In [1]:

```
import pandas as pd
```

In [2]:

```
import numpy as np
```

In [3]:

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

In [4]:

```
df.head()
```

Out[4]:

	Brand	Model	Selling_Price	Year	Seller_Type	Owner	KM_Driven	Ex_Showroom_Price
0	TVS	TVS XL 100	30000	2017	Individual	1st owner	8000	30490.0
1	Bajaj	Bajaj ct 100	18000	2017	Individual	1st owner	35000	32000.0
2	Yo	Yo Style	20000	2011	Individual	1st owner	10000	37675.0
3	Bajaj	Bajaj Discover 100	25000	2010	Individual	1st owner	43000	42859.0
4	Bajaj	Bajaj Discover 100	24999	2012	Individual	2nd owner	35000	42859.0

In [5]:

```
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
```

In [6]:

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

In [7]:

```
df.describe()
```

Out[7]:

	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

In [8]:

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

Out[8]:

```
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
```

In [9]:

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

Out[9]:

```
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
```

In [10]:

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

Out[10]:

```
Selling_Price
50000            49
25000            46
45000            43
40000            38
35000            38
                 ..
76000             1
77000             1
78500             1
86000             1
760000            1
Length: 99, dtype: int64
```

In [11]:

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

Out[11]:

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

In [12]:

```
df.columns
```

Out[12]:

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

In [13]:

```
df.shape
```

Out[13]:

```
(626, 8)
```

In [14]:

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

In [15]:

```
df.replace({'Owner':{'1st owner':0, '2nd owner':1, '3rd owner':2, '4th owner':3}},inplace=T
```

In [16]:

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

In [17]:

```
y.shape
```

Out[17]:

```
(626,)
```

In [18]:

```
y
```

Out[18]:

```
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
```

In [19]:

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

In [20]:

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

In [21]:

```
x.shape
```

Out[21]:

```
(626, 5)
```

In [22]:

```
x
```

Out[22]:

	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 × 5 columns

In [23]:

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

In [24]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state =252
```

In [25]:

```
x_train.shape, x_test.shape, y_train.shape,y_test.shape
```

Out[25]:

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

In [26]:

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

In [27]:

```
lr = LinearRegression()
```

In [28]:

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

Out[28]:

```
LinearRegression()
```

In [29]:

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

In [30]:

```
y_pred
```

Out[30]:

```
array([ 27210.52271467,  56340.08335169,  63471.94671996,  53627.63844777,
        55612.75744261,  53888.92259714,  33751.35275104,  60311.49501864,
       113713.05684468,  76639.49332963,  27826.73993812,  49919.83255837,
        65886.64311455,  26755.12664071,  48277.75426031, 127646.5607935 ,
        70047.1066163 ,  39350.6796366 ,  36081.0359788 ,  45360.79436347,
        48079.89470576,  44803.02464796,  55161.4402611 ,  71041.51821319,
        91689.22699173,  49301.53594633,  55988.19326256, 108171.54600298,
        32771.06897893,  25468.20072998,  17128.61806167, 179271.41130778,
        45698.99857623,  31371.09285094,  67886.5210673 ,  41492.49575813,
        56855.22238596,  47820.47003463,  74682.14053952,  24984.21822739,
        55374.00513699,  41412.36775219,  67991.6028776 ,  26553.59421833,
        89788.69870689,  45764.83633687, 133888.03770407, 106988.11382519,
        71176.40667715,  25332.25485948,  79512.43778819,  63914.38088175,
        28632.12110987,  53656.13623929,  -5396.37132904,  70377.44571172,
        33313.03576479,  53994.92478413,  67509.85836345,  59735.05378837,
        22199.83644223,  15374.18984157,  44510.7681941 ,  30279.52476755,
       108243.77037513,  19291.88958744,  53614.31297593,  59230.23269131,
        60174.21081081,  45924.63468732,  25770.81883498,  63471.36257807,
       242123.45729816,  61387.72544538,  56510.98127063,  48123.28087209,
        51668.27442009,  90279.76190494,  14827.76533572, 112437.70820513,
        35066.88027402,  30902.41069162,  31441.4892143 , 125593.75847167,
        27705.38813165, -11590.29205532,  15582.17108691,  75113.64511226,
       504085.44522347, 123545.42050119,  74770.89327692,  50747.47663252,
        44174.36182112,  25426.71561059,  30298.30524619,  47625.67836404,
        27850.37544806,  28845.23330926,  31580.38624702,  32309.63375627,
        47979.16788557,  65955.46375943,  13432.28218021,  15368.80064986,
        31973.23052416, 110353.92870547,  68181.49509131,  23143.49139801,
        53194.65732076,  34603.36376979,  56002.50967859,  62432.66994303,
       391470.77533248,   3558.29480894,  36019.18494311,  70876.34866548,
        72890.0066702 , 137596.01384367,  27620.36308875, 135789.30486862,
        39674.40366787,  58367.09244519,  42401.21202629,  61864.43795665,
        42688.89652834,  63710.34571026,  10604.39360077,  38458.8282094 ,
       112251.84744238, 115403.00577542,  13658.41734794,  36196.83359587,
        54146.22998929,  97297.85724852,  55029.68137266,  22923.2653344 ,
       104569.97029696,  41965.75852015,  38759.68546485,  28930.61369002,
        45231.66612551,  48475.43422772,  26739.72257309,  53598.65972207,
        32558.54954525,  32212.22834939,  68172.98738416,  71839.47716471,
        32003.46692215,  40652.69995967,  39935.92211841,  63444.41846201,
        44545.58187706, 120873.38389627,  60926.58683171,  62641.82167503,
        60816.4737999 ,  27098.95433577,  26803.64749625,  48956.00468622,
        62032.88118706,  26471.97495739, 104937.23068763, 132903.35788475,
        37469.20409416,  57579.12080118,  40371.00915744,  -7039.40662486,
        26485.40030073,  90782.42554161,  52153.21149322,  56453.74542448,
        80440.59425999,  31890.46870273,  49505.97985571,  24288.36959518,
        25540.47481574, 117708.26333954,  23399.66596754,  63678.40865459,
        70144.29372661,  33434.89010059,  60885.29444478,  58389.55370869,
        35118.7040347 ,  58729.45401961,  34627.95322449,  38583.46239728])
```

In [31]:

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

In [32]:

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

Out[32]:

```
554715615.5020787
```

In [33]:

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

Out[33]:

```
12225.737010391558
```

In [34]:

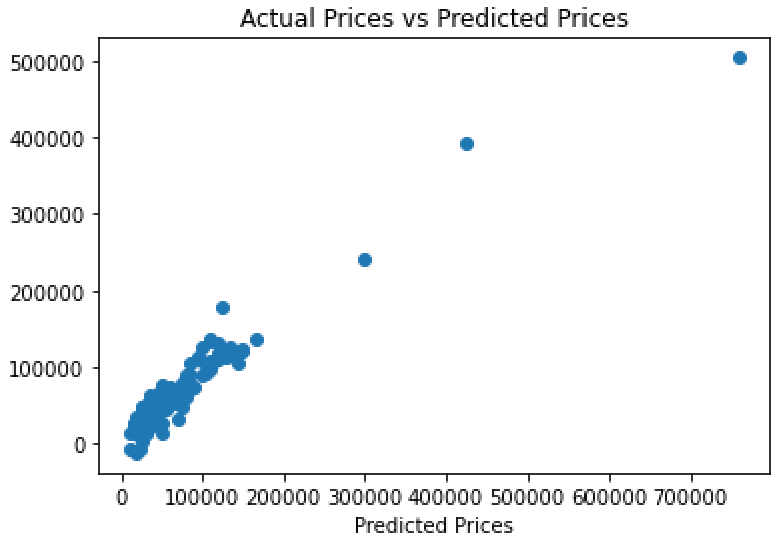
```
r2_score(y_test,y_pred)
```

Out[34]:

```
0.8810414402989845
```

In [35]:

```
import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.xlabel("Predicted Prices")
plt.title("Actual Prices vs Predicted Prices")
plt.show()
```



In [36]:

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

In [37]:

```
df_new
```

Out[37]:

	Brand	Model	Selling_Price	Year	Seller_Type	Owner	KM_Driven	Ex_Showroom_Price
523	Hero	Hero Xpulse 200	100000	2019	0	0	8500	107500.0

In [38]:

```
df_new.shape
```

Out[38]:

```
(1, 8)
```

In [39]:

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

In [40]:

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

In [41]:

```
y_pred_new
```

Out[41]:

```
array([ 3.40355519e+04,  3.46279532e+04,  1.06043936e+04,  8.79468274e+03,
        1.27922013e+04,  1.75468447e+04,  1.34322822e+04,  3.55829481e+03,
        2.65535942e+04,  2.78503754e+04,  2.76203631e+04,  4.03320613e+03,
        1.80285892e+04, -3.66011405e+02,  3.81139239e+04,  4.26888965e+04,
        3.81139239e+04, -5.39637133e+03,  1.47433984e+04,  2.88452333e+04,
        1.48180586e+04, -7.45212122e+03,  2.55404748e+04,  3.50668803e+04,
        2.56882341e+04,  2.54267156e+04,  3.02983052e+04,  2.57708188e+04,
        1.53741898e+04,  3.94017989e+04,  3.02589825e+04,  3.46033638e+04,
        2.61483050e+04,  4.93015359e+04,  3.14414892e+04,  3.18904687e+04,
        4.03658596e+04,  3.17582699e+04,  3.09024107e+04,  2.67397226e+04,
        4.01845405e+04,  4.59240388e+04,  2.74260041e+04,  1.71646942e+04,
        2.73414924e+04,  3.65196282e+04,  3.14341917e+04,  3.25585495e+04,
        1.65642377e+04,  3.20608229e+04,  3.22122283e+04,  1.34278404e+04,
        3.26389163e+04,  2.74920441e+04,  8.62507139e+03,  8.14332688e+03,
        1.38983312e+03,  3.22535207e+04,  2.28957371e+04,  2.31434914e+04,
        2.77053881e+04,  3.21158794e+04, -5.23267042e+03,  1.27396339e+04,
        2.29232653e+04, -1.04690583e+04,  8.14332688e+03,  3.70975821e+04,
       -1.25163613e+02,  2.30471425e+04,  1.83957789e+04,  4.62984958e+04,
```