225229141

SURUTHI S

PML LAB 4 - HOUSE PRICE PREDICTION USING LR WITH REGULARIZATION

STEP 1 : IMPORT DATASET

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [32]:
```
ames = pd.read_csv("/content/Ames_House_Sales_Cropped.csv")
```

In [25]:
```
ames.shape #81 columns and 2051 records
```

Out[25]: (1379, 39)

In [33]:
```
ames.info() # There are a lot of columns
```

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

In [ ]:
```
ames.describe().T #Summary statistics
```

In [ ]:
```
ames.isnull().sum()
```

STEP 2 : PREDICT SALE PRICE WITHOUT CATEGORICAL FEATURES

In [36]:
```
exc = ames[['BldgType','CentralAir']]
```

In [37]:
```
exc
```

Out[37]:

	BldgType	CentralAir
0	1Fam	Y
1	1Fam	Y
2	1Fam	Y
3	1Fam	Y
4	1Fam	Y
...	...	...
1374	1Fam	Y
1375	1Fam	Y
1376	1Fam	Y
1377	1Fam	Y
1378	1Fam	Y

1379 rows × 2 columns

In [38]:
```
ames = ames.select_dtypes(exclude = ['object'])
```

In [39]:
```
ames.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', 'YearBuil
t',
       'YearRemodAdd', 'YrSold', 'SalePrice'],
      dtype='object')
```

In [40]:
```
len(ames.columns)
```

Out[40]:
```
37
```

TRAIN TEST SPLIT

In [41]:
```
ames_copy = ames.copy()
```

In [42]:
```
ames_copy = ames_copy.dropna()
```

In [ ]:
```
ames_copy.isnull().sum()
```

In [44]:
```
ames_copy.shape
```

Out[44]: (1379, 37)

In [45]:
```
labels = ames_copy[['SalePrice']]
```

In [46]:
```
features = ames_copy.drop(columns=['SalePrice'])
```

In [47]:
```
from sklearn.model_selection import train_test_split
```

In [48]:
```
x_train,x_test,y_train,y_test = train_test_split(features,labels,test_size=0.25,random_state = 42)
```

In [49]:
```
from sklearn.linear_model import LinearRegression
```

In [50]:
```
reg = LinearRegression()
```

In [51]:
```
reg.fit(x_train,y_train)
```

Out[51]: LinearRegression()

In [52]:
```
y_pred = reg.predict(x_test)
```

In [53]:
```
from sklearn.metrics import mean_squared_error
```

In [54]:
```
mse = mean_squared_error(y_test,y_pred)
```

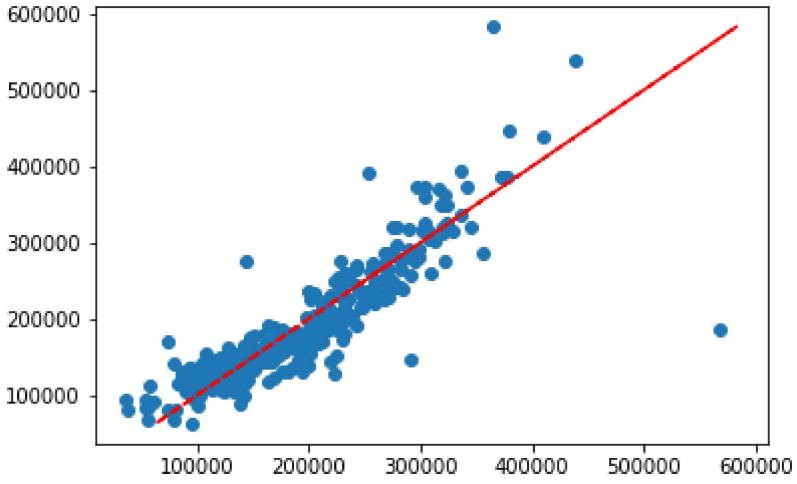
In [55]:
```
mse
```

Out[55]: 1474827325.5975182

STEP 3 : Scatter plot

In [56]:
```
plt.scatter(y_pred,y_test)
plt.plot(y_test,y_test,'r--')
```

Out[56]: [<matplotlib.lines.Line2D at 0x7f2a7c47b3d0>]



STEP 4 : ENCODE CATEGORICAL COLUMNS

In [ ]:
```
exc.dropna()
```

In [58]:
```
ames_copy.shape
```

Out[58]: (1379, 37)

In [ ]:
```
# ames_copy = ames_copy.join(except_data.set_index(ames_copy.index))
```

In [59]:
```
ames_cat=pd.merge(ames_copy, exc, left_index=True, right_index=True)
```

In [60]:
```
ames_cat.shape
```

Out[60]: (1379, 39)

In [ ]:
```
ames_cat.isnull().sum()
```

In [62]:
```
type(ames_cat)
```

Out[62]: pandas.core.frame.DataFrame

In [64]:
```
ames_cat[['BldgType']]
```

Out[64]:

	BldgType
0	1Fam
1	1Fam
2	1Fam
3	1Fam
4	1Fam
...	...
1374	1Fam
1375	1Fam
1376	1Fam
1377	1Fam
1378	1Fam

1379 rows × 1 columns

get dummies

In [65]:
```
ames_oh = pd.get_dummies(ames_cat)
```

In [66]:
```
ames_oh
```

Out[66]:

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

1379 rows × 44 columns

In [67]:
```
ames_oh.shape
```

Out[67]:
```
(1379, 44)
```

In [68]:
```
ames_oh.columns
```

Out[68]:
```
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', 'YearBuil
t',
       'YearRemodAdd', 'YrSold', 'SalePrice', 'BldgType_1Fam',
       'BldgType_2fmCon', 'BldgType_Duplex', 'BldgType_Twnhs',
       'BldgType_TwnhsE', 'CentralAir_N', 'CentralAir_Y'],
      dtype='object')
```

STEP 5 : Predict Sale Price WITH CATEGORICAL FEATURES

In [74]:
```
x_features_cat = ames_oh.drop('SalePrice',axis = 1)
```

In [75]:
```
x_features_cat.columns
```

Out[75]:
```
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', 'YearBuil
t',
       'YearRemodAdd', 'YrSold', 'BldgType_1Fam', 'BldgType_2fmCon',
       'BldgType_Duplex', 'BldgType_Twnhs', 'BldgType_TwnhsE', 'CentralAir_
N',
       'CentralAir_Y'],
      dtype='object')
```

In [76]:
```
x_label_cat = ames_oh[['SalePrice']]
```

In [ ]:
```
x_label_cat
```

In [73]:
```
lin_reg_cat = LinearRegression()
```

In [78]:
```
x_train_cat,x_test_cat,y_train_cat,y_test_cat = train_test_split(x_features_ca
t,x_label_cat,test_size = 0.25,random_state=42)
```

In [79]:
```
lin_reg_cat.fit(x_train_cat,y_train_cat)
#fitting the model
```

Out[79]:
```
LinearRegression()
```

In [80]:
```
y_pred_cat = lin_reg_cat.predict(x_test_cat)
```

In [83]:
```
mse_cat = mean_squared_error(y_pred_cat,y_test_cat)
```

In [84]:
```
mse_cat
```

Out[84]:
```
1461036570.1435425
```

STEP 6 : Nomalizing using Standard Scalar and Predict sales Price

In [93]:
```
from sklearn.preprocessing import StandardScaler
```

In [110]:
```
scale = StandardScaler()
```

In [111]:
```
#xtrain
scaled_x_train_norm = scale.fit_transform(x_train_cat)
```

In [112]:
```
# xtest
scaled_x_test_norm = scale.transform(x_test_cat)
```

In [114]:
```
lin_reg_norm = LinearRegression()
```

In [121]:
```
lin_reg_norm.fit(scaled_x_train_norm,y_train_cat)
```

Out[121]:
```
LinearRegression()
```

In [122]:
```
y_pred_norm = lin_reg_norm.predict(scaled_x_test_norm)
```

In [123]:
```
mse_norm = mean_squared_error(y_pred_norm,y_test_cat)
mse_norm
```

Out[123]:
```
1461036570.1437426
```

STEP 7 : Normalize using MinMax Scaler

In [124]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [125]:
```
minmax = MinMaxScaler()
```

In [126]:
```
# xtrain
scaled_x_train_minmax = minmax.fit_transform(x_train_cat)
```

In [127]:
```
# xtest
scaled_x_test_minmax = minmax.transform(x_test_cat)
```

In [128]:
```
lin_reg_std = LinearRegression()
```

In [129]:
```
lin_reg_std.fit(scaled_x_train_minmax,y_train_cat)
```

Out[129]:
```
LinearRegression()
```

In [130]:
```
y_pred_std = lin_reg_std.predict(scaled_x_test_minmax)
```

In [131]:
```
mse_std = mean_squared_error(y_pred_std,y_test_cat)
mse_std
```

Out[131]:
```
1461036570.143742
```

Step 8 SGD REGRESSION

In [132]:
```
from sklearn.linear_model import SGDRegressor
```

In [134]:
```
# using scaled_x_train_norm and scaled_x_test_norm
```

In [133]:
```
sgd = SGDRegressor()
```

In [135]:
```
sgd.fit(scaled_x_train_norm,y_train_cat)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/validation.py:993: DataC
onversionWarning: 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[135]:
```
SGDRegressor()
```

In [136]:
```
y_pred_sgd = sgd.predict(scaled_x_test_norm)
```

In [138]:
```
mse_sgd = mean_squared_error(y_pred_sgd,y_test_cat)
mse_sgd
```

Out[138]:
```
1427014934.7019606
```

L2 REGULARIZATION

In [139]:
```
from sklearn.linear_model import Ridge
```

In [140]:
```
ridge = Ridge()
```

In [141]:
```
# using scaled_x_train_norm and scaled_x_test_norm
```

In [142]:
```
ridge.fit(scaled_x_train_norm,y_train)
```

Out[142]:
```
Ridge()
```

In [143]:
```
y_pred_ridge = ridge.predict(scaled_x_test_norm)
```

In [144]:
```
mse_ridge = mean_squared_error(y_pred_ridge,y_test_cat)
mse_ridge
```

Out[144]:
```
1458946958.0904448
```

L1 Regularization

In [145]:
```
from sklearn.linear_model import Lasso
```

In [146]:
```
lasso=Lasso()
```

In [147]:
```
lasso.fit(scaled_x_train_norm,y_train)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_coordinate_desce
nt.py:647: ConvergenceWarning: Objective did not converge. You might want to
increase the number of iterations, check the scale of the features or conside
r increasing regularisation. Duality gap: 2.554e+11, tolerance: 6.588e+08
  model = cd_fast.enet_coordinate_descent(
```

Out[147]:
```
Lasso()
```

In [148]:
```
y_pred_lasso = lasso.predict(scaled_x_test_norm)
```

In [149]:
```
mse_lasso = mean_squared_error(y_pred_lasso,y_test_cat)
mse_lasso
```

Out[149]:
```
1460906418.115904
```

Step 9 RMSE

In [154]:
```
print("RMSE WITHOUT ONE HOT ENCODING : ",round(mean_squared_error(y_test,y_pre
d,squared=False)))
print("RMSE WITH ONE HOT ENCODING : ",round(mean_squared_error(y_pred_cat,y_te
st_cat,squared=False)))
print("RMSE WITH OHE and StandardScaler : ",round(mean_squared_error(y_pred_no
rm,y_test_cat,squared=False)))
print("RMSE WITH OHE and MinMaxScaler: ",round(mean_squared_error(y_pred_std,y
_test_cat,squared=False)))
print("RMSE OF SGDRegressor WITH OHE and StandardScaler : ",round(mean_squared
_error(y_pred_sgd,y_test_cat,squared=False)))
print("RMSE OF RidgeCV  WITH OHE and StandardScaler: ",round(mean_squared_erro
r(y_pred_ridge,y_test_cat,squared=False)))
print("RMSE OF LassoCV  WITH OHE and StandardScaler: ",round(mean_squared_erro
r(y_pred_lasso,y_test_cat,squared=False)))
```

```
RMSE WITHOUT ONE HOT ENCODING :  38403
RMSE WITH ONE HOT ENCODING :  38224
RMSE WITH OHE and StandardScaler :  38224
RMSE WITH OHE and MinMaxScaler:  38224
RMSE OF SGDRegressor WITH OHE and StandardScaler :  37776
RMSE OF RidgeCV  WITH OHE and StandardScaler:  38196
RMSE OF LassoCV  WITH OHE and StandardScaler:  38222
```