# CAR PRICE PREDICTION

## Importing dependencies

In [7]:
```
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
```

## The Dataset

In [8]:
```
car_dataset=pd.read_csv(r'C:\Users\HP\Downloads\archive (3)\car data.csv')
```

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 | 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 [10]:
```
car_dataset.tail()
```

Out[10]:

| | Car_Name | Year | Selling_Price | Present_Price | Kms_Driven | Fuel_Type | Seller_Type | Transmission | Owner |
|---|---|---|---|---|---|---|---|---|---|
| **296** | city | 2016 | 9.50 | 11.6 | 33988 | Diesel | Dealer | Manual | 0 |
| **297** | brio | 2015 | 4.00 | 5.9 | 60000 | Petrol | Dealer | Manual | 0 |
| **298** | city | 2009 | 3.35 | 11.0 | 87934 | Petrol | Dealer | Manual | 0 |
| **299** | city | 2017 | 11.50 | 12.5 | 9000 | Diesel | Dealer | Manual | 0 |
| **300** | brio | 2016 | 5.30 | 5.9 | 5464 | Petrol | Dealer | Manual | 0 |

In [11]:
```
car_dataset.shape
```

Out[11]:
```
(301, 9)
```

The given dataset has 301 rows and 9 columns.

## Checking for null values

In [13]:
```
car_dataset.isnull().sum()
```

Out[13]:
```
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
```

The given dataset has no null values.

## Checking the distribution of categorical data

In [17]:
```
print(car_dataset.Fuel_Type.value_counts())
print(car_dataset.Seller_Type.value_counts())
print(car_dataset.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
```

From the above data it can be observed that 239 cars are driven by petrol,60 cars are driven by diesel while only 2 cars are operated by CNG. Among the sold cars 195 cars are sold by dealers whereas 106 cars are sold by individuals. Among the cars 261 cars are manually operated while 40 cars are automatic.

# Encoding the categorical data

# Encoding Fuel_type column

In [18]:
```
car_dataset.replace({'Fuel_Type':{'Petrol':0,'Diesel':1,'CNG':2}},inplace=True)
car_dataset.replace({'Seller_Type':{'Dealer':0,'Individual':1}},inplace=True)
car_dataset.replace({'Transmission':{'Manual':0,'Automatic':1}},inplace=True)
```

In [19]:
```
car_dataset.head()
```

Out[19]:

| | 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 [22]:
```
car_dataset.tail()
```

Out[22]:

| | Car_Name | Year | Selling_Price | Present_Price | Kms_Driven | Fuel_Type | Seller_Type | Transmission | Owner |
|---|---|---|---|---|---|---|---|---|---|
| **296** | city | 2016 | 9.50 | 11.6 | 33988 | 1 | 0 | 0 | 0 |
| **297** | brio | 2015 | 4.00 | 5.9 | 60000 | 0 | 0 | 0 | 0 |
| **298** | city | 2009 | 3.35 | 11.0 | 87934 | 0 | 0 | 0 | 0 |
| **299** | city | 2017 | 11.50 | 12.5 | 9000 | 1 | 0 | 0 | 0 |
| **300** | brio | 2016 | 5.30 | 5.9 | 5464 | 0 | 0 | 0 | 0 |

# Splitting the data and Target

In [23]:
```
X=car_dataset.drop(columns=['Car_Name','Selling_Price'],axis=1)
Y=car_dataset['Selling_Price']
```

In [24]:
```
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 [25]:
```
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
```

## Splitting Training data and Test data

In [26]:
```
X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.1,random_state=2)
```

## Model Training

## 1. Linear Regression

In [27]:
```
lin_reg_model=LinearRegression()
```

In [28]:
```
lin_reg_model.fit(X_train,Y_train)
```

Out[28]:
```
▾ LinearRegression
LinearRegression()
```

## Model Evaluation on Training Data

In [29]:
```
training_data_prediction=lin_reg_model.predict(X_train)
```

## R squared error

In [30]:
```
error_score=metrics.r2_score(Y_train,training_data_prediction)
```

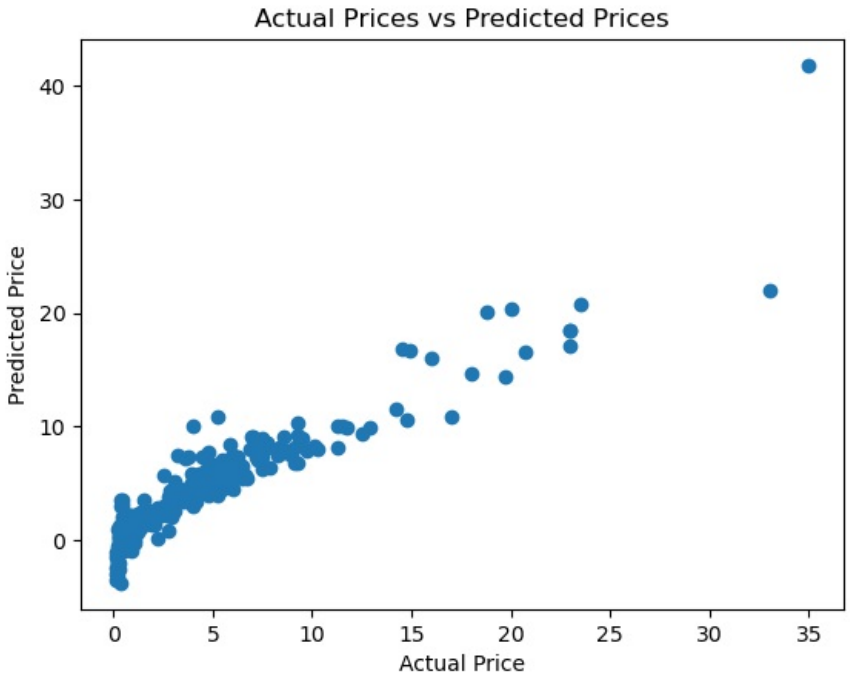
In [32]:
```
print('R squared error:',error_score)
```

```
R squared error: 0.87994516604937
```

## Visualize the actual prices and predicted prices

In [ ]:
```
plt.scatter(Y_train,training_data_prediction)
```

In [33]:
```
plt.scatter(Y_train,training_data_prediction)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual Prices vs Predicted Prices')
plt.show()
```




On the basis of the above scatter plot drawn according to this training dataset it can be observed that there is not that much difference between actual price and estimated price.

## Prediction on Training data

In [41]:
```
test_data_prediction=lin_reg_model.predict(X_test)
```

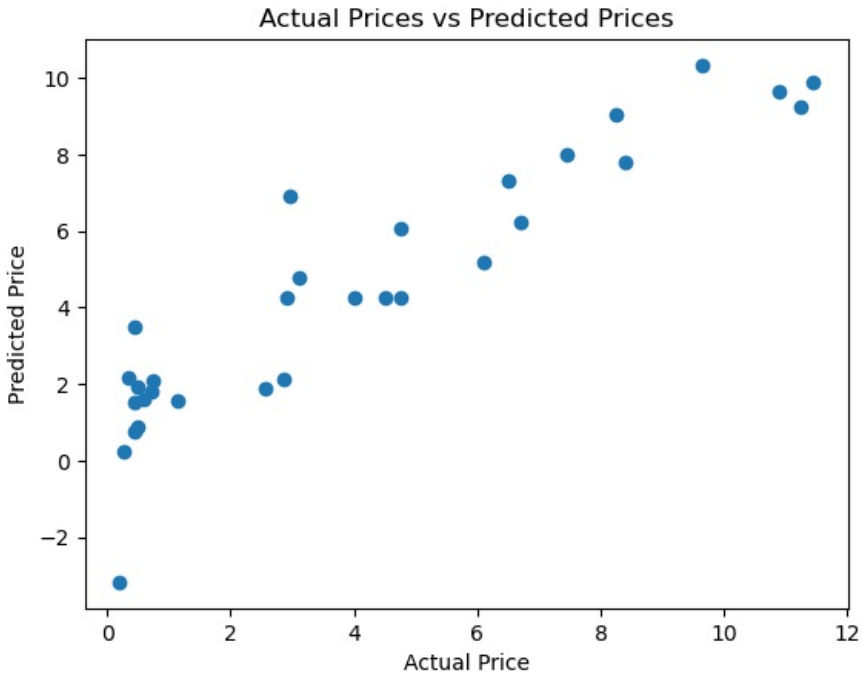
## R Squared Error :

In [45]:
```
error_score=metrics.r2_score(Y_test,test_data_prediction)
print('R squared error:',error_score)
```

```
R squared error: 0.8365766715025409
```

## Visualise the Actual Prices and Predicted Prices

In [47]:
```
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 [48]:
```
lass_reg_model=Lasso()
lass_reg_model.fit(X_train,Y_train)
```

Out[48]:
```
Lasso
Lasso()
```

## Model Evaluation of Training Data

In [49]:
```
training_data_prediction=lass_reg_model.predict(X_train)
```

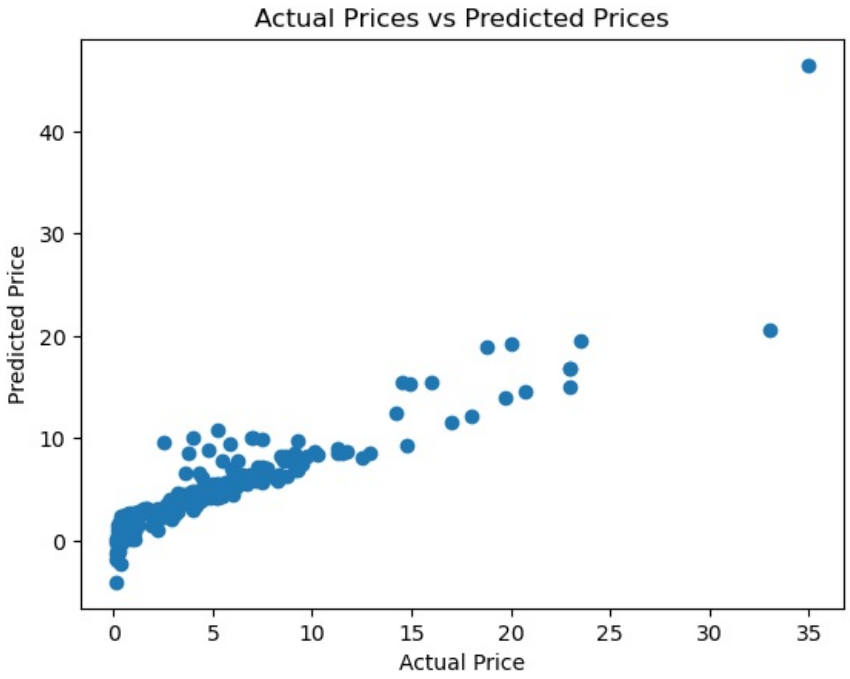
## R squared error

In [50]:
```
error_score=metrics.r2_score(Y_train,training_data_prediction)
print('R squared error:',error_score)
```

```
R squared error: 0.8427856123435794
```

## Visualize the actual prices and predicted prices

In [51]:
```
plt.scatter(Y_train,training_data_prediction)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual Prices vs Predicted Prices')
plt.show()
```




So in Lasso regression this is a very good fit.

# Prediction on training data

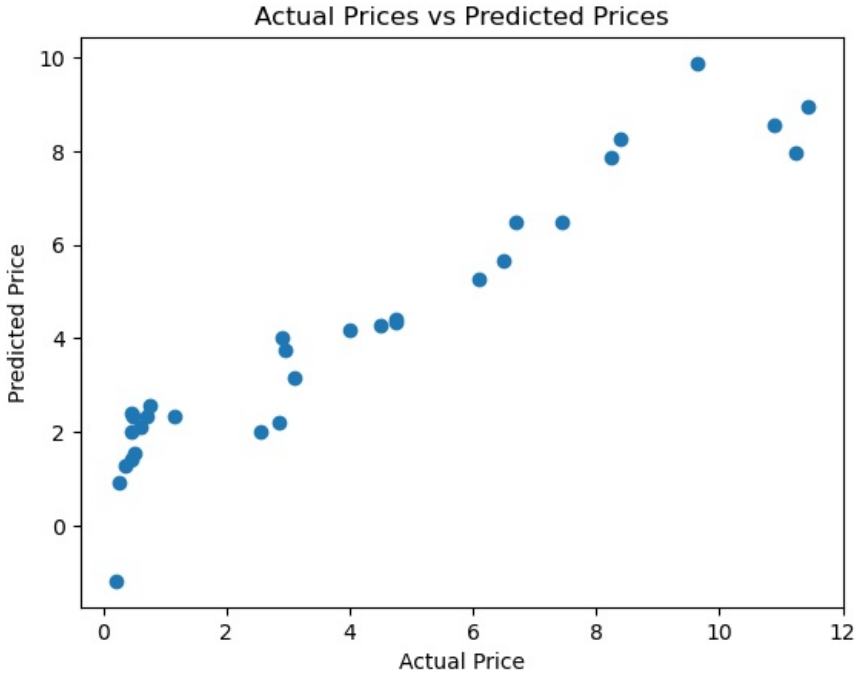
In [52]:
```
test_data_prediction=lass_reg_model.predict(X_test)
```

# R squared error

In [53]:
```
error_score=metrics.r2_score(Y_test,test_data_prediction)
print('R squared error:',error_score)
```

```
R squared error: 0.8709167941173195
```

In [54]:
```
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 [ ]: