Advanced Regression Assignment

Problem Statement - Part II

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 values of alpha from the Python code implementation which I got are

Ridge Optimal Value: 0.2

Lasso Optimal Value: 0.0001

Ridge Code snippet:

```
[121] optimalvalue_ridge = ridge_model_cv.best_params_['alpha']
      optimalvalue_ridge

      0.2
```

Lasso Code Snippet:

```
[134] optimalvalue_lasso = model_cv.best_params_['alpha']
      optimalvalue_lasso

      0.0001
```

Changes in the model after doubling the alpha values of Ridge and Lasso from the python code implementation as shown below

```
[144] # Doubling Lasso and Ridge Regression's alpha values
      optimalvalue_ridge *= 2
      optimalvalue_lasso *= 2
      print(f"Doubled alpha values of Ridge is {optimalvalue_ridge} and Lasso is {optimalvalue_lasso}")

      Doubled alpha values of Ridge is 0.4 and Lasso is 0.0002
```

Build Ridge Regression

```
[145] alpha = optimalvalue_ridge
      ridge = Ridge(alpha=alpha)
      ridge.fit(X_train, y_train)

      Ridge(alpha=0.4)
```

Ridge Model Evaluation: The values for R2 Score, RSS and MSE scores of Train & Test which I got for Ridge Regression are

Train r2 score is: 0.8505372645597429
Test r2 score is: 0.8082647020108327
Train RSS score is: 23.67725453973616
Test RSS score is: 14.24689520303935
Train MSE score is: 0.023167568042794677
Test MSE score is: 0.032527157997806734

Build Lasso Regressiion Model

```
[149] alpha = optimalvalue_lasso
      lasso = Lasso(alpha=alpha)
      lasso.fit(X_train, y_train)

      Lasso(alpha=0.0002)
```

Lasso Regression Model Evaluation: The values for R2 Score, RSS and MSE scores of Train & Test which I got for Lasso Regression are

Train r2 score is: 0.8491303112432895
Test r2 score is: 0.8151018837857215
Train RSS score is: 23.90013813477452
Test RSS score is: 13.738858272685114
Train MSE score is: 0.023385653752225555
Test MSE score is: 0.031367256330331314

```
[156] compare_df['Ridge_Double'] = ridge.coef_
compare_df['Lasso_Double'] = lasso.coef_
compare_df.sort_values(by='Lasso', ascending=False)
```

	Linear	Ridge	Lasso	Ridge_Double	Lasso_Double
OverallQual	1.023741	1.009965	1.025145	0.996694	1.026481
LotArea	0.673270	0.614842	0.628026	0.566708	0.582639
BedroomAbvGr	0.579957	0.573897	0.571375	0.566943	0.562927
GarageCars	0.485117	0.485899	0.483241	0.486905	0.481324
SaleType_Oth	0.400300	0.327984	0.285773	0.277357	0.171004
Fireplaces	0.250853	0.256050	0.252825	0.260696	0.254720
OverallCond	0.223055	0.220537	0.218540	0.218116	0.213970
BsmtFullBath	0.210227	0.212258	0.213259	0.213317	0.216310
Neighborhood_StoneBr	0.172174	0.171149	0.166031	0.170162	0.159880
Neighborhood_NoRidge	0.149528	0.150556	0.146721	0.151576	0.143923
Exterior1st_BrkFace	0.144035	0.144224	0.142106	0.144166	0.140141
Neighborhood_Crawfor	0.144784	0.143549	0.141820	0.142433	0.138932
LandContour_Low	0.131018	0.132477	0.121974	0.132985	0.112814
SaleCondition_Alloca	0.121186	0.115403	0.097772	0.110497	0.074204
LandContour_HLS	0.079300	0.079530	0.070430	0.079496	0.061575
Neighborhood_NPkVill	0.119057	0.104706	0.066076	0.092862	0.013121
LandContour_Lvl	0.060525	0.058123	0.051754	0.055858	0.042859
BsmtQual_Fa	0.037001	0.034528	0.029474	0.032180	0.021962
MSZoning_RH	-0.002293	-0.003698	-0.000000	-0.004810	-0.000000
RoofStyle_Gable	-0.022979	-0.023677	-0.022817	-0.024400	-0.022655
BsmtQual_Gd	-0.029431	-0.028455	-0.027196	-0.027554	-0.024969

Comparison of metrics after Regularization for Double Regression

```
[158] rg_metric = pd.Series(metric_double_r, name = 'Double Ridge Regression')
ls_metric = pd.Series(metric_double_l, name = 'Double Lasso Regression')

final_metrics_double = pd.concat([final_metrics, rg_metric, ls_metric], axis = 1)

final_metrics_double
```

	Metric	Linear Regression	Ridge Regression	Lasso Regression	Double Ridge Regression	Double Lasso Regression
0	R2 Score (Train)	0.850994	0.850854	0.850528	0.850537	0.849130
1	R2 Score (Test)	0.805790	0.807432	0.811135	0.808265	0.815102
2	RSS (Train)	23.604943	23.627025	23.678794	23.677255	23.900138
3	RSS (Test)	14.430774	14.308740	14.033643	14.246895	13.738858
4	MSE (Train)	0.151976	0.152047	0.152214	0.152209	0.152924
5	MSE (Test)	0.181513	0.180744	0.178998	0.180353	0.177108

From my python implementation observed that the most important predictor variables after the changes are

	Linear	Ridge	Lasso
OverallQual	1.023741	1.009965	1.025145
LotArea	0.673270	0.614842	0.628026
BedroomAbvGr	0.579957	0.573897	0.571375
GarageCars	0.485117	0.485899	0.483241
SaleType_Oth	0.400300	0.327984	0.285773
Fireplaces	0.250853	0.256050	0.252825
OverallCond	0.223055	0.220537	0.218540
BsmtFullBath	0.210227	0.212258	0.213259
Neighborhood_StoneBr	0.172174	0.171149	0.166031
Neighborhood_NoRidge	0.149528	0.150556	0.146721

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: From my python code implementation I got the final metrics for Ridge , Lasso, Double Ridge and Double Lasso R2 Score, RSS and MSE Train & Test values are

	Metric	Linear Regression	Ridge Regression	Lasso Regression	Double Ridge Regression	Double Lasso Regression
0	R2 Score (Train)	0.850994	0.850854	0.850528	0.850537	0.849130
1	R2 Score (Test)	0.805790	0.807432	0.811135	0.808265	0.815102
2	RSS (Train)	23.604943	23.627025	23.678794	23.677255	23.900138
3	RSS (Test)	14.430774	14.308740	14.033643	14.246895	13.738858
4	MSE (Train)	0.151976	0.152047	0.152214	0.152209	0.152924
5	MSE (Test)	0.181513	0.180744	0.178998	0.180353	0.177108

As per the above data I would prefer to choose Lasso Regression over Ridge Regression.

Why because:

- The values of R2 Score, RSS and MSE for Lasso Regression are slightly better value compared to Ridge Regression
- In Lasso Regression, we can push the model co-efficients to actual zero value. This means that the features which have co-efficent of 0 can be removed from the model. Hence this results in feature selection
- Complexity of the model also reduces, because we can remove the features with zero co-efficients

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 five most important predictor variables in lasso model

```
[159] # Looking at the current top 5 important predictor variables in Lasso model
      compare_df.sort_values(by='Lasso',ascending=False).Lasso.head(5)

      OverallQual     1.025145
      LotArea         0.628026
      BedroomAbvGr    0.571375
      GarageCars      0.483241
      SaleType_Oth    0.285773
      Name: Lasso, dtype: float64
```

```
[160] top5_vars = list(compare_df['Lasso'].sort_values(ascending=False).head(5).index)
      top5_vars

      ['OverallQual', 'LotArea', 'BedroomAbvGr', 'GarageCars', 'SaleType_Oth']
```

From the python code implementation created another model excluding the above five most important predictor variables.

```python
[161] # Drop the top 5 important predictor variables from X_train and X_test
      X_train = X_train.drop(top5_vars, axis=1)
      X_test = X_test.drop(top5_vars, axis=1)
      print(X_train.shape)
      print(X_test.shape)
```

```
(1022, 26)
(438, 26)
```

```python
[162] # list of alphas to tune

      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]}
      # Applying lasso regression with 5 fold cross validation

      lasso_new = Lasso()
      folds = 5
      lasso_model_cv = GridSearchCV(estimator=lasso_new,
                              param_grid=params,
                              scoring='neg_mean_absolute_error',
                              cv=folds,
                              return_train_score=True,
                              verbose=1)
      lasso_model_cv.fit(X_train, y_train)
```

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

The five most important predictor variables after creating the new model by excluding prior 5 most important predictor variables

```python
df_lasso.sort_values(by='Lasso', ascending=False).head(5)
```

	Lasso
Fireplaces	0.606832
Neighborhood_NoRidge	0.360582
Neighborhood_StoneBr	0.277471
OverallCond	0.256558
Exterior1st_BrkFace	0.161254

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:

- A robust model has low variance. It means that an unprecedented change in one or more features does not significantly alter the value of the predicted variable.
- Similarly, a generalizable model has reduced model complexity. As the number of features increase in the model, it becomes more complex which usually leads to low bias but high variance.
- To ensure sure a model is robust and generalizable, we should take care it doesn't overfit. This is because an overfitting model has very high variance and a smallest change in data affects the model prediction heavily. Such a model will identify all the patterns of a training data but fail to pick up the patterns in unseen test data.
- A generalizable model has enough features that it has as much low variance as possible. It can be observed from the Bias-Variance trade-off visual.
- The Ordinary least squares regression model is very sensitive to outliers, and they induce the high variance. To reduce this, we can go use regularization (Ridge/Lasso) which includes a penalty term in the cost function of the model. This penalty term would move the coefficients of the model towards 0 and hence it reduces the model complexity (as adding feature is not discouraged). It reduces the overfitting in the model.
- So regularization gives us high variance with a small trade-off in bias. Hence it helps us build a model which is robust and generalizable. A robust and generalizable model will have a good, consistent train as well as test accuracy.