Car Price Prediction

August 26, 2025

[30]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn import metrics
```

[2]:
```
car_data = pd.read_csv('car data.csv')
```

#First 5 rows of the dataframe

[4]:
```
car_data.head()
```

[4]:

	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

#Checking the number of rows and columns

[5]:
```
car_data.shape
```

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

[6]:
```
#getting some information about the dataframe
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
```

[7]:
```
#Checking Missing Value
car_data.isnull().sum()
```

[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
```

[9]:
```
#Checking the distribution of categorical data
print(car_data.Fuel_Type.value_counts())
print(car_data.Seller_Type.value_counts())
print (car_data.Transmission.value_counts())
```

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

#Encoding the category Data

[14]:
```
#encoding "Fuel_Type" Column
car_data.replace({'Fuel_Type':{'Petrol':0,'Diesel':1,'CNG':2}},inplace=True)

#encoding "Seller_Type" Column
car_data.replace({'Seller_Type':{'Dealer':0,'Individual':1}},inplace=True)

#encoding "Transmission" Column
car_data.replace({'Transmission':{'manual':0,'Automatic':1}},inplace=True)
```

[15]:
```
car_data.head()
```

[15]:
```
  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
```

#splitting the data

[19]:
```
x = car_data[['Year', 'Present_Price', 'Kms_Driven', 'Fuel_Type',␣
 ↪'Seller_Type', 'Transmission', 'Owner']]
y = car_data['Selling_Price']
```

[20]:
```
print(x)
```

```
     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 x 7 columns]
```

[21]:
```
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 and Test Data

[22]:
```
x_train,x_test, y_train, y_test = train_test_split(x, y, test_size =0.1,␣
 ↪random_state=2)
```

#Model_Training

#Loading the linear regression model

[33]:
```
lin_reg_model = LinearRegression()
```

[37]:
```
lin_reg_model.fit(x_train, y_train)
```

[37]:
```
LinearRegression()
```

#Model Evaluation

[39]:
```
#prediction on Training data
training_data_prediction = lin_reg_model.predict(x_train)
```

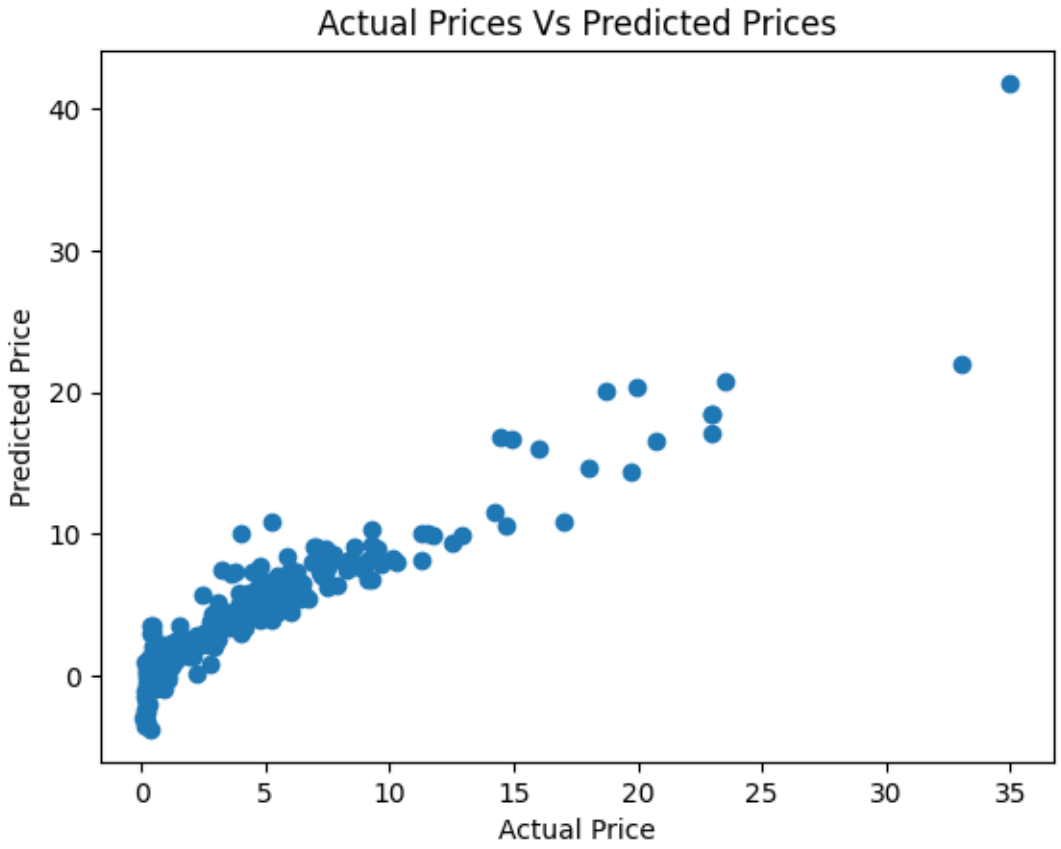
#Rsquared error

[41]:
```
error_score = metrics.r2_score(y_train, training_data_prediction)
print("Rsquared error : ", error_score)
```

```
Rsquared error :  0.8799451660493708
```

#Visualize the actual and Predicted prices:

[42]:
```
plt.scatter(y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices Vs Predicted Prices")
plt.show()
```



[43]:
```
#prediction on Training data
test_data_prediction = lin_reg_model.predict(x_test)
```

[45]:
```
error_score = metrics.r2_score(y_test, test_data_prediction)
print("Rsquared error : ", error_score)
```

```
Rsquared error :  0.8365766715026374
```

[46]:
```
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

[47]:
```
lass_reg_model = Lasso()
```

[48]:
```
lass_reg_model.fit(x_train, y_train)
```

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

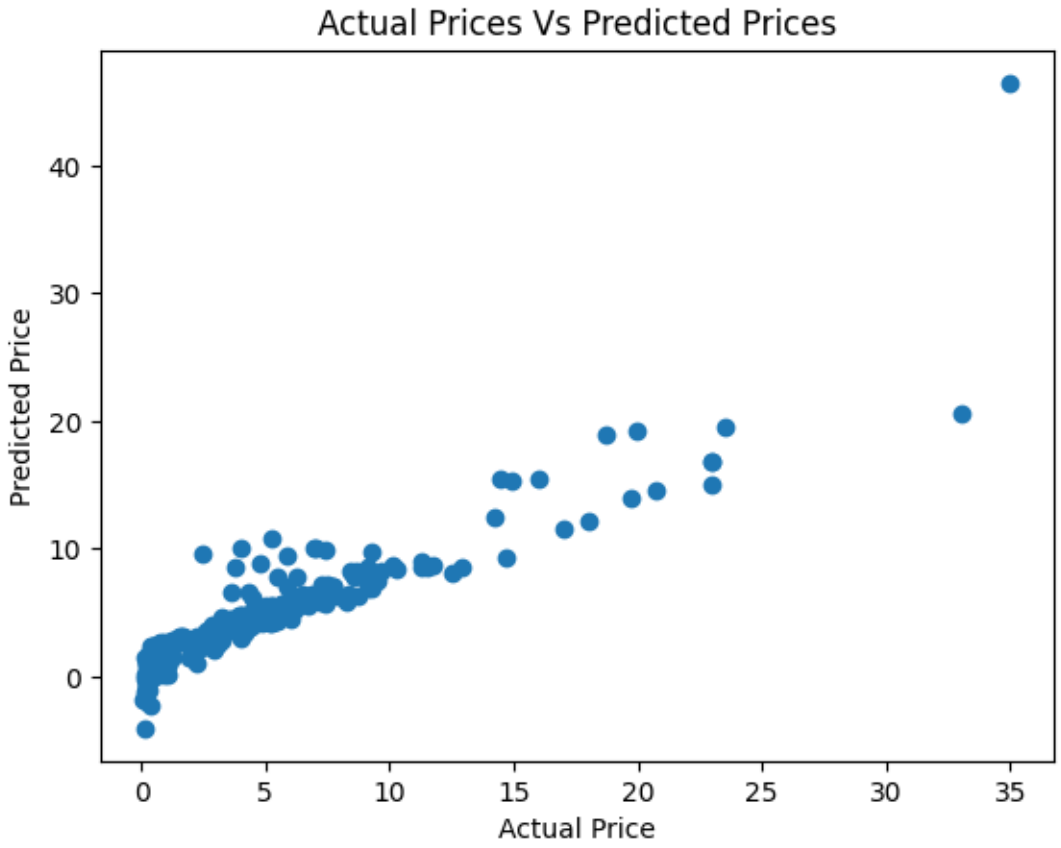
[49]:
```
#prediction on Training data
training_data_prediction = lass_reg_model.predict(x_train)
```

[50]:
```
error_score = metrics.r2_score(y_train, training_data_prediction)
print("Rsquared error : ", error_score)
```

```
Rsquared error :  0.8427856123435794
```

[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()
```



[52]:
```
#prediction on Training data
test_data_prediction = lass_reg_model.predict(x_test)
```

[53]:
```
error_score = metrics.r2_score(y_test, test_data_prediction)
print("Rsquared error : ", error_score)
```

```
Rsquared error :  0.8709167941173195
```

[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()
```


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

[ ]: