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:

According to the model, the optimal value for **Ridge is 100** and **Lasso is 0.01.**

I can see that as the value of alpha increases, the model complexity reduces. Higher values of alpha reduce overfitting, significantly high values can cause under fitting Therefore, and alpha value should be selected wisely.

Original Value of Ridge:-

Ridge

```
[91]: ridge_alpha = model_ridge_cv.best_params_['alpha']
      ridge_alpha

[91]: 100

[92]: ridge = Ridge(alpha=ridge_alpha)
      ridge.fit(X_train, y_train)

      # predict
      y_train_pred = ridge.predict(X_train)
      print(metrics.r2_score(y_true=y_train, y_pred=y_train_pred))
      y_test_pred_ridge = ridge.predict(X_test)
      print(metrics.r2_score(y_true=y_test, y_pred=y_test_pred_ridge))

      0.9037758130080379
      0.8017298407351707
```

After doubling the Ridge:-

```
# predict
y_train_pred_ridge = rid.predict(X_train)
print(metrics.r2_score(y_true=y_train, y_pred=y_train_pred_ridge))
y_test_pred_ridge = rid.predict(X_test)
print(metrics.r2_score(y_true=y_test, y_pred=y_test_pred_ridge))
```

```
200
0.8993682383109244
0.8164926766302475
```

Observations when we double the value of Ridge:-

- As we increase the value of alpha, model complexity decreases.
- Slightly the Train accuracy decease. And slightly test accuracy increases.

Original Value of Lasso:-

```
lasso_alpha = model_lasso_cv.best_params_['alpha']
lasso_alpha
```

```
0.01
```

```
lasso = Lasso(alpha=lasso_alpha)
lasso.fit(X_train, y_train)

# predict
y_train_pred = lasso.predict(X_train)
print(metrics.r2_score(y_true=y_train, y_pred=y_train_pred))
y_test_pred_lasso = lasso.predict(X_test)
print(metrics.r2_score(y_true=y_test, y_pred=y_test_pred_lasso))
```

```
0.9037567612397784
0.795785099673914
```

After doubling the Lasso Values

```
lasso_alpha_2 = lasso_alpha *2

print(lasso_alpha_2)

ls = Lasso(alpha=lasso_alpha_2)
ls.fit(X_train, y_train)

# predict
y_train_pred_ls = ls.predict(X_train)
print(metrics.r2_score(y_true=y_train, y_pred=y_train_pred_ls))
y_test_pred_lasso = ls.predict(X_test)
print(metrics.r2_score(y_true=y_test, y_pred=y_test_pred_lasso))
```

```
0.02
0.8983494355613376
0.8115867947914391
```

Observations when we double the value of Lasso:-

- We observe that accuracy of the model is slightly decreasing for train dataset and slightly increasing for test dataset.
- And also for lasso the coefficients are tending to zero (decreases) as we increase the Value of Alpha.

Top Features after changes along with the original values of Ridge and Lasso

In case of Ridge 5 few Features

```
#Lets find the top 5 features selected as part of model
top5_ridge_features_q1 = [x[0] for x in sorted(list(zip(cols, model_parameters)),key=lambda x: ab
print(top5_ridge_features)
print(top5_ridge_features_q1)
```

```
['LowQualFinSF', 'OverallQual', '2ndFlrSF', 'BedroomAbvGr', 'FullBath', 'TotalBsmtSF']
['LowQualFinSF', 'OverallQual', '2ndFlrSF', 'BedroomAbvGr', 'TotalBsmtSF', 'FullBath']
```

Observations:-

- Features ordering has changed for TotalBsmtSF
- Also the coefficient values decrease.

In Case of Lasso 5 Few Features

```
#Lets find the top 5 features selected as part of model
top5_lasso_features_q1 = [x[0] for x in sorted(list(zip(cols, model_parameters)),key=lambda x: abs(x[1
print(top5_lasso_features)
print(top5_lasso_features_q1)
```

```
['LowQualFinSF', 'OverallQual', 'BedroomAbvGr', 'FullBath', 'TotalBsmtSF', 'EnclosedPorch']
['LowQualFinSF', 'OverallQual', 'BedroomAbvGr', 'FullBath', 'TotalBsmtSF', 'EnclosedPorch']
```

Observations:- 1. The coefficient values decrease.
2. Top 5 features are same.

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:- I will choose Lasso Regression, as the lasso method is better than Ridge. Since in Lasso method it gives the coefficient as zero for the unimportant variables, Plus it penalize more on the high coefficient variables. Therefore, Lasso method gives the more simpler and robust model.

Question3:

After building the model, you realized 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:

Top five important predictor using Lasso regression from the model is as follows:

```
# finding the top 5 features selected as part of model
top5_lasso_features = [x[0] for x in sorted(list(zip(cols, model_parameters)),key=lambda x: abs(x[1]),reverse=True
top5_lasso_features
```

```
['LowQualFinSF',
 'OverallQual',
 'BedroomAbvGr',
 'FullBath',
 'TotalBsmtSF',
 'EnclosedPorch']
```

Top 5 :
['LowQualFinSF','OverallQual', 'BedroomAbvGr', 'FullBath', 'TotalBsmtSF', 'EnclosedPorch']

Removing these columns from the dataset, I performed the same LASSO regression steps again which steps are present in the Jupyter notebook attached.
we can see that new Top 5 new predictor we get are as follows :

```
top5_lasso_q3 = [x[0] for x in sorted(list(zip(cols_q3, model_parame
top5_lasso_q3
```

```
['1stFlrSF',
 'YearBuilt_Old',
 'GrLivArea',
 'BsmtExposure_4',
 '2ndFlrSF',
 'Condition2_PosN']
```

After Removing : top 5
['1stFlrSF','YearBuilt_Old', 'GrLivArea', 'BsmtExposure_4', '2ndFlrSF', 'Condition2_PosN']

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 is the property which stats that the model accuracy is constant when the model runs on any unseen data set.

For a model to be robust and generalizable, it needs to be perform well on both training data set and test data set. To get this we need to make the model simpler as much as possible.

By the using of regularization we can control the tradeoff 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 leads to over fitting hence it can be too much accurate on train data but fails when it run on the actual data or vice versa.