LAB 4 : House Price Prediciton using LR with Regularization

Name : MURALI KUMAR R

Roll no : 225229120

Step 1 : Import Dataset

In [62]:

```
import pandas as pd
```

In [63]:

```
data = pd.read_csv("Ames_House_Sales_Cropped.csv")
data.head()
```

Out[63]:

	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 [46]:

```
data.shape
```

Out[46]:

```
(1379, 39)
```

In [47]:

```
data.shape[1]
```

Out[47]:

```
39
```

In [70]:

```
data.dtypes
```

Out[70]:

```
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 [49]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1379 entries, 0 to 1378
Data columns (total 39 columns):
BldgType         1379 non-null object
CentralAir       1379 non-null object
1stFlrSF         1379 non-null float64
2ndFlrSF         1379 non-null float64
3SsnPorch        1379 non-null float64
BedroomAbvGr     1379 non-null int64
BsmtFinSF1       1379 non-null float64
BsmtFinSF2       1379 non-null float64
BsmtFullBath     1379 non-null int64
BsmtHalfBath     1379 non-null int64
BsmtUnfSF        1379 non-null float64
EnclosedPorch    1379 non-null float64
Fireplaces       1379 non-null int64
FullBath         1379 non-null int64
GarageArea       1379 non-null float64
GarageCars       1379 non-null int64
GarageYrBlt      1379 non-null float64
GrLivArea        1379 non-null float64
HalfBath         1379 non-null int64
KitchenAbvGr     1379 non-null int64
LotArea          1379 non-null float64
LotFrontage      1379 non-null float64
LowQualFinSF     1379 non-null float64
MSSubClass       1379 non-null int64
MasVnrArea       1379 non-null float64
MiscVal          1379 non-null float64
MoSold           1379 non-null int64
OpenPorchSF      1379 non-null float64
OverallCond      1379 non-null int64
OverallQual      1379 non-null int64
PoolArea         1379 non-null float64
ScreenPorch      1379 non-null float64
TotRmsAbvGrd     1379 non-null int64
TotalBsmtSF      1379 non-null float64
WoodDeckSF       1379 non-null float64
YearBuilt        1379 non-null int64
YearRemodAdd     1379 non-null int64
YrSold           1379 non-null int64
SalePrice        1379 non-null float64
dtypes: float64(21), int64(16), object(2)
memory usage: 420.2+ KB
```

In [50]:

```
data['CentralAir'].value_counts
print(data)
```

```
7         1Fam          Y    1107.0     983.0        0.0             3
8         1Fam          Y    1022.0     752.0        0.0             2
9       2fmCon          Y    1077.0       0.0        0.0             2
10        1Fam          Y    1040.0       0.0        0.0             3
11        1Fam          Y    1182.0    1142.0        0.0             4
12        1Fam          Y     912.0       0.0        0.0             2
13        1Fam          Y    1494.0       0.0        0.0             3
14        1Fam          Y    1253.0       0.0        0.0             2
15        1Fam          Y     854.0       0.0        0.0             2
16        1Fam          Y    1004.0       0.0        0.0             2
17      Duplex          Y    1296.0       0.0        0.0             2
18        1Fam          Y    1114.0       0.0        0.0             3
19        1Fam          Y    1339.0       0.0        0.0             3
20        1Fam          Y    1158.0    1218.0        0.0             4
21        1Fam          Y    1108.0       0.0        0.0             3
22        1Fam          Y    1795.0       0.0        0.0             3
23      TwnhsE          Y    1060.0       0.0        0.0             3
24        1Fam          Y    1060.0       0.0        0.0             3
25        1Fam          Y    1600.0       0.0        0.0             3
26        1Fam          Y     900.0       0.0        0.0             3
```

Step 2 : Predict Sale Price with out Categorical features

In [74]:

```
#1

df = data.drop("BldgType", axis='columns')
print(df)
```

```
     CentralAir  1stFlrSF  2ndFlrSF  3SsnPorch  BedroomAbvGr  BsmtFinSF1
\
0             Y     856.0     854.0        0.0             3       706.0
1             Y    1262.0       0.0        0.0             3       978.0
2             Y     920.0     866.0        0.0             3       486.0
3             Y     961.0     756.0        0.0             3       216.0
4             Y    1145.0    1053.0        0.0             4       655.0
5             Y     796.0     566.0      320.0             1       732.0
6             Y    1694.0       0.0        0.0             3      1369.0
7             Y    1107.0     983.0        0.0             3       859.0
8             Y    1022.0     752.0        0.0             2         0.0
9             Y    1077.0       0.0        0.0             2       851.0
10            Y    1040.0       0.0        0.0             3       906.0
11            Y    1182.0    1142.0        0.0             4       998.0
12            Y     912.0       0.0        0.0             2       737.0
13            Y    1494.0       0.0        0.0             3         0.0
14            Y    1253.0       0.0        0.0             2       733.0
15            Y     854.0       0.0        0.0             2         0.0
16            Y    1004.0       0.0        0.0             2       578.0
```

In [75]:

```
df = data.drop("CentralAir", axis='columns')
print(df)
```

```
     BldgType  1stFlrSF  2ndFlrSF  3SsnPorch  BedroomAbvGr  BsmtFinSF1  \
0        1Fam     856.0     854.0        0.0             3       706.0
1        1Fam    1262.0       0.0        0.0             3       978.0
2        1Fam     920.0     866.0        0.0             3       486.0
3        1Fam     961.0     756.0        0.0             3       216.0
4        1Fam    1145.0    1053.0        0.0             4       655.0
5        1Fam     796.0     566.0      320.0             1       732.0
6        1Fam    1694.0       0.0        0.0             3      1369.0
7        1Fam    1107.0     983.0        0.0             3       859.0
8        1Fam    1022.0     752.0        0.0             2         0.0
9      2fmCon    1077.0       0.0        0.0             2       851.0
10       1Fam    1040.0       0.0        0.0             3       906.0
11       1Fam    1182.0    1142.0        0.0             4       998.0
12       1Fam     912.0       0.0        0.0             2       737.0
13       1Fam    1494.0       0.0        0.0             3         0.0
14       1Fam    1253.0       0.0        0.0             2       733.0
15       1Fam     854.0       0.0        0.0             2         0.0
16       1Fam    1004.0       0.0        0.0             2       578.0
17     Duplex    1296.0       0.0        0.0             2         0.0
```

In [76]:

```
data.pop('BldgType')
```

Out[76]:

```
0         1Fam
1         1Fam
2         1Fam
3         1Fam
4         1Fam
5         1Fam
6         1Fam
7         1Fam
8         1Fam
9       2fmCon
10        1Fam
11        1Fam
12        1Fam
13        1Fam
14        1Fam
15        1Fam
16        1Fam
17      Duplex
18        1Fam
19        1Fam
20        1Fam
21        1Fam
22        1Fam
23      TwnhsE
24        1Fam
25        1Fam
26        1Fam
27        1Fam
28        1Fam
29        1Fam
         ...
1349      1Fam
1350      1Fam
1351      1Fam
1352      1Fam
1353    TwnhsE
1354      1Fam
1355      1Fam
1356      1Fam
1357      1Fam
1358      1Fam
1359      1Fam
1360      1Fam
1361      1Fam
1362      1Fam
1363    TwnhsE
1364      1Fam
1365      1Fam
1366      1Fam
1367      1Fam
1368      1Fam
1369      1Fam
1370      1Fam
1371      1Fam
1372    TwnhsE
```

```
1373      1Fam
1374      1Fam
1375      1Fam
1376      1Fam
1377      1Fam
1378      1Fam
Name: BldgType, Length: 1379, dtype: object
```

In [77]:

```
data.pop('CentralAir')
```

Out[77]:

```
0       Y
1       Y
2       Y
3       Y
4       Y
5       Y
6       Y
7       Y
8       Y
9       Y
10      Y
11      Y
12      Y
13      Y
14      Y
15      Y
16      Y
17      Y
18      Y
19      Y
20      Y
21      Y
22      Y
23      Y
24      Y
25      Y
26      Y
27      Y
28      Y
29      N
       ..
1349    Y
1350    Y
1351    Y
1352    Y
1353    Y
1354    Y
1355    Y
1356    Y
1357    Y
1358    Y
1359    Y
1360    Y
1361    Y
1362    Y
1363    Y
1364    Y
1365    N
1366    Y
1367    Y
1368    Y
1369    Y
1370    N
1371    Y
1372    Y
```

```
1373    Y
1374    Y
1375    Y
1376    Y
1377    Y
1378    Y
Name: CentralAir, Length: 1379, dtype: object
```

In [78]:

```
#2

X=pd.DataFrame(df)
colm=[1]
X=X[X.columns[colm]]
```

In [79]:

```
X
```

Out[79]:

	1stFlrSF
0	856.0
1	1262.0
2	920.0
3	961.0
4	1145.0
5	796.0
6	1694.0
7	1107.0
8	1022.0
9	1077.0
10	1040.0
11	1182.0
12	912.0
13	1494.0
14	1253.0
15	854.0
16	1004.0
17	1296.0
18	1114.0
19	1339.0
20	1158.0
21	1108.0
22	1795.0
23	1060.0
24	1060.0
25	1600.0
26	900.0
27	1704.0
28	1600.0
29	520.0
...	...
1349	1048.0
1350	804.0
1351	1440.0

	1stFlrSF
1352	734.0
1353	958.0
1354	968.0
1355	962.0
1356	1126.0
1357	1537.0
1358	864.0
1359	1932.0
1360	1236.0
1361	1040.0
1362	1423.0
1363	848.0
1364	1026.0
1365	952.0
1366	1422.0
1367	913.0
1368	1188.0
1369	1220.0
1370	796.0
1371	1578.0
1372	1072.0
1373	1221.0
1374	953.0
1375	2073.0
1376	1188.0
1377	1078.0
1378	1256.0

1379 rows × 1 columns

In [80]:

```
y=data['FullBath'].values
```

In [81]:

```
y
```

Out[81]:

```
array([2, 2, 2, ..., 2, 1, 1], dtype=int64)
```

In [85]:

```
type(X)
```

Out[85]:

```
pandas.core.frame.DataFrame
```

In [86]:

```
type(y)
```

Out[86]:

```
numpy.ndarray
```

In [87]:

```
#3

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [94]:

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

Out[94]:

	1stFlrSF
1038	1601.0
1091	1265.0
198	689.0
1081	1298.0
678	864.0
310	1056.0
981	800.0
1207	1128.0
1261	1656.0
1271	1776.0
526	1557.0
792	810.0
15	854.0
432	979.0
949	985.0
493	3138.0
71	1086.0
379	1165.0
231	840.0
391	1340.0
1139	1056.0
631	1839.0
1268	728.0
1119	1014.0
794	1165.0
454	916.0
70	885.0
816	1002.0
1072	986.0
63	1143.0
...	...
615	851.0
637	988.0
815	1372.0

	1stFlrSF
222	1694.0
1044	1500.0
864	1001.0
527	1392.0
707	1038.0
1223	1052.0
524	897.0
939	774.0
649	2444.0
291	793.0
298	1472.0
609	755.0
108	729.0
448	1493.0
1347	1252.0
1152	1211.0
1259	765.0
959	1360.0
433	832.0
518	928.0
683	1232.0
1248	1478.0
1102	959.0
636	2046.0
253	1363.0
103	1216.0
277	975.0

1034 rows × 1 columns

In [95]:

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

Out[95]:

	1stFlrSF
1117	1572.0
533	1535.0
90	680.0
214	1194.0
990	1302.0
1209	1573.0
260	2121.0
863	1015.0
524	897.0
878	1788.0
677	848.0
1268	728.0
485	789.0
313	943.0
880	1466.0
261	1156.0
904	561.0
940	1050.0
1052	1008.0
56	1426.0
254	1164.0
1317	1500.0
1076	952.0
50	816.0
807	1026.0
1159	1702.0
968	1264.0
305	1719.0
335	1167.0
79	1563.0
...	...
562	1402.0
973	1352.0
387	1056.0

	1stFlrSF
127	1214.0
1174	792.0
1226	4692.0
80	1065.0
181	855.0
514	768.0
1369	1220.0
578	1054.0
1253	1584.0
1018	848.0
381	1620.0
1016	1096.0
929	1164.0
1341	1844.0
480	1041.0
173	808.0
1066	847.0
664	1324.0
636	2046.0
302	1163.0
944	944.0
184	1713.0
1259	765.0
378	672.0
317	1621.0
1058	1040.0
320	841.0

345 rows × 1 columns

In [96]:

```
#4

model = LinearRegression()
```

In [97]:

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

Out[97]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [103]:

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

In [104]:

```
print(y_pred)
```

```
[1.7945931  1.77483572 1.31828013 1.59274747 1.65041765 1.79512708
 2.08774985 1.49716449 1.43415447 1.90993346 1.4079893  1.34391132
 1.37648429 1.4587177  1.73799088 1.57245611 1.25473614 1.5158539
 1.4934266  1.71663156 1.57672797 1.75614631 1.46352355 1.39090184
 1.5030383  1.86401091 1.63012629 1.87308862 1.57832992 1.78978725
 1.48381491 1.66910706 1.9814872  2.03274958 1.43255252 1.36847455
 1.75721428 1.38983387 1.24939631 1.32789183 1.48007702 1.48274694
 1.33323166 1.72250537 2.07493425 1.51478593 1.45070795 1.3257559
 1.34711522 1.51051406 1.47900906 1.77216581 1.55857255 1.65789341
 1.44430015 1.87415659 1.87789447 1.46993134 1.54148509 1.48274694
 1.83303988 1.44109625 1.76469004 1.79619505 1.64454383 1.56231043
 1.51051406 1.47580516 1.60876696 1.66964104 1.52760152 1.33963946
 1.71930147 1.48274694 1.55643661 1.39944557 1.86240896 1.87202065
 1.58153382 1.63119425 1.62104858 1.58740764 1.6995441  1.4619216
 2.08347798 1.5409511  1.49235864 2.30294505 1.62371849 1.50464025
 1.46672745 1.40264947 1.52119373 1.37167845 1.45017397 1.53828119
 1.4902227  1.51051406 1.55750458 1.62692239 1.72090342 1.6803207
 1.57886391 1.73905885 1.67284494 1.61731069 1.41653303 1.55750458
 1.67017502 1.84585548 1.76255411 1.46886338 1.6867285  1.29158097
 1.6327962  1.84959336 1.449106   1.28410521 1.57459204 1.5569706
 1.42934863 1.54949483 1.39570769 1.42774668 1.90352566 1.46352355
 1.71823351 1.48755279 1.43255252 1.86507887 1.56498034 1.39090184
 1.73692292 1.62425247 1.84104963 1.18905621 2.36114922 1.52760152
 1.53454331 1.56871823 1.64454383 1.85226328 1.59541738 1.43255252
 1.34391132 1.38769794 1.43575642 1.74439868 1.52813551 1.57138814
 1.64988367 1.36580463 2.16143952 1.39944557 1.78551538 1.55964051
 1.24672639 1.86294294 1.7849814  1.9686716  1.31400827 1.64027197
 1.45658177 1.42347481 1.48488287 1.42294083 1.90085574 1.354057
 1.85706913 1.55323272 1.44963999 1.41653303 1.53934915 1.21308545
 2.23245928 1.82876802 1.32415395 1.6835246  1.33269768 1.57512602
 1.71022376 1.61837866 1.93663261 1.36473666 1.5537667  1.52172771
 1.41653303 1.21308545 1.44536812 1.62692239 1.39303777 1.37488234
 1.58954357 1.63012629 1.38769794 1.47099931 1.5062422  1.54148509
 1.95478804 1.66216528 1.79993293 1.51051406 1.46298956 1.24138656
 1.35619293 1.73799088 1.38129014 2.2602264  1.59595137 1.51478593
 1.60235917 1.5884756  1.80954463 1.62158256 1.28944504 1.41653303
 1.72677724 1.51745584 1.69260232 1.82503014 1.51051406 1.85439921
 1.65682545 1.48274694 1.57726196 1.54896085 2.24260496 1.75988419
 1.38449404 1.90512761 1.4079893  1.47046533 1.48755279 1.39997955
 1.58633967 1.60022323 1.6803207  1.53934915 1.50090237 1.47794109
 1.33963946 1.46779541 1.2654158  1.70915579 1.639204   1.51692186
 1.53187339 1.59328145 1.51104805 1.46672745 1.41653303 1.82663209
 1.75935021 1.36847455 1.4683294  2.21750775 1.66430121 1.51051406
 1.28196928 1.74493267 1.73158309 1.4651255  1.51852381 1.51478593
 1.50036838 1.75935021 1.38449404 1.60342713 1.55910653 1.65949536
 1.51959178 1.56711628 1.67070901 1.51852381 1.46779541 1.74439868
 1.95318609 1.41653303 1.93289473 1.5062422  1.7817775  1.43255252
 1.91206939 1.51051406 1.5062422  1.39090184 1.39677565 1.5030383
 1.63653409 1.65468951 1.32041606 1.60663103 1.36313472 1.64988367
 1.54255305 1.92221507 2.10643926 1.44216422 1.63653409 1.56177645
 1.44376617 1.21308545 1.75507834 1.74653461 1.7753697  1.43629041
 1.84318556 1.36313472 1.32308598 1.60235917 1.50731017 1.63706807
 1.28410521 1.36847455 1.60022323 1.70381596 1.67711681 1.51905779
 1.60342713 1.37808624 3.46062051 1.52386364 1.41172718 1.36527065
 1.60663103 1.51798983 1.8010009  1.4079893  1.82022429 1.54041712
 1.57672797 1.93983651 1.51104805 1.38662997 1.40745532 1.66216528
```

```
 2.04770111 1.57619399 1.45925168 1.86988472 1.3636687  1.31400827
 1.82075827 1.51051406 1.40425142]
```

In [114]:

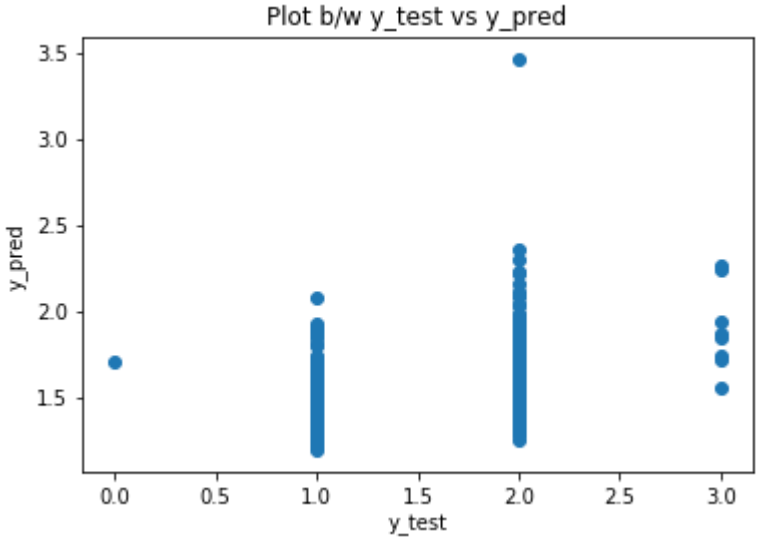
```
#5

import sklearn.metrics as metrics
mse=metrics.mean_squared_error(y_test,y_pred)
print("MSE without Categorical: ",mse)
```

```
MSE without Categorical:  0.25431785567631415
```

Step-3: [Create Scatter plot]

In [115]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(y_test,y_pred)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.title("Plot b/w y_test vs y_pred")
plt.show()
```



Step-4:[Encode Categorical columns]

In [116]:

```
encoding=pd.get_dummies(df)
```

In [118]:

```
print(encoding)
```

```
      1stFlrSF  2ndFlrSF  3SsnPorch  BedroomAbvGr  BsmtFinSF1  BsmtFinSF2
\
0        856.0     854.0        0.0             3       706.0         0.0
1       1262.0       0.0        0.0             3       978.0         0.0
2        920.0     866.0        0.0             3       486.0         0.0
3        961.0     756.0        0.0             3       216.0         0.0
4       1145.0    1053.0        0.0             4       655.0         0.0
5        796.0     566.0      320.0             1       732.0         0.0
6       1694.0       0.0        0.0             3      1369.0         0.0
7       1107.0     983.0        0.0             3       859.0        32.0
8       1022.0     752.0        0.0             2         0.0         0.0
9       1077.0       0.0        0.0             2       851.0         0.0
10      1040.0       0.0        0.0             3       906.0         0.0
11      1182.0    1142.0        0.0             4       998.0         0.0
12       912.0       0.0        0.0             2       737.0         0.0
13      1494.0       0.0        0.0             3         0.0         0.0
14      1253.0       0.0        0.0             2       733.0         0.0
15       854.0       0.0        0.0             2         0.0         0.0
16      1004.0       0.0        0.0             2       578.0         0.0
```

Step-5:[Predict Sale Price]

In [119]:

```
X=encoding.drop(["SalePrice"],axis=1)
```

In [121]:

```
print(X)
```

```
      1stFlrSF  2ndFlrSF  3SsnPorch  BedroomAbvGr  BsmtFinSF1  BsmtFinSF2
\
0        856.0     854.0        0.0             3       706.0         0.0
1       1262.0       0.0        0.0             3       978.0         0.0
2        920.0     866.0        0.0             3       486.0         0.0
3        961.0     756.0        0.0             3       216.0         0.0
4       1145.0    1053.0        0.0             4       655.0         0.0
5        796.0     566.0      320.0             1       732.0         0.0
6       1694.0       0.0        0.0             3      1369.0         0.0
7       1107.0     983.0        0.0             3       859.0        32.0
8       1022.0     752.0        0.0             2         0.0         0.0
9       1077.0       0.0        0.0             2       851.0         0.0
10      1040.0       0.0        0.0             3       906.0         0.0
11      1182.0    1142.0        0.0             4       998.0         0.0
12       912.0       0.0        0.0             2       737.0         0.0
13      1494.0       0.0        0.0             3         0.0         0.0
14      1253.0       0.0        0.0             2       733.0         0.0
15       854.0       0.0        0.0             2         0.0         0.0
16      1004.0       0.0        0.0             2       578.0         0.0
```

In [122]:

```
y=encoding["SalePrice"].values
```

In [123]:

```
y
```

Out[123]:

```
array([208500., 181500., 223500., ..., 266500., 142125., 147500.])
```

In [124]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25,random_state=42)
```

In [125]:

```
from sklearn.linear_model import LinearRegression
import sklearn.metrics as metrics
reg=LinearRegression()
reg.fit(X_train,y_train)
y_pred=reg.predict(X_test)
print(y)
mse_cd=metrics.mean_squared_error(y_test,y_pred)
print("MSE with Categorical data: ",mse_cd)
```

```
[208500. 181500. 223500. ... 266500. 142125. 147500.]
MSE with Categorical data:  1462257073.0662968
```

step-6:[Normalize using StandardScaler and Predict Sale Price]

In [126]:

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
ss_X_train=ss.fit_transform(X_train)
ss_X_train
```

Out[126]:

```
array([[ 0.39851037, -0.79290427, -0.11340519, ..., -0.16060402,
        -0.16683226,  3.45325933],
       [ 1.57467708, -0.79290427, -0.11340519, ..., -0.16060402,
        -0.16683226, -0.2895815 ],
       [ 0.37564751,  0.70143387, -0.11340519, ..., -0.16060402,
        -0.16683226, -0.2895815 ],
       ...,
       [ 1.22157303, -0.79290427, -0.11340519, ..., -0.16060402,
        -0.16683226, -0.2895815 ],
       [-1.11297817,  0.90510323, -0.11340519, ..., -0.16060402,
        -0.16683226, -0.2895815 ],
       [ 0.11145456, -0.79290427, -0.11340519, ..., -0.16060402,
        -0.16683226,  3.45325933]])
```

In [127]:

```
ss_X_test=ss.transform(X_test)
ss_X_test
```

Out[127]:

```
array([[ 0.85830772, -0.79290427, -0.11340519, ..., -0.16060402,
        -0.16683226,  3.45325933],
       [-0.64810018, -0.79290427, -0.11340519, ..., -0.16060402,
        -0.16683226, -0.2895815 ],
       [-0.21624632,  0.76093278, -0.11340519, ..., -0.16060402,
        -0.16683226, -0.2895815 ],
       ...,
       [-0.04350477,  0.37647826, -0.11340519, ..., -0.16060402,
        -0.16683226, -0.2895815 ],
       [-0.64301955,  1.46805451, -0.11340519, ..., -0.16060402,
        -0.16683226, -0.2895815 ],
       [-0.29499614,  0.81585486, -0.11340519, ..., -0.16060402,
        -0.16683226,  3.45325933]])
```

In [128]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(ss_X_train,y_train)
ss_y_pred=lr.predict(ss_X_test)
ss_y_pred
```

Out[128]:

```
array([257705.55315935, 113388.1884277 ,  82068.99572712, 204201.26191705,
       209061.22758409,  68369.62205087, 235169.83033484, 205515.2333543 ,
       187902.32698651, 234798.82649979, 101547.8958748 , 304548.16800268,
       102147.45845742, 289071.55986965, 204551.33784575, 142334.04415565,
       249538.99633789, 147926.91898742, 210427.56700492, 277452.40447932,
        91672.60985181, 154037.47433551, 160585.42237977, 243021.00643366,
       372535.29049953, 220231.32500124, 117171.98204788, 100999.24482024,
       318540.88664442, 167919.30013436, 258120.81660154, 201492.3980033 ,
       152722.64420558, 142857.11323697, 205244.27810938, 382301.07859311,
       125533.4935813 , 142044.95437736, 273532.67402936, 214113.92392392,
       170482.87228361, 124638.37580996, 197862.28384768, 376289.11946184,
       147994.00246676, 288031.11517941, 108980.70309896, 218349.21374847,
        88637.6446453 , 295106.81329392, 125893.80420659,  53346.087802  ,
       133513.53832426, 173768.28080508, 214788.07310828, 110830.40443226,
       101828.74096487, 198514.78186813, 139859.99612575, 313006.35543921,
        72830.27881854, 185229.91107443, 154210.44480603, 299390.76809475,
        77906.14167878, 324323.21875149, 165349.81819242, 112406.10625287,
       200991.83560402, 206543.45720845, 130800.46466233, 271444.95907303,
       119527.87722529, 143636.91898397,  75627.46462833, 137101.05750988,
        33760.4224089 , 164389.48167959, 273765.78837598, 116291.61441963,
       319918.59340852, 218979.25183253, 293793.88610029, 211568.46261197,
        91543.3749705 , 265974.97316117, 142666.18404089, 118913.21415751,
       143527.83158995, 279098.88504763, 269498.96336423, 301817.47032472,
       184208.51125745, 165292.03489123, 150263.65827628, 213833.2655703 ,
       249187.03928093, 224043.18809488, 274283.18900056, 245886.41969786,
       111947.29004332,  99646.03059904, 195321.81008127, 208166.59839861,
       155216.01959805, 218425.18368557, 212750.54097645, 122368.11310639,
       162757.63878089, 222169.88155819, 147014.49376385, 110368.43566673,
       319899.82904573, 298351.87124067, 337032.83390538,  88509.07429469,
       146988.69599587, 227389.81107481, 122594.14817476, 205282.51397642,
       217502.30210567, 105979.83992047, 133900.83294549, 220401.61669592,
       134533.75331447, 212537.80466928, 145822.85887485, 292001.916382  ,
       183929.46525771,  97526.47834009, 325926.21770803, 168336.1131855 ,
       123085.1220771 , 134220.19527173, 309659.61248556, 134621.72040037,
       281856.50464034, 130952.27132846,  90896.53528967, 161151.44855318,
       148245.57782647, 158743.4357484 , 178070.85215267, 257294.2940174 ,
       115127.62197178, 137952.99415548, 238164.54430471, 318660.27941592,
       104401.30865868, 193472.26032648, 172525.52490827, 144821.95912031,
        95416.55962006, 251000.87164896, 304342.18153791,  53250.32972054,
       284216.20870438,  93659.45817633, 138032.39579121, 164085.60702787,
       209625.15967015, 203555.39716468, 176647.91330201, 250327.41303183,
       142665.14167133, 134528.78912805, 171693.9768114 , 344207.91341043,
       180844.35359567,  99562.63214138, 131826.54405953, 102299.72625508,
       123291.80105031, 193395.9252095 , 156605.04243354, 269829.46043183,
       214190.70831182, 151984.7953014 , 116934.31887546, 245011.78106252,
       134633.66293969, 264929.22303568, 302668.94028882, 114015.05460455,
       146441.77913368, 141357.0027005 , 159538.24431022, 231676.9800001 ,
       233400.52995637, 140373.71019287, 356288.93969829, 125341.52789164,
       236478.64332758, 118544.29428475, 104052.33673557, 158273.87407569,
       146372.17687286, 414566.38246119, 337627.39928394, 292117.18150842,
       282078.91200937, 275067.19989016, 322612.66136864, 307637.10995205,
```

```
       202350.23801452, 142183.78896605, 157065.48055881, 264276.61742703,
       208753.75591994, 148579.51002889, 104862.16487731, 150591.38593666,
        99772.55251593, 298096.55815911, 342588.01923766, 255305.01771014,
       143960.71988075, 195119.05777514, 128666.72297026, 264711.22067369,
       123081.58596766, 133276.65106096, 307682.56638111, 167370.72621835,
       173611.40294569, 566922.89973793, 182845.90985561, 142761.38958302,
       195509.43613916, 108024.21249371, 181468.4749134 , 326585.34966799,
       138869.36625264, 197426.00590137, 116165.04575965,  56284.3053901 ,
       200955.88091972, 274048.32217841, 110417.34567976, 196135.43072017,
       234372.28691563, 121145.90339885, 137720.10310105, 289202.11769827,
       200172.92694974, 364189.74509201, 116030.51909338, 125795.16223833,
       225786.51294502, 172227.79801754,  93992.80359347, 178447.60844839,
       218222.12411208, 225543.68291325, 131988.93700397,  67715.73592553,
       243465.88203249, 138352.27496145, 113593.06302933,  85459.22003663,
       207984.89664431, 171931.87360976, 270275.52596967, 254063.71300068,
       103617.70924263, 221017.52163077, 184415.52627559, 170423.76300457,
       156753.51512201, 225827.88009121,  85239.58779949, 153124.07150444,
       180115.88285171, 200374.55472456, 181098.18784203, 225242.57395942,
        74711.27252524,  80969.30672958, 139758.57667939, 230578.094454  ,
       153170.20501125, 322971.46711662, 196110.76794079,  57211.01436041,
       206435.70181525, 110001.24115869, 196124.45986226,  98149.47173074,
       222762.29559005, 211717.28829211, 199106.79860015, 111408.67197075,
        98357.53762727, 236789.84151717, 184747.58056935, 189104.68624707,
       265561.92131553, 247902.83307981, 108699.68852259, 162096.8708319 ,
       267886.69464291, 116383.15554858, 220330.53931432,  99180.31360352,
       227632.64859861, 158389.66441237, 107719.45784191, 219484.51001045,
        81609.43998025,  76387.37318686,  98141.72708677, 303721.3408993 ,
       146369.01233242, 186764.52109091, 241560.14901282,  59680.7506418 ,
       144391.58352994, 108989.53482394, 440759.11465924, 120203.96498623,
       115464.82503806, 275628.10476953, 201938.278101  , 234381.86154612,
       201279.3818041 ,  99150.2336282 , 143911.25283385, 211637.42058795,
       166047.47184034, 162071.19891851, 150358.98354071, 191641.12002927,
        96701.13169592, 167691.2497733 ,  81094.67813886, 169629.50142262,
       175660.21784513, 165627.98345084, 133129.35910636, 205456.16714335,
       236574.8313558 , 124117.15926504, 173545.52840021, 259173.50874913,
       227170.15541242])
```

In [129]:

```
ss_mse=metrics.mean_squared_error(y_test,ss_y_pred)
print("SS_MSE: ",ss_mse)
```

```
SS_MSE:  1462257073.0675404
```

step-7: [Normalize using MinMaxScaler and Predict Sale Price]

In [130]:

```
from sklearn.preprocessing import MinMaxScaler
mm=MinMaxScaler()
mm_X_train=mm.fit_transform(X_train)
mm_X_test=mm.transform(X_test)
mm_lr=LinearRegression()
mm_lr.fit(mm_X_train,y_train)
mm_y_pred=mm_lr.predict(mm_X_test)
print("Predictions of scaled data using MinMaxScaler:",mm_y_pred)
```

```
Predictions of scaled data using MinMaxScaler: [257705.55315935 113388.18842
77   82068.99572712 204201.26191705
 209061.22758409  68369.62205087 235169.83033484 205515.2333543
 187902.32698651 234798.82649979 101547.8958748  304548.16800268
 102147.45845742 289071.55986965 204551.33784575 142334.04415565
 249538.99633789 147926.91898742 210427.56700492 277452.40447932
  91672.60985181 154037.47433551 160585.42237977 243021.00643366
 372535.29049953 220231.32500124 117171.98204788 100999.24482024
 318540.88664442 167919.30013436 258120.81660154 201492.3980033
 152722.64420558 142857.11323697 205244.27810938 382301.07859311
 125533.4935813  142044.95437736 273532.67402936 214113.92392392
 170482.87228361 124638.37580996 197862.28384768 376289.11946184
 147994.00246676 288031.11517941 108980.70309896 218349.21374847
  88637.6446453  295106.81329392 125893.80420659  53346.087802
 133513.53832426 173768.28080508 214788.07310828 110830.40443226
 101828.74096487 198514.78186813 139859.99612575 313006.35543921
  72830.27881854 185229.91107443 154210.44480603 299390.76809475
  77906.14167878 324323.21875149 165349.81819242 112406.10625287
 200991.83560402 206543.45720845 130800.46466233 271444.95907303
 119527.87722529 143636.91898397  75627.46462833 137101.05750988
  33760.4224089  164389.48167959 273765.78837598 116291.61441963
 319918.59340852 218979.25183253 293793.88610029 211568.46261197
  91543.3749705  265974.97316117 142666.18404089 118913.21415751
 143527.83158995 279098.88504763 269498.96336423 301817.47032472
 184208.51125745 165292.03489123 150263.65827628 213833.2655703
 249187.03928093 224043.18809488 274283.18900056 245886.41969786
 111947.29004332  99646.03059904 195321.81008127 208166.59839861
 155216.01959805 218425.18368557 212750.54097645 122368.11310639
 162757.63878089 222169.88155819 147014.49376385 110368.43566673
 319899.82904573 298351.87124067 337032.83390538  88509.07429469
 146988.69599587 227389.81107481 122594.14817476 205282.51397642
 217502.30210567 105979.83992047 133900.83294549 220401.61669592
 134533.75331447 212537.80466928 145822.85887485 292001.916382
 183929.46525771  97526.47834009 325926.21770803 168336.1131855
 123085.1220771  134220.19527173 309659.61248556 134621.72040037
 281856.50464034 130952.27132846  90896.53528967 161151.44855318
 148245.57782647 158743.4357484  178070.85215267 257294.2940174
 115127.62197178 137952.99415548 238164.54430471 318660.27941592
 104401.30865868 193472.26032648 172525.52490827 144821.95912031
  95416.55962006 251000.87164896 304342.18153791  53250.32972054
 284216.20870438  93659.45817633 138032.39579121 164085.60702787
 209625.15967015 203555.39716468 176647.91330201 250327.41303183
 142665.14167133 134528.78912805 171693.9768114  344207.91341043
 180844.35359567  99562.63214138 131826.54405953 102299.72625508
 123291.80105031 193395.9252095  156605.04243354 269829.46043183
 214190.70831182 151984.7953014  116934.31887546 245011.78106252
 134633.66293969 264929.22303568 302668.94028882 114015.05460455
 146441.77913368 141357.0027005  159538.24431022 231676.9800001
 233400.52995637 140373.71019287 356288.93969829 125341.52789164
 236478.64332758 118544.29428475 104052.33673557 158273.87407569
```

```
 146372.17687286 414566.38246119 337627.39928394 292117.18150842
 282078.91200937 275067.19989016 322612.66136864 307637.10995205
 202350.23801452 142183.78896605 157065.48055881 264276.61742703
 208753.75591994 148579.5100289  104862.16487731 150591.38593666
  99772.55251593 298096.55815911 342588.01923766 255305.01771014
 143960.71988075 195119.05777514 128666.72297026 264711.22067369
 123081.58596766 133276.65106096 307682.56638111 167370.72621835
 173611.40294569 566922.89973793 182845.90985561 142761.38958302
 195509.43613916 108024.21249371 181468.4749134  326585.34966799
 138869.36625264 197426.00590137 116165.04575965  56284.3053901
 200955.88091972 274048.32217841 110417.34567976 196135.43072017
 234372.28691563 121145.90339885 137720.10310105 289202.11769827
 200172.92694974 364189.74509201 116030.51909338 125795.16223833
 225786.51294502 172227.79801754  93992.80359347 178447.60844839
 218222.12411208 225543.68291325 131988.93700397  67715.73592553
 243465.88203249 138352.27496145 113593.06302933  85459.22003663
 207984.89664431 171931.87360976 270275.52596967 254063.71300068
 103617.70924263 221017.52163077 184415.52627559 170423.76300457
 156753.51512201 225827.88009121  85239.58779949 153124.07150444
 180115.88285171 200374.55472456 181098.18784203 225242.57395942
  74711.27252524  80969.30672958 139758.57667939 230578.094454
 153170.20501125 322971.46711662 196110.76794079  57211.01436041
 206435.70181525 110001.24115869 196124.45986226  98149.47173074
 222762.29559005 211717.28829211 199106.79860015 111408.67197075
  98357.53762727 236789.84151717 184747.58056935 189104.68624707
 265561.92131553 247902.83307981 108699.68852259 162096.8708319
 267886.69464291 116383.15554858 220330.53931432  99180.31360352
 227632.6485986  158389.66441237 107719.45784191 219484.51001045
  81609.43998025  76387.37318686  98141.72708677 303721.3408993
 146369.01233242 186764.52109091 241560.14901282  59680.7506418
 144391.58352994 108989.53482394 440759.11465924 120203.96498623
 115464.82503806 275628.10476953 201938.278101   234381.86154612
 201279.3818041   99150.2336282  143911.25283385 211637.42058795
 166047.47184034 162071.19891851 150358.98354071 191641.12002927
  96701.13169592 167691.2497733   81094.67813886 169629.50142262
 175660.21784513 165627.98345084 133129.35910636 205456.16714335
 236574.8313558  124117.15926504 173545.52840021 259173.50874913
 227170.15541242]
```

In [131]:

```
mm_mse=metrics.mean_squared_error(y_test,mm_y_pred)
print("MM_MSE: ",mm_mse)
```

```
MM_MSE:  1462257073.067539
```

Step-8: [Predict using SGD Regressor]

In [137]:

```
from sklearn.linear_model import Ridge
ridge=Ridge()
ridge.fit(ss_X_train, y_train)
ridge_y_pred=ridge.predict(ss_X_test)
print("Predictions of scaled data using RIDGERegression:", ridge_y_pred)
```

```
Predictions of scaled data using RIDGERegression: [257675.21878185 113319.44
691504  82426.8531785  204185.1476173
 209045.46343099  68247.80522325 235155.38604566 205503.82963227
 187881.60410267 234648.2723541  101547.26527193 304353.17716048
 102244.05444769 289088.50502266 204582.39746895 142463.70475451
 249432.70909352 147995.52146504 210440.95650059 277361.76968387
  91799.36799214 154078.55127001 160654.83612768 242931.22564738
 372489.23504359 220208.69536864 117278.33310713 100997.12063593
 318540.11892486 167965.2209004  258093.76335433 201355.68039411
 152601.21900182 142818.85647732 205237.97721647 382109.00518695
 125395.55149795 141881.80328848 273430.17974262 214039.79419147
 170520.03533111 124591.61768455 197672.02718268 375954.1016466
 148098.02572079 288062.51816821 108915.34295719 218358.99746359
  88769.12745814 295086.71345047 125832.02779986  53284.92422248
 133564.26153111 173629.04760403 214760.42297413 110902.98459738
 101891.550959   198673.11440851 139894.39453325 312959.35687414
  72818.15169921 185142.86789171 154286.30326483 299281.03359654
  77995.16988274 324301.69632026 165439.67719605 112530.91274318
 200968.52627589 206525.93237525 130939.22496303 271434.7949705
 119551.91193641 143728.82061829  75604.7283685  137013.29517687
  33816.54547949 164445.67704079 273719.15387422 116159.73585
 319877.76231905 219018.15943417 293740.58169924 211601.53340095
  91810.39803786 265860.0545862  142574.50436753 118832.45309803
 143490.54135189 279091.99937368 269438.09318135 301727.0838559
 184260.19536797 165355.29697745 150402.01619987 213872.26955603
 249128.98686698 223989.19219631 274251.05988757 245804.43504767
 111881.25459955  99657.6684917  195351.843636   208088.87541297
 155241.91929192 218296.93530891 212723.44756403 122493.6195012
 162753.82866948 222280.32500486 147011.07500783 110485.76657163
 319797.82747868 298192.33252995 336896.11445992  88449.90188413
 147012.80696412 227357.7821218  122782.38544341 205253.13054942
 217454.40610731 106373.45060735 133924.70269951 220426.97307487
 134640.28729071 212529.58483616 145754.79546682 291955.76032706
 183985.4227017   97516.00488861 325750.23832938 168326.36414289
 123155.64715005 134243.12101603 309487.81627873 134825.17771283
 281798.42989226 131024.95333871  90891.10868951 161312.88862556
 148185.8086024  158832.21037152 178093.73892855 257196.25098379
 115359.01609269 137973.8030797  238096.46896355 318583.45170274
 104557.55027164 193527.73725699 172473.39703614 144873.93782285
  95413.44014972 250979.92235412 304386.73933491  53396.59501557
 284192.8488887   93736.10626371 138194.51155661 164254.02011829
 209645.68545968 203581.05628313 176722.99877821 250445.90488473
 142969.18534636 134430.83650565 171744.92586304 344023.39522953
 180753.26425631  99560.74939251 131819.63617026 102357.91136512
 123300.43406927 193326.74075568 156536.83286561 269714.75179903
 214156.90598448 151993.77094628 116844.11984613 245019.77978004
 134481.33571312 264970.88988893 302605.10278253 114021.17186964
 146357.9032842  141375.13993318 159580.86215091 231688.26686353
 233379.53237891 140335.8976531  355923.26629173 125357.08261428
 236485.66216396 118546.58794596 103997.83103018 158348.18386585
 146404.50923854 414326.42981802 337601.15230749 292114.19281039
 281965.24115932 275008.86673536 322470.74877389 307590.60151978
 202384.41790098 142155.18578461 157091.26592771 264471.64678168
```

```
 208771.11633017 148619.91185316 104979.91970891 150554.01283765
  99780.50536918 298138.25238692 342503.09455521 255299.97310966
 144102.04203218 195288.84808526 128741.89941119 264712.10679388
 123149.85952644 133383.73579262 307718.7387752  167447.66155421
 173663.67862687 566582.90281067 182807.7303847  142843.12920552
 195579.34582492 108063.13136707 181423.74562746 326529.68398885
 138803.32151389 197434.255703   116338.70811335  56237.88388486
 200933.98976955 274032.03726347 110461.03399902 196192.78607164
 234311.66941853 121227.36028319 137677.35363422 289094.10836805
 200241.44729841 364243.2360422  116148.45662234 125735.16715732
 225728.80302225 172125.20704591  93908.44203237 178502.91106308
 218201.52458413 225590.31747515 132024.10672384  68050.20256714
 243494.78034445 138468.38236919 113566.65522356  85725.25106767
 207868.74329709 171866.32278808 270298.11674733 254013.72023277
 103627.1400818  220957.03549178 184580.89602946 170472.69404595
 156785.52169054 225825.55992986  85280.13810752 153050.37942766
 180050.65929795 200384.93269123 181113.17758844 224964.61191758
  75046.69334963  81308.12816122 139539.33310473 230487.45753918
 153119.2304394  322863.89774318 196101.65764331  57263.71208696
 206459.88513474 110053.81283205 196144.02465435  98305.6048353
 222735.34717952 211697.38685585 199162.05981537 111396.04724415
  98348.92970975 236815.35163547 184844.02588442 189166.78254955
 265587.28778557 247788.45260683 108708.52120735 161861.33079842
 267766.17648281 116324.51001268 220266.67149077  99400.4959219
 227647.57926513 158312.53989804 107884.78434848 219379.01864536
  81686.8015301   76534.96121447  98075.60330156 303686.040245
 146486.76179363 186618.33317125 241533.92035723  59640.09823959
 144393.63930317 109118.33846125 440552.78851163 120327.48097846
 115465.57529702 275652.37440942 201955.95911071 234437.34242956
 201529.99666749  99168.88155218 143697.5348757  211618.04210008
 165907.65506537 161959.66051615 150465.40969877 191644.02290767
  96598.42204402 167802.18677863  81139.99752567 169516.79470715
 175712.82289076 165541.55099882 133074.45034194 205435.66998365
 236434.36502101 124172.38788062 173618.35190103 259113.79460876
 227044.71362431]
```

In [138]:

```
ridge_mse=metrics.mean_squared_error(y_test, ridge_y_pred)
ridge_r2=metrics.r2_score(y_test, sgd_y_pred)
print("RIDGE_MSE:",ridge_mse)
```

```
RIDGE_MSE: 1460159352.8334732
```

Step-8:[Predict using Lasso Regression]

In [140]:

```
from sklearn.linear_model import Lasso
lasso=Lasso()
lasso.fit(ss_X_train, y_train)
lasso_y_pred=lasso.predict(ss_X_test)
print("Predictions of scaled data using LASSORegression:", lasso_y_pred)
```

```
Predictions of scaled data using LASSORegression: [257688.62013959 113378.58
465196  82125.58773807 204198.15312149
 209065.20183588  68333.4505404  235164.79039653 205505.05074323
 187882.78746063 234806.22623178 101553.51257063 304551.86459844
 102145.40386494 289084.77630439 204543.18358193 142360.3370027
 249536.11110157 147934.36397822 210420.41636048 277453.51435474
  91678.45623792 154036.86887337 160587.73458632 243004.77246411
 372536.84391976 220241.62560448 117186.1223909  101003.96536579
 318545.33279513 167925.68428499 258125.79443704 201463.52004961
 152728.1838202  142867.20016968 205242.72249516 382283.97348121
 125522.18645987 142020.04538444 273534.4953196  214089.81324338
 170485.83414843 124633.49877711 197871.93843483 376265.92884073
 147995.70644434 288019.1649188  108970.96620391 218352.49971956
  88634.01968863 295108.48724822 125887.35758567  53354.33354751
 133516.89521321 173750.53363401 214788.32097651 110840.53260426
 101827.26212896 198530.46990671 139852.91074653 313003.47787239
  72827.58959738 185222.75069929 154215.10249121 299384.16521557
  77904.49460825 324321.74540153 165361.81443884 112408.3131086
 200979.68673393 206533.31618919 130825.03513875 271433.5449031
 119509.53326636 143667.20877291  75631.52556976 137075.09979024
  33766.49559668 164374.8003249  273764.13434892 116273.95662899
 319907.37378891 218989.09078992 293789.24514065 211573.96613551
  91558.14898242 265953.36165254 142662.80829578 118901.58893461
 143527.9592124  279090.09000263 269478.61298456 301816.83827198
 184211.70269883 165298.70609582 150263.22247075 213824.03095807
 249179.45503397 224040.71856142 274279.44464317 245880.25791297
 111927.32519896  99650.82645988 195317.47097811 208166.33620574
 155210.90878372 218408.94422031 212741.9279616  122379.59274847
 162770.87408594 222186.42563575 147021.07584806 110375.39358199
 319885.95735682 298345.52248615 337035.68887068  88495.998677
 146987.66079042 227379.47058626 122598.96739588 205269.62150047
 217484.74886369 106043.46178396 133893.29139471 220404.73289108
 134555.16531598 212525.84070734 145820.14282828 291996.4279668
 183934.41101753  97527.80255768 325905.32343376 168343.83834281
 123100.38620811 134217.6544538  309650.39486469 134643.4773011
 281854.37429905 130975.01269917  90880.92796908 161167.95897175
 148219.16751927 158749.43489295 178081.22099081 257286.78234428
 115141.66109797 137950.2320763  238148.11603187 318674.94243973
 104413.286988   193465.04789711 172525.57607556 144819.9737262
  95425.75807521 251003.05685643 304351.70910507  53251.93820092
 284217.83001295  93678.02309351 138032.68193313 164115.558266
 209618.45430708 203524.8360259  176648.73662955 250325.97075182
 142688.04586982 134515.26499341 171694.2909658  344210.8347062
 180841.10835502  99560.49389852 131824.53363476 102302.57851104
 123280.34060445 193386.39620874 156599.00884686 269816.55670553
 214183.30828468 151973.22976189 116936.1643493  245009.75937293
 134618.35552761 264930.73163683 302675.63201062 113992.89904413
 146427.66363113 141378.48015036 159523.31772079 231677.18632706
 233396.33822941 140351.13810334 356267.10134726 125332.82602954
 236485.26506658 118556.85786569 104044.66792211 158280.11237668
 146374.11045915 414539.02191079 337621.10761643 292108.6741179
 282057.03668361 275070.99816959 322595.73971157 307639.27490633
 202345.04462546 142179.49546746 157067.38197623 264261.75501668
```

```
 208757.42150912 148581.2049137  104894.9018197  150557.07288602
  99772.21233888 298103.30051494 342587.96383351 255305.46908163
 143969.56797948 195137.70884501 128658.42456812 264720.76628244
 123075.8790375  133307.35587899 307686.03416163 167382.83272523
 173601.19446614 566921.0437147  182822.31094791 142764.03953833
 195531.70504567 108015.87290443 181467.99300921 326588.48266188
 138858.71349536 197415.15750243 116182.03125051  56258.57716971
 200953.13586729 274061.96879679 110413.66996243 196139.05884004
 234362.11762526 121142.27717003 137702.59969162 289199.73090026
 200172.118016   364178.72114027 116052.9293351  125781.68709592
 225780.57290533 172216.92570944  93989.71670705 178450.84807943
 218215.15293142 225546.2807888  131994.42829343  67756.62287051
 243475.20950411 138365.97349214 113581.50171127  85480.10646547
 207982.43653799 171927.93341116 270263.9054971  254070.69732869
 103615.40758428 221017.71326298 184416.97614974 170432.18460622
 156747.7773958  225831.20173693  85245.79939442 153113.26575346
 180108.33054701 200380.28175048 181098.53077491 225216.09603361
  74728.43680416  80998.56939577 139712.75578641 230592.0537597
 153151.7104358  322961.6895183  196110.83709213  57216.86656741
 206435.30783536 109981.47185922 196129.98241145  98153.82640961
 222767.20254311 211717.75458395 199111.80902732 111396.89547964
  98352.2391905  236796.65315528 184756.87709546 189102.87066984
 265570.91944503 247899.86162657 108685.02051787 162076.89947812
 267884.34987077 116381.78616191 220346.95746255  99189.62577855
 227628.4194206  158384.67433019 107758.09877273 219475.05629725
  81625.68230433  76408.39207469  98133.03644197 303724.25285605
 146366.73207021 186764.1964953  241544.34002608  59672.29590187
 144380.42074043 109012.12400539 440727.47005544 120213.58345962
 115472.35464142 275632.27701198 201946.74478173 234386.45982546
 201307.35147765  99144.32636216 143872.89037379 211625.045532
 166040.99627102 162069.79246025 150358.77043925 191633.7064664
  96693.57742659 167708.33352569  81114.75400638 169603.90099709
 175665.63917404 165626.15313319 133123.76866151 205441.38355818
 236560.73690896 124109.04491964 173553.09431305 259164.7081336
 227167.23595558]


C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:491: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterati
ons. Fitting data with very small alpha may cause precision problems.
  ConvergenceWarning)
```

In [141]:

```
lasso_mse=metrics.mean_squared_error(y_test, lasso_y_pred)
print("LASSO_MSE:",lasso_mse)
```

```
LASSO_MSE: 1462124246.4736216
```

Step-9:[RMSE]

In [142]:

```
import numpy as np
#RMSE without CD
print("RMSE without CD: ",np.sqrt(mse))
#RMSE with CD
print("RMSE with CD: ",np.sqrt(mse_cd))
#RMSE with CD and Standard Scaling
print("RMSE with CD and SS: ",np.sqrt(ss_mse))
#RMSE with CD and MinMaxScaling
print("RMSE with CD and MnMaxScaling: ",np.sqrt(mm_mse))
#RMSE of SGDRegressor with CD and StandardScaler
print("RMSE of SGDRegressor with CD and StandardScaler: ",np.sqrt(sgd_mse))
#RMSE of Ridgecv with CD and Standard Scaler
print("RMSE of Ridgecv with CD and Standard Scaler: ",np.sqrt(ridge_mse))
#RMSE of LassoCV with CD and StandardScaler
print("RMSE of LassoCV with CD and StandardScaler",np.sqrt(lasso_mse))
```

```
RMSE without CD:  0.5042993710845911
RMSE with CD:  38239.47009395262
RMSE with CD and SS:  38239.47009396888
RMSE with CD and MnMaxScaling:  38239.47009396886
RMSE of SGDRegressor with CD and StandardScaler:  37717.93953376125
RMSE of Ridgecv with CD and Standard Scaler:  38212.031519319586
RMSE of LassoCV with CD and StandardScaler 38237.7332810618
```

In [ ]:

```

```