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 of alpha :
    - Value of alpha for Ridge is 7.0
    - Value of alpha for Lasso is 0.001

2. Change in model if we double the value of alpha i.e. for Ridge(2*7=14) and Lasso (2*0.001=0.002)

R2 value for Ridge and Lasso with original values i.e. 7.0 and 0.001

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	0.830576	0.828631	0.829553
1	R2 Score (Test)	0.824461	0.825060	0.824574

R2 value after increasing the alpha value

Ridge:

In [303]:
```
# Ridge

alpha = 14
ridge = Ridge(alpha=alpha)

ridge.fit(X_train_new, y_train)
print(ridge.coef_)
y_pred_train = ridge.predict(X_train_new)
y_pred_test = ridge.predict(X_test_new)

r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
```

```
[ 0.4218449   0.33471306  0.23823131 -0.38248909 -0.14923465  0.23193081
  0.20459064 -0.24335936  0.23757521  0.16673668  0.09138682 -0.1011874
  0.13740151 -0.2316309   0.23342775  0.267126   -0.2281507  -0.22490086
 -0.27245768]
0.8262509947895839
0.8245307024040489
```

Lasso:

In [299]:
```
# Lasso

alpha =0.002

lasso = Lasso(alpha=alpha)

lasso.fit(X_train_new, y_train)
print(lasso.coef_)
y_pred_train = lasso.predict(X_train_new)
y_pred_test = lasso.predict(X_test_new)

r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)
```

```
[ 0.41708432  0.33424944  0.2412465  -0.40620018 -0.13423632  0.28415305
  0.23612476 -0.25679621  0.25937092  0.16968531  0.03424831 -0.
  0.1342917  -0.28916172  0.23595956  0.28128563 -0.26621128 -0.25772767
 -0.29424664]
0.8273462939040661
0.8235434500500931
```

3. Top 5 predictor after change is done are as follows : OverallQual, MSZoning_RM, 1stFlrSF, Neighborhood_ClearCr, GarageType_NA

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:

If we look at the below metrics which we have implemented during the assignment

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	0.830576	0.828631	0.829553
1	R2 Score (Test)	0.824461	0.825060	0.824574
2	RSS (Train)	172.982398	174.967281	174.026244
3	RSS (Test)	80.823039	80.547374	80.771042
4	MSE (Train)	0.411612	0.413967	0.412852
5	MSE (Test)	0.429077	0.428345	0.428939

As seen above R2 value is better in Ridge regression while comparing it with Lasso Regression but we will choose Lasso as we not see much difference because its giving feature selection option also. It has removed unwanted features from model without affecting the model accuracy. Which makes are model generalized and simple and accurate

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:

Dropping below features:

OverallQual, 1stFlrSF, MSZoning_RM, MSZoning_RM , Exterior2nd_Stucco and building the model again

```
In [307]: #OverallQual, 1stFlrSF, MSZoning_RM, MSZoning_RM , Exterior2nd_Stucco

In [308]: X_train_new = X_train_new.drop(['OverallQual', '1stFlrSF', 'MSZoning_RM', 'MSZoning_RM' , 'Exterior2nd_Stucco'],axis=1)

In [310]: X_test_new = X_test_new.drop(['OverallQual', '1stFlrSF', 'MSZoning_RM', 'MSZoning_RM' , 'Exterior2nd_Stucco'],axis=1)

In [311]: # Lasso

alpha =0.001

lasso = Lasso(alpha=alpha)

lasso.fit(X_train_new, y_train)
print(lasso.coef_)
y_pred_train = lasso.predict(X_train_new)
y_pred_test = lasso.predict(X_test_new)

r2_train_lr = r2_score(y_train, y_pred_train)
print(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print(r2_test_lr)

[ 0.2856664  -0.90377789  0.41581989  0.5690433  -0.37398965  1.18644364
  0.61186618  1.12285138 -0.18521784  0.6581996   0.53405839  0.86285695
 -0.57336942 -0.56885612 -0.74659245]
0.5417314421098622
0.5206449441707175
```

We have observed that R2 value of train and test got reduced to 0.54 and 0.52

New top 5 predictors now are as follows:

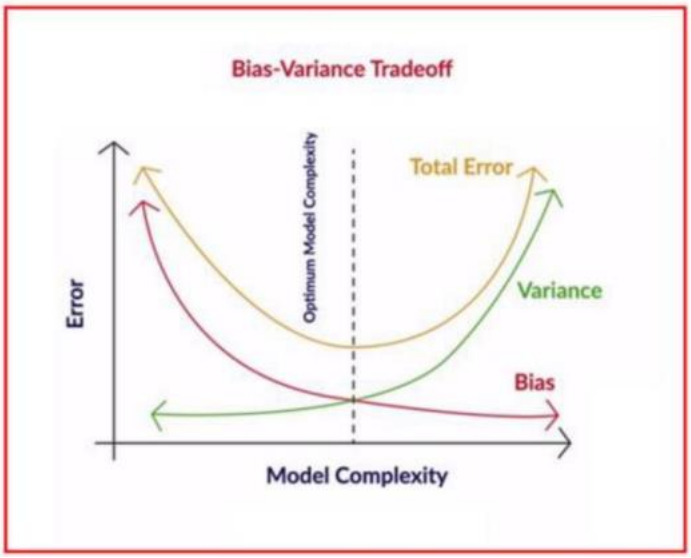
Neighborhood_BrDale, BsmtExposure_Gd, Exterior1st_BrkFace, Neighborhood_Somerst, GarageType_NA

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:

Robustness of a model implies, either the testing error of the model is consistent with the training error, the model performs well with enough stability even after adding some noise to the dataset. Thus, the robustness (or generalizability) of a model is a measure of its successful application to data sets other than the one used for training and testing. By the implementing regularization techniques, we can control the trade-off between model complexity and bias which is directly connected the robustness of the model. Regularization, helps in penalizing the coefficients for making the model too complex; thereby allowing only the optimal amount of complexity to the model. It helps in controlling the robustness of the model by making the model optimal simpler. Therefore, in order to make the model more robust and generalizable, one need to make sure that there is a delicate balance between keeping the model simple and not making it too naive to be of any use. Also, making a model simple leads to BiasVariance Trade-off: ➢ A complex model will need to change for every little change in the dataset and hence is very unstable and extremely sensitive to any changes in the training data. ➢ A simpler model that abstracts out some pattern followed by the data points given is unlikely to change wildly even if more points are added or removed. Bias helps you quantify, how accurate is the model likely to be on test data. A complex model can do an accurate job prediction provided there has to be enough training data. Models that are too naïve, for e.g., one that gives same results for all test inputs and makes no discrimination whatsoever has a very large bias as its expected error across all test inputs are very high. Variance is the degree of changes in the model itself with respect to changes in the training data. Thus, accuracy of the model can be maintained by keeping the balance between Bias and Variance as it minimizes the total error as shown in the below graph.


Bias-Variance Tradeoff
Error
Optimum Model Complexity
Total Error
Variance
Bias
Model Complexity