cross_validation

January 31, 2019

```
In [1]: from sklearn.datasets import load_digits
        import numpy as np
        digits = load_digits()
        print(digits.DESCR)
        x = digits.data
        y = digits.target
```

```
.. _digits_dataset:

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

**Data Set Characteristics:**

    :Number of Instances: 5620
    :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
http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

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.
```

```
In [3]: # hypotheses
        from sklearn import svm
        from sklearn.model_selection import train_test_split, cross_val_score
        from sklearn import datasets
        h1 = svm.LinearSVC(C=1.0)
        h2 = svm.SVC(kernel='rbf', degree=3, gamma=0.001, C=1.0)# classifier with a radial bas
        h3 = svm.SVC(kernel='poly', degree=3,gamma='auto', C=1.0) # third degree polynomial cl
```

```
In [14]: def cross_validate(eval_scoring): #https://scikit-learn.org/stable/modules/model_eval
             random_state = 1
             cv_folds = 10
             workers = -1 # use whole processor
             X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.30, random_s
             for hypotezis in [h1, h2, h3]:
                 scores = cross_val_score(hypotezis, X_train, Y_train, cv = cv_folds, scoring =
                 print("%s \nEffectiveness of cross validation:\naverage: %s \nstandard deviati
         print('***accuracy***')
         cross_validate(eval_scoring = 'accuracy')
         print('***f1***')
         cross_validate(eval_scoring = 'f1_weighted')
```

```
***accuracy***
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
Effectiveness of cross validation:
average: 0.9378222009981659
standard deviation: 0.013855685599020694

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
```

```
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Effectiveness of cross validation:
average: 0.9897240142951679
standard deviation: 0.007119379800261718

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='poly',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Effectiveness of cross validation:
average: 0.986570346539386
standard deviation: 0.01044358650683979

***f1***
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
Effectiveness of cross validation:
average: 0.9365417720889513
standard deviation: 0.01877199890743059

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Effectiveness of cross validation:
average: 0.9896685275498038
standard deviation: 0.00720548902289445

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='poly',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Effectiveness of cross validation:
average: 0.9865325829712441
standard deviation: 0.01041697274541247
```

In [ ]: