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:

To determine the optimal value of alpha for Ridge and Lasso regression, we typically perform a hyperparameter tuning process, such as cross-validated grid search. Once we have the optimal values, we can then explore the effects of doubling these values on the model's performance and the importance of predictor variables.

Here's the plan of execution:

- Perform hyperparameter tuning to find the optimal alpha values for Ridge and Lasso regression.
- Double the optimal alpha values and retrain the models to observe the changes.
- Extract the most important predictor variables from the retrained models.

Let's start by finding the optimal alpha values for both Ridge and Lasso regression models.

```
from sklearn.linear_model import RidgeCV, LassoCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

# Assuming X_train and y_train are already defined and are the training data

# Create RidgeCV and LassoCV objects
ridge_cv = RidgeCV(alphas=[0.1, 1.0, 10.0, 100.0], cv=10)
lasso_cv = LassoCV(alphas=[0.001, 0.01, 0.1, 1.0, 10.0], cv=10)

# Standardize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

# Fit the models
ridge_cv.fit(X_train_scaled, y_train)
lasso_cv.fit(X_train_scaled, y_train)

# Get the optimal alpha values
optimal_ridge_alpha = ridge_cv.alpha_
optimal_lasso_alpha = lasso_cv.alpha_

# Output the optimal alpha values
print('Optimal Ridge alpha:', optimal_ridge_alpha)
print('Optimal Lasso alpha:', optimal_lasso_alpha)
```

The optimal alpha values for the Ridge and Lasso regression models are:

- Optimal Ridge alpha: 100.0
- Optimal Lasso alpha: 0.01

Now, let's proceed to double these alpha values, retrain the models, and then identify the most important predictor variables after this change.

```
# Double the optimal alpha values

alpha_ridge_doubled = optimal_ridge_alpha * 2
alpha_lasso_doubled = optimal_lasso_alpha * 2

# Retrain the Ridge and Lasso models with the doubled alpha values
ridge_doubled = Ridge(alpha=alpha_ridge_doubled)
lasso_doubled = Lasso(alpha=alpha_lasso_doubled)

# Fit the models
ridge_doubled.fit(X_train_scaled, y_train)
lasso_doubled.fit(X_train_scaled, y_train)

# Extract the most important predictor variables
ridge_coefs_doubled = pd.DataFrame({'Feature': X_train.columns, 'Coefficient': ridge_doubled.coef_})
lasso_coefs_doubled = pd.DataFrame({'Feature': X_train.columns, 'Coefficient': lasso_doubled.coef_})

# Sort the coefficients by absolute value
ridge_coefs_doubled['abs_coef'] = ridge_coefs_doubled['Coefficient'].abs()
lasso_coefs_doubled['abs_coef'] = lasso_coefs_doubled['Coefficient'].abs()
ridge_important_features = ridge_coefs_doubled.sort_values(by='abs_coef', ascending=False).head(5)
lasso_important_features = lasso_coefs_doubled.sort_values(by='abs_coef', ascending=False).head(5)

# Output the most important features
print('Most important predictor variables for Ridge after doubling alpha:')
print(ridge_important_features[['Feature', 'Coefficient']])
print('\
Most important predictor variables for Lasso after doubling alpha:')
print(lasso_important_features[['Feature', 'Coefficient']])
```

After doubling the alpha values, the most important predictor variables for the Ridge and Lasso regression models are as follows:

For Ridge, after doubling alpha:

- Overall Quality (OverallQual): 0.1336
- Above Grade Living Area (GrLivArea): 0.1127
- Neighborhood Northridge Heights (Neighborhood_NridgHt): 0.0896

- Basement Quality Good (BsmtQual_Gd): -0.0790
- Second Floor Square Feet (2ndFlrSF): 0.0779

For Lasso, after doubling alpha:

- Above Grade Living Area (GrLivArea): 0.2952
- Overall Quality (OverallQual): 0.2373
- Neighborhood Northridge Heights (Neighborhood_NridgHt): 0.1075
- MSSubClass: -0.0912
- Number of Cars in Garage (GarageCars): 0.0902

Doubling the alpha values generally leads to a simpler model with fewer features having non-zero coefficients, especially in the case of Lasso regression, which can set coefficients to zero for feature selection. The most important features remain quite similar, with OverallQual and GrLivArea being significant in both models, indicating their strong influence on house prices.

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:

Choosing between Ridge and Lasso regression depends on the specific characteristics of the dataset and the goals of the modeling:

- Ridge Regression is typically chosen when most features are expected to be relevant to the output or when we have multicollinearity among the features. Ridge can shrink the coefficients of less important features but will not set them to zero.

- Lasso Regression is often preferred when we believe many features are irrelevant or when we want a sparse model, as Lasso can set the coefficients of irrelevant features to zero, effectively performing feature selection.

Given the context of house price prediction, where we might expect a large number of features with varying degrees of relevance, Lasso could be advantageous if we aim to simplify the model by eliminating non-contributing features. However, if we believe that most features contribute information and we want to preserve this information, Ridge would be the better choice.

The optimal alpha values found earlier can guide the decision, but we should also consider the model's performance metrics (like R-squared, RMSE, etc.) and the interpretability of the model. If the performance is similar, Lasso's feature selection property might make it more appealing for interpretability and practical use.

Ultimately, the choice would also be influenced by cross-validation results on the dataset, which would indicate which model generalizes better to unseen data. If one model significantly outperforms the other in cross-validation, that would be the preferred model to apply.

Question 3

After building the model, you realized 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:

```
# Exclude the top 5 most important predictor variables identified by the Lasso model

excluded_features = lasso_important_features['Feature'].tolist()
X_train_reduced = X_train.drop(columns=excluded_features)

# Standardize the reduced data
X_train_reduced_scaled = scaler.fit_transform(X_train_reduced)

# Retrain the Lasso model with the reduced dataset
lasso_reduced = LassoCV(alphas=[0.001, 0.01, 0.1, 1.0, 10.0], cv=10)
lasso_reduced.fit(X_train_reduced_scaled, y_train)

# Extract the most important predictor variables from the retrained model
lasso_coefs_reduced = pd.DataFrame({'Feature': X_train_reduced.columns, 'Coefficient': lasso_reduced.coef_})
lasso_coefs_reduced['abs_coef'] = lasso_coefs_reduced['Coefficient'].abs()
lasso_important_features_reduced = lasso_coefs_reduced.sort_values(by='abs_coef', ascending=False).head(5)

# Output the most important features from the retrained model
print('Most important predictor variables for the reduced Lasso model:')
print(lasso_important_features_reduced[['Feature', 'Coefficient']])
```

The five most important predictor variables for the reduced Lasso model, after excluding the previously identified top predictors, are:

- Second Floor Square Feet (2ndFlrSF): 0.1963
- First Floor Square Feet (1stFlrSF): 0.1906
- Basement Quality Typical (BsmtQual_TA): -0.1873
- Basement Quality Good (BsmtQual_Gd): -0.1740
- Kitchen Quality Typical (KitchenQual_TA): -0.1716

These features are now the most significant in the Lasso model with the original top predictors removed. The coefficients indicate their relative importance and the direction of their relationship with the target variable (house price).

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?

To ensure that a model is robust and generalizable, you can follow these steps:

- Cross-Validation: Use cross-validation techniques to assess how the model performs on unseen data. K-fold cross-validation is a common method where the data is divided into 'k' subsets, and the model is trained on 'k-1' subsets and validated on the remaining subset. This process is repeated 'k' times with each subset serving as the validation set once.
- Regularization: Apply regularization techniques like Ridge (L2 regularization) or Lasso (L1 regularization) to prevent overfitting by penalizing large coefficients.
- Feature Selection: Use feature selection methods to keep only the most relevant features, reducing the risk of overfitting and improving model interpretability.
- Ensemble Methods: Combine the predictions of several models to improve robustness and reduce the variance of the predictions.
- Hyperparameter Tuning: Optimize the model's hyperparameters through grid search, random search, or Bayesian optimization methods.
- Model Complexity: Choose an appropriate level of model complexity that balances bias and variance. A model that is too complex may overfit, while one that is too simple may underfit.
- Data Preprocessing: Ensure that the data is clean and preprocessed correctly, including handling missing values, outliers, and scaling features.
- Performance Metrics: Evaluate the model using appropriate performance metrics. For regression tasks, metrics like RMSE, MAE, and R-squared are common, while for classification tasks, accuracy, precision, recall, F1-score, and AUC-ROC might be used.
- Domain Knowledge: Incorporate domain knowledge to ensure that the model makes sense in the context of the problem.
- Model Updating: Update the model regularly with new data to ensure that it adapts to changes over time.

Implications for model accuracy:

- Bias-Variance Tradeoff: A more generalizable model may have slightly lower accuracy on the training data due to reduced overfitting but should perform better on unseen data.
- Performance Metrics: The choice of performance metrics can affect how you perceive the model's accuracy. It's important to choose a metric that aligns with the business objective.
- Data Representativeness: The model's accuracy depends on how well the training data represents the real-world data it will encounter.
- Model Complexity: Simplifying the model may reduce training accuracy but can increase the generalizability of the model.

In summary, a robust and generalizable model may not always have the highest accuracy on the training data but should maintain its performance across different datasets, which is crucial for real-world applications.