House Price Prediction

Assignment Questions

By-Kushagra Sengar

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?

Ans - Optimal value of alpha for ridge regression is 20.0 and optimal value of alpha for Lasso regression is 0.001.

- If we double the value of alphas for both the ridge and lasso regression , the difference between the training and testing r2 score slightly decreases.

Lasso

```
In [51]: #Building Final Lasso Model with Alpha=0.002
alpha =0.002
lasso = Lasso(alpha=alpha)
lasso.fit(X_train, y_train)

y_pred_train = lasso.predict(X_train)
y_pred_test = lasso.predict(X_test)


metric3 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print('Training r2_score : '+ str(r2_train_lr))
metric3.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print('Test r2_score : '+ str(r2_test_lr))
metric3.append(r2_test_lr)


rss1_lr = np.sum(np.square(y_train - y_pred_train))
print('Training RSS : '+ str(rss1_lr))
metric3.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print('Test RSS : '+str(rss2_lr))
metric3.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print('Training MSE : '+str(mse_train_lr))
metric3.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print('Test MSE : '+str(mse_test_lr))
metric3.append(mse_test_lr**0.5)

Training r2_score : 0.9129867336529865
Test r2_score : 0.8984483482941787
Training RSS : 88.84054494030075
Test RSS : 51.521433063483144
Training MSE : 0.08701326634701347
Test MSE : 0.11736089536100944
```

Ridge

```
In [52]: #Building final ridge model with alpha = 40.0

alpha = 40.0
ridge = Ridge(alpha=alpha)
ridge.fit(X_train, y_train)
# Lets calculate some metrics such as R2 score, RSS and RMSE
y_pred_train = ridge.predict(X_train)
y_pred_test = ridge.predict(X_test)

metric2 = []
r2_train_lr = r2_score(y_train, y_pred_train)
print('Training r2_score : ' + str(r2_train_lr))
metric2.append(r2_train_lr)

r2_test_lr = r2_score(y_test, y_pred_test)
print('Training r2_score : ' + str(r2_test_lr))
metric2.append(r2_test_lr)


rss1_lr = np.sum(np.square(y_train - y_pred_train))
print('Training RSS : '+ str(rss1_lr))
metric2.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print('Test RSS : '+str(rss2_lr))
metric2.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print('Training MSE : '+str(mse_train_lr))
metric2.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print('Test MSE : '+str(mse_test_lr))
metric2.append(mse_test_lr**0.5)

Training r2_score : 0.8729678958729985
Training r2_score : 0.8709114845537725
Training RSS : 129.69977831366856
Test RSS : 65.49204465027896
Training MSE : 0.12703210412700153
Test MSE : 0.14918461195963315
```

- Important variables before and after doubling alphas

Before

Alpha for ridge = 20.0

In [49]:
```
betas["Ridge"].sort_values()[:5]
```

Out[49]:
```
Neighborhood_Edwards    -0.211313
RoofMatl_ClyTile        -0.196451
LandContour_Bnk         -0.159451
Condition2_PosN         -0.142543
OverallCond_3           -0.125318
Name: Ridge, dtype: float64
```

In [50]:
```
betas["Ridge"].sort_values()[-5:]
```

Out[50]:
```
GrLivArea                0.206649
KitchenQual_Ex           0.212869
Neighborhood_StoneBr     0.225287
Neighborhood_NoRidge     0.262010
OverallQual_9            0.268890
Name: Ridge, dtype: float64
```

Alpha for Lasso = 0.001

In [47]:
```
betas["Lasso"].sort_values()[:5]
```

Out[47]:
```
RoofMatl_ClyTile        -6.427699
Condition2_PosN         -2.402356
OverallCond_3           -0.231336
Neighborhood_Edwards    -0.146541
AOP                     -0.135616
Name: Lasso, dtype: float64
```

In [48]:
```
betas["Lasso"].sort_values()[-5:]
```

Out[48]:
```
GrLivArea                0.322880
Neighborhood_NoRidge     0.359881
Neighborhood_StoneBr     0.427341
OverallQual_9            0.623743
OverallQual_10           1.246898
Name: Lasso, dtype: float64
```

After

Alpha for ridge = 40.0

In [54]:
```
betas['Ridge'].sort_values()[:5]
```

Out[54]:
```
Neighborhood_Edwards    -0.170775
LandContour_Bnk         -0.121371
KitchenQual_TA          -0.113190
RoofMatl_ClyTile        -0.109072
OverallQual_6           -0.100172
Name: Ridge, dtype: float64
```

In [55]:
```
betas['Ridge'].sort_values()[-5:]
```

Out[55]:
```
2ndFlrSF                 0.157890
Neighborhood_NoRidge     0.179142
KitchenQual_Ex           0.193069
GrLivArea                0.196246
OverallQual_9            0.201055
Name: Ridge, dtype: float64
```

Alpha for lasso =0.002

In [56]:
```
betas['Lasso'].sort_values()[-5:]
```

Out[56]:
```
GrLivArea                0.329891
Neighborhood_NoRidge     0.384527
Neighborhood_StoneBr     0.388688
OverallQual_9            0.585400
OverallQual_10           0.896411
Name: Lasso, dtype: float64
```

In [57]:
```
betas['Lasso'].sort_values()[:5]
```

Out[57]:
```
RoofMatl_ClyTile        -4.453120
Condition2_PosN         -1.400032
OverallCond_3           -0.178567
Neighborhood_Edwards    -0.154547
AOP                     -0.117755
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?

Ans - Thought there is no large difference between accuracies in ridge and lasso regression , but lasso regularization does feature elimination by making not so important feature as zero. Therefore, I would prefer lasso regression as it would only contain features that actually add value to the model.

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?

Ans- Five most important variable after eliminating the top 5 feature of Lasso regression are below-

```
In [63]: betas['Lasso'].sort_values()[-5:]

Out[63]: Exterior1st_BrkFace    0.240298
         KitchenQual_Ex         0.318224
         GrLivArea              0.326141
         RoofMatl_WdShngl       0.508949
         Exterior2nd_ImStucc    0.509891
         Name: Lasso, dtype: float64
```

Question 4

How can you make sure that a model is robust and generalizable? What are the implications of the same for the accuracy of the model and why?

Ans - By decreasing error/bias, we can achieve robustness and by decreasing variance we can achieve generalization. Since there is a trade-off between bias and variance therefore we need to balance the complexity of the model by balancing them using regularization.

The implication if the model is not robust and generalizable is that it might get overfit when the variance is too high and bias is low. The training accuracy is generally good in this case but testing accuracy is way low. Also, there is a possibility that model might be underfit i.e. low variance but high bias.