Assignment Part-II

Q1 . 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: The optimal value of alpha for Ridge regression is 5.0. The optimal value of alpha for lasso regression is 0.00001.

The changes that will take place in model when alpha is doubled is as follows:

Ridge

```
# Building Ridge Model by doubling the value of alpha to 10
ridge_double = Ridge(alpha=10,random_state=100)
ridge_double.fit(X_train,y_train)
ridge_double_coef = ridge_double.coef_
y_test_pred = ridge_double.predict(X_test)
print('The R2 Score of the model on the test dataset for doubled alpha is',r2_score(y_test, y_test_pred))
print('The MSE of the model on the test dataset for doubled alpha is', mean_squared_error(y_test, y_test_pred))
ridge_double_coeff = pd.DataFrame(np.atleast_2d(ridge_double_coef),columns=X_train.columns)
ridge_double_coeff = ridge_double_coeff.T
ridge_double_coeff.rename(columns={0: 'Ridge Doubled Alpha Co-Efficient'},inplace=True)
ridge_double_coeff.sort_values(by=['Ridge Doubled Alpha Co-Efficient'], ascending=False,inplace=True)
print('The most important predictor variables are as follows:')
ridge_double_coeff.head(20)
```

```
The R2 Score of the model on the test dataset for doubled alpha is 0.812919500441077
The MSE of the model on the test dataset for doubled alpha is 3.718349767624907e-05
The most important predictor variables are as follows:
```

	Ridge Doubled Alpha Co-Efficient
Neighborhood_NoRidge	0.005038
KitchenQual_Ex	0.004260
BsmtQual_Ex	0.003986
Neighborhood_Crawfor	0.003983
Neighborhood_StoneBr	0.003536
Neighborhood_NridgHt	0.003164
BsmtExposure_Gd	0.002969
BsmtFinType1_GLQ	0.002637
TotRmsAbvGrd	0.002583
Fireplaces	0.002417
Neighborhood_Somerst	0.002294
GarageFinish_RFn	0.002183
MSZoning_FV	0.002024
Exterior1st_BrkFace	0.001977
BsmtCond_Gd	0.001966
BsmtCond_TA	0.001837
BldgType_1Fam	0.001836
HouseStyle_1.5Fin	0.001769
LotConfig_CulDSac	0.001600
GarageType_Attchd	0.001386

Lasso

```
# Building Lasso Model by doubling the value of alpha to 0.0002
lasso_double = Lasso(alpha=0.00002,random_state=100)
lasso_double.fit(X_train,y_train)
lasso_double_coef = lasso_double.coef_
y_test_pred = lasso_double.predict(X_test)
print('The R2 Score of the model on the test dataset for doubled alpha is',r2_score(y_test, y_test_pred))
print('The MSE of the model on the test dataset for doubled alpha is', mean_squared_error(y_test, y_test_pred))
lasso_double_coeff = pd.DataFrame(np.atleast_2d(lasso_double_coef),columns=X_train.columns)
lasso_double_coeff = lasso_double_coeff.T
lasso_double_coeff.rename(columns={0: 'Lasso Doubled Alpha Co-Efficient'},inplace=True)
lasso_double_coeff.sort_values(by=['Lasso Doubled Alpha Co-Efficient'], ascending=False,inplace=True)
print('The most important predictor variables are as follows:')
lasso_double_coeff.head(20)
```

```
The R2 Score of the model on the test dataset for doubled alpha is 0.8145066500964101
The MSE of the model on the test dataset for doubled alpha is 3.6868041091195666e-05
The most important predictor variables are as follows:
```

	Lasso Doubled Alpha Co-Efficient
Neighborhood_NoRidge	0.006733
KitchenQual_Ex	0.006463
BsmtQual_Ex	0.006271
Neighborhood_Crawfor	0.005408
Neighborhood_StoneBr	0.005215
Neighborhood_NridgHt	0.003589
BsmtExposure_Gd	0.003463
BsmtFinType1_GLQ	0.002768
Neighborhood_Somerst	0.002764
BsmtQual_Gd	0.002716
TotRmsAbvGrd	0.002528
KitchenQual_Gd	0.002456
GarageFinish_RFn	0.002405
Fireplaces	0.002314
LotConfig_CulDSac	0.002307
BldgType_1Fam	0.002041
MSZoning_FV	0.001713
HouseStyle_1.5Fin	0.001667
Exterior1st_BrkFace	0.001587
GarageFinish_Fin	0.001491

As evident from above, when the alpha value is changed to 10 and 0.00002 respectively, R-squared value is changed to approx. 0.814.

The top predictor variables after this change is implemented are:

1. Neighbourhood
2. KitchenQual
3. Basement Qual
4. Basement Exposure
5. Basement Fin Type

Q2. 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. The R-squared value for Ridge is 0.8137 while for Lasso it is 0.81334. There is hardly any difference between the two. In the above assignment, the exploratory data analysis helped in eliminating multicollinear features. Also, a lot of variables showed very low variance across all sale prices. There were many null columns. All these were removed/handled before model building.

Since the initial analysis helped in feature selection, Ridge can be preferred over Lasso in this case.

Both reduce overfitting in the model almost in the equal amount.

Q3. 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?

Ans. Below screenshot shows the Lasso model built when top 5 predicted variables by Lasso are removed since they are not available.

The new Lasso model highlights top5 predictor variables: -

1. BaseExposure
2. Exterior1st
3. BasementFinType
4. LotConfig
5. Neighbourhood/TotalRmsAbvGrd (Since most neighbourhood variables are removed)

```
#Removing the 5 most important predictor variables from the incoming dataset
X_test_rfe2 = X_test.drop(['Neighborhood_NoRidge','KitchenQual_Ex','BsmtQual_Ex','Neighborhood_Crawfor','Neighborhood_StoneBr'],axis=1)
X_train_rfe2 = X_train.drop(['Neighborhood_NoRidge','KitchenQual_Ex','BsmtQual_Ex','Neighborhood_Crawfor','Neighborhood_StoneBr'],axis=1)
```

```
X_test_rfe2.head()
X_train_rfe2.shape
```

```
(868, 100)
```

```
# Building Lasso Model with the new dataset
lasso2 = Lasso(alpha=0.00001,random_state=100)
lasso2.fit(X_train_rfe2,y_train)
lasso2_coef = lasso2.coef_
y_test_pred = lasso2.predict(X_test_rfe2)
print('The R2 Score of the model on the test dataset is',r2_score(y_test, y_test_pred))
print('The MSE of the model on the test dataset is', mean_squared_error(y_test, y_test_pred))
lasso2_coeff = pd.DataFrame(np.atleast_2d(lasso2_coef),columns=X_train_rfe2.columns)
lasso2_coeff = lasso2_coeff.T
lasso2_coeff.rename(columns={0: 'Lasso Co-Efficient'},inplace=True)
lasso2_coeff.sort_values(by=['Lasso Co-Efficient'], ascending=False,inplace=True)
print('The most important predictor variables are as follows:')
lasso2_coeff.head(5)
```

```
The R2 Score of the model on the test dataset is 0.8044519550935129
The MSE of the model on the test dataset is 3.88664788180441e-05
The most important predictor variables are as follows:
```

	Lasso Co-Efficient
BsmtExposure_Gd	0.003273
Exterior1st_BrkFace	0.003260
BsmtFinType1_GLQ	0.003164
LotConfig_CulDSac	0.002939
Neighborhood_NridgHt	0.002729

	Lasso Co-Efficient
BsmtExposure_Gd	0.003273
Exterior1st_BrkFace	0.003260
BsmtFinType1_GLQ	0.003164
LotConfig_CulDSac	0.002939
Neighborhood_NridgHt	0.002729

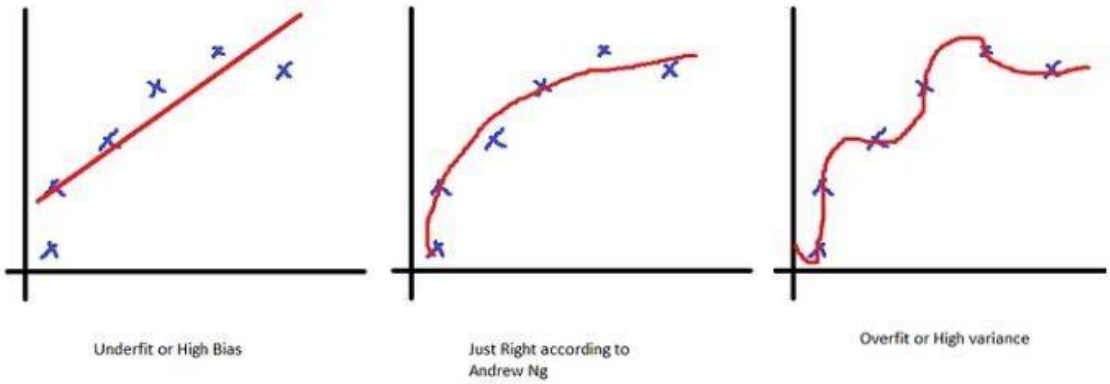
TotRmsAbvGrd	0.002675
Fireplaces	0.002581
GarageFinish_RFn	0.002134
LandContour_Low	0.002053
HouseStyle_1.5Fin	0.002002

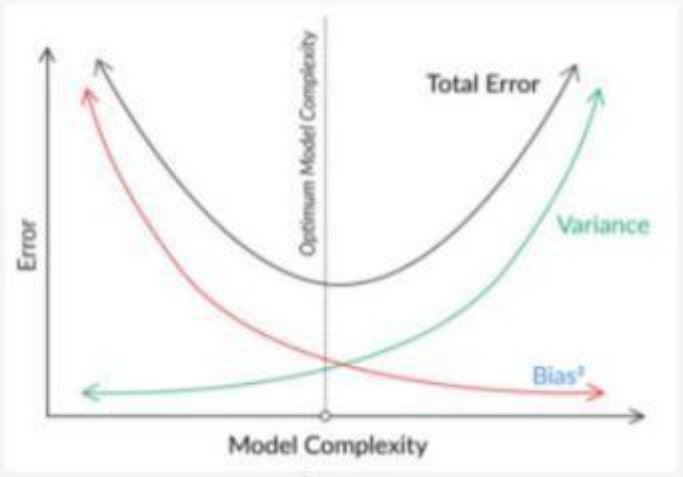
Q4. How can you make sure that a model is robust and generalisable? What are the implications of the same for the accuracy of the model and why?

Ans. We may Overfit the model or Underfit the model while training it on provided data. This can lead to inaccurate model. Besides Multicollinearity also leads to variability of coefficients.

Regularization helps us with ensuring the model is robust and generalisable. Regularization helps with managing model complexity by essentially shrinking the model coefficient estimates towards 0.

This discourages the model from becoming too complex. Below diagram explains this:





We need lowest total error, i.e., low bias and low variance, such that the model identifies all the patterns that it should and is also able to perform well with unseen data. For this, we need to manage model complexity: It should neither be too high, which would lead to overfitting, nor too low, which would lead to a model with high bias (Underfitting) that does not even identify necessary patterns in the data Implications

When we add penalty and try to get the model parameters that optimise updated cost function (RSS + Penalty), the coefficients that we get given the training data may not be the best in terms of accuracy. Although with this minor compromise in terms of bias, the variance of the model may see a marked reduction. Essentially, with regularization, we compromise by allowing a little bias for a significant gain in variance.