Hand Written Digit Prediction - Classification Analysis

The digits dataset consists of 8x8 pixel images of digits. The images attribute of dataset stores 8x8 arrays of grayscale values for each image.We will use these arrays to visualize the first 4 images. The target attribute of the dataset stores the digit each image represents

Import Library

```
import pandas as pd
```

```
import numpy as np
```

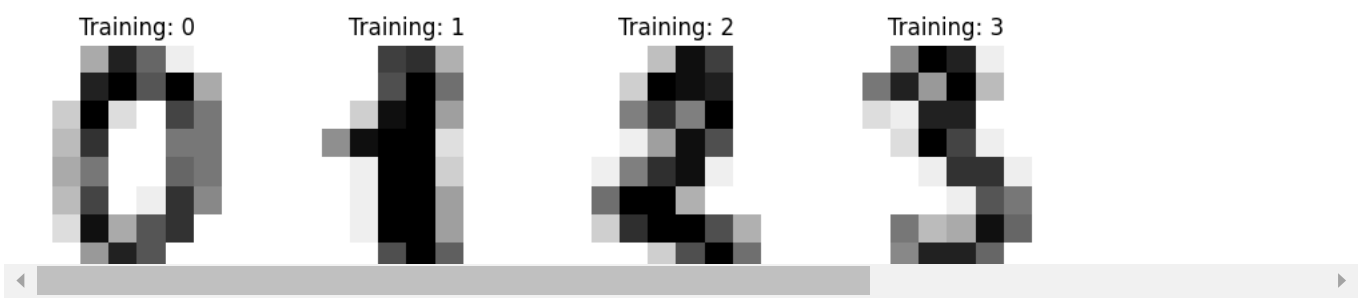
```
import matplotlib.pyplot as plt
```

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)
```



Data preprocessing

Flatten Image

```
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[0].shape
```

```
(64,)
```

```
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)
```

```
X_train.shape, X_test.shape,  y_train.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, 1, 6, 3, 3, 4, 4, 4, 6, 4, 3, 7, 7, 6, 8, 6, 9, 4, 9, 6, 8, 2,
       3, 7, 7, 1, 8, 5, 9, 2, 8, 2, 2, 7, 3, 3, 2, 1, 1, 4, 1, 7, 2, 3,
       4, 4, 2, 0, 2, 3, 4, 6, 7, 6, 1, 9, 5, 2, 0, 1, 9, 5, 8, 0, 4, 9,
       3, 9, 6, 3, 1, 9, 2, 6, 5, 1, 2, 7, 3, 2, 0, 2, 1, 7, 3, 1, 6, 5,
       8, 1, 8, 3, 3, 2, 9, 9, 6, 5, 9, 3, 8, 7, 1, 5, 4, 6, 2, 2, 4, 4,
       6, 7, 5, 7, 5, 8, 8, 0, 4, 3, 1, 8, 7, 5, 7, 4, 2, 9, 0, 5, 1, 9,
       9, 4, 8, 8, 1, 7, 9, 8, 0, 0, 2, 9, 0, 4, 1, 6, 9, 7, 7, 9, 1, 4,
       8, 0, 0, 2, 5, 0, 2, 1, 8, 4, 7, 5, 8, 5, 5, 4, 1, 8, 6, 3, 9, 5,
       5, 2, 0, 0, 0, 4, 5, 1, 0, 1, 2, 9, 1, 5, 4, 2, 5, 2, 9, 7, 7, 8,
       3, 8, 5, 3, 7, 5, 6, 8, 5, 9, 4, 5, 3, 4, 9, 4, 3, 1, 9, 5, 7, 3,
       9, 9, 9, 3, 5, 1, 1, 3, 3, 5, 9, 2, 6, 5, 6, 4, 4, 8, 1, 2, 9, 3,
       8, 5, 7, 3, 7, 6, 4, 5, 7, 7, 7, 9, 9, 7, 5, 7, 2, 3, 8, 0, 1, 7,
       6, 4, 9, 3, 1, 1, 6, 2, 3, 3, 2, 5, 5, 5, 3, 0, 3, 3, 4, 8, 2, 4,
       7, 4, 7, 1, 2, 2, 2, 9, 2, 2, 9, 6, 7, 1, 3, 9, 4, 8, 6, 9, 2, 6,
       4, 5, 3, 1, 2, 1, 7, 3, 8, 5, 2, 7, 5, 8, 1, 8, 1, 5, 0, 6, 7, 3,
       4, 1, 9, 1, 8, 8, 3, 2, 7, 2, 5, 0, 4, 1, 5, 7, 2, 8, 2, 6, 3, 6,
       2, 4, 2, 5, 8, 7, 0, 7, 1, 1, 0, 2, 9, 4, 9, 6, 6, 0, 9, 7, 3, 6,
       9, 0, 9, 8, 0, 2, 1, 1, 2, 7, 4, 9, 5, 4, 7, 8, 9, 3, 9, 2, 6, 6,
       8, 6, 5, 5, 7, 4, 0, 1, 6, 6, 0, 2, 8, 6, 5, 0, 9, 0, 0, 2, 7, 0,
       2, 9, 1, 4, 0, 1, 8, 4, 3, 0, 1, 5, 1, 0, 1, 1, 1, 6, 4, 4, 3, 2,
       9, 8, 4, 6, 5, 4, 4, 1, 1, 9, 1, 1, 1, 0, 7, 0, 7, 0, 0, 1, 0, 4,
       6, 1, 6, 4, 8, 6, 5, 7, 6, 8, 4, 2, 5, 7, 0, 6, 8, 8, 0, 9, 0, 5,
       7, 0, 5, 3, 0, 6, 2, 6, 2, 0, 1, 8, 9, 7, 6, 6, 9, 1, 9, 7, 6, 8,
       5, 5, 9, 0, 5, 6, 8, 5, 4, 1, 2, 5, 8, 9, 4, 7, 7, 0, 3, 8, 3, 2,
       5, 8, 7, 0, 3, 1, 7, 3, 7, 0, 4, 5])
```

Model Acccuracy

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

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

```
array([[47,  0,  0,  0,  1,  0,  0,  0,  0,  0],
       [ 0, 60,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 1,  0, 57,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  0, 48,  0,  1,  0,  0,  2,  0],
       [ 0,  0,  0,  0, 53,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 56,  0,  0,  0,  1],
       [ 1,  1,  0,  0,  0,  0, 49,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 56,  0,  2],
       [ 0,  0,  0,  0,  0,  0,  0,  0, 47,  0],
       [ 0,  0,  0,  2,  0,  1,  0,  0,  0, 52]])
```

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

```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97        48
           1       0.97      1.00      0.98        60
           2       1.00      0.98      0.99        58
           3       0.96      0.92      0.94        52
           4       0.98      0.98      0.98        54
           5       0.97      0.98      0.97        57
           6       1.00      0.96      0.98        51
           7       0.98      0.97      0.97        58
           8       0.96      1.00      0.98        47
           9       0.95      0.95      0.95        55

    accuracy                           0.97       540
   macro avg       0.97      0.97      0.97       540
weighted avg       0.97      0.97      0.97       540
```

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

	precision	recall	f1-score	support
0	0.96	0.98	0.97	48
1	0.97	1.00	0.98	60
2	1.00	0.98	0.99	58
3	0.96	0.92	0.94	52
4	0.98	0.98	0.98	54
5	0.97	0.98	0.97	57
6	1.00	0.96	0.98	51
7	0.98	0.97	0.97	58
8	0.96	1.00	0.98	47
9	0.95	0.95	0.95	55
accuracy			0.97	540
macro avg	0.97	0.97	0.97	540
weighted avg	0.97	0.97	0.97	540

Explanation: The primary objective of this project is to develop a machine learning model capable of accurately classifying handwritten digits (0-9) from image data. Using techniques such as data preprocessing, feature extraction, and model optimization.It involves the application of classification algorithms to assign the correct label (digit) to each image.

Start coding or generate with AI.