In [6]:
```python
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
from sklearn.linear_model import Lasso
from sklearn import metrics
```

Data Collection And Processing

In [13]:
```python
car_data = pd.read_csv(r"C:\Users\Manas Ranjan Kar\Downloads\archive (2)\car data.csv")
car_data.head()
```

Out[13]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

In [15]:
```python
car_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_Type    301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

In [18]:
```python
car_data.isnull().sum()
```

Out[18]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

In [19]:
```python
### Distribution of Categorical data

print(car_data.Fuel_Type.value_counts())
print(car_data.Seller_Type.value_counts())
print(car_data.Transmission.value_counts())
```

```
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
Dealer        195
Individual    106
Name: Seller_Type, dtype: int64
Manual       261
Automatic     40
Name: Transmission, dtype: int64
```

Encoding the categorical data

In [22]:
```python
car_data.replace({"Fuel_Type":{"Petrol":0,"Diesel":1,"CNG":2}},inplace=True)
car_data.replace({"Seller_Type":{"Dealer":0,"Individual":1}},inplace=True)
car_data.replace({"Transmission":{"Manual":0,"Automatic":1}},inplace=True)
```

In [24]:
```python
car_data.head(4)
```

Out[24]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	0	0	0	0
1	sx4	2013	4.75	9.54	43000	1	0	0	0
2	ciaz	2017	7.25	9.85	6900	0	0	0	0
3	wagon r	2011	2.85	4.15	5200	0	0	0	0

Splitting the data into train and test data

In [25]:
```python
x = car_data.drop(["Car_Name","Selling_Price"],axis=1)
y = car_data["Selling_Price"]
```

In [27]:
```python
x
```

Out[27]:

	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	2014	5.59	27000	0	0	0	0
1	2013	9.54	43000	1	0	0	0
2	2017	9.85	6900	0	0	0	0
3	2011	4.15	5200	0	0	0	0
4	2014	6.87	42450	1	0	0	0
...	...	...	...	...	...	...	...
296	2016	11.60	33988	1	0	0	0
297	2015	5.90	60000	0	0	0	0
298	2009	11.00	87934	0	0	0	0
299	2017	12.50	9000	1	0	0	0
300	2016	5.90	5464	0	0	0	0

301 rows × 7 columns

In [38]:
```python
X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size = 0.1, random_state = 2)
```

Model Training

Linear Regression

In [49]:
```python
lin_reg_model = LinearRegression()
```

In [40]:
```python
lin_reg_model.fit(X_train,Y_train)
```

Out[40]:
```
LinearRegression()
```

Model Evaluation

In [36]:
```python
training_data_prediction = lin_reg_model.predict(x_train)
```

In [41]:
```python
# R squared Error
error_score = metrics.r2_score(Y_train,training_data_prediction)
print("R squared Error :" ,error_score )
```

```
R squared Error : 0.8799451660493705
```

Visualize the actual and prdicted prices

In [43]:
```python
plt.scatter(Y_train,training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices ")
plt.show()
```

In [44]:
```python
test_data_prediction = lin_reg_model.predict(x_test)
```

In [46]:
```python
error_score = metrics.r2_score(Y_test,test_data_prediction)
print("R squared Error :" ,error_score )
```

```
R squared Error : 0.8365766715024661
```

In [47]:
```python
plt.scatter(Y_test,test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices ")
plt.show()
```

Lasso Regression

In [50]:
```python
lass_reg_model = Lasso()
```

In [51]:
```python
lass_reg_model.fit(X_train,Y_train)
```

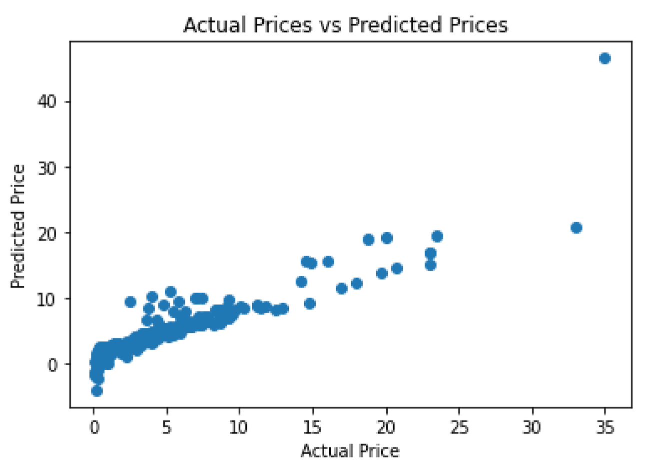
Out[51]:
```
Lasso()
```

In [53]:
```python
training_data_prediction = lass_reg_model.predict(X_train)
```

In [54]:
```python
error_score = metrics.r2_score(Y_train,training_data_prediction)
print("R squared Error :" ,error_score )
```

```
R squared Error : 0.8427856123435793
```

In [55]:
```python
plt.scatter(Y_train,training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices ")
plt.show()
```

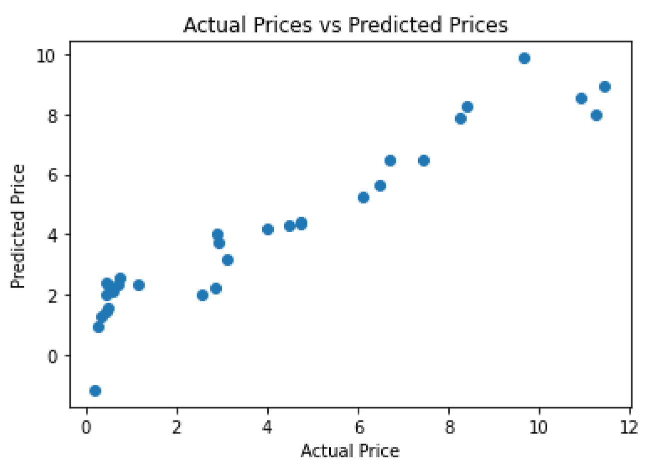


In [56]:
```python
test_data_prediction = lass_reg_model.predict(x_test)
```

In [57]:
```python
error_score = metrics.r2_score(Y_test,test_data_prediction)
print("R squared Error :" ,error_score )
```

```
R squared Error : 0.8709167941173195
```

In [58]:
```python
plt.scatter(Y_test,test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices ")
plt.show()
```



In [ ]: