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?

- Optimal value of lambda for Ridge Regression = **9.0**
- Optimal value of lambda for Lasso = **0.0001**

Let us double the optimal lambda for ridge regression and see the results

- Optimal value of lambda for Ridge Regression = **18**
- Optimal value of lambda for Lasso = **0.0002**

```python
## Define a function for checking metrics
def show_metrics(y_train, y_train_pred, y_test, y_pred):
    '''
    Takes in the values of true y_train and y_test, and predicted y_train and y_test.
    Prints out
    1. R-Squared (Train)
    2. R-Squared (Test)
    3. RSS (Train)
    4. RSS (Test)
    5. MSE (Train)
    6. MSE (Test)
    7. RMSE (Train)
    8. RMSE (Test)

    Returns a list containing all the above 8 metrics

    '''

    ## Create a list to save all metrics (will be used in creating a final summary in the end)
    metric = []

    ## R-squared of train and test data
    print("R-Squared (Train) =", '%.2f' % r2_score(y_train, y_train_pred))
    metric.append(r2_score(y_train, y_train_pred))
    print("R-Squared (Test) =", '%.2f' % r2_score(y_test, y_pred))
    metric.append(r2_score(y_test, y_pred))

    ## Residual sum of squares of train and test data
    rss_train = np.sum(np.square(y_train - y_train_pred))
    metric.append(rss_train)
    rss_test = np.sum(np.square(y_test - y_pred))
    metric.append(rss_test)
    print("RSS (Train) =", '%.2f' % rss_train)
    print("RSS (Test) =", '%.2f' % rss_test)


    ## Mean Squared Error of train and test data
    mse_train = mean_squared_error(y_train, y_train_pred)
    metric.append(mse_train)
    mse_test = mean_squared_error(y_test, y_pred)
    metric.append(mse_test)
    print("MSE (Train) =", '%.2f' % mse_train)
    print("MSE (Test) =", '%.2f' % mse_test)



    # Root Mean Squared Error for train and test data
    rmse_train = mse_train**0.5
    metric.append(rmse_train)
    rmse_test = mse_test**0.5
    metric.append(rmse_test)
    print("RMSE (Train) =", '%.2f' % rmse_train)
    print("RMSE (Test) =", '%.2f' % rmse_test)


    return metric
```

```
## Let us build the ridge regression model with double value of alpha i.e. 20
ridge = Ridge(alpha=18)

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

```
Ridge(alpha=18)
```

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

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

```
R-Squared (Train) = 0.92
R-Squared (Test) = 0.90
RSS (Train) = 10.96
RSS (Test) = 5.58
MSE (Train) = 0.01
MSE (Test) = 0.01
RMSE (Train) = 0.11
RMSE (Test) = 0.12
```

Let us check for the lasso regression

In [129]:
```
## 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)
```

Out[129]:
```
Lasso(alpha=0.0002)
```

In [130]:
```
## Make predictions
y_train_pred = lasso.predict(X_train)
y_pred = lasso.predict(X_test)
```

In [131]:
```
## Check metrics
lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)
```

```
R-Squared (Train) = 0.92
R-Squared (Test) = 0.90
RSS (Train) = 10.77
RSS (Test) = 5.55
MSE (Train) = 0.01
MSE (Test) = 0.01
RMSE (Train) = 0.11
RMSE (Test) = 0.12
```

Let us summarize the results we got

In [132]:
```
# Again creating a table which contain all the metrics

lr_table = {'Metric': ['R2 Score (Train)','R2 Score (Test)','RSS (Train)','RSS (Test)',
                       'MSE (Train)','MSE (Test)', 'RMSE (Train)', 'RMSE (Test)'],
            'Ridge Regression' : ridge_metrics,
            'Lasso Regression' : lasso_metrics
        }

final_metric = pd.DataFrame(lr_table, columns = ['Metric', 'Ridge Regression', 'Lasso Regression'] )
final_metric.set_index('Metric')
```

Out[132]:

Metric	Ridge Regression	Lasso Regression
R2 Score (Train)	0.917965	0.919395
R2 Score (Test)	0.901618	0.902225
RSS (Train)	10.964816	10.773763
RSS (Test)	5.584370	5.549935
MSE (Train)	0.011542	0.011341
MSE (Test)	0.013687	0.013603
RMSE (Train)	0.107433	0.106493
RMSE (Test)	0.116992	0.116631

Ridge regression

- The r2 score of train set has decreased from 91.8 to 91.7 when we double the alpha for ridge regression.
- The r2 test score remained at 90.1

Lasso regression

- The r2 score of train set has reduced from 91.95 to 91.93
- The r2 test score has increased from 90.19 to 90.22

The most common predictor variables after doubling the alpha values are

- GrLivArea
- OverallQual_8
- OverallQual_9
- Functional_Typ

- Neighborhood_Crawfor
- Exterior1st_BrkFace
- TotalBsmtSF
- CentralAir_Y

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?

The model we select and work with depends on the use case. When we have too many variables, our only goal is feature selection and it is here that Lasso regression comes in handy. If we don't want to have too large coefficients and reduction of coefficient magnitude is our primary goal, then we can go with the ridge regression 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?

After removing the top 5 most important predictor variables in the lasso model like

- MSZoning_RL

- GrLivArea

- MSZoning_RM

- MSZoning_FV

- OverallQual

The optimal alpha value is 0.001

After dropping our top 5 features, the predictor variables we get are

- TotalBsmtSF
- FullBath
- HalfBath
- GarageCars
- OverallCond

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?

A model becomes robust when variations in data do not affect its performance to a greater extent. A generalizable model can easily adapt to a new or unseen data. To make sure that our models are robust and generalizable, we should check if they overfit. If models overfit, it will have high variance and a small change in data can heavily impact the performance and results of the analysis. The overfit models fail to pick up the patterns in the test data.

Complex models have high accuracy in them. To make the model robust and generalisable, we have to reduce the variance at the cost of bias. With a small rise is bias, and reduction of accuracy score, we can strike the right balance of a model being robust and generalisable. Regularization techniques like Lasso and ridge regression helps us find the balance between model accuracy and complexity.