In [49]:
```
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	2014.803319	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[['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 [16]:
```
df.replace({'Owner':{'1st owner':0,'2nd owner':1,'3rd owner':2,'4th owner':3}},inplace=True)
```

In [17]:
```
#X = pd.get_dummies(X,columns =['Seller_Type0', 'Owner'],drop_first=True)
```

In [18]:
```
y = df['Selling_Price']
```

In [19]:
```
y.shape
```

Out[19]:
```
(626,)
```

In [20]:
```
y
```

Out[20]:
```
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 [21]:
```
x = df[['Year','Seller_Type','Owner','KM_Driven','Ex_Showroom_Price']]
```

In [22]:
```
#X df.drop ({'Brand','Model','Selling_Price'},axis=1)
```

In [23]:
```
x.shape
```

Out[23]:
```
(626, 5)
```

In [26]:
```
from sklearn.model_selection import train_test_split
```

In [27]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.3, random_state=72529)
```

In [28]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[28]:
```
((438, 5), (188, 5), (438,), (188,))
```

In [29]:
```
from sklearn.linear_model import LinearRegression
```

In [30]:
```
lr = LinearRegression()
```

In [32]:
```
lr.fit(x_train,y_train)
```

Out[32]:
```
LinearRegression()
```

In [33]:
```
y_pred = lr.predict(x_test)
```

In [34]:
```
y_pred.shape
```

Out[34]:
```
(188,)
```

In [35]:
```
y_pred
```

Out[35]:
```
array([138240.29654226,  35307.27629811,  41392.3146781 ,
        78589.64050907,  66330.33181893,  71450.75913796,
       128741.92385458,  31301.4907403 ,  16365.36217535,
        52645.82622187,  34162.66438303,  33108.28322892,
        20285.80469509,  22481.73317954,  18623.14544884,
        46071.59537129,  43382.2659943 ,  49240.05425507,
        52547.07580469,  63756.04007575,  49201.36726671,
        60476.56587229,  46568.1904032 ,  51727.28970532,
        57675.94515841,  52904.87730434,  22154.32221548,
        64738.47935315,  69961.82758377,  70237.03364442,
        10109.81723728,   7978.29684737,  50290.33945634,
        39002.28085522,  37636.24973153,  22214.51509801,
        25919.18719911,  49681.05425329,  54478.5448337 ,
       134286.41450486,  39465.59348847,  18856.93520069,
        29395.67326193,  52280.49539301,  54281.39627374,
        23871.16128248,  22768.36144976,  51947.79627002,
        74002.44137979,  60726.08485228,  55554.80098319,
        84302.58855953,  78761.17925924,  13940.70161622,
       116019.0995354 ,  14081.71349846,  44665.67959531,
       131436.62102645,  22829.68163124,   8188.18124434,
        98989.0543751 ,  73329.16019507,  59492.70884178,
        55000.52645284,  24443.01694916,  20675.98010482,
        55621.8738598 ,  24746.73279912,  65067.00272016,
        24599.65499752,  50617.16305576,  47910.83100397,
        49125.22125809,  54742.72180424,  42508.08197345,
        38727.07477456,  48829.02962818,  64112.04371617,
        53584.01982098,  54843.33533899,  67692.86373031,
       126519.60362767,  26501.93016865,  67856.9440369 ,
        36218.26246254,  34696.88795136,  51491.4888429 ,
        86313.65522794,  42211.37367351,  45322.89571985,
        47799.97285054,  102932.03791725,  32196.93603661,
        45716.09120145,  69711.00966411,  16931.96292963,
        20489.88095332,  28963.45154368,  44739.34616288,
        63468.42147204,  66619.70418353,  24182.69091813,
        31899.66302198,  33228.48506905,  55251.34662947,
        21012.87110226,  38759.45190052,  66393.40507015,
        54281.22548207,  52922.38019798,  64089.86871562,
       110620.02866249,  52781.43797999,  50481.02485062,
        54600.50644429,  110740.52764662,  48604.39480031,
       140992.5623721 ,  91457.62755442,   37668.69451781,
        73385.8202705 , 251938.90006593, 115841.02501263,
       136531.63426807,  28238.8208474 ,  52842.56315846,
        63314.77014161,  31188.37127546,  37092.05966479,
        78643.45979421,  50279.40061888,  38977.99790575,
        31588.58401208,  36994.47781951,  38419.49150795,
        31414.013563  ,  130271.76136431,  35704.39301389,
        54191.15197736,  51895.22928438,  31043.88123759,
        32558.17987778,  22186.09940144,  15425.81074276,
        32115.77818923,  41215.81820956,  72011.78109481,
       126282.35419302,  20390.2663526 ,  55400.39636476,
        50864.94573883,   87779.89530306,  23629.54522686,
       131453.00572154,  57428.55542153,  -2208.22086481,
        58451.70345829, 112323.27244037,  51881.06551377,
        40427.81846939,  94861.96975418,  61184.10657223,
        68346.70230126,  52331.23702063,  27752.47525724,
        71401.31807354,  36696.11481079,  36070.20279696,
        25901.3679022 ,  58724.9287498 ,  55704.85706306,
        60688.55133162,  59359.07183992,  46924.33944876,
        50465.30096557,  44875.83392819,  41411.15345077,
        59909.89024773,  51345.79156608,  34454.61844574,
        71609.40734916, 154935.58973255, 127381.82658877,
        60007.02180561,  69793.03455607,  29241.50034586,
        54255.60205173])
```

In [36]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [37]:
```
mean_squared_error(y_test,y_pred)
```

Out[37]:
```
152201375.067118
```

In [38]:
```
mean_absolute_error(y_test,y_pred)
```

Out[38]:
```
8925.457899521194
```

In [39]:
```
r2_score(y_test,y_pred)
```

Out[39]:
```
0.9003083084246579
```

In [40]:
```
import matplotlib.pyplot as plt
```

In [42]:
```
plt.scatter(y_test,y_pred)
plt.xlabel('Actual Prices')
plt.ylabel('Prediction Prices')
plt.title("Actual Prices vs Prediction Prices")
plt.show()
```

In [43]:
```
df_new = df.sample(1)
```

In [44]:
```
df_new
```

Out[44]:

	Brand	Model	Selling_Price	Year	Seller_Type	Owner	KM_Driven	Ex_Showroom_Price
253	Bajaj	Bajaj Discover 150S	40000	2014	0	0	50000	63331.0

In [45]:
```
df_new.shape
```

Out[45]:
```
(1, 8)
```

In [46]:
```
x_new = df_new.drop(['Brand','Model','Selling_Price'], axis = 1)
```

In [47]:
```
y_pred_new = lr.predict(x_new)
```

In [48]:
```
y_pred_new
```

Out[48]:
```
array([39773.67742035])
```

In [ ]:
```

```