**Assignment Part II – Subjective Questions**

**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:**

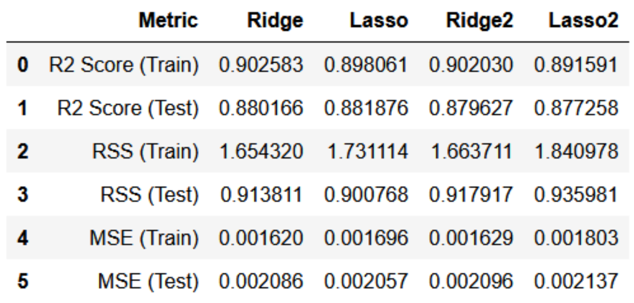
Optimal values for ridge and lasso regression are as follows:

Ridge  - 0.2

Lasso – 0.0001

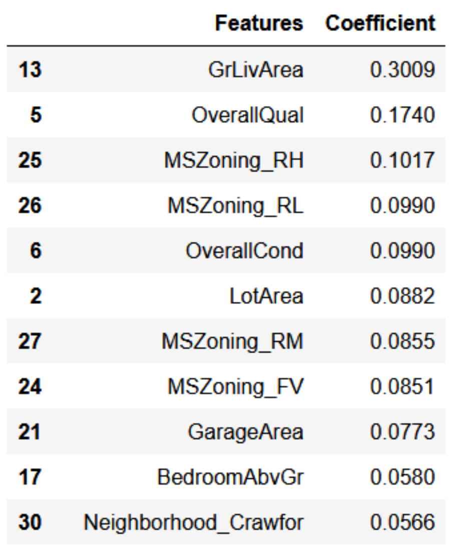
If we double the value of alpha

- The R2 scores of training and test data for both Ridge and Lasso decrease slightly.



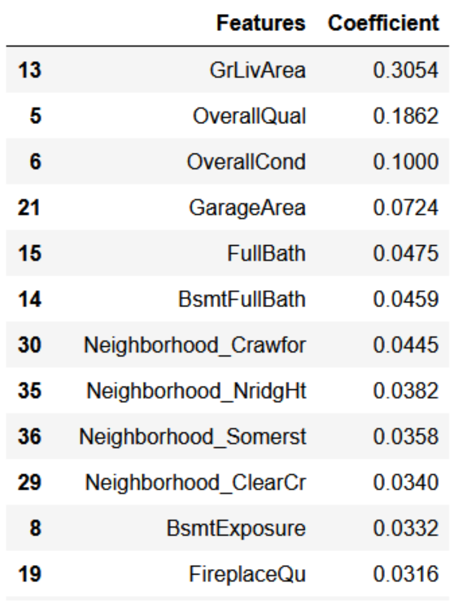
| | Metric | Ridge | Lasso | Ridge2 | Lasso2 |
|---|---|---|---|---|---|
| 0 | R2 Score (Train) | 0.902583 | 0.898061 | 0.902030 | 0.891591 |
| 1 | R2 Score (Test) | 0.880166 | 0.881876 | 0.879627 | 0.877258 |
| 2 | RSS (Train) | 1.654320 | 1.731114 | 1.663711 | 1.840978 |
| 3 | RSS (Test) | 0.913811 | 0.900768 | 0.917917 | 0.935981 |
| 4 | MSE (Train) | 0.001620 | 0.001696 | 0.001629 | 0.001803 |
| 5 | MSE (Test) | 0.002086 | 0.002057 | 0.002096 | 0.002137 |

- The predictors for both ridge and Lasso change slightly and their coefficient has also changed.
- The total number of predictors for Lasso is reduced from 42 to 38 as more number of coefficients of predictors are reduced to zero.
- The most important predictor variables and their coefficients after the change is implemented for both ridge and Lasso are as per below screenshots:
  **Ridge**



| | Features | Coefficient |
|---|---|---|
| 13 | GrLivArea | 0.3009 |
| 5 | OverallQual | 0.1740 |
| 25 | MSZoning_RH | 0.1017 |
| 26 | MSZoning_RL | 0.0990 |
| 6 | OverallCond | 0.0990 |
| 2 | LotArea | 0.0882 |
| 27 | MSZoning_RM | 0.0855 |
| 24 | MSZoning_FV | 0.0851 |
| 21 | GarageArea | 0.0773 |
| 17 | BedroomAbvGr | 0.0580 |
| 30 | Neighborhood_Crawfor | 0.0566 |

**Lasso**



| | Features | Coefficient |
|---|---|---|
| 13 | GrLivArea | 0.3054 |
| 5 | OverallQual | 0.1862 |
| 6 | OverallCond | 0.1000 |
| 21 | GarageArea | 0.0724 |
| 15 | FullBath | 0.0475 |
| 14 | BsmtFullBath | 0.0459 |
| 30 | Neighborhood_Crawfor | 0.0445 |
| 35 | Neighborhood_NridgHt | 0.0382 |
| 36 | Neighborhood_Somerst | 0.0358 |
| 29 | Neighborhood_ClearCr | 0.0340 |
| 8 | BsmtExposure | 0.0332 |
| 19 | FireplaceQu | 0.0316 |

**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 optimal lambda value in case of Ridge and Lasso is as below:
  Ridge – 0.2
  Lasso - 0.0001
- The Mean Squared error in case of Ridge and Lasso are:
  Ridge - 0.002086
  Lasso - 0.002057

The Mean Squared Error of Lasso is slightly lower than that of Ridge.

Also, since Lasso helps in feature reduction (as the coefficient value of the features shrinks to 0), Lasso has a better edge over Ridge.
Therefore, the variables predicted by Lasso can be applied to choose significant variables for predicting the price of a house.

**Question 3:**

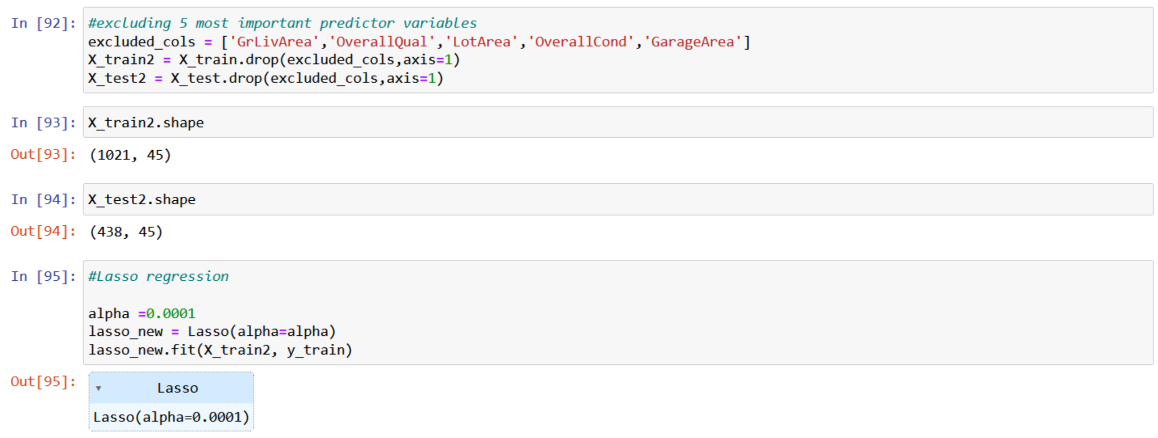
**After building the model, you realised that the five most important predictor variables in the lasso model are not available in the incoming data. You will now have to create another model excluding the five most important predictor variables. Which are the five most important predictor variables now?**

**Answer:**

After excluding the 5 most important variables and creating another Lasso model, it is observed that the five most important predictor variables are as follows:

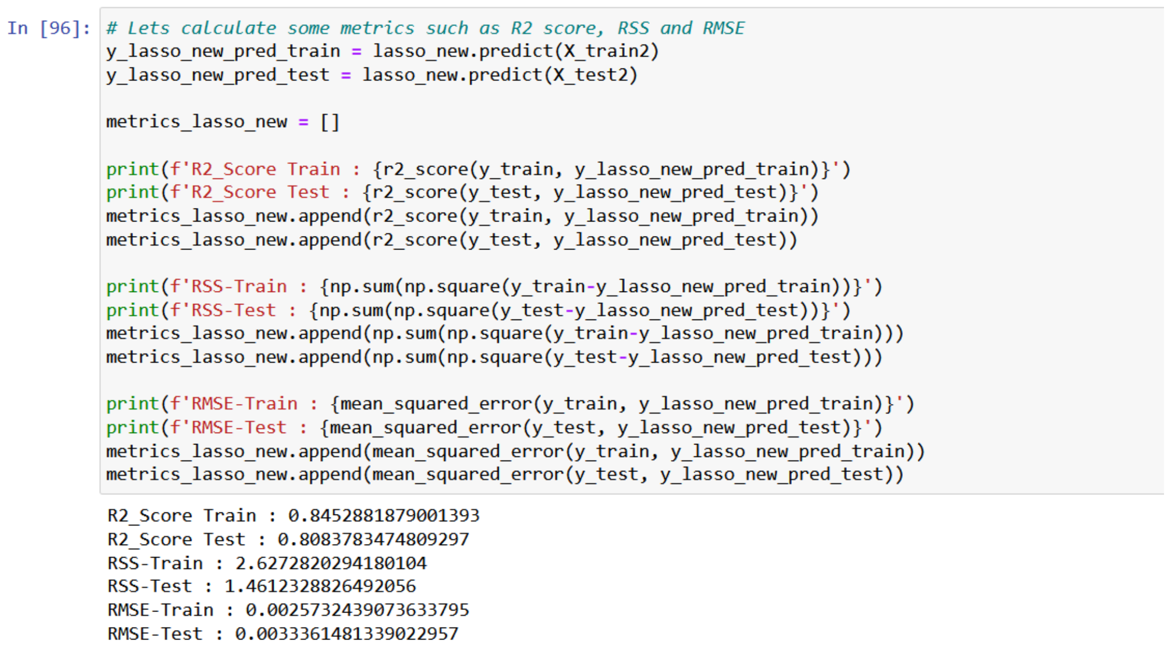
- Fullbath - Full bathrooms above grade
- BedroomAbvGr- Bedrooms above grade (does NOT include basement bedrooms)
- KitchenQual - Kitchen quality
- BsmtQual - height of the basement
- MSZoning_RL - general zoning classification of the sale, Residential Low Density

Below are the screenshots of the code snippets to create the new model and the revised metrics



```
In [92]: #excluding 5 most important predictor variables
         excluded_cols = ['GrLivArea','OverallQual','LotArea','OverallCond','GarageArea']
         X_train2 = X_train.drop(excluded_cols,axis=1)
         X_test2 = X_test.drop(excluded_cols,axis=1)

In [93]: X_train2.shape

Out[93]: (1021, 45)

In [94]: X_test2.shape

Out[94]: (438, 45)

In [95]: #Lasso regression

         alpha =0.0001
         lasso_new = Lasso(alpha=alpha)
         lasso_new.fit(X_train2, y_train)

Out[95]:     Lasso
         Lasso(alpha=0.0001)
```

It is observed that the R2 score decreases and the mean square error increases when we drop the top 5 predictor variables.



```
In [96]: # Lets calculate some metrics such as R2 score, RSS and RMSE
         y_lasso_new_pred_train = lasso_new.predict(X_train2)
         y_lasso_new_pred_test = lasso_new.predict(X_test2)

         metrics_lasso_new = []

         print(f'R2_Score Train : {r2_score(y_train, y_lasso_new_pred_train)}')
         print(f'R2_Score Test : {r2_score(y_test, y_lasso_new_pred_test)}')
         metrics_lasso_new.append(r2_score(y_train, y_lasso_new_pred_train))
         metrics_lasso_new.append(r2_score(y_test, y_lasso_new_pred_test))

         print(f'RSS-Train : {np.sum(np.square(y_train-y_lasso_new_pred_train))}')
         print(f'RSS-Test : {np.sum(np.square(y_test-y_lasso_new_pred_test))}')
         metrics_lasso_new.append(np.sum(np.square(y_train-y_lasso_new_pred_train)))
         metrics_lasso_new.append(np.sum(np.square(y_test-y_lasso_new_pred_test)))

         print(f'RMSE-Train : {mean_squared_error(y_train, y_lasso_new_pred_train)}')
         print(f'RMSE-Test : {mean_squared_error(y_test, y_lasso_new_pred_test)}')
         metrics_lasso_new.append(mean_squared_error(y_train, y_lasso_new_pred_train))
         metrics_lasso_new.append(mean_squared_error(y_test, y_lasso_new_pred_test))

         R2_Score Train : 0.8452881879001393
         R2_Score Test : 0.8083783474809297
         RSS-Train : 2.6272820294180104
         RSS-Test : 1.4612328826492056
         RMSE-Train : 0.0025732439073633795
         RMSE-Test : 0.0033361481339022957
```

**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:**

Robustness refers to a model’s ability to perform well under various conditions whereas generalizability refers to the model's ability to make accurate predictions on new unseen data. Below are some strategies to enhance robustness and generalizability of a model:

- **Diverse and Representative Data:**

  **Training Data**: Ensure that your training dataset is diverse and representative of the real-world scenarios the model will encounter. This helps the model learn a broader range of patterns.

  **Validation and Test Data**: Use separate datasets for validation and testing, and ensure that these datasets also cover a variety of situations.

- **Cross-Validation:**
  Use cross-validation to assess the model's performance across different subsets of the data. This provides a more reliable estimate of the model's generalization performance.
- **Hyperparameter Tuning:**
  Carefully tune hyperparameters to find the right balance between model complexity and simplicity. Hyperparameter tuning helps prevent overfitting and improves generalization.

- **Regularization Techniques:**

  Apply regularization methods like ridge and lasso methods prevent overfitting. Regularization helps the model generalize better to unseen data.

The implications of the same on model's accuracy are:

- **Trade-off Between Accuracy and Robustness:**

  Increasing robustness may sometimes come at the cost of reduced accuracy on the training set. However, the goal is to improve the model's performance on unseen data.

- **Avoiding Overfitting:**

  Strategies like regularization may prevent the model from fitting the training data too closely, reducing overfitting but potentially impacting training set accuracy.

- **Balancing Complexity:**

  Finding the right level of model complexity through hyperparameter tuning is crucial. A too complex model might overfit, while a too simple model might lack the capacity to generalize well.