Car Price Prediction using Linear Regression

Use Linear Regression(Ordinary Least Square)to Predict Car Price
This tutorial explains the necessary steps in coding.To get a solution viewers must realise that OLS require the fulfillment of assumptions for effective modeling To learn Regression technique join our free courses at ybifoundation.org

Get Understanding about Dataset

There are 9 variables in the dataset

1. Brand-manufacturing company
2. Model-model of cars
3. Year-year of manufacturing
4. Selling_Price-selling price of car
5. KM_Driven-total km driven
6. Fuel-type of fuel used in car
7. Seller_Type - type of seller
8. Transmission-type of transmission in car
9. Owner-whether current owner is first owner or repurchased

Import Library

In [3]:

```
import pandas as pd
import numpy as np
```

Import CSV as DataFrame

In [4]:

```
df= pd.read_csv('Dataset-main/Dataset-main/Car Price.csv')
```

Get the First FIve Rows of DataFrame

In [5]:

```
df.head()
```

Out[5]:

	Brand	Model	Year	Selling_Price	KM_Driven	Fuel	Seller_Type	Transmission	Owner
0	Maruti	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

Get Information of DataFrame

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Brand          4340 non-null   object
 1   Model          4340 non-null   object
 2   Year           4340 non-null   int64 
 3   Selling_Price  4340 non-null   int64 
 4   KM_Driven      4340 non-null   int64 
 5   Fuel           4340 non-null   object
 6   Seller_Type    4340 non-null   object
 7   Transmission   4340 non-null   object
 8   Owner          4340 non-null   object
dtypes: int64(3), object(6)
memory usage: 305.3+ KB
```

Get the Summary of Statistics

In [8]:

```
df.describe()
```

Out[8]:

	Year	Selling_Price	KM_Driven
count	4340.000000	4.340000e+03	4340.000000
mean	2013.090783	5.041273e+05	66215.777419
std	4.215344	5.785487e+05	46644.102194
min	1992.000000	2.000000e+04	1.000000
25%	2011.000000	2.087498e+05	35000.000000
50%	2014.000000	3.500000e+05	60000.000000
75%	2016.000000	6.000000e+05	90000.000000
max	2020.000000	8.900000e+06	806599.000000

In [9]:

```
df[['Brand']].value_counts()
```

Out[9]:

```
Brand
Maruti           1280
Hyundai           821
Mahindra          365
Tata              361
Honda             252
Ford              238
Toyota            206
Chevrolet         188
Renault           146
Volkswagen        107
Skoda              68
Nissan             64
Audi               60
BMW                39
Fiat               37
Datsun             37
Mercedes-Benz      35
Mitsubishi          6
Jaguar              6
Land                5
Ambassador          4
Volvo               4
Jeep                3
OpelCorsa           2
MG                  2
Isuzu               1
Force               1
Daewoo              1
Kia                 1
dtype: int64
```

In [11]:

```
df[['Model']].value_counts()
```

Out[11]:

```
Model
Maruti Swift Dzire VDI                      69
Maruti Alto 800 LXI                         59
Maruti Alto LXi                             47
Hyundai EON Era Plus                        35
Maruti Alto LX                              35
                                            ..
Mahindra KUV 100 G80 K4 Plus                 1
Mahindra KUV 100 mFALCON D75 K8              1
Mahindra KUV 100 mFALCON D75 K8 AW           1
Mahindra KUV 100 mFALCON G80 K2 Plus         1
Volvo XC60 D5 Inscription                    1
Length: 1491, dtype: int64
```

In [12]:

```
df[['Fuel']].value_counts()
```

Out[12]:

```
Fuel
Diesel      2153
Petrol      2123
CNG           40
LPG           23
Electric       1
dtype: int64
```

In [14]:

```
df[['Seller_Type']].value_counts()
```

Out[14]:

```
Seller_Type
Individual          3244
Dealer               994
Trustmark Dealer     102
dtype: int64
```

In [15]:

```
df[['Transmission']].value_counts()
```

Out[15]:

```
Transmission
Manual         3892
Automatic       448
dtype: int64
```

In [16]:

```
df[['Owner']].value_counts()
```

Out[16]:

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

Get Column Names

In [17]:

```
df.columns
```

Out[17]:

```
Index(['Brand', 'Model', 'Year', 'Selling_Price', 'KM_Driven', 'Fuel',
       'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

Get Shape of DataFrame

In [18]:

```
df.shape
```

Out[18]:

```
(4340, 9)
```

Get Encoding of Categorical Features

In [19]:

```
df.replace({'Fuel':{'Petrol':0,'Diesel':1,'CNG':2,'LPG':3,'Electric':4}},inplace=True)
```

In [20]:

```
df.replace({'Seller_Type':{'Individual':0,'Dealer':1,'Trustmark Dealer':2}},inplace=True)
```

In [21]:

```
df.replace({'Transmission':{'Manual':0,'Automatic':1}},inplace=True)
```

In [22]:

```
Second Owner':1,'Third Owner':2,'Fourth & Above Owner':3,'Test Drive Car':4}},inplace=True)
```

In [24]:

```
#X= pd.get_dummies(X, columns=['Fuel','Seller_Type','Transmission','Owner'],drop_first = Tr
```

Its always recommended to use dummy variables in case of categorical features

Define y(depedent or label or target variable) and X(independent or features or attribute Variable)

In [25]:

```
y = df['Selling_Price']
```

In [26]:

```
y.shape
```

Out[26]:

```
(4340,)
```

In [27]:

```
y
```

Out[27]:

```
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
```

In [28]:

```
X = df[['Year','KM_Driven','Fuel','Seller_Type','Transmission','Owner']]
```

or use drop function to define X

In [30]:

```
#X=df.drop(['Brand','Model','Selling_Price'],axis=1)
```

In [31]:

```
X.shape
```

Out[31]:

```
(4340, 6)
```

In [32]:

```
X
```

Out[32]:

	Year	KM_Driven	Fuel	Seller_Type	Transmission	Owner
0	2007	70000	0	0	0	0
1	2007	50000	0	0	0	0
2	2012	100000	1	0	0	0
3	2017	46000	0	0	0	0
4	2014	141000	1	0	0	1
...	...	...	...	...	...	...
4335	2014	80000	1	0	0	1
4336	2014	80000	1	0	0	1
4337	2009	83000	0	0	0	1
4338	2016	90000	1	0	0	0
4339	2016	40000	0	0	0	0

4340 rows × 6 columns

Get Train Test Split

In [33]:

```
from sklearn.model_selection import train_test_split
```

In [34]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=2529)
```

In [35]:

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[35]:

```
((3038, 6), (1302, 6), (3038,), (1302,))
```

Get Model Train

In [36]:

```
from sklearn.linear_model import LinearRegression
```

In [37]:

```
lr = LinearRegression()
```

In [38]:

```
lr.fit(X_train,y_train)
```

Out[38]:

```
LinearRegression()
```

Get Model Prediction

In [39]:

```
y_pred =lr.predict(X_test)
```

In [40]:

```
y_pred.shape
```

Out[40]:

```
(1302,)
```

In [41]:

```
y_pred
```

Out[41]:

```
array([502458.82786413, 646333.17428704, 521962.74075836, ...,
       620183.32683781, 315403.82788569, 731862.54196037])
```

Get Model Evaluation

In [42]:

```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

In [43]:

```
mean_squared_error(y_test,y_pred)
```

Out[43]:

```
193242972302.19547
```

In [44]:

```
mean_absolute_error(y_test,y_pred)
```

Out[44]:

```
228808.95522977872
```

In [45]:

```
r2_score(y_test,y_pred)
```

Out[45]:

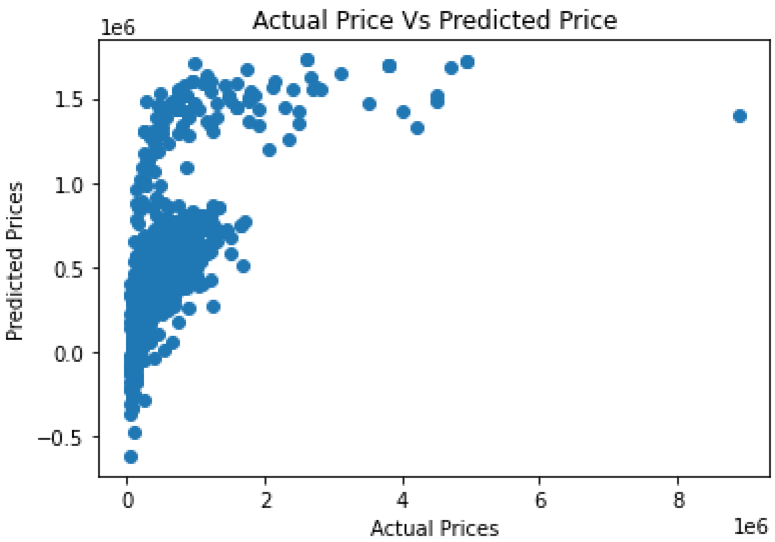
```
0.40755633943708414
```

R-Square is very low signfies need for model improvement Encourage viewers to find the probable reasons for model poor performance

Get Visualization of Actual Vs Predicted Results

In [46]:

```
import matplotlib.pyplot as plt
plt.scatter(y_test,y_pred)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual Price Vs Predicted Price')
plt.show()
```



Get Future Predictions

Lets select a random sample from existing dataset as new value Steps to follow

1.Extract a random row using sample function

2.Separate X and y

3.Predict

In [47]:

```
df_new =df.sample(1)
```

In [48]:

```
df_new
```

Out[48]:

	Brand	Model	Year	Selling_Price	KM_Driven	Fuel	Seller_Type	Transmission	Owner
3990	Maruti	Maruti Alto 800 LXI	2018	300000	17000	0	0	0	0

In [49]:

```
df_new.shape
```

Out[49]:

```
(1, 9)
```

In [50]:

```
X_new = df_new.drop(['Brand','Model','Selling_Price'],axis=1)
```

In [51]:

```
y_pred_new = lr.predict(X_new)
```

In [52]:

```
y_pred_new
```

Out[52]:

```
array([504452.60823062])
```

In [ ]:

```

```