In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
```

In [2]:
```
%matplotlib inline
```

In [3]:
```
from sklearn.datasets import load_digits
```

In [4]:
```
digits = load_digits()
```

In [5]:
```
digits.keys()
```

Out[5]:
```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DESCR'])
```

In [6]:
```
digits.DESCR
```

Out[6]:
```
".. _digits_dataset:\n\nOptical recognition of handwritten digits dataset\n----------------------
---------------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 1797\n
:Number of Attributes: 64\n    :Attribute Information: 8x8 image of integer pixels in the range
0..16.\n    :Missing Attribute Values: None\n    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
\n    :Date: July; 1998\n\nThis is a copy of the test set of the UCI ML hand-written digits datas
ets\nhttps://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits\n\nThe dat
a set contains images of hand-written digits: 10 classes where\neach class refers to a digit.\n\n
Preprocessing programs made available by NIST were used to extract\nnormalized bitmaps of handwri
tten digits from a preprinted form. From a\ntotal of 43 people, 30 contributed to the training se
t and different 13\nto the test set. 32x32 bitmaps are divided into nonoverlapping blocks of\n4x4
and the number of on pixels are counted in each block. This generates\nan input matrix of 8x8 whe
re each element is an integer in the range\n0..16. This reduces dimensionality and gives invarian
ce to small\ndistortions.\n\nFor info on NIST preprocessing routines, see M. D. Garris, J. L. Blu
e, G.\nT. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.\nL. Wilson, NIST F
orm-Based Handprint Recognition System, NISTIR 5469,\n1994.\n\n.. topic:: References\n\n  - C. Ka
ynak (1995) Methods of Combining Multiple Classifiers and Their\n    Applications to Handwritten
Digit Recognition, MSc Thesis, Institute of\n    Graduate Studies in Science and Engineering, Bog
azici University.\n  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.\n  - Ken
Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.\n    Linear dimensionalityreduction
using relevance weighted LDA. School of\n    Electrical and Electronic Engineering Nanyang Techno
logical University.\n    2005.\n  - Claudio Gentile. A New Approximate Maximal Margin Classificat
ion\n    Algorithm. NIPS. 2000.\n"
```

In [7]:
```
digits.images[0]
```

Out[7]:
```
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 [8]:
```
digits.data
```

Out[8]:
```
array([[ 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.]])
```

In [14]:
```
digits.target
```

Out[14]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

In [17]:
```
print('Image Data Shape', digits.images.shape)
```

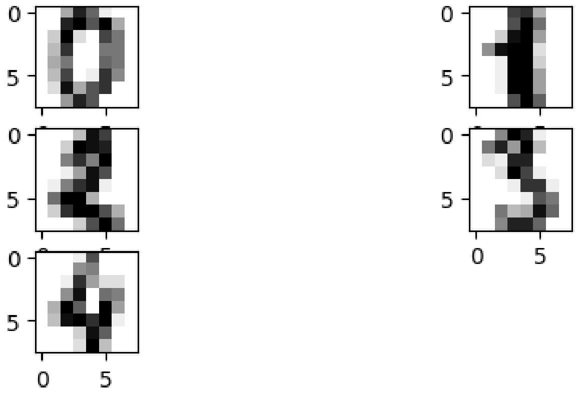
```
Image Data Shape (1797, 8, 8)
```

In [18]:
```
print('Label Data Shape', digits.target.shape)
```

```
Label Data Shape (1797,)
```

In [19]:
```
X = digits.images
```

In [22]:
```
rows = 5
columns = 2  # Change this to the desired number of columns
for i in range(5):
    plt.subplot(rows, columns, i + 1)
    plt.imshow(X[i], cmap=plt.cm.gray_r, interpolation='nearest')
```



In [23]:
```
from sklearn.metrics import accuracy_score,confusion_matrix # metrics error
from sklearn.model_selection import train_test_split # resampling method
```

In [24]:
```
X = digits.data
y = digits.target
```

In [25]:
```
from sklearn.multiclass import OneVsRestClassifier
```

In [26]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

In [27]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [28]:
```
knn = OneVsRestClassifier(KNeighborsClassifier())
```

In [29]:
```
knn.fit(X_train,y_train)
```

Out[29]:
```
OneVsRestClassifier
estimator: KNeighborsClassifier
KNeighborsClassifier
KNeighborsClassifier()
```

In [30]:
```
knn.predict(X_test[0].reshape(1,-1))
```

Out[30]:
```
array([2])
```

In [31]:
```
knn.predict(X_test[0:10])
```

Out[31]:
```
array([2, 8, 2, 6, 6, 7, 1, 9, 8, 5])
```

In [32]:
```
predictions = knn.predict(X_test)
```

In [33]:
```
%time
print('KNN Accuracy: %.3f' % accuracy_score(y_test,predictions))
```

```
Wall time: 0 ns
KNN Accuracy: 0.980
```

In [34]:
```
import seaborn as sns
```

In [35]:
```
cm = confusion_matrix(y_test,predictions)
plt.figure(figsize=(9,9))
sns.heatmap(cm,annot=True, fmt='.3f', linewidths=.5, square=True,cmap='Blues_r')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(accuracy_score(y_test,predictions))
plt.title(all_sample_title,size=15)
```

Out[35]:
```
Text(0.5, 1.0, 'Accuracy Score: 0.98')
```