car-prices-prediction

May 24, 2024

[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 from csv file to pandas dataframe
car_dataset = pd.read_csv('car prices.csv')
```

[3]:
```
# inspecting the first 5 rows of the dataframe
car_dataset.head()
```

[3]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

[4]:
```
# checking the number of rows and columns
car_dataset.shape
```

[4]:
```
(4340, 8)
```

[5]:
```
# getting some information about the dataset
car_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
```

```
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   name            4340 non-null   object
 1   year            4340 non-null   int64
 2   selling_price   4340 non-null   int64
 3   km_driven       4340 non-null   int64
 4   fuel            4340 non-null   object
 5   seller_type     4340 non-null   object
 6   transmission    4340 non-null   object
 7   owner           4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

[6]:
```
# checking the number of missing values
car_dataset.isnull().sum()
```

[6]:
```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
dtype: int64
```

[16]:
```
# checking the distribution of categorical data
print(car_dataset.fuel.value_counts())
print(car_dataset.seller_type.value_counts())
print(car_dataset.transmission.value_counts())
print(car_dataset.owner.value_counts())
```

```
fuel
0    2153
1    2123
2      40
4      23
5       1
Name: count, dtype: int64
seller_type
0    3244
1     994
3     102
Name: count, dtype: int64
transmission
0    3892
1     448
```

```
Name: count, dtype: int64
owner
First Owner             2832
Second Owner            1106
Third Owner              304
Fourth & Above Owner      81
Test Drive Car            17
Name: count, dtype: int64
```

[18]:
```
# encoding "Fuel_Type" Column
car_dataset.replace({'fuel':{'Diesel':0,'Petrol':1,'CNG':2, 'LPG':4, 'Electric':
 ↪5}},inplace=True)

# encoding "Seller_Type" Column
car_dataset.replace({'seller_type':{'Individual':0, 'Dealer':1, 'Trustmark␣
 ↪Dealer':3}},inplace=True)

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

# encoding "Owner" Column
car_dataset.replace({'owner':{'First Owner':0,'Second Owner':1, 'Third Owner':
 ↪2, 'Fourth & Above Owner':3, 'Test Drive Car':4 }},inplace=True)
```

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

[19]:
```
                      name  year  selling_price  km_driven  fuel  \
0             Maruti 800 AC  2007          60000      70000     1
1  Maruti Wagon R LXI Minor  2007         135000      50000     1
2      Hyundai Verna 1.6 SX  2012         600000     100000     0
3    Datsun RediGO T Option  2017         250000      46000     1
4     Honda Amaze VX i-DTEC  2014         450000     141000     0

   seller_type  transmission  owner
0            0             0      0
1            0             0      0
2            0             0      0
3            0             0      0
4            0             0      1
```

[20]:
```
X = car_dataset.drop(['name','selling_price'],axis=1)
Y = car_dataset['selling_price']
```

[21]:
```
print(X)
```

```
      year  km_driven  fuel  seller_type  transmission  owner
0     2007      70000     1            0             0      0
1     2007      50000     1            0             0      0
```

```
2     2012      100000      0              0              0       0
3     2017       46000      1              0              0       0
4     2014      141000      0              0              0       1
…      …          …    …            …              …      …
4335  2014       80000      0              0              0       1
4336  2014       80000      0              0              0       1
4337  2009       83000      1              0              0       1
4338  2016       90000      0              0              0       0
4339  2016       40000      1              0              0       0

[4340 rows x 6 columns]
```

[22]:
```
print(Y)
```

```
0        60000
1       135000
2       600000
3       250000
4       450000
         …
4335    409999
4336    409999
4337    110000
4338    865000
4339    225000
Name: selling_price, Length: 4340, dtype: int64
```

[23]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1,␣
↪random_state=2)
```

[24]:
```
# loading the linear regression model
lin_reg_model = LinearRegression()
```

[25]:
```
lin_reg_model.fit(X_train,Y_train)
```

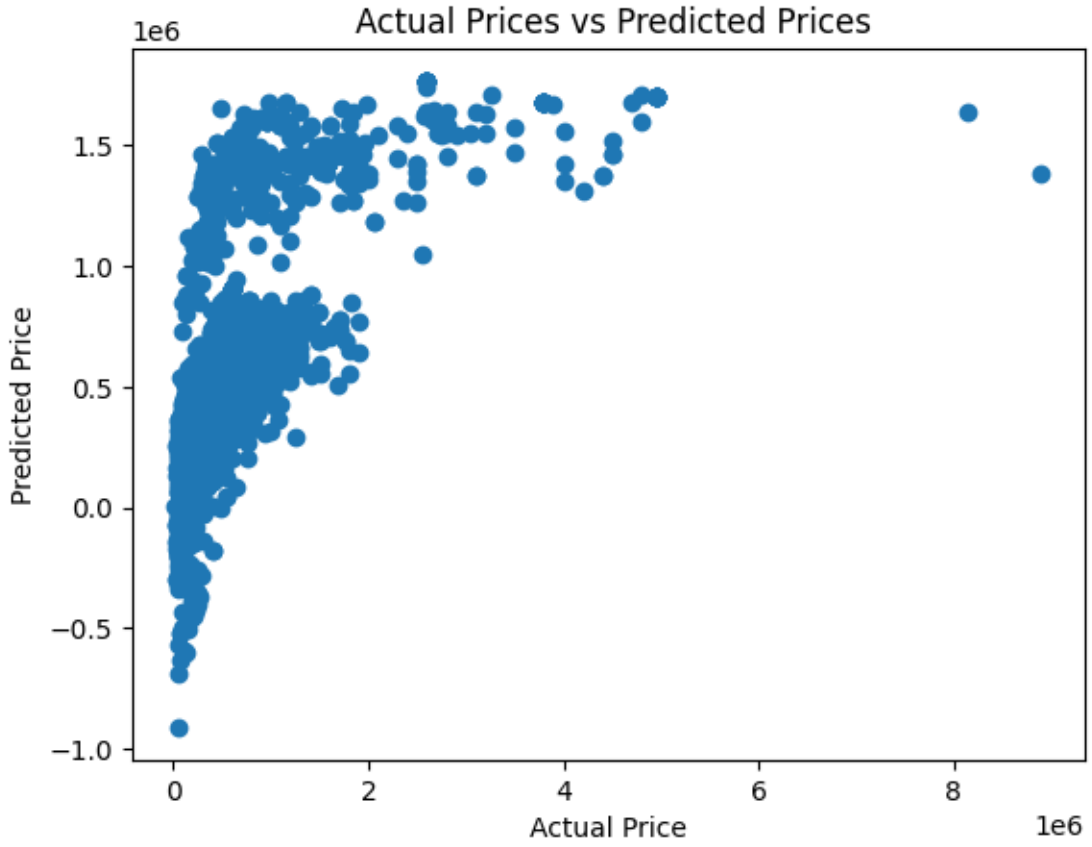
[25]:
```
LinearRegression()
```

[26]:
```
# prediction on Training data
training_data_prediction = lin_reg_model.predict(X_train)
```

[27]:
```
# R squared Error
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.4389170654811424
```

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

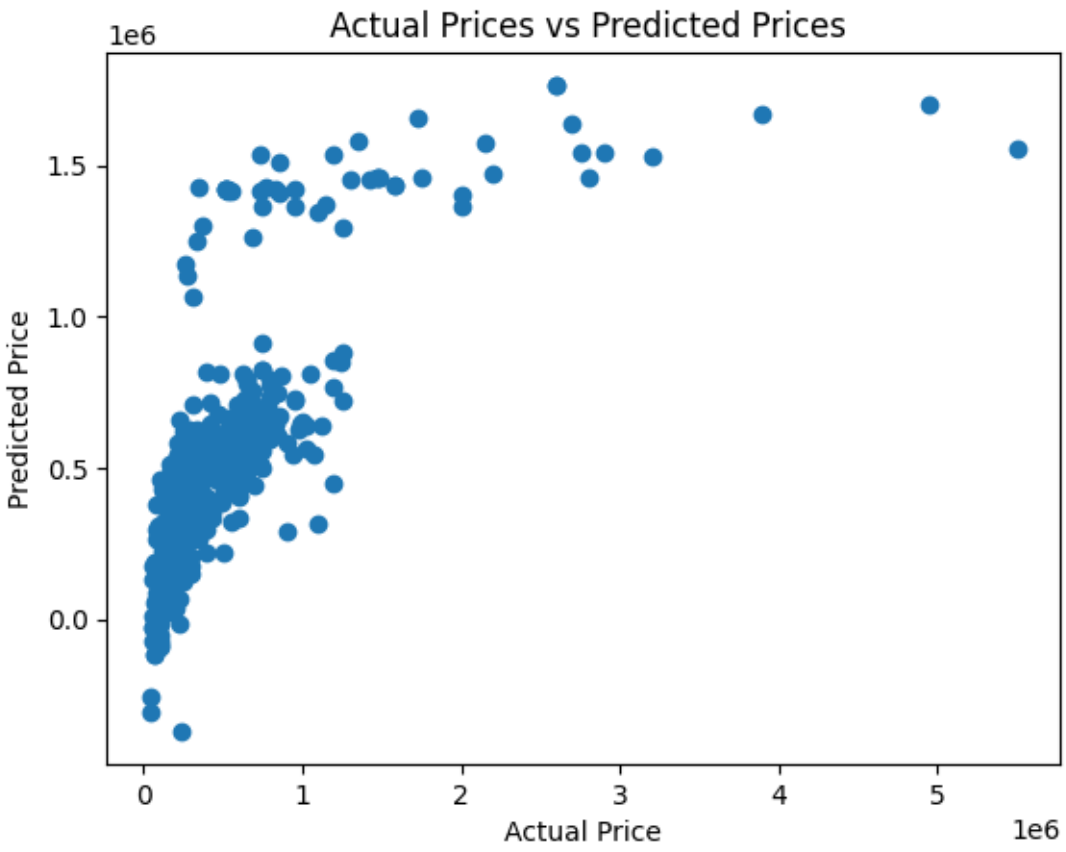


[29]:
```
# prediction on Training data
test_data_prediction = lin_reg_model.predict(X_test)
```

[30]:
```
# R squared Error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.5187339768078417
```

[31]:
```
plt.scatter(Y_test, test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```



[32]:
```
# loading the linear regression model
lass_reg_model = Lasso()
```

[33]:
```
lass_reg_model.fit(X_train,Y_train)
```

[33]:
```
Lasso()
```

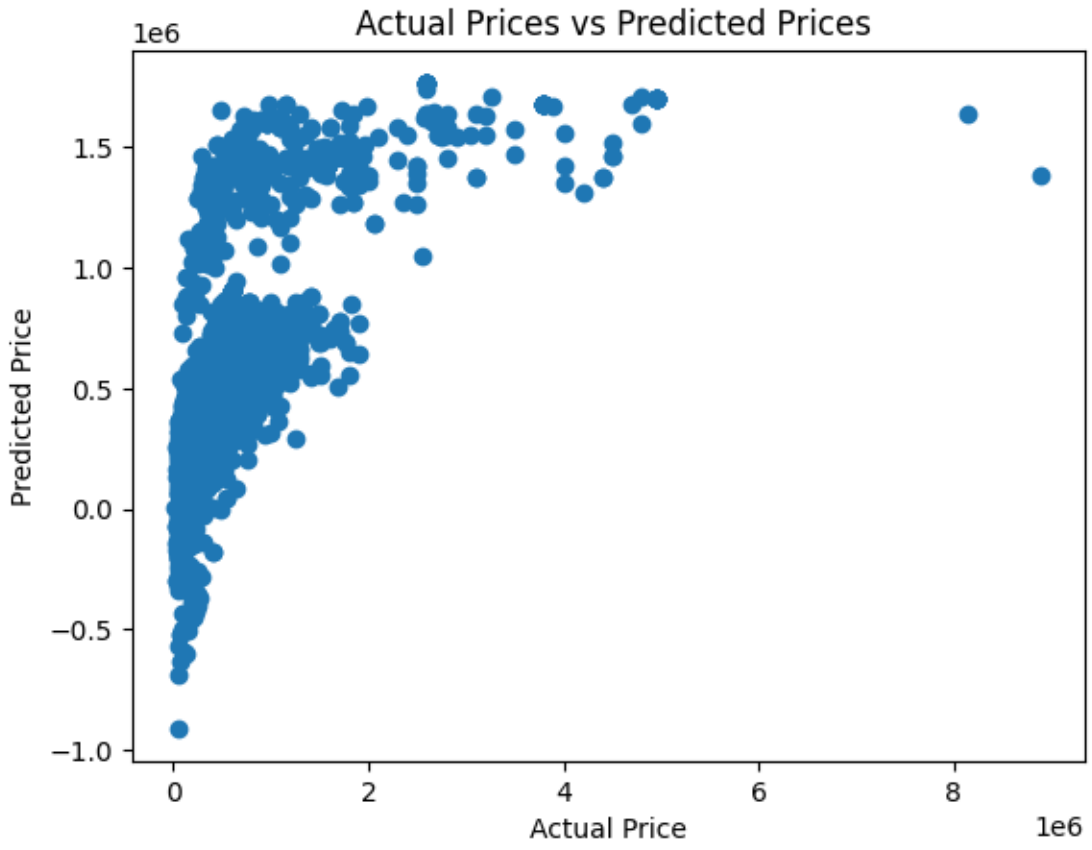
[34]:
```
# prediction on Training data
training_data_prediction = lass_reg_model.predict(X_train)
```

[35]:
```
# R squared Error
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.438917065432072
```

[36]:
```
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
```

```
plt.show()
```

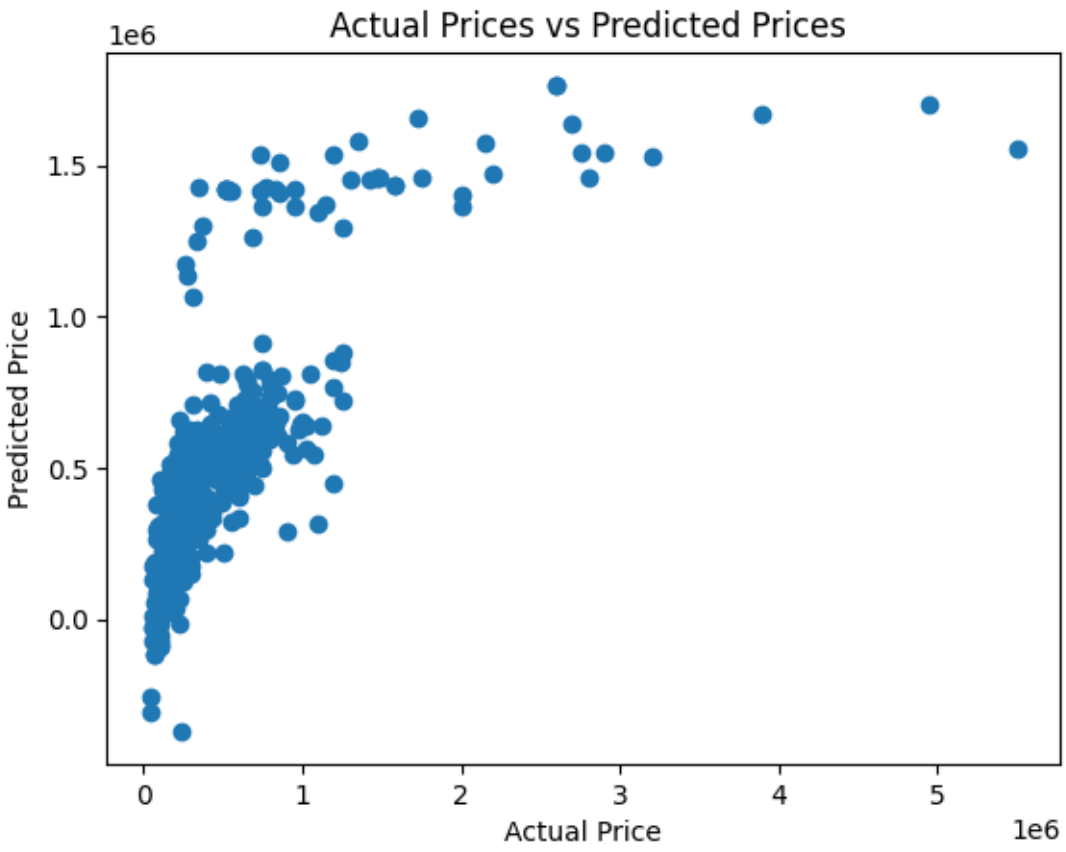


[37]:
```
# prediction on Training data
test_data_prediction = lass_reg_model.predict(X_test)
```

[38]:
```
# R squared Error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.5187332242453926
```

[39]:
```
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
1e6
1.5
1.0
0.5
0.0
Predicted Price
0
1
2
3
4
5
Actual Price
1e6