favy7n910

June 4, 2023

1 PREDICTION OF THE AMOUNT OF ELECTRICITY PRODUCED

We start by loading the libraries we will use in this exercise.

[1]:
```
#!/usr/bin/python3

from sys import argv, exit

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
```

We load the inputs and the labels.

[2]:
```
inputs = np.load('inputs.npy')

inputs.shape
```

[2]:
```
(200, 100)
```

[3]:
```
labels = np.load('labels.npy')
labels.shape
```

[3]:
```
(200, 1)
```

We display the inputs to see what they look like.

[4]:
```
print(inputs)
```

```
[[0.48929605 0.54823009 0.78954784 … 0.06574024 0.83155552 0.63574974]
 [0.19334964 0.42533153 0.17758109 … 0.93297484 0.18750137 0.4060811 ]
 [0.20858188 0.69655482 0.80855935 … 0.26235161 0.11683627 0.53794816]
 …
 [0.96534061 0.74722521 0.81930219 … 0.47769342 0.80389708 0.87808281]
 [0.27179979 0.97780377 0.15373318 … 0.1298077  0.92668414 0.97855557]
 [0.88861494 0.89620836 0.85159905 … 0.53349038 0.19949587 0.89726618]]
```

We separate the inputs and labels into train and test in order to train our model with the train part and test the model with the test part. We put 20% in the test.

[5]:
```
X_train, X_test, y_train, y_test = train_test_split(inputs, labels, test_size=0.
↪2, random_state=42)
X_train.shape
```

[5]:
```
(160, 100)
```

[6]:
```
clf = LinearRegression().fit(X_train, y_train)

print("test mean accuracy:")
print(clf.score(X_test, y_test))
```

```
test mean accuracy:
0.7844092212875204
```

With the linear regression model we get a score of 0.78 which is average.

[7]:
```
lso = Lasso(alpha=0).fit(X_train, y_train)

print("test mean accuracy:")
print(lso.score(X_test, y_test))
```

```
test mean accuracy:
0.7844092212875182
```

```
/var/folders/xs/j7p79qk53mjb3kj4_v3lw9280000gn/T/ipykernel_52810/4177933628.py:1
: UserWarning: With alpha=0, this algorithm does not converge well. You are
advised to use the LinearRegression estimator
  lso = Lasso(alpha=0).fit(X_train, y_train)
/usr/local/lib/python3.11/site-
packages/sklearn/linear_model/_coordinate_descent.py:631: UserWarning:
Coordinate descent with no regularization may lead to unexpected results and is
discouraged.
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.11/site-
packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 1.541e+00, tolerance: 6.840e-03 Linear regression models with null weight
for the l1 regularization term are more efficiently fitted using one of the
solvers implemented in sklearn.linear_model.Ridge/RidgeCV instead.
  model = cd_fast.enet_coordinate_descent(
```

With the Lasso model we get a score of 0.88 which is a little better than the Linearregression.

[8]:
```
rdg = Ridge(alpha=1, fit_intercept=False).fit(X_train, y_train)
```

```
print("test mean accuracy:")
print(rdg.score(X_test, y_test))
```

```
test mean accuracy:
0.8628146279955007
```

Finally we used the Ridge which has a score of 0.86 and which therefore obtained the best score by setting the argument alpha=1 and the fit_intercept=False.