Question 1:

What is the optimal value of alpha for ridge and lasso regression? What will be the changes in the model if you choose double the value of alpha for both ridge and lasso? What will be the most important predictor variables after the change is implemented?

Answer:

1. Optimal value for alpha in lasso is 0.001.
2. Optimal value for alpha in ridge is 1.0.

<u>For Lasso:</u>

```
lasso = Lasso(alpha=0.001)
lasso.fit(X_train,y_train)

y_train_pred = lasso.predict(X_train)
y_test_pred = lasso.predict(X_test)

print(r2_score(y_true=y_train,y_pred=y_train_pred))
print(r2_score(y_true=y_test,y_pred=y_test_pred))
```

```
0.8989346010264372
0.8463221330919919
```

```
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)))
lasso_coef.columns = ['Featuere','Coef']
```

```
lasso_coef.sort_values(by='Coef',ascending=False).head(5)
```

	Featuere	Coef
43	Exterior1st_BrkComm	1.254587
24	Neighborhood_StoneBr	0.372254
12	KitchenQual	0.361235
66	SaleType_Oth	0.352951
23	Neighborhood_NridgHt	0.338902

If alpha value is doubled for Lasso regression

```
lasso = Lasso(alpha=0.002)
lasso.fit(X_train,y_train)

y_train_pred = lasso.predict(X_train)
y_test_pred = lasso.predict(X_test)

print(r2_score(y_true=y_train,y_pred=y_train_pred))
print(r2_score(y_true=y_test,y_pred=y_test_pred))
```

```
0.8916888077199777
0.8508299549977508
```

```
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)))
lasso_coef.columns = ['Featuere','Coef']
```

```
lasso_coef.sort_values(by='Coef',ascending=False).head(5)
```

	Featuere	Coef
43	Exterior1st_BrkComm	1.254587
24	Neighborhood_StoneBr	0.372254
12	KitchenQual	0.361235
66	SaleType_Oth	0.352951
23	Neighborhood_NridgHt	0.338902

- By increasing the alpha value for lasso, r2 score for train and test data set doesn't impact much. However Coef of parameters have been decreased and also the parameter contribution order has been changed.
- Most important predictor variables after the change is implemented are shown in the figure.

For Ridge:

```
ridge = Ridge(alpha = 1.0)
ridge.fit(X_train,y_train)

y_pred_train = ridge.predict(X_train)
print(r2_score(y_train,y_pred_train))

y_pred_test = ridge.predict(X_test)
print(r2_score(y_test,y_pred_test))
```

```
0.9035088063630226
0.8406125658726896
```

```
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']
```

```
ridge_coef.sort_values(by='Coef',ascending=False).head(5)
```

	Feaure	Coef
43	Exterior1st_BrkComm	1.500835
24	Neighborhood_StoneBr	0.472051
23	Neighborhood_NridgHt	0.386557
25	Condition1_Feedr	0.384068
66	SaleType_Oth	0.380452

If alpha value is doubled for Ridge regression

```
ridge = Ridge(alpha = 2.0)
ridge.fit(X_train,y_train)

y_pred_train = ridge.predict(X_train)
print(r2_score(y_train,y_pred_train))

y_pred_test = ridge.predict(X_test)
print(r2_score(y_test,y_pred_test))
```

```
0.8995436773424049
0.8470036972644295
```

```
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']
```

```
ridge_coef.sort_values(by='Coef',ascending=False).head(5)
```

	Feaure	Coef
43	Exterior1st_BrkComm	1.225736
24	Neighborhood_StoneBr	0.446637
23	Neighborhood_NridgHt	0.380133
66	SaleType_Oth	0.368091
25	Condition1_Feedr	0.348490

- Similarly by increasing the alpha value for ridge, r2 score for train and test data set doesn't impact much. However Coef of parameters have been decreased and also the parameter contribution order has been changed.

Below Table shows the important parameter when alpha is doubled.

Lasso		Ridge	
Alpha = 0.001	Alpha = 0.002	Alpha = 1.0	Alpha = 2.0
Exterior1st_BrkComm	Exterior1st_BrkComm	Exterior1st_BrkComm	Exterior1st_BrkComm
Neighborhood_StoneBr	Neighborhood_StoneBr	Neighborhood_StoneBr	Neighborhood_StoneBr
KitchenQual	KitchenQual	Neighborhood_NridgHt	Neighborhood_NridgHt
SaleType_Oth	SaleType_Oth	Condition1_Feedr	SaleType_Oth
Neighborhood_NridgHt	Neighborhood_NridgHt	SaleType_Oth	Condition1_Feedr

Question 2:

You have determined the optimal value of lambda for ridge and lasso regression during the assignment. Now, which one will you choose to apply and why?

Answer:

As results for both lasso and redge are almost same.

Below are few observations.

1. As we can see from the data, most of the parameters affecting the result in low to medium range. So, Ridge will perform better.
2. Lasso can also perform well as it penalizes the parameter heavily.
3. Elastic Net can also be used by mixing the parameter alpha between lasso and ridge.

Question 3:

After building the model, you realised that the five most important predictor variables in the lasso model are not available in the incoming data. You will now have to create another model excluding the five most important predictor variables. Which are the five most important predictor variables now?

Answer:

In [360]:
```
X_train_lasso = X_train.drop(['Exterior1st_BrkComm','Neighborhood_StoneBr','KitchenQual','SaleType_Oth','Neighborhood_NridgHt'],a
X_test_lasso = X_test.drop(['Exterior1st_BrkComm','Neighborhood_StoneBr','KitchenQual','SaleType_Oth','Neighborhood_NridgHt'],axi
```

In [370]:
```
lasso = Lasso(alpha=0.001)
lasso.fit(X_train_lasso,y_train)

y_train_pred = lasso.predict(X_train_lasso)
y_test_pred = lasso.predict(X_test_lasso)

print(r2_score(y_true=y_train,y_pred=y_train_pred))
print(r2_score(y_true=y_test,y_pred=y_test_pred))
```

```
0.88614547224319
0.8261363400011135
```

In [371]:
```
model_param = list(lasso.coef_)
model_param.insert(0,lasso.intercept_)
cols = X_train_lasso.columns
cols.insert(0,'const')
lasso_coef = pd.DataFrame(list(zip(cols,model_param)))
lasso_coef.columns = ['Featuere','Coef']
```

In [372]:
```
lasso_coef.sort_values(by='Coef',ascending=False).head(5)
```

Out[372]:

	Featuere	Coef
40	Exterior1st_BrkFace	1.471939
62	SaleCondition_AdjLand	0.450180
12	MSZoning_FV	0.373858
3	BsmtQual	0.258615
22	Condition1_Feedr	0.246711

After removing the five most important predictor variables in lasso, we can see that there is a drop in R2 score for train and test. Below are the five most important predictor variables now.

- Exterior1st_BrkFace
- SaleCondition_AdjLand
- MSZoning_FV
- BsmtQual
- Condition1_Feedr

Question 4:

How can you make sure that a model is robust and generalisable? What are the implications of the same for the accuracy of the model and why?

Answer:

Both Robustness and Accuracy are important for any model.

For robustness, we have given less points to outliers also removed some outliers. Then using RFE, we have figured out most important 70 parameter which is a huge number itself. So the probability of the parametes affecting the price is in equilibrium. In case of training data prediction, it shows approx 90% which means model has not mugged up thee data. Model has the flexibility react when there is a change in any parameter. Hence, we can consider the model as robust and generalized to perform with unseen data.

For Accuracy, we have to check the prediction with actual price for unseen data. We can see that model is giving approx 85% for the unseen data where training prediction is 90%. We can increase the accuracy by increasing the number of affecting parameters. Also we can have a good look into the outliers. We can reduce more outliers, if these are not co-related.