# CL1-03

July 24, 2025

[ ]:
```
'''NAME:Aher Swami Sandip
ROLL NO.01
COURSE: AI&DS
CLASS: BE
SUB:Computer Laboratory-I (Machine Learning)'''
```

[ ]:
```
'''
PRACTICAL NO-03:
    Implementation of Support Vector Machines (SVM) for classifying images of␣
 ↪handwritten digits into their respective numerical classes (0 to 9).
'''
```

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

[2]:
```
from sklearn.datasets import load_digits
digits = load_digits(n_class = 10)
```

[3]:
```
digits
```

[3]:
```
{'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
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 datasets\nhttps://archive.ics.uci.edu/ml/datasets/Optical+Re
cognition+of+Handwritten+Digits\n\nThe data set contains images of hand-written
digits: 10 classes where\neach class refers to a digit.\n\nPreprocessing
programs made available by NIST were used to extract\nnormalized bitmaps of
handwritten digits from a preprinted form. From a\ntotal of 43 people, 30
contributed to the 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 integer 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)
Methods of Combining Multiple Classifiers and Their\n    Applications to
Handwritten Digit Recognition, MSc Thesis, Institute of\n    Graduate Studies in
Science and Engineering, Bogazici 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
Technological University.\n    2005.\n  - Claudio Gentile. A New Approximate
Maximal Margin Classification\n    Algorithm. NIPS. 2000.\n"}
```

[4]:
```
digits['data'][0].reshape(8,8)
```

[4]:
```
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.]])
```

[6]:
```
digits['data'][0]
```

[6]:
```
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.])
```

[7]:
```
digits['images'][1]
```

[7]:
```
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.]])
```

[8]:
```
digits['target'][0:9]
```

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

[15]:
```
digits['target'][0]
```

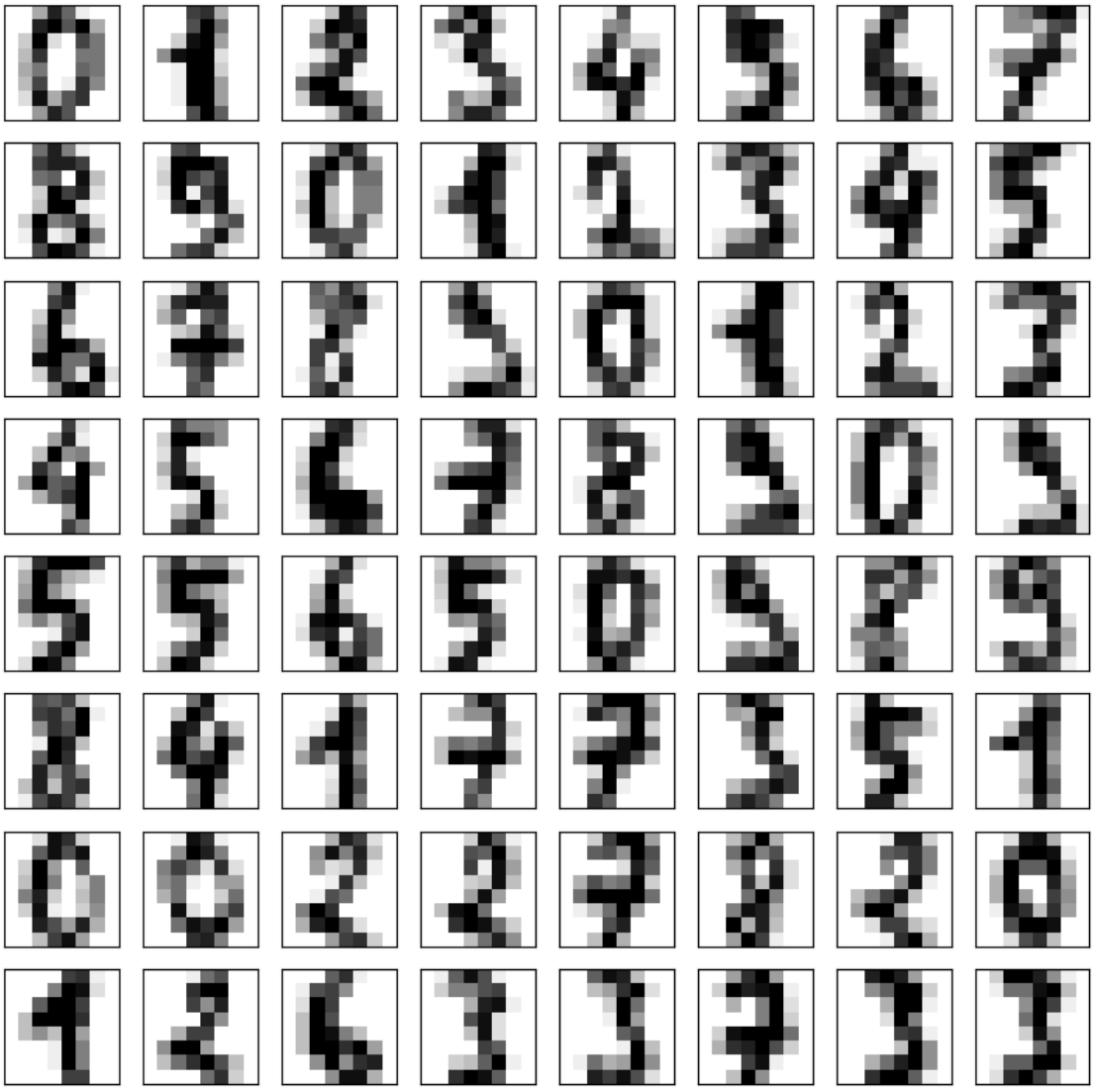
[15]:
```
0
```

[17]:
```
digits.images[0]
```

[17]:
```
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.]])
```

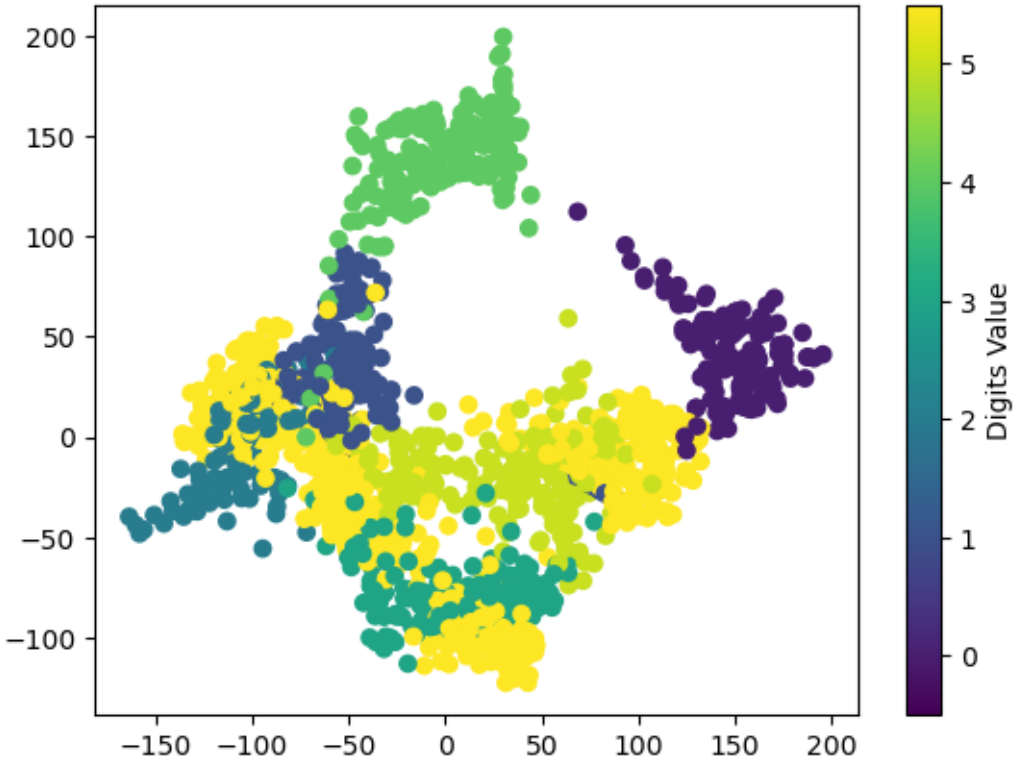
[19]:
```
# Each Digit is represented in digits.images as a matrix 0of 8x8 = 64 pixels.
  ↪Each of the 64 values represent a grayscale
# The Grayscale are then plotted in the right scale by the imshow method.
```

[21]:
```
fig, ax = plt.subplots(8, 8, figsize=(10, 10))

# Loop through each axis and plot the corresponding image
for i, axi in enumerate(ax.flat):
    axi.imshow(digits.images[i], cmap='binary')
    axi.set_xticks([])  # Remove x-ticks
    axi.set_yticks([])  # Remove y-ticks

plt.show()
```



[22]:
```
# Plotting - Clustering the data points after using Manifold Learning
```

[23]:
```
import warnings
warnings.filterwarnings("ignore")

from sklearn.manifold import Isomap
iso = Isomap(n_components = 2)
projection = iso.fit_transform(digits.data)
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)
```




[27]:
```
projection[:, 0][70]
```

[27]: -56.60683580684849

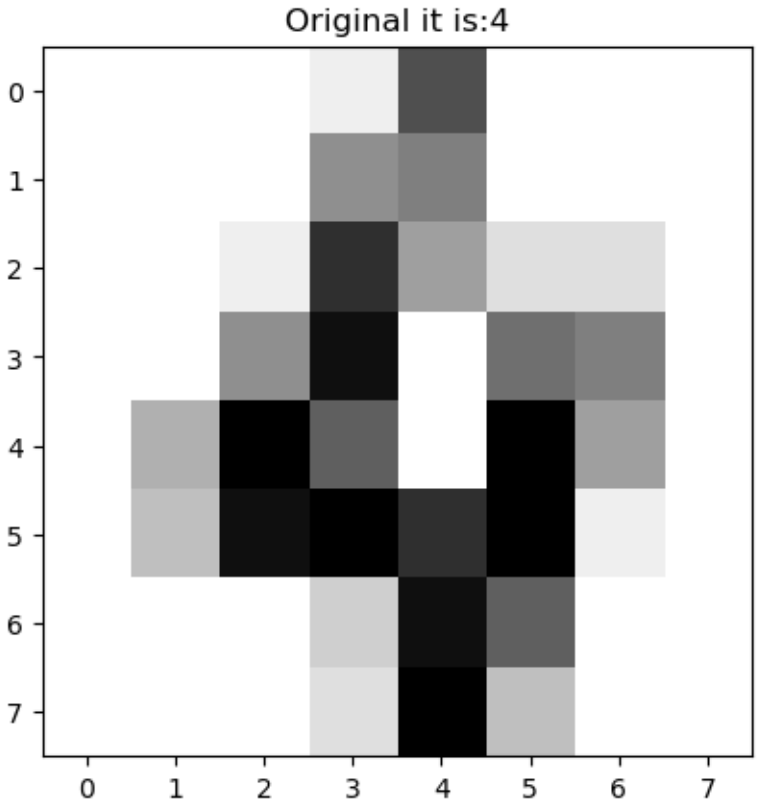
[29]:
```
projection[:, 1][70]
```

[29]: 61.95022367117498

[31]:
```
def view_digit(index):
    plt.imshow(digits.images[index], cmap = plt.cm.gray_r)
    plt.title('Original it is:' + str(digits.target[index]))
```

```
    plt.show()
```

[33]:
```
view_digit(4)
```




[ ]:
```

```