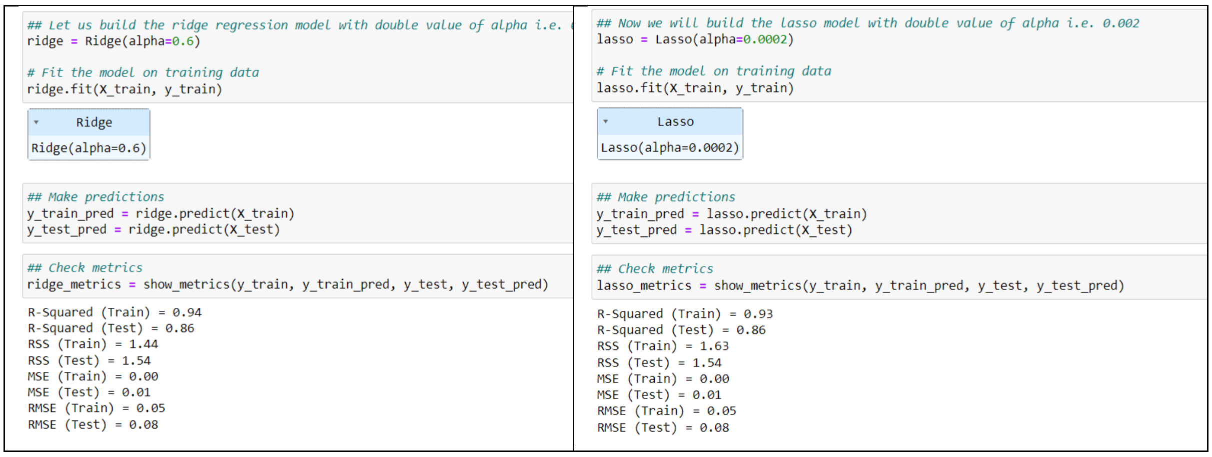
**Question 1. What is the optimal value of alpha for ridge and lasso regression? What will be the changes in the model if you choose to double the value of alpha for both ridge and lasso? What will be the most important predictor variables after the change is implemented?**

**Answer:** Optimal value of alpha for Ridge was 0.3 and Lasso was 0.0001. When we choose double value of alpha to 0.6 for Ridge and 0.0002, We get below changes:



```
## Let us build the ridge regression model with double value of alpha i.e.
ridge = Ridge(alpha=0.6)

# Fit the model on training data
ridge.fit(X_train, y_train)
```

```
## Make predictions
y_train_pred = ridge.predict(X_train)
y_test_pred = ridge.predict(X_test)
```

```
## Check metrics
ridge_metrics = show_metrics(y_train, y_train_pred, y_test, y_test_pred)
```

```
R-Squared (Train) = 0.94
R-Squared (Test) = 0.86
RSS (Train) = 1.44
RSS (Test) = 1.54
MSE (Train) = 0.00
MSE (Test) = 0.01
RMSE (Train) = 0.05
RMSE (Test) = 0.08
```

```
## Now we will build the lasso model with double value of alpha i.e. 0.002
lasso = Lasso(alpha=0.0002)

# Fit the model on training data
lasso.fit(X_train, y_train)
```

```
## Make predictions
y_train_pred = lasso.predict(X_train)
y_test_pred = lasso.predict(X_test)
```

```
## Check metrics
lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_test_pred)
```

```
R-Squared (Train) = 0.93
R-Squared (Test) = 0.86
RSS (Train) = 1.63
RSS (Test) = 1.54
MSE (Train) = 0.00
MSE (Test) = 0.01
RMSE (Train) = 0.05
RMSE (Test) = 0.08
```

Change in Ridge Regression

- R Squred on train data change from 0.93 to 0.94
- R Squred on test data change from 0.85 to 0.86

Change in Lasso Regression

- No change in R Squred on train data, current value is 0.93
- No change in R Squred on test data, current value is 0.86

The most important predictor variables after we double the alpha values are:

- GrLivArea
- OverallQual
- TotalBsmtSF
- YrBltAndRemod
- Total_sqr_footage
- Neighborhood_Crawfor
- Neighborhood_BrkSide
- Neighborhood_Somerst
- LotArea
- GarageArea
- MSZoning_RL
- 2ndFlrSF

**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 decision regarding the application of either Ridge or Lasso regression hinges on the specific requirements of the use case. If the scenario involves a plethora of variables and prioritizes feature selection, Lasso regression would be the preferred choice. On the other hand, if the aim is to mitigate the impact of excessively large coefficients and emphasize the reduction of coefficient magnitudes, Ridge Regression will be the selected model.

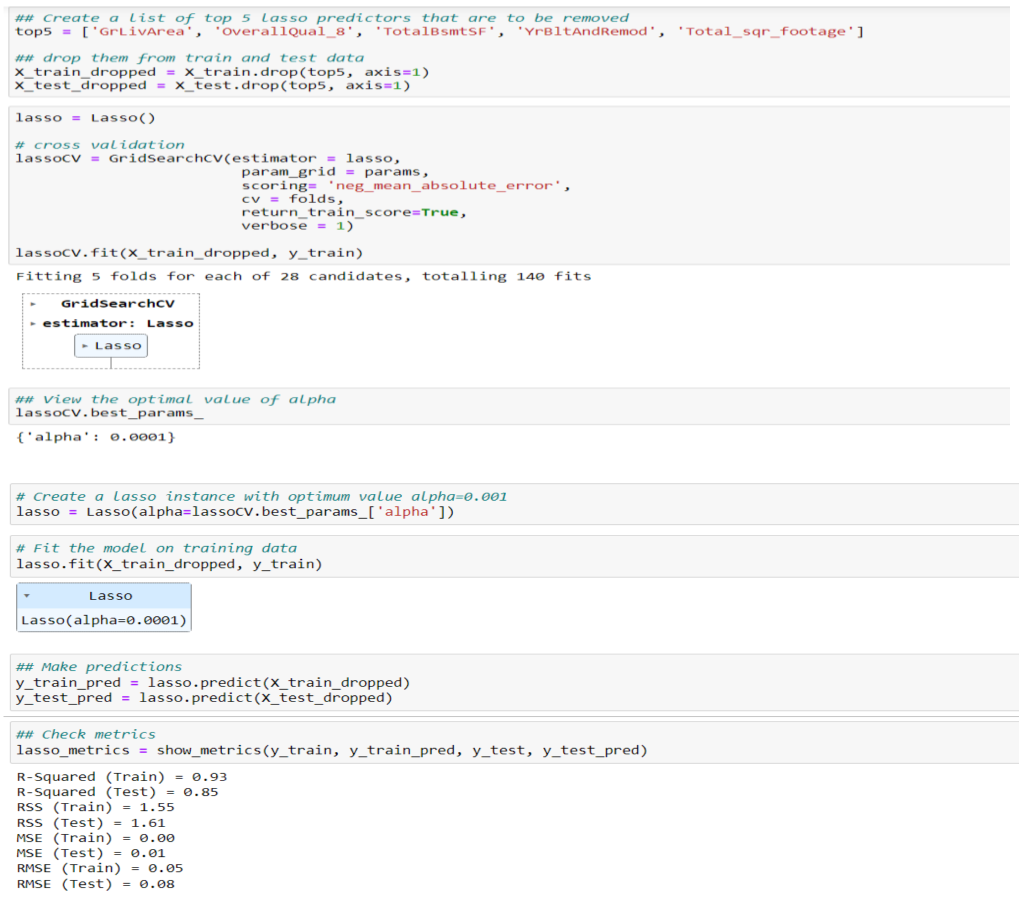
The optimum lambda value in case of Ridge and Lasso is as follows: -

- Ridge: 0.3
- Lasso: 0.0001

Since Lasso helps in feature reduction (as the coefficient value of some of the features become zero), Lasso has a better edge over Ridge and should be used as the final model.

**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:** To achieve this will drop the top 5 features in Lasso model and build the model again. Top 5 Lasso predictors were: *GrLivArea*, *OverallQual_8*, *TotalBsmtSF*, *YrBltAndRemod* and *Total_sqr_footage*



```
## Create a list of top 5 lasso predictors that are to be removed
top5 = ['GrLivArea', 'OverallQual_8', 'TotalBsmtSF', 'YrBltAndRemod', 'Total_sqr_footage']

## drop them from train and test data
X_train_dropped = X_train.drop(top5, axis=1)
X_test_dropped = X_test.drop(top5, axis=1)
```

```
lasso = Lasso()

# cross validation
lassoCV = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)

lassoCV.fit(X_train_dropped, y_train)
```

```
Fitting 5 folds for each of 28 candidates, totalling 140 fits
```

```
▸ GridSearchCV
▸ estimator: Lasso
  ▸ Lasso
```

```
## View the optimal value of alpha
lassoCV.best_params_
```

```
{'alpha': 0.0001}
```

```
# Create a lasso instance with optimum value alpha=0.001
lasso = Lasso(alpha=lassoCV.best_params_['alpha'])
```

```
# Fit the model on training data
lasso.fit(X_train_dropped, y_train)
```

```
▾ Lasso
Lasso(alpha=0.0001)
```

```
## Make predictions
y_train_pred = lasso.predict(X_train_dropped)
y_test_pred = lasso.predict(X_test_dropped)
```

```
## Check metrics
lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_test_pred)
```

```
R-Squared (Train) = 0.93
R-Squared (Test) = 0.85
RSS (Train) = 1.55
RSS (Test) = 1.61
MSE (Train) = 0.00
MSE (Test) = 0.01
RMSE (Train) = 0.05
RMSE (Test) = 0.08
```

After dropping our top 5 lasso predictors, we get the following new top 5 predictors as follows:

- 1stFlrSF
- 2ndFlrSF
- BsmtFinSF1
- LotArea
- Neighborhood_Crawfor

**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:** Ensuring the robustness and generalizability of a model is crucial for its overall effectiveness. According to Occam's Razor, when faced with two models exhibiting similar performance on finite training or test data, the preference should be for the one making fewer assumptions. This preference is grounded in several reasons:

- Simpler models tend to be more generic and widely applicable.
- They require fewer training samples for effective training, making them easier to train.
- Simpler models exhibit greater robustness, unlike complex models that can fluctuate significantly with changes in the training data set.

The trade-off between bias and variance further emphasizes the importance of model simplicity. Complex models, while capable of accurate predictions with sufficient training data, become unstable and overly sensitive to dataset changes. On the other hand, simpler models, by abstracting patterns from the data, remain more stable even with additions or removals of data points.

Regularization serves as a valuable tool in achieving the delicate balance between model simplicity and utility. By adding a regularization term to the cost function, one can control the complexity of the model. This regularization involves incorporating the absolute values or squares of the model parameters for regression tasks.

In essence, making a model simple contributes to the bias-variance trade-off. While a complex model may accurately predict outcomes with abundant training data, it becomes highly unstable. Conversely, a simpler model, abstracting key patterns, maintains stability and is less likely to change drastically with variations in the training data.

Therefore, to enhance a model's robustness and generalizability, prioritizing simplicity without oversimplification is paramount.