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()
```

Out[5]:
```
<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	7.749559e+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[['Seller_Type']].value_counts()
```

Out[10]:
```
Seller_Type
Individual    623
Dealer          3
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=True)
```

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.shape
```

Out[20]:
```
(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 =22529)
```

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
LinearRegression()
```

In [29]:
```
y_pred = lr.predict(X_test)
```

In [30]:
```
y_pred.shape
```

Out[30]:
```
(188,)
```

In [31]:
```
y_pred
```

Out[31]:
```
array([ 5.67719381e+04,  4.98088060e+04,  6.25566811e+04,  1.34333674e+05,
        5.40893717e+04,  1.41756580e+05,  7.73823521e+03,  7.39758489e+04,
        5.36499143e+04,  1.50036510e+05,  4.56921525e+04,  4.98001925e+04,
        1.52616194e+05,  7.17329097e+04,  2.30040935e+04,  1.85105499e+04,
        5.71446601e+04,  6.80959236e+04,  4.97525222e+04,  4.02778579e+04,
        4.60919059e+04,  1.33299423e+05,  1.03939684e+05,  3.55170690e+04,
        3.07713320e+04,  1.14804752e+05,  3.47999406e+04,  4.91423672e+04,
        4.14008648e+04,  6.57115829e+04,  5.25992403e+04,  9.33297385e+04,
        2.87890029e+04,  4.41204355e+04,  1.43905924e+02,  1.41012727e+05,
        6.68944957e+04,  2.85474269e+04,  2.93878214e+04,  1.03958270e+05,
        1.75057945e+04,  6.56776887e+04,  4.61855367e+04,  5.26692796e+04,
        3.12198067e+04,  2.95174352e+04,  3.67112097e+04,  4.31846848e+04,
        6.59594596e+04,  3.82542095e+04,  4.87586422e+04,  6.33032611e+04,
        7.77595023e+04,  5.38905297e+04,  1.21356168e+05,  6.41130801e+04,
        6.13588288e+04,  2.59616164e+04,  5.19673074e+04,  4.66161649e+04,
        5.16181881e+04,  3.27446040e+04,  3.59775498e+04,  1.06199873e+05,
        6.88818412e+04,  4.42360456e+04,  3.79028532e+04,  4.00827388e+04,
        2.58267537e+04,  5.33375265e+04,  1.81324595e+05,  5.30206522e+04,
        4.01105827e+04,  8.35955057e+04,  2.51703891e+04,  1.67003368e+04,
        8.02604700e+04,  4.36585832e+04,  2.86876539e+04,  4.98278218e+04,
        3.47052060e+04,  5.83706664e+04,  5.08032963e+04,  5.40586788e+04,
        1.19906410e+05,  7.31559945e+04,  6.57184002e+04,  3.84988944e+04,
        3.44195039e+04,  1.33115353e+05,  2.73767595e+04,  3.76480869e+04,
        7.34849422e+04,  4.51536489e+04,  1.22951658e+05,  5.51852028e+04,
        5.98329441e+04,  3.39998399e+04,  1.88129595e+04,  6.81463253e+04,
        2.46283070e+04,  4.06045373e+04,  8.38463718e+04,  7.83165993e+04,
        3.26180645e+04,  4.36216912e+04,  3.80707619e+04,  2.91179318e+04,
        5.17657230e+04,  3.19431544e+04,  3.59531152e+04,  -6.71608267e+03,
        -8.68485090e+03,  3.70284572e+04,  7.77238311e+04,  1.39569112e+04,
        6.02254297e+04,  2.72446394e+04,  1.65473330e+05,  1.68260294e+04,
        6.67212870e+04,  5.97844109e+04,  2.16237912e+04,  1.19936148e+05,
        5.95406464e+04,  5.85761308e+04,  2.58430324e+04,  5.96413346e+04,
        6.43484054e+04,  6.53483552e+04,  4.98854389e+04,  9.83289885e+03,
        4.38888698e+04,  4.85790152e+04,  1.85927363e+04,  4.62719330e+04,
        1.32443162e+05,  1.23372459e+05,  9.35620244e+03,  3.13996891e+04,
        8.51423910e+03,  4.01343458e+04,  6.26886661e+04,  2.35812697e+04,
        5.90544874e+04,  5.76044641e+04,  2.83131078e+04,  4.83332328e+04,
        1.89059941e+04,  7.31218058e+04,  4.23211600e+04,  4.55349926e+04,
        4.90717337e+04,  5.35559200e+04,  9.00410051e+04,  6.54770387e+04,
        1.36348133e+05,  1.08215360e+05,  1.19966577e+05,  3.27407330e+04,
        3.15121728e+04,  4.96308504e+04,  5.67414568e+04,  4.32572023e+04,
        -1.74613576e+04,  3.27233857e+04,  4.83834154e+04,  2.14358234e+04,
        5.59986892e+04,  8.38381939e+04,  3.60163081e+04,  1.38172227e+05,
        1.83260222e+05,  3.60879518e+04,  5.10543045e+04,  5.24583471e+04,
        2.94214740e+04,  3.38041468e+04,  9.11193543e+04,  5.85034815e+04,
        4.02803633e+04,  6.56727915e+04,  6.01896042e+03,  -1.52693775e+03,
        4.78513081e+04,  4.18239168e+04,  5.03606752e+04,  9.48166435e+05])
```

In [32]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

In [33]:
```
mean_squared_error(y_test,y_pred)
```

Out[33]:
```
460921752.47205395
```

In [34]:
```
mean_absolute_error(y_test,y_pred)
```

Out[34]:
```
12297.828196871411
```

In [35]:
```
r2_score(y_test, y_pred)
```

Out[35]:
```
0.8848058354063282
```

In [36]:
```
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()
```

In [37]:
```
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()
```

In [38]:
```
df_new = df.sample(1)
df_new
```

Out[38]:

	Brand	Model	Selling_Price	Year	Seller_Type	Owner	KM_Driven	Ex_Showroom_Price
578	Bajaj	Bajaj Dominar 400 [2018]	1700000	2018	0	0	16000	163331.0

In [39]:
```
df_new.shape
```

Out[39]:
```
(1, 8)
```

In [40]:
```
X_new = df_new.drop(['Brand', 'Model', 'Selling_Price'],axis = 1)
```

In [41]:
```
y_pred_new = lr.predict(X_new)
```

In [42]:
```
y_pred_new
```

Out[42]:
```
array([131505.96645724])
```

In [ ]: