**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 as follows:

Optimal Value of alpha for ridge regression: 0.10

Optimal Value of alpha for Lasso regression: 0.001

we got a decent score for both Ridge and Lasso regression with above specified alpha values.

Ridge: Train :90.1 Test :86.7

Lasso: Train :89.8 Test :86.4

The change in the model if we choose to double the value of alpha for both ridge and lasso is

Ridge: Train :90.9 Test :87.4

Lasso: Train :88.0 Test :84.9

Most important predicter variables are:

Ridge:

('GarageFinish_RFn', 0.092)

('GarageFinish_Unf', 0.094)

('SaleCondition_Normal', 0.099)

('SaleCondition_Others', 0.105)

('SaleCondition_Partial', 0.143)

Lasso:

('SaleCondition_Partial', 0.198)

('SaleCondition_Others', 0.12)

('SaleCondition_Normal', 0.098)

('GarageFinish_Unf', 0.084)

('GarageFinish_RFn', 0.079)

**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**:

Ridge regression consistently performs better than Lasso in terms of both training and test scores.

Doubling the value of alpha in Ridge leads to a further improvement in the test score.

In Lasso, doubling the value of alpha results in a decrease in both training and test scores.

Considering these observations, Ridge regression seems to be the more robust choice in this scenario. It provides higher test scores, and increasing the regularization strength (by doubling alpha) even improves the performance. Additionally, Ridge tends to be less sensitive to outliers compared to Lasso, which makes it a more stable choice

**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**:

To identify the five most important predictor variables after excluding the ones not available in the incoming data from the Lasso model, you would need to re-run the Lasso regression with the modified set of predictor variables. It's important to note that the importance of variables can change when certain features are excluded.

Assuming you've removed the five most important predictor variables identified in the Lasso model from the incoming data, you can train the new Lasso model and then check the coefficients of the remaining variables.

Here's a general approach to finding the new most important predictor variables:

Remove the five variables:

Remove the predictor variables that were identified as the most important in the original Lasso model and are not available in the incoming data.

Train the Lasso model:

Train the Lasso regression model using the modified dataset (excluding the five variables).

Identify important variables:

Check the coefficients of the remaining variables in the new Lasso model. The variables with non-zero coefficients are considered important in the context of the modified dataset.

**Code:**

```
X_train_modified = X_train.drop(['GarageFinish_RFn', 'GarageFinish_Unf', 'SaleCondition_Normal',
                                 'SaleCondition_Others', 'SaleCondition_Partial'], axis=1)

# Train the Lasso model
lasso_model_modified = Lasso(alpha=0.001)  # Use the optimal alpha value
lasso_model_modified.fit(X_train_modified, y_train)

# Identify important variables in the modified model
important_variables_modified = [(feature, coef) for feature, coef in zip(X_train_modified.columns,
                                                                         lasso_model_modified.coef_) if coef != 0]

# Print the top five important variables
top_five_important_variables_modified = sorted(important_variables_modified, key=lambda x: abs(x[1]),
reverse=True)[:5]

print(top_five_important_variables_modified)
```

**Output:**

```
[('OverallCond_Fair', -0.2155939274257587), ('OverallQual_Excellent', 0.192
91092889261635), ('OverallQual_Very Good', 0.1176587987680504), ('KitchenAb
vGr', -0.10406165353479654), ('MSSubClass_1-STORY 1945 & OLDER', -0.1027271
3763213603)]
```

**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 making accurate predictions on new, unseen data. Here are some key considerations and practices to enhance the robustness and generalizability of a model:

Cross-Validation:

Use techniques like k-fold cross-validation to assess how well the model performs on different subsets of the data. This helps to detect overfitting or underfitting and provides a more reliable estimate of the model's performance on unseen data.

Train-Test Split:

Split the dataset into separate training and testing sets. Train the model on the training set and evaluate its performance on the test set. This helps simulate how well the model will perform on new, unseen data.

Feature Engineering:

Carefully select and engineer features. Remove irrelevant or redundant features and transform variables appropriately. Feature scaling and normalization can be important for certain algorithms.

Regularization:

Apply regularization techniques like L1 (Lasso) or L2 (Ridge) regularization to prevent overfitting. Regularization adds a penalty term to the model's objective function, discouraging overly complex models.

Hyperparameter Tuning:

Optimize hyperparameters using techniques like grid search or randomized search. This ensures that the model is configured with the best set of hyperparameters for the given data.