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 : In this case, the optimal value for alpha for ridge and lasso regression are as follows:

- Ridge regression - Alpha = 9.0
- Lasso Regression - Alpha = 0.0001

The following are the changes in the model if these alpha values are doubled:

- Ridge Regression Alpha = 9.0 to 18.0

  Model Evaluation : Ridge Regression, alpha=9.0
  R2 score (train) : 0.9138
  R2 score (test) : 0.8675
  RMSE (train) : 0.115
  RMSE (test) : 0.1556

  Model Evaluation : Ridge Regression, alpha=18.0
  R2 score (train) : 0.9136
  R2 score (test) : 0.8679
  RMSE (train) : 0.1151
  RMSE (test) : 0.1554

- Lasso Regressin Alpha = 0.0001 to 0.0002

  Model Evaluation : Lasso Regression, alpha=0.0001
  R2 score (train) : 0.9138
  R2 score (test) : 0.8673
  RMSE (train) : 0.115
  RMSE (test) : 0.1557

  Model Evaluation : Lasso Regression, alpha=0.0002
  R2 score (train) : 0.9138
  R2 score (test) : 0.8678
  RMSE (train) : 0.115
  RMSE (test) : 0.1555

The "**1stFlrSF**" is the top predictor when alpha value of both ridge and lasso regression are doubled.

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: Regularizing coefficients is a crucial aspect of enhancing prediction accuracy while reducing variance and ensuring model interpretability. Ridge regression employs a tuning parameter known as lambda to introduce a penalty proportional to the square of the coefficients, which is determined through cross-validation. This penalty aims to minimize the residual sum of squares by constraining the magnitude of coefficients. With Ridge regression, coefficients with larger values are penalized more heavily. As we increase the lambda value, the model's variance decreases, while the bias remains relatively constant. Notably, Ridge regression includes all variables in the final model, unlike Lasso Regression.

On the other hand, Lasso regression also utilizes a tuning parameter called lambda, but it introduces a penalty based on the absolute value of coefficients, as determined through cross-validation. As the lambda value increases in Lasso, it drives coefficients closer to zero and can even force some variables to become exactly equal to zero, effectively performing variable selection. When the lambda value is small, Lasso behaves like simple linear regression, and as lambda increases, it introduces shrinkage, ultimately excluding variables with zero coefficients from the model.

Also, Lasso scored slightly better R2 score on test (unseen) data.

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: The top 5 predictor variables in the final model are:

- **1stFlrSF**
- **2ndFlrSF**
- **OverallQual**
- **OverallCond**
- **LotArea**

The top 5 predictor variables after dropping these variables and re-building the model are:

- **BsmtFinSF1**
- **LotArea**
- **BsmtUnfSF**
- **GarageArea**
- **KitchenQual**

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 real-world applicability and reliability. Here are several strategies and considerations to achieve this:

Outlier Handling: Detect and handle outliers in your data. Outliers can significantly affect model performance and generalization.

Model Selection: Experiment with different types of models and algorithms. It's possible that a simpler model may generalize better than a more complex one.

Data Splitting: Divide the dataset into at least two subsets: a training set and a testing set (or validation set). The training set is used to train the model, while the testing set is used to evaluate its performance. This helps ensure that the model doesn't just memorize the training data but can generalize to unseen data.

Cross-Validation: Using techniques like k-fold cross-validation to assess the model's performance on multiple subsets of the data. This can give us a more robust estimate of how well your model generalizes.

Feature Engineering: Carefully select and engineer features that are relevant to the problem. Removing irrelevant or redundant features can help the model focus on the most important information.

Regularization: Employ techniques like L1 (Lasso) and L2 (Ridge) regularization to prevent overfitting. Regularization adds penalties to the model's coefficients, discouraging them from taking on extreme values.

Hyperparameter Tuning: Fine-tune model hyperparameters using cross-validation to find the best configuration. This ensures that the model is not overfitting to specific hyperparameter values.

Implications for Model Accuracy:

Robustness and generalization are often trade-offs with model accuracy. When a model is overfitting (fitting too closely to the training data), it may have high accuracy on the training data but perform poorly on unseen data. Generalization techniques, like regularization and careful validation, may lead to slightly lower training accuracy but better performance on new, unseen data.

Achieving a good balance between training accuracy and test (or validation) accuracy is essential. The model should perform well on both datasets, indicating that it has learned the underlying patterns rather than just memorizing the training data.

It is known that overly complex models are more prone to overfitting, and simpler models may have lower training accuracy but higher generalization performance.

In summary, robust and generalizable models are less likely to suffer from overfitting and will perform better on new, unseen data. While this might lead to a trade-off with training accuracy, it ensures that the model is more reliable and practical for real-world applications.