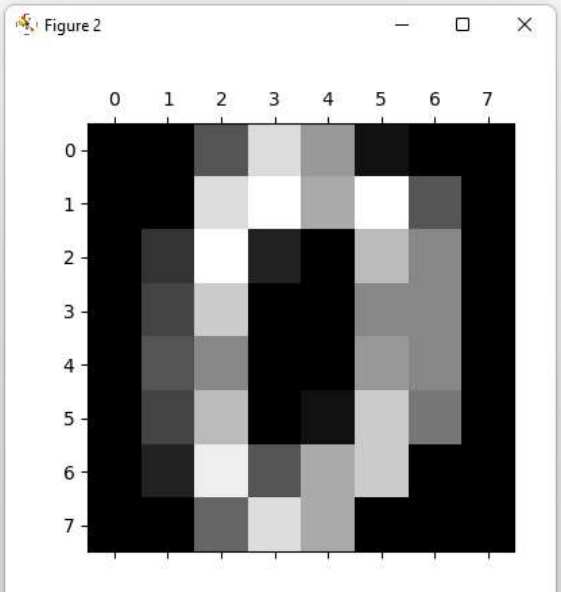
Expt: SVM

```
import cv2
import numpy as np
import matplotlib.pylab as plt
from sklearn import datasets, svm, metrics
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

digits=datasets.load_digits()
print('Digit dataset keys\n {}'.format(digits.keys()))

digits=load_digits()
print('shape of data:',digits.data.shape)
plt.gray()
plt.matshow(digits.images[0])

#print('dataset target name:\n{}\n'.format(digits.target_names))
#print('shape of dataset:{}'.format(digits.data.shape))
#print('shape of target:{}'.format(digits.target.shape))
#print('shape of images:',digits.images.shape)

#for i in range (10):
    #plt.subplot(2,5,i+1)
    #plt.axis('off')
    #plt.imshow(digits.images[i],cmap='gray_r',interpolation='nearest')
    #plt.title('target {}'.format(i))
plt.show()
```

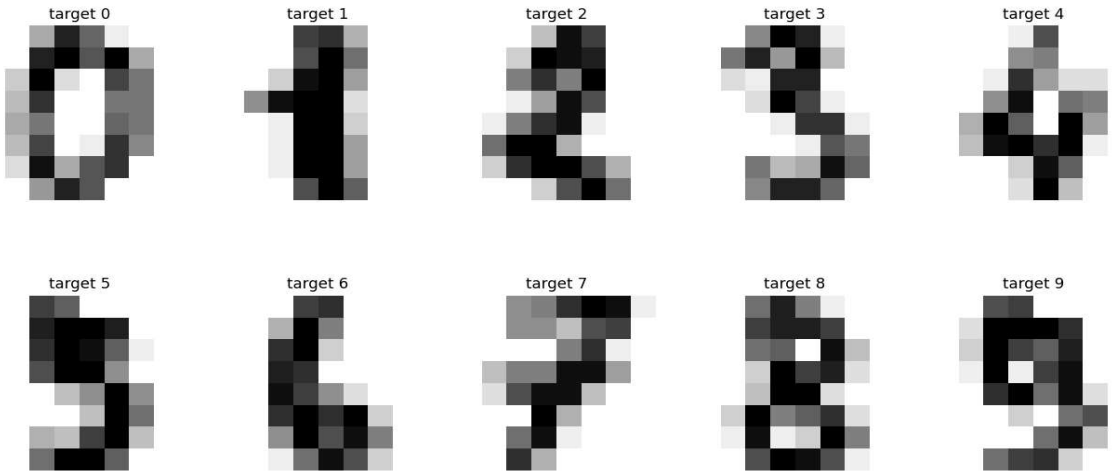
Output:

```
====================== RESTART: G:/IPMV/SVM/SVM 3.4.23.py ==============
========
Digit dataset keys
 dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names',
'images', 'DESCR'])
shape of data: (1797, 64)
```




```
import cv2
import numpy as np
import matplotlib.pylab as plt
from sklearn import datasets, svm, metrics
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

digits=datasets.load_digits()
print('Digit dataset keys\n {}'.format(digits.keys()))

digits=load_digits()
print('shape of data:',digits.data.shape)
#plt.gray()
#plt.matshow(digits.images[0])

print('dataset target name:\n{}\n'.format(digits.target_names))
print('shape of dataset:{}'.format(digits.data.shape))
print('shape of target:{}'.format(digits.target.shape))
print('shape of images:',digits.images.shape)

for i in range (10):
    plt.subplot(2,5,i+1)
    plt.axis('off')
    plt.imshow(digits.images[i],cmap='gray_r',interpolation='nearest')
    plt.title('target {}'.format(i))
plt.show()
```

Output:




```
import cv2
import numpy as np
import matplotlib.pylab as plt
from sklearn import datasets, svm, metrics
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

digits=datasets.load_digits()
print('Digit dataset keys\n {}'.format(digits.keys()))

digits=load_digits()
print('shape of data:',digits.data.shape)
#plt.gray()
#plt.matshow(digits.images[0])

print('dataset target name:\n{}\n'.format(digits.target_names))
print('shape of dataset:{}'.format(digits.data.shape))
print('shape of target:{}'.format(digits.target.shape))
print('shape of images:',digits.images.shape)

for i in range (10):
    plt.subplot(2,5,i+1)
    #plt.axis('off')
    plt.imshow(digits.images[i],cmap='gray_r',interpolation='nearest')
    plt.title('target {}'.format(i))

data_flattened=digits.images.reshape((len(digits.images),-1))
print('shape of data:',digits.data.shape)
x_train,x_test,y_train,y_test=train_test_split(data_flattened,digits.target,test_size=0.2,shuffle=False)
print('\ntraining data size={}'.format(x_train.shape))
print('training target size={}'.format(y_train.shape))
print('test data size={}'.format(x_test.shape))
print('test target size={}'.format(y_test.shape))

classifier=svm.SVC(gamma=0.5)
abc=classifier.fit(x_train,y_train)

y_pred=classifier.predict(x_test)
print('\ny=',y_pred[0])
print("\nConfusion Matrix:\n %s"%metrics.confusion_matrix(y_test, y_pred))

plt.show()
```

Output:

```
====================== RESTART: G:/IPMV/SVM/SVM 3.4.23.py ======================
Digit dataset keys
 dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DESCR'
shape of data: (1797, 64)
dataset target name:
[0 1 2 3 4 5 6 7 8 9]

shape of dataset:(1797, 64)
shape of target:(1797,)
shape of images: (1797, 8, 8)
shape of data: (1797, 64)

training data size=(898, 64)
training target size=(898,)
test data size=(899, 64)
test target size=(899,)

y= 3

Confusion Matrix:
 [[ 0  0  0 88  0  0  0  0  0  0]
 [ 0  0  0 91  0  0  0  0  0  0]
 [ 0  0  0 86  0  0  0  0  0  0]
 [ 0  0  0 91  0  0  0  0  0  0]
 [ 0  0  0 92  0  0  0  0  0  0]
 [ 0  0  0 91  0  0  0  0  0  0]
 [ 0  0  0 91  0  0  0  0  0  0]
 [ 0  0  0 89  0  0  0  0  0  0]
 [ 0  0  0 88  0  0  0  0  0  0]
 [ 0  0  0 92  0  0  0  0  0  0]]
```

```
import cv2
import numpy as np
import matplotlib.pylab as plt
from sklearn import datasets, svm, metrics
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

digits=datasets.load_digits()
print('Digit dataset keys\n {}'.format(digits.keys()))

digits=load_digits()
print('shape of data:',digits.data.shape)
#plt.gray()
#plt.matshow(digits.images[0])

print('dataset target name:\n{}\n'.format(digits.target_names))
print('shape of dataset:{}'.format(digits.data.shape))
print('shape of target:{}'.format(digits.target.shape))
print('shape of images:',digits.images.shape)

for i in range (10):
    plt.subplot(2,5,i+1)
    #plt.axis('off')
    plt.imshow(digits.images[i],cmap='gray_r',interpolation='nearest')
    plt.title('target {}'.format(i))

data_flattened=digits.images.reshape((len(digits.images),-1))
print('shape of data:',digits.data.shape)
x_train,x_test,y_train,y_test=train_test_split(data_flattened,digits.target,test_size=0.2,shuffle=True)
print('\ntraining data size={}'.format(x_train.shape))
print('training target size={}'.format(y_train.shape))
print('test data size={}'.format(x_test.shape))
print('test target size={}'.format(y_test.shape))

classifier=svm.SVC(gamma=0.5)
abc=classifier.fit(x_train,y_train)

y_pred=classifier.predict(x_test)
print('\ny=',y_pred[0])
print("\nConfusion Matrix:\n %s"%metrics.confusion_matrix(y_test, y_pred))

plt.show()
```

Output:

```
====================== RESTART: G:/IPMV/SVM/SVM 3.4.23.py ======================
Digit dataset keys
 dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DES
shape of data: (1797, 64)
dataset target name:
[0 1 2 3 4 5 6 7 8 9]

shape of dataset:(1797, 64)
shape of target:(1797,)
shape of images: (1797, 8, 8)
shape of data: (1797, 64)

training data size=(1437, 64)
training target size=(1437,)
test data size=(360, 64)
test target size=(360,)

y= 9

Confusion Matrix:
 [[ 0  0  0  0  0  0  0  0  0 45]
 [ 0  0  0  0  0  0  0  0  0 39]
 [ 0  0  0  0  0  0  0  0  0 38]
 [ 0  0  0  0  0  0  0  0  0 37]
 [ 0  0  0  0  0  0  0  0  0 39]
 [ 0  0  0  0  0  0  0  0  0 37]
 [ 0  0  0  0  0  0  0  0  0 40]
 [ 0  0  0  0  0  0  0  0  0 33]
 [ 0  0  0  0  0  0  0  0  0 24]
 [ 0  0  0  0  0  0  0  0  0 28]]
```