Day - 5 | Handwritten Digit Recognition | SVM

Importing Basic Libraries

```
import numpy as np
from sklearn.datasets import load_digits
```

Load Dataset

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64
Features	integers 0-16

+ Code + Text

```
dataset = load_digits()
```

Summarize Dataset

```
print(dataset.data)
print(dataset.target)

print(dataset.data.shape)
print(dataset.images.shape)

dataimageLength = len(dataset.images)
print(dataimageLength)
```

```
[[ 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.]]
[0 1 2 ... 8 9 8]
(1797, 64)
(1797, 8, 8)
1797
```

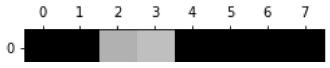
Visualize the Dataset

```
n=9 #No. of Sample out of Samples total 1797

import matplotlib.pyplot as plt
plt.gray()
plt.matshow(dataset.images[n])
plt.show()

dataset.images[n]
```

```
<Figure size 432x288 with 0 Axes>
```



Segregate Dataset into X(Input/IndependentVariable) & Y(Output/DependentVariable)

Input - Pixel | Output - Class


```
X = dataset.images.reshape((dataimageLength,-1))
X
```

```
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.]])
```

```
Y = dataset.target
Y
```

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

Splitting Dataset into Train & Test

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
print(X_train.shape)
print(X_test.shape)
```

```
(1347, 64)
(450, 64)
```

Training

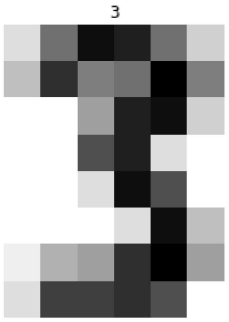
```
from sklearn import svm
model = svm.SVC(kernel='linear')
model.fit(X_train,y_train)
```

```
SVC(kernel='linear')
```

Predicting, what the digit is from Test Data

```
n=13
result = model.predict(dataset.images[n].reshape((1,-1)))
plt.imshow(dataset.images[n], cmap=plt.cm.gray_r, interpolation='nearest')
print(result)
print("\n")
plt.axis('off')
plt.title('%i' %result)
plt.show()
```

```
[3]
```



Prediction for Test Data

```
y_pred = model.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

Evaluate Model - Accuracy Score

```
from sklearn.metrics import accuracy_score
print("Accuracy of the Model: {0}%".format(accuracy_score(y_test, y_pred)*100))
```

```
Accuracy of the Model: 97.11111111111111%
```

Play with the Different Method

```
from sklearn import svm
model1 = svm.SVC(kernel='linear')
model2 = svm.SVC(kernel='rbf')
model3 = svm.SVC(gamma=0.001)
model4 = svm.SVC(gamma=0.001,C=0.1)

model1.fit(X_train,y_train)
model2.fit(X_train,y_train)
model3.fit(X_train,y_train)
model4.fit(X_train,y_train)

y_predModel1 = model1.predict(X_test)
y_predModel2 = model2.predict(X_test)
y_predModel3 = model3.predict(X_test)
y_predModel4 = model4.predict(X_test)

print("Accuracy of the Model 1: {0}%".format(accuracy_score(y_test, y_predModel1)*100))
print("Accuracy of the Model 2: {0}%".format(accuracy_score(y_test, y_predModel2)*100))
print("Accuracy of the Model 3: {0}%".format(accuracy_score(y_test, y_predModel3)*100))
print("Accuracy of the Model 4: {0}%".format(accuracy_score(y_test, y_predModel4)*100))
```

```
Accuracy of the Model 1: 97.11111111111111%
Accuracy of the Model 2: 99.11111111111111%
Accuracy of the Model 3: 99.55555555555556%
Accuracy of the Model 4: 96.66666666666667%
```

Double-click (or enter) to edit