In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn import metrics
```

In [2]:
```
car_dataset = pd.read_csv('car data prediction.csv')
car_dataset.head()
```

Out[2]:

	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 [3]:
```
car_dataset.shape
```

Out[3]:
```
(301, 9)
```

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

Out[4]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

In [5]:
```
##### Finding Missing Values
car_dataset.isnull().sum()
```

Out[5]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
```

In [6]:
```
#### checking distribution of categorical values
print(car_dataset['Owner'].value_counts())

print(car_dataset['Seller_Type'].value_counts())

print(car_dataset['Transmission'].value_counts())

print(car_dataset['Fuel_Type'].value_counts())
```

```
0    290
1     10
3      1
Name: Owner, dtype: int64
Dealer        195
Individual    106
Name: Seller_Type, dtype: int64
Manual       261
Automatic     40
Name: Transmission, dtype: int64
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
```

In [7]:
```
##### Encoding the categorical data
```

In [8]:
```
### encoding fuel_type column
car_dataset.replace({'Fuel_Type':{'Petrol':0,'Diesel':1,'CNG':2}},inplace=True)


### encoding Seller_Type column
car_dataset.replace({'Seller_Type':{'Dealer':0,'Individual':1}},inplace=True)


### encoding Transmission column
car_dataset.replace({'Transmission':{'Manual':0,'Automatic':1}},inplace=True)
```

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

Out[9]:

	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
4	swift	2014	4.60	6.87	42450	1	0	0	0

In [10]:
```
##### Separating the data and labels

X = car_dataset.drop(['Car_Name','Selling_Price'],axis=1)

Y = car_dataset['Selling_Price']
```

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

```
     Year  Present_Price  Kms_Driven  Fuel_Type  Seller_Type  Transmission  \
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

     Owner
0        0
1        0
2        0
3        0
4        0
..     ...
296      0
297      0
298      0
299      0
300      0

[301 rows x 7 columns]
```

In [12]:
```
print(Y)
```

```
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       ...
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 301, dtype: float64
```

In [13]:
```
#### splitting Training and Testing Data

X_train, X_test, Y_train, Y_test = train_test_split(X,Y , test_size=0.1 ,random_state = 2)
```

In [14]:
```
### Model Training
## 1. Linear Regression

lin_reg_model = LinearRegression()
```

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

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

In [16]:
```
###### Model Evaluation
### Prediction on training data
train_data_prediction = lin_reg_model.predict(X_train)
```

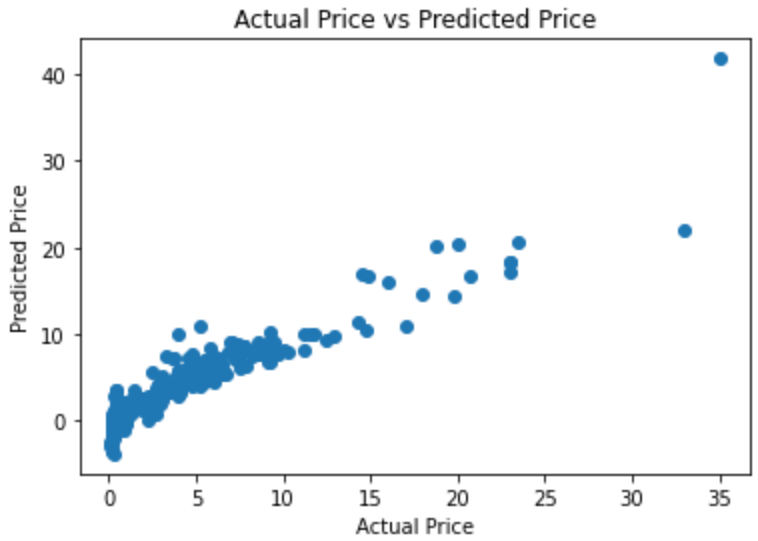
In [17]:
```
#### R - squared error

error_score= metrics.r2_score(Y_train,train_data_prediction)
print(error_score)
```

```
0.8799451660493705
```

In [18]:
```
#### Visualize the actual price and predicted prices

plt.scatter(Y_train,train_data_prediction)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual Price vs Predicted Price')
plt.show()
```



In [19]:
```
### Prediction on test data
test_data_prediction = lin_reg_model.predict(X_test)
```

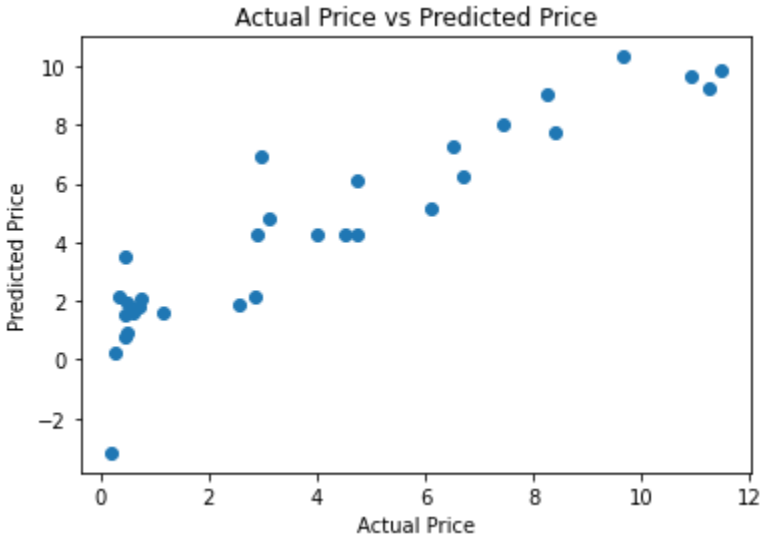
In [20]:
```
#### R - squared error

error_score= metrics.r2_score(Y_test,test_data_prediction)
print(error_score)
```

```
0.836576671502687
```

In [21]:
```
#### Visualize the actual price and predicted prices

plt.scatter(Y_test,test_data_prediction)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual Price vs Predicted Price')
plt.show()
```



In [22]:
```
#### loading the lasso regression model''''''
lass_reg_model = Lasso()
```

In [23]:
```
lass_reg_model.fit(X_train,Y_train)
```

Out[23]:
```
Lasso()
```

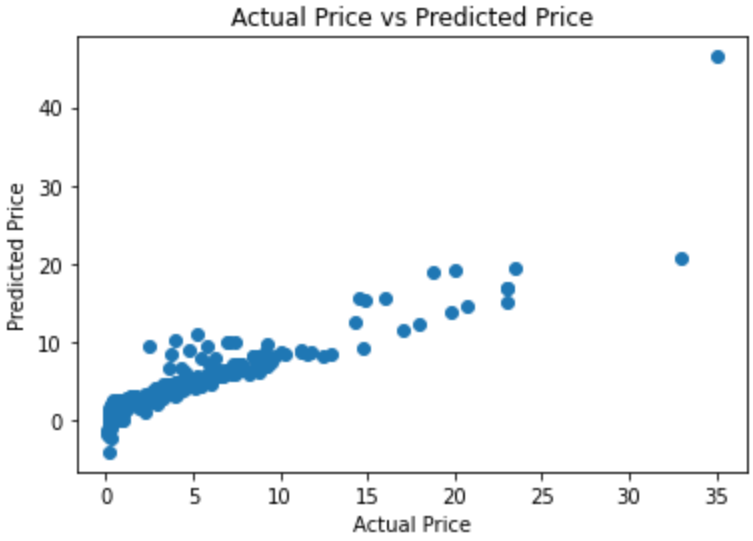
In [24]:
```
#### Model Evaluation
### Prediction on train data
train_data_prediction = lass_reg_model.predict(X_train)
```

In [25]:
```
#### R - squared error

error_score= metrics.r2_score(Y_train,train_data_prediction)
print(error_score)
```

```
0.8427856123435794
```

In [26]:
```
#### Visualize the actual price and predicted prices
plt.scatter(Y_train,train_data_prediction)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual Price vs Predicted Price')
plt.show()
```



In [27]:

```
### Prediction on test data
test_data_prediction = lass_reg_model.predict(X_test)
```

In [28]:

```
#### R - squared error

error_score= metrics.r2_score(Y_test,test_data_prediction)
print(error_score)
```

```
0.8709167941173195
```

In [29]:

```
#### Visualize the actual price and predicted prices
plt.scatter(Y_test,test_data_prediction)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual Price vs Predicted Price')
plt.show()
```