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:

Optimal value of alpha for Ridge and Lasso regression models are
Lasso model: 0.01 ***Ridge model***: 100

Changes when Lasso alpha is doubled:

Lasso Alpha = 0.01

```
lm = Lasso(alpha=0.01)
lm.fit(X_train,y_train)

y_train_pred = lm.predict(X_train)

y_test_pred  = lm.predict(X_test)
```

```
print("Train Score:",r2_score(y_true=y_train,y_pred=y_train_pred))
print("Test Score:",r2_score(y_true=y_test,y_pred=y_test_pred) )
```

```
Train Score: 0.9301660708341232
Test Score: 0.9045878725829546
```

Feature	Coef
BsmtFullBath	0.285
OverallCond	0.208
1stFlrSF	0.084
MasVnrArea	0.066
WoodDeckSF	0.066
BsmtFinSF2	0.052
OverallQual	0.049
Neighborhood_OldTown	0.046
Neighborhood_StoneBr	0.046
Foundation_Slab	0.046

[Doubled] Lasso Alpha = 0.02

```
#Lets build the model with double of optimal Lasso alpha i.e 0.02

lm = Lasso(alpha=0.02)
lm.fit(X_train,y_train)

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

print("Train Score:",r2_score(y_true=y_train,y_pred=y_train_pred))
print("Test Score:",r2_score(y_true=y_test,y_pred=y_test_pred) )
```

```
Train Score: 0.9202687167108632
Test Score: 0.9002482043978907
```

Feature	Coef
BsmtFullBath	0.297
OverallCond	0.228
1stFlrSF	0.085
WoodDeckSF	0.065
BsmtFinSF2	0.055
GarageArea	0.055
MasVnrArea	0.051
OverallQual	0.049
Foundation_Slab	0.039
E_FireplaceQu	0.037

Changes when Ridge alpha is doubled:

Ridge Alpha = 100

```
#Lets do it with Ridge Regression
ridge = Ridge(alpha=100)
ridge.fit(X_train,y_train)

y_train_pred = ridge.predict(X_train)
print("Train Score", r2_score(y_train,y_train_pred))
y_test_pred = ridge.predict(X_test)
print("Test Score", r2_score(y_test,y_test_pred))
```

```
Train Score 0.9384055987302959
Test Score 0.8955182282347336
```

Feature	Coef
OverallCond	0.146390
BsmtFullBath	0.120964
2ndFlrSF	0.083719
MasVnrArea	0.078174
1stFlrSF	0.074438
LowQualFinSF	0.071996
WoodDeckSF	0.064623
Neighborhood_OldTown	0.055463
BsmtFinSF2	0.054168
MSZoning_RM	0.053032

[Doubled] Ridge Alpha = 200

```
#Lets do Ridge with doubled regression
ridge = Ridge(alpha=200)
ridge.fit(X_train,y_train)

y_train_pred = ridge.predict(X_train)
print("Train Score", r2_score(y_train,y_train_pred))
y_test_pred = ridge.predict(X_test)
print("Test Score", r2_score(y_test,y_test_pred))
```

```
Train Score 0.9348836978381431
Test Score 0.8944959358890112
```

Feature	Coef
OverallCond	0.127924
BsmtFullBath	0.109464
2ndFlrSF	0.079007
1stFlrSF	0.068961
MasVnrArea	0.067613
LowQualFinSF	0.062239
WoodDeckSF	0.061999
Fireplaces	0.053759
BsmtFinSF2	0.051454
Neighborhood_OldTown	0.049844

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:

- The R-Squared value and other metrics are better with Lasso in our assignment so we choose Lasso over Ridge here.
- Also, Lasso removes the overhead of doing RFE as it automatically removes the insignificant variables by making their coefficients 0.

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:

<u>The initial lasso optimal model with top 5 features</u>:

```
lm = Lasso(alpha=0.01)
lm.fit(X_train,y_train)

y_train_pred = lm.predict(X_train)

y_test_pred  = lm.predict(X_test)
print("Train Score:",r2_score(y_true=y_train,y_pred=y_train_pred))
print("Test Score:",r2_score(y_true=y_test,y_pred=y_test_pred) )
```

```
Train Score: 0.9301660708341232
Test Score: 0.9045878725829546
```

```
model_parameter = list(lm.coef_)
model_parameter.insert(0,lm.intercept_)
model_parameter = [round(x,3) for x in model_parameter]
col = df_train.columns
col.insert(0,'Constant')
lasso_coef = pd.DataFrame(list(zip(col,model_parameter)))
lasso_coef.columns = ['Feature','Coef']
# Significant variables which predict price of house when used lass
lasso_coef.sort_values(by='Coef',ascending=False).head()
```

	Feature	Coef
14	BsmtFullBath	0.285
4	OverallCond	0.208
10	1stFlrSF	0.084
5	MasVnrArea	0.066
25	WoodDeckSF	0.066

<u>Now after removing those 5 features:</u>

```
#Removing the above top 5 features from the dataset
df_train=df_train.drop(['BsmtFullBath','OverallCond','1stFlrSF','MasVnrArea','WoodDeckSF'],axis='colu
df_test=df_test.drop(['BsmtFullBath','OverallCond','1stFlrSF','MasVnrArea','WoodDeckSF'],axis='column
```

```
#Our new data set
X_train2 = df_train
X_test2 = df_test
```

```
lm = Lasso(alpha=0.01)
lm.fit(X_train2,y_train)

y_train_pred = lm.predict(X_train2)

y_test_pred  = lm.predict(X_test2)
print("Train Score:",r2_score(y_true=y_train,y_pred=y_train_pred))
print("Test Score:",r2_score(y_true=y_test,y_pred=y_test_pred) )
```

```
Train Score: 0.9255481464734157
Test Score: 0.8975628279003851
```

```
model_parameter = list(lm.coef_)
model_parameter.insert(0,lm.intercept_)
model_parameter = [round(x,3) for x in model_parameter]
col = df_train.columns
col.insert(0,'Constant')
lasso_coef = pd.DataFrame(list(zip(col,model_parameter)))
lasso_coef.columns = ['Feature','Coef']
# Significant variables which predict price of house when used lasso
lasso_coef.sort_values(by='Coef',ascending=False).head()
```

	Feature	Coef
11	BsmtHalfBath	0.301
4	BsmtFinSF1	0.211
8	2ndFlrSF	0.083
21	OpenPorchSF	0.074
5	BsmtFinSF2	0.070

Question 4: Continued on next page

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:

Any model must be generalized so that the resultant model must perform equally well on test data. Also, the model should be working well on all other datasets. These are few metrics to ensure that:

- Model must be kept as simple as possible, complex models are weaker in terms of generalisation.
- Bias-Variance trade-off graph needs be to understood which suggests optimum trade-off for greater generalization
- Model must be scaled and transformed accordingly.
- Model with good R2-Square and less features is preferable than a model with Best R2-squared with too many features. Its more generalisable