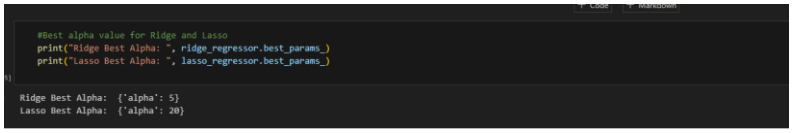
Question-1:

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:

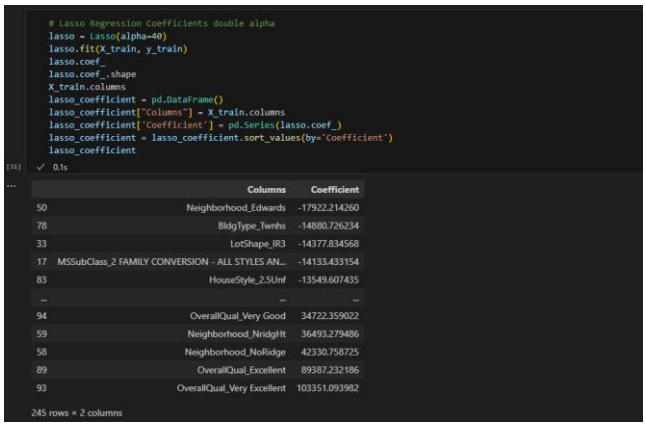
Optimal value for alpha for Ridge is Regression 5 and Lasso Regression is 20



```
#Best alpha value for Ridge and Lasso
print("Ridge Best Alpha: ", ridge_regressor.best_params_)
print("Lasso Best Alpha: ", lasso_regressor.best_params_)
```

```
Ridge Best Alpha:  {'alpha': 5}
Lasso Best Alpha:  {'alpha': 20}
```

If we choose double the value of alpha for both ridge and lasso, the coefficients of the features will be reduced further. This will lead to a more regularized model and will help in reducing overfitting. The model will become more robust and generalizable. However, the accuracy of the model may decrease as the model will be less flexible and may underfit the data.



```
# Lasso Regression Coefficients double alpha
lasso = Lasso(alpha=40)
lasso.fit(X_train, y_train)
lasso.coef_
lasso.coef_.shape
X_train.columns
lasso_coefficient = pd.DataFrame()
lasso_coefficient["Columns"] = X_train.columns
lasso_coefficient['Coefficient'] = pd.Series(lasso.coef_)
lasso_coefficient = lasso_coefficient.sort_values(by='Coefficient')
lasso_coefficient
```

✓ 0.1s

	Columns	Coefficient
50	Neighborhood_Edwards	-17922.214260
78	BldgType_Twnhs	-14880.726234
33	LotShape_IR3	-14377.834568
17	MSSubClass_2 FAMILY CONVERSION - ALL STYLES AN...	-14133.433154
83	HouseStyle_2.5Unf	-13549.607435
...	...	...
94	OverallQual_Very Good	34722.359022
59	Neighborhood_NridgHt	36493.279486
58	Neighborhood_NoRidge	42330.758725
89	OverallQual_Excellent	89387.232186
93	OverallQual_Very Excellent	103351.093982

245 rows × 2 columns

The Most Important Predictor Variables are: TotalhouseSF, Overall_Quality_Excellent, Neighborhood_NoRidge, Neighborhood_NridgHt

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:

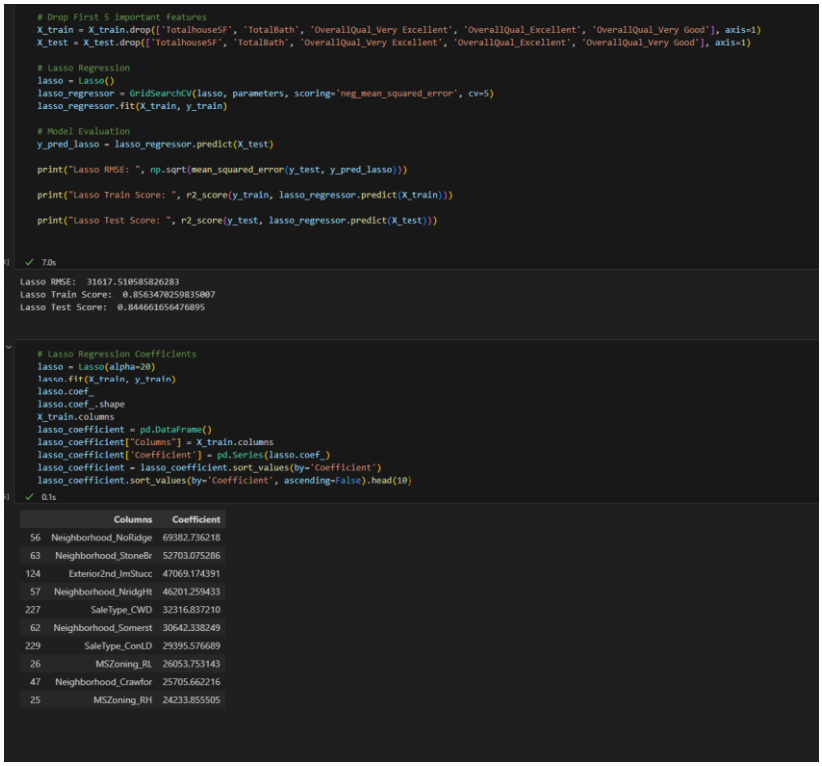
Lasso regression is my choice because it selects the relevant features and shrinks the coefficients of less relevant features to zero. It also costs less computationally than Ridge regression. Lasso regression is helpful when we have many features, and we want to prevent overfitting.

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:

Next 5 important coefficients are Neighborhood_NoRidge, Neighborhood_StoneBr, Exterior2nd_ImStucc, Neighborhood_NridgHt, SaleType_CWD



```
# Drop First 5 Important features
X_train = X_train.drop(['TotalhouseSF', 'TotalBath', 'OverallQual_Very Excellent', 'OverallQual_Excellent', 'OverallQual_Very Good'], axis=1)
X_test = X_test.drop(['TotalhouseSF', 'TotalBath', 'OverallQual_Very Excellent', 'OverallQual_Excellent', 'OverallQual_Very Good'], axis=1)

# Lasso Regression
lasso = Lasso()
lasso_regressor = GridSearchCV(lasso, parameters, scoring='neg_mean_squared_error', cv=5)
lasso_regressor.fit(X_train, y_train)

# Model Evaluation
y_pred_lasso = lasso_regressor.predict(X_test)

print("Lasso RMSE: ", np.sqrt(mean_squared_error(y_test, y_pred_lasso)))

print("Lasso Train Score: ", r2_score(y_train, lasso_regressor.predict(X_train)))

print("Lasso Test Score: ", r2_score(y_test, lasso_regressor.predict(X_test)))
```

✓ 7.0s

```
Lasso RMSE:  31617.510585826283
Lasso Train Score:  0.8563470259835007
Lasso Test Score:  0.844661656476895
```

```
# Lasso Regression Coefficients
lasso = Lasso(alpha=20)
lasso.fit(X_train, y_train)
lasso.coef_
lasso.coef_.shape
X_train.columns
lasso_coefficient = pd.DataFrame()
lasso_coefficient["Columns"] = X_train.columns
lasso_coefficient['Coefficient'] = pd.Series(lasso.coef_)
lasso_coefficient = lasso_coefficient.sort_values(by='Coefficient')
lasso_coefficient.sort_values(by='Coefficient', ascending=False).head(10)
```

✓ 0.1s

	Columns	Coefficient
56	Neighborhood_NoRidge	69382.736218
63	Neighborhood_StoneBr	52703.075286
124	Exterior2nd_ImStucc	47069.174391
57	Neighborhood_NridgHt	46201.259433
227	SaleType_CWD	32316.837210
62	Neighborhood_Somerst	30642.338249
229	SaleType_ConLD	29395.576689
26	MSZoning_RL	26053.753143
47	Neighborhood_Crawfor	25705.662216
25	MSZoning_RH	24233.855505

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:

A model is strong and adaptable if it does well on the training data and also on the new data. To ensure that a model is strong and adaptable, we can apply the following methods:

Cross-Validation: We can use cross-validation to test the performance of the model on different parts of the data. This will help us to know how the model will do on new data.

Regularization: We can use regularization methods like Lasso and Ridge regression to decrease

overfitting and make the model more adaptable.

Feature Selection: We can use feature selection methods to choose only the most relevant features and reduce the difficulty of the model.

Hyperparameter Tuning: We can use hyperparameter tuning to find the best parameters for the model and make it more adaptable.

The consequences of having a strong and adaptable model are that it will do well on new data and will be less likely to overfit. This will help us to make better predictions and improve the precision of the model.