Problem Statement - Part II

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 Regression = **10**
- Optimal value of alpha for Lasso = **0.001**

Python Code for if double the value of alpha for both ridge and Lasso:

Assignment Part 2:Coding for Question and Answer

Q1:What will be the changes in the model if you choose double the value of alpha for both ridge and lasso

In [167]:
```
## Let us build the ridge regression model with double value of alpha i.e. 20
ridge = Ridge(alpha=20)

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

Out[167]:
```
Ridge(alpha=20)
```

In [168]:
```
## Make predictions
y_train_pred = ridge.predict(X_train)
y_pred = ridge.predict(X_test)
```

In [169]:
```
## Check metrics
ridge_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)
```

```
R-Squared (Train) = 0.93
R-Squared (Test) = 0.93
RSS (Train) = 9.37
RSS (Test) = 2.82
MSE (Train) = 0.01
MSE (Test) = 0.01
RMSE (Train) = 0.09
RMSE (Test) = 0.10
```

In [170]:
```
## Now we will build the lasso model with double value of alpha i.e. 0.002
lasso = Lasso(alpha=0.002)

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

Out[170]:
```
Lasso(alpha=0.002)
```

In [171]:
```
## Make predictions
y_train_pred = lasso.predict(X_train)
y_pred = lasso.predict(X_test)
```

In [172]:
```
## Check metrics
lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)
```

```
R-Squared (Train) = 0.91
R-Squared (Test) = 0.91
RSS (Train) = 13.49
RSS (Test) = 3.45
MSE (Train) = 0.01
MSE (Test) = 0.01
RMSE (Train) = 0.11
RMSE (Test) = 0.11
```

In [173]:
```
# 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[173]:

Metric	Ridge Regression	Lasso Regression
R2 Score (Train)	0.93	0.91
R2 Score (Test)	0.93	0.91
RSS (Train)	9.37	13.49
RSS (Test)	2.82	3.45
MSE (Train)	0.01	0.01
MSE (Test)	0.01	0.01
RMSE (Train)	0.09	0.11
RMSE (Test)	0.10	0.11

Changes in Ridge Regression metrics:

- R2 score of train set decreased from 0.94 to 0.93
- R2 score of test set remained same at 0.93

Changes in Lasso metrics:

- R2 score of train set decreased from 0.92 to 0.91
- R2 score of test set decreased from 0.93 to 0.91

So we have implemented changes and now we look at the most important predictor variables

Now, we look at the most important predictor variables after the change is implemented.

In [178]:
```
## View the top 10 coefficients of Ridge regression in descending order
betas['Ridge'].sort_values(ascending=False)[:10]
```

Out[178]:
```
GrLivArea              0.08
OverallQual_8          0.07
OverallQual_9          0.06
Neighborhood_Crawfor   0.06
Functional_Typ         0.06
Exterior1st_BrkFace    0.06
OverallCond_9          0.05
TotalBsmtSF            0.05
CentralAir_Y           0.05
OverallCond_7          0.04
Name: Ridge, dtype: float64
```

In [179]:
```
## To interpret the ridge coefficients in terms of target, we have to take inverse log (i.e. e to the power) of betas
ridge_coeffs = np.exp(betas['Ridge'])
ridge_coeffs.sort_values(ascending=False)[:10]
```

Out[179]:
```
GrLivArea              1.08
OverallQual_8          1.07
OverallQual_9          1.07
Neighborhood_Crawfor   1.07
Functional_Typ         1.06
Exterior1st_BrkFace    1.06
OverallCond_9          1.06
TotalBsmtSF            1.05
CentralAir_Y           1.05
OverallCond_7          1.04
Name: Ridge, dtype: float64
```

```
In [180]: ## View the top 10 coefficients of Lasso in descending order
          betas['Lasso'].sort_values(ascending=False)[:10]

Out[180]: GrLivArea              0.11
          OverallQual_8          0.08
          OverallQual_9          0.08
          Functional_Typ         0.07
          Neighborhood_Crawfor   0.07
          TotalBsmtSF            0.05
          Exterior1st_BrkFace    0.04
          CentralAir_Y           0.04
          YearRemodAdd           0.04
          Condition1_Norm        0.03
          Name: Lasso, dtype: float64

In [181]: ## To interpret the lasso coefficients in terms of target, we have to take inverse log (i.e. 10 to the power) of betas
          lasso_coeffs = np.exp(betas['Lasso'])
          lasso_coeffs.sort_values(ascending=False)[:10]

Out[181]: GrLivArea              1.11
          OverallQual_8          1.09
          OverallQual_9          1.08
          Functional_Typ         1.07
          Neighborhood_Crawfor   1.07
          TotalBsmtSF            1.05
          Exterior1st_BrkFace    1.05
          CentralAir_Y           1.04
          YearRemodAdd           1.04
          Condition1_Norm        1.03
          Name: Lasso, dtype: float64
```

So, the most important predictor variables after we double the alpha values are:-

1. OverallQual_8
2. OverallQual_9
3. GrLivArea
4. Functional_Typ
5. Neighborhood_Crawfor
6. Exterior1st_BrkFace
7. TotalBsmtSF
8. 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?

Answer:

- The model we will choose to apply will depend on the use case.
- If we have too many variables and one of our primary goals is feature selection, then we will use **Lasso**.
- If we don't want to get too large coefficients and reduction of coefficient magnitude is one of our prime goals, then we will use **Ridge Regression**.

Question 3

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:

Q3 Excluding the five most important predictor variables in Lasso model and build the model again.

Top 5 Lasso predictors were:

`OverallQual_9`, `GrLivArea`, `OverallQual_8`, `Neighborhood_Crawfor` and `Exterior1st_BrkFace`

In [99]:
```
# Create a list of top 5 lasso predictors that are to be removed
top5 = ['OverallQual_9', 'GrLivArea', 'OverallQual_8', 'Neighborhood_Crawfor', 'Exterior1st_BrkFace']
```

In [100]:
```
## 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)
```

In [101]:
```
## Now to create a Lasso model
## we will run a cross validation on a list of alphas to find the optimum value of alpha

params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,
                    2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000]}

lasso = Lasso()

# cross validation

lassoCV = GridSearchCV(estimator = lasso,
                       param_grid = params,
                       scoring= 'neg_mean_absolute_error',
                       cv = 5,
                       return_train_score=True,
                       verbose = 1, n_jobs=-1)
lassoCV.fit(X_train_dropped, y_train)
```

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

Out[101]:
```
GridSearchCV(cv=5, estimator=Lasso(), n_jobs=-1,
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

In [102]:
```
## View the optimal value of alpha
lassoCV.best_params_
```

Out[102]:
```
{'alpha': 0.001}
```

Thus, we get optimum value of alpha as 0.001. Now we will build a lasso regression model using this value.

In [103]:
```
# Create a lasso instance with optimum value alpha=0.001
lasso = Lasso(alpha=0.001)
```

In [104]:
```
# Fit the model on training data
lasso.fit(X_train_dropped, y_train)
```

Out[104]:
```
Lasso(alpha=0.001)
```

In [105]:
```
## Make predictions
y_train_pred = lasso.predict(X_train_dropped)
y_pred = lasso.predict(X_test_dropped)
```

In [106]:
```
## Check metrics
lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)
```

```
R-Squared (Train) = 0.91
R-Squared (Test) = 0.92
RSS (Train) = 12.75
RSS (Test) = 3.02
MSE (Train) = 0.01
MSE (Test) = 0.01
RMSE (Train) = 0.10
RMSE (Test) = 0.10
```

Now, we will find the top 5 predictors

In [107]:
```
# 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)'],
            'Lasso Regression' : lasso_metrics
        }

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

Out[107]:

Metric	Lasso Regression
R2 Score (Train)	0.91
R2 Score (Test)	0.92
RSS (Train)	12.75
RSS (Test)	3.02
MSE (Train)	0.01
MSE (Test)	0.01
RMSE (Train)	0.10
RMSE (Test)	0.10

In [108]:
```
## Now we see the changes in coefficients after regularization

## First create empty dataframe with all the independent variables as indices
betas = pd.DataFrame(index=X_train_dropped.columns)
betas.rows = X_train_dropped.columns
betas
```

Out[108]:

LotFrontage
LotArea
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
BsmtFullBath

In [109]:
```
## Now fill in the values of betas, one column for ridge coefficients and one for lasso coefficients
betas['Lasso'] = lasso.coef_
# betas['Lasso'] = lasso.coef_
```

In [110]:
```
## View the betas/coefficients
betas
```

Out[110]:

	Lasso
LotFrontage	0.00
LotArea	0.02
YearRemodAdd	0.03
MasVnrArea	-0.00
BsmtFinSF1	0.03
BsmtFinSF2	0.00
BsmtUnfSF	-0.00
TotalBsmtSF	0.05
1stFlrSF	0.07
2ndFlrSF	0.10
LowQualFinSF	0.00

Now, we will look at the top 5 features significant in predicting the value of a house according to the new lasso model

In [111]:
```
## View the top 5 coefficients of Lasso in descending order
betas['Lasso'].sort_values(ascending=False)[:5]
```

Out[111]:
```
2ndFlrSF               0.10
Functional_Typ         0.07
1stFlrSF               0.07
MSSubClass_70          0.06
Neighborhood_Somerst   0.06
Name: Lasso, dtype: float64
```

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

1. 2ndFlrSF
2. Functional_Typ
3. 1stFlrSF
4. MSSubClass_70
5. Neighborhood_Somerst

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:

- A model is **robust** when any variation in the data does not affect its performance much.
- A **generalizable** model can adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model.
- To make sure a model is robust and generalizable, we must **take care it doesn't overfit**. This is because an overfitting model has very high variance and a smallest change in data affects the model prediction heavily. Such a model will identify all the patterns of a training data but fail to pick up the patterns in unseen test data.
- In other words, the model should not be too complex to be robust and generalizable.

- If we look at it from the perspective of **Accuracy**, a too complex model will have a very high accuracy. So, to make our model more robust and generalizable, we will have to decrease variance which will lead to some bias. Addition of bias means that accuracy will decrease.
- In general, we must find strike some balance between model accuracy and complexity. This can be achieved by Regularization techniques like Ridge Regression and Lasso.
