Question 1

What is the optimal value of alpha for ridge and lasso regression?

The optimal value for alpha for **ridge = 500, lasso = 1000**.

Ridge Regression

```
[647]: # list of alphas to tune - if value too high it will lead to underfitting, if
       # it will not handle the overfitting
       params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1,
        0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
        4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50, 100, 500, 1000 ]}

       ridge = Ridge()

       # cross validation
       folds = 5
       model_cv = GridSearchCV(estimator = ridge,
                               param_grid = params,
                               scoring= 'neg_mean_absolute_error',
                               cv = folds,
                               return_train_score=True,
                               verbose = 1)
       model_cv.fit(X_train, y_train)
       #https://scikit-learn.org/stable/modules/model_evaluation.html

       Fitting 5 folds for each of 28 candidates, totalling 140 fits

[647]: GridSearchCV(cv=5, estimator=Ridge(),
                    param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                          0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.
                                          4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50
                                          100, 500, 1000]},
                    return_train_score=True, scoring='neg_mean_absolute_error',
                    verbose=1)

[648]: # Printing the best hyperparameter alpha
       print(model_cv.best_params_)

       {'alpha': 500}
```

Lasso

```
[In [651]]: lasso = Lasso()

            # cross validation
            model_cv = GridSearchCV(estimator = lasso,
                                    param_grid = params,
                                    scoring= 'neg_mean_absolute_error',
                                    cv = folds,
                                    return_train_score=True,
                                    verbose = 1)

            model_cv.fit(X_train, y_train)

            Fitting 5 folds for each of 28 candidates, totalling 140 fits

[Out[651]]: GridSearchCV(cv=5, estimator=Lasso(),
                         param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                               0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                               4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                               100, 500, 1000]},
                         return_train_score=True, scoring='neg_mean_absolute_error',
                         verbose=1)

[In [652]]: # Printing the best hyperparameter alpha
            print(model_cv.best_params_)

            {'alpha': 1000}
```

What will be the changes in the model if you choose double the value of alpha for both ridge and lasso?

Ridge alpha after doubling: 1000, Impact:

As the alpha is doubled the

- R2 score decreases for both train and test data
- Coefficient values decreases as well

Lasso alpha after doubling: 2000, Impact:

As the alpha is doubled the

- R2 score decreases for both train and test data
- More features are removed from the model.
- Coefficient values decreases as well

After doubling:

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	8.933738e-01	8.419925e-01	8.472154e-01
1	R2 Score (Test)	-1.748964e+21	8.350443e-01	8.436382e-01
2	RSS (Train)	6.803497e+11	1.008198e+12	9.748723e+11
3	RSS (Test)	4.929826e+33	4.649628e+11	4.407389e+11
4	MSE (Train)	2.581388e+04	3.142390e+04	3.090018e+04
5	MSE (Test)	3.354893e+15	3.258157e+04	3.172149e+04

Before doubling:

	Metric	Linear Regression	Ridge Regression	Lasso Regression
0	R2 Score (Train)	8.933738e-01	8.630618e-01	8.693258e-01
1	R2 Score (Test)	-1.748964e+21	8.535683e-01	8.592494e-01
2	RSS (Train)	6.803497e+11	8.737617e+11	8.337925e+11
3	RSS (Test)	4.929826e+33	4.127488e+11	3.967355e+11
4	MSE (Train)	2.581388e+04	2.925389e+04	2.857697e+04
5	MSE (Test)	3.354893e+15	3.069770e+04	3.009632e+04

What will be the most important predictor variables after the change is implemented?

Top 5 predictor variables:

- GrLivArea
- OverallQual_9
- OverallQual_10
- OverallQual_8
- AgeBuilt

Coefficients:

```
GrLivArea              24032.360203
OverallQual_9          14281.982795
OverallQual_10         10327.892828
OverallQual_8           9782.560054

AgeBuilt               -7998.027639
Name: Lasso, dtype: float64
```

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?

I would choose Lasso regression as it offers a distinct advantage over ridge regression by incorporating a feature selection mechanism into the model.

- While ridge regression shrunk the coefficients towards zero without eliminating any variables, lasso regression has driven certain coefficients all the way to zero, effectively performing variable selection.
- This feature makes lasso particularly advantageous in current situation where there are many predictors (> 100), as it automatically identified and excluded irrelevant or redundant variables.
- Sparsity introduced by Lasso, allows for better generalization to 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?

The 5 most important predictors now are :

- 2ndFlrSF
- BsmtQual_TA
- BsmtQual_Gd
- 1stFlrSF
- KitchenQual_TA

Coefficients:

```
2ndFlrSF              21724.658045
1stFlrSF              14702.770678

KitchenQual_TA       -13769.484907
BsmtQual_Gd          -14767.217457
BsmtQual_TA          -15046.124657
Name: Lasso, dtype: float64
```

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?

A robust and generalizable model is less prone to overfitting. Overfitting occurs when the model fits the training data excessively well but fails to perform well on new, unseen data. To avoid overfitting following steps can be followed:

- **Data Splitting**: Splitting the available data into training and testing sets is crucial. The model should be trained on the training set and evaluated on the separate testing set.
- **Cross-Validation**: By iteratively splitting the data into different training and validation sets, and averaging the results.
- **Regularization Techniques**: Utilizing regularization techniques, such as ridge or lasso regression, can enhance model robustness. Regularization adds a penalty term to the regression equation, constraining the coefficient values. This prevents overfitting by reducing the impact of individual predictors and promoting a more generalized model.
- **Evaluation metrics :** Choosing appropriate evaluation metrics, such as mean squared error (MSE) or R-squared, helps assess the model's accuracy and generalizability. These metrics quantify the difference between predicted and actual values

Impact on accuracy

By avoiding overfitting and capturing the underlying patterns in the data, a robust model is more likely to accurately predict outcomes for **new observations**, leading to higher accuracy and reliability in real-world scenarios.