In [ ]:

```
# Importing the libraries.

import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.metrics import r2_score
from sklearn.impute import SimpleImputer
```

In [ ]:

```
# Importing Dataset as dataframe.

df = pd.read_csv("dataset.csv")
df.head()
```

Out[ ]:

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	C
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	

In [ ]:

```
# Exploring data.

df.describe()
```

Out[ ]:

	Unnamed: 0	Year	Kilometers_Driven	Mileage	Engine	
count	6019.000000	6019.000000	6.019000e+03	6017.000000	5983.000000	5977.00
mean	3009.000000	2013.358199	5.873838e+04	18.134961	1621.276450	5.2
std	1737.679967	3.269742	9.126884e+04	4.582289	601.355233	0.80
min	0.000000	1998.000000	1.710000e+02	0.000000	72.000000	0.00
25%	1504.500000	2011.000000	3.400000e+04	15.170000	1198.000000	5.00
50%	3009.000000	2014.000000	5.300000e+04	18.150000	1493.000000	5.00
75%	4513.500000	2016.000000	7.300000e+04	21.100000	1984.000000	5.00
max	6018.000000	2019.000000	6.500000e+06	33.540000	5998.000000	10.00

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         6019 non-null   int64  
 1   Name               6019 non-null   object 
 2   Location           6019 non-null   object 
 3   Year               6019 non-null   int64  
 4   Kilometers_Driven  6019 non-null   int64  
 5   Fuel_Type          6019 non-null   object 
 6   Transmission       6019 non-null   object 
 7   Owner_Type         6019 non-null   object 
 8   Mileage            6017 non-null   float64
 9   Engine             5983 non-null   float64
 10  Power              5983 non-null   object 
 11  Seats              5977 non-null   float64
 12  New_Price          824 non-null    object 
 13  Price              6019 non-null   float64
dtypes: float64(4), int64(3), object(7)
memory usage: 658.5+ KB
```

In [ ]:

```
# Finding missing values

df.isnull().sum()
```

Out[ ]:

```
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 [ ]:

```
# Dropping rows with null values

df.dropna(inplace=True)
```

In [ ]:

```
# Finding missing values

df.isnull().sum()
```

Out[ ]:

```
Unnamed: 0           0
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
New_Price            0
Price                0
dtype: int64
```

In [ ]:

```
# Exploring parameters

print(df.Fuel_Type.value_counts())
print(df.Transmission.value_counts())
print(df.Owner_Type.value_counts())
```

```
Fuel_Type
Diesel    443
Petrol    371
CNG         9
Name: count, dtype: int64
Transmission
Manual       512
Automatic    311
Name: count, dtype: int64
Owner_Type
First     765
Second     55
Third       3
Name: count, dtype: int64
```

In [ ]:

```
# Encoding dataframe

# For fuel types.
df.replace({'Fuel_Type': {'Petrol': 0, 'Diesel': 1, 'CNG': 2, 'LPG': 3, 'Ele

# For Transmission type
df.replace({'Transmission': {'Manual': 0, 'Automatic': 1}}, inplace=True)

# For Owner Type
df.replace({'Owner_Type': {'First': 0, 'Second': 1, 'Third': 2, 'Fourth & Ab
```

```
/tmp/ipykernel_358/133526643.py:4: FutureWarning: Downcasting behavior in `r
eplace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in
to the future behavior, set `pd.set_option('future.no_silent_downcasting', T
rue)`
  df.replace({'Fuel_Type': {'Petrol': 0, 'Diesel': 1, 'CNG': 2, 'LPG': 3, 'E
lectric': 4}}, inplace=True)
/tmp/ipykernel_358/133526643.py:7: FutureWarning: Downcasting behavior in `r
eplace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in
to the future behavior, set `pd.set_option('future.no_silent_downcasting', T
rue)`
  df.replace({'Transmission': {'Manual': 0, 'Automatic': 1}}, inplace=True)
/tmp/ipykernel_358/133526643.py:10: FutureWarning: Downcasting behavior in `
replace` is deprecated and will be removed in a future version. To retain th
e old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-i
n to the future behavior, set `pd.set_option('future.no_silent_downcasting',
True)`
  df.replace({'Owner_Type': {'First': 0, 'Second': 1, 'Third': 2, 'Fourth &
Above': 3}}, inplace=True)
```

In [ ]:

```
# Updated dataset

df.head()
```

Out[ ]:

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
2	2	Honda Jazz V	Chennai	2011	46000	0	0	
7	7	Toyota Innova Crysta 2.8 GX AT 8S	Mumbai	2016	36000	1	1	
10	10	Maruti Ciaz Zeta	Kochi	2018	25692	0	0	
15	15	Mitsubishi Pajero Sport 4X4	Delhi	2014	110000	1	0	
20	20	BMW 3 Series 320d	Kochi	2014	32982	1	1	

In [ ]:

```
# Splitting the dataframe into data and target.

X = df.drop(['Price', 'Name', 'New_Price', 'Location'], axis=1)
Y = df['Price']
```

In [ ]:

```
X.head()
```

Out[ ]:

	Unnamed: 0	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	E
2	2	2011	46000	0	0	0	18.20	1
7	7	2016	36000	1	1	0	11.36	2
10	10	2018	25692	0	0	0	21.56	1
15	15	2014	110000	1	0	0	13.50	2
20	20	2014	32982	1	1	0	22.69	1

In [ ]:

```
Y.head()
```

Out[ ]:

```
2      4.50
7     17.50
10     9.95
15    15.00
20    18.55
Name: Price, dtype: float64
```

In [ ]:

```
# Splitting data into training and testing sets.

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, r
```

In [ ]:

```
# Fitting the LinearRegression model

model = LinearRegression()
model.fit(X_train, Y_train)
```

Out[ ]:

```
▾ LinearRegression ⓘ ?
LinearRegression()
```

In [ ]:

```
# Implementing Prediction

prediction = model.predict(X_train)
```

In [ ]:

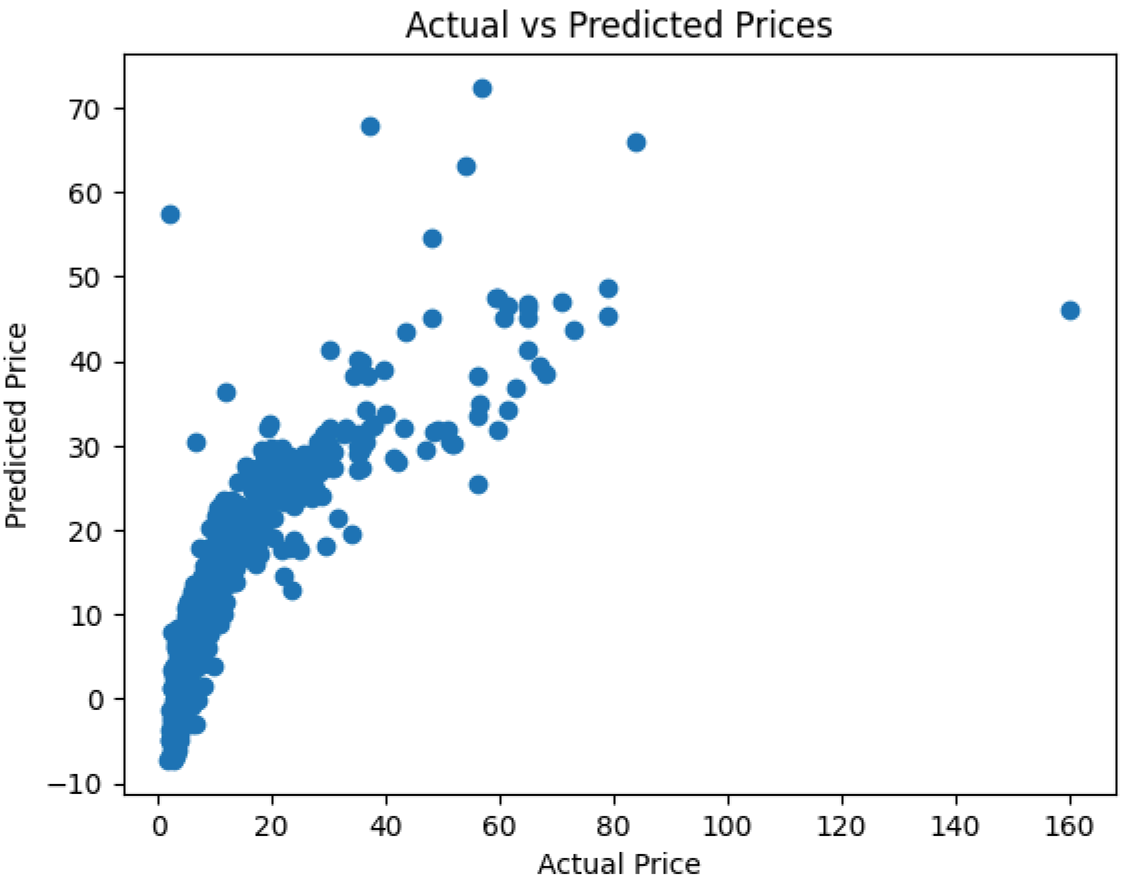
```
# Calculating model's accuracy

error_score = r2_score(Y_train, prediction)

error_score
```

Out[ ]:

```
0.6863803349632541
```

In [ ]:

```
# Representing accuracy visually using scatter plot fro training data

plt.scatter(Y_train, prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted Prices")
plt.show()
```

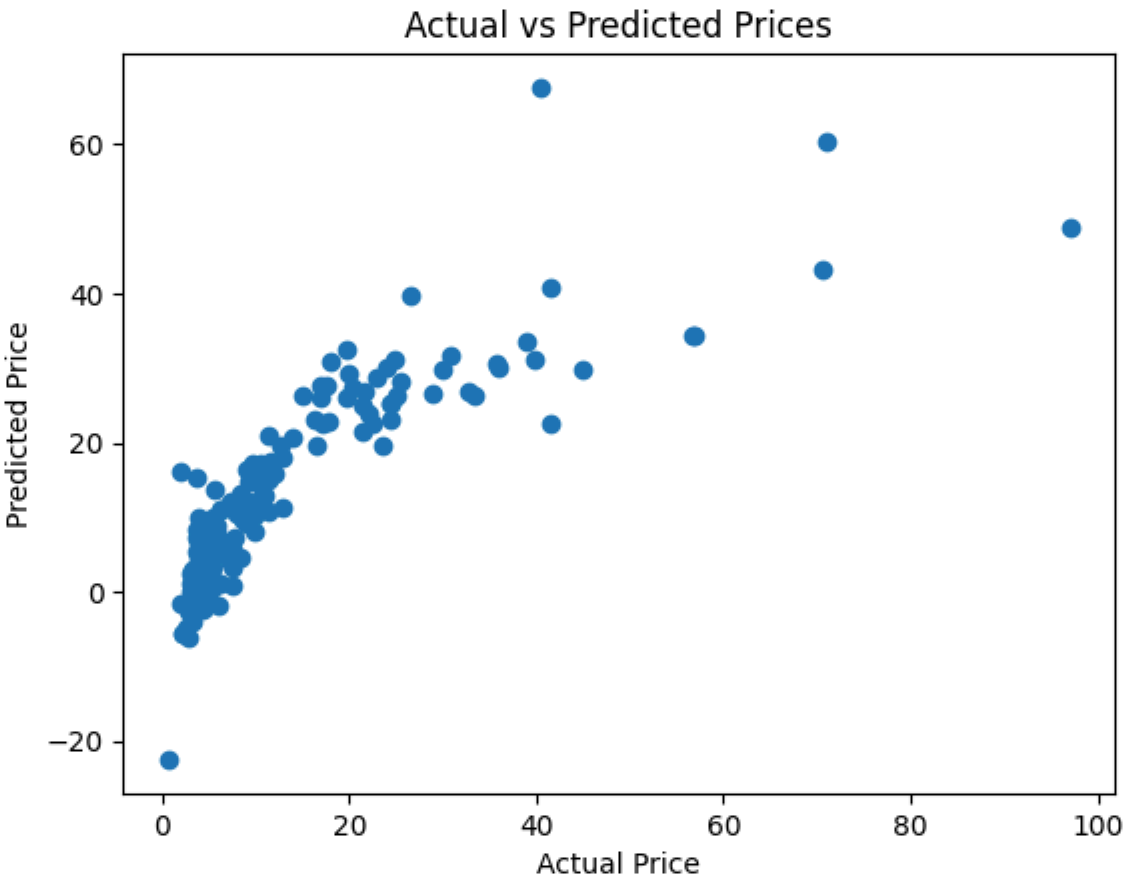


In [ ]:
```
# Predicting for testing dat

test_pred = model.predict(X_test)
```

In [ ]:
```
# Calculating accuracy for model using testing data

error_score = r2_score(Y_test, test_pred)

error_score
```

Out[ ]:
```
0.6997672112263285
```

In [ ]:
```
# Representing accuracy visually using scatter plot fro testing data

plt.scatter(Y_test, test_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted Prices")
plt.show()
```



In [ ]:

```
# Importing new data

nd = pd.read_csv("new_data.csv")
nd.head()
```

Out[ ]:

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
0	0	Honda City (V) 2015	Shahjahanpur	2015	89000	Diesel	Manual	

In [ ]:

```
# Encoding new data

# For fuel types.
nd.replace({'Fuel_Type': {'Petrol': 0, 'Diesel': 1, 'CNG': 2, 'LPG': 3, 'Ele

# For Transmission type
nd.replace({'Transmission': {'Manual': 0, 'Automatic': 1}}, inplace=True)

# For Owner Type
nd.replace({'Owner_Type': {'First': 0, 'Second': 1, 'Third': 2, 'Fourth & Ab

nd
```

```
/tmp/ipykernel_358/1013661178.py:4: FutureWarning: Downcasting behavior in `
replace` is deprecated and will be removed in a future version. To retain th
e old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-i
n to the future behavior, set `pd.set_option('future.no_silent_downcasting',
True)`
  nd.replace({'Fuel_Type': {'Petrol': 0, 'Diesel': 1, 'CNG': 2, 'LPG': 3, 'E
lectric': 4}}, inplace=True)
/tmp/ipykernel_358/1013661178.py:7: FutureWarning: Downcasting behavior in `
replace` is deprecated and will be removed in a future version. To retain th
e old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-i
n to the future behavior, set `pd.set_option('future.no_silent_downcasting',
True)`
  nd.replace({'Transmission': {'Manual': 0, 'Automatic': 1}}, inplace=True)
/tmp/ipykernel_358/1013661178.py:10: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain t
he old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-
in to the future behavior, set `pd.set_option('future.no_silent_downcastin
g', True)`
  nd.replace({'Owner_Type': {'First': 0, 'Second': 1, 'Third': 2, 'Fourth &
Above': 3}}, inplace=True)
```

Out[ ]:

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
0	0	Honda City (V) 2015	Shahjahanpur	2015	89000	1	0	

In [ ]:

```
# Pre-processing new data

pred_data = nd.drop(['Name', 'Location', 'Price'], axis = 1)
pred_data
```

Out[ ]:

	Unnamed: 0	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	En
0	0	2015	89000	1	0	0	21	

In [ ]:

```
# Predicting based on new data

new_prediciton = model.predict(pred_data)
new_prediciton
```

Out[ ]:

```
array([10.82642432])
```

LASSO REGRESSION MODEL

In [ ]:

```
# Fitting lasso model

lasso_model = Lasso()
lasso_model.fit(X_train, Y_train)
```

Out[ ]:

```
▾ Lasso ⓘ ?
Lasso()
```

In [ ]:

```
# Predicting via lasso model

lasso_pred = lasso_model.predict(X_train)
```

In [ ]:

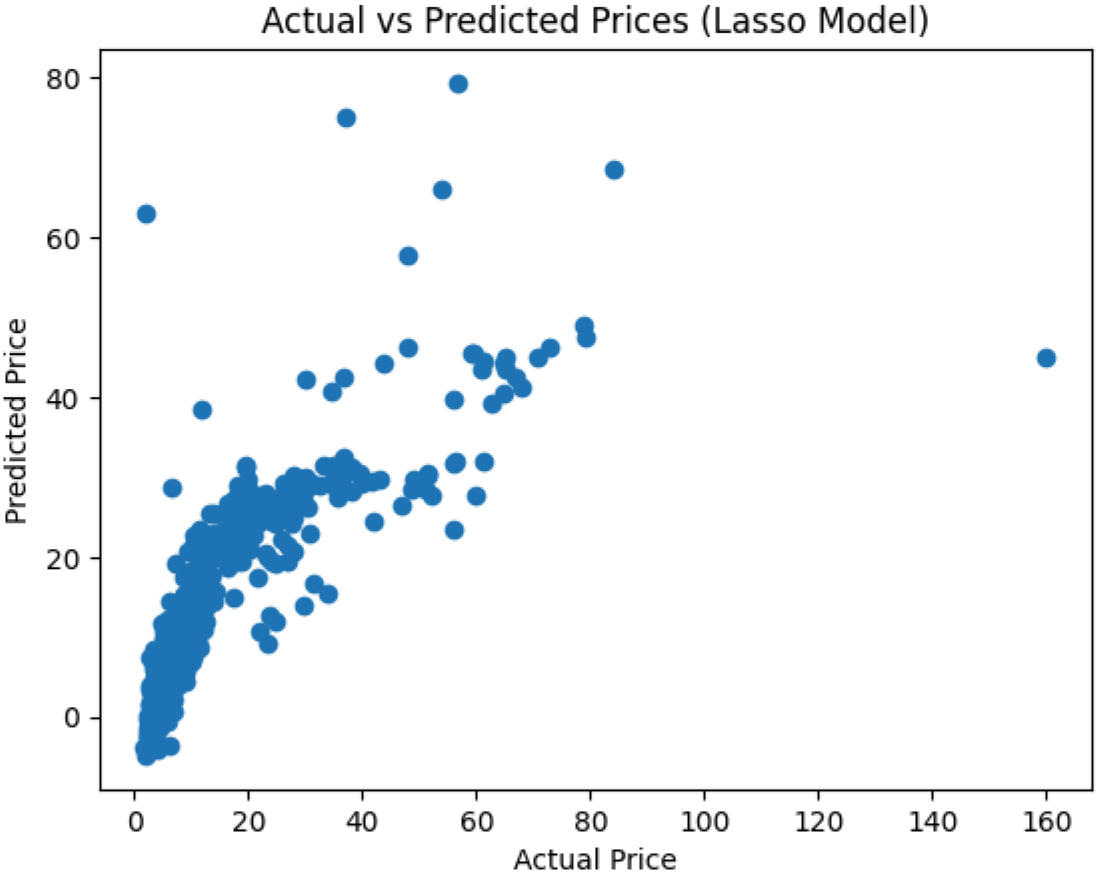
```
# Calculating lasso model's accuracy

error_score = r2_score(Y_train, lasso_pred)

error_score
```

Out[ ]:

```
0.6646931120946568
```

In [ ]:

```
# Representing lasso model's accuracy visually

plt.scatter(Y_train, lasso_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted Prices (Lasso Model)")
plt.show()
```


Actual vs Predicted Prices (Lasso Model)
Predicted Price
Actual Price