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 variable after the change is implemented?

Answer:

- Optimal value of lambda for Ridge Regression = 10
- Optimal value of lambda for Lasso = 0.001

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 metric:

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

- GrLivArea
- OverallQual_8
- OverallQual_9
- Functional_Typ
- Neighborhood_Crawfor
- Exteriorlst_BrkFace
- TotalBsmtSF
- CentralAir_Y

In ## 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[80]: Ridge(alpha=20)

In [ ]: ## Make predictions

```
Y_train_pred = ridge.predict(X_train)
Y_pred = ridge.predict(X_test)
```

In[ ]: ## Check metrics

```
ridge_metrics = show_metrics(y_train, y_train_pred,
y_test, y_pred)
```

Out[ ]: R-squared (Train) = 0.93
R-Squared (Test) = 0.93

In[ ]: ## 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[ ]: Lasso(alpha=0.002)

In[ ]: ## Make predictions

```
y_train_pred = lasso.predict(X_train)
y_pred = lasso.predict(X_train)
```

In[ ] ## check metrics

```
lasso_metrics = show_metrics(y_train, y_train_pred,
y_test, y_pred)
```

Out[ ]: R-squared (Train) = 0.91
R-Squared (Test) = 0.91

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 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:

- Here, we will drop the top 5 features in Lasso model and build the model again.

Top 5 Lasso predictors were: OverallQual_9, GrLivArea, OverallQual_8, Neighborhood_Crawfor and Exteriorlst_BrkFace

```
In [ ] ## Create a list of top 5 lasso predictors that are
       to be removed.
       top5 = ['OverallQual_9', 'GrLivArea', 'OverallQual_8',
               'Neighborhood_Crawfor', 'Exterior1st_BrkFace']
In [ ] ## drop them from train and test data
       X_train_dropped = X_train.drop(top5, axis=1)
       X_test_dropped = X_~~train~~test.drop(top5, axis = 1)
In [ ] ## Now to create a Lasso model
       ## we will run a cross validation on a list of alp-
       -has 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)
Out [ ] Fitting 5 folds for each of 28 candidates, totalling 140 fits
   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)
```

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

```
In[ ]: # Create a lasso instance with optimum
       value alpha = 0.001
       lasso = Lasso(alpha = 0.001)
In[ ]: # Fit the model on training data
       lasso.fit(X_train_dropped, y_train)
Out[ ]: Lasso(alpha=0.001)
In[ ]: # Make predictions
       y_train_pred = lasso.predict(X_train_dropped)
       y_pred = lasso.predict(X_test_dropped)
In[101]: ## Check metrics
       lasso_metrics = show_metrics(y_train, y_train_
       pred, y_test, y_pred)
Out[ ]: R-Squared (Train) = 0.91
        R-Squared (Test) = 0.92
```

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

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

- 2ndFlrSF
- Functional_Typ
- 1stFlrSF
- MSSubClass_70
- 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 is able to adapt property 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 have to **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 in order to be robust and generalizable.
- If we look at it from the prespective 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 have to find strike some balance between model accuracy and complexity. This can be achieved by Regularization techniques like Ridge Regression and Lasso.