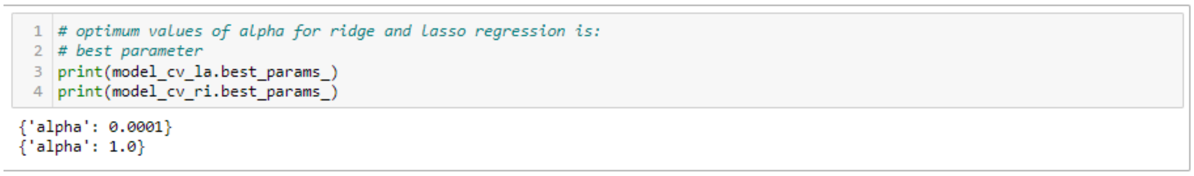
Problem Statement Part II

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 double the value of alpha for both ridge and lasso? What will be the most important predictor variables after the change is implemented?

Answer:

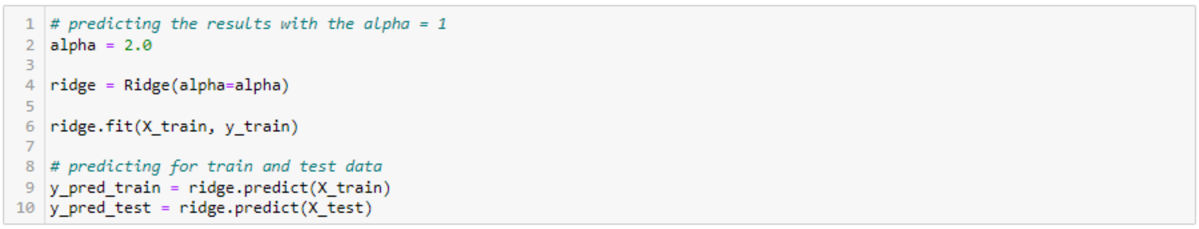
The best alpha values for the ridge and lasso models is as follows:



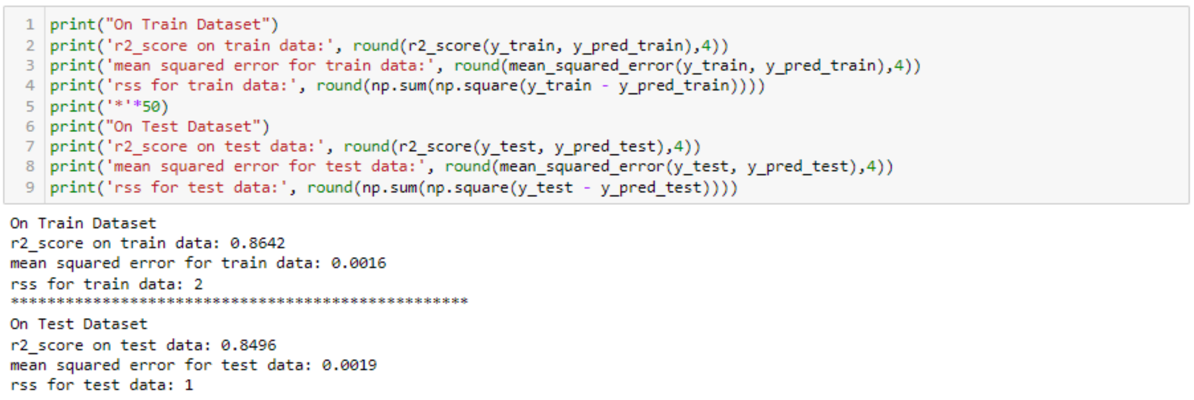
```
# optimum values of alpha for ridge and lasso regression is:
# best parameter
print(model_cv_la.best_params_)
print(model_cv_ri.best_params_)
```

```
{'alpha': 0.0001}
{'alpha': 1.0}
```

Ridge Regression:



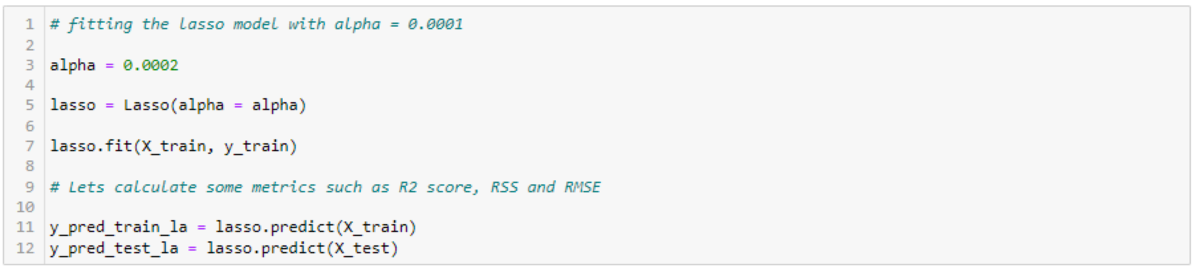
```
# predicting the results with the alpha = 1
alpha = 2.0

ridge = Ridge(alpha=alpha)

ridge.fit(X_train, y_train)

# predicting for train and test data
y_pred_train = ridge.predict(X_train)
y_pred_test = ridge.predict(X_test)
```



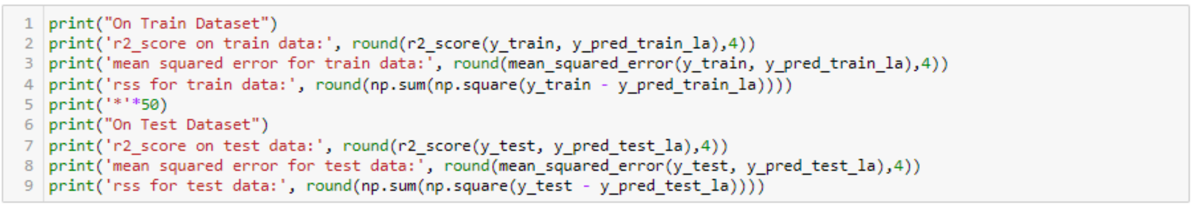
```
print("On Train Dataset")
print('r2_score on train data:', round(r2_score(y_train, y_pred_train),4))
print('mean squared error for train data:', round(mean_squared_error(y_train, y_pred_train),4))
print('rss for train data:', round(np.sum(np.square(y_train - y_pred_train))))
print('*'*50)
print("On Test Dataset")
print('r2_score on test data:', round(r2_score(y_test, y_pred_test),4))
print('mean squared error for test data:', round(mean_squared_error(y_test, y_pred_test),4))
print('rss for test data:', round(np.sum(np.square(y_test - y_pred_test))))
```

```
On Train Dataset
r2_score on train data: 0.8642
mean squared error for train data: 0.0016
rss for train data: 2
**************************************************
On Test Dataset
r2_score on test data: 0.8496
mean squared error for test data: 0.0019
rss for test data: 1
```

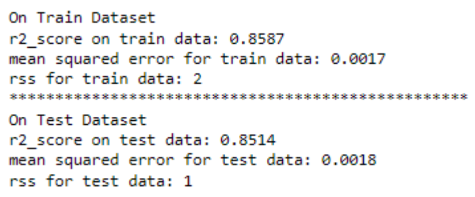
Lasso Regression



```
# fitting the lasso model with alpha = 0.0001

alpha = 0.0002

lasso = Lasso(alpha = alpha)

lasso.fit(X_train, y_train)

# Lets calculate some metrics such as R2 score, RSS and RMSE

y_pred_train_la = lasso.predict(X_train)
y_pred_test_la = lasso.predict(X_test)
```



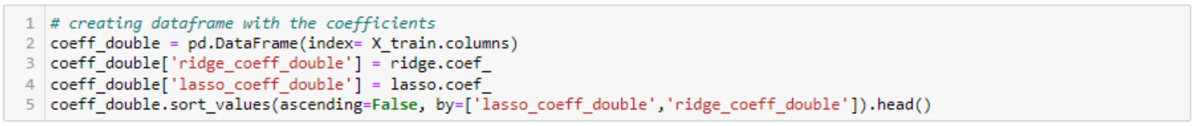
```python
print("On Train Dataset")
print('r2_score on train data:', round(r2_score(y_train, y_pred_train_la),4))
print('mean squared error for train data:', round(mean_squared_error(y_train, y_pred_train_la),4))
print('rss for train data:', round(np.sum(np.square(y_train - y_pred_train_la))))
print('*'*50)
print("On Test Dataset")
print('r2_score on test data:', round(r2_score(y_test, y_pred_test_la),4))
print('mean squared error for test data:', round(mean_squared_error(y_test, y_pred_test_la),4))
print('rss for test data:', round(np.sum(np.square(y_test - y_pred_test_la))))
```



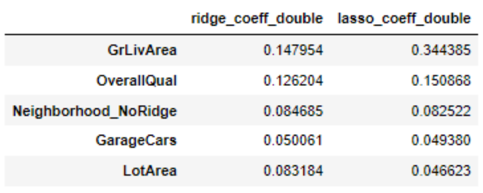
```
On Train Dataset
r2_score on train data: 0.8587
mean squared error for train data: 0.0017
rss for train data: 2
**************************************************
On Test Dataset
r2_score on test data: 0.8514
mean squared error for test data: 0.0018
rss for test data: 1
```

The best value of alpha for ridge regression and lasso regression is obtained as 1.0 and 0.0001. For this case the evaluation score of r2 is obtained as 0.8668 and 0.8522 for train and test dataset of ridge regression and in case of lasso regression r2 score of 0.8634 and 0.8558 is obtained for train and test sets.

If we doubles the values of alpha for both regression to 2.0 and 0.0002, then for r2 score is slightly decreases for the train and test datasets. The top 5 coefficient are given below, as a result of multiplying the alpha value by 2.



```python
# creating dataframe with the coefficients
coeff_double = pd.DataFrame(index= X_train.columns)
coeff_double['ridge_coeff_double'] = ridge.coef_
coeff_double['lasso_coeff_double'] = lasso.coef_
coeff_double.sort_values(ascending=False, by=['lasso_coeff_double','ridge_coeff_double']).head()
```



	ridge_coeff_double	lasso_coeff_double
GrLivArea	0.147954	0.344385
OverallQual	0.126204	0.150868
Neighborhood_NoRidge	0.084685	0.082522
GarageCars	0.050061	0.049380
LotArea	0.083184	0.046623

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:

Ridge and Lasso are the two regulization techiniques, which is used to regularize the coefficient and imporve the predictive power of the developed model. Along with balancing the bias-variance trade off so that the model complixicity is maintained.

Ridge Regression:

In ridge regression techinique, we used lambda function along with the shrinkage panelty as sum of squared model coefficient.

LASSO Regression:

In LASSO regression uses the labda function with the sum of the absolute values of all the coefficient present in the model.

The LASSO regression shrinks the coefficient estimates towards zero (0). in this mothod the panelty term pushes some of the coefficient estimates to be exactly to 0. given that the hyperparameter tuning is large.

while performing LASSO regression, the model performs the feature selection and because of this the interpretation of the model generated by LASSO makes it easier as compared to model generated by Ridge.

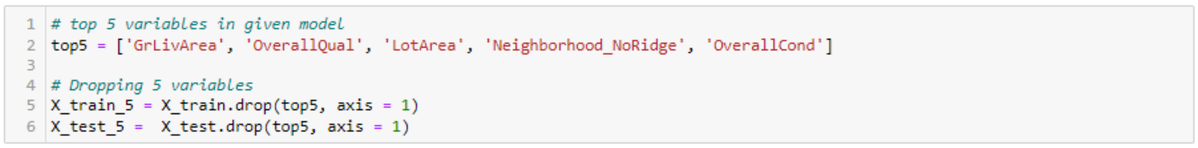
Question 3

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?

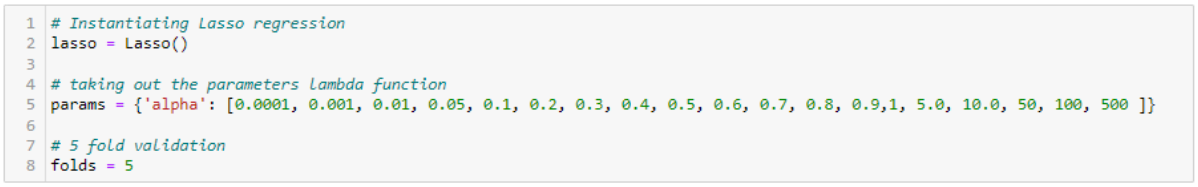
Answer:

In the given model to 5 variables according to the lasso regression is as follows: GrLivArea, OverallQual, LotArea, Neighborhood_NoRidge, OverallCond.

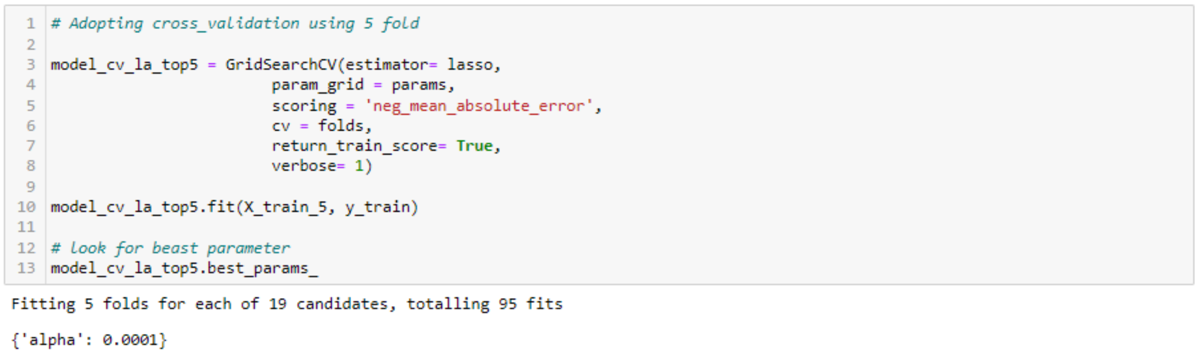
After building the new model by dropping above variables we obtain top 5 variables as: 1stFlrSF, 2ndFlrSF, GarageCars, MasVnrArea, Neighborhood_StoneBr.



```
# top 5 variables in given model
top5 = ['GrLivArea', 'OverallQual', 'LotArea', 'Neighborhood_NoRidge', 'OverallCond']

# Dropping 5 variables
X_train_5 = X_train.drop(top5, axis = 1)
X_test_5 =  X_test.drop(top5, axis = 1)
```



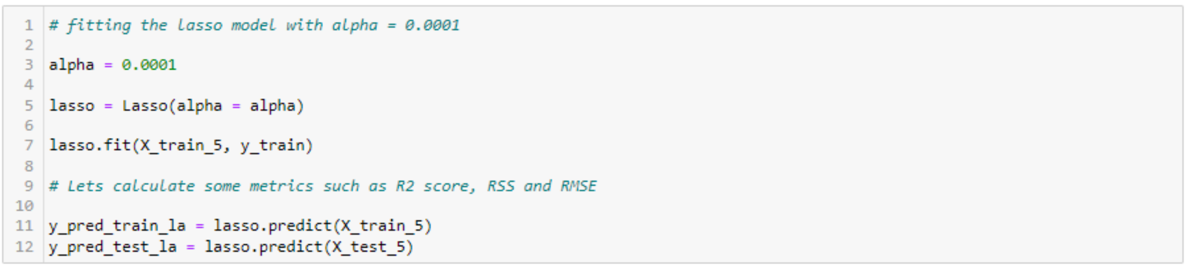
```
# Instantiating Lasso regression
lasso = Lasso()

# taking out the parameters lambda function
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, 5.0, 10.0, 50, 100, 500 ]}

# 5 fold validation
folds = 5
```



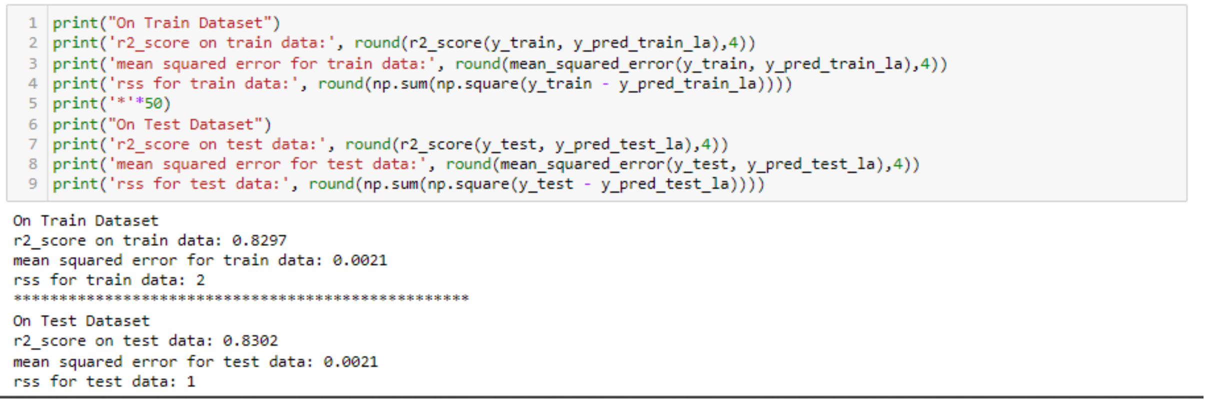
```
# Adopting cross_validation using 5 fold

model_cv_la_top5 = GridSearchCV(estimator= lasso,
                        param_grid = params,
                        scoring = 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score= True,
                        verbose= 1)

model_cv_la_top5.fit(X_train_5, y_train)

# look for beast parameter
model_cv_la_top5.best_params_
```

```
Fitting 5 folds for each of 19 candidates, totalling 95 fits
```

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

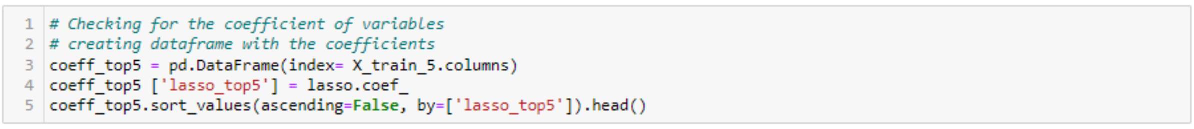


```python
# fitting the lasso model with alpha = 0.0001

alpha = 0.0001

lasso = Lasso(alpha = alpha)

lasso.fit(X_train_5, y_train)

# Lets calculate some metrics such as R2 score, RSS and RMSE

y_pred_train_la = lasso.predict(X_train_5)
y_pred_test_la = lasso.predict(X_test_5)
```

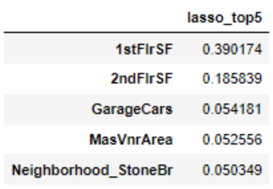


```python
print("On Train Dataset")
print('r2_score on train data:', round(r2_score(y_train, y_pred_train_la),4))
print('mean squared error for train data:', round(mean_squared_error(y_train, y_pred_train_la),4))
print('rss for train data:', round(np.sum(np.square(y_train - y_pred_train_la))))
print('*'*50)
print("On Test Dataset")
print('r2_score on test data:', round(r2_score(y_test, y_pred_test_la),4))
print('mean squared error for test data:', round(mean_squared_error(y_test, y_pred_test_la),4))
print('rss for test data:', round(np.sum(np.square(y_test - y_pred_test_la))))
```

```
On Train Dataset
r2_score on train data: 0.8297
mean squared error for train data: 0.0021
rss for train data: 2
**************************************************
On Test Dataset
r2_score on test data: 0.8302
mean squared error for test data: 0.0021
rss for test data: 1
```



```python
# Checking for the coefficient of variables
# creating dataframe with the coefficients
coeff_top5 = pd.DataFrame(index= X_train_5.columns)
coeff_top5 ['lasso_top5'] = lasso.coef_
coeff_top5.sort_values(ascending=False, by=['lasso_top5']).head()
```



	lasso_top5
1stFlrSF	0.390174
2ndFlrSF	0.185839
GarageCars	0.054181
MasVnrArea	0.052556
Neighborhood_StoneBr	0.050349

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:

The model should be as straightforward as feasible because this will increase its robustness and generalizability while reducing accuracy. The Bias-Variance trade-off may also be used to understand it. The bias increases with model complexity while decreasing variance and increasing generalizability. Its accuracy implication is that a robust and generalizable model would perform similarly on both training and test data, i.e., the accuracy does not change significantly for training and test data.

When a model is unable to learn from the data, bias occurs. High bias prevents the model from learning specifics from the data. Model's performance on training and test data is subpar.

Variance is a model mistake that occurs when the model attempts to overlearn from the data. High variance implies that the model performs very well on training data since it has been extremely well trained on this type of data, but it performs dreadfully on testing data because it was undiscovered data for the model.

To prevent data from being over- or under-fitted, bias and variance must be balanced.