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:

Ridge:

```
[72]: #Change the alpha value from 0.2 to 0.4
alpha = 0.4
ridge2 = Ridge(alpha=alpha)
ridge2.fit(X_train, y_train)
```

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

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

r2_test_lr = 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 = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric2.append(mse_train_lr**0.5)

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

```
0.8493925315158765
0.7227178594455519
697766171149.2766
540181391563.0237
683414467.3352367
1233290848.3174057
```

R2score on training data has decreased but it has increased on testing data

Lasso

```
[84]: #Changed alpha 10 to 20
alpha =20
lasso20 = Lasso(alpha=alpha)
lasso20.fit(X_train, y_train)
```

```
[84]: Lasso(alpha=20)
```

```
[87]: # 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 = r2_score(y_train, y_pred_train)
print(r2_train_lr)
metric3.append(r2_train_lr)

r2_test_lr = 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 = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

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

```
0.8486495212465435
0.7069829335203139
701208546457.5881
570835057772.6117
686786039.6254536
1303276387.6087024
```

score of training data has decrease and it has increase on testing data

Most important predictor variables after changing Alpha are

LotArea---------------Lot size in square feet

OverallQual---------Rates the overall material and finish of the house

OverallCond--------Rates the overall condition of the house

YearBuilt-------------Original construction date

BsmtFinSF1--------Type 1 finished square feet

TotalBsmtSF------- Total square feet of basement area

GrLivArea-----------Above grade (ground) living area square feet

TotRmsAbvGrd----Total rooms above grade (does not include bathrooms)

Street_Pave--------Pave road access to property

RoofMatl_Metal----Roof material_Metal

Predictors are same but the coefficent of these predictor has changed

<u>Question 2</u>
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?

<u>Answer:</u>
The optimal lambda value in case of Ridge and Lasso is as below:
Ridge - 0.5
Lasso - 20

The R Squared value in case of Ridge and Lasso are:
Ridge - 0.89647
Lasso - 0.89664

Lasso's R Squared Value is somewhat higher than that of Ridge.

Lasso also has an advantage over Ridge because it aids in feature reduction (when the coefficient value of one of the features becomes 0).

According to Lasso, the zoning classification, living area square feet, overall quality and condition of the house, and other factors all determine the price. The house's foundation type, The number of automobiles that can fit in the garage, the total basement space in square feet, and the finished square feet area of the basement.

<u>Question 3</u>

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.

<u>Answer:</u>

In [93]:
```
X_train.columns
```

Out[93]:
```
Index(['MSSubClass_90', 'MSSubClass_160', 'MSSubClass_180', 'MSSubClass_190',
       'Neighborhood_ClearCr', 'Neighborhood_Crawfor', 'Neighborhood_NoRidge',
       'Condition1_PosA', 'Condition2_PosN', 'Condition2_RRAe',
       'BldgType_2fmCon', 'BldgType_Duplex', 'OverallQual_2', 'OverallQual_3',
       'OverallQual_4', 'OverallQual_5', 'OverallQual_6', 'OverallQual_7',
       'OverallQual_8', 'OverallQual_9', 'OverallQual_10', 'OverallCond_2',
       'OverallCond_3', 'OverallCond_4', 'OverallCond_5', 'OverallCond_6',
       'OverallCond_7', 'OverallCond_8', 'OverallCond_9', 'RoofMatl_CompShg',
       'RoofMatl_Membran', 'RoofMatl_Metal', 'RoofMatl_Roll',
       'RoofMatl_Tar&Grv', 'RoofMatl_WdShake', 'RoofMatl_WdShngl',
       'Exterior1st_AsphShn', 'Exterior1st_CBlock', 'Exterior2nd_AsphShn',
       'Exterior2nd_CBlock', 'ExterQual_Fa', 'Foundation_Wood', 'BsmtQual_Fa',
       'BsmtQual_Gd', 'BsmtQual_TA', 'BsmtCond_Po', 'BsmtFinType2_Unf',
       'BsmtFinSF2_>0', 'Heating_OthW', 'HeatingQC_Po', 'BsmtFullBath_2',
       'FullBath_2', 'FullBath_3', 'BedroomAbvGr_8', 'KitchenAbvGr_1',
       'KitchenAbvGr_2', 'KitchenAbvGr_3', 'KitchenQual_Fa', 'KitchenQual_Gd',
       'KitchenQual_TA', 'TotRmsAbvGrd_14', 'Functional_Sev', 'Fireplaces_3',
       'GarageCars_4', 'GarageQual_Fa', 'GarageQual_Gd', 'GarageQual_Po',
       'GarageQual_TA', 'GarageCond_Fa', 'GarageCond_Gd', 'GarageCond_Po',
       'GarageCond_TA', 'PoolQC_Gd', 'SaleType_New', 'SaleCondition_Partial'],
      dtype='object')
```

'MSSubClass_90', 'MSSubClass_160', 'MSSubClass_180', 'MSSubClass_190','Neighborhood_ClearCr' are the top 5 important predictors.

Dropping top 5 varaibles

In [96]:
```
X_train2 = X_train.drop(['MSSubClass_90', 'MSSubClass_160', 'MSSubClass_180', 'MSSubClass_190',
       'Neighborhood_ClearCr'],axis=1)
X_test2 = X_test.drop(['MSSubClass_90', 'MSSubClass_160', 'MSSubClass_180', 'MSSubClass_190',
       'Neighborhood_ClearCr'],axis=1)
```

Applying Lasso to find new top 5 predictors ¶

7]:
```
# alpha 10
alpha =10
lasso21 = Lasso(alpha=alpha)
lasso21.fit(X_train2, y_train)
```

7]:
```
Lasso(alpha=10)
```

8]:
```
# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = lasso21.predict(X_train2)
y_pred_test = lasso21.predict(X_test2)

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

r2_test_lr = 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 = mean_squared_error(y_train, y_pred_train)
print(mse_train_lr)
metric3.append(mse_train_lr**0.5)

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

```
0.8323327415310201
0.6628226186749455
776804378604.4325
656865049741.5614
760827011.3657517
1499691894.3871264
```

R2score of training and testing data has decreased

```
In [100]: #important predictor variables
betas = pd.DataFrame(index=X_train2.columns)
betas.rows = X_train.columns
betas['Lasso21'] = lasso21.coef_
pd.set_option('display.max_rows', None)
betas.head(68)
```

OverallCond_3	-26552.180698
OverallCond_4	-8225.127971
OverallCond_5	-2841.731594
OverallCond_6	0.000000
OverallCond_7	4185.731603
OverallCond_8	1872.858280
OverallCond_9	24377.659848
RoofMatl_CompShg	-17460.926871
RoofMatl_Membran	35455.462418
RoofMatl_Metal	19544.315564
RoofMatl_Roll	-0.000000
RoofMatl_Tar&Grv	20489.118676
RoofMatl_WdShake	-1489.202852

The new five most important predictor variables

OverallQual_8

OverallQual_9

OverallQual_10

BsmtFullBath_2

BsmtFinSF2_>0

Question 4

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

Answer:

According to Occam's Razor, if two models have similar 'performance' with finite training or test data, we should choose the one that makes less test data errors for the following reasons: - • Simpler models are more 'generic' and applicable to a wider range of situations.

• Simpler models require fewer training samples to be effective, making them easier to train than more complex models.

• Models that are simpler are more reliable.

o Changes in the training data set cause complex models to shift dramatically.

o Simple models have a low variance but a high bias, while complicated models have a low variance but a high bias.

• In the training set, simpler models make more errors. Over fitting is caused by complex models, which work well for training samples but fail miserably when applied to real-world data.

To make the model more robust and generalizable, keep it simple, but not too simple, as it would be useless.

To make the model easier to understand, regularisation might be applied. Regularization aids in striking the difficult balance between keeping the model basic while still ensuring that it is not too naive to be useful. Regularization in regression entails adding a regularisation factor to the cost that adds up the absolute values or squares of the model's parameters.

Furthermore, simplifying a model results in a Bias-Variance Trade-off:

• Because a complicated model must alter for every small change in the dataset, it is particularly unstable and sensitive to changes in the training data.

• Even if more data points are added or withdrawn, a simpler model that abstracts out some pattern followed by the data points presented is unlikely to vary dramatically.

Bias is a metric that measures how accurate a model is on test data. If there is enough training data, a complicated model can provide an accurate job forecast. Models that be overly naive, for example,