era Plus	Jaipur	2013	65000	Petrol	Manual	First	
8	Honda City 1.5 S MT	Mumbai	2011	66000	Petrol	Manual	Second	
9	Mahindra XUV500 W6 2WD	Coimbatore	2015	54684	Diesel	Manual	First	
10	Audi Q5 2008-2012 2.0 TDI	Mumbai	2012	78000	Diesel	Automatic	Second	
11	Hyundai Grand i10 Magna	Jaipur	2016	21000	Petrol	Manual	First	
12	Toyota Corolla H5	Chennai	2007	90000	Petrol	Manual	Third	
13	Maruti Swift Vdi BSIII	Coimbatore	2008	87628	Diesel	Manual	First	
14	Nissan Terrano XL	Mumbai	2014	45000	Petrol	Manual	First	
15	BMW X1 sDrive20d	Coimbatore	2013	30788	Diesel	Automatic	First	

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	M
16	BMW 3 Series GT 320d Luxury Line	Hyderabad	2015	39524	Diesel	Automatic	First	
17	Ford Ikon 1.4 TDCi DuraTorq	Chennai	2009	140000	Diesel	Manual	First	
18	Maruti Swift AMT ZXI	Kochi	2019	15409	Petrol	Automatic	First	
19	Maruti Swift Dzire VXi	Jaipur	2015	36502	Petrol	Manual	First	

In [25]:
```
1 df_train['Engine'].value_counts().count()
```

Out[25]: 147

In [26]:
```
1 df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1234 entries, 0 to 1233
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               1234 non-null   object 
 1   Location           1234 non-null   object 
 2   Year               1234 non-null   int64  
 3   Kilometers_Driven  1234 non-null   int64  
 4   Fuel_Type          1234 non-null   object 
 5   Transmission       1234 non-null   object 
 6   Owner_Type         1234 non-null   object 
 7   Mileage            1234 non-null   float64
 8   Engine             1234 non-null   float64
 9   Power              1234 non-null   float32
 10  Seats              1234 non-null   float32
 11  Brands             1234 non-null   object 
dtypes: float32(2), float64(2), int64(2), object(6)
memory usage: 106.2+ KB
```

In [27]:
```
1 df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               6019 non-null   object 
 1   Location           6019 non-null   object 
 2   Year               6019 non-null   int64  
 3   Kilometers_Driven  6019 non-null   int64  
 4   Fuel_Type          6019 non-null   object 
 5   Transmission       6019 non-null   object 
 6   Owner_Type         6019 non-null   object 
 7   Mileage            6019 non-null   float64
 8   Engine             6019 non-null   float64
 9   Power              6019 non-null   float32
 10  Seats              6019 non-null   float32
 11  Price              6019 non-null   float64
 12  Brands             6019 non-null   object 
dtypes: float32(2), float64(3), int64(2), object(6)
memory usage: 564.4+ KB
```

In [28]:
```
1 df_train['Owner_Type'].value_counts()
```

Out[28]:
```
Owner_Type
First             4929
Second             968
Third              113
Fourth & Above       9
Name: count, dtype: int64
```

In [29]:
```
1 def owner_in_INT(x):
2     if x == 'First':
3         return 1
4     elif x == 'Second':
5         return 2
6     elif x == 'Third':
7         return 3
8     else:
9         return 4
```

In [30]:
```
1 df_train['Owner_Type'] = df_train['Owner_Type'].apply(lambda x: owner_i
2 df_test['Owner_Type'] = df_test['Owner_Type'].apply(lambda x: owner_in_
```

In [31]:
```
1 df_train['Transmission'].value_counts()
```

Out[31]:
```
Transmission
Manual       4299
Automatic    1720
Name: count, dtype: int64
```

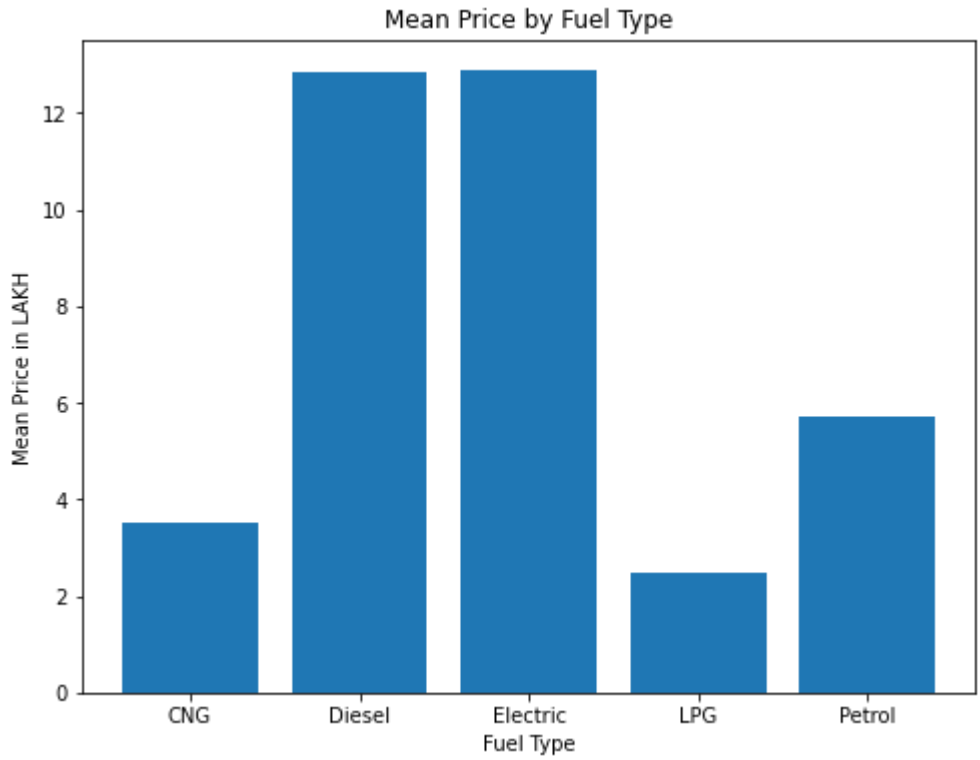
In [32]:
```
1 df_train['Fuel_Type'].value_counts()
```

Out[32]:
```
Fuel_Type
Diesel      3205
Petrol      2746
CNG           56
LPG           10
Electric       2
Name: count, dtype: int64
```

In [33]:
```
1 import matplotlib.pyplot as plt
```

In [34]:
```
1 grouped = df_train.groupby('Fuel_Type')['Price'].mean().reset_index()
```

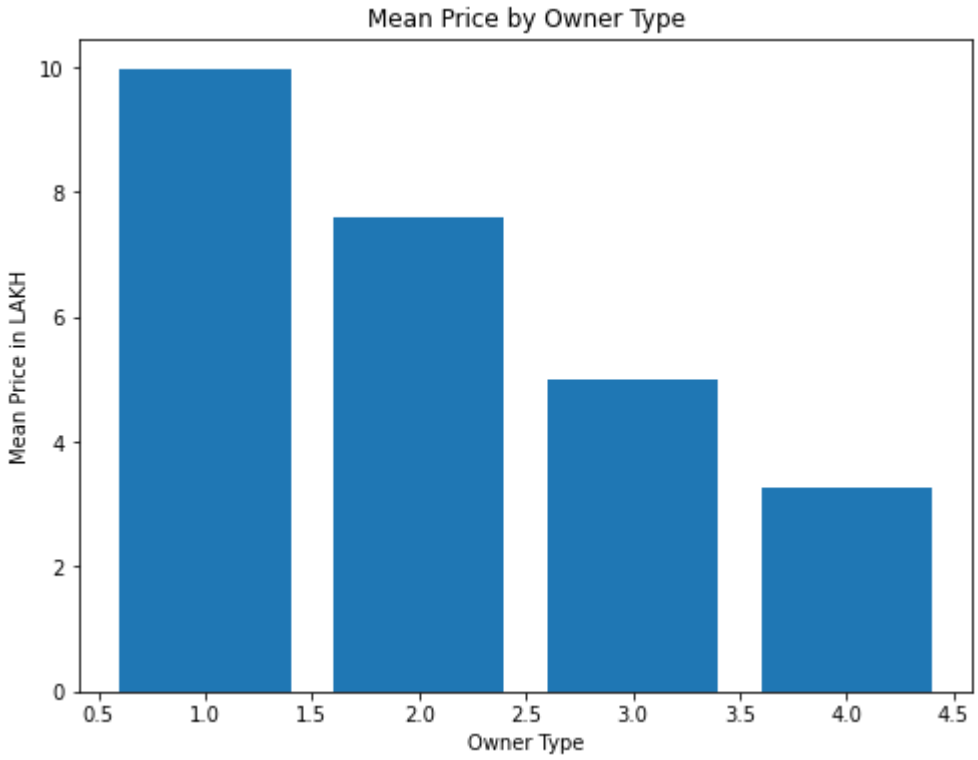
In [35]:
```
1 plt.figure(figsize=(8, 6))
2 plt.bar(grouped['Fuel_Type'], grouped['Price'])
3 plt.xlabel('Fuel Type')
4 plt.ylabel('Mean Price in LAKH')
5 plt.title('Mean Price by Fuel Type')
6 plt.show()
```



In [36]:
```
1 grouped = df_train.groupby('Owner_Type')['Price'].mean().reset_index()
```

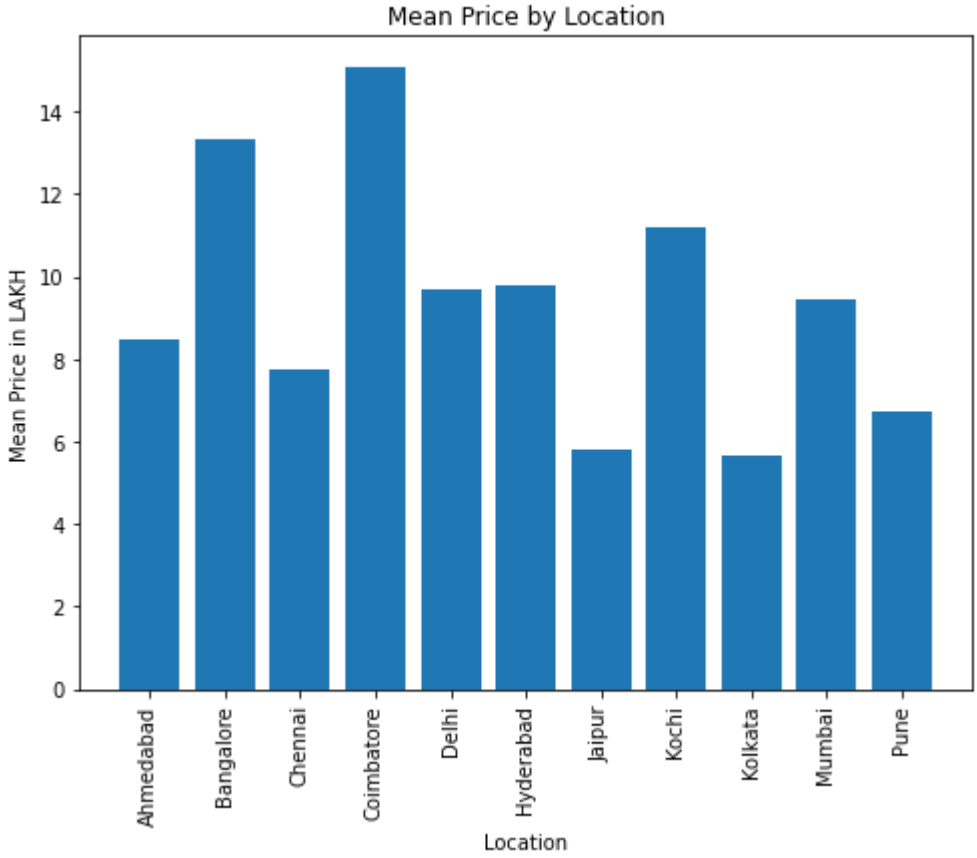
In [37]:

```
plt.figure(figsize=(8, 6))
plt.bar(grouped['Owner_Type'], grouped['Price'])
plt.xlabel('Owner Type')
plt.ylabel('Mean Price in LAKH')
plt.title('Mean Price by Owner Type')
plt.show()
```

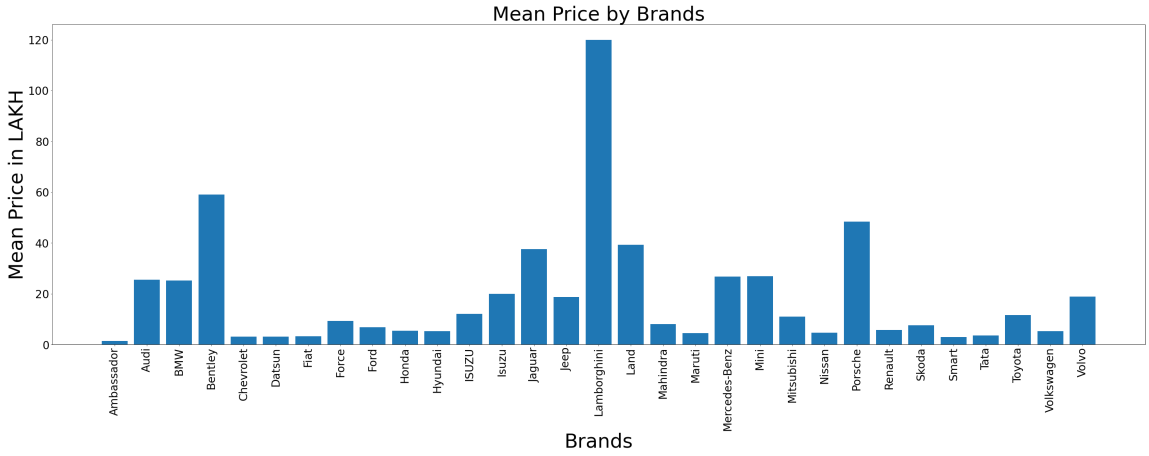


In [38]:

```
grouped = df_train.groupby('Location')['Price'].mean().reset_index()
plt.figure(figsize=(8, 6))
plt.bar(grouped['Location'], grouped['Price'])
plt.xlabel('Location')
plt.ylabel('Mean Price in LAKH')
plt.title('Mean Price by Location')
plt.xticks(rotation=90)
plt.show()
```



In [39]:
```
grouped = df_train.groupby('Brands')['Price'].mean().reset_index()
plt.figure(figsize=(50, 15))
plt.bar(grouped['Brands'], grouped['Price'])
plt.xlabel('Brands',fontsize=50)
plt.ylabel('Mean Price in LAKH',fontsize=50)
plt.title('Mean Price by Brands',fontsize=50)
plt.xticks(rotation=90)
plt.xticks(fontsize=28)
plt.yticks(fontsize=28)

plt.show()
```



In [40]:
```
df_train['Brands'] = df_train['Brands'].str.lower()
df_test['Brands'] = df_test['Brands'].str.lower()
```

In [41]:

```
# serties of brand acording to its popuilarity
# ['Maruti','Hyundai','Honda','Toyota','Mahindra','Tata','Renault','For

brands_list = ['Maruti','Hyundai','Honda','Toyota','Mahindra','Tata','F

series0f_34  = np.arange(34)
brands2 = {
    'Popularity':series0f_34,
    'Brands':brands_list
}

brands2 = pd.DataFrame(brands2)
brands2['Brands'] = brands2['Brands'].str.lower()
brands2
```

Out[41]:

	Popularity	Brands
0	0	maruti
1	1	hyundai
2	2	honda
3	3	toyota
4	4	mahindra
5	5	tata
6	6	renault
7	7	ford
8	8	nissan
9	9	volkswagen
10	10	skoda
11	11	fiat
12	12	mercedes-benz
13	13	bmw
14	14	audi
15	15	kia
16	16	mg motors
17	17	jeep
18	18	datsun
19	19	land
20	20	jaguar
21	21	mitsubishi
22	22	mini
23	23	volvo
24	24	porsche
25	25	force
26	26	chevrolet
27	27	isuzu
28	28	smart
29	29	bentley
30	30	lamborghini
31	31	ambassador
32	32	hindustan
33	33	opelcorsa

In [42]:
```
df_train = pd.merge(df_train, brands2, on='Brands', how='left')
df_test = pd.merge(df_test, brands2, on='Brands', how='left')

df_train
```

Out[42]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	1
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	1
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	1
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	1
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	2
...	...	...	...	...	...	...	...
6014	Maruti Swift VDI	Delhi	2014	27365	Diesel	Manual	1
6015	Hyundai Xcent 1.1 CRDi S	Jaipur	2015	100000	Diesel	Manual	1
6016	Mahindra Xylo D4 BSIV	Jaipur	2012	55000	Diesel	Manual	2
6017	Maruti Wagon R VXI	Kolkata	2013	46000	Petrol	Manual	1
6018	Chevrolet Beat Diesel	Hyderabad	2011	47000	Diesel	Manual	1

6019 rows × 14 columns

In [43]:
```
len(df_train['Name'].value_counts())
```

Out[43]: 1876

In [44]:
```
df_test[df_test['Popularity'].isnull()]
```

Out[44]:

Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage

In [45]:
```
# df = df.rename(columns={'Mileage': 'Mileage(kmpl)', 'Engine': 'Engine
```

In [46]:
```
# bin_size = 20000
# df['Driven_Class'] = pd.cut(df['Kilometers_Driven'], bins=range(0, ma

# grouped_data = df['Driven_Class'].value_counts().sort_index()
# grouped_data.plot(kind='bar')

# plt.xlabel('Driven_Class')
# plt.ylabel('Count')
# plt.title('Bar Graph of Continuous Numbers Grouped by Bins')
# plt.xticks(rotation=45)  # Rotate x-axis labels for better readabilit
# plt.xticks(rotation=90)
# plt.show()
```

In [ ]:
```

```

In [47]:
```
# print('MIN:',df['Kilometers_Driven'].min())
# print('MAX:',df['Kilometers_Driven'].max())
```

In [48]:
```
#cov_mat = df.cov()
#cov_mat
```

In [49]:
```
df_train.columns
```

Out[49]:
```
Index(['Name', 'Location', 'Year', 'Kilometers_Driven', 'Fuel_Type',
       'Transmission', 'Owner_Type', 'Mileage', 'Engine', 'Power', 'Seat
s',
       'Price', 'Brands', 'Popularity'],
      dtype='object')
```

In [50]:
```
df_test.columns
```

Out[50]:
```
Index(['Name', 'Location', 'Year', 'Kilometers_Driven', 'Fuel_Type',
       'Transmission', 'Owner_Type', 'Mileage', 'Engine', 'Power', 'Seat
s',
       'Brands', 'Popularity'],
      dtype='object')
```

Label Encoding

In [51]:
```
# Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

In [52]:
```
label_encoder = LabelEncoder()
```

In [53]:
```
columns_to_label_encode_for_df_train = ['Name',
                            'Location',
                            'Year',
                            'Kilometers_Driven',
                            'Fuel_Type',
                            'Transmission',
                            'Owner_Type',
                            'Mileage',
                            'Engine',
                            'Power',
                            'Seats',
                            'Brands',
                            'Popularity',
                            'Price'
                           ]
columns_to_label_encode_for_df_test = ['Name',
                                       'Location',
                                       'Year',
                                       'Kilometers_Driven',
                                       'Fuel_Type',
                                       'Transmission',
                                       'Owner_Type',
                                       'Mileage',
                                       'Engine',
                                       'Power',
                                       'Seats',
                                       'Brands',
                                       'Popularity'
                                      ]
```

In [54]:
```
# label_encoders = {}
# label_encoders2 = {}
ENCODED_df_train = pd.DataFrame()
ENCODED_df_test = pd.DataFrame()

for column in columns_to_label_encode_for_df_train:
    label_encoder = LabelEncoder()
    ENCODED_df_train[column] = label_encoder.fit_transform(df_train[col
#     label_encoders[column] = label_encoder

for column in columns_to_label_encode_for_df_test:
    label_encoder = LabelEncoder()
    ENCODED_df_test[column] = label_encoder.fit_transform(df_test[colum
#     label_encoders2[column] = label_encoder
```

In [55]:
```
1 ENCODED_df_train
```

Out[55]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Milea
0	1200	9	12	2362	0	1	0	4
1	512	10	17	1128	1	1	0	2
2	486	2	13	1356	4	1	0	2
3	1059	2	14	2693	1	1	0	3
4	23	3	15	1120	1	0	1	1
...	...	...	...	...	...	...	...	
6014	1159	4	16	596	1	1	0	4
6015	668	6	17	2828	1	1	0	3
6016	932	6	14	1709	1	1	1	1
6017	1207	8	15	1356	4	1	0	2
6018	165	5	13	1401	1	1	0	4

6019 rows × 14 columns

In [56]:
```
1 ENCODED_df_test
```

Out[56]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Milea
0	413	4	14	274	0	1	0	2
1	408	3	13	384	3	1	1	2
2	723	9	17	211	1	1	0	
3	689	5	12	723	1	1	0	2
4	333	9	14	161	3	1	0	1
...	...	...	...	...	...	...	...	
1229	758	5	11	640	1	1	0	2
1230	745	9	15	430	3	0	0	1
1231	582	8	12	155	1	1	0	2
1232	745	10	13	366	3	0	2	1
1233	536	7	14	549	1	0	0	

1234 rows × 13 columns

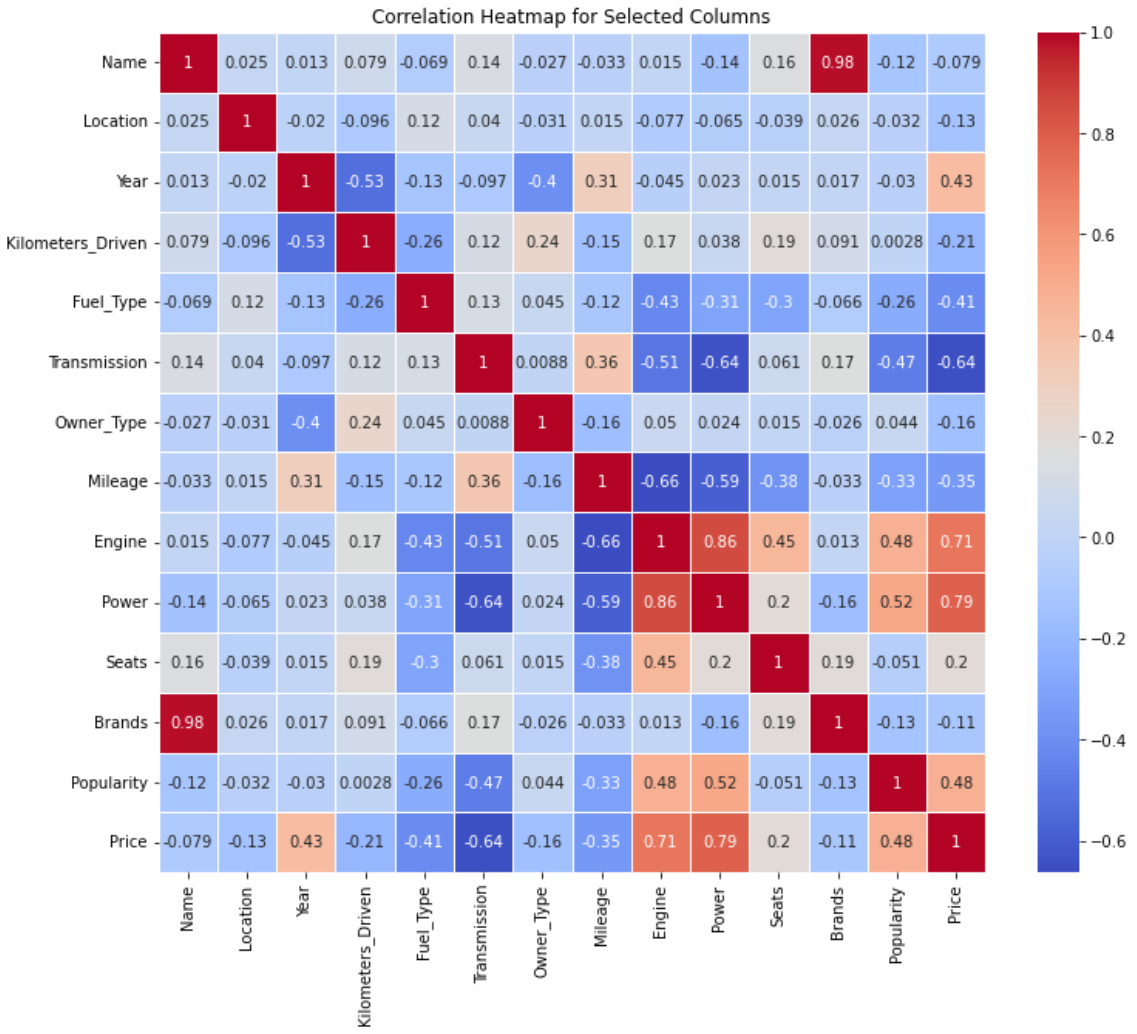
In [57]:
```
1 ENCODED_df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Name               6019 non-null   int32
 1   Location           6019 non-null   int32
 2   Year               6019 non-null   int64
 3   Kilometers_Driven  6019 non-null   int64
 4   Fuel_Type          6019 non-null   int32
 5   Transmission       6019 non-null   int32
 6   Owner_Type         6019 non-null   int64
 7   Mileage            6019 non-null   int64
 8   Engine             6019 non-null   int64
 9   Power              6019 non-null   int64
 10  Seats              6019 non-null   int64
 11  Brands             6019 non-null   int32
 12  Popularity         6019 non-null   int64
 13  Price              6019 non-null   int64
dtypes: int32(5), int64(9)
memory usage: 540.9 KB
```

In [58]:
```
1 ENCODED_df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1234 entries, 0 to 1233
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Name               1234 non-null   int32
 1   Location           1234 non-null   int32
 2   Year               1234 non-null   int64
 3   Kilometers_Driven  1234 non-null   int64
 4   Fuel_Type          1234 non-null   int32
 5   Transmission       1234 non-null   int32
 6   Owner_Type         1234 non-null   int64
 7   Mileage            1234 non-null   int64
 8   Engine             1234 non-null   int64
 9   Power              1234 non-null   int64
 10  Seats              1234 non-null   int64
 11  Brands             1234 non-null   int32
 12  Popularity         1234 non-null   int64
dtypes: int32(5), int64(8)
memory usage: 101.4 KB
```

In [59]:
```
# columns_to_include = ['Mileage', 'Power', 'Price', 'ENCODED_Location'

# Create a DataFrame with only the selected columns
selected_columns_df = ENCODED_df_train[columns_to_label_encode_for_df_t

# Calculate the correlation matrix
correlation_matrix = selected_columns_df.corr()

# Create a heatmap using Seaborn
plt.figure(figsize=(12, 10))  # Set the figure size
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths

# Set a title
plt.title('Correlation Heatmap for Selected Columns')

# Display the heatmap
plt.show()
```



In [60]:
```
ENCODED_df_train.columns
```

Out[60]:
```
Index(['Name', 'Location', 'Year', 'Kilometers_Driven', 'Fuel_Type',
       'Transmission', 'Owner_Type', 'Mileage', 'Engine', 'Power', 'Seat
s',
       'Brands', 'Popularity', 'Price'],
      dtype='object')
```

In [61]:
```
ENCODED_df_test.columns
```

Out[61]:
```
Index(['Name', 'Location', 'Year', 'Kilometers_Driven', 'Fuel_Type',
       'Transmission', 'Owner_Type', 'Mileage', 'Engine', 'Power', 'Seat
s',
       'Brands', 'Popularity'],
      dtype='object')
```

In [62]:
```
# Split the data into features (X) and the target variable (y)
X = ENCODED_df_train[['Name', 'Location', 'Year', 'Kilometers_Driven',
       'Transmission', 'Owner_Type', 'Mileage', 'Engine', 'Power', 'Sea
       'Brands', 'Popularity']]
y = ENCODED_df_train['Price']

# for test dataset
XT =ENCODED_df_test[['Name', 'Location', 'Year', 'Kilometers_Driven', '
       'Transmission', 'Owner_Type', 'Mileage', 'Engine', 'Power', 'Sea
       'Brands', 'Popularity']]
```

In [63]:
```
X
```

Out[63]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Milea
0	1200	9	12	2362	0	1	0	4
1	512	10	17	1128	1	1	0	2
2	486	2	13	1356	4	1	0	2
3	1059	2	14	2693	1	1	0	3
4	23	3	15	1120	1	0	1	1
...	...	...	...	...	...	...	...	
6014	1159	4	16	596	1	1	0	4
6015	668	6	17	2828	1	1	0	3
6016	932	6	14	1709	1	1	1	1
6017	1207	8	15	1356	4	1	0	2
6018	165	5	13	1401	1	1	0	4

6019 rows × 13 columns

In [64]:
```
#encoded_train_dataset = df[['Kilometers_Driven', 'ENCODED_Location','M
#         'ENCODED_Year', 'ENCODED_Fuel_Type', 'ENCODED_Transmission',
#         'ENCODED_Owner_Type', 'ENCODED_Engine', 'ENCODED_Seats',
#         'ENCODED_Brands','Price']]
#encoded_test_dataset = df_test[['Kilometers_Driven', 'ENCODED_Location
#         'ENCODED_Year', 'ENCODED_Fuel_Type', 'ENCODED_Transmission',
#         'ENCODED_Owner_Type', 'ENCODED_Engine', 'ENCODED_Seats',
#         'ENCODED_Brands','Price']]
```

In [65]:
```
# ENCODED_df_train.to_csv('ENCODED_DATASET/ENCODED_df_train.csv')
```

In [66]:
```
# ENCODED_df_test.to_csv('ENCODED_DATASET/ENCODED_df_test.csv')
```

In [67]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
```

In [68]:
```
# Create a Random Forest Regressor model
rf_model = RandomForestRegressor(n_estimators=300, random_state=59)  #
```

In [69]:
```
# Train the model on the training data
rf_model.fit(X_train, y_train)
```

Out[69]:
```
RandomForestRegressor(n_estimators=300, random_state=59)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [70]:
```
# Make predictions on the test data
y_pred = rf_model.predict(X_test)
```

In [71]:
```
# Evaluate the model
print(f'mean_squared_error:\t{mean_squared_error(y_test, y_pred)}')
print(f'r2_score:\t {r2_score(y_test, y_pred)}')
```

```
mean_squared_error:     5816.906048340954
r2_score:        0.9518512132046182
```

In [72]:
```
# df_test_preprocessed = preprocessor.transform(df_test)
# log_price = model.predict(df_test_preprocessed) # in log scale
# price = np.expm1(log_price) # in original scale

y_test_price = rf_model.predict(XT)



# Price In dollars
# price_usd = price[0] * 1219
# print('Price in USD:', price_usd)
```

In [73]:
```
##Price Lakh
# print('Price : ', price)
```

In [74]:
```
len(y_test_price)
```

Out[74]:
```
1234
```

In [75]:
```
# df_test['Price'] = price
```

In [76]:
```
# df_test
```

In [77]:
```
# yT = df_test['Price']
yT = y_test_price
```

In [78]:
```
X_train = X
y_train = y
x_test = XT
y_test = yT
```

In [79]:
```
X_train.shape
```

Out[79]: (6019, 13)

In [80]:
```
y_train.shape
```

Out[80]: (6019,)

In [81]:
```
x_test.shape
```

Out[81]: (1234, 13)

In [82]:
```
y_test.shape
```

Out[82]: (1234,)

In [83]:
```
# from sklearn.svm import SVR
```

In [84]:
```
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0
```

In [85]:
```
rf_model = RandomForestRegressor(n_estimators=300, random_state=59)
```

In [86]:
```
rf_model.fit(X_train, y_train)
```

Out[86]: RandomForestRegressor(n_estimators=300, random_state=59)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [87]:
```
y_pred = rf_model.predict(x_test)
```

In [88]:
```
len(y_pred)
```

Out[88]: 1234

In [89]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 833.1576971262322
R-squared: 0.9871682187180083
```

In [90]:
```
# Feature importance
feature_importances = rf_model.feature_importances_
feature_names = X.columns
```

In [91]:
```
r2 = r2_score(y_test, y_pred)
print(f"R-squared: {r2}")
```

```
R-squared: 0.9871682187180083
```

In [92]:
```
# SVR(kernel='rbf', C=100, epsilon=0.1)
# R-squared: 0.281805700964871


```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [93]:
```
# X = df[['Kilometers_Driven', 'ENCODED_Location', 'Mileage', 'ENCODED_
# y = df['Price']
```

In [94]:
```
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

```

In [95]:
```
# import numpy as np
# import pandas as pd
# from sklearn.model_selection import train_test_split
# from sklearn.preprocessing import StandardScaler
# from sklearn.svm import SVR
# from sklearn.metrics import mean_squared_error, r2_score
# from sklearn.model_selection import GridSearchCV

```

In [96]:
```
# scaler = StandardScaler()
# X_train = scaler.fit_transform(X_train)
# X_test = scaler.transform(X_test)
```

In [97]:
```
# svr = SVR()
```

In [98]:
```
# param_grid = {
#     'C': [0.1, 1, 10],
#     'kernel': ['linear', 'rbf', 'poly'],
#     'gamma': ['scale', 'auto'] + [0.001, 0.01, 0.1, 1],
#     'epsilon': [0.01, 0.1, 1],
# }
```

In [99]:
```
# grid_search = GridSearchCV(svr, param_grid, cv=5, scoring='r2', n_job
# grid_search.fit(X_train, y_train)
```

In [100]:
```
# best_svr = grid_search.best_estimator_
```

In [101]:
```
# best_svr.fit(X_train, y_train)
```

In [102]:
```
# y_pred = best_svr.predict(X_test)
```

In [103]:
```
# mse = mean_squared_error(y_test, y_pred)
# r2 = r2_score(y_test, y_pred)
```

In [104]:
```
# print(f"Mean Squared Error: {mse}")
# print(f"R-squared: {r2}")
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:

```
# 300 ********************** 10%
# Mean Squared Error: 0.7012467287979528
#R-squared: 0.9785732740875935
-------------------------------------------------------------------------
# 300 ****************** 15%
# Mean Squared Error: 0.870868331813521
#R-squared: 0.9748220299187007
-------------------------------------------------------------------------
# 500 ************** 15%
# Mean Squared Error: 0.6259481733495921
#R-squared: 0.9817243918662796
-------------------------------------------------------------------------
# 1000 ********** 15%
# Mean Squared Error: 0.561442539478491
#R-squared: 0.9835857151869357
-------------------------------------------------------------------------
# 2000 ******** 15%
# Mean Squared Error: 0.5782290706143449
#R-squared: 0.9831744894732686
-------------------------------------------------------------------------
# 3000 ******* 15%
# Mean Squared Error: 0.5713350853500679
#R-squared: 0.9833754012450003
-------------------------------------------------------------------------
# 10000 ***** 15%
# Mean Squared Error: 0.5438541424724417
#R-squared: 0.984096368565446
-------------------------------------------------------------------------
# 9500 * 25%
# Mean Squared Error: 1.9506152371726249
#R-squared: 0.95150844330424
-------------------------------------------------------------------------
# 8800 * 25%
# Mean Squared Error: 1.9681587827521663
#R-squared: 0.9510333953334813
-------------------------------------------------------------------------
# 8200 * 25%
# Mean Squared Error: 1.9862523679636237
#R-squared: 0.9506367771136948
-------------------------------------------------------------------------
# 1200 * 25%
# Mean Squared Error: 2.018974274723978
#R-squared: 0.949753767806515
-------------------------------------------------------------------------
# 1000 ** 25%
# Mean Squared Error: 2.089248999998699
#R-squared: 0.9480803688109836
-------------------------------------------------------------------------
# 20000
# Mean Squared Error: 11.472260994326327
#R-squared: 0.888171352023855
-------------------------------------------------------------------------
# 10000 *
# Mean Squared Error: 13.486594282088824
# R-squared: 0.8925860841336213
-------------------------------------------------------------------------
# 2000
# Mean Squared Error: 17.67241745870815
# R-squared: 0.856391269077879
-------------------------------------------------------------------------
# 1000
```

```
# Mean Squared Error: 17.760826942912427
# R-squared: 0.8556728402688198
--------------------------------------------------------------------------------
# 200
# Mean Squared Error: 17.974536491107553
# R-squared: 0.8539362042328077
--------------------------------------------------------------------------------
# 125 - 25%
# Mean Squared Error: 18.41418326057995
# R-squared: 0.8503635682442418
--------------------------------------------------------------------------------
# 150 - 25%
# Mean Squared Error: 18.158129222254857
# R-squared: 0.8524443019965577
--------------------------------------------------------------------------------
# 100 - 25%
# Mean Squared Error: 18.416525185418656
# R-squared: 0.8503445374096212
```

In [ ]:

```
1
```