NAME : Arul kumar ARK

ROLL NO : 22522903

LAB : 4

House Price Prediction Using LR With Regularization

Step : 1

In [1]:
```
import pandas as pd
df=pd.read_csv('Ames_House_Sales_Cropped.csv')
```

In [2]:
```
df.head()
```

Out[2]:

	BldgType	CentralAir	1stFlrSF	2ndFlrSF	3SsnPorch	BedroomAbvGr	BsmtFinSF1	BsmtFinSF2	BsmtFullBath	BsmtHalfBath	...	OverallQual	PoolArea
0	1Fam	Y	856.0	854.0	0.0	3	706.0	0.0	1	0	...	7	0.0
1	1Fam	Y	1262.0	0.0	0.0	3	978.0	0.0	0	1	...	6	0.0
2	1Fam	Y	920.0	866.0	0.0	3	486.0	0.0	1	0	...	7	0.0
3	1Fam	Y	961.0	756.0	0.0	3	216.0	0.0	1	0	...	7	0.0
4	1Fam	Y	1145.0	1053.0	0.0	4	655.0	0.0	1	0	...	8	0.0

5 rows × 39 columns

In [3]:
```
df.shape
```

Out[3]:
```
(1379, 39)
```

In [4]:
```
df.size
```

Out[4]:
```
53781
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['BldgType', 'CentralAir', '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', 'SalePrice'],
      dtype='object')
```

In [7]:
```
type(df)
```

Out[7]:
```
pandas.core.frame.DataFrame
```

In [8]:
```
df.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 [11]:
```
df["SalePrice"].value_counts
print(df)
```

```
     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
5       1Fam          Y     796.0     566.0      320.0             1
6       1Fam          Y    1694.0       0.0        0.0             3
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
```

Step : 2

In [14]:
```
d1=df.drop(["BldgType","CentralAir"],axis=1)
print(d1)
```

```
      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
17      1296.0       0.0        0.0             2         0.0         0.0
```

In [17]:
```
#Prepare X matrix(36 feature columns) and y vector (salesPrice column)
X=dff.drop(["SalePrice"],axis=1)
print(X)
```

```
17      1296.0       0.0        0.0             2         0.0         0.0
18      1114.0       0.0        0.0             3       646.0         0.0
19      1339.0       0.0        0.0             3       504.0         0.0
20      1158.0    1218.0        0.0             4         0.0         0.0
21      1108.0       0.0        0.0             3         0.0         0.0
22      1795.0       0.0        0.0             3         0.0         0.0
23      1060.0       0.0        0.0             3       840.0         0.0
24      1060.0       0.0        0.0             3       188.0       668.0
25      1600.0       0.0        0.0             3         0.0         0.0
26       900.0       0.0        0.0             3       234.0       486.0
27      1704.0       0.0        0.0             3      1218.0         0.0
28      1600.0       0.0        0.0             2      1277.0         0.0
29       520.0       0.0        0.0             1         0.0         0.0
...        ...       ...        ...           ...         ...         ...
1349    1048.0     510.0        0.0             3       580.0         0.0
1350     804.0       0.0        0.0             2       510.0         0.0
1351    1440.0       0.0        0.0             3       678.0         0.0
1352     734.0    1104.0        0.0             4         0.0         0.0
1353     958.0       0.0        0.0             2       958.0         0.0
1354     968.0       0.0        0.0             4         0.0         0.0
```

In [19]:
```
y=dff["SalePrice"].values
y
```

Out[19]:
```
array([208500., 181500., 223500., ..., 266500., 142125., 147500.])
```

In [20]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25,random_state=42)
```

In [21]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,y_train)
```

Out[21]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [23]:
```
y_pred=reg.predict(X_test)
print(y_pred)
```

```
[257434.93050745 111083.7347476  100018.05832303 204028.53821314
 207319.25418312  38036.30928932 234153.38582868 205076.12689608
 187014.1265524  235636.78799031 100976.50943769 304119.53646983
 101769.64600713 288758.46001593 204767.89338328 145002.47279616
 248322.97436852 151533.68038634 209697.39458712 278312.23574148
  93329.85601474 153784.14868375 163237.87451138 241495.9121289
 372429.94726945 221672.07367609 119658.75685737 100039.78137073
 319435.0289208  172088.64665031 258772.64470117 199597.36129642
 156349.69283212 142311.22500101 204807.33161321 379608.17785443
 124176.23377551 141127.25006141 273832.43247645 212547.04205553
 169474.57176142 123305.65719311 200341.24016791 377957.32307114
 146941.7171239  283693.48760668 106694.19786718 221733.02172508
  87733.47014004 295005.66546305 125250.62019285  53569.11256022
 133831.0508612  170901.4292971  213273.17757301 110818.14723232
 103893.81849706 200665.21921033 137764.84464902 311840.64900748
  53035.52267594 184866.47797858 154675.79963879 298477.53600006
  78162.5407468  323728.93535803 169417.44138514 112718.07937635
 200547.87648954 205739.30910765 130292.88475647 271581.37221152
 119489.76459232 126635.29232505  77972.26780485 138563.37305272
  35555.51251487 163787.98080893 271929.13690186 115260.62742896
 319958.55004404 218515.16541529 291730.97232824 211176.53228423
  93761.81278837 263031.10150322 141886.3966699  118491.50416092
 143580.34191288 278395.85370477 265558.84177945 300839.51324225
 182673.3443797  168604.5492604  148322.24627425 213781.03244613
 247750.58741    221693.90686786 274474.96290575 242912.73279161
 111041.26737678 100160.66573767 193531.44933265 206050.88712632
 154584.47386041 217440.52365094 210904.23539265 125864.27688695
 171322.66919858 224459.70210717 155548.12334499 111942.7871775
 316624.83474109 298116.31992277 335573.77762719  86738.88506816
 146692.58771903 228471.5403292  123374.07354542 204867.33148103
 217343.21720916 120569.43114727 134354.1238425  223860.01643704
 138360.76267459 210765.75439351 145423.97229834 291155.26974376
 183089.17904138  98716.55524167 323896.46706846 170447.30657102
 123643.15346299 134345.24799853 309423.49789818 136836.97808937
 282083.02092584 132621.2237255   90821.15043377 164193.14328588
 147333.60026797 162030.92949963 177621.20328218 254054.45691355
 119092.92298357 140081.84030886 236509.39317212 318578.43961646
 105538.81578042 192023.58779383 174531.98294663 146314.75164225
  95146.46236358 250936.32825846 304056.57810773  55008.42319119
 281810.05395732  93402.68236907 138283.4460375  166415.3192293
 209758.26998556 205602.16976218 177026.58819395 248435.62725108
 145190.55457313 133283.94785764 171593.4339452  344925.23412401
 181194.80288331  98431.28117762 131287.46561773 110849.96802284
 131283.81966769 194221.20494195 156966.39268757 268340.58471943
 211804.83660278 151971.72881374 115946.37551381 243618.89320955
 136292.7606826  265335.41730193 302175.59064213  85054.45272476
 146007.43532809 150439.28821412 159145.83079286 232246.23120855
 231571.13532009 139178.99313677 355030.619324   127659.29864106
 235339.23724177 116707.21916901 103103.75191051 159574.23418457
 149485.12469048 408898.40096809 335368.65026123 290500.91914899
 281845.43486756 275341.8171841  321705.52591338 307549.31796899
 201661.19536934 142047.81494961 157876.45367574 258695.10810332
 207592.92972056 146121.60182471 107032.8224214  151245.32235213
 100804.2036383  296980.23153195 341413.30991677 255158.86136987
 147006.3495448  199721.99255174 127291.41093944 264030.58756771
 121760.19396639 134138.79148625 307661.79229119 170331.68886912
 173593.89056669 567240.20119424 182277.72265598 144159.48526101
 199515.17896325 107691.20503754 181383.00130498 328401.72331
 139225.20214766 197660.10401731 116340.37479629  54949.57465389
 199375.67314461 272915.38777822 119902.62390774 197573.81604289
 231881.98496935 119555.98822715 128709.5470232  289158.80585494
 200733.61408622 364228.39366113 119308.94568436 124086.7225403
 225161.62739518 171005.22659817  93631.91767228 178506.48207587
 215661.64584809 226225.30038962 132391.85608602  73443.41948308
 242913.00968351 141042.39030912 114633.38186071  89093.07024776
 208503.15479642 172828.98634139 270797.22136173 253693.44924276
 103518.85057198 220521.27460707 185034.32129269 169518.90296985
 154846.17459061 224605.38839967  85972.23828462 151323.12661355
 180628.92554331 200106.98029834 179499.8172794  223755.69902625
  73639.52734133  86011.8149164  110000.03909838 231518.89807261
 153369.35976787 320899.35432162 194412.83170324  57741.65743007
 203877.63015232 109803.05085704 194633.36256922  99160.35922065
 230240.81594374 211484.77477044 198342.97693781 109749.28794511
  95564.23930705 235396.44527254 184775.90385427 188806.96222903
 265328.01705249 245862.13668724 107422.00402518 164786.63106786
 268303.4442141  116075.01592356 222914.33666726 102600.4329218
 226400.49839176 156093.55758651 112003.82907499 218089.37220887
  81085.92660263  85015.51341434  98208.34263398 304262.27895919
 143757.79043406 195367.3512202  241835.21911123  60944.84144938
 143731.15831778 110459.58351734 438412.24654518 123265.07239636
 116119.87930811 277961.71617821 201687.81739715 237947.02580917
 206117.84987333  99453.60459331 115124.57035037 209795.31353811
 167537.15563245 162970.26517749 151279.43057796 189724.86274087
  96455.47006118 168652.82586918  83051.31596953 171340.42959854
 176868.33711878 175644.68961803 131414.67048231 204543.74960377
```

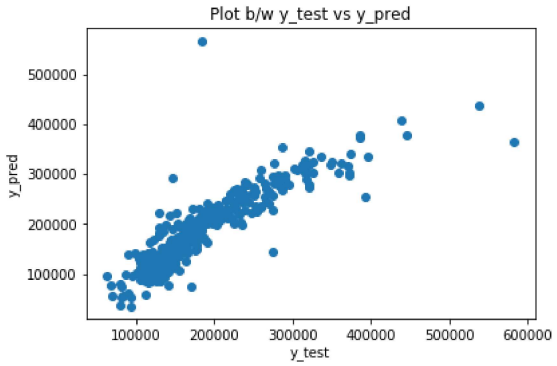
```
233147.40044683 125478.90203711 175060.50167495 258877.30470763
229115.65129666]
```

In [24]:
```
import sklearn.metrics as metrics
mse=metrics.mean_squared_error(y_test,y_pred)
print("MSE without Categorical: ",mse)
```

```
MSE without Categorical:  1474827325.5975406
```

Step : 3

In [25]:
```
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

In [27]:
```
encoding=pd.get_dummies(df)
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
17      1296.0       0.0        0.0             2         0.0         0.0
```

Step : 5

In [29]:
```
X=encoding.drop(["SalePrice"],axis=1)
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
17      1296.0       0.0        0.0             2         0.0         0.0
```

In [30]:
```
y=encoding["SalePrice"].values
y
```

Out[30]:
```
array([208500., 181500., 223500., ..., 266500., 142125., 147500.])
```

In [31]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25,random_state=42)
```

In [32]:
```
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:  1461036570.1436336
```

Step : 6

In [33]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
ss_X_train=ss.fit_transform(X_train)
ss_X_train
```

Out[33]:
```
array([[ 0.39851037, -0.79290427, -0.11340519, ...,  3.45325933,
        -0.22777619,  0.22777619],
       [ 1.57467708, -0.79290427, -0.11340519, ..., -0.2895815 ,
        -0.22777619,  0.22777619],
       [ 0.37564751,  0.70143387, -0.11340519, ..., -0.2895815 ,
        -0.22777619,  0.22777619],
       ...,
       [ 1.22157303, -0.79290427, -0.11340519, ..., -0.2895815 ,
        -0.22777619,  0.22777619],
       [-1.11297817,  0.90510323, -0.11340519, ..., -0.2895815 ,
        -0.22777619,  0.22777619],
       [ 0.11145456, -0.79290427, -0.11340519, ...,  3.45325933,
        -0.22777619,  0.22777619]])
```

In [34]:
```
ss_X_test=ss.transform(X_test)
ss_X_test
```

Out[34]:
```
array([[ 0.85830772, -0.79290427, -0.11340519, ...,  3.45325933,
        -0.22777619,  0.22777619],
       [-0.64810018, -0.79290427, -0.11340519, ..., -0.2895815 ,
        -0.22777619,  0.22777619],
       [-0.21624632,  0.76093278, -0.11340519, ..., -0.2895815 ,
         4.39027446, -4.39027446],
       ...,
       [-0.04350477,  0.37647826, -0.11340519, ..., -0.2895815 ,
        -0.22777619,  0.22777619],
       [-0.64301955,  1.46805451, -0.11340519, ..., -0.2895815 ,
        -0.22777619,  0.22777619],
       [-0.29499614,  0.81585486, -0.11340519, ...,  3.45325933,
        -0.22777619,  0.22777619]])
```

In [36]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(ss_X_train,y_train)
ss_y_pred=lr.predict(ss_X_test)
print(ss_y_pred)
```

```
[257746.67634751 113257.14215097  83086.83449793 204109.44696503
 209062.51895279  69091.78945845 235181.17333175 205722.26326666
 187627.82768487 234276.07879635 101443.5208025  304189.5355825
 102111.08803612 289185.34424259 204499.21833429 142096.29751997
 249636.03306251 147881.8828702  210574.13858086 277457.62922997
  91416.02948336 153937.03347396 160626.96732406 243147.94924865
 372541.07042985 219861.59417986 116849.37194182 100880.38852652
 318604.3072889  168091.93524469 258085.96044038 201316.99711925
 154599.59088249 142746.1064703  205138.6701111  382254.53304499
 124930.2956851  141597.13070899 273664.48982728 214149.35804835
 170457.21544846 124570.77597839 197619.98987752 376155.99238854
 148003.84323244 288043.80788135 108672.5169885  218565.03839169
  88455.07127747 295191.90839717 125798.92162385  54900.46063498
 133431.33010679 173734.6432617  214868.38940441 110305.00323981
 101777.6571833  198417.39484009 139859.94217493 313011.39577057
  71373.41173384 185200.29760901 153952.30442378 299603.66711185
  76797.97905956 324386.66707621 165286.38031963 112128.08996347
 200841.50826584 206567.94099559 130662.03800731 271533.51448715
 119169.71075355 144780.32426223  75275.27013335 136814.73927026
  35548.77310788 163999.4816694  273975.35104906 116033.21218901
 319960.29175167 219051.51124482 293875.7312285  211801.06700944
  91269.29114983 265733.72817127 142679.76528577 118577.18340652
 143674.08252179 279159.65322884 269371.00847357 301749.6475942
 184234.99923566 165120.97530255 149749.83444921 213942.08827205
 249106.5262464  224105.75907123 274402.65899366 245923.74180536
 111650.50965645  99272.63102469 195172.55269882 208067.81804138
 155001.86752871 218149.55729734 212687.26428796 122414.02996443
 162732.20657544 221859.13237471 146948.24111576 109948.53523129
 319916.30228215 298406.39751083 337078.1346426   88053.83690322
 146957.67256334 227538.58670511 122778.70033749 205374.95008286
 217590.01872639 107383.84101482 133741.2940345  220233.83033811
 134541.44321812 212453.09527909 145667.19230495 291995.67805595
 183987.92894577  97709.62989828 325909.11123175 168153.0537991
 122835.11047077 134056.79102667 309609.27342891 134547.87576178
 281848.2892548  130902.414594    92709.96908775 160928.91702672
 148231.56791932 158731.16111448 178201.93181948 257204.46954565
 114890.4575372  138122.68344433 238125.08207166 318625.89235561
 104214.83201718 193580.28738241 172127.95271212 144833.00763331
  95559.79845962 250974.08034067 304496.84969754  53072.22208749
 284199.95101748  93322.70762404 137832.3476692  164298.42069277
 209757.60013178 203537.90106371 176604.23461749 250435.04474144
 142419.38528636 134385.83846204 171734.49004071 344239.73144709
 182540.85994093  99608.62865269 131558.48391034 102401.37979263
 123453.73975651 193361.22440516 156609.55466075 269877.79864932
 214063.10991695 151945.88694257 116602.06046004 244961.96222125
 134853.80033622 264897.14158982 302718.95386478 112752.48763543
 146428.01022778 141383.04049711 159106.57051208 231871.86707197
 233361.12369078 140206.8801076  356203.87108972 125107.24732751
 236557.13984092 118583.44962369 103727.34373498 158205.16435346
 146206.40366386 414475.89127582 337712.90757194 292098.02863639
 282089.2821333  275216.7535936  322615.55168033 307605.46888381
 202379.61396859 141568.5679281  156911.80391702 264235.34590098
 208838.68106865 148493.50013207 104577.21175208 149737.50802682
 101508.22184593 298194.31400841 342555.6575081  255293.49747035
 143932.11991761 195052.21568926 128768.16960182 264721.74453629
 122744.14619783 133171.58386505 307876.67207273 167186.04489823
 173665.54160608 566793.1709959  182882.72846953 142976.55390157
 195389.49394275 110000.52031067 181425.05998225 326697.98513269
 138753.30659611 197531.51535532 116224.68828072  57916.66921872
 200892.30994253 274120.15715879 110287.72265007 195830.60427729
 234290.62236658 121029.23351415 137768.1152297  289104.52688668
 200107.84920963 364178.66342092 116071.36887504 125706.22322028
 225918.36241065 172279.20082007  93552.9758358  178432.96188201
 218235.04083372 225391.98842006 133701.6669314   67245.43233413
 243425.96315121 138199.03345666 113586.6774028   84741.3662601
 207895.82887217 172018.96542022 270379.96625141 254301.48609865
 103491.65134387 221152.52727064 184240.39768591 170413.54111433
 156767.28870157 225874.2187855   85067.50220226 152614.17988863
 180110.39735029 200365.40564938 181240.47868432 225088.28569024
  74809.64993437  81056.03121343 138531.42801458 231176.80557566
 152880.69516053 322880.51400012 196244.29857021  56720.24928342
 206525.2911202  109558.50672761 196173.16696132 100008.43053437
 222772.90040602 211538.69107333 199089.61402063 111290.16182517
  98159.20986117 236839.06037874 184863.02084498 189201.7028789
 265541.99840726 247878.42019041 108244.54720887 161881.3193246
 267831.06097132 116127.88591933 220313.03060259  98999.66742238
 227727.64329725 158059.62456719 107633.36590698 219812.49714578
  81429.50018227  76334.09584861  97716.92609735 303748.10861981
 146166.41447046 186676.02055521 241412.73122161  61413.0262216
 144287.26086056 108854.17859528 440736.67250284 120123.18222097
 115307.43887445 275889.34669779 201914.61172598 234443.75355834
 201332.72073446  98680.14396138 142896.01985585 211430.3435048
 167955.3826565  161745.94065555 150329.86791651 191659.61015824
  98638.48036051 167674.58655871  81251.41418888 169289.1270314
 175822.84359349 165634.36808713 132604.67079394 205548.45244987
```

```
236536.65155886 125744.91988448 173333.02038791 259092.37349783
227361.55928746]
```

In [37]:
```
ss_mse=metrics.mean_squared_error(y_test,ss_y_pred)
print("SS_MSE: ",ss_mse)
```

```
SS_MSE:  1461036570.1437433
```

Step : 7

In [38]:
```
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: [257746.67634751 113257.14215097  83086.83449793 204109.44696503
 209062.51895279  69091.78945845 235181.17333175 205722.26326666
 187627.82768487 234276.07879635 101443.5208025  304189.5355825
 102111.08803612 289185.34424259 204499.21833429 142096.29751997
 249636.03306251 147881.8828702  210574.13858086 277457.62922997
  91416.02948336 153937.03347396 160626.96732406 243147.94924865
 372541.07042985 219861.59417986 116849.37194182 100880.38852652
 318604.3072889  168091.93524469 258085.96044038 201316.99711925
 154599.59088249 142746.1064703  205138.6701111  382254.53304499
 124930.2956851  141597.13070899 273664.48982728 214149.35804835
 170457.21544846 124570.77597839 197619.98987752 376155.99238854
 148003.84323244 288043.80788135 108672.5169885  218565.03839169
  88455.07127747 295191.90839717 125798.92162385  54900.46063498
 133431.33010679 173734.6432617  214868.38940441 110305.00323981
 101777.6571833  198417.39484009 139859.94217493 313011.39577057
  71373.41173384 185200.29760901 153952.30442378 299603.66711185
  76797.97905956 324386.66707621 165286.38031963 112128.08996347
 200841.50826584 206567.94099559 130662.03800731 271533.51448715
 119169.71075355 144780.32426223  75275.27013335 136814.73927026
  35548.77310788 163999.4816694  273975.35104906 116033.21218901
 319960.29175167 219051.51124482 293875.7312285  211801.06700944
  91269.29114983 265733.72817127 142679.76528577 118577.18340652
 143674.08252179 279159.65322884 269371.00847357 301749.6475942
 184234.99923566 165120.97530255 149749.83444921 213942.08827205
 249106.5262464  224105.75907123 274402.65899366 245923.74180536
 111650.50965645  99272.63102469 195172.55269882 208067.81804138
 155001.86752871 218149.55729734 212687.26428796 122414.02996443
 162732.20657544 221859.13237471 146948.24111576 109948.53523129
 319916.30228215 298406.39751083 337078.1346426   88053.83690322
 146957.67256334 227538.58670511 122778.70033749 205374.95008286
 217590.01872639 107383.84101482 133741.2940345  220233.83033811
 134541.44321812 212453.09527909 145667.19230495 291995.67805595
 183987.92894577  97709.62989828 325909.11123175 168153.0537991
 122835.11047077 134056.79102667 309609.27342891 134547.87576178
 281848.2892548  130902.414594    92709.96908775 160928.91702672
 148231.56791932 158731.16111448 178201.93181948 257204.46954565
 114890.4575372  138122.68344433 238125.08207166 318625.89235561
 104214.83201718 193580.28738241 172127.95271212 144833.00763331
  95559.79845962 250974.08034067 304496.84969754  53072.22208749
 284199.95101748  93322.70762404 137832.3476692  164298.42069277
 209757.60013178 203537.90106371 176604.23461749 250435.04474144
 142419.38528636 134385.83846204 171734.49004071 344239.73144709
 182540.85994093  99608.62865269 131558.48391034 102401.37979263
 123453.73975651 193361.22440516 156609.55466075 269877.79864932
 214063.10991695 151945.88694257 116602.06046004 244961.96222125
 134853.80033622 264897.14158982 302718.95386478 112752.48763543
 146428.01022778 141383.04049711 159106.57051208 231871.86707197
 233361.12369078 140206.8801076  356203.87108972 125107.24732751
 236557.13984092 118583.44962369 103727.34373498 158205.16435346
 146206.40366386 414475.89127582 337712.90757194 292098.02863639
 282089.2821333  275216.7535936  322615.55168033 307605.46888381
 202379.61396859 141568.5679281  156911.80391702 264235.34590098
 208838.68106865 148493.50013207 104577.21175208 149737.50802682
 101508.22184593 298194.31400841 342555.6575081  255293.49747035
 143932.11991761 195052.21568926 128768.16960183 264721.74453629
 122744.14619783 133171.58386505 307876.67207273 167186.04489823
 173665.54160608 566793.1709959  182882.72846953 142976.55390157
 195389.49394275 110000.52031067 181425.05998225 326697.98513269
 138753.30659611 197531.51535532 116224.68828072  57916.66921872
 200892.30994253 274120.15715879 110287.72265007 195830.60427729
 234290.62236658 121029.23351415 137768.1152297  289104.52688668
 200107.84920963 364178.66342092 116071.36887504 125706.22322028
 225918.36241065 172279.20082007  93552.9758358  178432.96188201
 218235.04083371 225391.98842006 133701.6669314   67245.43233413
 243425.96315121 138199.03345666 113586.6774028   84741.3662601
 207895.82887217 172018.96542022 270379.96625141 254301.48609865
 103491.65134387 221152.52727064 184240.39768591 170413.54111433
 156767.28870157 225874.2187855   85067.50220226 152614.17988863
 180110.39735029 200365.40564938 181240.47868432 225088.28569024
  74809.64993437  81056.03121343 138531.42801458 231176.80557566
 152880.69516053 322880.51400012 196244.29857021  56720.24928342
 206525.2911202  109558.50672761 196173.16696132 100008.43053437
 222772.90040602 211538.69107333 199089.61402063 111290.16182517
  98159.20986117 236839.06037874 184863.02084498 189201.7028789
 265541.99840726 247878.42019041 108244.54720887 161881.3193246
 267831.06097132 116127.88591933 220313.03060259  98999.66742238
 227727.64329725 158059.62456719 107633.36590698 219812.49714578
  81429.50018227  76334.09584861  97716.92609735 303748.10861981
 146166.41447047 186676.02055521 241412.73122161  61413.0262216
 144287.26086056 108854.17859528 440736.67250284 120123.18222097
 115307.43887445 275889.34669779 201914.61172598 234443.75355834
 201332.72073446  98680.14396138 142896.01985585 211430.3435048
 167955.3826565  161745.94065555 150329.86791651 191659.61015824
  98638.48036051 167674.58655871  81251.41418888 169289.1270314
 175822.84359349 165634.36808713 132604.67079394 205548.45244987
```

```
236536.65155886 125744.91988448 173333.02038791 259092.37349783
227361.55928746]
```

In [39]:
```
mm_mse=metrics.mean_squared_error(y_test,mm_y_pred)
print("MM_MSE: ",mm_mse)
```

```
MM_MSE:  1461036570.1437414
```

Step : 8

In [40]:
```
from sklearn.linear_model import SGDRegressor
sgd=SGDRegressor()
sgd.fit(ss_X_train, y_train)
sgd_y_pred=sgd.predict(ss_X_test)
print("Predictions of scaled data using SGDRegressor:", sgd_y_pred)
```

```
Predictions of scaled data using SGDRegressor: [261575.86203383 104401.47535968  93589.62507687 205277.79618898
 209315.59620006  64095.11100681 240310.46497874 209679.18436441
 187262.75087683 234490.58858669  93186.67398421 306860.44827732
  99916.52504884 298517.22462015 205543.97025634 144560.8001904
 246342.64998292 147847.77004451 212975.13143963 279886.8282912
  95758.03947457 148673.43110284 161044.03143073 243169.24958431
 380915.48690049 222608.86299907 120018.90832985  91764.41048613
 331635.62580176 166704.4992182  264101.58257193 199360.30233265
 146076.71790852 129609.64708128 206847.93223478 392242.42098527
 112559.04555315 142181.04788766 277865.04898823 208975.04983715
 166245.59092905 117240.25956004 189680.79567426 374072.69572682
 150587.73915903 297784.4007117   97571.3110894  223250.74021615
  82821.48051254 305138.0695295  118928.9884398   39449.705145
 136446.74543439 174166.03056902 212572.28576407 103413.15114715
 100937.45328304 207449.58368069 137541.09841511 324241.37075318
  63554.61686518 180429.39871884 155365.55512557 301027.81348181
  73851.66838977 337864.42859027 176167.00108224 109546.29045204
 203072.86803955 205526.50071497 129500.63440368 272663.33341191
 112043.10211316 148543.53063166  61999.50180966 131166.75391899
  26930.60584474 165055.54338217 281580.28463657 100082.77398581
 332358.84984929 221527.38890639 301261.28750364 217312.94469078
  97913.12305227 265572.83859072 133031.75684241 105059.51705422
 135902.55633732 285021.35924768 277281.71473054 301566.65501388
 182788.83501997 169101.9622288  155214.90265617 218442.88278957
 250765.16440874 220783.34159291 279670.84800147 242760.85586546
 105509.13581785  98433.72771344 201039.13365027 207407.1273751
 155742.18572636 214303.0884518  213894.77202422 120214.51921919
 157693.31703168 233670.07341946 138241.62571319 110217.63785899
 325832.82956169 295955.15971568 342177.2645456   76613.73677759
 145150.99996111 229202.4568152  124643.76390217 201124.58869743
 215064.00994912 119142.96717731 129000.11571788 227724.77055994
 133480.40092856 210462.09817425 138308.87236583 293987.09343756
 182045.86013693  87630.32281544 326630.71584968 169920.5770093
 125367.23989413 130495.75241225 314331.62106955 134299.29104484
 282275.14866078 129015.10320977  87103.98493653 166490.4561084
 141165.76450491 159789.16025569 176694.96335294 257596.547362
 126265.04381669 135553.75532031 243166.35169196 325810.58656361
 104611.4731495  195360.95306991 169079.20558801 147057.36876146
  80907.96870213 249676.08162399 319980.2681367   47283.26544818
 294198.86213688  84315.79512242 137152.36647227 169038.3714152
 213028.4551153  210504.47325512 177587.97904546 267001.44001038
 152372.71222541 124441.28786111 171053.4548349  354545.42962083
 183479.99444907  88581.51849128 127090.09285313  96204.03441439
 112048.98231839 185967.58066876 152346.17615979 269103.36291124
 211793.9686347  149723.43361126 108911.86846428 252822.35771472
 117981.8845331  269985.46051695 307805.91200395 116229.02864773
 138526.7375121  132739.55545499 159178.07829224 240410.2391052
 235696.64567505 135975.92637104 362712.57700425 126942.60109186
 241357.33861792 109469.56117189  93706.37914108 162411.92378466
 148914.02457199 419308.79512267 352895.546426   295830.57547389
 283863.03141366 277138.44848038 322740.66016359 314261.03318615
 205615.16174169 135298.41863369 154206.0776369  341453.62104696
 206466.92940217 146358.29373092 104102.05614151 146052.18998973
  96749.92880018 302225.14587743 352557.23436692 255833.38750821
 145774.84113112 203181.7597928  128361.98074518 263656.99651829
 122528.65208265 131800.53164452 322527.23091521 170500.32232737
 173228.300628   588072.3846438  182399.24287458 146050.19479356
 201428.90787158 110700.20314564 175533.40705756 338580.87848136
 129215.80505856 195023.72847859 115667.20383782  46870.28472893
 195997.66506428 275698.45040479 105050.54999663 207495.75990219
 238330.76796095 115608.46147391 136834.62118537 290045.86653626
 205994.02015723 383469.40453924 111472.42092483 116405.38349934
 221072.75234305 162113.91849962  82083.59346569 178977.90166034
 215685.28575461 231615.25942316 136835.02642853  68597.19088649
 242314.88112861 141364.08854704 105522.52032181  93058.28924794
 204815.15307853 165851.80184115 275654.49356271 259397.3592503
  93263.56975927 219120.42998281 194580.85207729 165580.83319504
 151668.59540458 230040.26279948  77749.92312376 149471.3342081
 175345.69716488 196860.80206681 176462.75109975 215729.58927319
 138640.7236722   93151.0095794  134492.23559499 227471.08775103
 147115.93836695 324359.94700272 192706.13226653  46899.09488847
 209974.53015333 108899.15770242 197408.29557167  98227.11844152
 223265.12756186 213995.50279251 207392.2215238  110061.1131844
  94021.28565641 243671.13221527 191639.73874488 193253.96096818
 266860.02838347 242082.64565414 104367.44198253 145833.2590129
 273431.63923945 107308.90703125 219351.4132555  105348.61648331
 231815.8422397  155144.52997003 100676.47732982 221929.95987589
  74438.94391235  71739.79524838  84884.44987396 314658.67415408
 218090.01765794 175193.9562324  253182.4675157   51316.46787752
 142754.11125192 111070.27765498 458835.78253092 114022.07548388
 112666.14931759 284457.66771205 197619.08435852 235813.81198064
 219447.61128581  93277.95843783 135705.10943976 209342.49185457
 163165.81146112 146655.38212161 152964.94791716 189395.24436883
  89616.9604968  173372.29322911  68841.59992556 169197.23361504
 178125.41755917 154288.48748212 130961.84494807 202200.34642997
```

```
 234315.03091738 123317.60530338 176266.11349468 266608.86527095
 220689.47945904]
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-packages\sklearn\linear_model\stochastic_gradien
t.py:128: FutureWarning: max_iter and tol parameters have been added in <class 'sklearn.linear_model.stochastic_gradient.SGD
Regressor'> in 0.19. If both are left unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults
to max_iter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

In [41]:

```
sgd_mse=metrics.mean_squared_error(y_test, sgd_y_pred)
print("SGD_MSE:",sgd_mse)
```

```
SGD_MSE: 1505789243.6148124
```

In [42]:
```
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: [257716.40699318 113188.83783395  83441.68622524 204093.86685461
 209046.77119427  68968.95970424 235166.85073532 205709.18701907
 187608.88092307 234127.91986613 101442.61753652 303996.83085985
 102207.81104997 289201.83025972 204530.52882459 142227.09469735
 249528.85892308 147950.18211509 210586.82424173 277366.73836724
  91543.67482183 153978.54592712 160695.6139235  243057.53405206
 372495.11731184 219840.91884082 116956.95574687 100878.46795325
 318603.53970706 168136.34840742 258058.73912129 201180.97018482
 154470.04859036 142707.68415143 205133.17128488 382062.98759242
 124795.49109482 141435.74207829 273560.58911851 214075.03676925
 170494.5832031  124523.92339387 197429.91336254 375822.0370805
 148108.22483297 288075.31154386 108608.62662542 218573.47618249
  88587.3998216  295171.57452763 125737.07110059  54833.67931716
 133481.71022526 173594.36273487 214840.0746121  110380.72176421
 101840.39812262 198576.07688518 139894.58865439 312964.61939685
  71367.79689775 185112.92538069 154029.67686384 299492.62825149
  76893.88517706 324365.26359029 165376.0785674  112254.3704445
 200819.07169113 206550.44800728 130800.86405237 271523.0063293
 119196.14940404 144867.37804993  75254.15912227 136728.43245701
  35597.59046173 164058.34969126 273927.80986891 115902.45464437
 319919.85988921 219089.91634213 293822.16290467 211832.71038627
  91537.74860448 265620.40558168 142587.28712764 118497.81994793
 143635.30184965 279152.61891253 269311.1413232  301659.46840395
 184286.35253708 165184.74414324 149891.99972881 213980.4719389
 249049.04209643 224051.13974668 274369.96952527 245841.61643708
 111585.23921014  99285.36721984 195203.7216061  207989.92499748
 155028.88974981 218022.83766913 212660.79772296 122538.52182907
 162728.69874836 221971.18357146 146945.09391568 110067.72428323
 319814.93647477 298246.38587609 336941.24594611  87997.01636518
 146981.74764146 227505.54712528 122965.2544359  205344.88888641
 217541.64749768 107771.77680053 133765.67836363 220260.06460657
 134647.30943604 212445.40571604 145599.53125656 291949.61121746
 184043.36581726  97697.33647965 325733.33460483 168143.84275303
 122906.74152174 134080.67606878 309437.41884789 134751.59695772
 281790.27252733 130974.90646071  92697.37462197 161091.5892275
 148171.12579169 158819.52635399 178223.97509849 257107.00153039
 115122.62646454 138141.80491198 238057.41121163 318549.26633768
 104371.57134771 193635.25761937 172077.14595925 144885.03564327
  95554.92759806 250953.38099348 304541.02238179  53219.0448574
 284176.85264899  93400.71982017 137995.60254859 164465.60061817
 209777.47549039 203563.29570254 176679.65980431 250553.4609179
 142724.97393041 134288.63707225 171785.11537111 344054.87023123
 182442.95972596  99605.68463271 131552.70068821 102458.6004887
 123461.42545918 193291.78508491 156540.42729331 269762.81773183
 214029.99936865 151955.03692603 116513.08932336 244970.39361538
 134699.2149871  264939.1730608  302655.25814127 112765.72171924
 146343.51688839 141400.89021689 159152.01443607 231882.27246478
 233340.47229044 140170.23115364 355837.95593261 125123.6071953
 236563.7287866  118584.91492874 103674.34219114 158279.89652213
 146239.3763445  414236.75239941 337687.01869556 292095.33884883
 281975.96957962 275157.5265404  322473.52623347 307559.16658245
 202413.61321578 141543.52304339 156938.45535856 264430.93571626
 208855.42690191 148534.23761202 104695.9156791  149705.7343585
 101509.83486213 298235.68040125 342471.41568024 255288.59034683
 144073.37987936 195222.06016888 128842.95043884 264722.55264243
 122814.74476713 133278.4795676  307911.95867229 167263.61643095
 173717.5283684  566453.75411051 182844.43105375 143056.66623968
 195459.9517828  110031.5315688  181380.27781101 326642.05493544
 138687.73082    197539.06193031 116397.95911794  57864.29190174
 200870.72982498 274103.15023701 110331.98749734 195890.18985712
 234230.62113975 121111.45520781 137725.6391029  288996.92043677
 200176.68183782 364232.50615443 116188.37153176 125646.49499415
 225859.82479563 172176.23953285  93470.49188531 178488.39571047
 218214.18556022 225439.50186818 133730.67670966  67582.3120787
 243455.16368461 138315.47306896 113560.16506458  85012.3660346
 207779.94088431 171952.5210419  270402.27760786 254250.36145244
 103501.7808304  221091.33159311 184406.69839004 170462.34462181
 156799.2718875  225871.67203851  85108.65590281 152543.1330664
 180044.96550306 200375.75191088 181254.45434812 224810.46414272
  75144.10407593  81393.67998095 138318.56519304 231082.14246835
 152831.73330562 322773.27244389 196234.30285594  56775.45518337
 206549.14444971 109613.97109879 196192.45958022 100157.9526367
 222746.15990955 211519.5786086  199144.99012675 111277.72707786
  98150.90782231 236864.33886554 184958.66514372 189263.28121445
 265567.5169301  247763.83830809 108255.75070666 161646.28974509
 267710.34135243 116069.83636697 220248.79455542  99220.40041001
 227742.28703343 157984.25550973 107798.91813747 219704.63978513
  81507.24588044  76481.69523773  97652.61391762 303713.02140765
 146284.60208933 186530.14642609 241387.0772794   61365.50896352
 144289.97596438 108982.77136529 440530.78372543 120246.58324455
 115308.70156278 275911.79875926 201932.31328927 234498.94963676
 201583.06918458  98700.9196783  142687.06236746 211412.15970588
 167807.49540674 161636.28609433 150436.44626318 191662.32787741
  98527.88338021 167785.99502907  81294.88719425 169177.67605055
 175874.39752831 165547.40376838 132551.96525579 205527.52636725
```

```
 236395.29650392 125794.65643289 173406.69375672 259033.27747268
 227234.37218332]
```

In [43]:

```
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: 1458946958.0904446
```

In [44]:

```
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: [257729.69686228 113247.68204486  83142.29454527 204106.43998851
 209066.49021888  69054.81228022 235176.11930442 205711.84838767
 187608.59515698 234284.06084395 101449.25460271 304193.6360213
 102109.0711376  289198.43257911 204491.12189758 142122.85239194
 249633.03850946 147889.3873303  210566.82234788 277458.74148889
  91422.16023407 153936.54170005 160629.24135716 243131.57205681
 372542.62040315 219872.31056047 116863.87010022 100885.24049767
 318608.68254302 168098.13679091 258090.97848895 201288.31490053
 154603.02948577 142756.31460187 205137.22911272 382237.47687556
 124919.66167664 141572.72803651 273666.16449995 214125.2076817
 170460.20631671 124565.97224865 197629.92077736 376132.94977617
 148005.53413165 288031.84338491 108663.12260663 218568.08286906
  88451.65003835 295193.48660115 125792.5796616   54906.97302469
 133434.7793381  173716.93120033 214868.54616558 110315.71602181
 101776.23316753 198433.20185705 139852.85532953 313008.51176235
  71372.35173051 185193.16837526 153957.25028227 299596.82600383
  76797.56928684 324385.12193927 165298.44677344 112130.60821731
 200829.52731526 206557.77140637 130686.75945749 271522.00161621
 119151.76652787 144809.33683478  75279.72416438 136789.1008269
  35552.84939405 163985.23467214 273973.46020853 116015.84441149
 319949.02350894 219061.27031023 293870.99785338 211806.30759287
  91284.37825021 265712.38394797 142676.37351153 118565.93663895
 143674.04796073 279150.78978229 269350.79958902 301749.09017645
 184238.16223454 165127.83870291 149749.971759   213932.73020163
 249099.03142859 224103.21830794 274398.78074231 245917.53728663
 111630.87488119  99277.84590206 195168.37745894 208067.66777104
 154996.99469861 218133.62585452 212678.72001038 122425.46620394
 162745.4701208  221876.0238869  146954.89801987 109955.96291884
 319902.41000277 298399.98697946 337080.93758278  88041.26768065
 146956.67018354 227528.07763404 122783.32119227 205361.95445653
 217572.36672847 107445.89506261 133733.93273959 220237.13185881
 134562.85513962 212441.22565528 145664.6490619  291990.19815932
 183992.81011514  97710.74841533 325888.23318497 168160.98330115
 122850.65143328 134054.43008094 309600.11075927 134569.71283508
 281846.16771076 130925.21004783  92692.33683927 160945.68481687
 148205.17349014 158737.18245717 178212.15350491 257197.05522751
 114904.770572   138119.73032986 238108.69937308 318640.59412828
 104227.02007034 193572.95306041 172128.45114871 144831.00706529
  95568.83603287 250976.29637637 304506.20094194  53074.03126872
 284201.58870536  93341.64733158 137832.8569521  164328.13080684
 209750.74474024 203507.35157691 176605.11071809 250433.47941814
 142442.56296303 134372.47735183 171734.75893015 344242.61402369
 182535.72149748  99606.43760465 131556.77246267 102404.11838037
 123442.09968805 193351.73679664 156603.51624856 269864.8398754
 214055.85227124 151934.36274075 116604.28215523 244959.99451503
 134838.25785952 264898.68559226 302725.58777033 112731.7446967
 146413.91028759 141404.48991242 159092.12693511 231871.85339525
 233356.97471515 140184.4923673  356182.13216459 125098.80530538
 236563.67234591 118595.96441677 103720.0363061  158211.47689343
 146208.52042129 414448.62868843 337706.51862255 292089.54342658
 282067.39287353 275220.38527458 322598.62605758 307607.66942813
 202374.38645492 141564.96112    156913.87655793 264220.52332684
 208842.25175567 148495.28954641 104610.26939919 149704.15186406
 101505.94260441 298200.94930269 342555.63710936 255293.96192183
 143941.00968853 195070.94331289 128759.7558806  264731.27944764
 122738.81214254 133202.40385339 307879.92219547 167198.36515005
 173655.27272885 566791.4626781  182859.08759158 142978.96172471
 195411.89902191 109989.97237563 181424.62759165 326700.99083302
 138742.78214564 197520.5489765  116241.60249896  57889.11718413
 200889.63670003 274133.72241038 110284.19196382 195834.57014334
 234280.54148409 121025.73789444 137750.5533365  289102.2483371
 200107.11126573 364167.65133585 116093.7426804  125692.84631828
 225912.27728164 172268.27034232  93550.38096287 178436.21724116
 218228.05492206 225394.75569643 133705.24490148  67286.84252492
 243435.3352886  138212.91208172 113575.12067421  84763.0571555
 207893.46649157 172014.92872346 270368.22866089 254308.20269938
 103489.48936776 221152.56639884 184242.04200553 170421.97540171
 156761.53497604 225877.48743629  85073.90472036 152603.94331392
 180102.85191822 200371.14356128 181240.6625634  225061.98394381
  74826.71147585  81085.1939571  138486.98130548 231190.09881719
 152862.5234824  322870.83873105 196244.21793811  56726.64940312
 206524.79485726 109539.23259487 196178.63293578 100010.70807563
 222777.79438513 211539.35671092 199094.64072609 111278.51445755
  98154.13024319 236845.81680032 184872.18562098 189199.77805113
 265551.01867187 247875.47829531 108230.38633121 161861.58888709
 267828.77750227 116126.80380291 220329.46846636  99009.18854187
 227723.3048807  158054.99967295 107672.10313119 219802.67568716
  81445.94291229  76355.17286645  97708.71173269 303750.99022453
 146164.36964155 186675.79492752 241397.08830316  61402.63825225
 144276.21302137 108876.91545721 440705.04938866 120132.90211529
 115315.14265229 275893.22780201 201923.10589601 234448.28372038
 201360.63783727  98674.76411393 142858.79213935 211418.19972226
 167946.77034753 161744.898088   150329.68930056 191652.17462118
  98628.76165898 167691.68791082  81271.31765368 169263.90002432
 175828.08068586 165632.53323128 132599.66558174 205533.56435214
```

```
 236522.60127319 125734.98875162 173340.82150737 259083.66035875
 227358.42738081]

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 iterations. Fitting data with very small alpha may cause precision problems.
  ConvergenceWarning)
```

In [45]:
```
lasso_mse=metrics.mean_squared_error(y_test, lasso_y_pred)
print("LASSO_MSE:",lasso_mse)
```

```
LASSO_MSE: 1460906418.115903
```

Step : 9

In [48]:
```
import numpy as np
print("RMSE without CD: ",np.sqrt(mse))
print("RMSE with CD: ",np.sqrt(mse_cd))
print("RMSE with CD and SS: ",np.sqrt(ss_mse))
print("RMSE with CD and MnMaxScaling: ",np.sqrt(mm_mse))
print("RMSE of SGDRegressor with CD and StandardScaler: ",np.sqrt(sgd_mse))
print("RMSE of Ridgecv with CD and Standard Scaler: ",np.sqrt(ridge_mse))
print("RMSE of LassoCV with CD and StandardScaler",np.sqrt(lasso_mse))
```

```
RMSE without CD:  38403.48064430541
RMSE with CD:  38223.50808263985
RMSE with CD and SS:  38223.50808264128
RMSE with CD and MnMaxScaling:  38223.50808264125
RMSE of SGDRegressor with CD and StandardScaler:  38804.500301058026
RMSE of Ridgecv with CD and Standard Scaler:  38196.164180326334
RMSE of LassoCV with CD and StandardScaler 38221.805531867576
```

In [ ]:
```

```