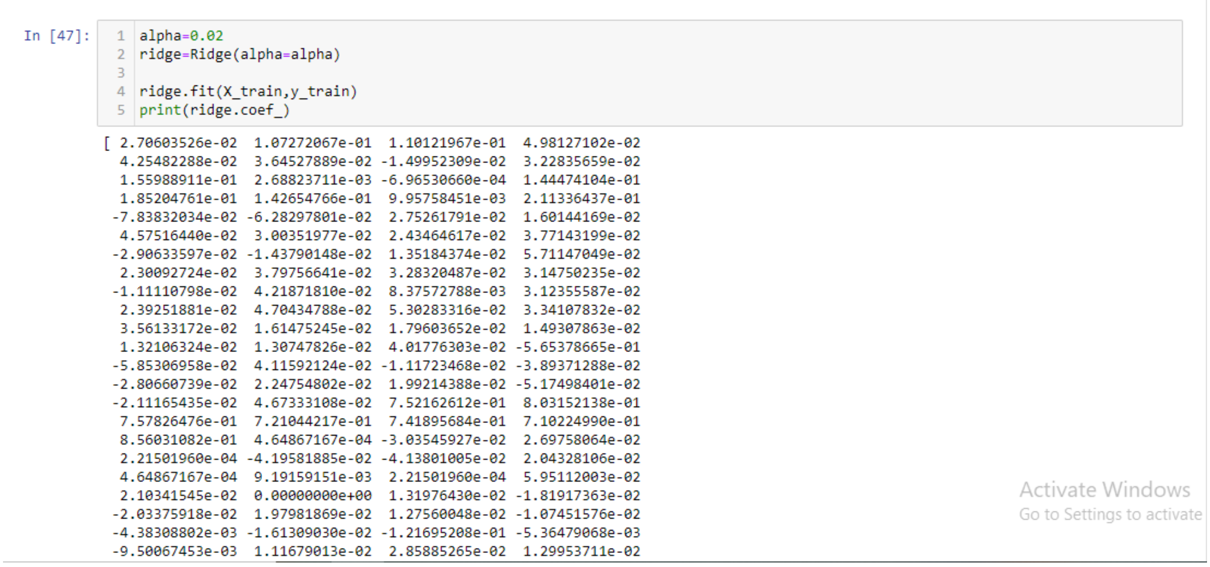
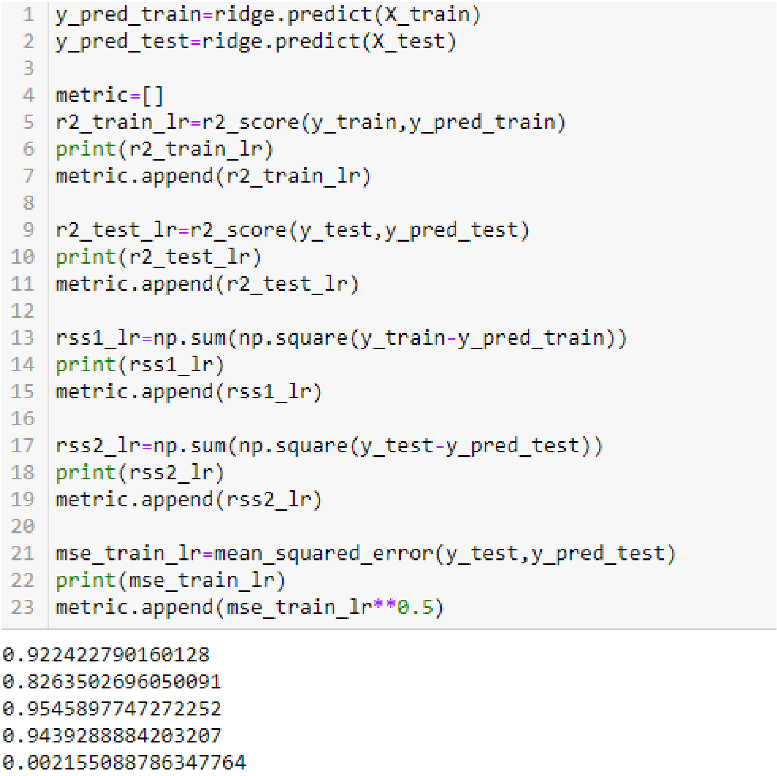
Q1 ANS:



In [47]:

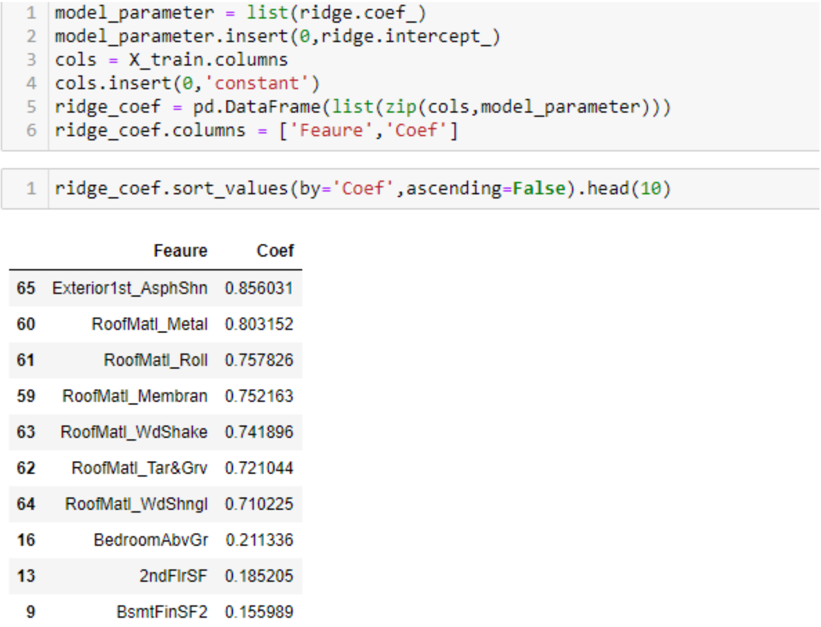
```
alpha=0.02
ridge=Ridge(alpha=alpha)

ridge.fit(X_train,y_train)
print(ridge.coef_)
```

```
[ 2.70603526e-02  1.07272067e-01  1.10121967e-01  4.98127102e-02
  4.25482288e-02  3.64527889e-02 -1.49952309e-02  3.22835659e-02
  1.55988911e-01  2.68823711e-03 -6.96530660e-04  1.44474104e-01
  1.85204761e-01  1.42654766e-01  9.95758451e-03  2.11336437e-01
 -7.83832034e-02 -6.28297801e-02  2.75261791e-02  1.60144169e-02
  4.57516440e-02  3.00351977e-02  2.43464617e-02  3.77143199e-02
 -2.90633597e-02 -1.43790148e-02  1.35184374e-02  5.71147049e-02
  2.30092724e-02  3.79756641e-02  3.28320487e-02  3.14750235e-02
 -1.11110798e-02  4.21871810e-02  8.37572788e-03  3.12355587e-02
  2.39251881e-02  4.70434788e-02  5.30283316e-02  3.34107832e-02
  3.56133172e-02  1.61475245e-02  1.79603652e-02  1.49307863e-02
  1.32106324e-02  1.30747826e-02  4.01776303e-02 -5.65378665e-01
 -5.85306958e-02  4.11592124e-02 -1.11723468e-02 -3.89371288e-02
 -2.80660739e-02  2.24754802e-02  1.99214388e-02 -5.17498401e-02
 -2.11165435e-02  4.67333108e-02  7.52162612e-01  8.03152138e-01
  7.57826476e-01  7.21044217e-01  7.41895684e-01  7.10224990e-01
  8.56031082e-01  4.64867167e-04 -3.03545927e-02  2.69758064e-02
  2.21501960e-04 -4.19581885e-02 -4.13801005e-02  2.04328106e-02
  4.64867167e-04  9.19159151e-03  2.21501960e-04  5.95112003e-02
  2.10341545e-02  0.00000000e+00  1.31976430e-02 -1.81917363e-02
 -2.03375918e-02  1.97981869e-02  1.27560048e-02 -1.07451576e-02
 -4.38308802e-03 -1.61309030e-02 -1.21695208e-01 -5.36479068e-03
 -9.50067453e-03  1.11679013e-02  2.85885265e-02  1.29953711e-02
```



```
y_pred_train=ridge.predict(X_train)
y_pred_test=ridge.predict(X_test)

metric=[]
r2_train_lr=r2_score(y_train,y_pred_train)
print(r2_train_lr)
metric.append(r2_train_lr)

r2_test_lr=r2_score(y_test,y_pred_test)
print(r2_test_lr)
metric.append(r2_test_lr)

rss1_lr=np.sum(np.square(y_train-y_pred_train))
print(rss1_lr)
metric.append(rss1_lr)

rss2_lr=np.sum(np.square(y_test-y_pred_test))
print(rss2_lr)
metric.append(rss2_lr)

mse_train_lr=mean_squared_error(y_test,y_pred_test)
print(mse_train_lr)
metric.append(mse_train_lr**0.5)
```

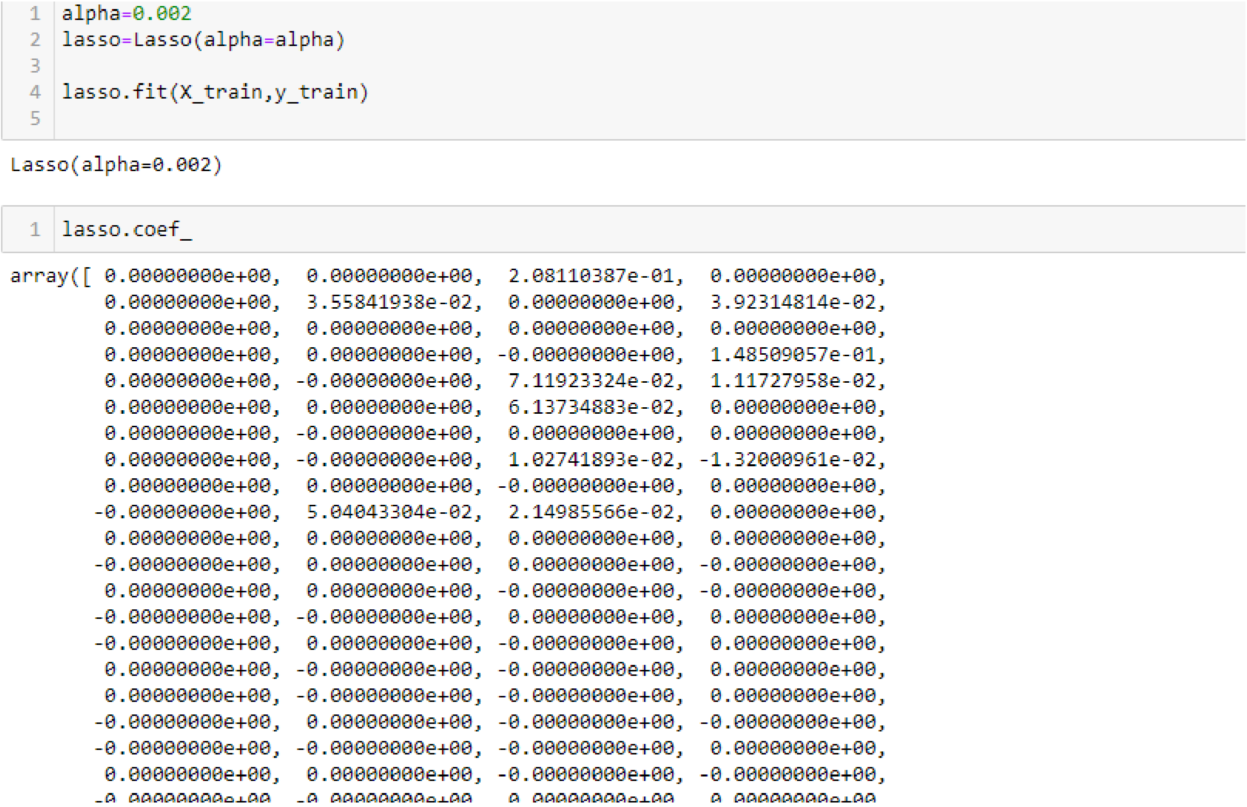
```
0.922422790160128
0.8263502696050091
0.9545897747272252
0.9439288884203207
0.002155088786347764
```



```python
model_parameter = list(ridge.coef_)
model_parameter.insert(0,ridge.intercept_)
cols = X_train.columns
cols.insert(0,'constant')
ridge_coef = pd.DataFrame(list(zip(cols,model_parameter)))
ridge_coef.columns = ['Feaure','Coef']
```

```python
ridge_coef.sort_values(by='Coef',ascending=False).head(10)
```

	Feaure	Coef
65	Exterior1st_AsphShn	0.856031
60	RoofMatl_Metal	0.803152
61	RoofMatl_Roll	0.757826
59	RoofMatl_Membran	0.752163
63	RoofMatl_WdShake	0.741896
62	RoofMatl_Tar&Grv	0.721044
64	RoofMatl_WdShngl	0.710225
16	BedroomAbvGr	0.211336
13	2ndFlrSF	0.185205
9	BsmtFinSF2	0.155989

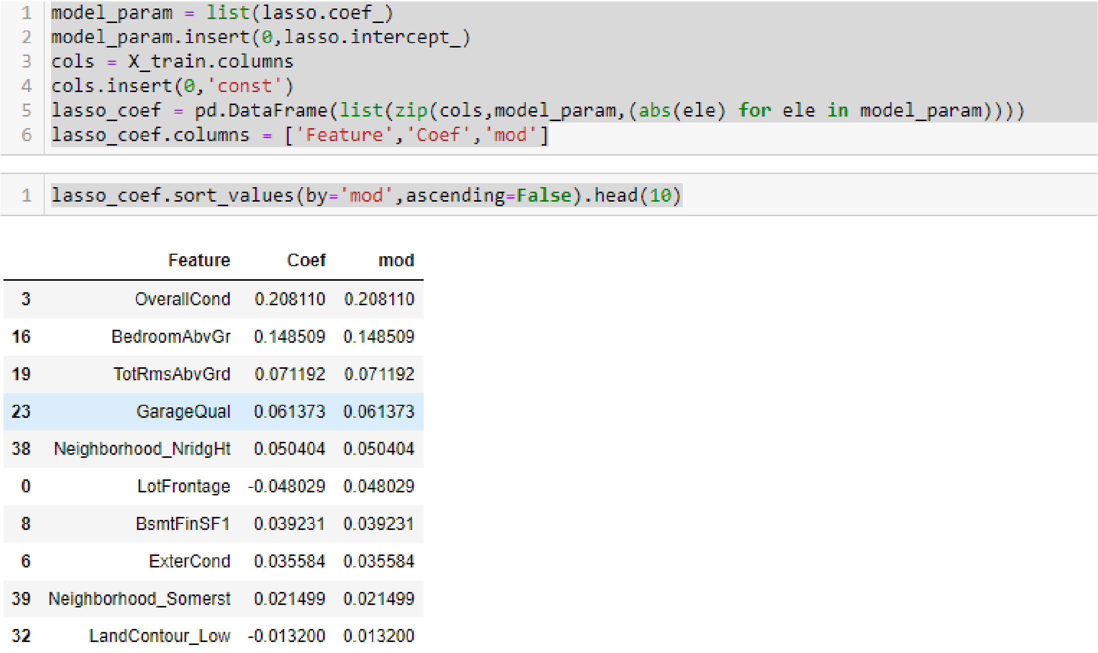


```python
alpha=0.002
lasso=Lasso(alpha=alpha)

lasso.fit(X_train,y_train)

```

```
Lasso(alpha=0.002)
```

```python
lasso.coef_
```

```
array([ 0.00000000e+00,  0.00000000e+00,  2.08110387e-01,  0.00000000e+00,
        0.00000000e+00,  3.55841938e-02,  0.00000000e+00,  3.92314814e-02,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00, -0.00000000e+00,  1.48509057e-01,
        0.00000000e+00, -0.00000000e+00,  7.11923324e-02,  1.11727958e-02,
        0.00000000e+00,  0.00000000e+00,  6.13734883e-02,  0.00000000e+00,
        0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00, -0.00000000e+00,  1.02741893e-02, -1.32000961e-02,
        0.00000000e+00,  0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00,  5.04043304e-02,  2.14985566e-02,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
        0.00000000e+00,  0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
       -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
        0.00000000e+00, -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
        0.00000000e+00, -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
       -0.00000000e+00, -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
       -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
```



```python
model_param = list(lasso.coef_)
model_param.insert(0,lasso.intercept_)
cols = X_train.columns
cols.insert(0,'const')
lasso_coef = pd.DataFrame(list(zip(cols,model_param,(abs(ele) for ele in model_param))))
lasso_coef.columns = ['Feature','Coef','mod']
```

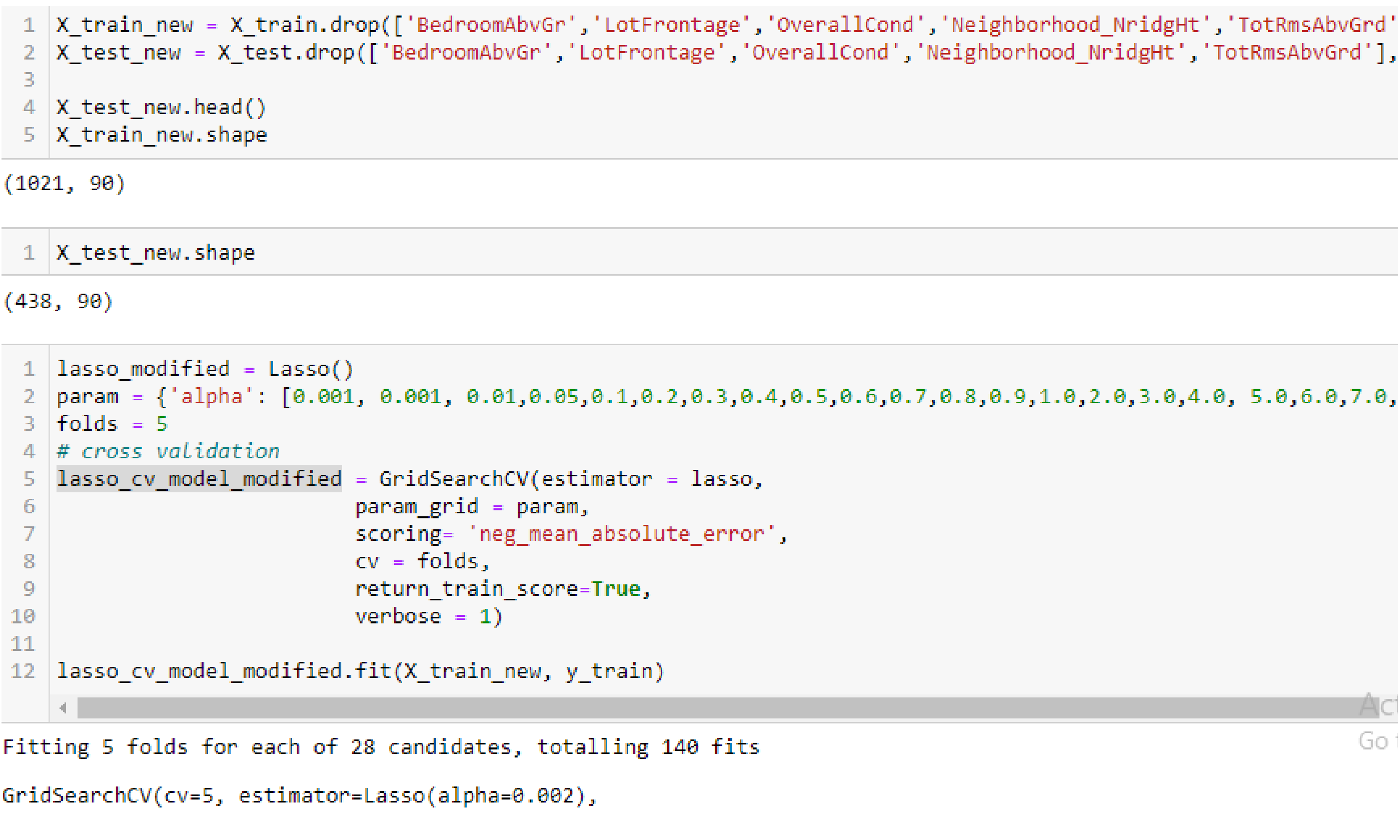
```python
lasso_coef.sort_values(by='mod',ascending=False).head(10)
```

	Feature	Coef	mod
3	OverallCond	0.208110	0.208110
16	BedroomAbvGr	0.148509	0.148509
19	TotRmsAbvGrd	0.071192	0.071192
23	GarageQual	0.061373	0.061373
38	Neighborhood_NridgHt	0.050404	0.050404
0	LotFrontage	-0.048029	0.048029
8	BsmtFinSF1	0.039231	0.039231
6	ExterCond	0.035584	0.035584
39	Neighborhood_Somerst	0.021499	0.021499
32	LandContour_Low	-0.013200	0.013200

Q2 ANS:

I will choose Lasso because r2 score of train and test are almost similar and also RSS, MSE small.

Q3 ANS



```
X_train_new = X_train.drop(['BedroomAbvGr','LotFrontage','OverallCond','Neighborhood_NridgHt','TotRmsAbvGrd'
X_test_new = X_test.drop(['BedroomAbvGr','LotFrontage','OverallCond','Neighborhood_NridgHt','TotRmsAbvGrd'],

X_test_new.head()
X_train_new.shape
```

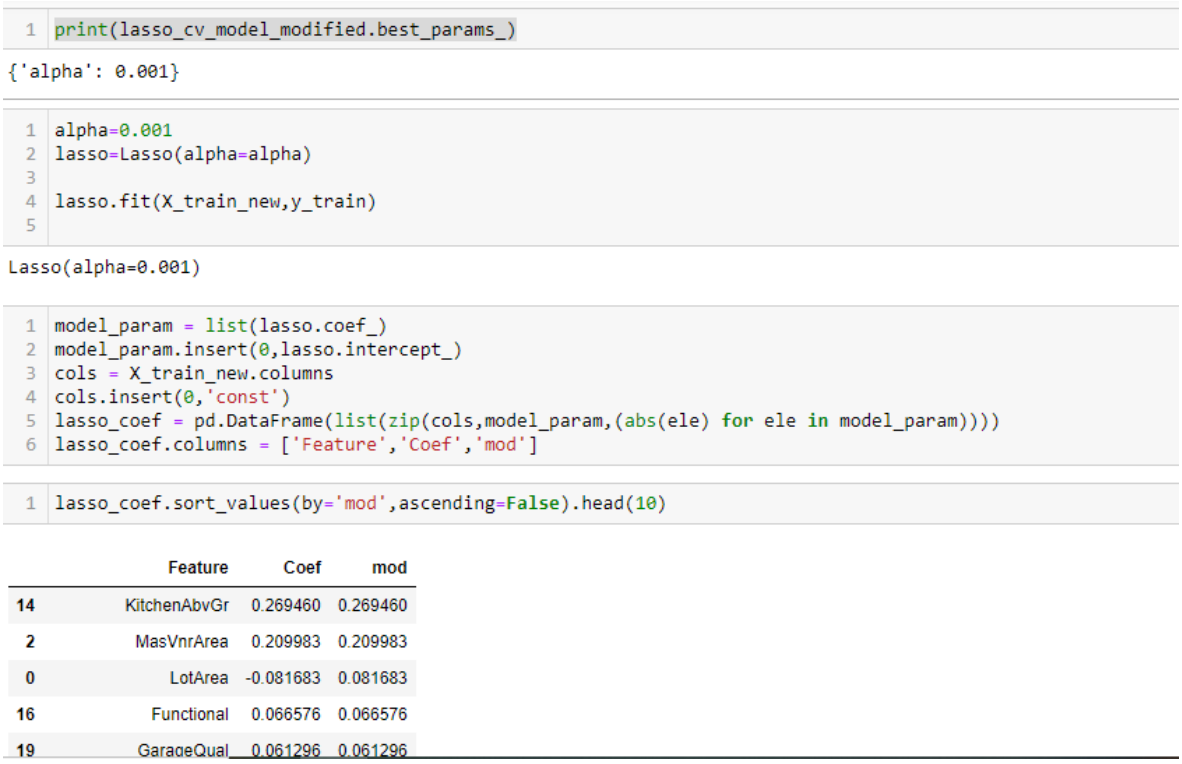
```
(1021, 90)
```

```
X_test_new.shape
```

```
(438, 90)
```

```
lasso_modified = Lasso()
param = {'alpha': [0.001, 0.001, 0.01,0.05,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,2.0,3.0,4.0, 5.0,6.0,7.0,
folds = 5
# cross validation
lasso_cv_model_modified = GridSearchCV(estimator = lasso,
                        param_grid = param,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)

lasso_cv_model_modified.fit(X_train_new, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits

GridSearchCV(cv=5, estimator=Lasso(alpha=0.002),
```



```
print(lasso_cv_model_modified.best_params_)
```

```
{'alpha': 0.001}
```

```
alpha=0.001
lasso=Lasso(alpha=alpha)

lasso.fit(X_train_new,y_train)

```

```
Lasso(alpha=0.001)
```

```
model_param = list(lasso.coef_)
model_param.insert(0,lasso.intercept_)
cols = X_train_new.columns
cols.insert(0,'const')
lasso_coef = pd.DataFrame(list(zip(cols,model_param,(abs(ele) for ele in model_param))))
lasso_coef.columns = ['Feature','Coef','mod']
```

```
lasso_coef.sort_values(by='mod',ascending=False).head(10)
```

	Feature	Coef	mod
14	KitchenAbvGr	0.269460	0.269460
2	MasVnrArea	0.209983	0.209983
0	LotArea	-0.081683	0.081683
16	Functional	0.066576	0.066576
19	GarageQual	0.061296	0.061296

Q4 ANS

I can say that model is strengthen after the ridge and lasso regression and the model have train, test almost similar values so my model is robust and it can be used in different models of different

Variables so that it is generalizable. The accuracy of the model in lasso is high and also in ridge.