2.Bias Variance for Linear Lasso and Ridge Regresssion

October 28, 2021

Bias Variance for Linear, Ridge, and Lasso Regression

```
[28]: # Import necessary package
      import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import seaborn as sns
```

0.0.1 Step 1: Load the dataset

```
[29]: # Load dataset into pandas dataframe
      df=pd.read_csv("E:\\MY LECTURES\\DATA SCIENCE\\3.Programs\\dataset\\Housing.
       ↪csv")
      # Change this location based on the location of dataset in your machine
```

```
[30]: # Display the first five records
      df.head()
```

```
[30]:       price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
      0  13300000  7420         4          2        3      yes        no       no
      1  12250000  8960         4          4        4      yes        no       no
      2  12250000  9960         3          2        2      yes        no      yes
      3  12215000  7500         4          2        2      yes        no      yes
      4  11410000  7420         4          1        2      yes       yes      yes

        hotwaterheating airconditioning  parking prefarea furnishingstatus
      0              no             yes        2      yes        furnished
      1              no             yes        3       no        furnished
      2              no              no        2      yes   semi-furnished
      3              no             yes        3      yes        furnished
      4              no             yes        2       no        furnished
```

```
[31]: # Dataset shape (number of rows and columns)
      df.shape
```

```
[31]: (545, 13)
```

0.0.2 Step 2: Apply EDA

You may apply univariate and bivariate analysis

0.0.3 Step 3. Pre-process and extract the features

```
[32]: temp = df[['price','area','mainroad']]
```

```
[33]: temp.head()
```

```
[33]:       price  area mainroad
      0  13300000  7420      yes
      1  12250000  8960      yes
      2  12250000  9960      yes
      3  12215000  7500      yes
      4  11410000  7420      yes
```

One hot encoding - replacing categorical values with numerical number - preprocessing technique

```
[34]: temp = pd.get_dummies(temp, drop_first=True)
      temp.head()
```

```
[34]:       price  area  mainroad_yes
      0  13300000  7420             1
      1  12250000  8960             1
      2  12250000  9960             1
      3  12215000  7500             1
      4  11410000  7420             1
```

```
[35]: X = temp.drop('price', axis=1)
      Y = temp['price']
```

input feature independent feature or predictor feature. All features except Price. output feature dependent feature or response feature or target feature. Price feature.

0.0.4 Step 4. Split the data for training and testing

```
[36]: # Splitting dataset into training and testing set
      from sklearn.model_selection import train_test_split
      x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2,␣
       ↪random_state = 2)
```

0.0.5 Step 5: Training phase (bulding the model)

1. Multiple Linear regression

[37]:
```
# Fitting line on two dimension on the training set
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train, y_train)
```

[37]:
```
LinearRegression()
```

[38]:
```
# R2 score for training data
linear_train_R2 = model.score(x_train, y_train)
linear_train_R2
```

[38]:
```
0.3166619989728189
```

[39]:
```
# R2 score for testing data
linear_test_R2 = model.score(x_test, y_test)
linear_test_R2
```

[39]:
```
0.27023189107167933
```

[40]:
```
# Bias and Variance
#!pip install mlxtend
from mlxtend.evaluate import bias_variance_decomp
Linear_mse, Linear_bias, Linear_variance =  bias_variance_decomp(model, x_train.
 ↪values, y_train.values, x_test.values, y_test.values, loss='mse',␣
 ↪random_seed=123, num_rounds=100)
```

[41]:
```
# Display Bias and Variance
print("Mean Squared Error: ", round(Linear_mse, 4))
print("Bias: ",round(Linear_bias, 4))
print("Variance: ",round(Linear_variance, 4))
```

```
Mean Squared Error:  2615079328367.502
Bias:  2593444760309.601
Variance:  21634568057.9013
```

2. Lasso (L1) regression

[42]:
```
from sklearn import linear_model
lasso_model = linear_model.Lasso(alpha=50, max_iter=100, tol=0.1)
lasso_model.fit(x_train, y_train)
```

[42]:
```
Lasso(alpha=50, max_iter=100, tol=0.1)
```

```
[43]: # R2 score for training data
      lasso_train_R2 = lasso_model.score(x_train, y_train)
      lasso_train_R2
```

```
[43]: 0.3166619925369154
```

```
[44]: # R2 score for testing data
      lasso_test_R2 = lasso_model.score(x_test, y_test)
      lasso_test_R2
```

```
[44]: 0.27020772784231295
```

```
[45]: # Bias and Variance
      Lasso_mse, Lasso_bias, Lasso_variance =  bias_variance_decomp(lasso_model,␣
       ↪x_train.values, y_train.values, x_test.values, y_test.values, loss='mse',␣
       ↪random_seed=3, num_rounds=100)
```

```
[46]: # Display Bias and Variance
      print("Mean Squared Error: ", round(Lasso_mse, 4))
      print("Bias: ",round(Lasso_bias, 4))
      print("Variance: ",round(Lasso_variance, 4))
```

```
Mean Squared Error:  2611322795234.76
Bias:  2593250631409.784
Variance:  18072163824.9759
```

3. Ridge (L2) regression

```
[47]: from sklearn.linear_model import Ridge
      ridge_model = Ridge(alpha=50, max_iter=100, tol=0.1)
      ridge_model.fit(x_train, y_train)
```

```
[47]: Ridge(alpha=50, max_iter=100, tol=0.1)
```

```
[48]: # R2 score for training data
      ridge_train_R2 = ridge_model.score(x_train, y_train)
      ridge_train_R2
```

```
[48]: 0.31304271936160477
```

```
[49]: # R2 score for testing data
      ridge_test_R2 = ridge_model.score(x_test, y_test)
      ridge_test_R2
```

```
[49]: 0.24866676643523355
```

[50]:
```
# Bias and Variance
Ridge_mse, Ridge_bias, Ridge_variance =  bias_variance_decomp(ridge_model,␣
 ↪x_train.values, y_train.values, x_test.values, y_test.values, loss='mse',␣
 ↪random_seed=3, num_rounds=100)
```

[51]:
```
# Display Bias and Variance
print("Mean Squared Error: ", round(Ridge_mse, 4))
print("Bias: ",round(Ridge_bias, 4))
print("Variance: ",round(Ridge_variance, 4))
```

```
Mean Squared Error:  2686698579089.878
Bias:  2670452435029.522
Variance:  16246144060.3572
```

0.0.6 Underfitting and overfitting observation

[52]:
```
print("Method \t R2_Taining   R2_Testing")
print("=============================")
print("Linear  ",round(linear_train_R2,2)*100,"\t   ",␣
 ↪round(linear_test_R2,2)*100)
print("Lasso   ",round(lasso_train_R2,2)*100,"\t   ",␣
 ↪round(lasso_test_R2,2)*100)
print("Ridge   ",round(ridge_train_R2,2)*100,"\t   ",␣
 ↪round(ridge_test_R2,2)*100)
```

```
Method   R2_Taining   R2_Testing
=============================
Linear   32.0       27.0
Lasso    32.0       27.0
Ridge    31.0       25.0
```

0.0.7 MSE, Bias and Variace observation

[53]:
```
print("Method \t\t MSE \t\t\t  Bias \t\t\t  Variance")
print("===============================================================================")
print("Linear  ",Linear_mse,"\t   ", Linear_bias, "\t   ",Linear_variance)
print("Ridge  ",Ridge_mse,"\t   ", Ridge_bias, "\t   ",Ridge_variance)
print("Lasso  ",Lasso_mse,"\t   ", Lasso_bias, "\t   ",Lasso_variance)
```

```
Method           MSE                      Bias                    Variance
===============================================================================
Linear   2615079328367.502          2593444760309.601       21634568057.901257
Ridge   2686698579089.878           2670452435029.522       16246144060.357227
Lasso   2611322795234.76            2593250631409.784       18072163824.975895
```