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?

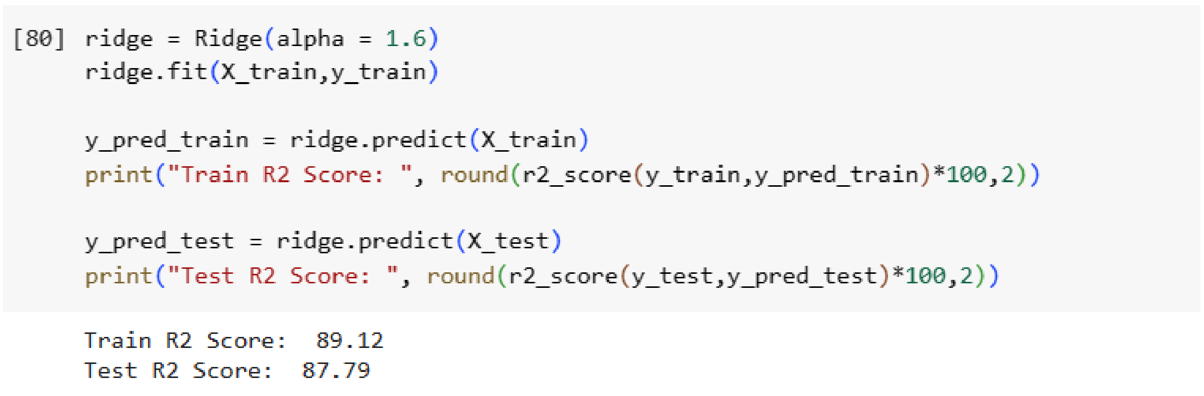
Answer:

The optimal value of alpha for ridge and lasso regression: -

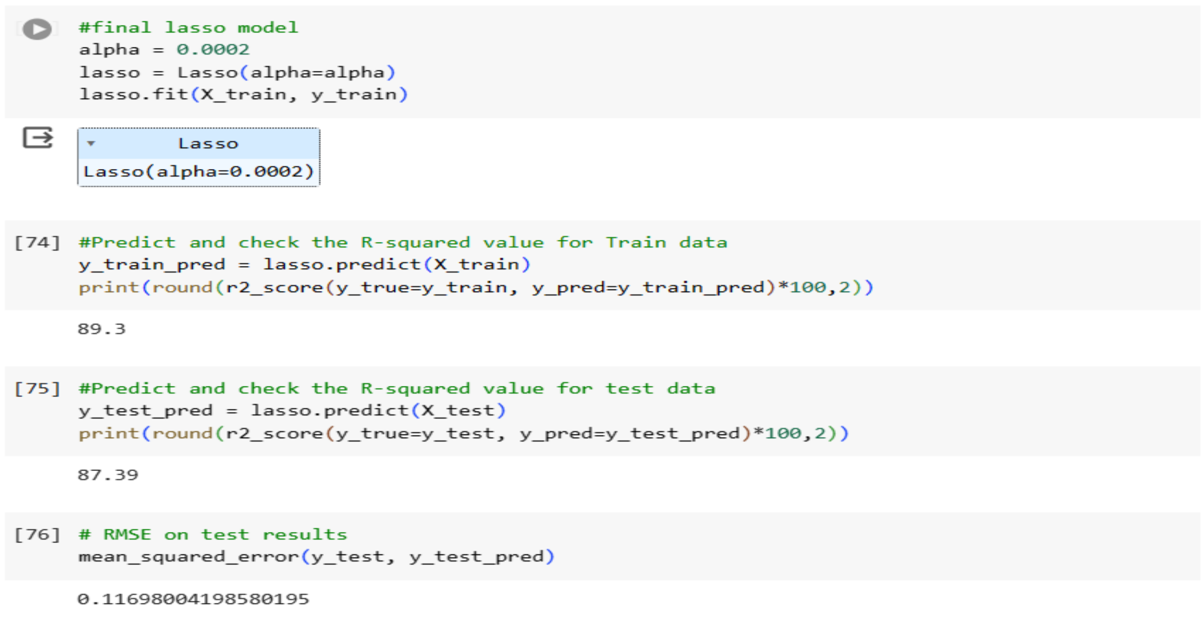
Ridge Alpha value: 0.8

Lasso Alpha value: 0.0001

After doubling the value of alpha for both ridge & lasso, below are the impact on R2 score & RMSE.

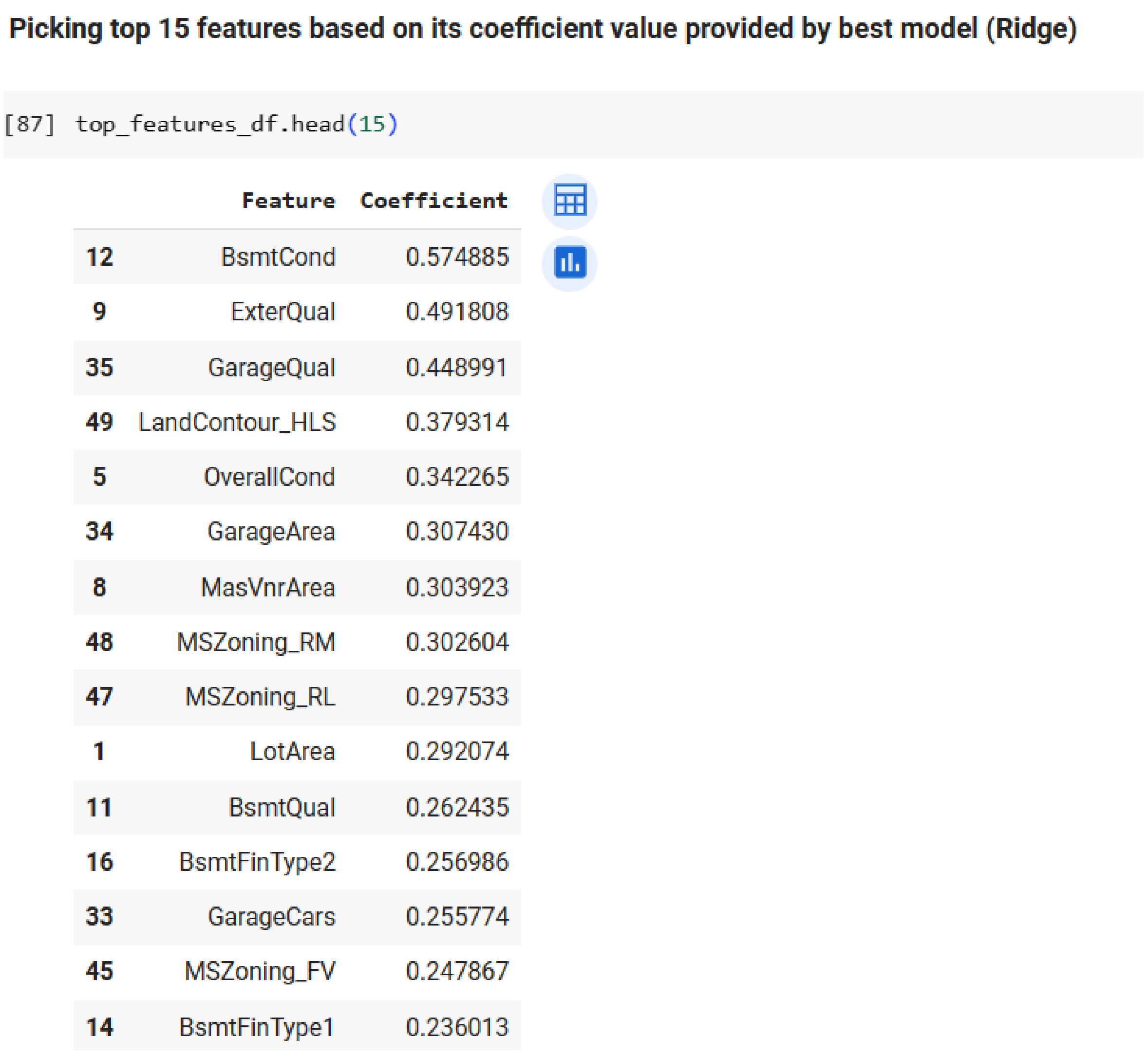


```
[80] ridge = Ridge(alpha = 1.6)
     ridge.fit(X_train,y_train)

     y_pred_train = ridge.predict(X_train)
     print("Train R2 Score: ", round(r2_score(y_train,y_pred_train)*100,2))

     y_pred_test = ridge.predict(X_test)
     print("Test R2 Score: ", round(r2_score(y_test,y_pred_test)*100,2))

     Train R2 Score:  89.12
     Test R2 Score:  87.79
```



```
#final lasso model
alpha = 0.0002
lasso = Lasso(alpha=alpha)
lasso.fit(X_train, y_train)

      Lasso
Lasso(alpha=0.0002)

[74] #Predict and check the R-squared value for Train data
     y_train_pred = lasso.predict(X_train)
     print(round(r2_score(y_true=y_train, y_pred=y_train_pred)*100,2))

     89.3

[75] #Predict and check the R-squared value for test data
     y_test_pred = lasso.predict(X_test)
     print(round(r2_score(y_true=y_test, y_pred=y_test_pred)*100,2))

     87.39

[76] # RMSE on test results
     mean_squared_error(y_test, y_test_pred)

     0.11698004198580195
```

Top 15 features remain almost same but their coefficient value changed.



Picking top 15 features based on its coefficient value provided by best model (Ridge)

```
[87] top_features_df.head(15)
```

	Feature	Coefficient
12	BsmtCond	0.574885
9	ExterQual	0.491808
35	GarageQual	0.448991
49	LandContour_HLS	0.379314
5	OverallCond	0.342265
34	GarageArea	0.307430
8	MasVnrArea	0.303923
48	MSZoning_RM	0.302604
47	MSZoning_RL	0.297533
1	LotArea	0.292074
11	BsmtQual	0.262435
16	BsmtFinType2	0.256986
33	GarageCars	0.255774
45	MSZoning_FV	0.247867
14	BsmtFinType1	0.236013

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:

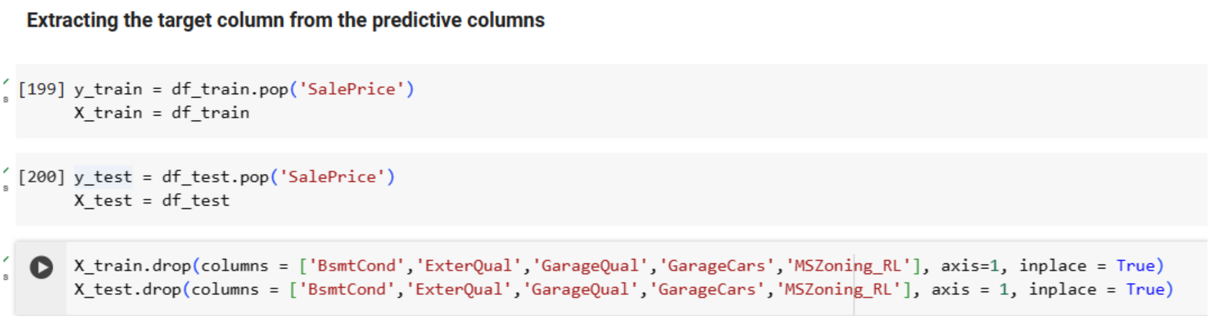
The R2 score of Ridge regression slightly surpasses that of Lasso regression for the test dataset, indicating its superiority in addressing this problem. Therefore, we will opt for Ridge regression to tackle this challenge.

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 removing the top 5 predictive variable, when I built ridge and lasso regression model below are the result:

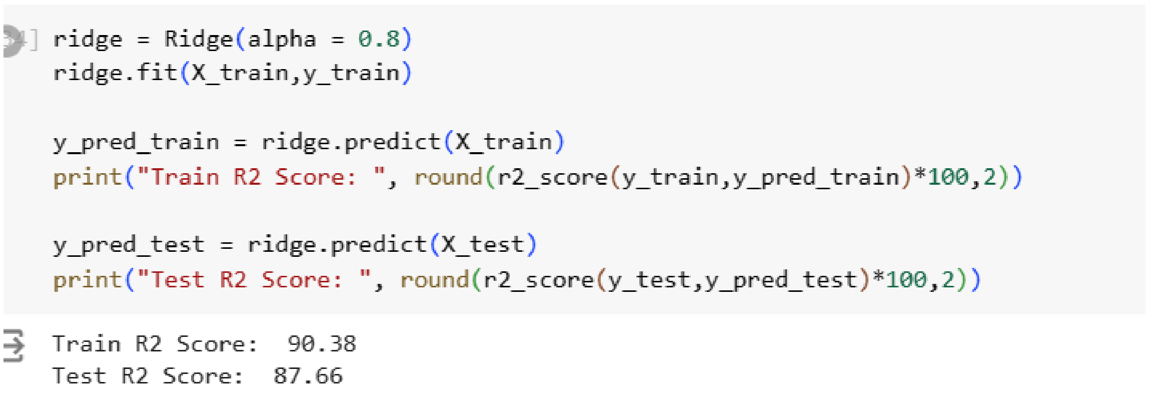


Extracting the target column from the predictive columns

```
[199] y_train = df_train.pop('SalePrice')
      X_train = df_train
```

```
[200] y_test = df_test.pop('SalePrice')
      X_test = df_test
```

```
X_train.drop(columns = ['BsmtCond','ExterQual','GarageQual','GarageCars','MSZoning_RL'], axis=1, inplace = True)
X_test.drop(columns = ['BsmtCond','ExterQual','GarageQual','GarageCars','MSZoning_RL'], axis = 1, inplace = True)
```



```
ridge = Ridge(alpha = 0.8)
ridge.fit(X_train,y_train)

y_pred_train = ridge.predict(X_train)
print("Train R2 Score: ", round(r2_score(y_train,y_pred_train)*100,2))

y_pred_test = ridge.predict(X_test)
print("Test R2 Score: ", round(r2_score(y_test,y_pred_test)*100,2))
```

```
Train R2 Score:  90.38
Test R2 Score:  87.66
```

```
[227] #final lasso model
      alpha = 0.0001
      lasso = Lasso(alpha=alpha)
      lasso.fit(X_train, y_train)
```

```
      ▾      Lasso
      Lasso(alpha=0.0001)
```

```
[228] #Predict and check the R-squared value for Train data
      y_train_pred = lasso.predict(X_train)
      print(round(r2_score(y_true=y_train, y_pred=y_train_pred)*100,2))
```

```
      90.54
```

```
[229] #Predict and check the R-squared value for test data
      y_test_pred = lasso.predict(X_test)
      print(round(r2_score(y_true=y_test, y_pred=y_test_pred)*100,2))
```

```
      86.91
```

```
[230] # RMSE on test results
      mean_squared_error(y_test, y_test_pred)
```

```
      0.12139262867276523
```

Top 5 features post building the new model are:

	Feature	Coefficient
34	OpenPorchSF	0.504151
13	BsmtFinSF1	0.503744
18	CentralAir	0.434378
46	LandContour_Lvl	0.416693
10	BsmtQual	0.412363

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:

Ensuring that a model is robust and generalizable is crucial for its effectiveness in real-world applications. Here are several strategies to achieve this and their implications for model accuracy:

Train-Test Split: Splitting the available data into separate training and test sets allows us to evaluate the model's performance on unseen data. By assessing how well the model generalizes to the test set, we gain insights into its robustness. If the model performs well on both the training and test sets, it's likely to be more robust and generalizable.

Cross-Validation: Techniques like k-fold cross-validation help in assessing the model's performance across multiple subsets of the data. This provides a more comprehensive evaluation of the model's generalization ability and robustness.

Regularization: Regularization techniques such as Lasso or Ridge regression can prevent overfitting by penalizing overly complex models. These techniques add constraints to the model, encouraging simpler and more generalizable solutions. While regularization may slightly reduce the accuracy on the training set, it often leads to improved performance on unseen data by reducing overfitting.

Feature Engineering: Thoughtful feature selection and engineering can improve the model's generalization ability. Removing irrelevant features or transforming them into more meaningful representations can help the model focus on the most informative aspects of the data. This can lead to a more accurate and generalizable model by reducing noise and increasing signal.

Implications for Model Accuracy:

Training Accuracy vs. Test Accuracy: A robust and generalizable model will typically exhibit similar performance on both the training and test sets. If there's a significant gap between the training and test accuracies, it suggests that the model may be overfitting the training data and failing to generalize well to unseen data.

Stability Across Multiple Datasets: A robust model should maintain consistent performance across different datasets that share similar characteristics. If the model's performance varies widely across different datasets, it indicates that it may be sensitive to specific quirks or biases in the training data.