Oasis Infobyte Internship, May- 2023

Mayuri Arun Pathak, Data Science Intern

Car Price Prediction With Machine Learning

- Objective: To predict car price with machine learning.
- Dataset: https://raw.githubusercontent.com/amankharwal/Website-data/master/CarPrice.csv
- Dataset description:

- Steps we follow:
- 1)Import required libraries
- 2)Data Collection and Processing
- 3)Encoding the Categorical Data:
- 4)Splitting the data and Target
- 5)Splitting Training and Test data
- 6)Model Training
- 7)Model Evaluation
- 8)Visualize the actual prices and Predicted prices

Import required libraries

In [2]:
```
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
```

Data Collection and Processing

In [30]:
```
#loading the dataset
car_dataset = pd.read_csv('https://raw.githubusercontent.com/amankharwal/Website-data/master/CarPrice.csv')
```

In [31]:
```
# inspecting the first 5 rows of the dataframe
car_dataset.head()
```

Out[31]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22

5 rows × 26 columns

In [32]:
```
# checking the number of rows and columns
car_dataset.shape
```

Out[32]:
```
(205, 26)
```

In [33]:
```
# getting some information about the dataset
car_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

In [34]:
```
# checking the number of missing values
car_dataset.isnull().sum()
```

Out[34]:
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

In [35]:
```
# checking the distribution of categorical data
print(car_dataset.fueltype.value_counts())
print(car_dataset.aspiration.value_counts())
print(car_dataset.doornumber.value_counts())
print(car_dataset.carbody.value_counts())
print(car_dataset.drivewheel.value_counts())
print(car_dataset.enginelocation.value_counts())
print(car_dataset.fuelsystem.value_counts())
print(car_dataset.cylindernumber.value_counts())
print(car_dataset.enginetype.value_counts())
```

```
gas       185
diesel     20
Name: fueltype, dtype: int64
std      168
turbo     37
Name: aspiration, dtype: int64
four    115
two      90
Name: doornumber, dtype: int64
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: carbody, dtype: int64
fwd    120
rwd     76
4wd      9
Name: drivewheel, dtype: int64
front    202
rear       3
Name: enginelocation, dtype: int64
mpfi    94
2bbl    66
idi     20
1bbl    11
spdi     9
4bbl     3
mfi      1
spfi     1
Name: fuelsystem, dtype: int64
four      159
six        24
five       11
eight       5
two         4
three       1
twelve      1
Name: cylindernumber, dtype: int64
ohc      148
ohcf      15
ohcv      13
dohc      12
l         12
rotor      4
dohcv      1
Name: enginetype, dtype: int64
```

Encoding the Categorical Data:

In [36]:
```
#encoding "fueltype" column
car_dataset.replace({'fueltype':{'gas':0,'diesel':1}},inplace=True)

#encoding "aspiration" column
car_dataset.replace({'aspiration':{'std':0,'turbo':1}},inplace=True)

#encoding "doornumber" column
car_dataset.replace({'doornumber':{'four':1,'two':1}},inplace=True)

#encoding "carbody" column
car_dataset.replace({'carbody':{'sedan':0,'hatchback':1,'wagon':2,'hardtop':3,'convertible':4}},inplace=True)

#encoding "drivewheel" column
car_dataset.replace({'drivewheel':{'fwd':0,'rwd':1,'4wd':2}},inplace=True)

#encoding "enginelocation" column
car_dataset.replace({'enginelocation':{'front':0,'rear':1}},inplace=True)

#encoding "fuelsystem" column
car_dataset.replace({'fuelsystem':{'mpfi':0,'2bbl':1,'idi':2,'1bbl':3,'spdi':3,'4bbl':4,'mfi':5,'spfi':6}},inplace=True)

#encoding"cylindernumber" column
car_dataset.replace({'cylindernumber':{'four':0,'six':1,'five':2,'eight':3,'two':4,'three':5,'twelve':6}},inplace=True)

#encoding "enginetype" column
car_dataset.replace({'enginetype':{'ohc':0,'ohcf':1,'ohcv':2,'dohc':3,'l':4,'rotor':5,'dohcv':6}},inplace=True)
```

In [37]:
```
car_dataset.head()
```

Out[37]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg
0	1	3	alfa-romero giulia	0	0	1	4	1	0	88.6	...	130	0	3.47	2.68	9.0	111	5000	21	27
1	2	3	alfa-romero stelvio	0	0	1	4	1	0	88.6	...	130	0	3.47	2.68	9.0	111	5000	21	27
2	3	1	alfa-romero Quadrifoglio	0	0	1	1	1	0	94.5	...	152	0	2.68	3.47	9.0	154	5000	19	26
3	4	2	audi 100 ls	0	0	1	0	0	0	99.8	...	109	0	3.19	3.40	10.0	102	5500	24	30
4	5	2	audi 100ls	0	0	1	0	2	0	99.4	...	136	0	3.19	3.40	8.0	115	5500	18	22

5 rows × 26 columns

Splitting the data and Target

In [38]:
```
X = car_dataset.drop(['CarName','price'],axis=1)
Y = car_dataset['price']
```

In [39]:
```
print(X)
```

```
     car_ID  symboling  fueltype  aspiration  doornumber  carbody  drivewheel  \
0         1          3         0           0           1        4           1
1         2          3         0           0           1        4           1
2         3          1         0           0           1        1           1
3         4          2         0           0           1        0           0
4         5          2         0           0           1        0           2
..      ...        ...       ...         ...         ...      ...         ...
200     201         -1         0           0           1        0           1
201     202         -1         0           1           1        0           1
202     203         -1         0           0           1        0           1
203     204         -1         1           1           1        0           1
204     205         -1         0           1           1        0           1

     enginelocation  wheelbase  carlength  ...  cylindernumber  enginesize  \
0                 0       88.6      168.8  ...               0         130
1                 0       88.6      168.8  ...               0         130
2                 0       94.5      171.2  ...               1         152
3                 0       99.8      176.6  ...               0         109
4                 0       99.4      176.6  ...               2         136
..              ...        ...        ...  ...             ...         ...
200               0      109.1      188.8  ...               0         141
201               0      109.1      188.8  ...               0         141
202               0      109.1      188.8  ...               1         173
203               0      109.1      188.8  ...               1         145
204               0      109.1      188.8  ...               0         141

     fuelsystem  boreratio  stroke  compressionratio  horsepower  peakrpm  \
0             0       3.47    2.68               9.0         111     5000
1             0       3.47    2.68               9.0         111     5000
2             0       2.68    3.47               9.0         154     5000
3             0       3.19    3.40              10.0         102     5500
4             0       3.19    3.40               8.0         115     5500
..          ...        ...     ...               ...         ...      ...
200           0       3.78    3.15               9.5         114     5400
201           0       3.78    3.15               8.7         160     5300
202           0       3.58    2.87               8.8         134     5500
203           2       3.01    3.40              23.0         106     4800
204           0       3.78    3.15               9.5         114     5400

     citympg  highwaympg
0         21          27
1         21          27
2         19          26
3         24          30
4         18          22
..       ...         ...
200       23          28
201       19          25
202       18          23
203       26          27
204       19          25

[205 rows x 24 columns]
```

In [40]:
```
print(Y)
```

```
0      13495.0
1      16500.0
2      16500.0
3      13950.0
4      17450.0
        ...
200    16845.0
201    19045.0
202    21485.0
203    22470.0
204    22625.0
Name: price, Length: 205, dtype: float64
```

Splitting Training and Test data

In [41]:
```
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 [42]:
```
# loading the linear regression model
lin_reg_model = LinearRegression()
```

In [43]:
```
lin_reg_model.fit(X_train,Y_train)
```

Out[43]:
```
LinearRegression()
```

Model Evaluation

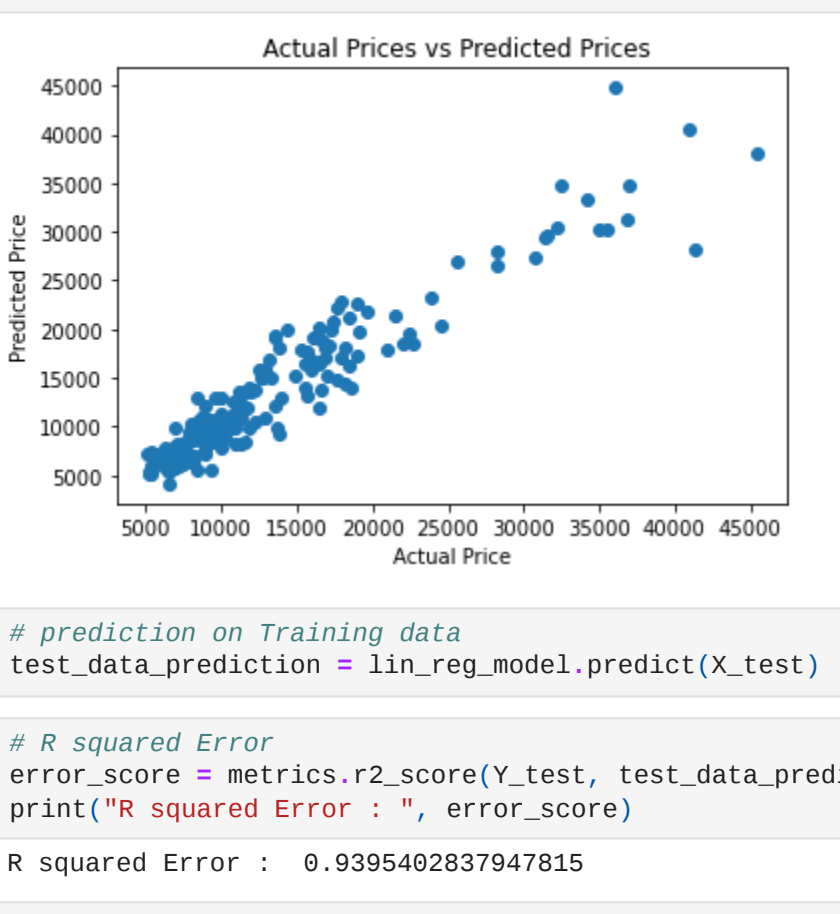
In [44]:
```
# prediction on Training data
training_data_prediction = lin_reg_model.predict(X_train)
```

In [45]:
```
# R squared Error
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.9027472016097514
```

Visualize the actual prices and Predicted prices

In [46]:
```
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```

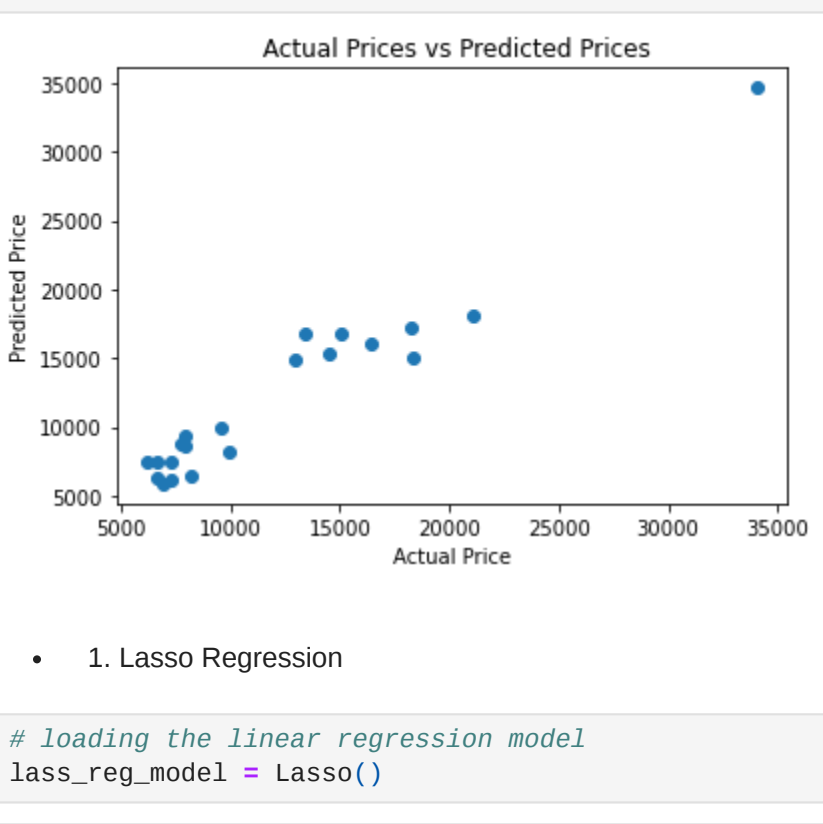


In [47]:
```
# prediction on Training data
test_data_prediction = lin_reg_model.predict(X_test)
```

In [48]:
```
# R squared Error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.9395402837947815
```

In [49]:
```
plt.scatter(Y_test, test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```



- 1. Lasso Regression

In [50]:
```
# loading the linear regression model
lass_reg_model = Lasso()
```

In [51]:
```
lass_reg_model.fit(X_train,Y_train)
```

Out[51]:
```
Lasso()
```

Model Evaluation

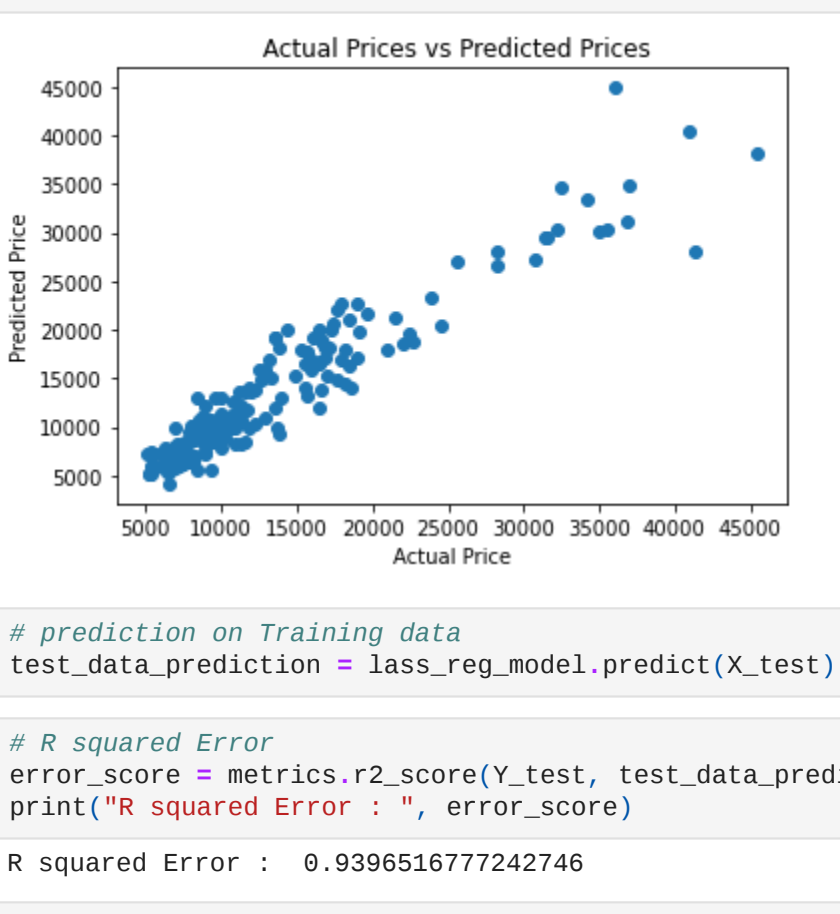
In [52]:
```
# prediction on Training data
training_data_prediction = lass_reg_model.predict(X_train)
```

In [53]:
```
# R squared Error
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.9027278059187078
```

Visualize the actual prices and Predicted prices

In [54]:
```
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```

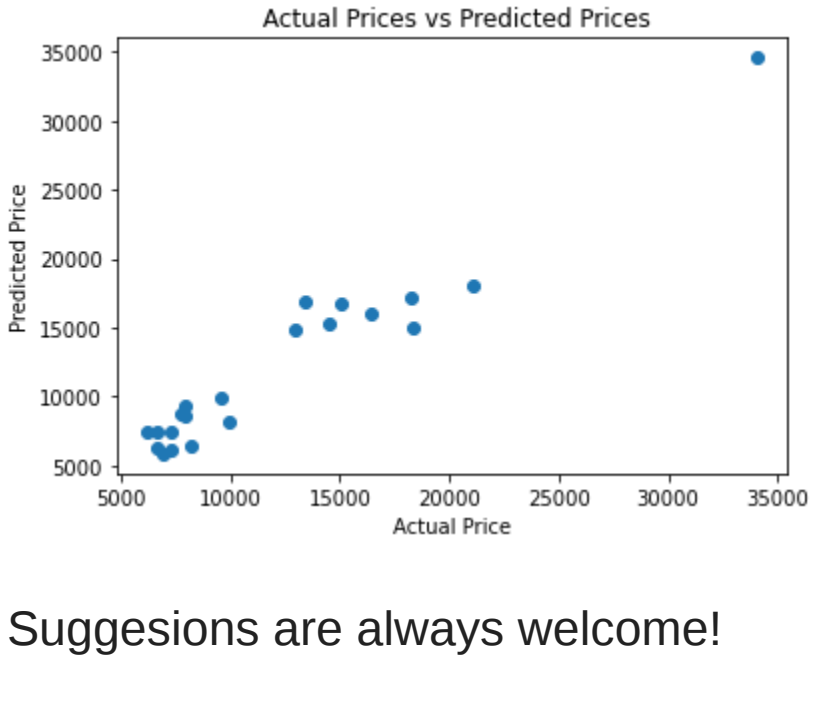


In [55]:
```
# prediction on Training data
test_data_prediction = lass_reg_model.predict(X_test)
```

In [56]:
```
# R squared Error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.9396516777242746
```

In [57]:
```
plt.scatter(Y_test, test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```



Suggesions are always welcome!

Thank You!