225229151

House Prediction using LR with Regualarization ¶

STEP 1

In [1]:
```
import pandas as pd
```

In [79]:
```
df = pd.read_csv('Ames_House_Sales_Cropped.csv')
```

In [25]:
```
df.head()
```

Out[25]:

	BldgType	CentralAir	1stFlrSF	2ndFlrSF	3SsnPorch	BedroomAbvGr	BsmtFinSF1	BsmtFinSF2	BsmtFullBath	BsmtHalfBath	...	OverallQual	PoolArea	Scr
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 [4]:
```
df.shape
```

Out[4]:
```
(1379, 39)
```

In [5]:
```
df.columns
```

Out[5]:
```
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 [9]:
```
df.dtypes
```

Out[9]:
```
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 [10]:
```
df.info
```

Out[10]:
```
<bound method DataFrame.info of      BldgType CentralAir  1stFlrSF  2ndFlrSF  3SsnPorch  BedroomAbvGr  \
0        1Fam          Y     856.0     854.0        0.0             3
1        1Fam          Y    1262.0       0.0        0.0             3
2        1Fam          Y     920.0     866.0        0.0             3
3        1Fam          Y     961.0     756.0        0.0             3
4        1Fam          Y    1145.0    1053.0        0.0             4
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
```

In [27]:
```
sr = df.columns
```

In [28]:
```
sr.value_counts()
```

Out[28]:
```
OpenPorchSF      1
BsmtUnfSF        1
GrLivArea        1
TotalBsmtSF      1
OverallCond      1
GarageArea       1
YrSold           1
GarageYrBlt      1
EnclosedPorch    1
GarageCars       1
MoSold           1
KitchenAbvGr     1
YearBuilt        1
SalePrice        1
CentralAir       1
BsmtFinSF2       1
LowQualFinSF     1
BsmtHalfBath     1
BsmtFinSF1       1
BsmtFullBath     1
OverallQual      1
MasVnrArea       1
2ndFlrSF         1
BldgType         1
HalfBath         1
1stFlrSF         1
YearRemodAdd     1
Fireplaces       1
3SsnPorch        1
TotRmsAbvGrd     1
PoolArea         1
MiscVal          1
FullBath         1
WoodDeckSF       1
LotFrontage      1
LotArea          1
MSSubClass       1
ScreenPorch      1
BedroomAbvGr     1
dtype: int64
```

STEP 2

In [26]:
```
df.head()
```

Out[26]:

	BldgType	CentralAir	1stFlrSF	2ndFlrSF	3SsnPorch	BedroomAbvGr	BsmtFinSF1	BsmtFinSF2	BsmtFullBath	BsmtHalfBath	...	OverallQual	PoolArea	Scr
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 [29]:
```
df.pop('BldgType')
```

Out[29]:
```
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 [31]:
```
df.head
```

```
23           Y    1060.0       0.0        0.0            3       840.0
24           Y    1060.0       0.0        0.0            3       188.0
25           Y    1600.0       0.0        0.0            3         0.0
26           Y     900.0       0.0        0.0            3       234.0
27           Y    1704.0       0.0        0.0            3      1218.0
28           Y    1600.0       0.0        0.0            2      1277.0
29           N     520.0       0.0        0.0            1         0.0
...        ...       ...       ...        ...          ...         ...
1349         Y    1048.0     510.0        0.0            3       580.0
1350         Y     804.0       0.0        0.0            2       510.0
1351         Y    1440.0       0.0        0.0            3       678.0
1352         Y     734.0    1104.0        0.0            4         0.0
1353         Y     958.0       0.0        0.0            2       958.0
1354         Y     968.0       0.0        0.0            4         0.0
1355         Y     962.0     830.0        0.0            3         0.0
1356         Y    1126.0       0.0        0.0            3       936.0
1357         Y    1537.0       0.0        0.0            3         0.0
1358         Y     864.0       0.0        0.0            3       616.0
1359         Y    1932.0       0.0      304.0            2      1336.0
1360         Y    1236.0       0.0        0.0            2       600.0
```

In [33]:
```
df.pop('CentralAir')
```

Out[33]:
```
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 [39]:
```
df.columns
```

Out[39]:
```
Index(['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 [35]:
```
df.shape
```

Out[35]:
```
(1379, 37)
```

In [51]:
```
df_ames = df.copy()
```

In [54]:
```
y = df_ames[['SalePrice']]
```

In [55]:
```
x = df_ames.drop(columns=['SalePrice'])
```

In [56]:
```
from sklearn.model_selection import train_test_split
```

In [57]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state = 42)
```

In [58]:
```
from sklearn.linear_model import LinearRegression
```

In [59]:
```
lr = LinearRegression()
```

In [60]:
```
lr.fit(x_train,y_train)
```

Out[60]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [61]:
```
y_pred = lr.predict(x_test)
```

In [62]:
```
from sklearn.metrics import mean_squared_error
```

In [63]:
```
mse = mean_squared_error(y_test,y_pred)
```

In [64]:
```
mse
```

Out[64]:
```
1474827325.5975406
```

STEP 3

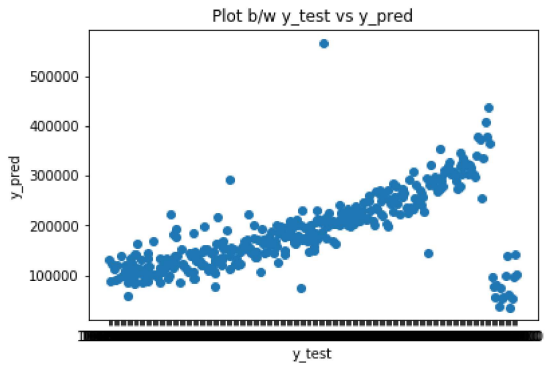
SCATTER PLOT

In [66]:
```
import matplotlib as plt
```

In [67]:
```
import seaborn as sns
```

In [76]:
```
%matplotlib inline

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



STEP 4

In [80]:
```
enc =pd.get_dummies(df)
```

In [81]:
```
enc
```

Out[81]:

	1stFlrSF	2ndFlrSF	3SsnPorch	BedroomAbvGr	BsmtFinSF1	BsmtFinSF2	BsmtFullBath	BsmtHalfBath	BsmtUnfSF	EnclosedPorch	...	YearRemodAdd
0	856.0	854.0	0.0	3	706.0	0.0	1	0	150.0	0.0	...	2003
1	1262.0	0.0	0.0	3	978.0	0.0	0	1	284.0	0.0	...	1976
2	920.0	866.0	0.0	3	486.0	0.0	1	0	434.0	0.0	...	2002
3	961.0	756.0	0.0	3	216.0	0.0	1	0	540.0	272.0	...	1970
4	1145.0	1053.0	0.0	4	655.0	0.0	1	0	490.0	0.0	...	2000
5	796.0	566.0	320.0	1	732.0	0.0	1	0	64.0	0.0	...	1995
6	1694.0	0.0	0.0	3	1369.0	0.0	1	0	317.0	0.0	...	2005
7	1107.0	983.0	0.0	3	859.0	32.0	1	0	216.0	228.0	...	1973
8	1022.0	752.0	0.0	2	0.0	0.0	0	0	952.0	205.0	...	1950
9	1077.0	0.0	0.0	2	851.0	0.0	1	0	140.0	0.0	...	1950
10	1040.0	0.0	0.0	3	906.0	0.0	1	0	134.0	0.0	...	1965
11	1182.0	1142.0	0.0	4	998.0	0.0	1	0	177.0	0.0	...	2006
12	912.0	0.0	0.0	2	737.0	0.0	1	0	175.0	0.0	...	1962
13	1494.0	0.0	0.0	3	0.0	0.0	0	0	1494.0	0.0	...	2007
14	1253.0	0.0	0.0	2	733.0	0.0	1	0	520.0	176.0	...	1960
15	854.0	0.0	0.0	2	0.0	0.0	0	0	832.0	0.0	...	2001
16	1004.0	0.0	0.0	2	578.0	0.0	1	0	426.0	0.0	...	1970
17	1296.0	0.0	0.0	2	0.0	0.0	0	0	0.0	0.0	...	1967
18	1114.0	0.0	0.0	3	646.0	0.0	1	0	468.0	0.0	...	2004
19	1339.0	0.0	0.0	3	504.0	0.0	0	0	525.0	0.0	...	1965
20	1158.0	1218.0	0.0	4	0.0	0.0	0	0	1158.0	0.0	...	2006
21	1108.0	0.0	0.0	3	0.0	0.0	0	0	637.0	205.0	...	1950
22	1795.0	0.0	0.0	3	0.0	0.0	0	0	1777.0	0.0	...	2002
23	1060.0	0.0	0.0	3	840.0	0.0	1	0	200.0	0.0	...	1976
24	1060.0	0.0	0.0	3	188.0	668.0	1	0	204.0	0.0	...	2001
25	1600.0	0.0	0.0	3	0.0	0.0	0	0	1566.0	0.0	...	2007
26	900.0	0.0	0.0	3	234.0	486.0	0	1	180.0	0.0	...	2000
27	1704.0	0.0	0.0	3	1218.0	0.0	1	0	486.0	0.0	...	2008
28	1600.0	0.0	0.0	2	1277.0	0.0	1	0	207.0	0.0	...	1997
29	520.0	0.0	0.0	1	0.0	0.0	0	0	520.0	87.0	...	1950
...	...	...	...	...	...	...	...	...	...	...	...	...
1349	1048.0	510.0	0.0	3	580.0	0.0	1	0	333.0	0.0	...	1950
1350	804.0	0.0	0.0	2	510.0	0.0	1	0	278.0	154.0	...	1992
1351	1440.0	0.0	0.0	3	678.0	0.0	0	0	762.0	99.0	...	1981
1352	734.0	1104.0	0.0	4	0.0	0.0	0	0	732.0	0.0	...	2005
1353	958.0	0.0	0.0	2	958.0	0.0	0	0	0.0	0.0	...	1976
1354	968.0	0.0	0.0	4	0.0	0.0	0	0	656.0	0.0	...	2007
1355	962.0	830.0	0.0	3	0.0	0.0	1	0	936.0	0.0	...	2000
1356	1126.0	0.0	0.0	3	936.0	0.0	1	0	190.0	0.0	...	1977
1357	1537.0	0.0	0.0	3	0.0	0.0	1	0	1319.0	0.0	...	2005
1358	864.0	0.0	0.0	3	616.0	0.0	0	0	248.0	0.0	...	1971
1359	1932.0	0.0	304.0	2	1336.0	0.0	1	0	596.0	0.0	...	2008
1360	1236.0	0.0	0.0	2	600.0	0.0	1	0	312.0	158.0	...	1996
1361	1040.0	685.0	0.0	3	315.0	110.0	0	0	114.0	216.0	...	1979
1362	1423.0	748.0	0.0	3	0.0	0.0	0	0	588.0	0.0	...	1994
1363	848.0	0.0	0.0	1	697.0	0.0	1	0	151.0	0.0	...	2004
1364	1026.0	981.0	0.0	3	765.0	0.0	1	0	252.0	0.0	...	2008
1365	952.0	0.0	0.0	2	0.0	0.0	0	0	952.0	0.0	...	1950
1366	1422.0	0.0	0.0	3	0.0	0.0	0	0	1422.0	0.0	...	2004
1367	913.0	0.0	0.0	3	187.0	627.0	1	0	0.0	252.0	...	1966
1368	1188.0	0.0	0.0	3	593.0	0.0	0	0	595.0	0.0	...	1962
1369	1220.0	870.0	0.0	3	1079.0	0.0	1	0	141.0	0.0	...	1996
1370	796.0	550.0	0.0	2	0.0	0.0	0	0	560.0	0.0	...	2000
1371	1578.0	0.0	0.0	3	0.0	0.0	0	0	1573.0	0.0	...	2009
1372	1072.0	0.0	0.0	2	547.0	0.0	1	0	0.0	0.0	...	2005

	1stFlrSF	2ndFlrSF	3SsnPorch	BedroomAbvGr	BsmtFinSF1	BsmtFinSF2	BsmtFullBath	BsmtHalfBath	BsmtUnfSF	EnclosedPorch	...	YearRemodAdd
1373	1221.0	0.0	0.0	2	410.0	0.0	1	0	811.0	0.0	...	2005
1374	953.0	694.0	0.0	3	0.0	0.0	0	0	953.0	0.0	...	2000
1375	2073.0	0.0	0.0	3	790.0	163.0	1	0	589.0	0.0	...	1988
1376	1188.0	1152.0	0.0	4	275.0	0.0	0	0	877.0	0.0	...	2006
1377	1078.0	0.0	0.0	2	49.0	1029.0	1	0	0.0	112.0	...	1996
1378	1256.0	0.0	0.0	3	830.0	290.0	1	0	136.0	0.0	...	1965

1379 rows × 44 columns

STEP 5

In [83]:
```
X=enc.drop(["SalePrice"],axis=1)
```

In [85]:
```
Y=enc["SalePrice"].values
```

In [86]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=.25,random_state=42)
```

In [87]:
```
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)
```

```
[208500. 181500. 223500. ... 266500. 142125. 147500.]
```

In [88]:
```
mse_cd=metrics.mean_squared_error(Y_test,Y_pred)
print("MSE with Categorical data: ",mse_cd)
```

```
MSE with Categorical data:  1461036570.1436336
```

STEP 6

In [90]:
```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
ss_X_train=std.fit_transform(X_train)
ss_X_train
```

Out[90]:
```
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 [91]:
```
ss_X_test=std.transform(X_test)
ss_X_test
```

Out[91]:
```
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 [92]:
```
from sklearn.linear_model import LinearRegression
lr1=LinearRegression()
lr1.fit(ss_X_train,y_train)
ss_y_pred=lr1.predict(ss_X_test)
ss_y_pred
```

Out[92]:
```
array([[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],
```

In [94]:
```
ss_mse=metrics.mean_squared_error(Y_test,ss_y_pred)
print(ss_mse)
```

```
1461036570.1437433
```

STEP 7

In [95]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [96]:
```
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)
```

Out[96]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [97]:
```
mm_y_pred=mm_lr.predict(mm_X_test)
print(mm_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 128768.16960183 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 [98]:
```
mm_mse=metrics.mean_squared_error(Y_test,mm_y_pred)
print(mm_mse)
```

```
1461036570.1437414
```

STEP 8

In [99]:
```
from sklearn.linear_model import SGDRegressor
```

In [100]:
```
sgd=SGDRegressor()
sgd.fit(ss_X_train, y_train)
sgd_y_pred=sgd.predict(ss_X_test)
sgd_y_pred
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:128: FutureWarning: max_iter and tol parameters have been added in <class 'sklearn.linear_model.stochastic_gradient.SGDRegressor'> 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)
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-packages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[100]:
```
array([241623.3785292 , 128206.18294903,  98337.71029037, 198442.08194723,
       205766.46190098,  79930.29777682, 221435.07462463, 193514.74839538,
       208709.50189482, 219953.75899932, 119722.76496267, 284347.38359574,
       129687.08705534, 264599.32948801, 203826.20128119, 148867.99307159,
       251315.73247814, 154112.42725806, 207985.23361726, 263879.74221871,
       108209.78819073, 181545.59737979, 169329.32118491, 233203.8341299 ,
       339331.63779874, 201242.26183716, 117271.09896082, 118295.32707646,
       289485.71739652, 174295.39922112, 233547.7686967 , 195933.8897223 ,
       157260.76173433, 173811.31294647, 208153.68037469, 343674.15386045,
       146503.90006786, 131718.33891559, 248967.21945758, 220964.41751134,
       191208.43119069, 135226.10452256, 185735.29818564, 320330.23548457,
       150406.47567045, 283679.42724976, 126567.01945179, 202474.39838189,
       113562.0656939 , 265512.14896281, 132198.92631958,  76123.95584526,
       143313.99887151, 165056.33640899, 225393.10568294, 135698.75040338,
       102699.72506857, 191231.7419109 , 143239.01659923, 277739.16854763,
        92646.53997459, 179172.48684017, 165330.77142724, 284587.6614351 ,
       103021.98821792, 298887.14358237, 142040.4000021 , 147074.55625313,
       192134.39193979, 211959.7142355 , 156404.79231645, 271206.53247482,
       147174.75574113, 149895.84742399,  84902.82170495, 118740.10663494,
        62338.0519251 , 183942.27548958, 257121.71301293, 134145.52795376,
       287335.7104357 , 221334.01017365, 266477.00188116, 205233.45581888,
       118773.65246669, 261705.12400425, 151582.48276923, 136206.5094035 ,
       154635.83790939, 266267.63744061, 247054.91072161, 293644.70379285,
       199031.49774156, 158916.25842704, 172265.78587676, 219930.8529564 ,
       250822.93925024, 236582.1594967 , 259237.99923256, 245181.80789506,
       118456.91221285, 103051.80356841, 192576.77435963, 207496.93762685,
       157083.2627773 , 213675.11150777, 207436.52571591, 141621.46647853,
       177851.78290372, 191782.86943344, 170622.4798801 , 126593.09341991,
       289507.7168145 , 286839.20295214, 307684.36301131, 112939.44663297,
       146536.06546928, 219703.05256861, 148942.09569135, 213978.52693001,
       226164.53577579, 124873.17837719, 151506.61517816, 215034.27241384,
       141156.12875101, 225383.95874411, 146560.42409401, 278442.72470567,
       196600.03253296, 112286.28522789, 314803.54187799, 161856.91535774,
       122569.56983878, 147078.8726571 , 272290.92911646, 166524.37442205,
       279556.1118347 , 132567.32146958, 114164.41457446, 163042.85587426,
       163459.5094635 , 169489.57888677, 174072.25205117, 251913.77615275,
       112963.00349711, 142188.34389091, 231208.272225  , 290929.71782956,
       127402.14117775, 209466.143177  , 155277.28916705, 144536.52046235,
       115122.79109107, 245675.22824292, 273610.36054915,  84978.14163479,
       267686.14133602, 119926.3038729 , 155677.48661582, 156697.0358221 ,
       206692.48193928, 181246.50038101, 193244.25602069, 228449.27284571,
       167922.30770836, 149324.78200241, 183411.64996834, 296775.74526467,
       179821.33311884, 123113.15553274, 143465.62331924, 132143.36186673,
       157232.96033411, 210531.25266296, 166435.51740634, 250300.60012972,
       222325.93367978, 150376.1500113 , 120680.07467865, 233807.54944601,
       145265.97027733, 264698.79123023, 279213.50429576, 108250.08514163,
       149134.73245484, 157325.40435992, 176676.70304949, 205928.05312001,
       222307.66129799, 147190.36018627, 285533.24400468, 115693.32500134,
       229407.32769139, 137468.09964868, 120204.86336368, 155389.64798041,
       131915.13809578, 395435.12250594, 296060.03550321, 290684.09545012,
       266466.15572831, 256755.966423  , 308959.21741226, 282833.99727353,
       200106.80081311, 145322.36098196, 158477.60386146, 126114.86850825,
       214779.15563961, 153401.3255438 , 119655.60347934, 172702.25170056,
       121019.70013924, 282677.26800276, 308235.90170514, 249973.70625178,
       163567.91889634, 193820.68410274, 133974.18774761, 261880.21600644,
       140555.01873235, 154832.21170511, 284016.2166022 , 168650.4311526 ,
       187238.59809426, 452558.84191389, 175639.45518683, 130858.95252074,
       191065.74163085, 128598.02326873, 175656.20136768, 284064.74970184,
       143642.68535779, 209128.13964996, 140267.22158868,  90288.42564622,
       204579.25056894, 271931.03788077, 136203.27146597, 176710.95215327,
       223861.76927737, 138872.91145537, 138978.75584111, 272204.7274921 ,
       209606.16504159, 335975.53737196, 130251.95660889, 133844.62115372,
       231231.15009761, 176327.57937838, 111139.69982794, 189241.96068883,
       226797.59858561, 223165.7263952 , 154292.46278865,  86233.74206032,
       243545.65542391, 146279.47445156, 128828.87904028, 101999.6949719 ,
       195519.46174966, 179406.55551017, 259618.64715094, 236055.89075311,
       125254.12155898, 216499.4960035 , 193794.67118477, 190658.69833623,
       169098.32984145, 213338.40061356, 102702.0100813 , 148899.05335938,
       169509.50502889, 203667.40156133, 191999.77353882, 201608.71002797,
       -33221.05115566,  92390.0199757 , 135697.57017318, 208977.27423672,
       153106.03446058, 308171.73329603, 205596.56218048,  77945.97017458,
       213181.25263342, 124141.58394658, 195908.74175289, 130293.36668367,
       226502.14109251, 213454.43525927, 184371.34263465, 113697.96469827,
       108232.68690806, 234367.0040718 , 186680.32574766, 192627.24063162,
       270529.33312461, 253007.47277363, 130593.69631547, 147380.5061284 ,
       239580.07774364, 137972.0049455 , 215465.08512346, 115287.6685526 ,
       226553.40214611, 180847.05213205, 135810.17356777, 192351.22214099,
       108231.02222366, 111820.81086032, 123845.2999832 , 265242.98095392,
        -8934.75961547, 195763.61609717, 226090.50506472,  86200.34182757,
       153893.25441177, 121733.39014   , 384396.25391733, 143782.64623351,
       117160.04618649, 253184.78109716, 212765.64821994, 226660.68799317,
       190186.6038845 , 120898.23295898, 141257.12872175, 222666.86252393,
       170362.13954993, 185676.03393587, 165424.41283164, 192074.45977079,
       121356.82927906, 152035.11343808, 114797.14795083, 159760.80726385,
       169337.41644698, 179881.60282349, 140882.4143879 , 217684.86744285,
```

```
       233893.59192322, 156054.57914274, 185002.9125136 , 247134.82186214,
       230613.35712056])
```

In [101]:
```
sgd_mse=metrics.mean_squared_error(y_test, sgd_y_pred)
sgd_mse
```

Out[101]:
```
1802733633.0389442
```

In [102]:
```
from sklearn.linear_model import Ridge
```

In [103]:
```
ridge=Ridge()
ridge.fit(ss_X_train, y_train)
ridge_y_pred=ridge.predict(ss_X_test)
ridge_y_pred
```

Out[103]:
```
array([[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],
```

In [104]:
```
ridge_mse=metrics.mean_squared_error(y_test, ridge_y_pred)
ridge_r2=metrics.r2_score(y_test, sgd_y_pred)
ridge_mse
```

Out[104]:
```
1458946958.0904446
```

In [105]:
```
from sklearn.linear_model import Lasso
```

In [106]:
```
lasso=Lasso()
lasso.fit(ss_X_train, Y_train)
lasso_y_pred=lasso.predict(ss_X_test)
```

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

In [107]:
```
lasso_y_pred
```

Out[107]:
```
array([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])
```

In [109]:
```
lasso_mse=metrics.mean_squared_error(Y_test, lasso_y_pred)
lasso_mse
```

Out[109]:
```
1460906418.115903
```

STEP 9

In [113]:
```
import numpy as np
print("RMSE without one hot encoding: ",np.sqrt(mse))
print("RMSE with one hot encoding: ",np.sqrt(mse_cd))
print("RMSE with OHE and Standard Scaling: ",np.sqrt(ss_mse))
print("RMSE with CD and MinMax Scaling: ",np.sqrt(mm_mse))
print("RMSE of SGDRegressor with OHE and Standard Scaler: ",np.sqrt(sgd_mse))
print("RMSE of Ridgecv with OHE and Standard Scaler: ",np.sqrt(ridge_mse))
print("RMSE of LassoCV with OHE and Standard Scaler",np.sqrt(lasso_mse))
```

```
RMSE without one hot encoding:  38403.48064430541
RMSE with one hot encoding:  38223.50808263985
RMSE with OHE and Standard Scaling:  38223.50808264128
RMSE with CD and MinMax Scaling:  38223.50808264125
RMSE of SGDRegressor with OHE and Standard Scaler:  42458.61082323519
RMSE of Ridgecv with OHE and Standard Scaler:  38196.164180326334
RMSE of LassoCV with OHE and Standard Scaler 38221.805531867576
```

In [ ]: