

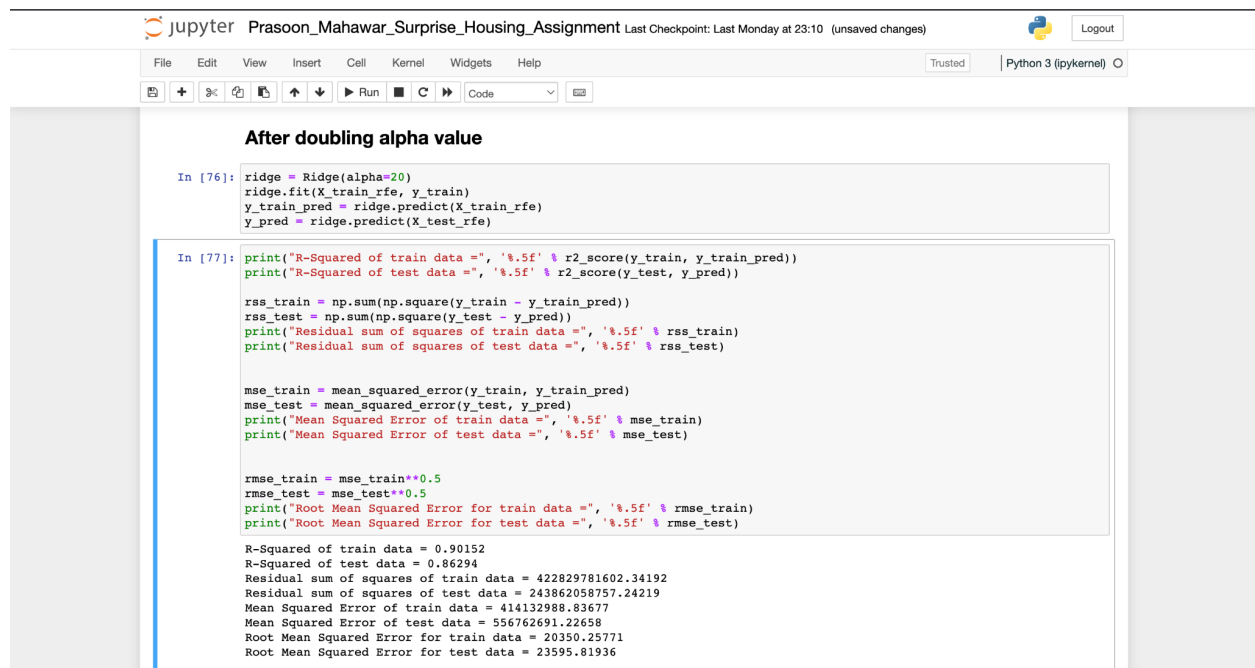
Question 1

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 1:

- Optimal alpha value for ridge regression is 10
- Optimal alpha value for lasso regression is 7

After doubling the alpha value for ridge



The screenshot shows a Jupyter Notebook titled "Prasoon_Mahawar_Surprise_Housing_Assignment". The interface includes a top bar with the Jupyter logo, the notebook title, and a "Logout" button. Below the top bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar with icons for file operations and execution is also present. The main area displays two code cells. The first cell, labeled "In [76]:", defines a Ridge model with alpha=20 and fits it to training data. The second cell, labeled "In [77]:", prints various performance metrics for both training and test data. The output of the second cell shows that the R-squared values remain unchanged at 0.90152 for training and 0.86294 for testing, while the residual sum of squares, mean squared error, and root mean squared error all increase compared to the previous state.

```
After doubling alpha value

In [76]: ridge = Ridge(alpha=20)
         ridge.fit(X_train_rfe, y_train)
         y_train_pred = ridge.predict(X_train_rfe)
         y_pred = ridge.predict(X_test_rfe)

In [77]: print("R-Squared of train data =", '%.5f' % r2_score(y_train, y_train_pred))
         print("R-Squared of test data =", '%.5f' % r2_score(y_test, y_pred))

         rss_train = np.sum(np.square(y_train - y_train_pred))
         rss_test = np.sum(np.square(y_test - y_pred))
         print("Residual sum of squares of train data =", '%.5f' % rss_train)
         print("Residual sum of squares of test data =", '%.5f' % rss_test)

         mse_train = mean_squared_error(y_train, y_train_pred)
         mse_test = mean_squared_error(y_test, y_pred)
         print("Mean Squared Error of train data =", '%.5f' % mse_train)
         print("Mean Squared Error of test data =", '%.5f' % mse_test)

         rmse_train = mse_train**0.5
         rmse_test = mse_test**0.5
         print("Root Mean Squared Error for train data =", '%.5f' % rmse_train)
         print("Root Mean Squared Error for test data =", '%.5f' % rmse_test)

R-Squared of train data = 0.90152
R-Squared of test data = 0.86294
Residual sum of squares of train data = 422829781602.34192
Residual sum of squares of test data = 243862058757.24219
Mean Squared Error of train data = 414132988.83677
Mean Squared Error of test data = 556762691.22658
Root Mean Squared Error for train data = 20350.25771
Root Mean Squared Error for test data = 23595.81936
```

Changes in ridge regression:

- R2 score of train set remained same at 0.90
- R2 score of test set remained same at 0.86

After doubling the alpha value for lasso

jupyter Prasoon_Mahawar_Surprise_Housing_Assignment Last Checkpoint: Last Monday at 23:10 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (pykernel) O

```

In [78]: lasso = Lasso(alpha=14.0)
         lasso.fit(X_train_rfe, y_train)
         y_train_pred_lasso = lasso.predict(X_train_rfe)
         y_pred_lasso = lasso.predict(X_test_rfe)

In [79]: print("R-Squared of train data =", '%.5f' % r2_score(y_train, y_train_pred_lasso))
         print("R-Squared of test data =", '%.5f' % r2_score(y_test, y_pred_lasso))

         rss_train_lasso = np.sum(np.square(y_train - y_train_pred_lasso))
         rss_test_lasso = np.sum(np.square(y_test - y_pred_lasso))
         print("Residual sum of squares of train data =", '%.5f' % rss_train_lasso)
         print("Residual sum of squares of test data =", '%.5f' % rss_test_lasso)

         mse_train_lasso = mean_squared_error(y_train, y_train_pred_lasso)
         mse_test_lasso = mean_squared_error(y_test, y_pred_lasso)
         print("Mean Squared Error of train data =", '%.5f' % mse_train_lasso)
         print("Mean Squared Error of test data =", '%.5f' % mse_test_lasso)

         rmse_train_lasso = mse_train**0.5
         rmse_test_lasso = mse_test**0.5
         print("Root Mean Squared Error for train data =", '%.5f' % rmse_train_lasso)
         print("Root Mean Squared Error for test data =", '%.5f' % rmse_test_lasso)

R-Squared of train data = 0.92243
R-Squared of test data = 0.81001
Residual sum of squares of train data = 333069517184.83844
Residual sum of squares of test data = 338031724719.94019
Mean Squared Error of train data = 326218919.86762
Mean Squared Error of test data = 771761928.58434
Root Mean Squared Error for train data = 20350.25771
Root Mean Squared Error for test data = 23595.81936

```

Changes in lasso regression:

- R2 score of train set remained same at 0.92
- R2 score of test set increased from at 0.80 to 0.81

Most important predictor variables after doubling the alpha value

For Ridge:- **GrLivArea, OverallQual_9, OverallQual_8, Neighborhood_Crawfor, BsmtExposure_Gd**

For Lasso:- **Condition2_PosA, OverallQual_9, Neighborhood_Crawfor, OverallCond_9, GrLivArea**

jupyter Prasoon_Mahawar_Surprise_Housing_Assignment Last Checkpoint: Last Monday at 23:10 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (pykernel) O

```

GarageYrBlt_1939 4666.025703 15351.482646
GarageYrBlt_2008 8723.078410 14871.609116
GarageQual_Gd 7634.446916 17204.771705
PoolQC_Gd -7158.981754 -159515.768581
PoolQC_NA 4698.968166 -1297.984011

In [83]: # Viewing new top 5 for ridge
         comparison['Ridge'].sort_values(ascending=False)[:5]

Out[83]: GrLivArea 26404.591545
         OverallQual_9 23319.311714
         OverallQual_8 20115.877460
         Neighborhood_Crawfor 18678.351046
         BsmtExposure_Gd 17932.729893
         Name: Ridge, dtype: float64

In [84]: # Viewing new top 5 for lasso
         comparison['Lasso'].sort_values(ascending=False)[:5]

Out[84]: Condition2_PosA 50140.234437
         OverallQual_9 35623.089637
         Neighborhood_Crawfor 30092.183000
         OverallCond_9 29225.553420
         GrLivArea 25375.627157
         Name: Lasso, dtype: float64

In [ ]:

```

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 2:

- Lasso regression is used when there are too many variables and we want to select features from it.
- Ridge regression is used when we want to reduce the coefficient magnitude.



So as per the problem statement we need to tell variables are significant in predicting the price of a house and also the R² score for lasso is less than ridge so we will apply Lasso regression.

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 3

After performing the analysis as shown in the image below after removing the top five predictor variables from the previous lasso model


Prasoon_Mahawar_Surprise_Housing_Assignment
Last Checkpoint: Last Monday at 23:10 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Python 3 (pykernel)

```

In [72]: # For question 3

# Removing the top 5 variables of lasso

top5 = ['Condition2_PosA', 'OverallQual_9', 'Neighborhood_Crawfor', 'OverallCond_9', 'GrLivArea']

X_train_dropped = X_train_rfe.drop(top5, axis=1)
X_test_dropped = X_test_rfe.drop(top5, axis=1)

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

/Users/prasoonmahawar/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Conv
ergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of t
he features or consider increasing regularisation. Duality gap: 2.667e+09, tolerance: 3.477e+08
  model = cd_fast.enet_coordinate_descent(
/Users/prasoonmahawar/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Conv
ergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of t
he features or consider increasing regularisation. Duality gap: 2.461e+09, tolerance: 3.466e+08
  model = cd_fast.enet_coordinate_descent(
/Users/prasoonmahawar/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Conv
ergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of t
he features or consider increasing regularisation. Duality gap: 2.206e+09, tolerance: 3.435e+08
  model = cd_fast.enet_coordinate_descent(
/Users/prasoonmahawar/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Conv
ergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of t
he features or consider increasing regularisation. Duality gap: 2.013e+09, tolerance: 3.419e+08
  model = cd_fast.enet_coordinate_descent(

Out[72]: GridSearchCV(cv=5, estimator=Lasso(), n_jobs=-1,
                    param_grid={'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]},
                    return_train_score=True, scoring='neg_mean_absolute_error',
                    verbose=1)

In [73]: lassoCV.best_params_

Out[73]: {'alpha': 20}

In [74]: lasso = Lasso(alpha=20)
lasso.fit(X_train_dropped, y_train)

## Make predictions
y_train_pred = lasso.predict(X_train_dropped)
y_pred = lasso.predict(X_test_dropped)

In [75]: betas = pd.DataFrame(index=X_train_dropped.columns)
betas.rows = X_train_dropped.columns
betas['Lasso'] = lasso.coef_
betas['Lasso'].sort_values(ascending=False)[:5]

Out[75]: Neighborhood_NoRidge    50954.883619
GarageQual_Gd                26650.565921
Neighborhood_ClearCr         23224.704672
OverallQual_8                 21806.094355
GarageYrBlt_1939             19089.118391
Name: Lasso, dtype: float64

```

The new five most important variable will be:

Neighborhood_NoRidge, GarageQual_Gd, Neighborhood_ClearCr, OverallQual_8, GarageYrBlt_1939

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 4

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 so the model should not be too complex in order to be robust and generalizable.

Now a 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 get a tra between model accuracy and complexity. This can be achieved by Regularization techniques like Ridge Regression and Lasso.