Class_09.04.2023_Holdout_cross_Val

April 9, 2023

100 Data points
80/20 split , training data is geetting 80 datapoints
testing data is getting 20 data points

data

a
b
c
d
e
f
g
h
i
j

Training data = a-h
testing data = j,i

k = 3

- the entire dataset will be divided into 3 equal parts
- suppose the parts are called as p1,p2,p3
- there would be 3 iterations as a whole to perform train/test split
- 1st iteration - p1 will be the testing data/p2 and p3 will be training data - e1
- 2nd iteration - p2 will be the testing data/p1 and p3 will be training data - e2
- 3rd iteration - p3 will be testing data/p1 and p2 will be training data - e3
- Final error estimate - (e1+e2+e3)/k - (e1+e2+e3)/3

1. Every single data point is used for both training and testing
2. Optimum error calculation is possible

[27]:
```
from sklearn.model_selection import KFold
```

[28]:
```
kf = KFold(n_splits=3)
kf
```

[28]:
```
KFold(n_splits=3, random_state=None, shuffle=False)
```

[29]:
```
for train,test in kf.split([1,2,3,4,5,6,7,8,9]):
    print(train,test)
```

```
[3 4 5 6 7 8] [0 1 2]
[0 1 2 6 7 8] [3 4 5]
[0 1 2 3 4 5] [6 7 8]
```

[30]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_digits
```

[31]:
```
digits = load_digits()
```

[32]:
```
digits.keys()
```

[32]:
```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images',
'DESCR'])
```

[33]:
```
print(digits['DESCR'])
```

```
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 1797
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digit
s

The data set contains images of hand-written digits: 10 classes where
each class refers to a digit.

Preprocessing programs made available by NIST were used to extract
normalized bitmaps of handwritten digits from a preprinted form. From a
total of 43 people, 30 contributed to the training set and different 13
to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of
4x4 and the number of on pixels are counted in each block. This generates
an input matrix of 8x8 where each element is an integer in the range
```

```
0..16. This reduces dimensionality and gives invariance to small
distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.
T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.
L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,
1994.

.. topic:: References

  - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their
    Applications to Handwritten Digit Recognition, MSc Thesis, Institute of
    Graduate Studies in Science and Engineering, Bogazici University.
  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
  - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.
    Linear dimensionalityreduction using relevance weighted LDA. School of
    Electrical and Electronic Engineering Nanyang Technological University.
    2005.
  - Claudio Gentile. A New Approximate Maximal Margin Classification
    Algorithm. NIPS. 2000.
```

[34]:
```
print(digits['feature_names'])
```

```
['pixel_0_0', 'pixel_0_1', 'pixel_0_2', 'pixel_0_3', 'pixel_0_4', 'pixel_0_5',
'pixel_0_6', 'pixel_0_7', 'pixel_1_0', 'pixel_1_1', 'pixel_1_2', 'pixel_1_3',
'pixel_1_4', 'pixel_1_5', 'pixel_1_6', 'pixel_1_7', 'pixel_2_0', 'pixel_2_1',
'pixel_2_2', 'pixel_2_3', 'pixel_2_4', 'pixel_2_5', 'pixel_2_6', 'pixel_2_7',
'pixel_3_0', 'pixel_3_1', 'pixel_3_2', 'pixel_3_3', 'pixel_3_4', 'pixel_3_5',
'pixel_3_6', 'pixel_3_7', 'pixel_4_0', 'pixel_4_1', 'pixel_4_2', 'pixel_4_3',
'pixel_4_4', 'pixel_4_5', 'pixel_4_6', 'pixel_4_7', 'pixel_5_0', 'pixel_5_1',
'pixel_5_2', 'pixel_5_3', 'pixel_5_4', 'pixel_5_5', 'pixel_5_6', 'pixel_5_7',
'pixel_6_0', 'pixel_6_1', 'pixel_6_2', 'pixel_6_3', 'pixel_6_4', 'pixel_6_5',
'pixel_6_6', 'pixel_6_7', 'pixel_7_0', 'pixel_7_1', 'pixel_7_2', 'pixel_7_3',
'pixel_7_4', 'pixel_7_5', 'pixel_7_6', 'pixel_7_7']
```

[35]:
```
print(digits['target'])
```

```
[0 1 2 … 8 9 8]
```

[36]:
```
print(digits['target_names'])
```

```
[0 1 2 3 4 5 6 7 8 9]
```

[37]:
```
print(digits.data)
```

```
[[ 0.  0.  5. …  0.  0.  0.]
 [ 0.  0.  0. … 10.  0.  0.]
 [ 0.  0.  0. … 16.  9.  0.]
 …
```

```
[ 0.  0.  1. … 6.  0.  0.]
[ 0.  0.  2. … 12.  0.  0.]
[ 0.  0. 10. … 12.  1.  0.]]
```

[38]:
```
x = digits.data
```

[39]:
```
y = digits.target
```

[40]:
```
from sklearn.preprocessing import StandardScaler
```

[41]:
```
scaler = StandardScaler()
```

[42]:
```
x = scaler.fit_transform(x)
```

[43]:
```
x
```

[43]:
```
array([[ 0.        , -0.33501649, -0.04308102, …, -1.14664746,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, …,  0.54856067,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, …,  1.56568555,
         1.6951369 , -0.19600752],
       …,
       [ 0.        , -0.33501649, -0.88456568, …, -0.12952258,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -0.67419451, …,  0.8876023 ,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649,  1.00877481, …,  0.8876023 ,
        -0.26113572, -0.19600752]])
```

[44]:
```
xtrain , xtest , ytrain , ytest = train_test_split(x,y,test_size = 0.2)
```

[45]:
```
xtrain.shape
```

[45]:
```
(1437, 64)
```

[46]:
```
ytrain.shape
```

[46]:
```
(1437,)
```

[47]:
```
xtest.shape
```

[47]:
```
(360, 64)
```

[48]:
```
ytest.shape
```

[48]:
```
(360,)
```

[49]:
```
lr = LogisticRegression()
rf = RandomForestClassifier(n_estimators = 30)
svm = SVC()
```

[50]:
```
models = [lr,rf,svm]
model_names = ['logistic regression','random forest classifier','support␣
 ↪vector']
```

[51]:
```
for i in models:
    i.fit(xtrain,ytrain)
    print(f'the score for model{i} is {i.score(xtest,ytest)}')
```

```
the score for modelLogisticRegression() is 0.9666666666666667
the score for modelRandomForestClassifier(n_estimators=30) is 0.9694444444444444
the score for modelSVC() is 0.9916666666666667
```

[52]:
```
from sklearn.model_selection import cross_val_score
```

[57]:
```
lr_cross_val = cross_val_score(LogisticRegression(),x,y,cv = 5)
svm_cross_val = cross_val_score(SVC(),x,y,cv = 5)
rf_cross_val = cross_val_score(RandomForestClassifier(),x,y,cv = 5)
```

[58]:
```
print(lr_cross_val)
```

```
[0.91388889 0.87777778 0.94428969 0.9637883  0.89693593]
```

[59]:
```
print(svm_cross_val)
```

```
[0.96388889 0.93055556 0.94986072 0.95543175 0.94707521]
```

[60]:
```
print(rf_cross_val)
```

```
[0.93055556 0.9        0.95543175 0.9637883  0.92200557]
```

[ ]:
```

```