```
# importing the hand written digit dataset
from sklearn import datasets

# digit contain the dataset
digits = datasets.load_digits()

# dir function use to display the attributes of the dataset
dir(digits)
```

```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

```
# outputting the picture value as a series of numbers
print(digits.images[0])
```

```
[[ 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.]]
```

```
# importing the matplotlib libraries pyplot function
import matplotlib.pyplot as plt
# defining the function plot_multi

def plot_multi(i):
    nplots = 16
    fig = plt.figure(figsize=(15, 15))
    for j in range(nplots):
        plt.subplot(4, 4, j+1)
        plt.imshow(digits.images[i+j], cmap='binary')
        plt.title(digits.target[i+j])
        plt.axis('off')
    # printing the each digits in the dataset.
    plt.show()

    plot_multi(0)
```

```
# converting the 2 dimensional array to one dimensional array
y = digits.target
x = digits.images.reshape((len(digits.images), -1))

# gives the shape of the data
x.shape
```

```
(1797, 64)
```

```
# printing the one-dimensional array's values
x[0]
```

```
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.])
```

```
# Very first 1000 photographs and
# labels will be used in training.
x_train = x[:1000]
y_train = y[:1000]

# The leftover dataset will be utilised to
# test the network's performance later on.
x_test = x[1000:]
y_test = y[1000:]
```

```
# importing the MLP classifier from sklearn
from sklearn.neural_network import MLPClassifier

# calling the MLP classifier with specific parameters
mlp = MLPClassifier(hidden_layer_sizes=(15,),
                    activation='logistic',
                    alpha=1e-4, solver='sgd',
                    tol=1e-4, random_state=1,
                    learning_rate_init=.1,
                    verbose=True)
```

```
mlp.fit(x_train, y_train)
```

```
Iteration 1, loss = 2.22958289
Iteration 2, loss = 1.91207743
Iteration 3, loss = 1.62507727
Iteration 4, loss = 1.32649842
Iteration 5, loss = 1.06100535
Iteration 6, loss = 0.83995513
Iteration 7, loss = 0.67806075
Iteration 8, loss = 0.55175832
Iteration 9, loss = 0.45840445
Iteration 10, loss = 0.39149735
Iteration 11, loss = 0.33676351
Iteration 12, loss = 0.29059880
Iteration 13, loss = 0.25437208
Iteration 14, loss = 0.22838372
Iteration 15, loss = 0.20200554
Iteration 16, loss = 0.18186565
Iteration 17, loss = 0.16461183
Iteration 18, loss = 0.14990228
Iteration 19, loss = 0.13892154
Iteration 20, loss = 0.12833784
Iteration 21, loss = 0.12138920
Iteration 22, loss = 0.11407971
Iteration 23, loss = 0.10677664
Iteration 24, loss = 0.10037149
Iteration 25, loss = 0.09593187
Iteration 26, loss = 0.09250135
Iteration 27, loss = 0.08676698
Iteration 28, loss = 0.08356043
Iteration 29, loss = 0.08209789
Iteration 30, loss = 0.07649168
Iteration 31, loss = 0.07410898
Iteration 32, loss = 0.07126869
Iteration 33, loss = 0.06926956
Iteration 34, loss = 0.06578496
Iteration 35, loss = 0.06374913
Iteration 36, loss = 0.06175492
Iteration 37, loss = 0.05975664
Iteration 38, loss = 0.05764485
Iteration 39, loss = 0.05623663
Iteration 40, loss = 0.05420966
Iteration 41, loss = 0.05413911
Iteration 42, loss = 0.05256140
Iteration 43, loss = 0.05020265
Iteration 44, loss = 0.04902779
Iteration 45, loss = 0.04788382
Iteration 46, loss = 0.04655532
Iteration 47, loss = 0.04586089
Iteration 48, loss = 0.04451758
Iteration 49, loss = 0.04341598
Iteration 50, loss = 0.04238096
Iteration 51, loss = 0.04162200
Iteration 52, loss = 0.04076839
Iteration 53, loss = 0.04003180
Iteration 54, loss = 0.03907774
Iteration 55, loss = 0.03815565
Iteration 56, loss = 0.03791975
Iteration 57, loss = 0.03706276
Iteration 58, loss = 0.03617874
Iteration 59, loss = 0.03593227
Iteration 60, loss = 0.03504175
Iteration 61, loss = 0.03441259
Iteration 62, loss = 0.03397449
Iteration 63, loss = 0.03326990
Iteration 64, loss = 0.03305025
Iteration 65, loss = 0.03244893
Iteration 66, loss = 0.03191504
Iteration 67, loss = 0.03132169
Iteration 68, loss = 0.03079707
Iteration 69, loss = 0.03044946
Iteration 70, loss = 0.03005546
Iteration 71, loss = 0.02960555
Iteration 72, loss = 0.02912799
Iteration 73, loss = 0.02859103
Iteration 74, loss = 0.02825959
Iteration 75, loss = 0.02788968
Iteration 76, loss = 0.02748725
Iteration 77, loss = 0.02721247
Iteration 78, loss = 0.02686225
Iteration 79, loss = 0.02635636
Iteration 80, loss = 0.02607439
Iteration 81, loss = 0.02577613
Iteration 82, loss = 0.02553642
Iteration 83, loss = 0.02518749
Iteration 84, loss = 0.02484300
```