Importing the necessary libraries

In [1]:
```
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 [2]:
```
car_dataset = pd.read_csv("C:\\Users\\Lenovo\\Downloads\\car data.csv")
```

In [3]:
```
car_dataset.head()
```

Out[3]:

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

In [4]:
```
car_dataset.shape
```

Out[4]:
```
(301, 9)
```

In [5]:
```
car_dataset.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   Driven_kms     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Selling_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 [6]:
```
car_dataset.isnull().sum()
```

Out[6]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Driven_kms       0
Fuel_Type        0
Selling_type     0
Transmission     0
Owner            0
dtype: int64
```

In [7]:
```
#checking the distribution of categorical data
print(car_dataset.Fuel_Type.value_counts())
print(car_dataset.Selling_type.value_counts())
print(car_dataset.Transmission.value_counts())
```

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

Encoding the categorical data

In [8]:
```
car_dataset.replace({'Fuel_Type':{'Petrol':0, 'Diesel':1, 'CNG':2}}, inplace=True)
car_dataset.replace({'Selling_type':{'Dealer':0, 'Individual':1}}, inplace=True)
car_dataset.replace({'Fuel_Type':{'Manual':0, 'Automatic':1}}, inplace=True)
```

In [9]:
```
car_dataset.head()
```

Out[9]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmissior
0	ritz	2014	3.35	5.59	27000	0	0	Manua
1	sx4	2013	4.75	9.54	43000	1	0	Manua
2	ciaz	2017	7.25	9.85	6900	0	0	Manua
3	wagon r	2011	2.85	4.15	5200	0	0	Manua
4	swift	2014	4.60	6.87	42450	1	0	Manua

Splitting the data and target

In [10]:
```
X = car_dataset.drop(['Car_Name', 'Selling_Price', 'Transmission'], axis=1)
Y = car_dataset['Selling_Price']
```

In [11]:
```
print(X)
```

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

[301 rows x 6 columns]
```

Splitting training and test data

In [12]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1, random_sta
```

Model Training

1. Linear Regression

In [13]:
```
lin_reg_model = LinearRegression()
```

In [14]:
```
lin_reg_model.fit(X_train,Y_train)
```

Out[14]:
```
LinearRegression()
```

Model Evaluation

In [15]:
```
training_data_prediction = lin_reg_model.predict(X_train)
```

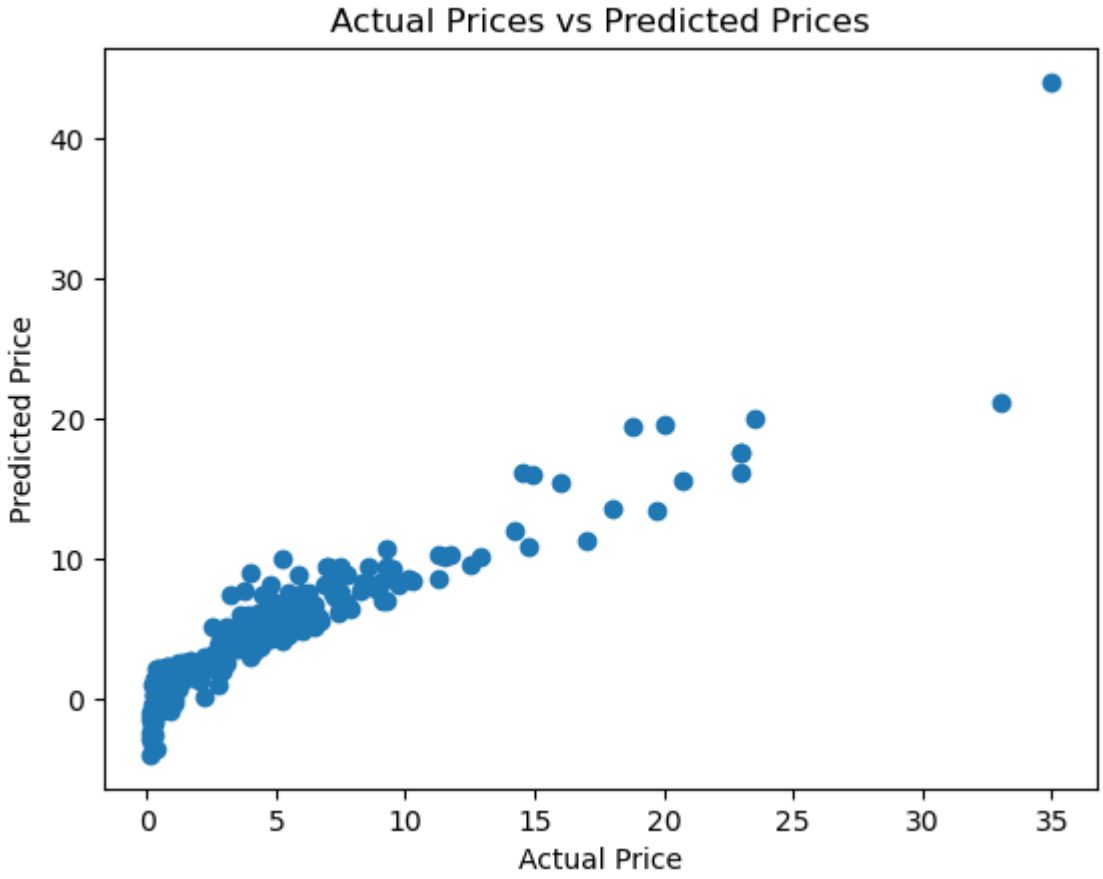
In [16]:
```
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R Squared Error : ", error_score)
```

```
R Squared Error :  0.8716571597791489
```

Visualize the actual prices and predicted prices

In [17]:
```
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 [18]:
```
test_data_prediction = lin_reg_model.predict(X_test)
```

In [19]:
```
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R Squared Error : ", error_score)
```

```
R Squared Error :  0.8340577535179488
```

In [20]:
```
plt.scatter(Y_test, test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices")
plt.show()
```

2. Lasso Regression

In [21]:
```
lass_reg_model = Lasso()
```

In [22]:
```
lass_reg_model.fit(X_train,Y_train)
```

Out[22]:
```
Lasso
Lasso()
```

Model Evaluation

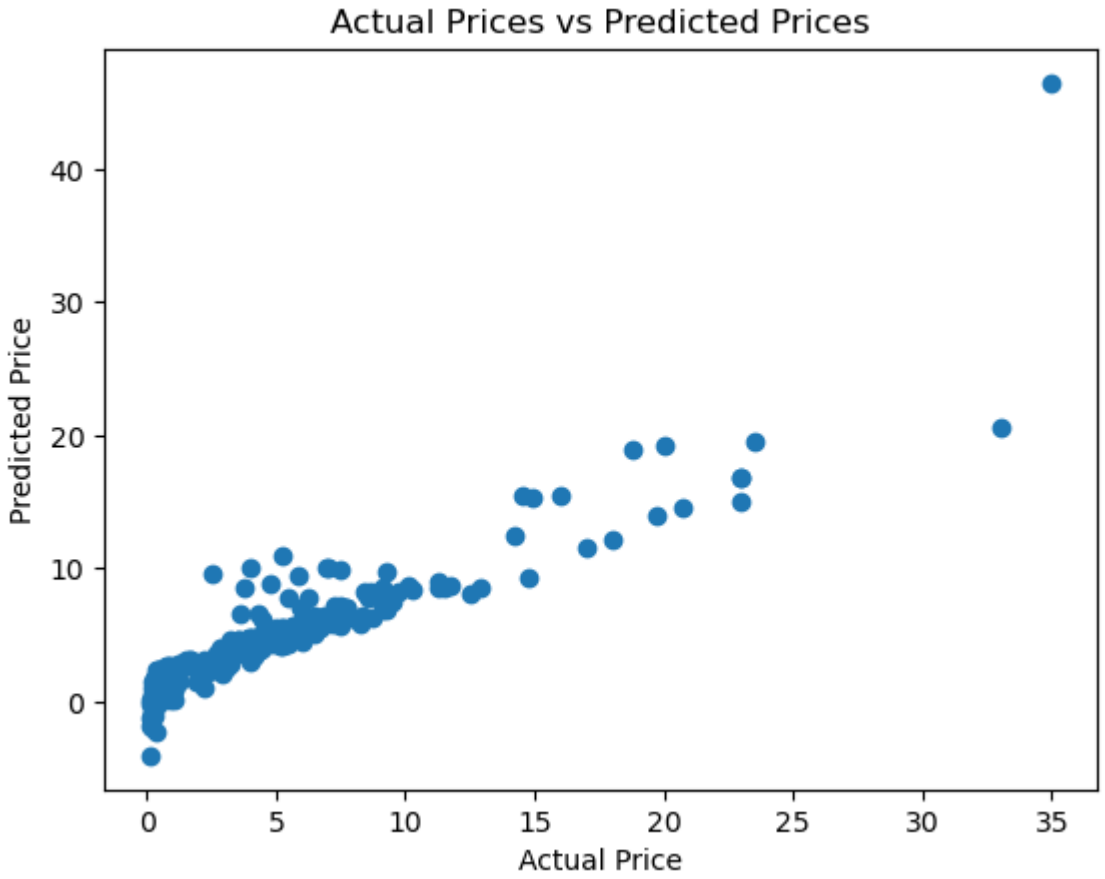
In [23]:
```
training_data_prediction = lass_reg_model.predict(X_train)
```

In [24]:
```
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R Squared Error : ", error_score)
```

```
R Squared Error :  0.8424480718240741
```

Visualize the actual prices and predicted prices

In [25]:
```
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 [26]:
```
test_data_prediction = lass_reg_model.predict(X_test)
```

In [27]:
```
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R Squared Error : ", error_score)
```

```
R Squared Error :  0.8709763132343402
```

In [28]:
```
plt.scatter(Y_test, test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices")
plt.show()
```


Actual Prices vs Predicted Prices
Predicted Price
Actual Price
0
2
4
6
8
10
12