Name : Tavhare Ruchita Sharad

class : BE AI&DS

Roll No.: 61

Subject: Computer Laboratory-I

Title:Implementation of Support Vector Machines (SVM) for classifying images of handwritten digits into their respective numerical classes (0 to 9).

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
```

In [8]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [10]:
```
from sklearn.datasets import load_digits
digits = load_digits(n_class=10)
digits
```

Out[10]:

```
{'data': 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.]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'frame': None,
 'feature_names': ['pixel_0_0',
  'pixel_0_1',
  'pixel_0_2',
  'pixel_0_3',
  'pixel_0_4',
  'pixel_0_5',
  'pixel_0_6',
  'pixel_0_7',
  'pixel_1_0',
  'pixel_1_1',
  'pixel_1_2',
  'pixel_1_3',
  'pixel_1_4',
  'pixel_1_5',
  'pixel_1_6',
  'pixel_1_7',
  'pixel_2_0',
  'pixel_2_1',
  'pixel_2_2',
  'pixel_2_3',
  'pixel_2_4',
  'pixel_2_5',
  'pixel_2_6',
  'pixel_2_7',
  'pixel_3_0',
  'pixel_3_1',
  'pixel_3_2',
  'pixel_3_3',
  'pixel_3_4',
  'pixel_3_5',
  'pixel_3_6',
  'pixel_3_7',
  'pixel_4_0',
  'pixel_4_1',
  'pixel_4_2',
  'pixel_4_3',
  'pixel_4_4',
  'pixel_4_5',
  'pixel_4_6',
  'pixel_4_7',
  'pixel_5_0',
  'pixel_5_1',
  'pixel_5_2',
  'pixel_5_3',
  'pixel_5_4',
```

```
 'pixel_5_5',
 'pixel_5_6',
 'pixel_5_7',
 'pixel_6_0',
 'pixel_6_1',
 'pixel_6_2',
 'pixel_6_3',
 'pixel_6_4',
 'pixel_6_5',
 'pixel_6_6',
 'pixel_6_7',
 'pixel_7_0',
 'pixel_7_1',
 'pixel_7_2',
 'pixel_7_3',
 'pixel_7_4',
 'pixel_7_5',
 'pixel_7_6',
 'pixel_7_7'],
'target_names': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
'images': array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ..., 15.,  5.,  0.],
        [ 0.,  3., 15., ..., 11.,  8.,  0.],
        ...,
        [ 0.,  4., 11., ..., 12.,  7.,  0.],
        [ 0.,  2., 14., ..., 12.,  0.,  0.],
        [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

       [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
        [ 0.,  0.,  0., ...,  9.,  0.,  0.],
        [ 0.,  0.,  3., ...,  6.,  0.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

       [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
        [ 0.,  0.,  3., ..., 14.,  0.,  0.],
        [ 0.,  0.,  8., ..., 16.,  0.,  0.],
        ...,
        [ 0.,  9., 16., ...,  0.,  0.,  0.],
        [ 0.,  3., 13., ..., 11.,  5.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.]],

       ...,

       [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ...,  2.,  1.,  0.],
        [ 0.,  0., 16., ..., 16.,  5.,  0.],
        ...,
        [ 0.,  0., 16., ..., 15.,  0.,  0.],
        [ 0.,  0., 15., ..., 16.,  0.,  0.],
        [ 0.,  0.,  2., ...,  6.,  0.,  0.]],
```

```
        [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
         [ 0.,  0., 14., ..., 15.,  1.,  0.],
         [ 0.,  4., 16., ..., 16.,  7.,  0.],
         ...,
         [ 0.,  0.,  0., ..., 16.,  2.,  0.],
         [ 0.,  0.,  4., ..., 16.,  2.,  0.],
         [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

        [[ 0.,  0., 10., ...,  1.,  0.,  0.],
         [ 0.,  2., 16., ...,  1.,  0.,  0.],
         [ 0.,  0., 15., ..., 15.,  0.,  0.],
         ...,
         [ 0.,  4., 16., ..., 16.,  6.,  0.],
         [ 0.,  8., 16., ..., 16.,  8.,  0.],
         [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digits d
ataset\n--------------------------------------------------\n\n**Data Set Char
acteristics:**\n\n:Number of Instances: 1797\n:Number of Attributes: 64\n:Att
ribute 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:Da
te: July; 1998\n\nThis is a copy of the test set of the UCI ML hand-written d
igits datasets\nhttps://archive.ics.uci.edu/ml/datasets/Optical+Recognition+o
f+Handwritten+Digits\n\nThe data set contains images of hand-written digits:
10 classes where\neach class refers to a digit.\n\nPreprocessing programs mad
e available by NIST were used to extract\nnormalized bitmaps of handwritten d
igits from a preprinted form. From a\ntotal of 43 people, 30 contributed to t
he training set 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 where each element is an i
nteger in the range\n0..16. This reduces dimensionality and gives invariance
to small\ndistortions.\n\nFor info on NIST preprocessing routines, see M. D.
Garris, J. L. Blue, G.\nT. Candela, D. L. Dimmick, J. Geist, P. J. Grother,
S. A. Janet, and C.\nL. Wilson, NIST Form-Based Handprint Recognition System,
NISTIR 5469,\n1994.\n\n.. dropdown:: References\n\n  - C. Kaynak (1995) Metho
ds of Combining Multiple Classifiers and Their\n    Applications to Handwritt
en Digit Recognition, MSc Thesis, Institute of\n    Graduate Studies in Scien
ce and Engineering, Bogazici University.\n  - E. Alpaydin, C. Kaynak (1998) C
ascading Classifiers, Kybernetika.\n  - Ken Tang and Ponnuthurai N. Suganthan
and Xi Yao and A. Kai Qin.\n    Linear dimensionalityreduction using relevanc
e weighted LDA. School of\n    Electrical and Electronic Engineering Nanyang
Technological University.\n    2005.\n  - Claudio Gentile. A New Approximate
Maximal Margin Classification\n    Algorithm. NIPS. 2000.\n"}
```

In [11]:
```
digits['data'][0].reshape(8,8)
```

Out[11]:
```
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 [12]:
```
digits['images'][1]
```

Out[12]:
```
array([[ 0.,  0.,  0., 12., 13.,  5.,  0.,  0.],
       [ 0.,  0.,  0., 11., 16.,  9.,  0.,  0.],
       [ 0.,  0.,  3., 15., 16.,  6.,  0.,  0.],
       [ 0.,  7., 15., 16., 16.,  2.,  0.,  0.],
       [ 0.,  0.,  1., 16., 16.,  3.,  0.,  0.],
       [ 0.,  0.,  1., 16., 16.,  6.,  0.,  0.],
       [ 0.,  0.,  1., 16., 16.,  6.,  0.,  0.],
       [ 0.,  0.,  0., 11., 16., 10.,  0.,  0.]])
```

In [16]:
```
digits['target'][0:9]
```

Out[16]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

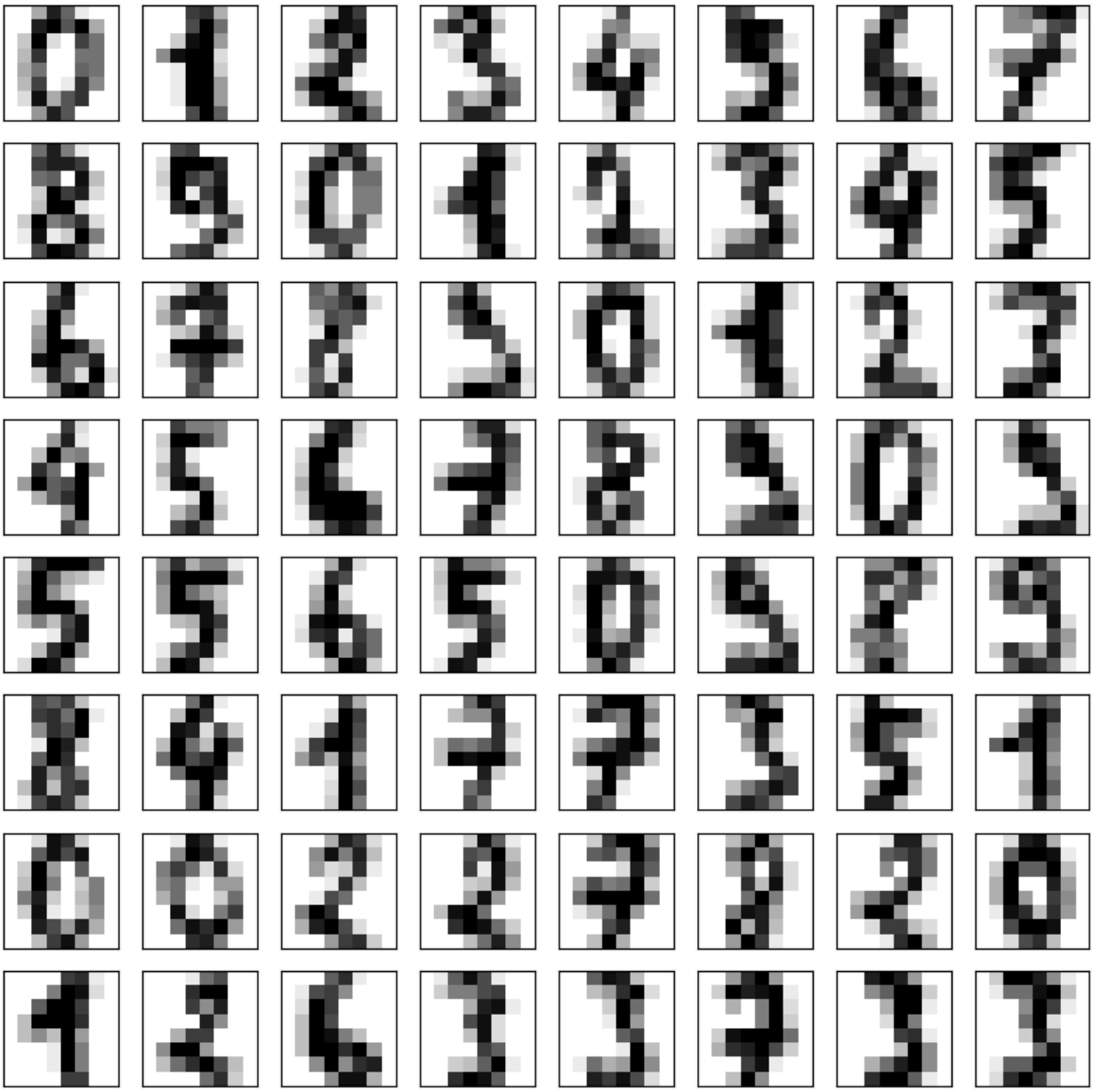
In [18]:
```
digits['target'][0]
```

Out[18]:
```
0
```

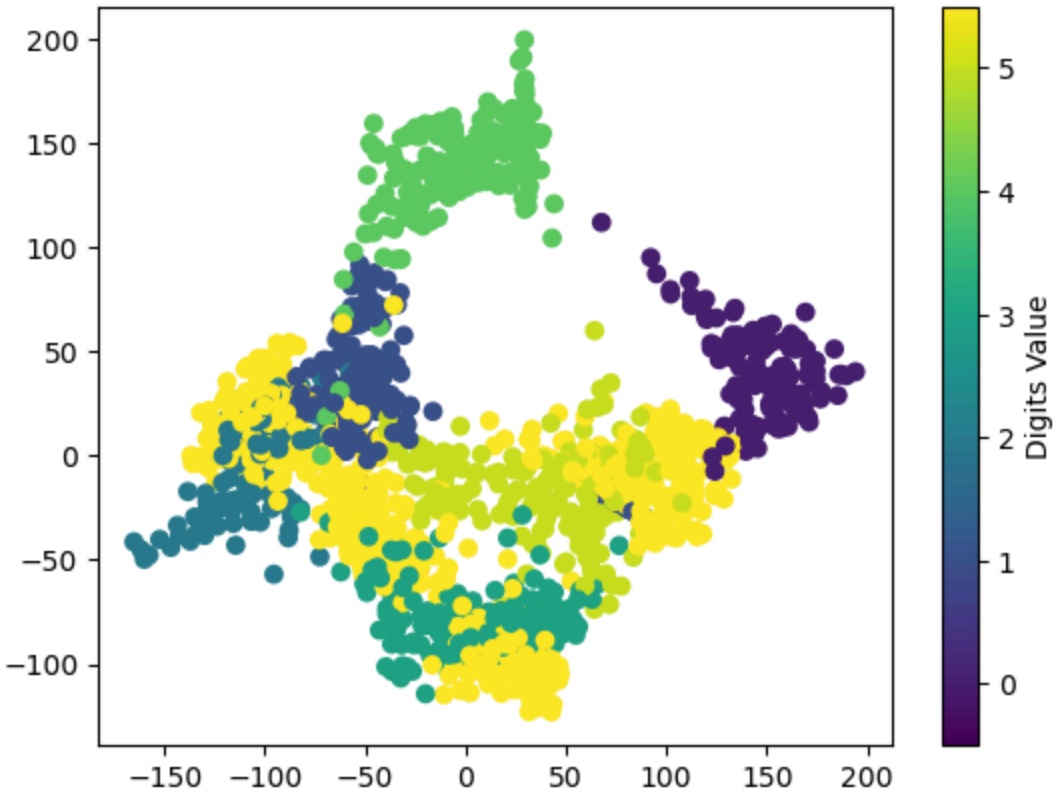
In [20]:
```
digits.images[0]
```

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 [22]:
```
fig, ax = plt.subplots(8,8, figsize=(10,10))
for i, axi in enumerate(ax.flat):
    axi.imshow(digits.images[i], cmap='binary')
    axi.set(xticks=[], yticks=[])
```



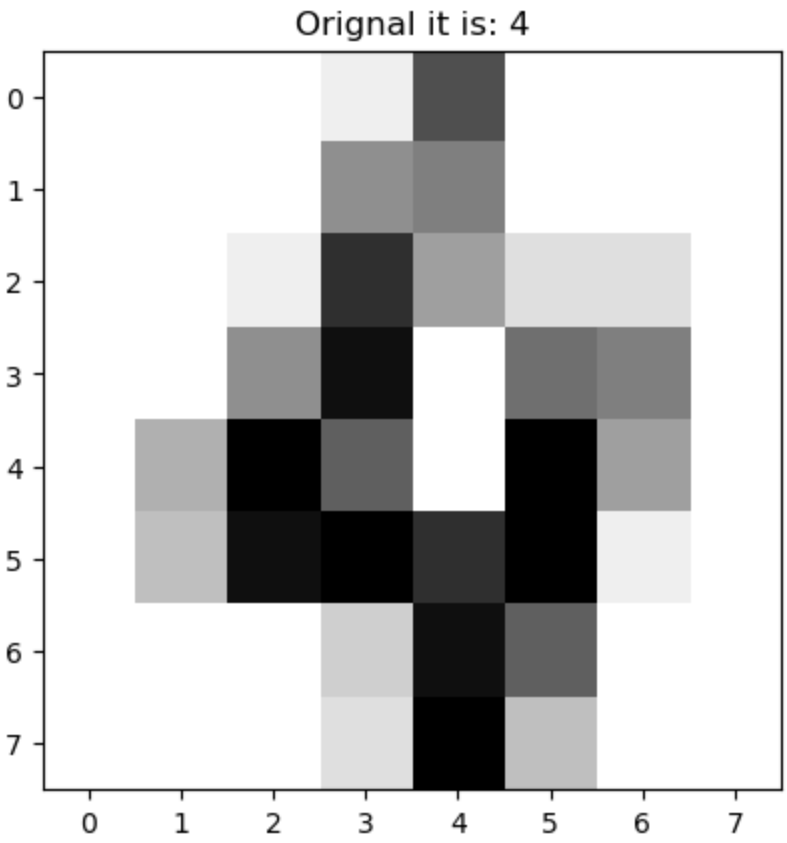
In [23]:
```
from sklearn.manifold import Isomap
iso = Isomap(n_components=2)
projection = iso.fit_transform(digits.data) # digits.data - 64 dimensions to 2
plt.scatter(projection[:, 0], projection[:, 1], c=digits.target, cmap="viridis
plt.colorbar(ticks=range(10), label='Digits Value')
plt.clim(-0.5, 5.5)
```



In [25]:
```
print(projection[:, 0][70], projection[:, 1][70])
```

```
-57.04584000890165 62.024221615823386
```

In [26]:
```
def view_digit(index):
    plt.imshow(digits.images[index] , cmap = plt.cm.gray_r)
    plt.title('Orignal it is: '+ str(digits.target[index]))
    plt.show()
view_digit(4)
```



In [27]:
```
main_data = digits['data']
targets = digits['target']
from sklearn import svm
svc = svm.SVC(gamma=0.001 , C = 100)
```

In [28]:
```
svc.fit(main_data[:1500] , targets[:1500])
predictions = svc.predict(main_data[1501:])
list(zip(predictions , targets[1501:]))
```

Out[28]:
```
[(7, 7),
 (4, 4),
 (6, 6),
 (3, 3),
 (1, 1),
 (3, 3),
 (9, 9),
 (1, 1),
 (7, 7),
 (6, 6),
 (8, 8),
 (4, 4),
 (3, 3),
 (1, 1),
 (4, 4),
 (0, 0),
 (5, 5),
 (3, 3),
 (6, 6),
 (9, 9),
 (6, 6),
 (1, 1),
 (7, 7),
 (5, 5),
 (4, 4),
 (4, 4),
 (7, 7),
 (2, 2),
 (8, 8),
 (2, 2),
 (2, 2),
 (5, 5),
 (7, 7),
 (9, 9),
 (5, 5),
 (4, 4),
 (8, 8),
 (8, 8),
 (4, 4),
 (9, 9),
 (0, 0),
 (8, 8),
 (9, 9),
 (8, 8),
 (0, 0),
 (1, 1),
 (2, 2),
 (3, 3),
 (4, 4),
 (5, 5),
 (6, 6),
 (7, 7),
 (1, 8),
 (9, 9),
```

(0, 0),
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(9, 9),
(0, 0),
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(1, 8),
(9, 9),
(4, 0),
(9, 9),
(5, 5),
(5, 5),
(6, 6),
(5, 5),
(0, 0),
(9, 9),
(8, 8),
(9, 9),
(8, 8),
(4, 4),
(1, 1),
(7, 7),
(7, 7),
(3, 3),
(5, 5),
(1, 1),
(0, 0),
(0, 0),
(2, 2),
(2, 2),
(7, 7),
(8, 8),
(2, 2),
(0, 0),
(1, 1),
(2, 2),
(6, 6),
(8, 3),
(3, 3),
(7, 7),
(3, 3),
(3, 3),
(4, 4),
(6, 6),

(6, 6),
(6, 6),
(9, 4),
(9, 9),
(1, 1),
(5, 5),
(0, 0),
(9, 9),
(5, 5),
(2, 2),
(8, 8),
(0, 0),
(1, 1),
(7, 7),
(6, 6),
(3, 3),
(2, 2),
(1, 1),
(7, 7),
(9, 4),
(6, 6),
(3, 3),
(1, 1),
(3, 3),
(9, 9),
(1, 1),
(7, 7),
(6, 6),
(8, 8),
(4, 4),
(3, 3),
(1, 1),
(4, 4),
(0, 0),
(5, 5),
(3, 3),
(6, 6),
(9, 9),
(6, 6),
(1, 1),
(7, 7),
(5, 5),
(4, 4),
(4, 4),
(7, 7),
(2, 2),
(2, 2),
(5, 5),
(7, 7),
(8, 9),
(5, 5),
(9, 4),
(4, 4),
(5, 9),

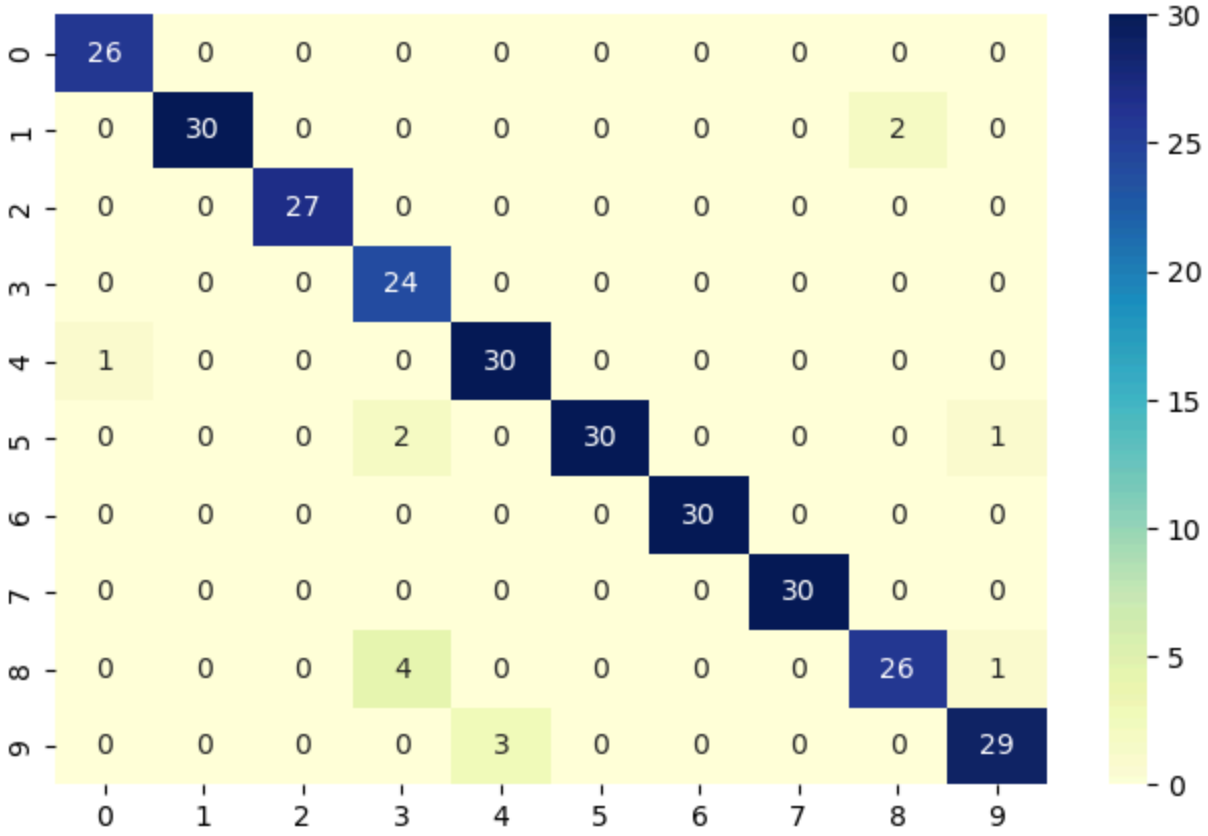
(0, 0),
(8, 8),
(9, 9),
(8, 8),
(0, 0),
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(0, 0),
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(0, 0),
(1, 1),
(2, 2),
(8, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(0, 0),
(9, 9),
(5, 5),
(5, 5),
(6, 6),
(5, 5),
(0, 0),
(9, 9),
(8, 8),
(9, 9),
(8, 8),
(4, 4),
(1, 1),
(7, 7),
(7, 7),
(3, 3),
(5, 5),
(1, 1),
(0, 0),
(0, 0),

```
(2, 2),
(2, 2),
(7, 7),
(8, 8),
(2, 2),
(0, 0),
(1, 1),
(2, 2),
(6, 6),
(8, 3),
(8, 3),
(7, 7),
(5, 3),
(3, 3),
(4, 4),
(6, 6),
(6, 6),
(6, 6),
(4, 4),
(9, 9),
(1, 1),
(5, 5),
(0, 0),
(9, 9),
(5, 5),
(2, 2),
(8, 8),
(2, 2),
(0, 0),
(0, 0),
(1, 1),
(7, 7),
(6, 6),
(3, 3),
(2, 2),
(1, 1),
(7, 7),
(4, 4),
(6, 6),
(3, 3),
(1, 1),
(3, 3),
(9, 9),
(1, 1),
(7, 7),
(6, 6),
(8, 8),
(4, 4),
(5, 3),
(1, 1),
(4, 4),
(0, 0),
(5, 5),
(3, 3),
```

```
 (6, 6),
 (9, 9),
 (6, 6),
 (1, 1),
 (7, 7),
 (5, 5),
 (4, 4),
 (4, 4),
 (7, 7),
 (2, 2),
 (8, 8),
 (2, 2),
 (2, 2),
 (5, 5),
 (7, 7),
 (9, 9),
 (5, 5),
 (4, 4),
 (8, 8),
 (8, 8),
 (4, 4),
 (9, 9),
 (0, 0),
 (8, 8),
 (9, 9),
 (8, 8)]
```

In [33]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
cm = confusion_matrix(predictions, targets[1501:])
conf_matrix = pd.DataFrame(data = cm)
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu");
```



In [36]:
```
cm
```

Out[36]:
```
array([[26,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 30,  0,  0,  0,  0,  0,  0,  2,  0],
       [ 0,  0, 27,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 24,  0,  0,  0,  0,  0,  0],
       [ 1,  0,  0,  0, 30,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  2,  0, 30,  0,  0,  0,  1],
       [ 0,  0,  0,  0,  0,  0, 30,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 30,  0,  0],
       [ 0,  0,  0,  4,  0,  0,  0,  0, 26,  1],
       [ 0,  0,  0,  0,  3,  0,  0,  0,  0, 29]], dtype=int64)
```

In [38]:
```
from sklearn.metrics import classification_report
print(classification_report(predictions, targets[1501:]))
```

	precision	recall	f1-score	support
0	0.96	1.00	0.98	26
1	1.00	0.94	0.97	32
2	1.00	1.00	1.00	27
3	0.80	1.00	0.89	24
4	0.91	0.97	0.94	31
5	1.00	0.91	0.95	33
6	1.00	1.00	1.00	30
7	1.00	1.00	1.00	30
8	0.93	0.84	0.88	31
9	0.94	0.91	0.92	32
accuracy			0.95	296
macro avg	0.95	0.96	0.95	296
weighted avg	0.96	0.95	0.95	296

In [ ]: