Assignment 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:

Considering the scoring= 'neg_mean_absolute_error' for GridSearch, The best value for alpha is below:

Lasso:

The best alpha value – 0.00014

```
In [802]: # Printing the best hyperparameter alpha
          print(model_cv.best_params_)

          {'alpha': 0.00014}
```

Ridge:

The best alpha value is 7.5

```
In [797]: # Printing the best hyperparameter alpha
          print(model_cv.best_params_)

          {'alpha': 7.5}
```

Doubling the alpha value:

Lasso: The Best 5 Predictors are

(-0.361, 'Condition2_PosN'),

(0.305, 'GrLivArea'),

(0.173, 'OverallQual_10'),

(0.099, 'OverallQual_9'),

(0.086, 'TotalBsmtSF')

```
In [836]: alpha_lasso = 0.00028
          lasso = Lasso(alpha=alpha_lasso)

          lasso.fit(X_train, y_train)

Out[836]: Lasso(alpha=0.00028)
```

```
In [837]: # Lasso model parameters
          model_parameters = list(lasso.coef_)
          model_parameters.insert(0, lasso.intercept_)
          model_parameters = [round(x, 3) for x in model_parameters]
          cols = X_train.columns
          cols = cols.insert(0, "constant")
          lasso_selected_features = []
          for k,v in sorted(list(zip(model_parameters,cols)), key=lambda x:abs(x[0]), reverse=True)[1:31]:
              lasso_selected_features.append(v)
          sorted(list(zip(model_parameters,cols)), key=lambda x:abs(x[0]), reverse=True)

Out[837]: [(-0.361, 'Condition2_PosN'),
           (0.305, 'GrLivArea'),
           (0.173, 'OverallQual_10'),
           (0.099, 'OverallQual_9'),
           (0.086, 'TotalBsmtSF'),
```

After doubling the alpha, more penalization is applied during Train and R2 train has come down. Overfitting is more addressed. Below is metrices after doubling the alpha.

R2_Train - 0.9243091889231924

R2_Test - 0.8764023843296719

RSS Train - 0.9408069488445079

RSS Test - 0.6000984003102732

MSE Train - 0.0009830793613840208

MSE Test - 0.001460093431411857

Ridge: -

After doubling the alpha for Ridge model, the best Predictors are:

(0.055, 'OverallQual_10'),

(0.053, 'constant'),

(0.053, 'GrLivArea'),

(0.049, '1stFlrSF'),

(0.046, 'Neighborhood_NoRidge'),

(0.045, 'TotalBsmtSF'),

(0.044, '2ndFlrSF'),

(0.043, 'BsmtFinSF1')

Ridge Regression Model after doubling alpha

```
alpha_ridge = 15
ridge = Ridge(alpha=alpha_ridge)

ridge.fit(X_train, y_train)
```

```
Ridge(alpha=15)
```

```
# ridge model parameters
model_parameters = list(ridge.coef_)
model_parameters.insert(0, ridge.intercept_)
model_parameters = [round(x, 3) for x in model_parameters]
cols = X_train.columns
cols = cols.insert(0, "constant")
sorted(list(zip(model_parameters,cols)), key=lambda x:abs(x[0]), reverse=True)
```

```
[(0.055, 'OverallQual_10'),
 (0.053, 'constant'),
 (0.053, 'GrLivArea'),
 (0.049, '1stFlrSF'),
 (0.046, 'Neighborhood_NoRidge'),
 (0.045, 'TotalBsmtSF'),
 (0.044, '2ndFlrSF'),
 (0.043, 'BsmtFinSF1'),
```

After doubling the alpha, model is more penalized and metrices have changes:

```
R2_Train - 0.8984072275834828
R2_Test - 0.8939427045591484
RSS Train - 1.2627581192761772
RSS Test - 0.5149356076985248
MSE Train - 0.001319496467373226
MSE Test - 0.0012528846902640505
```

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: -

After tuning alpha parameter for Lasso little bit more, I am able to increase the R2 score for the test records. Below the score for Lasso after tuning:

```
R2_Train - 0.915100206570713
R2_Test - 0.8835588480797233
RSS Train - 1.055270969849743
RSS Test - 0.565351917338148
MSE Train - 0.0011026864888712047
MSE Test - 0.0013755521103117957
```

Ridge:

```
R2_Train - 0.9129532624930286
R2_Test - 0.896243593968991
RSS Train - 1.0819566385370085
RSS Test - 0.5037641943452096
MSE Train - 0.0011305712001431645
MSE Test - 0.0012257036358764223
```

Since the R2 score for Lasso and Ridge are almost same. I would go for Lasso. As, we have more than 250 parameters in the model, to explain the business use case feature selection is required.

Lasso would be more inferential.

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?

Answer:

We can refer the Jupyter notebook file for more details. But, the is not big impact of removing the columns. The R2 score on both test and train data is almost similar to the earlier model created with 5 important variables.

In [865]:
```
params = {'alpha': [0.00005,0.0001, 0.00011, 0.00013, 0.00014, 0.00015, 0.00016, 0.0002, 0.0003, 0.0004,
                    0.0006, 0.0007, 0.0008, 0.0009, 0.01, 0.05]}
lasso = Lasso()
# cross validation
model_cv = GridSearchCV(estimator = lasso,
                        param_grid = params,
                        scoring= 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score=True,
                        verbose = 1)
model_cv.fit(X_train[X_train_cols], y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
```

Out[865]:
```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [5e-05, 0.0001, 0.00011, 0.00013, 0.00014,
                                   0.00015, 0.00016, 0.0002, 0.0003, 0.0004,
                                   0.0006, 0.0007, 0.0008, 0.0009, 0.01,
                                   0.05]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

In [866]:
```
# Printing the best hyperparameter alpha
print(model_cv.best_params_)
```

```
{'alpha': 0.00013}
```

```
r2_test_lr = r2_score(y_test, y_pred_test)
print(f'R2_Test - {r2_test_lr}')
metric3.append(r2_test_lr)

rss1_lr = np.sum(np.square(y_train - y_pred_train))
print(f'RSS Train - {rss1_lr}')
metric3.append(rss1_lr)

rss2_lr = np.sum(np.square(y_test - y_pred_test))
print(f'RSS Test - {rss2_lr}')
metric3.append(rss2_lr)

mse_train_lr = mean_squared_error(y_train, y_pred_train)
print(f'MSE Train - {mse_train_lr}')
metric3.append(mse_train_lr**0.5)

mse_test_lr = mean_squared_error(y_test, y_pred_test)
print(f'MSE Test - {mse_test_lr}')
metric3.append(mse_test_lr**0.5)
```

```
R2_Train - 0.9126170627247462
R2_Test - 0.8949625836613352
RSS Train - 1.086135469146698
RSS Test - 0.5099838308020781
MSE Train - 0.0011349377943016697
MSE Test - 0.0012408365712945938
```

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?

We can say that the Ridge and Lasso model created are robust and generalized because:

1. The R2 score for both test and train data set is quite close.
2. With applying with have penalized more complex model and generalized it to get good accuracy on Test and Train data both.
3. If you will see that when we create a model using Lasso Columns, then all the assumption of Linear regression are met.

In [827]:
```
# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((y_train - y_train_pred), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)                  # Plot heading
plt.xlabel('Errors', fontsize = 18)                         # X-label
plt.show()
```