# Lab #4: House Price Prediction using LR with Regularization

## Step1. [Import dataset]. Using Pandas, import "Ames_House_Sales_Cropped.csv" file and print properties such as head, shape, columns, dtype, info and value_counts

In [1]:

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

In [30]:

```
df = pd.read_csv('Ames_House_Sales_Cropped.csv')
```

In [31]:

```
df
```

Out[31]:

| | BldgType | CentralAir | 1stFlrSF | 2ndFlrSF | 3SsnPorch | BedroomAbvGr | BsmtFinSF1 | Bsmt |
|---|---|---|---|---|---|---|---|---|
| **0** | 1Fam | Y | 856.0 | 854.0 | 0.0 | 3 | 706.0 | |
| **1** | 1Fam | Y | 1262.0 | 0.0 | 0.0 | 3 | 978.0 | |
| **2** | 1Fam | Y | 920.0 | 866.0 | 0.0 | 3 | 486.0 | |
| **3** | 1Fam | Y | 961.0 | 756.0 | 0.0 | 3 | 216.0 | |
| **4** | 1Fam | Y | 1145.0 | 1053.0 | 0.0 | 4 | 655.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **1374** | 1Fam | Y | 953.0 | 694.0 | 0.0 | 3 | 0.0 | |
| **1375** | 1Fam | Y | 2073.0 | 0.0 | 0.0 | 3 | 790.0 | |
| **1376** | 1Fam | Y | 1188.0 | 1152.0 | 0.0 | 4 | 275.0 | |
| **1377** | 1Fam | Y | 1078.0 | 0.0 | 0.0 | 2 | 49.0 | |
| **1378** | 1Fam | Y | 1256.0 | 0.0 | 0.0 | 3 | 830.0 | |

1379 rows × 39 columns

In [5]:

```
df.head()
```

Out[5]:

| | BldgType | CentralAir | 1stFlrSF | 2ndFlrSF | 3SsnPorch | BedroomAbvGr | BsmtFinSF1 | BsmtFin |
|---|---|---|---|---|---|---|---|---|
| **0** | 1Fam | Y | 856.0 | 854.0 | 0.0 | 3 | 706.0 | |
| **1** | 1Fam | Y | 1262.0 | 0.0 | 0.0 | 3 | 978.0 | |
| **2** | 1Fam | Y | 920.0 | 866.0 | 0.0 | 3 | 486.0 | |
| **3** | 1Fam | Y | 961.0 | 756.0 | 0.0 | 3 | 216.0 | |
| **4** | 1Fam | Y | 1145.0 | 1053.0 | 0.0 | 4 | 655.0 | |

5 rows × 39 columns

In [6]:

```
df.shape
```

Out[6]:

```
(1379, 39)
```

In [7]:

```
df.columns
```

Out[7]:

```
Index(['BldgType', 'CentralAir', '1stFlrSF', '2ndFlrSF', '3SsnPorch',
       'BedroomAbvGr', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtFullBath',
       'BsmtHalfBath', 'BsmtUnfSF', 'EnclosedPorch', 'Fireplaces', 'FullBat
h',
       'GarageArea', 'GarageCars', 'GarageYrBlt', 'GrLivArea', 'HalfBath',
       'KitchenAbvGr', 'LotArea', 'LotFrontage', 'LowQualFinSF', 'MSSubClas
s',
       'MasVnrArea', 'MiscVal', 'MoSold', 'OpenPorchSF', 'OverallCond',
       'OverallQual', 'PoolArea', 'ScreenPorch', 'TotRmsAbvGrd', 'TotalBsmtS
F',
       'WoodDeckSF', 'YearBuilt', 'YearRemodAdd', 'YrSold', 'SalePrice'],
      dtype='object')
```

In [10]:

```
df.dtypes
```

Out[10]:

```
BldgType          object
CentralAir        object
1stFlrSF         float64
2ndFlrSF         float64
3SsnPorch        float64
BedroomAbvGr       int64
BsmtFinSF1       float64
BsmtFinSF2       float64
BsmtFullBath       int64
BsmtHalfBath       int64
BsmtUnfSF        float64
EnclosedPorch    float64
Fireplaces         int64
FullBath           int64
GarageArea       float64
GarageCars         int64
GarageYrBlt      float64
GrLivArea        float64
HalfBath           int64
KitchenAbvGr       int64
LotArea          float64
LotFrontage      float64
LowQualFinSF     float64
MSSubClass         int64
MasVnrArea       float64
MiscVal          float64
MoSold             int64
OpenPorchSF      float64
OverallCond        int64
OverallQual        int64
PoolArea         float64
ScreenPorch      float64
TotRmsAbvGrd       int64
TotalBsmtSF      float64
WoodDeckSF       float64
YearBuilt          int64
YearRemodAdd       int64
YrSold             int64
SalePrice        float64
dtype: object
```

In [12]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1379 entries, 0 to 1378
Data columns (total 39 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   BldgType       1379 non-null   object 
 1   CentralAir     1379 non-null   object 
 2   1stFlrSF       1379 non-null   float64
 3   2ndFlrSF       1379 non-null   float64
 4   3SsnPorch      1379 non-null   float64
 5   BedroomAbvGr   1379 non-null   int64  
 6   BsmtFinSF1     1379 non-null   float64
 7   BsmtFinSF2     1379 non-null   float64
 8   BsmtFullBath   1379 non-null   int64  
 9   BsmtHalfBath   1379 non-null   int64  
 10  BsmtUnfSF      1379 non-null   float64
 11  EnclosedPorch  1379 non-null   float64
 12  Fireplaces     1379 non-null   int64  
 13  FullBath       1379 non-null   int64  
 14  GarageArea     1379 non-null   float64
 15  GarageCars     1379 non-null   int64  
 16  GarageYrBlt    1379 non-null   float64
 17  GrLivArea      1379 non-null   float64
 18  HalfBath       1379 non-null   int64  
 19  KitchenAbvGr   1379 non-null   int64  
 20  LotArea        1379 non-null   float64
 21  LotFrontage    1379 non-null   float64
 22  LowQualFinSF   1379 non-null   float64
 23  MSSubClass     1379 non-null   int64  
 24  MasVnrArea     1379 non-null   float64
 25  MiscVal        1379 non-null   float64
 26  MoSold         1379 non-null   int64  
 27  OpenPorchSF    1379 non-null   float64
 28  OverallCond    1379 non-null   int64  
 29  OverallQual    1379 non-null   int64  
 30  PoolArea       1379 non-null   float64
 31  ScreenPorch    1379 non-null   float64
 32  TotRmsAbvGrd   1379 non-null   int64  
 33  TotalBsmtSF    1379 non-null   float64
 34  WoodDeckSF     1379 non-null   float64
 35  YearBuilt      1379 non-null   int64  
 36  YearRemodAdd   1379 non-null   int64  
 37  YrSold         1379 non-null   int64  
 38  SalePrice      1379 non-null   float64
dtypes: float64(21), int64(16), object(2)
memory usage: 420.3+ KB
```

In [13]:

```
df.value_counts
```

Out[13]:

```
<bound method DataFrame.value_counts of      BldgType CentralAir  1stFlrSF
2ndFlrSF  3SsnPorch  BedroomAbvGr  \
0        1Fam          Y     856.0     854.0        0.0            3
1        1Fam          Y    1262.0       0.0        0.0            3
2        1Fam          Y     920.0     866.0        0.0            3
3        1Fam          Y     961.0     756.0        0.0            3
4        1Fam          Y    1145.0    1053.0        0.0            4
...       ...        ...       ...       ...        ...          ...
1374     1Fam          Y     953.0     694.0        0.0            3
1375     1Fam          Y    2073.0       0.0        0.0            3
1376     1Fam          Y    1188.0    1152.0        0.0            4
1377     1Fam          Y    1078.0       0.0        0.0            2
1378     1Fam          Y    1256.0       0.0        0.0            3

      BsmtFinSF1  BsmtFinSF2  BsmtFullBath  BsmtHalfBath  ...  OverallQual
\
0          706.0         0.0             1             0  ...            7
1          978.0         0.0             0             1  ...            6
2          486.0         0.0             1             0  ...            7
3          216.0         0.0             1             0  ...            7
4          655.0         0.0             1             0  ...            8
...          ...         ...           ...           ...  ...          ...
1374         0.0         0.0             0             0  ...            6
1375       790.0       163.0             1             0  ...            6
1376       275.0         0.0             0             0  ...            7
1377        49.0      1029.0             1             0  ...            5
1378       830.0       290.0             1             0  ...            5

      PoolArea  ScreenPorch  TotRmsAbvGrd  TotalBsmtSF  WoodDeckSF  YearBuil
t  \
0          0.0          0.0             8        856.0         0.0       200
3
1          0.0          0.0             6       1262.0       298.0       197
6
2          0.0          0.0             6        920.0         0.0       200
1
3          0.0          0.0             7        756.0         0.0       191
5
4          0.0          0.0             9       1145.0       192.0       200
0
...        ...          ...           ...          ...         ...
...
1374       0.0          0.0             7        953.0         0.0       199
9
1375       0.0          0.0             7       1542.0       349.0       197
8
1376       0.0          0.0             9       1152.0         0.0       194
1
1377       0.0          0.0             5       1078.0       366.0       195
0
1378       0.0          0.0             6       1256.0       736.0       196
5

      YearRemodAdd  YrSold  SalePrice
0             2003    2008   208500.0
```

```
1             1976    2007   181500.0
2             2002    2008   223500.0
3             1970    2006   140000.0
4             2000    2008   250000.0
...            ...     ...        ...
1374          2000    2007   175000.0
1375          1988    2010   210000.0
1376          2006    2010   266500.0
1377          1996    2010   142125.0
1378          1965    2008   147500.0

[1379 rows x 39 columns]>
```

**Step2. [Predict Sale Price without Categorical features]**

**1.Drop both categorical features – BldgType and CentralAir (USE drop() and pop() methods)**

**2.Prepare X matrix (36 feature columns) and y vector (ie., SalePrice column)**

**3.Split dataset for training and testing as X_train, X_test, y_train, y_test (use 25% test size).**

**4.Create LinearRegression model, fit on training set and predict on test set**

**5.Compute Mean Squared Error (MSE) on actual values and predicted values (you will get output as 1474827326.0).**

In [14]:

```
df.pop('CentralAir')
```

Out[14]:

```
0       Y
1       Y
2       Y
3       Y
4       Y
       ..
1374    Y
1375    Y
1376    Y
1377    Y
1378    Y
Name: CentralAir, Length: 1379, dtype: object
```

In [16]:

```
ndf=df.drop(['BldgType'],axis=1)
ndf
```

Out[16]:

| | 1stFlrSF | 2ndFlrSF | 3SsnPorch | BedroomAbvGr | BsmtFinSF1 | BsmtFinSF2 | BsmtFullBath |
|---|---|---|---|---|---|---|---|
| **0** | 856.0 | 854.0 | 0.0 | 3 | 706.0 | 0.0 | 1 |
| **1** | 1262.0 | 0.0 | 0.0 | 3 | 978.0 | 0.0 | 0 |
| **2** | 920.0 | 866.0 | 0.0 | 3 | 486.0 | 0.0 | 1 |
| **3** | 961.0 | 756.0 | 0.0 | 3 | 216.0 | 0.0 | 1 |
| **4** | 1145.0 | 1053.0 | 0.0 | 4 | 655.0 | 0.0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **1374** | 953.0 | 694.0 | 0.0 | 3 | 0.0 | 0.0 | 0 |
| **1375** | 2073.0 | 0.0 | 0.0 | 3 | 790.0 | 163.0 | 1 |
| **1376** | 1188.0 | 1152.0 | 0.0 | 4 | 275.0 | 0.0 | 0 |
| **1377** | 1078.0 | 0.0 | 0.0 | 2 | 49.0 | 1029.0 | 1 |
| **1378** | 1256.0 | 0.0 | 0.0 | 3 | 830.0 | 290.0 | 1 |

1379 rows × 37 columns

In [18]:

```
y=ndf['SalePrice']
y
```

Out[18]:

```
0       208500.0
1       181500.0
2       223500.0
3       140000.0
4       250000.0
          ...
1374    175000.0
1375    210000.0
1376    266500.0
1377    142125.0
1378    147500.0
Name: SalePrice, Length: 1379, dtype: float64
```

In [19]:

```
col=['1stFlrSF', '2ndFlrSF', '3SsnPorch','BedroomAbvGr','BsmtFinSF1','BsmtFinSF2','BsmtFull
X=ndf[col]
X
```

Out[19]:

| | 1stFlrSF | 2ndFlrSF | 3SsnPorch | BedroomAbvGr | BsmtFinSF1 | BsmtFinSF2 | BsmtFullBath |
|---|---|---|---|---|---|---|---|
| **0** | 856.0 | 854.0 | 0.0 | 3 | 706.0 | 0.0 | 1 |
| **1** | 1262.0 | 0.0 | 0.0 | 3 | 978.0 | 0.0 | 0 |
| **2** | 920.0 | 866.0 | 0.0 | 3 | 486.0 | 0.0 | 1 |
| **3** | 961.0 | 756.0 | 0.0 | 3 | 216.0 | 0.0 | 1 |
| **4** | 1145.0 | 1053.0 | 0.0 | 4 | 655.0 | 0.0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **1374** | 953.0 | 694.0 | 0.0 | 3 | 0.0 | 0.0 | 0 |
| **1375** | 2073.0 | 0.0 | 0.0 | 3 | 790.0 | 163.0 | 1 |
| **1376** | 1188.0 | 1152.0 | 0.0 | 4 | 275.0 | 0.0 | 0 |
| **1377** | 1078.0 | 0.0 | 0.0 | 2 | 49.0 | 1029.0 | 1 |
| **1378** | 1256.0 | 0.0 | 0.0 | 3 | 830.0 | 290.0 | 1 |

1379 rows × 36 columns

In [20]:

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

In [21]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75,test_size=0.25)
```

In [22]:

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

In [23]:

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

Out[23]:

```
LinearRegression()
```

In [24]:

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

Out[24]:

```
array([278935.46760284, 184809.97301153, 182520.43414718,  76483.10947771,
       150887.34597605, 131258.56301645, 148585.24002343, 219097.29021401,
       111773.72913835, 154881.76229039, 256457.65115608, 318666.12035276,
       159561.7926296 , 169107.86682563, 284670.76042656, 181162.50230694,
       227278.48539688, 114165.89801476, 294931.91089851, 293854.41405179,
       111435.02196622, 382585.58559473, 165973.92154629, 169983.25590793,
       233764.73014481, 256793.62698861, 141571.18375276, 236487.34960355,
       110407.21426591, 256140.69565805, 116395.91328469, 246533.22979261,
       297638.63909922,  67305.0554531 ,  99905.83719632, 127802.47702706,
       164970.2581652 , 107724.12361544, 295939.4760221 , 191015.64411747,
       118704.79917791, 123437.69393744, 174063.85709086, 289498.87116956,
       224265.54740821, 324401.30417851, 255494.61362604, 239541.04302594,
        55786.42143874, 147705.02545961, 108706.2471813 , 179216.94913311,
       176248.36836102,  94056.17368155, 154939.16090248, 155810.5399734 ,
       121532.36318911, 183547.46384496, 189454.96089569,  89208.20961407,
       352481.06371935, 179664.02452966, 123126.97142585, 213608.94529321,
       175539.14858387, 143633.83226572, 253326.60653631, 117216.66854023,
       203411.64478104, 194715.49377294, 229472.71664896, 115651.09160892,
       178241.14827662, 236505.51781256, 219307.57126792, 204502.34677502,
       280994.60010451,  80073.90667113, 181434.71323535, 345032.82419352,
       236500.06566161, 212514.13302149, 192407.95486381, 142416.55271327,
       259468.7204445 , 120750.14329522, 226366.87209352,  96780.09952695,
       127163.17111758, 273574.37672692, 108517.67103149, 256173.98073332,
       228790.59367182,  88910.50049467, 107165.22214306, 217146.62770519,
        58964.35975604, 105063.33080949, 320544.61249786,  86138.74731506,
       116839.9388387 , 105712.01945374,  88230.90550046, 107652.68223998,
       287090.16939842, 229465.77535541, 158494.22780582, 204864.27385771,
       239609.37680689, 110442.75382776, 122941.92801304, 196560.69862189,
       281001.20763735, 220768.98319365, 175182.06264764, 177842.14883526,
       144528.98007098,  89249.33171787, 244389.89418398, 216270.93679401,
       248220.02700825, 138666.51927131, 154560.04755823, 221112.55156718,
       188362.59021471, 307203.62609492, 189581.32178303, 183577.46991967,
       225368.77356014, 174519.66453824, 101554.978302  , 293361.03611142,
       154012.24355422, 215330.16029933,  61812.07114214, 179234.77485403,
       225272.72744757, 177373.59763724, 151409.50180686, 111112.42636732,
       199536.361842  , 156681.59243024, 251611.1346998 , 134180.23992892,
       358140.1693967 , 154567.52731523, 204091.40490632, 121956.1236706 ,
       125826.70580346,  84907.0843445 , 196986.38415147,  85904.43755063,
       150656.93797411, 118650.29179834, 195818.871078  , 175546.47631959,
       170931.96314849, 208790.97470297,  67379.38866588, 187151.29277591,
       186719.96869974, 134660.74721073, 105614.18483508,  79513.26298969,
       124566.70915102, 122550.68752771, 206222.27957192, 207601.22965933,
       146159.01052126, 152363.80695361, 145495.0606997 , 173907.24721372,
       152747.5053473 ,  49317.18217061, 213494.73086963,  44082.42523023,
       225190.28027351, 258281.36606756, 311000.95634462, 106095.4649713 ,
       271811.12874892, 159944.11359786, 226011.83194734, 174465.73073269,
       123728.0160802 , 181138.90667533, 111038.64757558,  89889.85094022,
       123564.48045469, 250749.37890992, 226613.92673256, 210434.24370752,
       207660.39092568, 292946.37527581, 334214.98921072, 133817.05041267,
       111125.04462064, 139295.3398275 , 102650.43024674, 194340.182154  ,
       122023.34009653, 276061.24429208, 269946.5997433 , 196227.72886623,
       264731.11726338,  89901.09609902, 304188.42231002, 375521.98855508,
       111007.44643434, 350583.50095325, 342750.39394037, 110615.57193354,
       145956.36088972, 172249.53317499, 232427.35295181, 152857.99881604,
```

```
       119692.98800232, 116730.19848642, 204857.36322241, 244716.63533763,
       121656.14394599,  80838.88733147, 196484.62439457, 143926.78270525,
       106731.16943813, 194536.2272504 , 158474.12778809, 222614.21867115,
       101549.90251678, 147893.15229663, 320820.25505894,  54731.41824227,
       174055.67716066, 107403.96861738,  20317.15304097, 197697.74888904,
        87979.6495762 , 245044.00505482, 137239.8607613 , 178076.30627442,
       202934.38670694, 208592.35841259, 182895.66157271, 173665.02493717,
       187506.78653531, 191522.33185992, 195263.35443505, 262094.61033262,
       161424.46981284, 156789.32222922, 103129.22065772, 219923.35629208,
       159162.70875961, 259900.44388485, 192866.20003135, 312664.28514545,
       188000.48911174, 183052.42071648, 118131.16319897, 105606.97184016,
       174562.14609183,  98671.51317334, 110028.85776064, 247856.70997153,
        96614.73642864, 204370.74006904, 199319.43354319, 276169.70745246,
       137136.46481863, 189550.95222814, 127295.63955043, 258359.51237967,
       127024.88223935, 155902.77325132, 144286.75347553, 332376.7892343 ,
       243087.52260861, 210040.44140568, 114455.28504463, 202191.37155523,
       339030.87673504,  90609.78663929, 305497.84068171, 184070.6180781 ,
       287059.69348526,  79610.54167769, 294912.04178084,  66993.51568241,
       200053.34479378, 122697.83324424, 178671.12407925, 124541.03500813,
       256665.37732096, 233212.15027303,  94485.1448137 , 177833.98011138,
       138853.91780848, 183255.58095821, 154441.37726845, 152409.41675422,
       191323.66654094, 251395.00733515, 308279.40016699, 141436.09277925,
       270164.76638018, 221497.30954601, 147840.06207748, 295243.09480318,
       224575.0239214 , 367455.91468599, 230812.55543866, 105480.92918135,
       120875.02893366, 153886.45181622, 123686.71604039, 344316.10144485,
       223039.5741509 , 117427.00403197, 388531.26435691, 255607.40556548,
       244320.6800141 , 273483.63005768, 222328.99257133, 308105.45493609,
       283690.0908143 , 226343.71533342, 136431.10961914, 150610.65617637,
        44036.2995959 , 153481.99615241, 169535.67163367, 187240.21625236,
       291409.819297  ,  87683.66518659, 232799.80069605, 160632.59383701,
       138829.37808239, 166849.87559837,  95769.8914938 , 219251.88582997,
       136377.60902172, 147907.69764383, 294279.56152334, 194101.09245019,
       127886.62776903])
```

In [25]:

```
from sklearn.metrics import mean_squared_error
```

In [26]:

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

Out[26]:

```
930012256.3673348
```

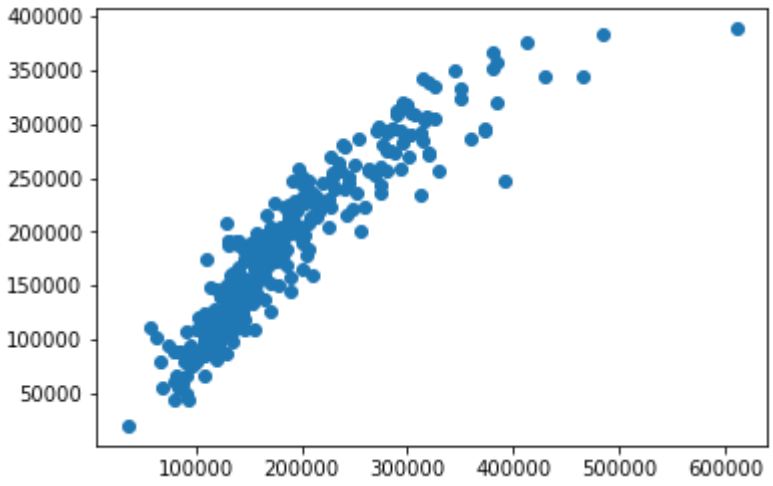
**Step3. [Create Scatter Plot]. Plot Scatterplot between y_test and y_pred.**

In [27]:

```
plt.scatter(y_test,y_pred)
```

Out[27]:

```
<matplotlib.collections.PathCollection at 0x3e09ca1130>
```




**Step4. [Encode Categorical columns]. Using get_dummies() method, perform one hot encoding on the two categorical columns, BldgType and CentralAir.Now, you will get 5 columns for BldgType variable and 2 columns for CentralAir column. So, now you have 43 independent variables and 1 dependent variable**

In [32]:

```
encode_df=pd.get_dummies(df, columns=["CentralAir","BldgType"])
encode_df.head()
```

Out[32]:

| | 1stFlrSF | 2ndFlrSF | 3SsnPorch | BedroomAbvGr | BsmtFinSF1 | BsmtFinSF2 | BsmtFullBath | Bs |
|---|---|---|---|---|---|---|---|---|
| **0** | 856.0 | 854.0 | 0.0 | 3 | 706.0 | 0.0 | 1 | |
| **1** | 1262.0 | 0.0 | 0.0 | 3 | 978.0 | 0.0 | 0 | |
| **2** | 920.0 | 866.0 | 0.0 | 3 | 486.0 | 0.0 | 1 | |
| **3** | 961.0 | 756.0 | 0.0 | 3 | 216.0 | 0.0 | 1 | |
| **4** | 1145.0 | 1053.0 | 0.0 | 4 | 655.0 | 0.0 | 1 | |

5 rows × 44 columns

In [34]:

```
encode_df.shape
```

Out[34]:

```
(1379, 44)
```

In [35]:

```
encode_df.columns
```

Out[35]:

```
Index(['1stFlrSF', '2ndFlrSF', '3SsnPorch', 'BedroomAbvGr', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtFullBath', 'BsmtHalfBath', 'BsmtUnfSF',
       'EnclosedPorch', 'Fireplaces', 'FullBath', 'GarageArea', 'GarageCar
s',
       'GarageYrBlt', 'GrLivArea', 'HalfBath', 'KitchenAbvGr', 'LotArea',
       'LotFrontage', 'LowQualFinSF', 'MSSubClass', 'MasVnrArea', 'MiscVal',
       'MoSold', 'OpenPorchSF', 'OverallCond', 'OverallQual', 'PoolArea',
       'ScreenPorch', 'TotRmsAbvGrd', 'TotalBsmtSF', 'WoodDeckSF', 'YearBuil
t',
       'YearRemodAdd', 'YrSold', 'SalePrice', 'CentralAir_N', 'CentralAir_
Y',
       'BldgType_1Fam', 'BldgType_2fmCon', 'BldgType_Duplex', 'BldgType_Twnh
s',
       'BldgType_TwnhsE'],
      dtype='object')
```

**Step5. [Predict Sale Price with Categorical features]**

In [37]:

```
en_y=encode_df['SalePrice']
en_y
```

Out[37]:

```
0       208500.0
1       181500.0
2       223500.0
3       140000.0
4       250000.0
          ...
1374    175000.0
1375    210000.0
1376    266500.0
1377    142125.0
1378    147500.0
Name: SalePrice, Length: 1379, dtype: float64
```

In [38]:

```
ecol=['1stFlrSF', '2ndFlrSF', '3SsnPorch', 'BedroomAbvGr', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtFullBath', 'BsmtHalfBath', 'BsmtUnfSF',
       'EnclosedPorch', 'Fireplaces', 'FullBath', 'GarageArea', 'GarageCars',
       'GarageYrBlt', 'GrLivArea', 'HalfBath', 'KitchenAbvGr', 'LotArea',
       'LotFrontage', 'LowQualFinSF', 'MSSubClass', 'MasVnrArea', 'MiscVal',
       'MoSold', 'OpenPorchSF', 'OverallCond', 'OverallQual', 'PoolArea',
       'ScreenPorch', 'TotRmsAbvGrd', 'TotalBsmtSF', 'WoodDeckSF', 'YearBuilt',
       'YearRemodAdd', 'YrSold', 'CentralAir_N', 'CentralAir_Y',
       'BldgType_1Fam', 'BldgType_2fmCon', 'BldgType_Duplex', 'BldgType_Twnhs',
       'BldgType_TwnhsE']
en_X=encode_df[ecol]
en_X
```

Out[38]:

| | 1stFlrSF | 2ndFlrSF | 3SsnPorch | BedroomAbvGr | BsmtFinSF1 | BsmtFinSF2 | BsmtFullBath |
|---|---|---|---|---|---|---|---|
| **0** | 856.0 | 854.0 | 0.0 | 3 | 706.0 | 0.0 | 1 |
| **1** | 1262.0 | 0.0 | 0.0 | 3 | 978.0 | 0.0 | 0 |
| **2** | 920.0 | 866.0 | 0.0 | 3 | 486.0 | 0.0 | 1 |
| **3** | 961.0 | 756.0 | 0.0 | 3 | 216.0 | 0.0 | 1 |
| **4** | 1145.0 | 1053.0 | 0.0 | 4 | 655.0 | 0.0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **1374** | 953.0 | 694.0 | 0.0 | 3 | 0.0 | 0.0 | 0 |
| **1375** | 2073.0 | 0.0 | 0.0 | 3 | 790.0 | 163.0 | 1 |
| **1376** | 1188.0 | 1152.0 | 0.0 | 4 | 275.0 | 0.0 | 0 |
| **1377** | 1078.0 | 0.0 | 0.0 | 2 | 49.0 | 1029.0 | 1 |
| **1378** | 1256.0 | 0.0 | 0.0 | 3 | 830.0 | 290.0 | 1 |

1379 rows × 43 columns

In [42]:

```
en_X_train, en_X_test, en_y_train, en_y_test = train_test_split(en_X,en_y, train_size=0.75,
```

In [44]:

```
model1 = LinearRegression()
model1.fit(en_X_train,en_y_train)
```

Out[44]:

```
LinearRegression()
```

In [46]:

```
en_y_pred=model1.predict(en_X_test)
en_y_pred
```

Out[46]:

```
array([164570.2289465 ,  85752.90331142, 198303.44446022, 122475.14440184,
       175345.78182408, 242830.9535424 , 327131.8650892 , 208850.75498362,
       182132.06296091,  63457.73994568, 154016.77303419, 271424.75748695,
       196168.36447424, 231896.15634764,  99949.01255692, 199586.65283043,
       294455.29422805, 237034.98692393, 156399.80969278, 143322.19881947,
        87317.2057708 , 459342.96310705, 180678.56647699,  80595.77914091,
       150473.52639588, 163119.60771014,  87932.58954817, 298702.16262193,
       139136.14996624, 122356.71597519, 253724.88904318, 124943.48205879,
       190378.91666766, 144311.69637699, 138626.02445074, 108390.1027098 ,
       233825.57182049, 321069.91502648, 152596.17234961, 126562.81992625,
       178895.51749365, 174107.6608674 , 192718.60711014,  84271.1131178 ,
       190926.80720668, 351883.95006538, 237798.33729765, 194988.74848599,
        67864.93144598, 148773.78149566, 141287.95243988, 123734.33873651,
       217729.45043704, 264523.39376933, 284496.30996522, 128633.42861504,
       317463.52519012, 229500.59769751, 216913.34511063, 307023.56295031,
       358046.85973153,  88894.0337679 , 213448.6569423 , 151371.96447342,
       238996.23694776, 206439.35207547, 204300.08235509, 220515.23746768,
       179526.34220759,  97765.62725349, 133030.81385702, 121228.16797878,
       214547.60810556, 163777.79551537, 164400.69495282, 120648.22075953,
       200289.83660161, 530827.98834513, 181721.08012888, 139279.91703074,
       211403.97993066,  88370.23134691,  56223.93985288, 101568.38551049,
        42264.51151882, 269839.9568285 , 219235.93693798, 215559.95896189,
       126180.17980655,  94017.76423989, 186571.38998992, 239285.74298627,
       141136.12202831, 100748.65571204, 234996.84810377, 173317.89471206,
       181170.946485  , 249499.12608691,  59117.53960148, 229825.43138893,
       119229.96389029, 205049.89031161, 289237.16371655, 376713.79583967,
       235314.99706438, 181872.51623   , 131801.04010263,  81714.62196234,
       210450.63983055, 196657.14311654, 116418.2029669 , 157203.17235198,
       141407.83746433, 278946.26986854, 120653.05708992, 147431.71254596,
       120051.95320744, 357597.60225624, 147189.71484067, 143672.2215596 ,
        61735.64050934, 102990.92664158, 343747.05725358, 309335.75077198,
        95077.1881864 , 193859.12199213, 115282.82098695, 164363.93263394,
       299418.50596946, 160799.72506617, 237317.14924853, 150426.28776088,
       141741.9803877 ,  80971.65674589, 116287.05395   , 199189.16365547,
       221398.20911546, 164624.5967068 , 185917.00418407, 232085.78361981,
       161510.37140786, 104607.19634645, 200311.61620566, 193889.93887375,
       360335.63024868, 246880.79734329, 205218.08855951, 149324.66818697,
       219012.52290642, 167950.75652282, 145145.31856183, 213907.37040049,
       166208.00694456, 177565.36744714, 213638.54963519, 128934.89419079,
       127262.06873791, 125732.53616288, 208350.8422771 , 187367.98102733,
       151979.80533747,  98238.80563602, 216376.11908759, 172434.74819286,
       219380.06611483, 169934.88709872, 119439.25910028,  94943.44404374,
       194718.96343604, 119048.86428917, 223908.48489293, 371007.58160088,
       131761.77339371, 202862.19185752, 167584.64261554, 164880.4508151 ,
       135912.95156555, 204349.79412468, 191707.2384559 , 166312.47307931,
       248073.37082883, 158225.96622289, 208273.00886696, 206677.62501901,
       145066.05374636, 188972.2930174 , 112066.39047524,  60843.20451136,
       233862.34059841, 227048.34860413, 236280.16982428, 202861.74018479,
       256347.47173762, 163072.37056444, 208215.47237552, 158121.8268503 ,
       138950.11023881, 131493.60318191, 135309.51482775, 153401.6418395 ,
        87988.96611115, 164203.71884155,  77495.00725865, 190624.2880739 ,
       169614.69937832, 214544.41436752, 158770.76775524, 172081.73067651,
        72404.36280403, 134968.67610532, 121233.70422819, 106284.16747329,
```

```
178255.45274396, 111941.93036186, 214003.21483718, 118547.46238145,
212332.22284787, 167744.85468928, 106327.02745505, 260835.23902291,
207737.0688117 , 240112.93783816, 162717.39329454,  74013.00523018,
214061.17644297, 201153.36902766, 205434.63928939, 204243.08249187,
138793.30501713, 140952.80163748, 191512.24334499, 152215.21020833,
324085.39591175, 304774.23081278, 190512.0628138 , 148811.52086832,
183278.8260974 , 190898.89420528,  91427.2383567 , 240968.26030391,
185624.17527773, 109241.47600847, 120318.3714745 , 180293.28625808,
185887.05318141, 147999.96663853, 203640.95270351, 207262.65897148,
146553.60081434, 225114.56488878, 195969.54558455, 135946.68422957,
 98360.86043366, 145093.48293363, 187527.90880649, 112485.22475333,
162337.18285397, 137396.86809542, 107782.38128734, 200885.0785942 ,
186191.74170784, 246205.62704462, 236007.79504838, 109664.72894365,
 66069.691295  , 271856.0319398 , 252559.63167735, 131192.23890725,
203706.62006734, 112676.08277595, 134454.5979042 , 122227.56150572,
225246.88731473, 207800.75783298, 276095.60530002, 134212.74190231,
158840.30722338, 161414.7159675 , 247093.81102185, 125407.70093713,
140731.92957038, 285092.65610423, 122166.41527203, 265957.16664687,
141846.17715973, 215257.29075487, 133644.67328412, 150444.80270245,
138159.96247511, 370790.96568457, 190143.1932688 , 138562.74477412,
157486.7882223 , 206893.81926977, 193463.29137986, 162321.36289893,
247387.66849631, 176097.83285155, 319862.93484745, 108892.79206792,
157281.06435892, 114875.28570174, 150046.14802086, 224016.84426732,
182743.95014861, 184259.86747379, 216961.78153856, 217129.89206972,
150042.28914975, 183894.29966913, 241653.96996239, 185373.50085041,
142737.94707988, 213872.65242864, 227772.66374861, 371458.8533591 ,
159852.55997952, 105446.77692945,  74753.49864747, 218084.73229221,
115766.14039221, 124870.49571699, 158277.63592713,  42046.97387378,
298295.97719517, 113869.82743718, 311823.56766763, 150483.58747027,
113422.79568058, 110758.54387485, 281329.77955246, 168222.64224303,
186504.6443444 , 118499.73767561, 118439.53660929, 159798.94608422,
285467.65714161, 184380.8672137 , 429972.49494309, 157974.11370633,
198929.95549095, 229184.4784674 , 339430.23265359, 188218.7187889 ,
155268.47972511])
```

In [47]:

```
mean_squared_error(en_y_test,en_y_pred)
```

Out[47]:

```
1603931409.178651
```

**Step6.[Normalize using StandardScaler and Predict Sale Price]**

In [48]:

```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
```

In [49]:

```
s= scale.fit_transform(en_X_train)
s
```

Out[49]:

```
array([[-0.75507716,  1.18098615, -0.10770338, ..., -0.17580466,
        -0.16683226, -0.28761522],
       [ 0.84436732, -0.81547136, -0.10770338, ..., -0.17580466,
        -0.16683226,  3.4768675 ],
       [ 4.34791239, -0.81547136, -0.10770338, ..., -0.17580466,
        -0.16683226, -0.28761522],
       ...,
       [-0.77030997, -0.81547136, -0.10770338, ..., -0.17580466,
        -0.16683226, -0.28761522],
       [ 2.26101815, -0.81547136, -0.10770338, ..., -0.17580466,
        -0.16683226, -0.28761522],
       [ 0.72758249, -0.81547136, -0.10770338, ..., -0.17580466,
        -0.16683226, -0.28761522]])
```

In [50]:

```
s1 = scale.transform (en_X_test)
s1
```

Out[50]:

```
array([[ 1.37751549, -0.81547136, -0.10770338, ...,  5.68813139,
        -0.16683226, -0.28761522],
       [ 0.24267154,  0.23565892, -0.10770338, ..., -0.17580466,
        -0.16683226, -0.28761522],
       [-0.98356923,  1.0544825 , -0.10770338, ..., -0.17580466,
        -0.16683226, -0.28761522],
       ...,
       [ 1.68724918, -0.81547136, -0.10770338, ..., -0.17580466,
        -0.16683226, -0.28761522],
       [-0.83124118,  1.15798549, -0.10770338, ..., -0.17580466,
        -0.16683226, -0.28761522],
       [ 1.49937792, -0.81547136, -0.10770338, ...,  5.68813139,
        -0.16683226, -0.28761522]])
```

In [51]:

```
model2 = LinearRegression()
model2.fit(s,en_y_train)
```

Out[51]:

```
LinearRegression()
```

In [52]:

```
se_y_pred = model2.predict(s1)
se_y_pred
```

Out[52]:

```
array([164570.22894655,  85752.90331148, 198303.4444603 , 122475.14440177,
       175345.78182411, 242830.95354235, 327131.86508911, 208850.75498374,
       182132.06296088,  63457.73994563, 154016.77303418, 271424.75748691,
       196168.36447425, 231896.15634765,  99949.01255693, 199586.65283044,
       294455.29422812, 237034.98692395, 156399.80969275, 143322.19881946,
        87317.20577081, 459342.96310711, 180678.56647711,  80595.7791409 ,
       150473.5263959 , 163119.60771012,  87932.58954817, 298702.16262202,
       139136.14996625, 122356.71597514, 253724.88904314, 124943.48205877,
       190378.91666769, 144311.69637702, 138626.0244507 , 108390.10270977,
       233825.57182049, 321069.91502646, 152596.17234961, 126562.81992627,
       178895.51749359, 174107.66086738, 192718.60711018,  84271.11311782,
       190926.80720662, 351883.95006535, 237798.33729765, 194988.748486  ,
        67864.93144594, 148773.78149566, 141287.95243988, 123734.33873654,
       217729.45043705, 264523.39376926, 284496.30996528, 128633.4286151 ,
       317463.52519014, 229500.59769751, 216913.34511065, 307023.5629503 ,
       358046.85973159,  88894.03376787, 213448.65694224, 151371.96447341,
       238996.23694776, 206439.35207551, 204300.0823551 , 220515.2374677 ,
       179526.34220757,  97765.6272535 , 133030.81385704, 121228.16797873,
       214547.60810557, 163777.7955153 , 164400.69495282, 120648.22075952,
       200289.83660161, 530827.98834524, 181721.08012888, 139279.91703071,
       211403.97993059,  88370.23134689,  56223.93985289, 101568.38551036,
        42264.51151872, 269839.95682852, 219235.93693798, 215559.95896183,
       126180.17980651,  94017.76423984, 186571.38998992, 239285.74298627,
       141136.12202832, 100748.65571206, 234996.84810376, 173317.89471207,
       181170.94648492, 249499.12608694,  59117.53960138, 229825.43138894,
       119229.96389044, 205049.8903116 , 289237.16371656, 376713.79583962,
       235314.9970644 , 181872.51622998, 131801.04010258,  81714.6219623 ,
       210450.63983049, 196657.14311654, 116418.20296697, 157203.17235194,
       141407.83746423, 278946.26986855, 120653.05708991, 147431.71254586,
       120051.95320748, 357597.60225625, 147189.71484064, 143672.22155962,
        61735.64050933, 102990.9266416 , 343747.05725362, 309335.75077205,
        95077.18818639, 193859.12199209, 115282.82098686, 164363.93263394,
       299418.5059694 , 160799.72506624, 237317.14924855, 150426.28776082,
       141741.98038769,  80971.65674594, 116287.05394993, 199189.16365541,
       221398.20911547, 164624.59670678, 185917.00418407, 232085.78361985,
       161510.37140789, 104607.1963465 , 200311.61620573, 193889.93887367,
       360335.63024878, 246880.79734332, 205218.08855951, 149324.66818695,
       219012.52290643, 167950.75652274, 145145.3185618 , 213907.37040049,
       166208.00694456, 177565.36744706, 213638.54963519, 128934.89419081,
       127262.06873794, 125732.53616292, 208350.84227703, 187367.98102726,
       151979.8053375 ,  98238.80563596, 216376.11908756, 172434.74819292,
       219380.06611489, 169934.88709875, 119439.25910027,  94943.44404374,
       194718.96343606, 119048.86428917, 223908.48489295, 371007.58160084,
       131761.7733937 , 202862.19185749, 167584.64261553, 164880.4508151 ,
       135912.95156556, 204349.79412469, 191707.23845592, 166312.47307932,
       248073.37082873, 158225.96622294, 208273.00886694, 206677.62501894,
       145066.05374631, 188972.29301732, 112066.39047525,  60843.20451138,
       233862.34059848, 227048.34860413, 236280.16982431, 202861.74018481,
       256347.4717376 , 163072.37056451, 208215.47237559, 158121.82685026,
       138950.11023881, 131493.60318192, 135309.51482778, 153401.6418395 ,
        87988.96611119, 164203.71884157,  77495.00725866, 190624.28807396,
       169614.69937832, 214544.41436747, 158770.76775523, 172081.73067662,
        72404.36280403, 134968.67610533, 121233.70422815, 106284.16747337,
       178255.45274396, 111941.930362  , 214003.2148373 , 118547.46238146,
```

```
212332.22284787, 167744.85468928, 106327.02745505, 260835.23902289,
207737.06881171, 240112.93783827, 162717.39329451,  74013.00523018,
214061.17644298, 201153.36902759, 205434.63928942, 204243.08249182,
138793.3050171 , 140952.80163746, 191512.24334502, 152215.21020836,
324085.39591178, 304774.23081285, 190512.06281379, 148811.52086828,
183278.82609735, 190898.89420526,  91427.23835665, 240968.26030388,
185624.17527773, 109241.47600849, 120318.37147449, 180293.28625808,
185887.05318141, 147999.96663851, 203640.95270348, 207262.6589715 ,
146553.60081428, 225114.56488868, 195969.54558458, 135946.68422952,
 98360.86043365, 145093.48293363, 187527.90880647, 112485.22475333,
162337.18285387, 137396.86809537, 107782.38128735, 200885.0785942 ,
186191.74170786, 246205.62704466, 236007.79504849, 109664.72894361,
 66069.691295  , 271856.03193978, 252559.63167736, 131192.23890726,
203706.62006731, 112676.08277586, 134454.59790426, 122227.56150567,
225246.88731479, 207800.75783312, 276095.60529998, 134212.74190233,
158840.30722344, 161414.71596746, 247093.81102187, 125407.70093715,
140731.92957045, 285092.65610426, 122166.41527204, 265957.1666469 ,
141846.17715971, 215257.29075479, 133644.67328416, 150444.80270244,
138159.96247507, 370790.96568459, 190143.19326877, 138562.74477407,
157486.7882223 , 206893.81926978, 193463.29137984, 162321.36289896,
247387.66849623, 176097.83285157, 319862.93484749, 108892.79206788,
157281.06435888, 114875.28570176, 150046.14802088, 224016.84426723,
182743.95014858, 184259.86747375, 216961.78153849, 217129.89206965,
150042.28914979, 183894.29966913, 241653.96996241, 185373.50085041,
142737.94707987, 213872.65242873, 227772.66374871, 371458.85335927,
159852.55997952, 105446.77692943,  74753.49864743, 218084.73229234,
115766.14039216, 124870.49571701, 158277.6359271 ,  42046.97387374,
298295.97719524, 113869.82743721, 311823.5676677 , 150483.58747027,
113422.7956805 , 110758.54387486, 281329.77955245, 168222.64224302,
186504.64434444, 118499.73767559, 118439.5366092 , 159798.94608423,
285467.65714171, 184380.86721368, 429972.49494319, 157974.11370633,
198929.95549095, 229184.47846748, 339430.23265358, 188218.71878896,
155268.47972515])
```

In [53]:

```
mean_squared_error(en_y_test,se_y_pred)
```

Out[53]:

```
1603931409.178804
```

**Step7. [Normalize using MinMaxScaler and Predict Sale Price]**

In [55]:

```
from sklearn.preprocessing import MinMaxScaler
m_scaler = MinMaxScaler()
```

In [57]:

```
ms = m_scaler.fit_transform(en_X_train)
ms1 = m_scaler.transform(en_X_test)
```

In [58]:

```
model3 =LinearRegression()
model3.fit(ms,en_y_train)
```

Out[58]:

```
LinearRegression()
```

In [60]:

```
me_y_pred = model3.predict(ms1)
me_y_pred
```

Out[60]:

```
array([164570.22894655,  85752.90331148, 198303.4444603 , 122475.14440177,
       175345.78182411, 242830.95354235, 327131.86508911, 208850.75498374,
       182132.06296088,  63457.73994563, 154016.77303418, 271424.75748691,
       196168.36447425, 231896.15634765,  99949.01255693, 199586.65283044,
       294455.29422812, 237034.98692395, 156399.80969275, 143322.19881946,
        87317.20577081, 459342.96310711, 180678.56647711,  80595.7791409 ,
       150473.5263959 , 163119.60771012,  87932.58954817, 298702.16262202,
       139136.14996625, 122356.71597514, 253724.88904314, 124943.48205877,
       190378.91666769, 144311.69637702, 138626.0244507 , 108390.10270977,
       233825.57182049, 321069.91502646, 152596.17234961, 126562.81992627,
       178895.51749359, 174107.66086738, 192718.60711018,  84271.11311782,
       190926.80720662, 351883.95006535, 237798.33729765, 194988.748486  ,
        67864.93144594, 148773.78149566, 141287.95243988, 123734.33873654,
       217729.45043706, 264523.39376926, 284496.30996528, 128633.4286151 ,
       317463.52519014, 229500.59769751, 216913.34511065, 307023.5629503 ,
       358046.85973159,  88894.03376787, 213448.65694224, 151371.96447341,
       238996.23694776, 206439.35207551, 204300.0823551 , 220515.2374677 ,
       179526.34220757,  97765.6272535 , 133030.81385704, 121228.16797873,
       214547.60810557, 163777.7955153 , 164400.69495282, 120648.22075952,
       200289.83660161, 530827.98834524, 181721.08012888, 139279.91703071,
       211403.97993059,  88370.23134689,  56223.93985289, 101568.38551036,
        42264.51151872, 269839.95682852, 219235.93693798, 215559.95896183,
       126180.17980651,  94017.76423985, 186571.38998992, 239285.74298627,
       141136.12202832, 100748.65571206, 234996.84810376, 173317.89471207,
       181170.94648492, 249499.12608694,  59117.53960138, 229825.43138894,
       119229.96389044, 205049.8903116 , 289237.16371656, 376713.79583962,
       235314.9970644 , 181872.51622998, 131801.04010258,  81714.6219623 ,
       210450.63983049, 196657.14311654, 116418.20296697, 157203.17235194,
       141407.83746423, 278946.26986855, 120653.05708991, 147431.71254586,
       120051.95320748, 357597.60225625, 147189.71484064, 143672.22155962,
        61735.64050933, 102990.9266416 , 343747.05725362, 309335.75077205,
        95077.18818639, 193859.12199209, 115282.82098686, 164363.93263394,
       299418.5059694 , 160799.72506624, 237317.14924855, 150426.28776082,
       141741.98038769,  80971.65674594, 116287.05394993, 199189.16365541,
       221398.20911546, 164624.59670678, 185917.00418407, 232085.78361985,
       161510.37140789, 104607.1963465 , 200311.61620573, 193889.93887367,
       360335.63024878, 246880.79734332, 205218.08855951, 149324.66818695,
       219012.52290643, 167950.75652274, 145145.3185618 , 213907.37040049,
       166208.00694456, 177565.36744706, 213638.54963519, 128934.89419081,
       127262.06873794, 125732.53616292, 208350.84227703, 187367.98102726,
       151979.8053375 ,  98238.80563596, 216376.11908756, 172434.74819292,
       219380.06611489, 169934.88709875, 119439.25910027,  94943.44404374,
       194718.96343606, 119048.86428917, 223908.48489295, 371007.58160084,
       131761.7733937 , 202862.19185749, 167584.64261553, 164880.4508151 ,
       135912.95156556, 204349.79412469, 191707.23845592, 166312.47307932,
       248073.37082873, 158225.96622294, 208273.00886694, 206677.62501894,
       145066.05374631, 188972.29301731, 112066.39047525,  60843.20451138,
       233862.34059848, 227048.34860413, 236280.16982431, 202861.74018481,
       256347.4717376 , 163072.37056451, 208215.47237559, 158121.82685026,
       138950.11023881, 131493.60318193, 135309.51482778, 153401.6418395 ,
        87988.96611119, 164203.71884157,  77495.00725866, 190624.28807396,
       169614.69937832, 214544.41436747, 158770.76775523, 172081.73067662,
        72404.36280403, 134968.67610533, 121233.70422815, 106284.16747337,
       178255.45274396, 111941.930362  , 214003.2148373 , 118547.46238146,
```

```
       212332.22284787, 167744.85468928, 106327.02745505, 260835.23902289,
       207737.06881171, 240112.93783827, 162717.39329451,  74013.00523018,
       214061.17644298, 201153.36902759, 205434.63928942, 204243.08249182,
       138793.3050171 , 140952.80163746, 191512.24334502, 152215.21020836,
       324085.39591178, 304774.23081285, 190512.06281379, 148811.52086828,
       183278.82609735, 190898.89420526,  91427.23835665, 240968.26030387,
       185624.17527773, 109241.47600849, 120318.37147449, 180293.28625808,
       185887.05318141, 147999.96663851, 203640.95270348, 207262.6589715 ,
       146553.60081428, 225114.56488868, 195969.54558458, 135946.68422952,
        98360.86043365, 145093.48293363, 187527.90880647, 112485.22475333,
       162337.18285387, 137396.86809537, 107782.38128735, 200885.0785942 ,
       186191.74170786, 246205.62704466, 236007.79504849, 109664.72894361,
        66069.691295  , 271856.03193978, 252559.63167735, 131192.23890726,
       203706.62006731, 112676.08277586, 134454.59790426, 122227.56150567,
       225246.88731479, 207800.75783312, 276095.60529998, 134212.74190233,
       158840.30722344, 161414.71596746, 247093.81102187, 125407.70093715,
       140731.92957045, 285092.65610426, 122166.41527204, 265957.1666469 ,
       141846.17715971, 215257.29075479, 133644.67328416, 150444.80270244,
       138159.96247507, 370790.96568459, 190143.19326877, 138562.74477407,
       157486.7882223 , 206893.81926978, 193463.29137984, 162321.36289896,
       247387.66849623, 176097.83285157, 319862.93484749, 108892.79206788,
       157281.06435888, 114875.28570176, 150046.14802088, 224016.84426723,
       182743.95014858, 184259.86747375, 216961.78153849, 217129.89206965,
       150042.28914979, 183894.29966913, 241653.96996241, 185373.50085041,
       142737.94707987, 213872.65242873, 227772.66374871, 371458.85335927,
       159852.55997952, 105446.77692943,  74753.49864743, 218084.73229234,
       115766.14039216, 124870.49571701, 158277.6359271 ,  42046.97387374,
       298295.97719524, 113869.82743721, 311823.5676677 , 150483.58747027,
       113422.7956805 , 110758.54387486, 281329.77955245, 168222.64224302,
       186504.64434444, 118499.73767559, 118439.5366092 , 159798.94608423,
       285467.65714171, 184380.86721368, 429972.49494319, 157974.11370633,
       198929.95549095, 229184.47846748, 339430.23265358, 188218.71878896,
       155268.47972515])
```

In [61]:

```
mean_squared_error(en_y_test,me_y_pred)
```

Out[61]:

```
1603931409.1788046
```

### Step8. [Predict using SGD Regressor]¶

In [62]:

```
from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline
```

In [63]:

```
SGD = make_pipeline(StandardScaler(), SGDRegressor(max_iter=1000, tol=1e-3))
SGD.fit(X, y)
```

Out[63]:

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('sgdregressor', SGDRegressor())])
```

In [64]:

```
r_y_pred = SGD.predict(X)
r_y_pred
```

Out[64]:

```
array([234288.73879777, 194490.33616557, 228853.39973738, ...,
       227979.03201414, 122838.6043447 , 145512.54381131])
```

In [65]:

```
mean_squared_error(y,r_y_pred)
```

Out[65]:

```
1236182786.527191
```

### Step8. [Predict using Ridge Regression]

In [66]:

```
from sklearn.linear_model import Ridge
```

In [69]:

```
RidgeCV = Ridge(alpha=1.0)
RidgeCV.fit(s,en_y_train)
ridge_y_pred = RidgeCV.predict(s)
ridge_y_pred
```

Out[69]:

```
array([201846.8910729 , 254209.54024493, 338463.77172917, ...,
        88246.41853185, 352706.2962025 , 210331.1763083 ])
```

In [70]:

```
mean_squared_error(en_y_train,ridge_y_pred)
```

Out[70]:

```
1065862697.5487247
```

### Step8. [Predict using Lasso Regression]

In [71]:

```
from sklearn.linear_model import Lasso
```

In [72]:

```
LassoCV = Lasso(alpha=1.0)
LassoCV.fit(s,en_y_train)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_
descent.py:529: ConvergenceWarning: Objective did not converge. You might wa
nt to increase the number of iterations. Duality gap: 188735847371.2207, tol
erance: 631484515.8030617
  model = cd_fast.enet_coordinate_descent(
```

Out[72]:

```
Lasso()
```

In [74]:

```
lasso_y_pred = LassoCV.predict(s)
lasso_y_pred
```

Out[74]:

```
array([201847.50954454, 254242.82719469, 338702.94581108, ...,
        88327.65667808, 352770.5563025 , 210188.32752228])
```

In [75]:

```
mean_squared_error(en_y_train,lasso_y_pred)
```

Out[75]:

```
1065858586.2635835
```

**Step9.[RMSE]. Print Root Mean Squared Error values (use numpy.sqrt() method) as below and compare error values.¶**

In [76]:

```
from math import sqrt
```

In [77]:

```
print("RMSE without one hot encoding:  ",sqrt(mean_squared_error(y_test,y_pred)))
print("RMSE with one hot encoding:  ",sqrt(mean_squared_error(en_y_test,en_y_pred)))
print("RMSE with one and Standard Scaling:  ",sqrt(mean_squared_error(en_y_test,se_y_pred))
print("RMSE with one and MinMax Scaling:  ",sqrt(mean_squared_error(en_y_test,me_y_pred)))
print("RMSE of SGDRegressor with one and Standard Scaler:  ",sqrt(mean_squared_error(y,r_y_
print("RMSE of RigdCV with one and Standard Scaler:  ",sqrt(mean_squared_error(en_y_train,r
print("RMSE of LassoCV with one and Standard Scaler:  ",sqrt(mean_squared_error(en_y_train,
```

```
RMSE without one hot encoding:   30496.102314350515
RMSE with one hot encoding:   40049.11246430626
RMSE with one and Standard Scaling:   40049.11246430817
RMSE with one and MinMax Scaling:   40049.11246430818
RMSE of SGDRegressor with one and Standard Scaler:   35159.39115694683
RMSE of RigdCV with one and Standard Scaler:   32647.552703820307
RMSE of LassoCV with one and Standard Scaler:   32647.489739083823
```