Advanced Regression
Assignment - 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:
Optimal value of lambda for Ridge Regression - 10
Optimal value of lambda for Lasso Regression - 0.001

```
[96]: ## Let us build the ridge regression model with double value of alpha i.e. 20
      ridge = Ridge(alpha=20)

      # Fit the model on training data
      ridge.fit(X_train, y_train)

[96]: Ridge
      Ridge(alpha=20)

[100]: # Make predictions
       y_train_pred = ridge.predict(X_train)
       y_pred = ridge.predict(X_test)

[101]: # Check metrics
       ridge_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)

       R-Squared (Train) = 0.93
       R-Squared (Test) = 0.93
       RSS (Train) = 9.37
       RSS (Test) = 2.82
       MSE (Train) = 0.01
       MSE (Test) = 0.01
       RMSE (Train) = 0.09
       RMSE (Test) = 0.10

[102]: # Now we will build the lasso model with double value of alpha i.e. 0.002
       lasso = Lasso(alpha=0.002)

       # Fit the model on training data
       lasso.fit(X_train, y_train)
```

```
[103]: # Make predictions
       y_train_pred = lasso.predict(X_train)
       y_pred = lasso.predict(X_test)

[104]: # Check metrics
       lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)

       R-Squared (Train) = 0.91
       R-Squared (Test) = 0.91
       RSS (Train) = 13.49
       RSS (Test) = 3.45
       MSE (Train) = 0.01
       MSE (Test) = 0.01
       RMSE (Train) = 0.11
       RMSE (Test) = 0.11

[105]: # Again creating a table which contain all the metrics

       lr_table = {'Metric': ['R2 Score (Train)','R2 Score (Test)','RSS (Train)','RSS (Test)',
                              'MSE (Train)','MSE (Test)', 'RMSE (Train)', 'RMSE (Test)'],
                   'Ridge Regression' : ridge_metrics,
                   'Lasso Regression' : lasso_metrics
                  }

       final_metric = pd.DataFrame(lr_table, columns = ['Metric', 'Ridge Regression', 'Lasso Regression'] )
       final_metric.set_index('Metric')
```

Metric	Ridge Regression	Lasso Regression
R2 Score (Train)	0.93	0.91
R2 Score (Test)	0.93	0.91
RSS (Train)	9.37	13.49
RSS (Test)	2.82	3.45
MSE (Train)	0.01	0.01
MSE (Test)	0.01	0.01
RMSE (Train)	0.09	0.11
RMSE (Test)	0.1	0.11

Advanced Regression
Assignment - Part II

Changes in Ridge Regression metrics:
- R2 score of train set decreased from 0.94 to 0.93
- R2 score of test set remained same at 0.93

Changes in Lasso Regression metrics:
- R2 score of train set decreased from 0.92 to 0.91
- R2 score of test set decreased from 0.93 to 0.91

```
# View the betas/coefficients
betas
```

Show 10 entries Search:

	Lasso
LotFrontage	0
LotArea	0.02
YearRemodAdd	0.03
MasVnrArea	0
BsmtFinSF1	0.03
BsmtFinSF2	0
BsmtUnfSF	0
TotalBsmtSF	0.05
1stFlrSF	0.07
2ndFlrSF	0.1

Showing 1 to 10 of 379 entries Previous 1 2 3 4 5 ... 38 Next

Now, we look at the most important predictor variables after the change is implemented.

10]:
```
# View the top 10 coefficients of Ridge regression in descending order
betas['Ridge'].sort_values(ascending=False)[:10]
```

10]:

	Ridge
GrLivArea	0.08
OverallQual_8	0.07
OverallQual_9	0.06
Neighborhood_Crawfor	0.06
Functional_Typ	0.06
Exterior1st_BrkFace	0.06
OverallCond_9	0.05
TotalBsmtSF	0.05
CentralAir_Y	0.05
OverallCond_7	0.04

11]:
```
# To interpret the ridge coefficients in terms of target, we have to take inverse log (i.e. e to the power) of betas
ridge_coeffs = np.exp(betas['Ridge'])
ridge_coeffs.sort_values(ascending=False)[:10]
```

11]:

	Ridge
GrLivArea	1.08
OverallQual_8	1.07
OverallQual_9	1.07
Neighborhood_Crawfor	1.07
Functional_Typ	1.06
Exterior1st_BrkFace	1.06
OverallCond_9	1.06
TotalBsmtSF	1.05
CentralAir_Y	1.05
OverallCond_7	1.04

Advanced Regression
Assignment - Part II

```
12]: # View the top 10 coefficients of Lasso in descending order
     betas['Lasso'].sort_values(ascending=False)[:10]
```

12]:

	Lasso
GrLivArea	0.11
OverallQual_8	0.08
OverallQual_9	0.08
Functional_Typ	0.07
Neighborhood_Crawfor	0.07
TotalBsmtSF	0.05
Exterior1st_BrkFace	0.04
CentralAir_Y	0.04
YearRemodAdd	0.04
Condition1_Norm	0.03

```
13]: # To interpret the lasso coefficients in terms of target, we have to take inverse log (i.e. 10 to the power) of betas
     lasso_coeffs = np.exp(betas['Lasso'])
     lasso_coeffs.sort_values(ascending=False)[:10]
```

13]:

	Lasso
GrLivArea	1.11
OverallQual_8	1.09
OverallQual_9	1.08
Functional_Typ	1.07
Neighborhood_Crawfor	1.07
TotalBsmtSF	1.05
Exterior1st_BrkFace	1.05
CentralAir_Y	1.04
YearRemodAdd	1.04
Condition1_Norm	1.03

So, the most important predictor variables after we double the alpha values are:

- `GrLivArea`
- `OverallQual_8`
- `OverallQual_9`
- `Functional_Typ`
- `Neighborhood_Crawfor`
- `Exterior1st_BrkFace`
- `TotalBsmtSF`
- `CentralAir_Y`

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:

The choice between Ridge and Lasso regression depends on the specific requirements of your business problem and the characteristics of your data. Ridge regression can be used when there is multicollinearity among the predictor variables, as it tends to shrink the coefficients of correlated variables towards each other; Ridge regression also avoids the risk of overfitting and complexity of the model; while Lasso regression is useful when the number of predictor variables is large and you want to identify the most important variables i.e. feature selection in the model.

Advanced Regression
Assignment - Part II

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:

Previously, top 5 Lasso predictors were:

- OverallQual_9
- GrLivArea
- OverallQual_8
- Neighborhood_Crawfor
- Exterior1st_BrkFace

```
# Top 5 Lasso predictors were:
* `OverallQual_9`, `GrLivArea`, `OverallQual_8`,`Neighborhood_Crawfor` and `Exterior1st_BrkFace`
```

```
[114]: # Create a list of top 5 lasso predictors that are to be removed
top5 = ['OverallQual_9', 'GrLivArea', 'OverallQual_8', 'Neighborhood_Crawfor', 'Exterior1st_BrkFace']
```

```
[115]: # drop them from train and test data
X_train_dropped = X_train.drop(top5, axis=1)
X_test_dropped = X_test.drop(top5, axis=1)
```

```
[116]: # Now to create a Lasso model
# we will run a cross validation on a list of alphas to find the optimum value of alpha

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]}

lasso = Lasso()

# cross validation

lassoCV = GridSearchCV(estimator = lasso,
                       param_grid = params,
                       scoring= 'neg_mean_absolute_error',
                       cv = 5,
                       return_train_score=True,
                       verbose = 1, n_jobs=-1)
lassoCV.fit(X_train_dropped, y_train)
```

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

[116]: GridSearchCV
▸ estimator: Lasso
▸ Lasso

```
[117]: # View the optimal value of alpha
lassoCV.best_params_
```

```
[117]: {'alpha': 0.001}
```

Thus, we get optimum value of alpha as 0.001. Now we will build a lasso regression model using this value.

```
[118]: # Create a lasso instance with optimum value alpha=0.001
lasso = Lasso(alpha=0.001)
```

```
[119]: # Fit the model on training data
lasso.fit(X_train_dropped, y_train)
```

[119]: Lasso
Lasso(alpha=0.001)

```
[120]: # Make predictions
y_train_pred = lasso.predict(X_train_dropped)
y_pred = lasso.predict(X_test_dropped)
```

```
[121]: # Check metrics
lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)
```

```
R-Squared (Train) = 0.91
R-Squared (Test) = 0.92
RSS (Train) = 12.75
RSS (Test) = 3.02
MSE (Train) = 0.01
MSE (Test) = 0.01
RMSE (Train) = 0.10
RMSE (Test) = 0.10
```

Now, we will find the top 5 predictors

```
[122]: # Creating a table which contain all the metrics

lr_table = {'Metric': ['R2 Score (Train)','R2 Score (Test)','RSS (Train)','RSS (Test)',
                       'MSE (Train)','MSE (Test)', 'RMSE (Train)', 'RMSE (Test)'],
            'Lasso Regression' : lasso_metrics
        }

final_metric = pd.DataFrame(lr_table, columns = ['Metric', 'Lasso Regression'] )
final_metric.set_index('Metric')
```

Advanced Regression
Assignment - Part II

[122]:

Lasso Regression	
Metric	
R2 Score (Train)	0.91
R2 Score (Test)	0.92
RSS (Train)	12.75
RSS (Test)	3.02
MSE (Train)	0.01
MSE (Test)	0.01
RMSE (Train)	0.1
RMSE (Test)	0.1

[123]:
```
# Now we see the changes in coefficients after regularization

# First create empty datafame with all the independent variables as indices
betas = pd.DataFrame(index=X_train_dropped.columns)
betas.rows = X_train_dropped.columns
betas
```

[124]:
```
# Now fill in the values of betas, one column for ridge coefficients and one for lasso coefficients
betas['Lasso'] = lasso.coef_
# betas['Lasso'] = lasso.coef_
```

[125]:
```
# View the betas/coefficients
betas
```

[125]: Show 10 entries Search:

	Lasso
LotFrontage	0
LotArea	0.02
YearRemodAdd	0.03
MasVnrArea	0
BsmtFinSF1	0.03
BsmtFinSF2	0
BsmtUnfSF	0
TotalBsmtSF	0.05
1stFlrSF	0.07
2ndFlrSF	0.1

Now, we will look at the top 5 features significant in predicting the value of a house according to the new lasso model

[126]:
```
# View the top 5 coefficients of Lasso in descending order
betas['Lasso'].sort_values(ascending=False)[:5]
```

[126]:

	Lasso
2ndFlrSF	0.1
Functional_Typ	0.07
1stFlrSF	0.07
MSSubClass_70	0.06
Neighborhood_Somerst	0.06

After dropping our top 5 lasso predictors, we get the following new top 5 predictors:

- `2ndFlrSF`
- `Functional_Typ`
- `1stFlrSF`
- `MSSubClass_70`
- `Neighborhood_Somerst`

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:

1) **Splitting data into training & validation sets:** This helps provide accurate estimate of the model's performance by evaluating on multiple subsets of the data. Thereafter, reducing the variance in the model's predictions.

2) **Regularization:** Using Ridge/ Lasso regression helps prevent overfitting and also improve model's generalization.
3) **Feature Selection Engineering:** Mindful selection of model features is important while making model predictions.
4) **Hyperparameter Tuning:** Using hyperparameter tuning techniques such as Grid Search can improve predictions on unseen data.

These are some ways to implement to make a model generalized for real-world scenarios.