Bike Price Prediction using Linear Regression

Import Library

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

Import CSV file as Dataframe

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

Get the first five rows of dataframe

```
df.head()
```


Get Information of Dataframe

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1061 entries, 0 to 1060
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Brand             1061 non-null   object
```

```
 1   Model             1061 non-null   object
 2   Selling_Price     1061 non-null   int64
 3   Year              1061 non-null   int64
 4   Seller_Type       1061 non-null   object
 5   Owner             1061 non-null   object
 6   KM_Driven         1061 non-null   int64
 7   Ex_Showroom_Price  626 non-null    float64
dtypes: float64(1), int64(3), object(4)
memory usage: 66.4+ KB
```

Get Missing values drop

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

Get Summary Statistics

```
df.describe()
```

Saving...

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
```

Saving...

```
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 Shape of Dataframe

```
df.shape
```

```
(626, 8)
```

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)
```

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

Define target variable (y) and features (X)

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

```
y.shape
```

Saving...

```
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
```

Get 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=2529)
```

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

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

Get Model Train

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

```
model = LinearRegression()
```

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

```
LinearRegression()
```

Get Model Prediction

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

```
y_pred.shape
```

```
(188,)
```

```
y_pred
```

```
array([ 27210.52271465,  56340.08335163,  63471.94671996,  53627.63844785,
        55612.75744268,  53888.92259719,  33751.35275102,  60311.4950183 ,
       113713.05684467,  76639.49332954,  27826.7399381 ,  49919.83255841,
        65886.64311457,  26755.12664064,  48277.75426038, 127646.56079335,
        70047.10661635,  39350.67963653,  36081.03597878,  45360.79436339,
        48079.89470577,  44803.02464799,  55161.44026111,  71041.51821318,
        91689.22699159,  49301.53594645,  55988.19326252, 108171.54600296,
        32771.06897901,  25468.20072996,  17128.61806164, 179271.41130746,
        45698.99857622,  31371.09285079,  67886.52106737,  41492.49575815,
        56855.22238602,  47820.47003468,  74682.14053958,  24984.21822736,
        55374.00513699,  41412.36775222,  67991.60287764,  26553.59421844,
        89788.69870689,  45764.83633686, 133888.03770389, 106988.113825  ,
                        25485946,  79512.43778826,  63914.38088173,
                        13623937,  -5396.37132904,  70377.44571174,
                        92478411,  67509.85836352,  59735.05378847,
        22199.83644217,  15374.18984158,  44510.76819427,  30279.52476752,
       108243.77037514,  19291.8895874 ,  53614.312976  ,  59230.23269131,
        60174.2108109 ,  45924.63468736,  25770.81883496,  63471.36257814,
       242123.45729792,  61387.72544548,  56510.98127074,  48123.28087213,
        51668.27442011,  90279.76190495,  14827.76533556, 112437.70820504,
        35066.88027405,  30902.41069172,  31441.48921433, 125593.75847157,
        27705.38813164, -11590.29205553,  15582.17108685,  75113.64511232,
       504085.44522282, 123545.42050116,  74770.89327697,  50747.47663245,
        44174.3618212 ,  25426.7156106 ,  30298.3052462 ,  47625.67836414,
        27850.37544807,  28845.23330928,  31580.38624692,  32309.63375635,
        47979.16788554,  65955.46375944,  13432.28218017,  15368.80064986,
        31973.23052409, 110353.92870546,  68181.49509136,  23143.49139797,
        53194.65732076,  34603.36376989,  56002.50967868,  62432.66994305,
```

Saving...

```
391470.77533201,   3558.29480891,  36019.18494305,  70876.34866549,
 72890.00667025, 137596.01384364,  27620.36308877, 135789.30486854,
 39674.40366791,  58367.0924453 ,  42401.21202624,  61864.4379567 ,
 42688.89652842,  63710.34571021,  10604.39360071,  38458.82820943,
112251.84744225, 115403.00577536,  13658.41734785,  36196.83359584,
 54146.22998932,  97297.85724851,  55029.68137265,  22923.26533437,
104569.97029689,  41965.75852017,  38759.68546491,  28930.61369011,
 45231.66612551,  48475.43422775,  26739.7225731 ,  53598.65972203,
 32558.54954524,  32212.22834942,  68172.98738422,  71839.47716461,
 32003.46692215,  40652.69995971,  39935.92211843,  63444.41846202,
 44545.5818771 , 120873.38389616,  60926.58683174,  62641.82167496,
 60816.47379994,  27098.95433573,  26803.64749618,  48956.00468627,
 62032.88118713,  26471.97495723, 104937.23068766, 132903.3578847 ,
 37469.2040942 ,  57579.12080094,  40371.00915736,  -7039.40662503,
 26485.40030077,  90782.42554145,  52153.21149321,  56453.74542453,
 80440.59426003,  31890.46870273,  49505.97985573,  24288.36959514,
 25540.47481573, 117708.26333955,  23399.66596746,  63678.40865459,
 70144.29372668,  33434.89010059,  60885.29444481,  58389.55370878,
 35118.7040348 ,  58729.4540196 ,  34627.9532246 ,  38583.4623973 ])
```

Get Model Evaluation

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

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

```
554715615.5043668
```

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

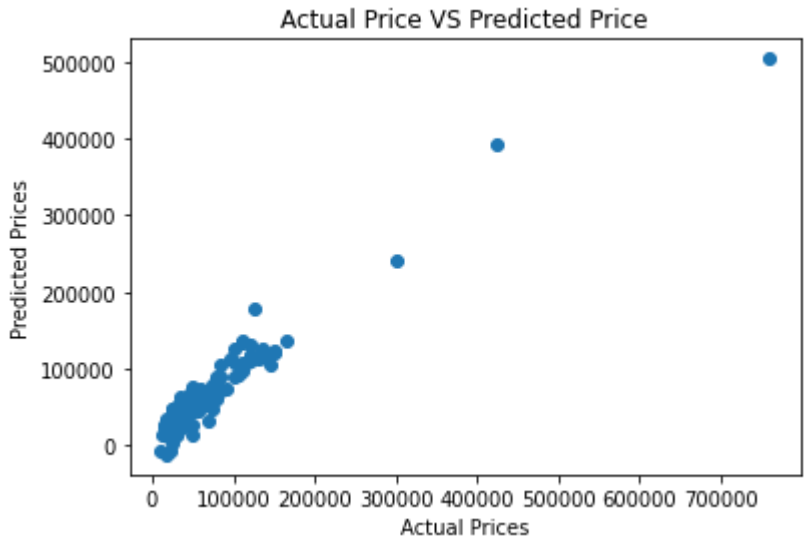
```
12225.7370104107
```

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

Saving...

Get 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()
```



Get Future Predictions

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

```
df_new
```

	Brand	Model	Selling_Price	Year	Seller_Type	Owner	KM_Driven	Ex_Showroom_Price
	Honda							

```
df_new.shape
```

```
(1, 8)
```

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

Saving...

```
y_pred_new
```

```
array([22923.26533437])
```

✓ 0s completed at 9:42 PM

Saving...