Hand Written Digit Prediction - Classification Analysis

The 8x8 pixel pictures of numbers make up the digits dataset. Each image's 8x8 array of grayscale values is stored in the dataset's pictures attributes. These arrays will be used to visualise the first four pictures. Each image's assigned number is stored in the dataset's target characteristics.

Import Library

```
import pandas
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

Import Data

```
from sklearn.datasets import load_digits
```

```
df = load_digits()
```

```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, label in zip(axes, df.images, df.target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
    ax.set_title("Training: %i" % label)
```

Saved successfully!

Data Preprocessing

```
df.images.shape
```

```
(1797, 8, 8)
```

```
df.images[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.]])
```

```
df.images[0].shape
```

```
(8, 8)
```

```
len(df.images)
```

```
1797
```

```
n_samples = len(df.images)
data = df.images.reshape((n_samples, -1))
```

```
data[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.])
```



```
data.shape
```

```
(1797, 64)
```

Scaling Image Data

```
data.min()
```

```
0.0
```

```
data.max()
```

```
16.0
```

```
data = data/16
```

```
data.min()
```

```
0.0
```

```
data.max()
```

```
1.0
```

```
data[0]
```

```
array([0.    , 0.    , 0.3125, 0.8125, 0.5625, 0.0625, 0.    , 0.    ,
       0.    , 0.    , 0.8125, 0.9375, 0.625 , 0.9375, 0.3125, 0.    ,
       0.    , 0.1875, 0.9375, 0.125 , 0.    , 0.6875, 0.5   , 0.    ,
       0.    , 0.25  , 0.75  , 0.    , 0.    , 0.5   , 0.5   , 0.    ,
       0.    , 0.3125, 0.5   , 0.    , 0.    , 0.5625, 0.5   , 0.    ,
       0.    , 0.25  , 0.6875, 0.    , 0.0625, 0.75  , 0.4375, 0.    ,
       0.    , 0.125 , 0.875 , 0.3125, 0.625 , 0.75  , 0.    , 0.    ,
       0.    , 0.    , 0.375 , 0.8125, 0.625 , 0.    , 0.    , 0.    ])
```

Train Test Split Data

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(data, df.target, test_size=0.3)
```

Saved successfully!

```
.shape, y_test.shape
```

```
((1257, 64), (540, 64), (1257,), (540,))
```

Random Forest Model

```
from sklearn.ensemble import RandomForestClassifier
```

```
rf = RandomForestClassifier()
```

```
rf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

Predict Test Data

```
y_pred = rf.predict(X_test)
```

```
y_pred
```

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

Saved successfully!

```
from sklearn.metrics import confusion_matrix, classification_report
```

```
confusion_matrix(y_test, y_pred)
```

```
array([[48,  0,  0,  0,  1,  0,  0,  0,  0,  0],
       [ 0, 48,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 55,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 49,  0,  1,  0,  1,  0,  0],
       [ 0,  0,  0,  0, 49,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 52,  0,  0,  1,  1],
       [ 0,  0,  0,  0,  0,  0, 65,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 61,  0,  0],
```

```
[ 0,  2,  0,  0,  0,  0,  0,  1, 52,  0],
[ 0,  0,  0,  0,  0,  1,  0,  1,  0, 50]])
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        49
           1       0.96      1.00      0.98        48
           2       1.00      1.00      1.00        55
           3       1.00      0.96      0.98        51
           4       0.98      0.98      0.98        50
           5       0.96      0.96      0.96        54
           6       1.00      1.00      1.00        65
           7       0.94      1.00      0.97        61
           8       0.98      0.95      0.96        55
           9       0.98      0.96      0.97        52

    accuracy                           0.98       540
   macro avg       0.98      0.98      0.98       540
weighted avg       0.98      0.98      0.98       540
```

Saved successfully!

✓ 0s completed at 1:58 PM