

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 = 07**
- **Optimal value of lambda for Lasso = 0.0001**

Let us build the ridge regression model with double value of alpha i.e. 20

```
ridge = Ridge(alpha=14)
```

Fit the model on training data

```
ridge.fit(X_train, y_train)
```

Make predictions

```
y_train_pred = ridge.predict(X_train)
```

```
y_pred = ridge.predict(X_test)
```

Check metrics

```
ridge_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)
```

Now we will build the lasso model with double value of alpha i.e. 0.002

```
lasso = Lasso(alpha=0.0002)
```

Fit the model on training data

```
lasso.fit(X_train, y_train)
```

Make predictions

```
y_train_pred = lasso.predict(X_train)
```

```
y_pred = lasso.predict(X_test)
```

Check metrics

```
lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)
```

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')
```

Changes in Ridge Regression metrics:

- ✓ **R2 score of train set decreased from 0.95 to 0.94**
- ✓ **R2 score of test set same 0.90**

Changes in Lasso metrics:

- ✓ **R2 score of train set decreased from 0.96 to 0.95**
- ✓ **R2 score of test set increase from 0.85 to 0.87**

Now we see the changes in coefficients after regularization.

First create empty dataframe with all the independent variables as indices.

```
betas = pd.DataFrame(index=X.columns)  
  
betas.rows = X.columns
```

Now fill in the values of betas, one column for ridge coefficients and one for lasso coefficients

```
betas['Ridge'] = ridge.coef_  
betas['Lasso'] = lasso.coef_
```

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

View the top 10 coefficients of Ridge regression in descending order

```
betas['Ridge'].sort_values(ascending=False)[:10]
```

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]
```

View the top 10 coefficients of Lasso in descending order

```
betas['Lasso'].sort_values(ascending=False)[:10]
```

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]
```

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

- ✓ GrLivArea
- ✓ OverallQual_8
- ✓ OverallQual_9
- ✓ Neighborhood_Crawfor
- ✓ OverallCond_9
- ✓ Exterior1st_BrkFace

Question 2

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

Answer

- The model we will choose to apply will depend on the use case.
- If we have too many variables and one of our primary goals is feature selection, then we will use **Lasso**.
- If we don't want to get too large coefficients and reduction of coefficient magnitude is one of our prime goals, then we will use **Ridge Regression**.

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

Here, we will drop the top 5 features in Lasso model and build the model again.

Top 5 Lasso predictors were:

- ✓ OverallCond_9
- ✓ OverallQual_9,
- ✓ Condition2_PosA
- ✓ OverallQual_8
- ✓ Neighborhood_Crawfor

Create a list of top 5 lasso predictors that are to be removed

```
top5 = ['OverallCond_9', 'OverallQual_9', 'Condition2_PosA', 'OverallQual_8',  
        'Neighborhood_Crawfor']
```

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)
```

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

View the optimal value of alpha

```
lassoCV.best_params_
```

Thus, we get optimum value of alpha as 0.0001. Now we will build a lasso regression model using this value.

Create a lasso instance with optimum value alpha=0.0001

```
lasso = Lasso(alpha=0.0001)
```

Fit the model on training data

```
lasso.fit(X_train_dropped, y_train)
```

Make predictions

```

y_train_pred = lasso.predict(X_train_dropped)
y_pred = lasso.predict(X_test_dropped)

```

Check metrics

```
lasso_metrics = show_metrics(y_train, y_train_pred, y_test, y_pred)
```

Now, we will find the top 5 predictors

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')

```

Now we see the changes in coefficients after regularization

First create empty dataframe with all the independent variables as indices

```

betas = pd.DataFrame(index=X_train_dropped.columns)
betas.rows = X_train_dropped.columns

```

Now fill in the values of betas, one column for ridge coefficients and one for lasso coefficients

```
betas['Lasso'] = lasso.coef_
```

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

```
betas['Lasso'].sort_values(ascending=False)[:5]
```

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

- ✓ SaleType_ConLD
- ✓ SaleCondition_Alloca
- ✓ GrLivArea
- ✓ Exterior1st_BrkFace
- ✓ MSZoning_FV

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 model is **robust** when any variation in the data does not affect its performance much.
- A **generalizable** model is able to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model.
- To make sure a model is robust and generalizable, we have to **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.
- In other words, the model should not be too complex in order to be robust and generalizable.

- If we look at it from the perspective of **Accuracy**, a too complex model will have a very high accuracy. So, to make our model more robust and generalizable, we will have to decrease variance which will lead to some bias. Addition of bias means that accuracy will decrease.
- In general, we have to find strike some balance between model accuracy and complexity. This can be achieved by Regularization techniques like Ridge Regression and Lasso.