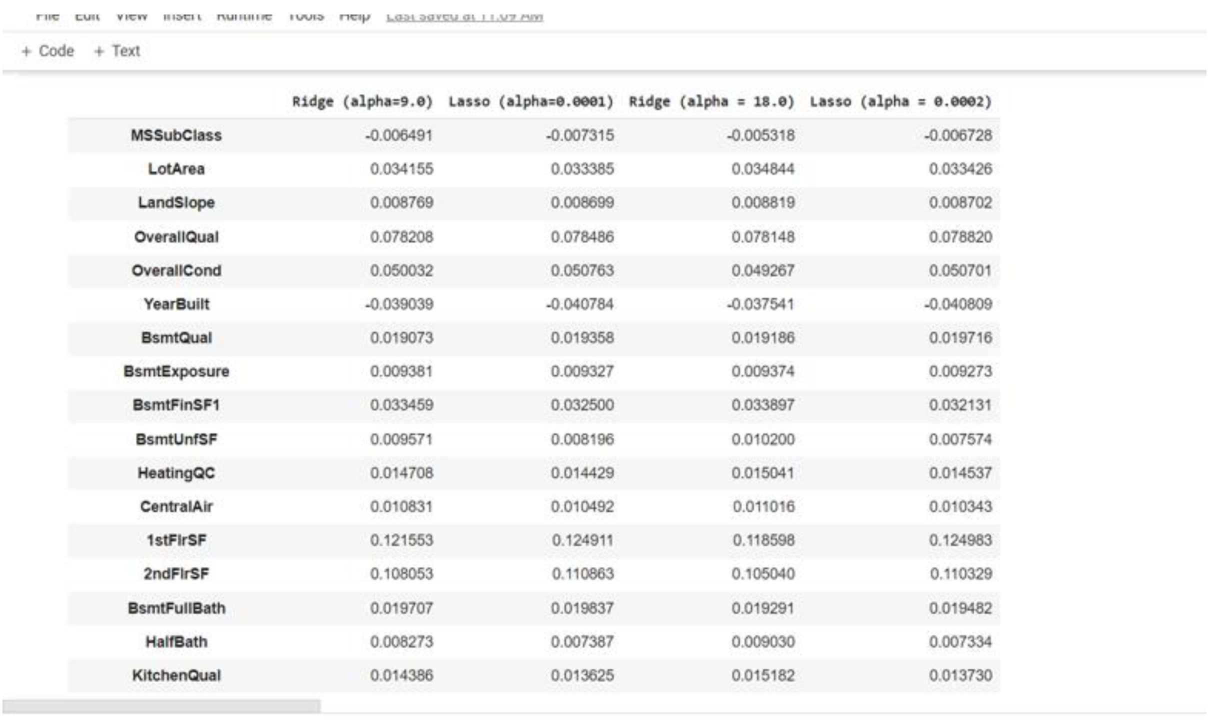
Lasso Regression and Ridge Regression

Question

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?

Ridge and Lasso Regression Model are built with optimum alpha calculated in Grid Search CV method. Optimum alpha = 9.0 for ridge and 0.0001 for lasso model.

After double the values are Ridge Regression model was able to achieve R2 score of 0.87 on test data i.e. 87% of the variance in test data can be explained by the model.



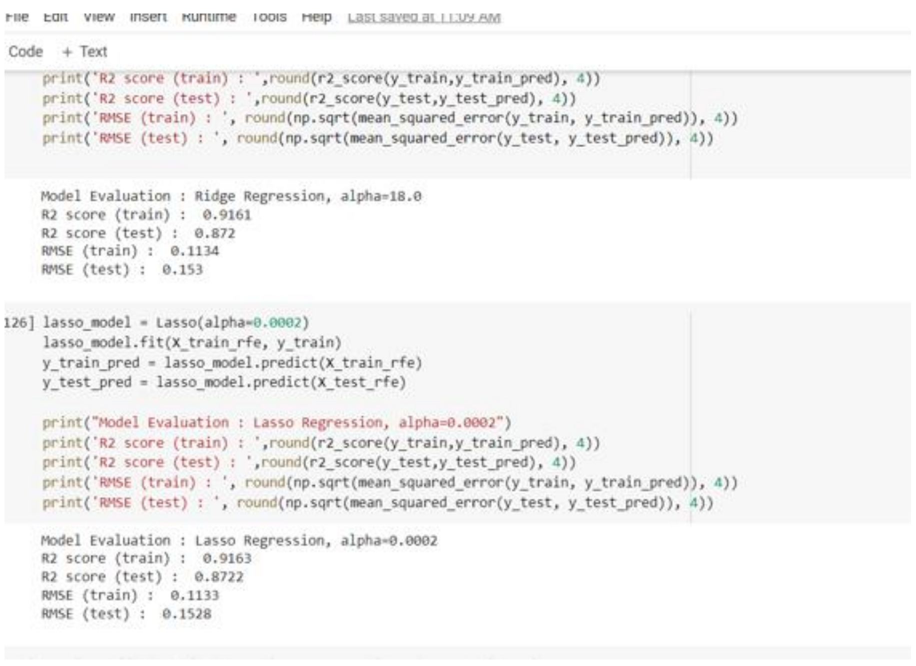
	Ridge (alpha=9.0)	Lasso (alpha=0.0001)	Ridge (alpha = 18.0)	Lasso (alpha = 0.0002)
MSSubClass	-0.006491	-0.007315	-0.005318	-0.006728
LotArea	0.034155	0.033385	0.034844	0.033426
LandSlope	0.008769	0.008699	0.008819	0.008702
OverallQual	0.078208	0.078486	0.078148	0.078820
OverallCond	0.050032	0.050763	0.049267	0.050701
YearBuilt	-0.039039	-0.040784	-0.037541	-0.040809
BsmtQual	0.019073	0.019358	0.019186	0.019716
BsmtExposure	0.009381	0.009327	0.009374	0.009273
BsmtFinSF1	0.033459	0.032500	0.033897	0.032131
BsmtUnfSF	0.009571	0.008196	0.010200	0.007574
HeatingQC	0.014708	0.014429	0.015041	0.014537
CentralAir	0.010831	0.010492	0.011016	0.010343
1stFlrSF	0.121553	0.124911	0.118598	0.124983
2ndFlrSF	0.108053	0.110863	0.105040	0.110329
BsmtFullBath	0.019707	0.019837	0.019291	0.019482
HalfBath	0.008273	0.007387	0.009030	0.007334
KitchenQual	0.014386	0.013625	0.015182	0.013730

There is no much difference after doubled the values as over all the alpha value are same

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?

Root Mean Square Error = 0.1531 on test data, that means the prediction made by the model can off by 0.1531 unit.


File Edit View Insert Runtime Tools Help Last saved at 11:09 AM
Code + Text

```
    print('R2 score (train) : ',round(r2_score(y_train,y_train_pred), 4))
    print('R2 score (test) : ',round(r2_score(y_test,y_test_pred), 4))
    print('RMSE (train) : ', round(np.sqrt(mean_squared_error(y_train, y_train_pred)), 4))
    print('RMSE (test) : ', round(np.sqrt(mean_squared_error(y_test, y_test_pred)), 4))

    Model Evaluation : Ridge Regression, alpha=18.0
    R2 score (train) :  0.9161
    R2 score (test) :  0.872
    RMSE (train) :  0.1134
    RMSE (test) :  0.153

126] lasso_model = Lasso(alpha=0.0002)
    lasso_model.fit(X_train_rfe, y_train)
    y_train_pred = lasso_model.predict(X_train_rfe)
    y_test_pred = lasso_model.predict(X_test_rfe)

    print("Model Evaluation : Lasso Regression, alpha=0.0002")
    print('R2 score (train) : ',round(r2_score(y_train,y_train_pred), 4))
    print('R2 score (test) : ',round(r2_score(y_test,y_test_pred), 4))
    print('RMSE (train) : ', round(np.sqrt(mean_squared_error(y_train, y_train_pred)), 4))
    print('RMSE (test) : ', round(np.sqrt(mean_squared_error(y_test, y_test_pred)), 4))

    Model Evaluation : Lasso Regression, alpha=0.0002
    R2 score (train) :  0.9163
    R2 score (test) :  0.8722
    RMSE (train) :  0.1133
    RMSE (test) :  0.1528
```

The mean squared error for both the ridge and lasso are same

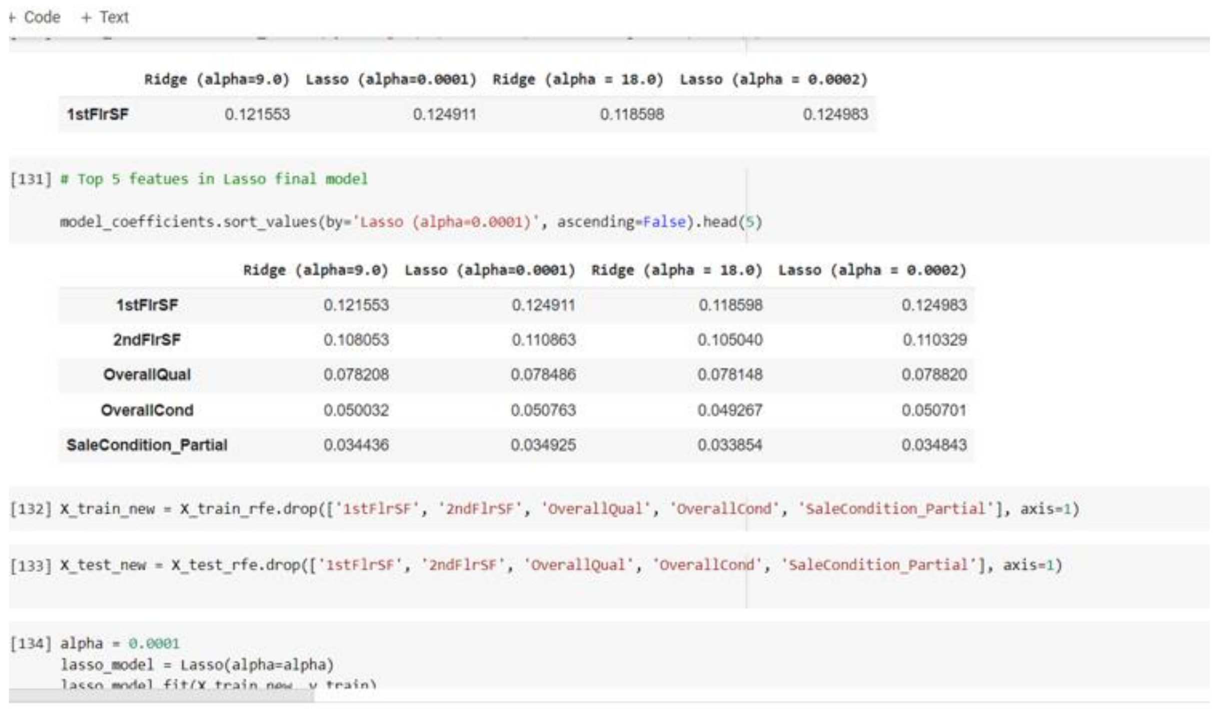
Since lasso helps in feature reduction as the co efficient value of some of the features are become zero .

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?


+ Code + Text

	Ridge (alpha=9.0)	Lasso (alpha=0.0001)	Ridge (alpha = 18.0)	Lasso (alpha = 0.0002)
1stFlrSF	0.121553	0.124911	0.118598	0.124983

```
[131] # Top 5 featues in Lasso final model

      model_coefficients.sort_values(by='Lasso (alpha=0.0001)', ascending=False).head(5)
```

	Ridge (alpha=9.0)	Lasso (alpha=0.0001)	Ridge (alpha = 18.0)	Lasso (alpha = 0.0002)
1stFlrSF	0.121553	0.124911	0.118598	0.124983
2ndFlrSF	0.108053	0.110863	0.105040	0.110329
OverallQual	0.078208	0.078486	0.078148	0.078820
OverallCond	0.050032	0.050763	0.049267	0.050701
SaleCondition_Partial	0.034436	0.034925	0.033854	0.034843

```
[132] X_train_new = X_train_rfe.drop(['1stFlrSF', '2ndFlrSF', 'OverallQual', 'OverallCond', 'SaleCondition_Partial'], axis=1)
```

```
[133] X_test_new = X_test_rfe.drop(['1stFlrSF', '2ndFlrSF', 'OverallQual', 'OverallCond', 'SaleCondition_Partial'], axis=1)
```

```
[134] alpha = 0.0001
      lasso_model = Lasso(alpha=alpha)
      lasso_model.fit(X_train_new, y_train)
```

Top 5 features

1stFlrSF
2ndFlrSF
OverallQual
OverallCond
SaleCondition_Partial

We build a losso model in the Jupiter notebook after removing these attributes from the datset
The r2 of the new model without the top 5 get dropped
Hence mean squared error inc

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?

As Per, Occam's Razor— given two models that show similar 'performance' in the finite training or test data, we should pick the one that makes fewer on the test data due to following reasons:- ✦ Simpler models are usually more 'generic' and are more widely applicable

- ✦ Simpler models require fewer training samples for effective training than the more complex ones and hence are easier to train.
- ✦ Simpler models are more robust. ○ Complex models tend to change wildly with changes in the training data set ○ Simple models have low variance, high bias and complex models have low bias, high variance
  - ○ Simpler models make more errors in the training set. Complex models lead to overfitting — they work very well for the training samples, fail miserably when applied to other test samples

Therefore, to make the model more robust and generalizable, make the model simple but not simpler which will not be of any use.

Regularization can be used to make the model simpler. Regularization helps to strike the delicate balance between keeping the model simple and not making it too naive to be of any use. For regression, regularization involves adding a regularization term to the cost that adds up the absolute values or the squares of the parameters of the model.

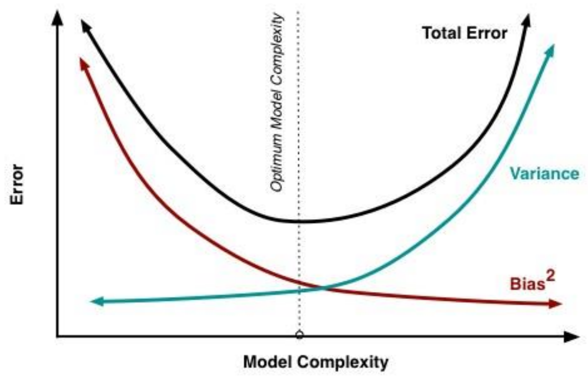
Also, Making a model simple leads to Bias-Variance Trade-off:

- A complex model will need to change for every little change in the dataset and hence is very unstable and extremely sensitive to any changes in the training data.
- A simpler model that abstracts out some pattern followed by the data points given is unlikely to change wildly even if more points are added or removed.

Bias quantifies how accurate is the model likely to be on test data. A complex model can do an accurate job prediction provided there is enough training data. Models that are too naïve, for e.g., one that gives same answer to all test inputs and makes no discrimination whatsoever has a very large bias as its expected error across all test inputs are very high.

Variance refers to the degree of changes in the model itself with respect to changes in the training data.

Thus accuracy of the model can be maintained by keeping the balance between Bias and Variance as it minimizes the total error as shown in the below graph.


Optimum Model Complexity
Total Error
Variance
Bias^2
Error
Model Complexity