linear_ridge_lasso_elasticNet_regression

March 30, 2024

[1]:
```
## Aditya Agre
## 121B1B006
## second part: Linear Regression, Ridge, Lasso, and ElasticNet models
## Diabetes dataset
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

[3]:
```
diabetes = datasets.load_diabetes()
diabetes
```

[3]:
```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, …, -0.00259226,
          0.01990749, -0.01764613],
        [-0.00188202, -0.04464164, -0.05147406, …, -0.03949338,
         -0.06833155, -0.09220405],
        [ 0.08529891,  0.05068012,  0.04445121, …, -0.00259226,
          0.00286131, -0.02593034],
        …,
        [ 0.04170844,  0.05068012, -0.01590626, …, -0.01107952,
         -0.04688253,  0.01549073],
        [-0.04547248, -0.04464164,  0.03906215, …,  0.02655962,
          0.04452873, -0.02593034],
        [-0.04547248, -0.04464164, -0.0730303 , …, -0.03949338,
         -0.00422151,  0.00306441]]),
 'target': array([151.,  75., 141., 206., 135.,  97., 138.,  63., 110., 310.,
101.,
         69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
         68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
         87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
        259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
        128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
        150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
        200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
         42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
         83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
        104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
```

```
       173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
       107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
        60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
       197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
        59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
       237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
       143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
       142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
        77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
        78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
       154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
        71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
       150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
       145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
        94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
        60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
        31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
       114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
       191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
       244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
       263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
        77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
        58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
       140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
       219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
        43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
        84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
        94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
       220.,  57.]),
 'frame': None,
 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTen
baseline variables, age, sex, body mass index, average blood\npressure, and six
blood serum measurements were obtained for each of n =\n442 diabetes patients,
as well as the response of interest, a\nquantitative measure of disease
progression one year after baseline.\n\n**Data Set Characteristics:**\n\n
:Number of Instances: 442\n\n  :Number of Attributes: First 10 columns are
numeric predictive values\n\n  :Target: Column 11 is a quantitative measure of
disease progression one year after baseline\n\n  :Attribute Information:\n
- age     age in years\n      - sex\n      - bmi     body mass index\n      - bp
average blood pressure\n      - s1      tc, total serum cholesterol\n      - s2
ldl, low-density lipoproteins\n      - s3      hdl, high-density lipoproteins\n
- s4      tch, total cholesterol / HDL\n      - s5      ltg, possibly log of
serum triglycerides level\n      - s6      glu, blood sugar level\n\nNote: Each
of these 10 feature variables have been mean centered and scaled by the standard
deviation times the square root of `n_samples` (i.e. the sum of squares of each
column totals 1).\n\nSource
```

```
URL:\nhttps://www4.stat.ncsu.edu/~boos/var.select/diabetes.html\n\nFor more
information see:\nBradley Efron, Trevor Hastie, Iain Johnstone and Robert
Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with
discussion),
407-499.\n(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)\n',
 'feature_names': ['age',
  'sex',
  'bmi',
  'bp',
  's1',
  's2',
  's3',
  's4',
  's5',
  's6'],
 'data_filename': 'diabetes_data_raw.csv.gz',
 'target_filename': 'diabetes_target.csv.gz',
 'data_module': 'sklearn.datasets.data'}
```

[4]:
```
diabetes.feature_names
```

[4]:
```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

[5]:
```
x = diabetes.data
y = diabetes.target
```

[6]:
```
x.shape, y.shape
```

[6]:
```
((442, 10), (442,))
```

[7]:
```
from sklearn.model_selection import train_test_split
```

[8]:
```
x1, x2, y1, y2 = train_test_split(x,y,test_size=0.3,random_state=99)
```

[ ]:
```

```

[9]:
```
# Linear Regression
from sklearn.linear_model import LinearRegression
```

[10]:
```
lr_object = LinearRegression()
```

[11]:
```
lr_object.fit(x1,y1)
```

[11]:
```
LinearRegression()
```

[12]:
```
predictions_on_x2 = lr_object.predict(x2)
```

[13]:
```
comparison_Linear_regression = pd.DataFrame({'Given_values_y2' : y2,␣
 ↪'predicted_values' : predictions_on_x2})
print(comparison_Linear_regression)
```

```
     Given_values_y2  predicted_values
0               75.0         77.999103
1              128.0        170.447121
2              125.0        109.036606
3              332.0        223.843071
4               37.0         87.384304
..               ...               ...
128             48.0        202.349391
129            172.0        144.663523
130             51.0         82.400892
131            277.0        185.443544
132             94.0        101.563213

[133 rows x 2 columns]
```

[27]:
```
lr_object.score(x2,y2)
```

[27]: 0.4545709909725648

[ ]:
```

```

[14]:
```
# Ridge regression. (Still linear)
from sklearn.linear_model import Ridge
```

[15]:
```
ridge_obj = Ridge(alpha=0.1)
ridge_obj.fit(x1,y1)
Ridge()
```

[15]: Ridge()

[16]:
```
predictions_on_x2 = ridge_obj.predict(x2)
```

[17]:
```
comparison_Ridge_regression = pd.DataFrame({'Given_values_y2' : y2,␣
 ↪'predicted_values' : predictions_on_x2})
print(comparison_Ridge_regression)
```

```
     Given_values_y2  predicted_values
0               75.0         89.451853
1              128.0        167.891164
2              125.0        105.681527
3              332.0        219.512632
4               37.0         86.868393
..               ...               ...
128             48.0        192.038147
129            172.0        143.200097
```

```
130             51.0         80.603296
131            277.0        184.413866
132             94.0        105.191343

[133 rows x 2 columns]
```

[28]:
```
ridge_obj.score(x2,y2)
```

[28]: 0.46062181702662

[ ]:
```

```

[18]:
```
## Lasso
from sklearn.linear_model import Lasso
```

[19]:
```
lasso_obj = Lasso(alpha = 0.1)
lasso_obj.fit(x1,y1)
Lasso()
```

[19]: Lasso()

[20]:
```
predictions = lasso_obj.predict(x2)
```

[21]:
```
comparison_Lasso_regr = pd.DataFrame({'Given_values_y2' : y2,␣
 ↪'predicted_values' : predictions_on_x2})
print(comparison_Lasso_regr)
```

```
     Given_values_y2  predicted_values
0               75.0         89.451853
1              128.0        167.891164
2              125.0        105.681527
3              332.0        219.512632
4               37.0         86.868393
..               ...               ...
128             48.0        192.038147
129            172.0        143.200097
130             51.0         80.603296
131            277.0        184.413866
132             94.0        105.191343

[133 rows x 2 columns]
```

[29]:
```
lasso_obj.score(x2,y2)
```

[29]: 0.4672409597780036

[ ]:
```

```

[22]:
```
## Elastic Net
from sklearn.linear_model import ElasticNet
```

[43]:
```
elastic_object = ElasticNet(alpha = 0.00005)
elastic_object.fit(x1,y1)
ElasticNet()
```

[43]:
```
ElasticNet()
```

[44]:
```
predictions = elastic_object.predict(x2)
```

[45]:
```
comparison_elastic_regr = pd.DataFrame({'Given_values_y2' : y2,␣
 ↪'predicted_values' : predictions_on_x2})
print(comparison_elastic_regr)
```

```
     Given_values_y2  predicted_values
0               75.0         89.451853
1              128.0        167.891164
2              125.0        105.681527
3              332.0        219.512632
4               37.0         86.868393
..               …               …
128             48.0        192.038147
129            172.0        143.200097
130             51.0         80.603296
131            277.0        184.413866
132             94.0        105.191343

[133 rows x 2 columns]
```

[46]:
```
elastic_object.score(x2,y2)
```

[46]:
```
0.45312270953163125
```

[ ]:
```
## Therefore, after comparison using R squared test, the best performance is␣
 ↪shown by Lasso regression.
```