Project 2: Recognizing Handwritten Digits with scikit-learn ¶

~ Ankita Komal

In [22]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Import scikit-learn

In [23]:

```
from sklearn import svm
svc = svm.SVC(gamma=0.001, C=100.)
```

Load Dataset

In [24]:

```
from sklearn import datasets
digits = datasets.load_digits()
```

In [26]:

```
digits.images[0]import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Out[26]:

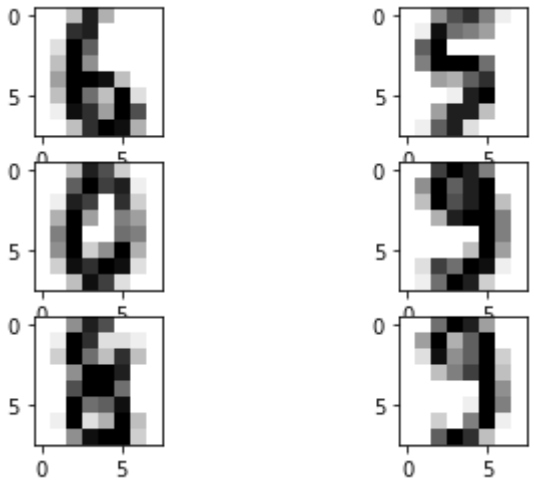
```
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

In [5]:

```
plt.subplot(321)
plt.imshow(digits.images[1701], cmap=plt.cm.gray_r, interpolation='nearest')
plt.subplot(322)
plt.imshow(digits.images[1702], cmap=plt.cm.gray_r, interpolation='nearest')
plt.subplot(323)
plt.imshow(digits.images[1703], cmap=plt.cm.gray_r, interpolation='nearest')
plt.subplot(324)
plt.imshow(digits.images[1704], cmap=plt.cm.gray_r, interpolation='nearest')
plt.subplot(325)
plt.imshow(digits.images[1705], cmap=plt.cm.gray_r, interpolation='nearest')
plt.subplot(326)
plt.imshow(digits.images[1706], cmap=plt.cm.gray_r, interpolation='nearest')
```

Out[5]:

```
<matplotlib.image.AxesImage at 0x1f7df4624c0>
```



Training and testing the dataset and predict

In [6]:

```
svc.fit(digits.data[1:1600], digits.target[1:1600])
svc.predict(digits.data[1701:1707])
```

Out[6]:

```
array([6, 5, 0, 9, 8, 9])
```

Fit and Predict the dataset

In [7]:

```
svc.fit(digits.data[1:1790], digits.target[1:1790])
svc.predict(digits.data[1791:1796])
```

Out[7]:

```
array([4, 9, 0, 8, 9])
```

Import libraries

In [9]:

```
from sklearn import datasets
from sklearn import svm
from matplotlib import pyplot as plt
```

Loading the Digits dataset

In [10]:

```
svc = svm.SVC(gamma=0.001, C=100.)
digits = datasets.load_digits()
```

Lots of information about the datasets by using the DESCR attribute

In [11]:

```
print(digits.DESCR)
```

```
.. _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
https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+D
igits (https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwr
itten+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 [12]:

```
print(digits.images[0])
```

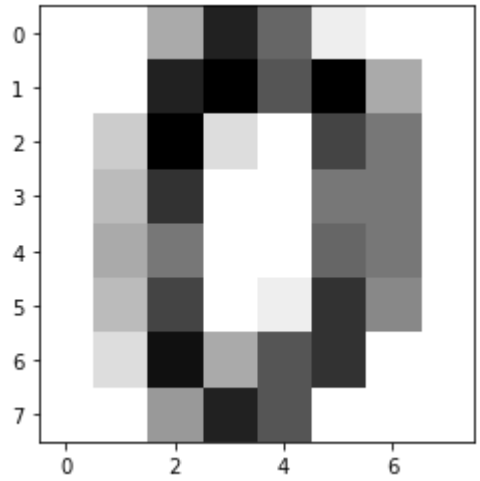
```
[[ 0.  0.  5. 13.  9.  1.  0.  0.]
 [ 0.  0. 13. 15. 10. 15.  5.  0.]
 [ 0.  3. 15.  2.  0. 11.  8.  0.]
 [ 0.  4. 12.  0.  0.  8.  8.  0.]
 [ 0.  5.  8.  0.  0.  9.  8.  0.]
 [ 0.  4. 11.  0.  1. 12.  7.  0.]
 [ 0.  2. 14.  5. 10. 12.  0.  0.]
 [ 0.  0.  6. 13. 10.  0.  0.  0.]]
```

In [13]:

```
plt.imshow(digits.images[0], cmap=plt.cm.gray_r, interpolation='nearest')
```

Out[13]:

```
<matplotlib.image.AxesImage at 0x1f7dfb90ac0>
```



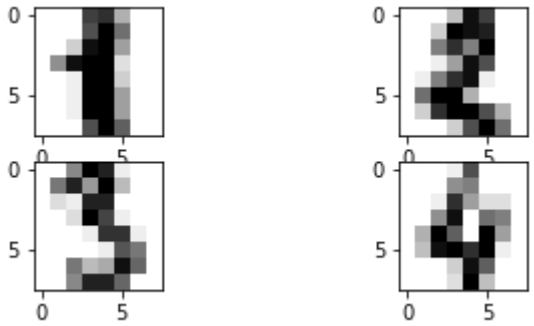
In [14]:

```
print(digits.target)
```

```
[0 1 2 ... 8 9 8]
```

In [15]:

```
for i in range(1,5):
    plt.subplot(3,2,i)
    plt.imshow(digits.images[i], cmap=plt.cm.gray_r,interpolation='nearest')
```



In [16]:

```
svc.fit(digits.data[1:1790],digits.target[1:1790])
svc.predict(digits.data[1:7])
```

Out[16]:

```
array([1, 2, 3, 4, 5, 6])
```

In [17]:

```
digits.target[1:7]
```

Out[17]:

```
array([1, 2, 3, 4, 5, 6])
```

In [18]:

```
from sklearn.metrics import accuracy_score
accuracy_score(svc.predict(digits.data[1:7]),digits.target[1:7])
```

Out[18]:

```
1.0
```

In [19]:

```
a=digits.target[1:7]
b=svc.predict(digits.data[1:7])

for i in range(len(a)):
    yes = 0
    no = 0
    if a[i] == b[i]:
        yes += 1
    else:
        no += 1
```

In [20]:

```
accuracy=(yes/(no+yes))*100

print(accuracy)
```

```
100.0
```

Conclusion: Here we learn ,how to import dataset from sklearn,how to build model and make prediction using function fit() and predict() .We also calculate accuracy by using the Sklearn library.