Chapter 8 - Demo svm.SVC

In [0]:
```
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
```

In [0]:
```
digits = datasets.load_digits()
```

In [0]:
```
digits.data
```

Out[3]:
```
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 [0]:
```
digits.target
```

Out[4]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

In [0]:
```
digits.target.size
```

Out[5]:
```
1797
```

In [0]:
```
X,y = digits.data, digits.target
```

In [0]:
```
from sklearn.model_selection import train_test_split
```

In [0]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

In [0]:
```
from sklearn import svm
```

In [0]:
```
clf = svm.SVC(gamma=0.001, C=100) # các tham số cho mô hình hoạt động tốt hơn:
```

In [0]:
```
clf.fit(X_train,y_train)
```

Out[11]:
```
SVC(C=100, break_ties=False, 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)
```

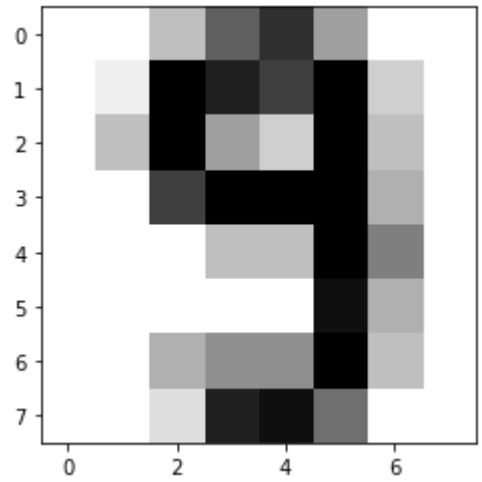
In [0]:
```
y_pred = clf.predict(X_test)
#y_pred
```

In [0]:
```
#y_test
```

In [0]:
```
from sklearn.metrics import accuracy_score
print("Accuracy is ", accuracy_score(y_test,y_pred)*100,"%")
```

```
Accuracy is  99.07407407407408 %
```

In [0]:
```
plt.imshow(digits.images[-5], cmap=plt.cm.gray_r, interpolation='nearest')
plt.show()
```



In [0]:
```
digits.data[-5]
```

Out[16]:
```
array([ 0.,  0.,  4., 10., 13.,  6.,  0.,  0.,  0.,  1., 16., 14., 12.,
       16.,  3.,  0.,  0.,  4., 16.,  6.,  3., 16.,  4.,  0.,  0.,  0.,
       12., 16., 16., 16.,  5.,  0.,  0.,  0.,  0.,  4.,  4., 16.,  8.,
        0.,  0.,  0.,  0.,  0.,  0., 15.,  5.,  0.,  0.,  0.,  5.,  7.,
        7., 16.,  4.,  0.,  0.,  0.,  2., 14., 15.,  9.,  0.,  0.])
```

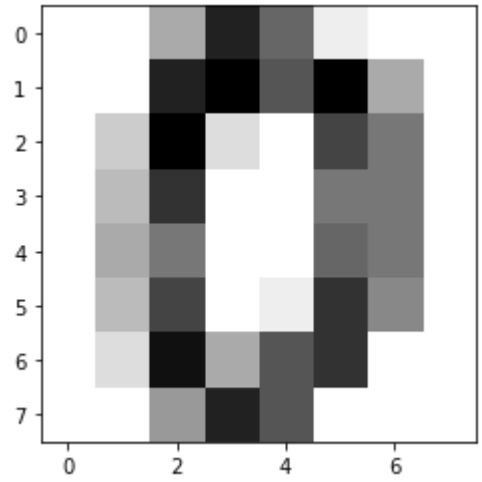
In [0]:
```
type(digits.data[-5])
```

Out[17]:
```
numpy.ndarray
```

In [0]:
```
num = clf.predict(np.array([digits.data[-5]]))
num
```

Out[18]:
```
array([9])
```

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



In [0]:
```
digits.data[0]
```

Out[20]:
```
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 [0]:
```
type(digits.data[0])
```

Out[21]:
```
numpy.ndarray
```

In [0]:
```
num = clf.predict(np.array([digits.data[0]]))
num
```

Out[22]:
```
array([0])
```

In [0]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[69  0  0  0  0  0  0  0  0  0]
 [ 0 58  0  0  0  0  0  0  0  0]
 [ 0  0 48  0  0  0  0  0  0  0]
 [ 0  0  0 57  0  0  0  0  0  0]
 [ 0  0  0  0 48  0  0  0  0  0]
 [ 0  0  0  0  0 54  1  0  0  1]
 [ 0  1  0  0  0  0 56  0  0  0]
 [ 0  0  0  0  0  0  0 51  0  0]
 [ 0  2  0  0  0  0  0  0 39  0]
 [ 0  0  0  0  0  0  0  0  0 55]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        69
           1       0.95      1.00      0.97        58
           2       1.00      1.00      1.00        48
           3       1.00      1.00      1.00        57
           4       1.00      1.00      1.00        48
           5       1.00      0.96      0.98        56
           6       0.98      0.98      0.98        57
           7       1.00      1.00      1.00        51
           8       1.00      0.95      0.97        41
           9       0.98      1.00      0.99        55

    accuracy                           0.99       540
   macro avg       0.99      0.99      0.99       540
weighted avg       0.99      0.99      0.99       540
```

In [0]:
```

```