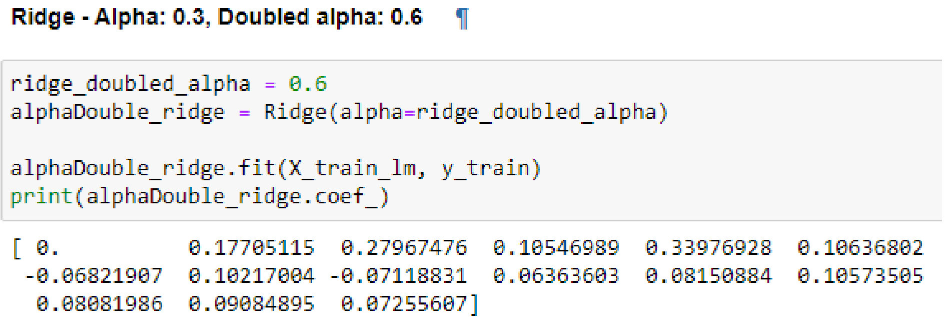
Advanced Regression Subjective Questions

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?

I. Optimal value of alpha for ridge and lasso regression?
   - Optimal value of alpha for Ridge regression: 0.3 Optimal value of alpha for Lasso regression: 0.0001

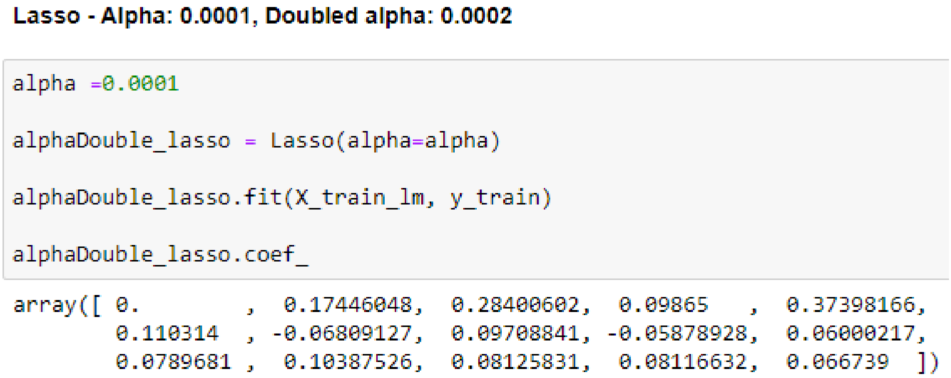
II. Let's create a new model with doubled alpha values for Ridge and Lasso (Detailed code execution details can be found on Python Jupyter notebook)
   - 

     Ridge - Alpha: 0.3, Doubled alpha: 0.6 ¶

     ```
     ridge_doubled_alpha = 0.6
     alphaDouble_ridge = Ridge(alpha=ridge_doubled_alpha)

     alphaDouble_ridge.fit(X_train_lm, y_train)
     print(alphaDouble_ridge.coef_)
     ```

     ```
     [ 0.          0.17705115  0.27967476  0.10546989  0.33976928  0.10636802
      -0.06821907  0.10217004 -0.07118831  0.06363603  0.08150884  0.10573505
       0.08081986  0.09084895  0.07255607]
     ```

   - 

     Lasso - Alpha: 0.0001, Doubled alpha: 0.0002

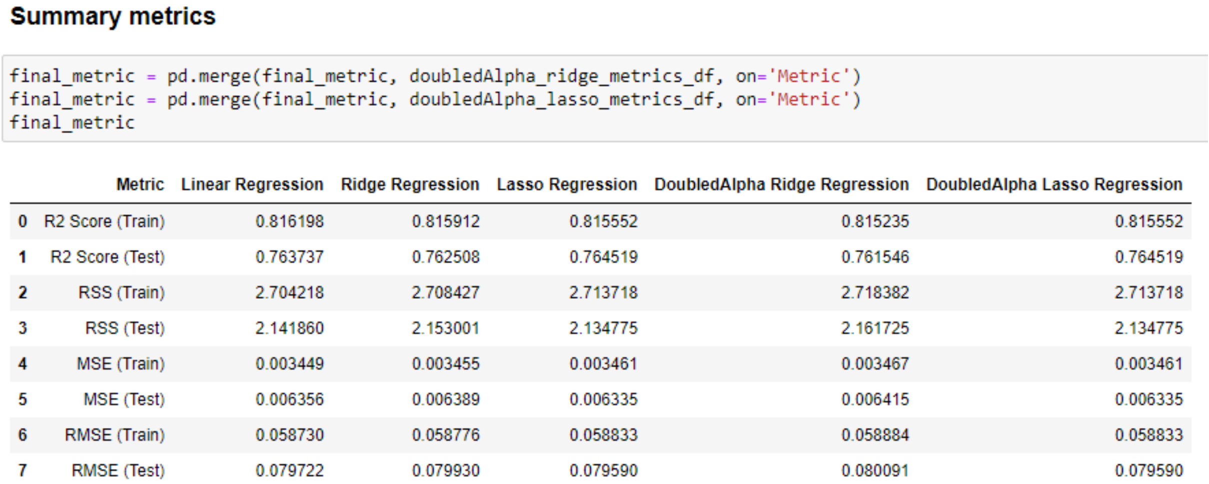
     ```
     alpha =0.0001

     alphaDouble_lasso = Lasso(alpha=alpha)

     alphaDouble_lasso.fit(X_train_lm, y_train)

     alphaDouble_lasso.coef_
     ```

     ```
     array([ 0.        ,  0.17446048,  0.28400602,  0.09865   ,  0.37398166,
             0.110314  , -0.06809127,  0.09708841, -0.05878928,  0.06000217,
             0.0789681 ,  0.10387526,  0.08125831,  0.08116632,  0.066739  ])
     ```

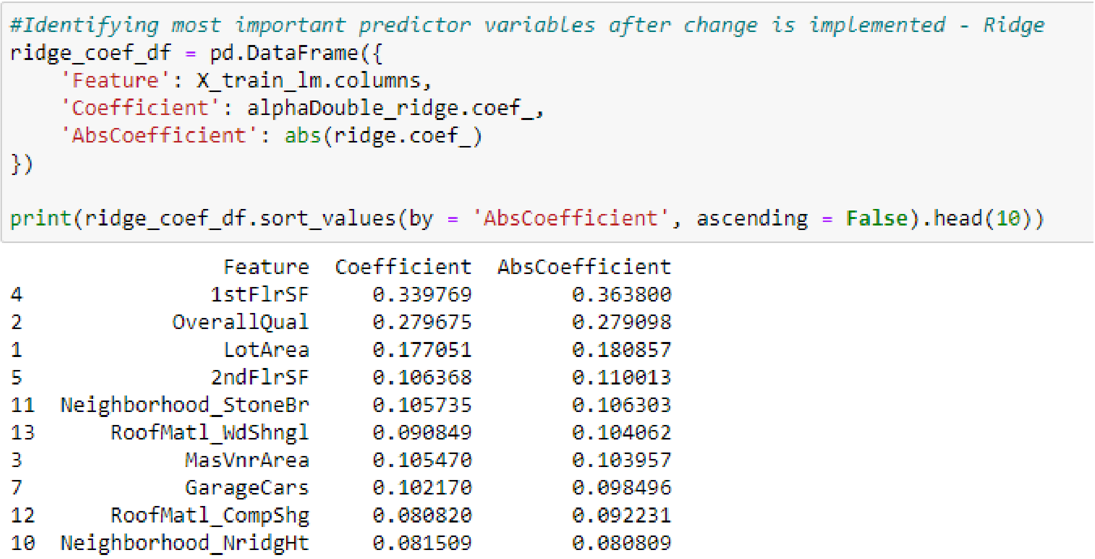
   - 

     Summary metrics

     ```
     final_metric = pd.merge(final_metric, doubledAlpha_ridge_metrics_df, on='Metric')
     final_metric = pd.merge(final_metric, doubledAlpha_lasso_metrics_df, on='Metric')
     final_metric
     ```

	Metric	Linear Regression	Ridge Regression	Lasso Regression	DoubledAlpha Ridge Regression	DoubledAlpha Lasso Regression
0	R2 Score (Train)	0.816198	0.815912	0.815552	0.815235	0.815552
1	R2 Score (Test)	0.763737	0.762508	0.764519	0.761546	0.764519
2	RSS (Train)	2.704218	2.708427	2.713718	2.718382	2.713718
3	RSS (Test)	2.141860	2.153001	2.134775	2.161725	2.134775
4	MSE (Train)	0.003449	0.003455	0.003461	0.003467	0.003461
5	MSE (Test)	0.006356	0.006389	0.006335	0.006415	0.006335
6	RMSE (Train)	0.058730	0.058776	0.058833	0.058884	0.058833
7	RMSE (Test)	0.079722	0.079930	0.079590	0.080091	0.079590

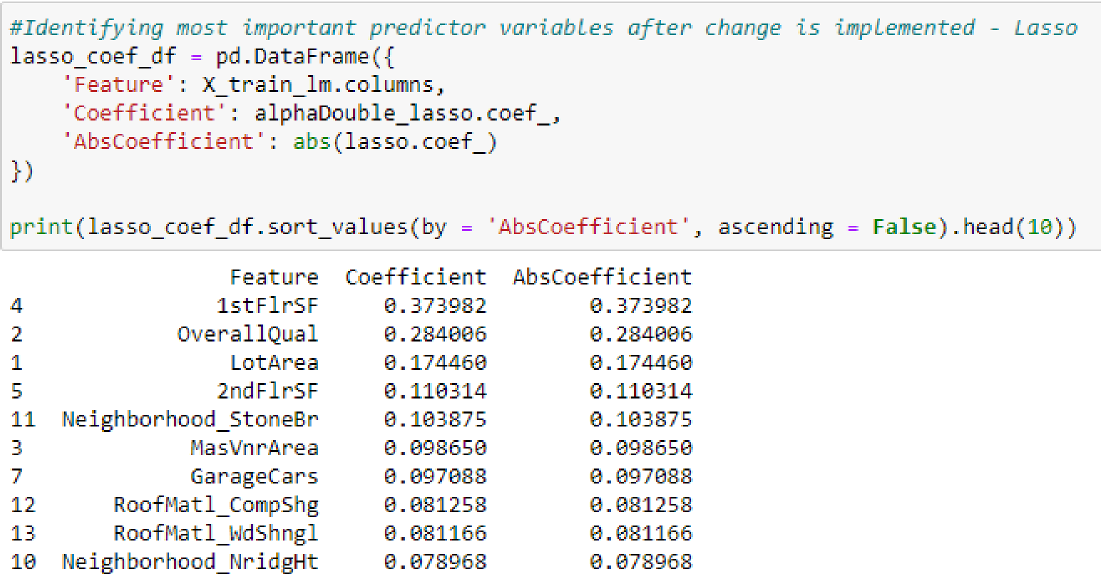
III. Most important predictor variables after the change



```python
#Identifying most important predictor variables after change is implemented - Ridge
ridge_coef_df = pd.DataFrame({
    'Feature': X_train_lm.columns,
    'Coefficient': alphaDouble_ridge.coef_,
    'AbsCoefficient': abs(ridge.coef_)
})

print(ridge_coef_df.sort_values(by = 'AbsCoefficient', ascending = False).head(10))
```

```
                 Feature  Coefficient  AbsCoefficient
4               1stFlrSF     0.339769        0.363800
2            OverallQual     0.279675        0.279098
1                LotArea     0.177051        0.180857
5               2ndFlrSF     0.106368        0.110013
11  Neighborhood_StoneBr     0.105735        0.106303
13        RoofMatl_WdShngl     0.090849        0.104062
3             MasVnrArea     0.105470        0.103957
7             GarageCars     0.102170        0.098496
12        RoofMatl_CompShg     0.080820        0.092231
10  Neighborhood_NridgHt     0.081509        0.080809
```



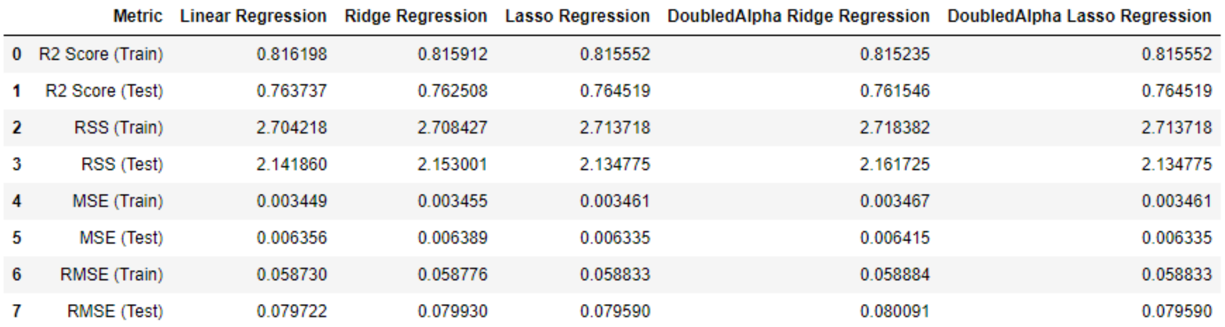
```python
#Identifying most important predictor variables after change is implemented - Lasso
lasso_coef_df = pd.DataFrame({
    'Feature': X_train_lm.columns,
    'Coefficient': alphaDouble_lasso.coef_,
    'AbsCoefficient': abs(lasso.coef_)
})

print(lasso_coef_df.sort_values(by = 'AbsCoefficient', ascending = False).head(10))
```

```
                 Feature  Coefficient  AbsCoefficient
4               1stFlrSF     0.373982        0.373982
2            OverallQual     0.284006        0.284006
1                LotArea     0.174460        0.174460
5               2ndFlrSF     0.110314        0.110314
11  Neighborhood_StoneBr     0.103875        0.103875
3             MasVnrArea     0.098650        0.098650
7             GarageCars     0.097088        0.097088
12        RoofMatl_CompShg     0.081258        0.081258
13        RoofMatl_WdShngl     0.081166        0.081166
10  Neighborhood_NridgHt     0.078968        0.078968
```

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?

I. 

	Metric	Linear Regression	Ridge Regression	Lasso Regression	DoubledAlpha Ridge Regression	DoubledAlpha Lasso Regression
0	R2 Score (Train)	0.816198	0.815912	0.815552	0.815235	0.815552
1	R2 Score (Test)	0.763737	0.762508	0.764519	0.761546	0.764519
2	RSS (Train)	2.704218	2.708427	2.713718	2.718382	2.713718
3	RSS (Test)	2.141860	2.153001	2.134775	2.161725	2.134775
4	MSE (Train)	0.003449	0.003455	0.003461	0.003467	0.003461
5	MSE (Test)	0.006356	0.006389	0.006335	0.006415	0.006335
6	RMSE (Train)	0.058730	0.058776	0.058833	0.058884	0.058833
7	RMSE (Test)	0.079722	0.079930	0.079590	0.080091	0.079590

II. After carefully analyzing the above summary metrics obtained, we can derive below points:

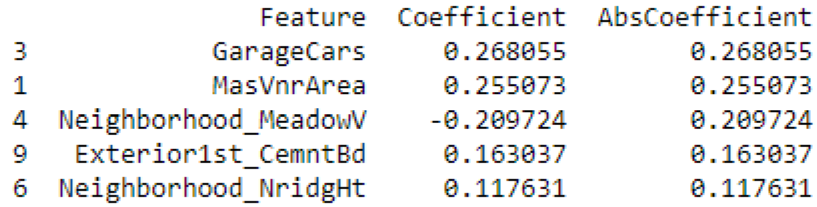
- **Lasso Regression (alpha=0.0001)** has the **highest R2 score** on the test set (0.764519) among all the models for Ridge and Lasso. This indicates that it explains a high proportion of variance in the test data.
- **Lasso Regression (alpha=0.0001)** has the **lowest RSS** (Residual Sum of Squares) on the test set (2.134775) among all the models for Ridge and Lasso. This indicates that it has the smallest prediction errors on the test data.
- **Lasso Regression (alpha=0.0001)** has the **lowest MSE** (Mean Squared Error) on the test set (0.006335). Low MSE value suggests that the model has better accuracy and performance.
- **Lasso Regression (alpha=0.0001)** also has the **lowest RMSE** (Root Mean Squared Error) on the test set (0.079590). This is also another indicator of better accuracy and performance of the model.
- **Summary**: Based on these findings we can say that, for this particular dataset, Lasso regression with alpha value of 0.0001 appears to be the best 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?

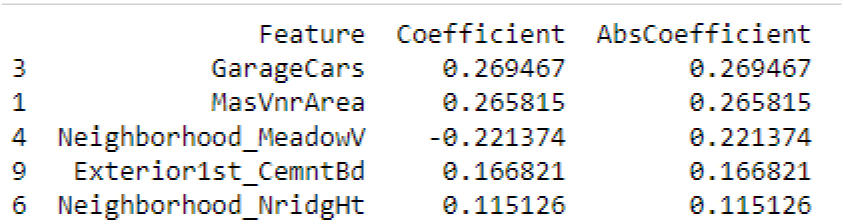
I. After elimination of initial Top 5 - Here are the other important predictor variables (Detailed code execution in Python Jupyter notebook)

- Ridge



```
                 Feature  Coefficient  AbsCoefficient
3             GarageCars     0.268055        0.268055
1             MasVnrArea     0.255073        0.255073
4   Neighborhood_MeadowV    -0.209724        0.209724
9    Exterior1st_CemntBd     0.163037        0.163037
6   Neighborhood_NridgHt     0.117631        0.117631
```

- Lasso



```
                 Feature  Coefficient  AbsCoefficient
3             GarageCars     0.269467        0.269467
1             MasVnrArea     0.265815        0.265815
4   Neighborhood_MeadowV    -0.221374        0.221374
9    Exterior1st_CemntBd     0.166821        0.166821
6   Neighborhood_NridgHt     0.115126        0.115126
```

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?

I. Below are some points to ensure that a model is robust and generalizable:

- **Occam’s Razor** also known as the principle of simplicity suggests that a model should be as simple as necessary. With this concept we require fewer assumptions and we eliminate unnecessary complexity within the model.
- **Bias-Variance Tradeoff**: Bias measures how accurately a model can describe the actual task at hand. Variance measures how flexible the model is with respect to changes in the training data. As complexity increases, bias reduces and variance increases, and we aim to find the optimal point where the total model error is the least.
- **Avoiding overfitting**: A model memorizes the data rather than learning the underlying trends in the data in which case it performs very well on train data but this causes a problem when we test on unseen or real world data. Thus an over-fit model can provide inaccurate predictions and cannot perform well on unseen data.
- **Regularization**:
  - Prevents overfitting by adding penalty for large coefficients.
  - Feature selection helps with picking the most important features.

    - Ridge and Lasso regularization methods can be used depending on nature of data and the goal of the analysis.

- **Cross Validation**:
    - It involves splitting the dataset into multiple subsets or folds, training and testing the model on different subsets and then averaging the evaluation metrics across these iterations. It can be used to evaluate how well model can perform on unseen data and also for selecting the optimal hyper-parameters.

II. Implication of generalized model

- A generalized model is less likely to over-fit on training data and more likely to learn and understand the data trends and hence perform well on the test / unseen data this improving the model accuracy.
- It is more consistent across different datasets as a generalized model does not memorize the training data whereas understanding the underlying trends and patterns makes it less sensitive to minor changes in the data.
- A generalized model is also more scalable