MNIST KNN Classification

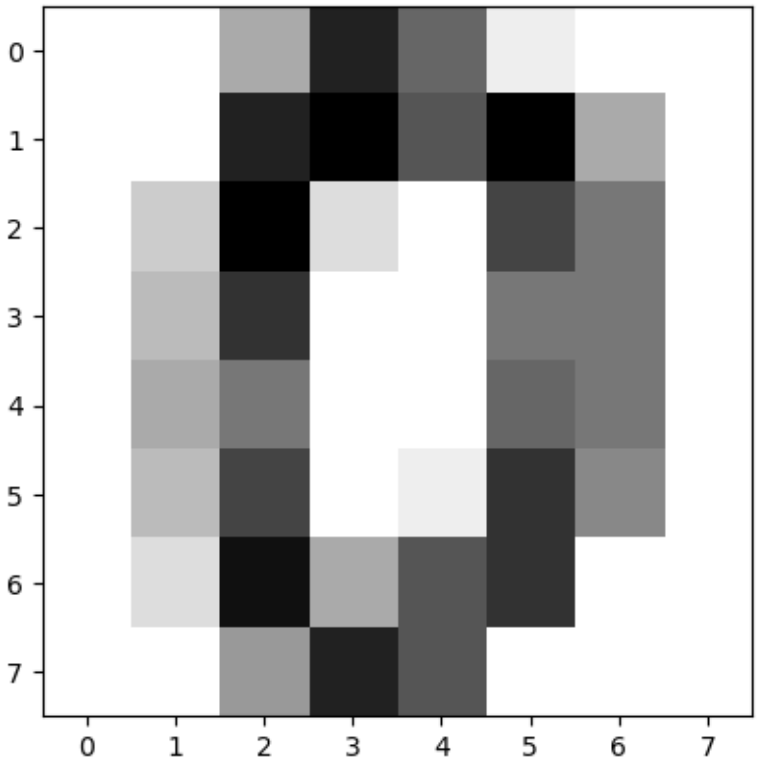
September 13, 2025

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

[2]:
```
data = datasets.load_digits()
data.images[0]
```

[2]:
```
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.]])
```

[3]:
```
plt.subplot()
plt.imshow(data.images[0], cmap = plt.cm.gray_r);
```



[4]:
```
X = data.images.reshape(len(data.images), -1)
X
```

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

[5]:
```
X.shape
```

[5]:
```
(1797, 64)
```

[6]:
```
y = data.target
y
```

[6]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

[7]:
```
len(y)
```

[7]: 1797

[8]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[9]:
```
knn_clf = KNeighborsClassifier(n_neighbors = 200)
knn_clf.fit(X[:1000], y[:1000])
```

[9]:
```
KNeighborsClassifier(n_neighbors=200)
```

Now we want to predict first 1000 datas then we will predict next 797 datas.

[10]:
```
from sklearn import metrics
```

[11]:
```
p = knn_clf.predict(X[:1000])
e = y[:1000]
print(metrics.classification_report(e,p))
```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        99
           1       0.88      0.55      0.67       102
           2       0.82      0.82      0.82       100
           3       0.75      0.90      0.82       104
           4       0.95      0.88      0.91        98
           5       0.96      0.74      0.84       100
           6       0.88      0.98      0.93       101
           7       0.83      0.99      0.90        99
           8       0.77      0.87      0.82        98
           9       0.71      0.66      0.68        99

    accuracy                           0.84      1000
   macro avg       0.84      0.84      0.83      1000
weighted avg       0.84      0.84      0.83      1000
```

[12]:
```
print(metrics.confusion_matrix(e,p))
```

```
[[99  0  0  0  0  0  0  0  0  0]
 [ 0 56 15  3  0  0  8  2 12  6]
 [ 0  3 82  4  0  0  0  2  9  0]
 [ 1  0  0 94  0  0  0  3  2  4]
 [ 1  2  0  0 86  0  3  6  0  0]
 [ 5  0  0  6  1 74  0  0  0 14]
 [ 1  0  0  0  0  0 99  0  1  0]
 [ 0  0  0  0  0  1  0 98  0  0]
 [ 0  1  3  4  0  1  2  0 85  2]
 [ 5  2  0 14  4  1  0  7  1 65]]
```

So the accuracy is 83%. So the error is high. Also on the datas that we train them. Thats awful.

[13]:
```
p = knn_clf.predict(X[1000:])
e = y[1000:]
print(metrics.classification_report(e,p))
```

```
              precision    recall  f1-score   support

           0       0.87      0.99      0.92        79
           1       0.96      0.57      0.72        80
           2       0.91      0.79      0.85        77
           3       0.62      0.87      0.73        79
           4       0.96      0.84      0.90        83
           5       0.83      0.78      0.81        82
           6       0.90      0.99      0.94        80
           7       0.84      0.95      0.89        80
           8       0.76      0.71      0.73        76
           9       0.62      0.62      0.62        81

    accuracy                           0.81       797
   macro avg       0.83      0.81      0.81       797
weighted avg       0.83      0.81      0.81       797
```

[14]:
```
print(metrics.confusion_matrix(e,p))
```

```
[[78  0  0  0  1  0  0  0  0  0]
 [ 0 46  1  3  2  1  1  0 12 14]
 [ 3  0 61 11  0  0  0  0  0  2]
 [ 0  0  0 69  0  2  0  6  2  0]
 [ 5  0  0  0 70  0  3  4  1  0]
 [ 0  0  0  4  0 64  5  0  0  9]
 [ 0  1  0  0  0  0 79  0  0  0]
 [ 0  0  2  0  0  0  0 76  2  0]
 [ 0  1  3  4  0  5  0  3 54  6]
 [ 4  0  0 20  0  5  0  2  0 50]]
```

So as we expect, the accuracy is less (81%).

0.0.1 One of the important issue and the big one is neighbors = 200. So it is the main cause the accuracy is less. lets make it 3 as follows:

[15]:
```
knn_clf_new = KNeighborsClassifier(n_neighbors = 3, weights='distance')
knn_clf_new.fit(X[:1000], y[:1000])
```

[15]:
```
KNeighborsClassifier(n_neighbors=3, weights='distance')
```

[16]:
```
p = knn_clf_new.predict(X[1000:])
e = y[1000:]
print(metrics.classification_report(e,p))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        79
           1       0.97      0.96      0.97        80
           2       0.97      0.95      0.96        77
           3       0.91      0.91      0.91        79
           4       0.99      0.94      0.96        83
           5       0.95      0.99      0.97        82
           6       0.99      1.00      0.99        80
           7       0.98      1.00      0.99        80
           8       0.94      0.96      0.95        76
           9       0.93      0.91      0.92        81

    accuracy                           0.96       797
   macro avg       0.96      0.96      0.96       797
weighted avg       0.96      0.96      0.96       797
```

[18]:
```
print(metrics.confusion_matrix(e,p))
```

```
[[78  0  0  0  1  0  0  0  0  0]
 [ 0 77  1  0  0  0  0  0  2  0]
 [ 1  0 73  3  0  0  0  0  0  0]
 [ 0  0  1 72  0  1  0  2  2  1]
 [ 0  0  0  0 78  0  0  0  0  5]
 [ 0  0  0  0  0 81  1  0  0  0]
 [ 0  0  0  0  0  0 80  0  0  0]
 [ 0  0  0  0  0  0  0 80  0  0]
 [ 0  2  0  1  0  0  0  0 73  0]
 [ 0  0  0  3  0  3  0  0  1 74]]
```

So the accuracy is good now (96%).