Advanced Regression

Subjective Questions & Answers on Advanced Regression

Question 1

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

Let's take a look at the following result of the model cross validation.

```
model_ridge, scaler_ridge, params_ridge = build_model_with_ridge(
```

```
Best Params: {'alpha': 2.2}

R-square of train data: 0.9475163965229382
R-square of tests data: 0.9081402689819881
```

```
model_lasso, scaler_lasso, params_lasso = build_model_with_lasso
```

```
Best Params: {'alpha': 67, 'max_iter': 7500}

R-square of train data:  0.9446967650805644
R-square of tests data:  0.9100114560180644

Model Coeficients: 254
Model Coeficients != 0: 117
```

The optimal value of alpha for Ridge is 2.2, and for Lasso is 67.

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

Ridge with Double Value of Best Alpha ¶

Now let's double the value of our best alpha and see the result.

```
model_ridge_best_alpha_double, _, X_train_ridge_best_alpha_double, _, X_test_ridge_best_alpha_double, _ = build_mode
```

```
R-square of train data: 0.941798558198405
R-square of tests data: 0.9071262869453414
```

R-square of the train and test data are reduced slightly, with less than 1% difference, it shows that the model is becoming more generic with the score on the train data reduced, but no longer able to increase the test score, and reduced the test score instead.

Lasso with Double Value of Best Alpha

Now let's double the value of our best alpha and see the result.

```
model_lasso_best_alpha_double, _, X_train_lasso_best_alpha_double, _, X_test_lasso_best_alpha_double, _ = build_mode
```

```
R-square of train data:  0.9344386292068105
R-square of tests data:  0.9050395296764807

Model Coeficients: 254
Model Coeficients > 0: 92
```

R-square of the train and test data are reduced slightly, with 1% difference, a few reductions of feature being selected from 117 down to 92. It shows that the model is becoming more generic with the score on the train data reduced, but no longer able to increase the test score, and reduced the test score instead.

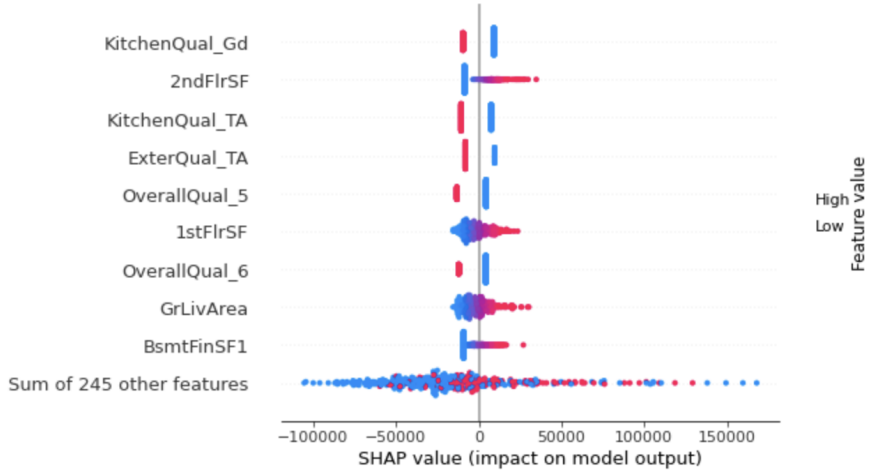
To reduce redundancy, I take the liberty of taking screenshots of the work from the notebook of this assignment, to see the result and have an experiment over it, you can go to the last section of the notebook.

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

Ridge with Double Value of Best Alpha Most Important Predictors

```
beeswarm(model_ridge_best_alpha_double, X_train_ridge_best_alpha_double, X_test_ridge_best_alpha_double)
```

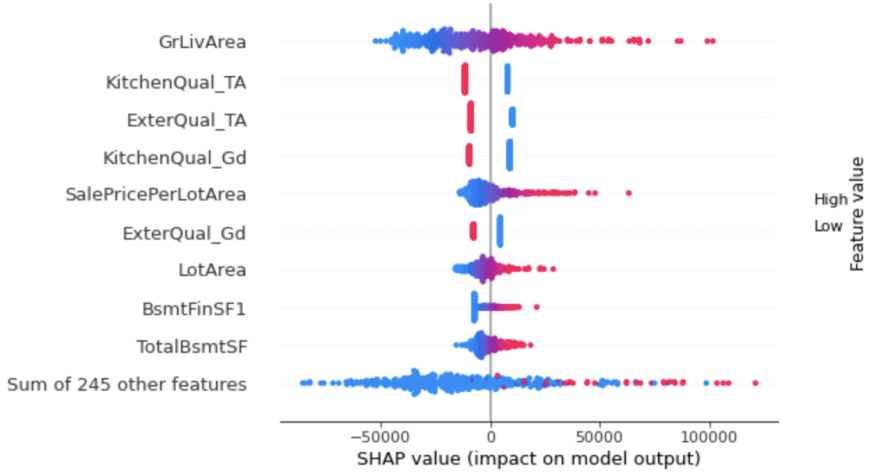
```
Permutation explainer: 398it [00:19, 10.00it/s]
```



Lasso with Double Value of Best Alpha Most Important Predictors

```
beeswarm(model_lasso_best_alpha_double, X_train_lasso_best_alpha_double, X_test_lasso_best_alpha_double)
```

```
Permutation explainer: 398it [00:20,  9.89it/s]
```



To reduce redundancy, I take the liberty of taking screenshots of the work from the notebook of this assignment, to see the result and have an experiment over it, you can go to the last section of the notebook.

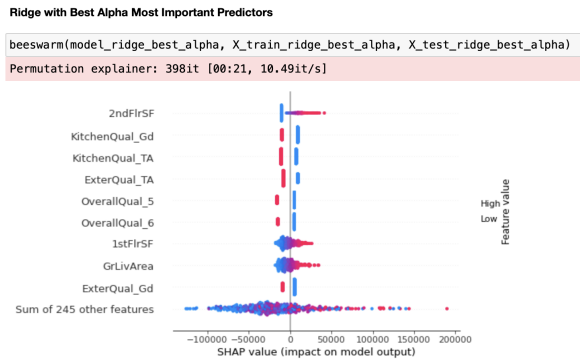
Question 2

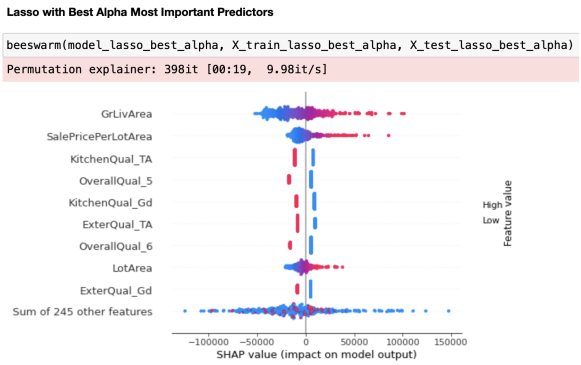
You have determined the optimal value of lambda for ridge and lasso regression during the assignment. Now, which one will you choose to apply and why?

Model with Lasso regularization will be the choice, there are 2 reasons:

1. Lasso produces less features.
2. Lasso surfaces the most important features better than Ridge.

For the second reason, let's see the following visualization of feature importance both on model with Ridge and Lasso regularization to have a better picture.





It's clear that Lasso (on the right) surfaces the most important features better than Ridge (on the left) by looking at the top 10 most important predictors.

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.

3.2.1. Lasso Model with Exclusion

Five most important variables in the lasso model are as follows:

1. GrLivArea
2. SalePricePerLotArea
3. KitchenQual
4. OverallQual
5. ExterQual

The first two columns are simple as they are numerical columns, but the other three are categorical columns that has been one-hot encoded, therefore we need to make sure we are excluding all of the one-hot encoded columns of those variables.

```
excluded = [
    'GrLivArea', 'SalePricePerLotArea',
    'KitchenQual_Ex', 'KitchenQual_Gd', 'KitchenQual_TA', 'KitchenQual_Fa', 'KitchenQual_Po',
    'OverallQual_1', 'OverallQual_2', 'OverallQual_3', 'OverallQual_4', 'OverallQual_5', 'OverallQual_6', 'OverallQu
    'ExterQual_Ex', 'ExterQual_Gd', 'ExterQual_TA', 'ExterQual_Fa', 'ExterQual_Po',
]
```

Finally let's re-build out model with the excluded columns.

```
model_lasso_excluded, scaler_lasso_excluded, params_lasso_excluded, X_train_lasso_excluded, _, X_test_lasso_excluded
```

```
Best Params: {'alpha': 67, 'max_iter': 7500}

R-square of train data:  0.9273750799031806
R-square of tests data:  0.8841631490797605

Model Coeficients: 238
Model Coeficients != 0: 122
```

To reduce redundancy, I take the liberty of taking screenshots of the work from the notebook of this assignment, to see the result and have an experiment over it, you can go to the last section of the notebook

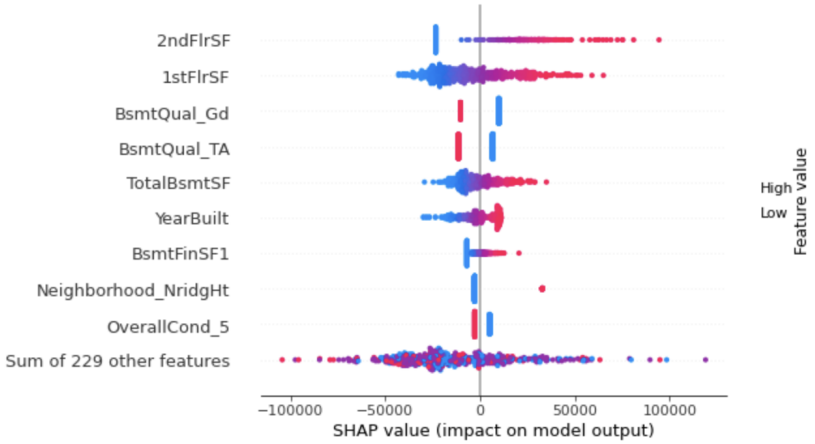
Which are the five most important predictor variables now?

3.2.2. Lasso Model with Exclusion Most Important Predictors

Now let's find out what are the most important features of the new model.

```
beeswarm(model_lasso_excluded, X_train_lasso_excluded, X_test_lasso_excluded)
```

```
Permutation explainer: 398it [00:17,  9.65it/s]
```



To summarize, the top 5 most important predictors are:

1. 2ndFlrSF
2. 1stFlrSF
3. BsmtQual
4. TotalBsmtSF
5. YearBuilt

It looks like square feet of each floor from the second, down to basement, along the the basement quality and the year being built are the top 5 most important features, and they totally makes sense.

To reduce redundancy, I take the liberty of taking screenshots of the work from the notebook of this assignment, to see the result and have an experiment over it, you can go to the last section of the notebook.

Question 4

How can you make sure that a model is robust and generalisable?

Use Regularization

```
build_model_with_multiple_linear_regression()
```

```
R-square of train data:  0.9587418236616178
R-square of tests data:  -2.7588271526530036e+19
```

```
model_lasso, scaler_lasso, params_lasso, _, _,
```

```
Best Params: {'alpha': 69, 'max_iter': 7500}

R-square of train data:  0.8498239232046165
R-square of tests data:  0.44626806492796756

Model Coeficients: 33
Model Coeficients != 0: 26
```

By using regularization, e.g. Lasso, we are generalizing our model, at the cost of the decreasing of our train score, but results in the increased test score, as you can see with multiple linear regression the test score is so bad that its worst than just drawing a horizontal line, improved to 44%.

Add Dummy Variables

1.2.4. Transform Categorical Variable to Dummy Variable

Finally, lets transform all categorical variable to dummy variable so that we can run linear regression on these variables, we also need to drop the first values of each categorical variable to reduce multicolinearity that may arise from the variables, and then join the dummies with the dataframe.

```
df_dtypes = pd.DataFrame(df.dtypes, columns=['dtype'])
df_dtypes_object = df_dtypes[df_dtypes['dtype'] == 'object']

df_dummies = pd.get_dummies(df[df_dtypes_object.index], drop_first=True)
df = pd.concat([df, df_dummies], axis=1)
```

```
model_lasso, scaler_lasso, params_lasso, _, _,
```

```
Best Params: {'alpha': 69, 'max_iter': 7500}

R-square of train data:  0.8498239232046165
R-square of tests data:  0.44626806492796756

Model Coeficients: 33
Model Coeficients != 0: 26
```

```
model_lasso, scaler_lasso, params_lasso, _, _,
```

```
Best Params: {'alpha': 66, 'max_iter': 7500}

R-square of train data:  0.9326532928698544
R-square of tests data:  0.644753340545128

Model Coeficients: 254
Model Coeficients != 0: 126
```

On the left is before adding dummy variables, and on the right is after adding dummy variables, the model performance can be increased both on the train and test data.

Add Meaningful Derived Metrics

1.3. Derived Metrics

1.3.1. Price per Square Foot

Price per square foot adds weight in the housing industry, it acts as a standard that used as a reference to the general value of a house or land without considering the overall lot and/or building area. Let's create a new derived metric from these two variables called sale price per lot area.

```
df['SalePricePerLotArea'] = df['SalePrice'] / df['LotArea']
```

```
model_lasso, scaler_lasso, params_lasso, _, _,
```

```
Best Params: {'alpha': 66, 'max_iter': 7500}

R-square of train data:  0.9326532928698544
R-square of tests data:  0.644753340545128

Model Coeficients: 254
Model Coeficients != 0: 126
```

```
model_lasso, scaler_lasso, params_lasso, _, _
```

```
Best Params: {'alpha': 62, 'max_iter': 7500}

R-square of train data:  0.9362239354126333
R-square of tests data:  0.6697364308515054

Model Coeficients: 255
Model Coeficients != 0: 131
```

On the left is before adding meaningful derived metrics, and on the right is after adding meaningful derived metrics, the model can pick up 2% increased of performance.

Remove Outliers

```
model_lasso, scaler_lasso, params_lasso, _, _
```

```
Best Params: {'alpha': 62, 'max_iter': 7500}

R-square of train data:  0.9362239354126333
R-square of tests data:  0.6697364308515054

Model Coeficients: 255
Model Coeficients != 0: 131
```

```
model_lasso, scaler_lasso, params_lasso, _, _
```

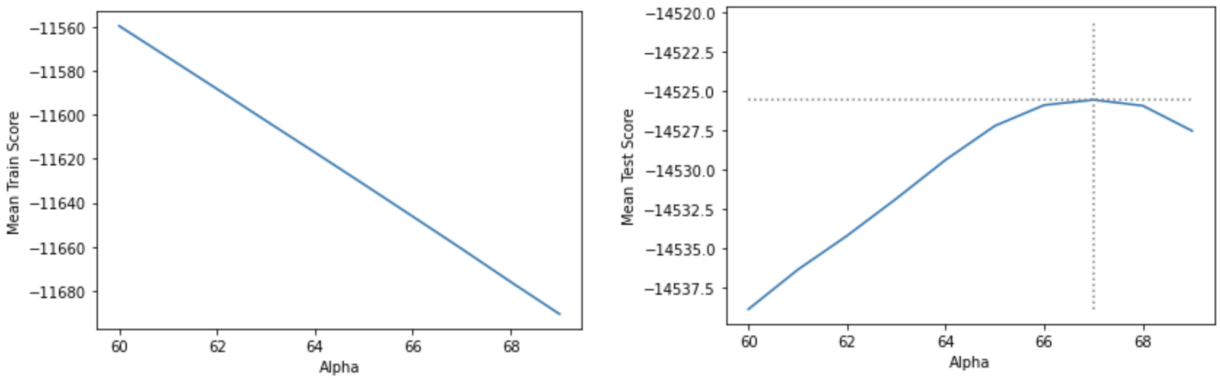
```
Best Params: {'alpha': 67, 'max_iter': 7500}

R-square of train data:  0.9446967650805644
R-square of tests data:  0.9100114560180644

Model Coeficients: 254
Model Coeficients != 0: 117
```

On the left is before outlier removal, on the right is after outlier removal, just by removing 17 rows of outliers, we can increase from 66% to 91%.

What are the implications of the same for the accuracy of the model and why?



The above screenshot is taken from the notebook. The more regularized our model, the more decreasing our train data performance, the more increasing our test data performance, but at one point the test data performance is at its peak, and decreasing gradually.

In this case, the performance of the model is at its best with alpha equal to 67, there is no point of using alpha beyond that as it will reduce the performance, and also it would require a higher iteration and eventually decrease the training time.