Bike Price Prediction using Linear Regression

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

```
df =
pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Bike
%20Prices.csv')
```

Get Missing Values Drop

```
df=df.dropna()
```

Get Information of Dataframe

```
df.info()   #gives column name, count,  not null category, D-type(data
type)

<class 'pandas.core.frame.DataFrame'>
Int64Index: 626 entries, 0 to 625
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Brand              626 non-null    object 
 1   Model              626 non-null    object 
 2   Selling_Price      626 non-null    int64  
 3   Year               626 non-null    int64  
 4   Seller_Type        626 non-null    object 
 5   Owner              626 non-null    object 
 6   KM_Driven          626 non-null    int64  
 7   Ex_Showroom_Price  626 non-null    float64
dtypes: float64(1), int64(3), object(4)
memory usage: 44.0+ KB

df.describe()   #gives the linear relation of each column with another
column

       Selling_Price         Year      KM_Driven  Ex_Showroom_Price
count     626.000000   626.000000     626.000000       6.260000e+02
mean    59445.164537  2014.800319   32671.576677       8.795871e+04
std     59904.350888     3.018885   45479.661039       7.749659e+04
```

```
min      6000.000000  2001.000000    380.000000       3.049000e+04
25%     30000.000000  2013.000000  13031.250000       5.485200e+04
50%     45000.000000  2015.000000  25000.000000       7.275250e+04
75%     65000.000000  2017.000000  40000.000000       8.703150e+04
max    760000.000000  2020.000000  585659.000000      1.278000e+06
```

```
df.head()
```

```
   Brand               Model  Selling_Price  Year Seller_Type
Owner  \
0    TVS          TVS XL 100          30000  2017  Individual  1st
owner
1  Bajaj       Bajaj  ct 100          18000  2017  Individual  1st
owner
2     Yo            Yo Style          20000  2011  Individual  1st
owner
3  Bajaj  Bajaj Discover 100          25000  2010  Individual  1st
owner
4  Bajaj  Bajaj Discover 100          24999  2012  Individual  2nd
owner

   KM_Driven  Ex_Showroom_Price
0       8000            30490.0
1      35000            32000.0
2      10000            37675.0
3      43000            42859.0
4      35000            42859.0
```

```
df.isnull().sum() #(df.isna().sum() gives same result)
#gives the sum of all null values columns-wise
```

```
Brand                0
Model                0
Selling_Price        0
Year                 0
Seller_Type          0
Owner                0
KM_Driven            0
Ex_Showroom_Price    0
dtype: int64
```

```
df.nunique()   #gives total no. of unique entries
```

```
Brand                 18
Model                183
Selling_Price         99
Year                  18
Seller_Type            2
Owner                  4
KM_Driven            219
```

```
Ex_Showroom_Price    230
dtype: int64
```

```
df.shape
```

```
(626, 8)
```

#Get Categories and Counts of Categorical Variables

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

```
Brand
Honda       170
Bajaj       143
Hero        108
Yamaha       94
Royal        40
TVS          23
Suzuki       18
KTM           6
Mahindra      6
Kawasaki      4
UM            3
Activa        3
Harley        2
Vespa         2
BMW           1
Hyosung       1
Benelli       1
Yo            1
dtype: int64
```

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

```
Model
Honda Activa [2000-2015]                       23
Honda CB Hornet 160R                           22
Bajaj Pulsar 180                               20
Yamaha FZ S V 2.0                              16
Bajaj Discover 125                             16
                                               ..
Royal Enfield Thunderbird 500                   1
Royal Enfield Continental GT [2013 - 2018]      1
Royal Enfield Classic Stealth Black             1
Royal Enfield Classic Squadron Blue             1
Yo Style                                        1
Length: 183, dtype: int64
```

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

```
Seller_Type
Individual     623
```

```
Dealer          3
dtype: int64
```

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

```
Owner
1st owner    556
2nd owner     66
3rd owner      3
4th owner      1
dtype: int64
```

Get Column Names

```
df.columns
```

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

Get Encoding of Categorical Features

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

```
df.replace({'Owner':{'1st owner':0,'2nd owner':1,'3rd owner':2,'4th 
owner':3}},inplace=True)
```

```
#X=pd.get_dummies(X,columns=['Seller_Type','Owner'],drop_first=True)
```

```
df
```

```
        Brand                          Model  Selling_Price  Year  \
0         TVS                     TVS XL 100          30000  2017   
1       Bajaj                   Bajaj  ct 100          18000  2017   
2          Yo                       Yo Style          20000  2011   
3       Bajaj             Bajaj Discover 100          25000  2010   
4       Bajaj             Bajaj Discover 100          24999  2012   
..        ...                            ...            ...   ...   
621    Harley      Harley-Davidson Street 750         330000  2014   
622  Kawasaki  Kawasaki Ninja 650 [2018-2019]         300000  2011   
623  Kawasaki  Kawasaki Ninja 650 [2018-2019]         425000  2017   
624    Suzuki                 Suzuki GSX S750         760000  2019   
625    Harley      Harley-Davidson Street Bob         750000  2013   

     Seller_Type  Owner  KM_Driven  Ex_Showroom_Price  
0              0      0       8000            30490.0  
1              0      0      35000            32000.0  
2              0      0      10000            37675.0  
3              0      0      43000            42859.0  
4              0      1      35000            42859.0  
```

```
..           ...    ...        ...              ...
621            0      3       6500         534000.0
622            0      0      12000         589000.0
623            0      1      13600         599000.0
624            0      0       2800         752020.0
625            0      1      12000        1278000.0

[626 rows x 8 columns]
```

Define y

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

```
y.shape
```

```
(626,)
```

```
y
```

```
0       30000
1       18000
2       20000
3       25000
4       24999
        ...
621    330000
622    300000
623    425000
624    760000
625    750000
Name: Selling_Price, Length: 626, dtype: int64
```

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

```
X.shape
```

```
(626, 5)
```

```
X
```

```
     Year  Seller_Type  Owner  KM_Driven  Ex_Showroom_Price
0    2017            0      0       8000            30490.0
1    2017            0      0      35000            32000.0
2    2011            0      0      10000            37675.0
3    2010            0      0      43000            42859.0
4    2012            0      1      35000            42859.0
..    ...          ...    ...        ...                ...
621  2014            0      3       6500           534000.0
622  2011            0      0      12000           589000.0
623  2017            0      1      13600           599000.0
624  2019            0      0       2800           752020.0
625  2013            0      1      12000          1278000.0
```

```
[626 rows x 5 columns]
```

Train Test Split

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

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=
train_test_split(X,y,test_size=0.3,random_state=2323)
```

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

```
((438, 5), (188, 5), (438,), (188,))
```

Get Model Test

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

```
lr= LinearRegression()
```

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

```
LinearRegression()
```

Get Model Prediction

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

```
y_pred.shape
```

```
(188,)
```

```
y_pred
```

```
array([ 35870.79961642,  65547.96126415,  12371.09961996,
64148.8895839 ,
         7062.69388653,  54402.63506136,  -1912.44035159,
29775.20161928,
        14243.08480753,  22080.83178243,  84025.43006156,
38292.33063928,
        29354.69337304,  21525.66151869,  51718.07386067,
133108.79218061,
        36717.94936617,  61355.37093641,  28387.67135344,
43699.33828063,
        73829.7721122 , 121281.37404139, 257025.1574888 , -
13165.16881032,
        46855.75420218,  52525.81083583,  21528.77516102,
58963.76859935,
        36453.23827565,  59092.23873301,  52537.87619985,
31275.42528094,
```

```
        50620.8218115 ,  67631.38902032,  56630.86444395,
117293.32518539,
        34037.48760149,  49778.37056618, 111705.80584799,
21536.11351636,
        44867.13135661,  74909.17128765, 421105.83467059,
73670.58645457,
        51287.25308003,  14423.62915765,  99846.44490246,
54487.16853425,
       101946.69838917,  79576.38985325, 117362.89863505,
37565.98215828,
        68125.63747921,  35121.70517691,  52265.51803945,
54517.4438797 ,
        46366.66325235, 192872.63669555,  50035.40444538,
95226.6560928 ,
        50547.81489465,  25268.83224858,  62635.88716991,
34669.44918167,
       117369.04807865,  79570.16256859,  29637.7731142 ,
29989.47887016,
         7756.2225401 ,  29835.95697668,  67271.25283088,
44340.23851247,
        91684.46261394,  63810.8340198 ,  80449.25240847,
35102.22261493,
        43725.02582983,  61355.87846011,  32852.18571348,
47285.70027847,
        81095.64161965,  53340.80472393,  35507.12858801,
127940.00023025,
        54748.27265082,  33764.27590934,  21555.24112081,
47793.88009026,
        99600.47368742,  26427.56110399,  50843.0311352 ,
2393.36675538,
        44245.38463449,  30557.61972929,  64331.44738662,
46052.30174221,
       127948.19300162,  25875.99311672,  54013.85916036,
47862.33637501,
        54548.65103582,  45496.28368746,  31788.72526496,
25415.45959194,
        31554.54862708,  41029.82132688,  46789.06447165,
68798.88433921,
       118678.1466251 ,  38280.26527526,  51618.90192728,
60607.30748903,
        40486.80042414,  23634.458532  ,  48728.72961516,
133108.01377003,
        49844.92467094,  14273.22715688,  65610.31437702,
111888.54121412,
        71709.81448763, 140861.10089833,  29964.08104634,
49711.03130947,
        95879.34214946,  58311.43819235,  21723.74978983,
134163.14808106,
       132056.05882463,  55999.53310058,  40556.27936137,
71719.15541461,
```

```
        61303.17569047,  69364.60182646,  51074.45481062,
28320.44222635,
        51773.58311705, 239835.53704415,  64318.1023156 ,
37399.68698658,
        38289.86152092,  59716.48021255,  21721.41455808,
14608.88730471,
        52467.53834686,  50611.39269332,  50026.06245928,
11788.30069698,
        59391.43797137,  51698.22439083,  59672.88921996,
38739.74209449,
        52091.16644551,  17201.62221796,  70016.38987149,
30835.15898424,
       101950.59044208,  46806.18950446,  60690.11143484,
126922.35408882,
         7089.9382569 ,  84927.52216659, 109939.32315248,
60867.65085507,
        13311.86850318,  28477.46071707,  35870.79961642,
40632.64055195,
       130166.96427995,  68066.86002907,  34033.59554858,
54548.88455899,
        42884.61600956,  37735.67541855,  25863.9277527 ,
58185.3564429 ,
       138145.48035431, 112389.92337885,  60911.95193693,
50891.34804191,
       121286.7450744 ,  30918.49434596,  51936.08269254,
12699.70125396,
        57749.07933698,  45996.1445545 ,  23637.15540436,
35095.67384671])
```

Get Model Evaluation

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

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

```
293014701.66187763
```

```
mean_absolute_percentage_error(y_test,y_pred)
```

```
0.24947046843526693
```

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

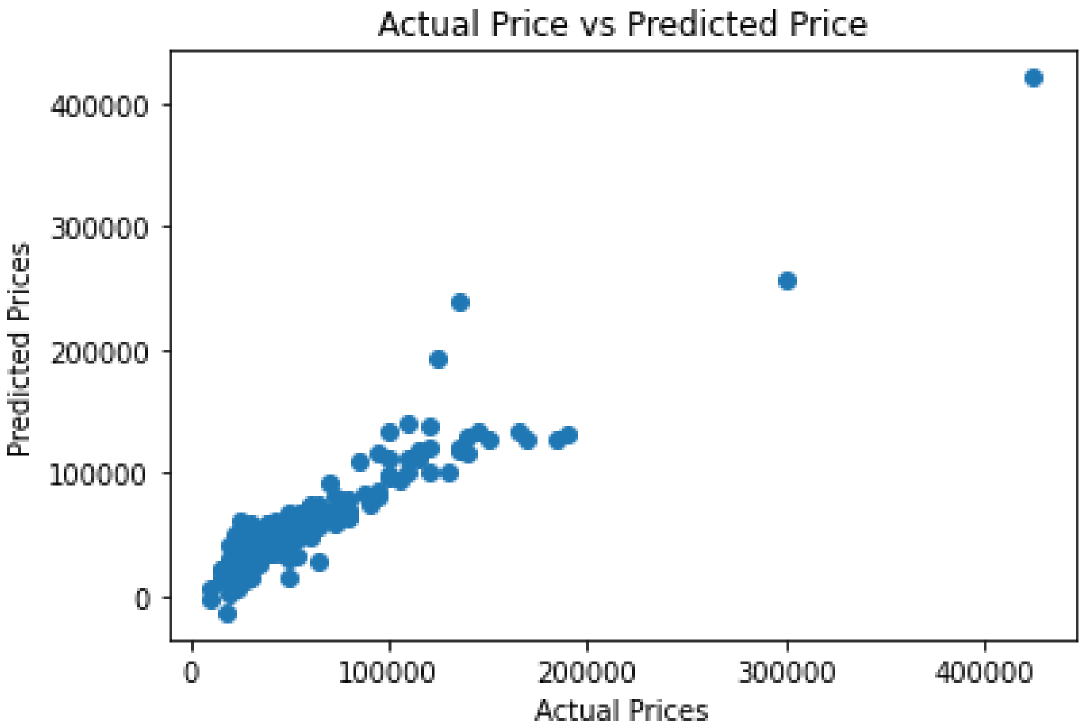
```
0.8738616789555578
```

#Geet Visualization of Actual Vs Predicted Results

```
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()
```




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

```
X_new
```

```
   Brand                Model  Selling_Price  Year  Seller_Type  Owner  \
4  Bajaj  Bajaj Discover 100          24999  2012            0      1

   KM_Driven  Ex_Showroom_Price
4      35000            42859.0
```

```
X_new.shape
```

```
(1, 8)
```

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

```
X_new
```

```
   Year  Seller_Type  Owner  KM_Driven  Ex_Showroom_Price
4  2012            0      1      35000            42859.0
```

```
X_new.shape
```

```
(1, 5)
```

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

```
array([11014.37068462])
```