Car Price Prediction

(1) Import Python 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
```

(2) Loading the Data Set

In [2]:
```
path = r'D:\IITG\portfolio_finance\car_price\car_details.csv'
```

In [3]:
```
car_dataset = pd.read_csv(path)
car_dataset.head()
```

Out[3]:

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

In [4]:
```
car_dataset.tail()
```

Out[4]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_1
296	city	2016	9.50	11.6	33988	Diesel	
297	brio	2015	4.00	5.9	60000	Petrol	
298	city	2009	3.35	11.0	87934	Petrol	
299	city	2017	11.50	12.5	9000	Diesel	
300	brio	2016	5.30	5.9	5464	Petrol	

(3) Exploring the Data Set

In [5]:
```
car_dataset.shape
```

Out[5]:
```
(301, 9)
```

In [6]:
```
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   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 [7]:
```
car_dataset.isnull().sum()
```

Out[7]:
```
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 [8]:
```
car_dataset.Fuel_Type.value_counts()
```

Out[8]:
```
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
```

In [9]:
```
car_dataset.Seller_Type.value_counts()
```

Out[9]:
```
Dealer        195
Individual    106
Name: Seller_Type, dtype: int64
```

In [10]:
```
car_dataset.Transmission.value_counts()
```

Out[10]:
```
Manual       261
Automatic     40
Name: Transmission, dtype: int64
```

(4) Encoding the Data

In [11]:
```
car_dataset.replace({'Fuel_Type':{'Petrol':0,'Diesel':1,'CNG':2}},inplace=Tr
car_dataset.replace({'Seller_Type':{'Dealer':0,'Individual':1}},inplace=True
car_dataset.replace({'Transmission':{'Manual':0,'Automatic':1}},inplace=True
```

In [12]:
```
car_dataset.head()
```

Out[12]:

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

(5) Building the Linear Regression Model

Split the Data

In [13]:
```
X = car_dataset.drop(['Car_Name','Selling_Price'],axis=1)
Y = car_dataset['Selling_Price']
```

In [14]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1, r
```

Create and Fit the Model

In [15]:
```
lin_reg_model = LinearRegression()
lin_reg_model.fit(X_train,Y_train)
```

Out[15]:
```
LinearRegression()
```

(6) Evaluating the Linear Regression Model

In [16]:
```
training_data_prediction = lin_reg_model.predict(X_train)
training_data_prediction[:5]
```

Out[16]:
```
array([ 3.73088505,  5.60702081,  7.79779356, -1.88374756,  6.71614572])
```

In [19]:
```
test_data_prediction = lin_reg_model.predict(X_test)
test_data_prediction[:5]
```

Out[19]:
```
array([10.32892855,  0.77165673,  4.26482324,  4.78985002,  9.88701568])
```

R-squared Error

In [17]:
```
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R squared Error : ", error_score)
```

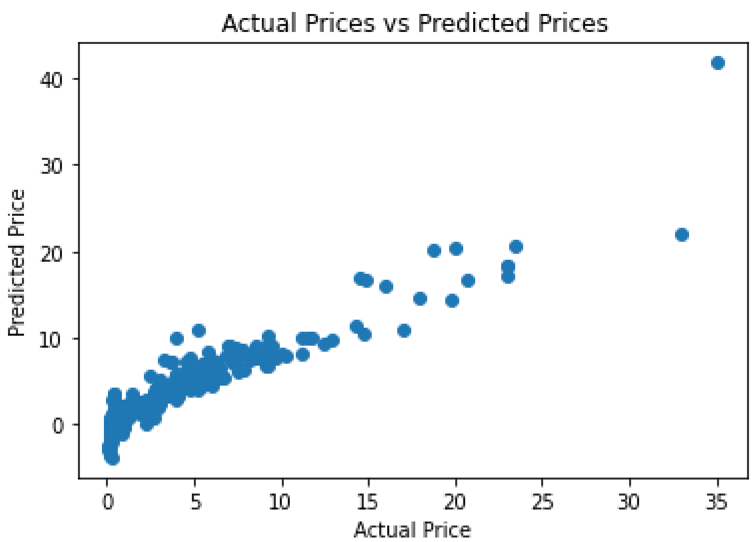
```
R squared Error :  0.8799451660493701
```

In [20]:
```
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared Error : ", error_score)
```

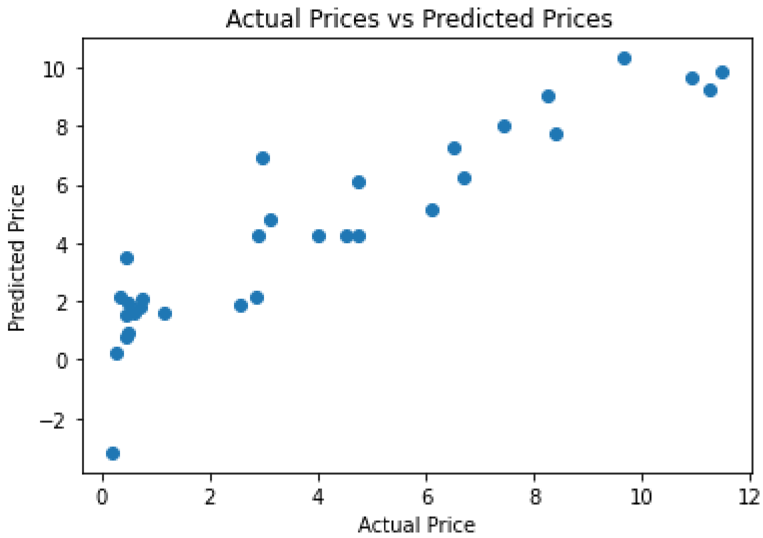
```
R squared Error :  0.836576671502687
```

(7) Visualizing the Linear Regression Model

In [18]:
```
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 [21]:
```
plt.scatter(Y_test, test_data_prediction)

plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")

plt.show()
```



(8) Building the Lasso Regression Model

Create and Fit the Model

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

Out[22]: Lasso()

(9) Evaluating the Lasso Regression Model

In [23]:
```
training_data_prediction = lass_reg_model.predict(X_train)
training_data_prediction[:5]
```

Out[23]: array([3.56679076, 5.60257564, 8.28781371, -0.83081431, 5.2753988])

In [24]:
```
test_data_prediction = lass_reg_model.predict(X_test)
test_data_prediction[:5]
```

Out[24]: array([9.87888122, 1.42396266, 4.33267834, 3.17313445, 8.95590579])

R-squared Error

In [25]:
```
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R squared Error : ", error_score)
```

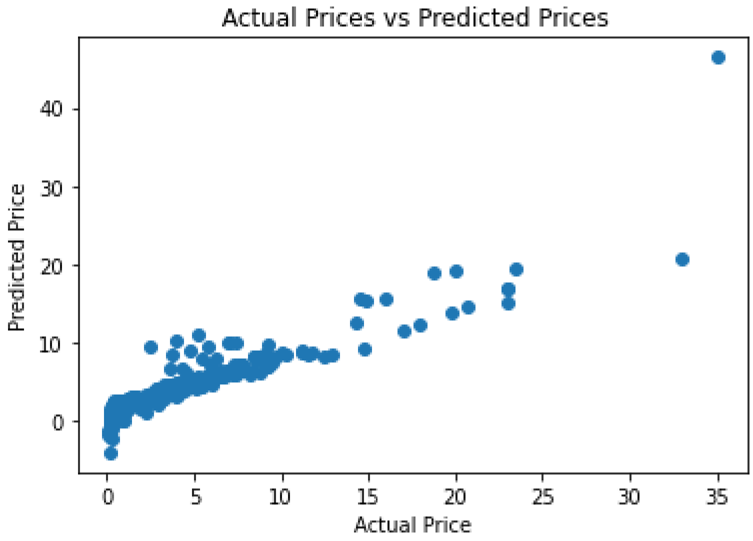
```
R squared Error :  0.8427856123435794
```

In [26]:
```
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.8709167941173195
```

(10) Visualizing the Lasso Regression Model

In [27]:
```
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 [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()
```