Regularaization

April 2, 2023

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn import datasets
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
```

[2]:
```
# Checking all available datasets in sklearn
for data in dir(datasets):
     if data.startswith("load"):
        print(data)
```

```
load_breast_cancer
load_diabetes
load_digits
load_files
load_iris
load_linnerud
load_sample_image
load_sample_images
load_svmlight_file
load_svmlight_files
load_wine
```

[3]:
```
x,y = datasets.make_regression(n_samples = 20 , n_features = 5 , n_informative␣
 ↪= 3 , n_targets = 1 , random_state = 42 )
```

[4]:
```
x
```

[4]:
```
array([[-1.10633497, -0.47917424,  0.81252582, -1.19620662, -0.18565898],
       [-0.29900735,  0.8219025 , -1.98756891,  0.09176078,  0.08704707],
       [-0.676922  ,  0.32408397,  1.03099952,  0.61167629, -0.38508228],
       [ 0.32875111, -0.50175704,  0.51326743, -0.5297602 ,  0.91540212],
```

```
       [ 0.37569802,  0.11092259, -0.29169375, -0.60063869, -1.15099358],
       [-1.95967012, -1.22084365,  0.19686124, -1.32818605,  0.2088636 ],
       [ 0.0675282 ,  1.46564877, -0.54438272, -1.42474819, -0.2257763 ],
       [-0.70205309,  0.09707755, -0.39210815, -0.32766215,  0.96864499],
       [ 0.24196227, -0.46341769, -1.72491783, -1.91328024, -0.46572975],
       [ 0.76743473, -0.23413696,  0.54256004, -0.46947439,  1.57921282],
       [-0.01349722, -0.60170661,  0.82254491, -1.05771093,  1.85227818],
       [ 0.26105527, -1.46351495, -0.23458713,  0.00511346,  0.29612028],
       [ 0.31424733, -0.56228753, -1.4123037 , -0.90802408, -1.01283112],
       [ 1.05712223, -0.71984421, -1.76304016,  0.34361829, -0.46063877],
       [ 0.64768854,  0.49671415, -0.23415337,  1.52302986, -0.1382643 ],
       [ 1.0035329 ,  1.35624003, -0.64511975,  0.36163603, -0.07201012],
       [ 1.47789404, -0.21967189, -0.8084936 , -0.51827022,  0.35711257],
       [-0.11564828,  0.73846658, -1.47852199, -0.3011037 ,  0.17136828],
       [-0.03582604,  0.36139561, -2.6197451 ,  1.56464366,  1.53803657],
       [-0.30921238,  0.93128012,  0.97554513,  0.33126343, -0.83921752]])
```

[5]:
```
y
```

[5]:
```
array([-37.38601562,  32.65643567,  -3.13930716,   3.32849217,
       -16.43111729, -69.80999161,  56.97255362,  17.15130032,
       -26.78749633,  34.50130044,  15.86286415, -51.38251302,
       -42.19953322, -27.6070018 ,  25.42597962,  67.04043398,
        16.47723408,  33.24440858,  48.81661351,  16.52659471])
```

[6]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.2 , random_state␣
 ↪= 42)
```

[7]:
```
xtrain
```

[7]:
```
array([[ 0.24196227, -0.46341769, -1.72491783, -1.91328024, -0.46572975],
       [-1.95967012, -1.22084365,  0.19686124, -1.32818605,  0.2088636 ],
       [ 0.26105527, -1.46351495, -0.23458713,  0.00511346,  0.29612028],
       [ 0.32875111, -0.50175704,  0.51326743, -0.5297602 ,  0.91540212],
       [-0.03582604,  0.36139561, -2.6197451 ,  1.56464366,  1.53803657],
       [ 1.47789404, -0.21967189, -0.8084936 , -0.51827022,  0.35711257],
       [ 1.05712223, -0.71984421, -1.76304016,  0.34361829, -0.46063877],
       [-0.676922  ,  0.32408397,  1.03099952,  0.61167629, -0.38508228],
       [ 0.76743473, -0.23413696,  0.54256004, -0.46947439,  1.57921282],
       [-0.30921238,  0.93128012,  0.97554513,  0.33126343, -0.83921752],
       [ 0.37569802,  0.11092259, -0.29169375, -0.60063869, -1.15099358],
       [ 0.31424733, -0.56228753, -1.4123037 , -0.90802408, -1.01283112],
       [-0.70205309,  0.09707755, -0.39210815, -0.32766215,  0.96864499],
       [-0.01349722, -0.60170661,  0.82254491, -1.05771093,  1.85227818],
       [ 0.64768854,  0.49671415, -0.23415337,  1.52302986, -0.1382643 ],
       [ 0.0675282 ,  1.46564877, -0.54438272, -1.42474819, -0.2257763 ]])
```

[8]:
```
xtest
```

[8]:
```
array([[-1.10633497, -0.47917424,  0.81252582, -1.19620662, -0.18565898],
       [-0.11564828,  0.73846658, -1.47852199, -0.3011037 ,  0.17136828],
       [ 1.0035329 ,  1.35624003, -0.64511975,  0.36163603, -0.07201012],
       [-0.29900735,  0.8219025 , -1.98756891,  0.09176078,  0.08704707]])
```

[9]:
```
ytrain
```

[9]:
```
array([-26.78749633, -69.80999161, -51.38251302,   3.32849217,
        48.81661351,  16.47723408, -27.6070018 ,  -3.13930716,
        34.50130044,  16.52659471, -16.43111729, -42.19953322,
        17.15130032,  15.86286415,  25.42597962,  56.97255362])
```

[10]:
```
ytest
```

[10]:
```
array([-37.38601562,  33.24440858,  67.04043398,  32.65643567])
```

[11]:
```
lr = LinearRegression()
lr.fit(xtrain,ytrain)
```

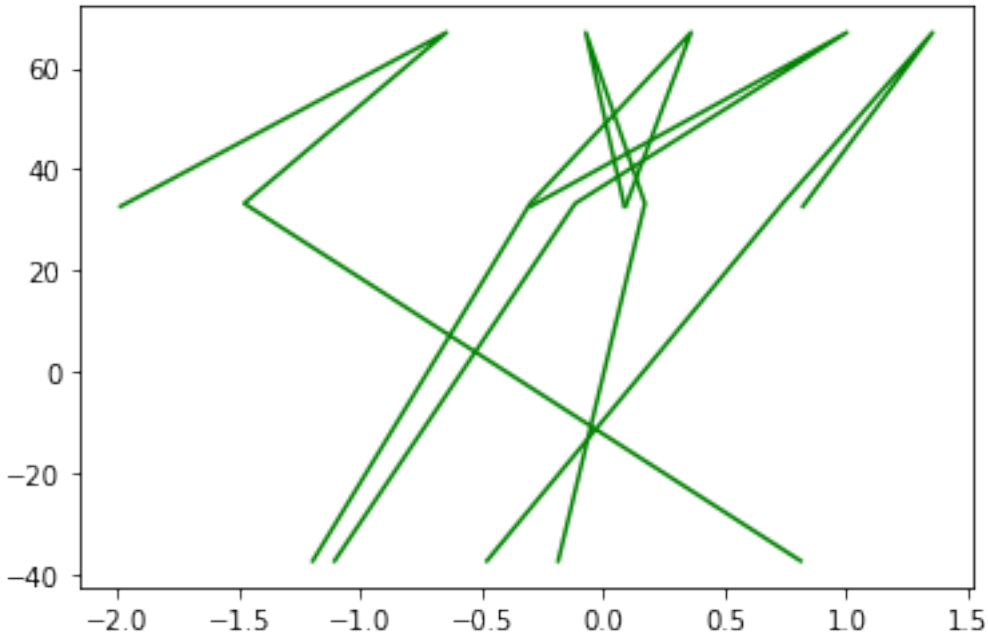
[11]:
```
LinearRegression()
```

[14]:
```
ypred = lr.predict(xtest)
```

[15]:
```
ypred
```

[15]:
```
array([-37.38601562,  33.24440858,  67.04043398,  32.65643567])
```

[18]:
```
#displaying the regression line
plt.plot(xtest,ypred,color = 'g')
plt.show()
```



[19]:
```
# Using ridge and lasso
from sklearn.linear_model import Ridge , Lasso
```

[22]:
```
lmodel = Lasso()
lmodel.fit(xtrain,ytrain)
```

[22]: Lasso()

[23]:
```
rmodel = Ridge()
rmodel.fit(xtrain,ytrain)
```

[23]: Ridge()

[24]:
```
rmodel.score(xtest,ytest)
```

[24]: 0.9822010721661684

[25]:
```
lmodel.score(xtest,ytest)
```

[25]: 0.9943749364100763

[26]:
```
rypred = rmodel.predict(xtest)
```

[27]:
```
lypred = lmodel.predict(xtest)
```

[28]:
```
ytest
```

[28]:
```
array([-37.38601562,  33.24440858,  67.04043398,  32.65643567])
```

[33]:
```
ypred
```

[33]:
```
array([-37.38601562,  33.24440858,  67.04043398,  32.65643567])
```

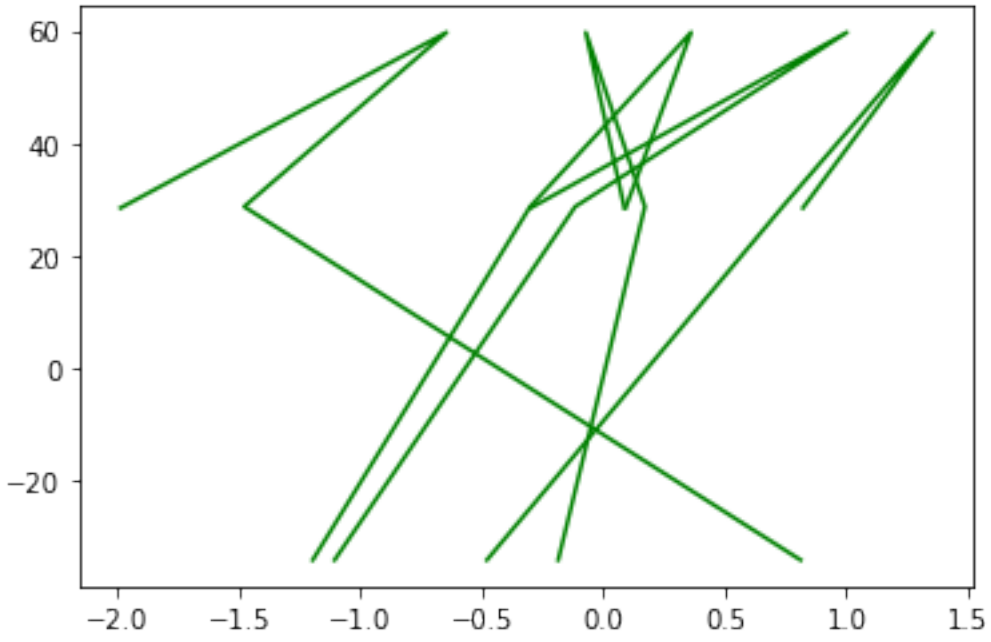
[29]:
```
rypred
```

[29]:
```
array([-34.18125096,  28.68057599,  59.68797944,  28.44465534])
```

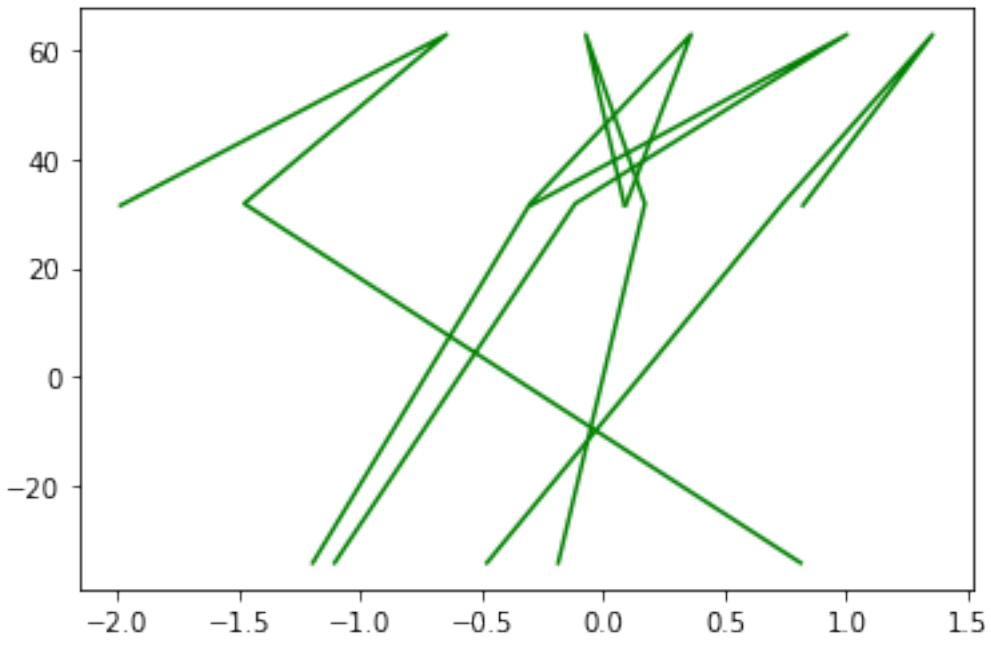
[30]:
```
lypred
```

[30]:
```
array([-34.06689983,  31.79121928,  62.80313925,  31.45814313])
```

[31]:
```
#ridge curve
plt.plot(xtest,rypred,color = 'g')
plt.show()
```



[32]:
```
# Lasso curve
plt.plot(xtest,lypred,color = 'g')
plt.show()
```



[40]:
```
# changing alpha value for ridge and lasso
alpha_lst = [0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.2,0.3]
for i in alpha_lst:
    r1model = Ridge(alpha = i)
    l1model = Lasso(alpha = i)
    ridge_model = r1model.fit(xtrain,ytrain)
    lasso_model = l1model.fit(xtrain,ytrain)
    rmpred = r1model.predict(xtest)
    lmpred = l1model.predict(xtest)
    print(f"the ridge model is score is {r1model.score(xtest,ytest)} while the␣
 ↪alpha value is {i} and the error array is {rmpred-ypred}")
    print(f"the lasso model is score is {l1model.score(xtest,ytest)} while the␣
 ↪alpha value is {i} and the error array is {lmpred-ypred}")
```

```
the ridge model is score is 0.9999976783938865 while the alpha value is 0.01 and
the error array is [ 0.03562357 -0.0529091  -0.08350739 -0.04877973]
the lasso model is score is 0.9999994375031381 while the alpha value is 0.01 and
the error array is [ 0.03318626 -0.01453592 -0.04237328 -0.01198812]
the ridge model is score is 0.9999907403588116 while the alpha value is 0.02 and
the error array is [ 0.07116587 -0.1056482  -0.16678441 -0.09740342]
the lasso model is score is 0.999997750050292 while the alpha value is 0.02 and
the error array is [ 0.06633211 -0.02910721 -0.08475195 -0.02402154]
the ridge model is score is 0.9999792258002949 while the alpha value is 0.03 and
the error array is [ 0.10662722 -0.15821813 -0.24983209 -0.14587183]
the lasso model is score is 0.9999949376441584 while the alpha value is 0.03 and
the error array is [ 0.0995312  -0.04363099 -0.12712239 -0.03599424]
```

```
the ridge model is score is 0.9999631742675877 while the alpha value is 0.04 and
the error array is [ 0.14200791 -0.21061969 -0.33265142 -0.19418573]
the lasso model is score is 0.9999910001328697 while the alpha value is 0.04 and
the error array is [ 0.13272676 -0.05815926 -0.16949504 -0.04797249]
the ridge model is score is 0.9999426249572687 while the alpha value is 0.05 and
the error array is [ 0.17730827 -0.26285369 -0.41524339 -0.24234586]
the lasso model is score is 0.9999859375916574 while the alpha value is 0.05 and
the error array is [ 0.16592233 -0.07268752 -0.21186768 -0.05995073]
the ridge model is score is 0.9999176167167971 while the alpha value is 0.06 and
the error array is [ 0.2125286  -0.31492094 -0.49760899 -0.29035296]
the lasso model is score is 0.9999797500205214 while the alpha value is 0.06 and
the error array is [ 0.1991179  -0.08721579 -0.25424033 -0.07192898]
the ridge model is score is 0.9998881880480256 while the alpha value is 0.07 and
the error array is [ 0.2476692  -0.36682222 -0.57974919 -0.33820778]
the lasso model is score is 0.9999724374194617 while the alpha value is 0.07 and
the error array is [ 0.23231347 -0.10174406 -0.29661298 -0.08390723]
the ridge model is score is 0.9998543771106733 while the alpha value is 0.08 and
the error array is [ 0.28273039 -0.41855834 -0.66166498 -0.38591105]
the lasso model is score is 0.9999639997884784 while the alpha value is 0.08 and
the error array is [ 0.26550903 -0.11627232 -0.33898562 -0.09588548]
the ridge model is score is 0.9998162217257589 while the alpha value is 0.09 and
the error array is [ 0.31771246 -0.47013007 -0.74335733 -0.43346351]
the lasso model is score is 0.9999544371275714 while the alpha value is 0.09 and
the error array is [ 0.2987046  -0.13080059 -0.38135827 -0.10786372]
the ridge model is score is 0.9997737593789953 while the alpha value is 0.1 and
the error array is [ 0.35261572 -0.5215382  -0.8248272  -0.48086587]
the lasso model is score is 0.9999437494367407 while the alpha value is 0.1 and
the error array is [ 0.33190017 -0.14532885 -0.42373092 -0.11984197]
the ridge model is score is 0.999120266727726 while the alpha value is 0.2 and
the error array is [ 0.69737865 -1.02678981 -1.62749745 -0.94679107]
the lasso model is score is 0.9997749974082093 while the alpha value is 0.2 and
the error array is [ 0.66382529 -0.29063604 -0.84745872 -0.23965617]
the ridge model is score is 0.9980749469789836 while the alpha value is 0.3 and
the error array is [ 1.03457985 -1.51650538 -2.408939   -1.39847284]
the lasso model is score is 0.9994937438855456 while the alpha value is 0.3 and
the error array is [ 0.99574245 -0.43595037 -1.27118781 -0.35947949]
```

[ ]: