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?

ANS: The optimal value of alpha for ridge is 1.0 and for lasso is 10

When alpha is doubled for both ridge and lasso:

```
In [80]: alpha = 2
         ridge_double = Ridge(alpha=alpha)
         ridge_double.fit(X_train1, y_train)
         ridge_double.coef_

Out[80]: array([ 55922.64099152, 110944.01449034,  33226.59346912,  54344.57360742,
                 52663.73120259,  74096.7077244 ,  71476.12308962,  35224.75935294,
                 85326.4150886 , -44604.71580078,  53633.21011303,  40419.43203757,
                -21531.67739208,  -5843.96036449,   7274.21797607,  11164.95960847,
                -23655.80506081, -21223.13372113, -51867.90207426, -60497.04412176,
                 -4021.78699852,  -6282.92559451, -15094.63922484, -20812.38112219,
                 16458.79375822])
```

```
In [81]: # r2 score , RSS , MSE
         y_pred_train = ridge_double.predict(X_train1)
         y_pred_test = ridge_double.predict(X_test1)

         metric4 = []
         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)

         0.882087717315285
         0.8710808825348301
         596084124320.2523
         320797350989.88525
         667507418.0517943
         729084888.6133755
```

Ans:

For ridge regression, when alpha is doubled RSS is slight decreased for train data but increase for test data

In [83]:
```python
alpha = 20
lasso_double = Lasso(alpha=alpha)
lasso_double.fit(X_train1 , y_train)
```
Out[83]:
```
Lasso(alpha=20)
```

In [84]:
```python
lasso_double.coef_
```
Out[84]:
```
array([ 63617.88766866, 121719.07214784,  36948.76523524,  53764.54809509,
         50458.15381368,  78209.33350175,   8244.95814088,      0.        ,
        162804.6803033 , -61134.17037463,  50757.77487375,  59515.00105242,
        -29661.61477569, -11645.85579454,   1966.05833938,  16580.03100738,
        -59674.58728343, -49678.51453129, -57016.33603395, -63508.82903034,
            -0.        ,  -4450.46804293, -31654.7831583 , -30830.83079772,
         21222.40311262])
```

In [85]:
```python
# Lets calculate some metrics such as R2 score, RSS and RMSE

y_pred_train = lasso_double.predict(X_train1)
y_pred_test = lasso_double.predict(X_test1)

metric5 = []
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)
```
```
0.8854019697956436
0.8670105921065013
579329522996.7144
330925704432.2682
648745266.5136778
752103873.7097005
```

- For Lasso regression, when alpha is doubled R2-square for train data is slightly reduced and for test data slightly increased.

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?

Ans: Lasso method is better than Ridge in terms of not only reducing the high values of coefficients but setting them to zero equivalent.

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?

- The top 5 features in Lasso are : LotArea,OverallQual,YearBuilt,BsmtFinSF1,TotalBsmtSF.

In [88]:
```
# dropping the top 5 features
X_train_2 = X_train1.drop(['LotArea','OverallQual','YearBuilt','BsmtFinSF1','TotalBsmtSF'],axis=1)
X_test_2 = X_test1.drop(['LotArea','OverallQual','YearBuilt','BsmtFinSF1','TotalBsmtSF'],axis=1)
```

In [89]:
```
X_train_2.head()
```

Out[89]:

	OverallCond	1stFlrSF	2ndFlrSF	GrLivArea	BedroomAbvGr	TotRmsAbvGrd	Street_Pave	LandSlope_Sev	Condition2_PosN	RoofStyle_Shed	RoofMatl_M
1108	0.500	0.170306	0.460583	0.407819	0.500000	0.444444	1	0	0	0	
745	1.000	0.252911	0.955928	0.753286	0.666667	0.888889	1	0	0	0	
1134	0.500	0.158661	0.424581	0.377486	0.500000	0.444444	1	0	0	0	
512	0.500	0.139738	0.000000	0.129424	0.500000	0.222222	1	0	0	0	
43	0.625	0.166667	0.000000	0.154365	0.500000	0.222222	1	0	0	0	

In [90]:
```
X_test_2.head()
```

Out[90]:

	OverallCond	1stFlrSF	2ndFlrSF	GrLivArea	BedroomAbvGr	TotRmsAbvGrd	Street_Pave	LandSlope_Sev	Condition2_PosN	RoofStyle_Shed	RoofMatl_M
990	0.50	0.337336	0.611421	0.644422	0.5	0.444444	1	0	0	0	
1161	0.75	0.422125	0.000000	0.390967	0.5	0.444444	1	0	0	0	
1369	0.50	0.432314	0.000000	0.400404	0.5	0.555556	1	0	0	0	
329	0.50	0.042213	0.369957	0.239973	0.5	0.333333	1	0	0	0	
262	0.75	0.266376	0.000000	0.246714	0.5	0.333333	1	0	0	0	

In [91]:
```
# Lasso after removing the top 5 features
alpha =10
lasso_top5 = Lasso(alpha=alpha)
lasso_top5.fit(X_train_2, y_train)
```

Out[91]:
```
Lasso(alpha=10)
```

In [92]:
```
y_pred_train = lasso_top5.predict(X_train_2)
y_pred_test = lasso_top5.predict(X_test_2)

metric6 = []
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)
```

```
0.7988346707068132
0.7588103209258127
1016954777102.8658
600167078819.8167
1138807141.2126157
1364016088.226856
```

- R2 score of train and test data has descreased

In [93]:
```
#important predictor variables after removing the top 5
betas = pd.DataFrame(index=X_train_2.columns)
betas.rows = X_train1.columns
betas['lasso_top5_removed'] = lasso_top5.coef_
pd.set_option('display.max_rows', None)
betas.head(68)
```

Out[93]:

	lasso_top5_removed
OverallCond	7403.774043
1stFlrSF	163379.262938
2ndFlrSF	12227.759048
GrLivArea	186638.919740
BedroomAbvGr	-71218.036474
TotRmsAbvGrd	41610.305613
Street_Pave	101376.262107
LandSlope_Sev	-40205.679947
Condition2_PosN	0.000000
RoofStyle_Shed	53262.728685
RoofMatl_Metal	84219.173436
Exterior1st_Stone	-124162.644239
Exterior2nd_CBlock	-139534.253019
ExterQual_Gd	-77170.982079
ExterQual_TA	-108569.936019
BsmtCond_Po	-122646.594039
KitchenQual_TA	-11135.858324
Functional_Maj2	-48462.215856
SaleType_CWD	-64725.438438
SaleType_Con	52937.625483

-- Five important predictor variables after remove the top 5 are

- 1stFlrSF
- GrLivArea
- Street_Pave
- RoofMatl_Metal
- RoofStyle_Shed

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?

Ans: Robustness is the property that is tested on a training sample and on a similar testing sample, the performance and Accuracy are close for both algorithms (Lasso & Ridge). By the means of regularization, we can control the trade-off between Model complexity and bias which is directly connected to the robustness of the model. Penalizing the coefficients for making the model too complex but just allowing the appropriate amount of complexity controls the robustness of the model. Accuracy and robustness may be at the odds to each other as too much accurate model can be prey to over fitting hence it can be too much accurate on train data but fails when it faces the actual data or vice versa.