Assignment - Part 2 Questionnaires

Q1. 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?

- **Ridge regression**: -optimal alpha is **9**
- **Lasso regression**: -optimal alpha is **0.001**

In [117]:
```
#Q1
##Change the alpha value from 9 to 18 [double the initial alpha]
alpha = 18

ridge2 = Ridge(alpha=alpha)

ridge2.fit(X_train,y_train)
```

Out[117]:
```
Ridge(alpha=18)
```

In [118]:
```
# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = ridge2.predict(X_train)
y_pred_test = ridge2.predict(X_test)

metric2 = []
r2_train_lr = metrics.r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric2.append(r2_train_lr)

r2_test_lr = metrics.r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric2.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric2.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric2.append(rss2_lr)

mse_train_lr = metrics.mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric2.append(mse_train_lr**0.5)

mse_test_lr = metrics.mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric2.append(mse_test_lr**0.5)

#Alpha 9

# R2 Train 93.8%
# R2 Test 92.2%
```

```
0.9317264720456169
0.9198286734264514
10.369626757208717
2.8963549850865244
0.010348928899409898
0.011539262888790935
```

Rsquare of training and testing data has decreased

In [119]:
```
#Changed alpha 0.001 to 0.002 [double the initial alpha]
alpha =0.002
lasso20 = Lasso(alpha=alpha)
lasso20.fit(X_train, y_train)
```

Out[119]:
```
Lasso(alpha=0.002)
```

In [120]:
```
# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = lasso20.predict(X_train)
y_pred_test = lasso20.predict(X_test)

metric3 = []
r2_train_lr = metrics.r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric3.append(r2_train_lr)

r2_test_lr = metrics.r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric3.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric3.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric3.append(rss2_lr)

mse_train_lr = metrics.mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

mse_test_lr = metrics.mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric3.append(mse_test_lr**0.5)

#Alpha 0.001

# R2 Train 92.3%
# R2 Test 91.8%
```

```
0.9043512080759172
0.9085440122655303
14.527479416229339
3.30403670878186
0.014498482451326684
0.013163492863672748
```

Rsquare of training and testing data has decreased

In [121]:
```
#important predictor variables
betas = pd.DataFrame(index=X_train.columns)
betas.rows = X_train.columns
betas['Ridge2'] = ridge2.coef_
betas['Ridge'] = ridge.coef_
betas['Lasso'] = lasso.coef_
betas['Lasso20'] = lasso20.coef_
pd.set_option('display.max_rows', None)
betas.head(20)
```

Out[121]:

	Ridge2	Ridge	Lasso	Lasso20
LotFrontage	0.000037	0.000036	0.000040	0.000030
LotArea	0.000007	0.000007	0.000007	0.000008
MasVnrArea	0.000032	0.000022	0.000017	0.000019
BsmtFinSF1	0.000067	0.000060	0.000150	0.000171
BsmtFinSF2	0.000063	0.000068	0.000100	0.000109
BsmtUnfSF	-0.000022	-0.000024	0.000055	0.000065
TotalBsmtSF	0.000108	0.000104	0.000044	0.000051
CentralAir	0.047394	0.048425	0.043427	0.051892
1stFlrSF	0.000126	0.000118	0.000272	0.000267
2ndFlrSF	0.000119	0.000112	0.000249	0.000267
LowQualFinSF	-0.000068	-0.000053	0.000068	0.000066
GrLivArea	0.000176	0.000177	0.000044	0.000047
GarageYrBlt	0.000101	0.000114	0.000122	0.000176
GarageArea	0.000098	0.000081	0.000143	0.000179
WoodDeckSF	0.000082	0.000084	0.000087	0.000079
OpenPorchSF	0.000121	0.000103	0.000155	0.000182
EnclosedPorch	0.000167	0.000163	0.000207	0.000207
3SsnPorch	0.000226	0.000217	0.000240	0.000266
ScreenPorch	0.000000	0.000000	0.000000	0.000000
PoolArea	0.000000	0.000000	0.000000	0.000000

Top 10 features of ridge for alpha 18

The most important variable after the changes has been implemented for ridge regression are as

- Neighborhood_Crawfor
- Functional_Typ
- OverallCond_9
- OverallCond_8
- OverallQual_9
- SaleCondition_Normal
- Neighborhood_StoneBr
- OverallCond_7
- OverallQual_8
- SaleCondition_Partial

In [122]:
```
#top 10 Features after alpha 18 for Ridge
(betas[['Ridge2']].sort_values(by=['Ridge2'], ascending = False)).head(10)
```

Out[122]:

	Ridge2
Neighborhood_Crawfor	0.078860
Functional_Typ	0.077721
OverallCond_9	0.074737
OverallCond_8	0.071730
OverallQual_9	0.061549
SaleCondition_Normal	0.059855
Neighborhood_StoneBr	0.054949
OverallCond_7	0.052231
OverallQual_8	0.050335
SaleCondition_Partial	0.048681

Top 10 features of lasso for alpha 0.002

The most important variable after the changes has been implemented for lasso regression are as

- SaleCondition_Partial
- Neighborhood_Crawfor
- Functional_Typ
- OverallQual_9
- OverallCond_8
- CentralAir
- SaleCondition_Normal
- OverallCond_7
- GarageCond_TA
- OverallQual_8

In [123]:
```
#top 10 Features after alpha 0.002 for lasso
(betas[['Lasso20']].sort_values(by=['Lasso20'], ascending = False)).head(10)
```

Out[123]:

	Lasso20
SaleCondition_Partial	0.083015
Neighborhood_Crawfor	0.082898
Functional_Typ	0.080223
OverallQual_9	0.060670
OverallCond_8	0.056270
CentralAir	0.051892
SaleCondition_Normal	0.049381
OverallCond_7	0.049005
GarageCond_TA	0.048400
OverallQual_8	0.038010

Top 10 features of lasso for alpha 0.001

In [124]:
```
(betas[['Lasso']].sort_values(by=['Lasso'], ascending = False)).head(10)
```

Out[124]:

	Lasso
OverallCond_9	0.114314
Neighborhood_Crawfor	0.110850
OverallQual_9	0.108165
SaleCondition_Partial	0.103962
Functional_Typ	0.085734
OverallCond_8	0.077962
SaleCondition_Normal	0.064210
Neighborhood_StoneBr	0.061873
OverallCond_7	0.056507
OverallQual_8	0.054637

Top 10 features of ridge for alpha 9

In [125]:
```
(betas[['Ridge']].sort_values(by=['Ridge'], ascending = False)).head(10)
```

Out[125]:

	Ridge
OverallCond_9	0.103396
Neighborhood_Crawfor	0.098124
Functional_Typ	0.087704
OverallCond_8	0.083031
OverallQual_9	0.080163
Neighborhood_StoneBr	0.078575
SaleCondition_Normal	0.068712
SaleCondition_Partial	0.061341
OverallQual_8	0.058988
MSZoning_FV	0.057750

Q2. 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?

index	LR	Ridge_RFE	Ridge	Lasso
R2 Train	95.3%	94.8%	93.8%	92.3%
R2 Test	88.5%	91.9%	92.2%	91.8%

Based on the Rsquare value on training data we can say that the Ridge is better. Where as if we compare the variation in R2 for training and test Lasso performs better, I will choose **Lasso regression** as would be a better option it helps in feature elimination and the model will be robust.

Q3. 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?

In [126]:
```
# Droping top 5 predictor
X_train1 = X_train.drop(['OverallCond_9','Neighborhood_Crawfor','OverallQual_9','SaleCondition_Partial','Functio
X_test1 = X_test.drop(['OverallCond_9','Neighborhood_Crawfor','OverallQual_9','SaleCondition_Partial','Functiona
```

In [127]:
```
alpha =0.001
lasso2 = Lasso(alpha=alpha)
lasso2.fit(X_train1, y_train)
```

Out[127]:
```
Lasso(alpha=0.001)
```

In [128]:
```
# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = lasso2.predict(X_train1)
y_pred_test = lasso2.predict(X_test1)

metric3 = []
r2_train_lr = metrics.r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric3.append(r2_train_lr)

r2_test_lr = metrics.r2_score(y_test, y_pred_test)
print(r2_test_lr)
metric3.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(rss1_lr)
metric3.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(rss2_lr)
metric3.append(rss2_lr)

mse_train_lr = metrics.mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

mse_test_lr = metrics.mean_squared_error(y_test, y_pred_test)
print(mse_test_lr)
metric3.append(mse_test_lr**0.5)
#Alpha 0.001

# R2 Train 92.3%
# R2 Test 91.8%
```

```
0.914050745999649
0.9134752989892844
13.054279026560103
3.125883776858834
0.013028222581397308
0.01245372022652922
```

Rsquare of training and testing data has decreased

In [129]:
```
#important predictor variables
betas = pd.DataFrame(index=X_train1.columns)
betas.rows = X_train1.columns
betas['Lasso2'] = lasso2.coef_
pd.set_option('display.max_rows', None)
(betas[['Lasso2']].sort_values(by=['Lasso2'], ascending = False)).head(10)
```

Out[129]:

	Lasso2
SaleType_New	0.086907
OverallCond_8	0.066005
Neighborhood_StoneBr	0.064343
SaleCondition_Normal	0.055949
GarageCond_TA	0.051816
CentralAir	0.049147
Condition1_Norm	0.048650
OverallCond_7	0.045727
MSZoning_FV	0.044892
MSSubClass_70	0.044450

Five most important predictor variables

- SaleType_New
- OverallCond_8
- Neighborhood_StoneBr
- SaleCondition_Normal
- GarageCond_TA

Q4. 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?

The model needs to be robust and generalizable so that outliers in the training data do not impact. The model also has to be generalisable so that the test accuracy and training score are in expectable margin. The model needs to be accurate for datasets other than the ones which were used for training with is the test.

Outlier analysis needs to be done and only those which are relevant to the dataset need to be kept, as the outlier may affect the accuracy.As the model needs to be robust for predictive analysis.