Hand Written Digit Prediction-Classification Analysis

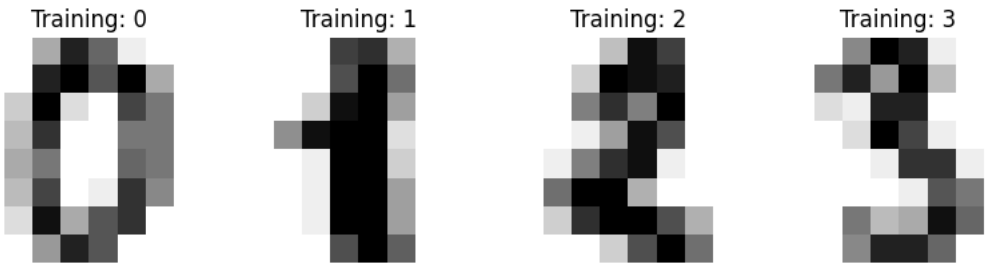
The digit dataset consists of 8x8 pixel images of digit. The images attribute of the dataset stores 8x8 array of grayscale values for each images. We will use these arrays to visusalize the first 4 images. The target attribute of the dataset store the digit each image Represent.

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

```
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_sample = len(df.images)
data = df.images.reshape((n_sample, -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)
```

Scalling Images 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 mode

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

Model Accuracy

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

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

```
array([[62,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 57,  0,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 69,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  0, 44,  0,  1,  0,  1,  0,  1],
       [ 0,  0,  0,  0, 46,  0,  0,  2,  0,  0],
       [ 0,  0,  0,  0,  0, 56,  0,  0,  0,  2],
       [ 0,  0,  0,  0,  1,  0, 46,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 56,  0,  0],
       [ 0,  2,  1,  0,  0,  0,  0,  0, 37,  1],
       [ 0,  0,  0,  0,  0,  0,  0,  1,  1, 51]])
```

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

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        62
           1       0.95      0.98      0.97        58
           2       0.99      1.00      0.99        69
           3       0.98      0.92      0.95        48
           4       0.98      0.96      0.97        48
           5       0.98      0.97      0.97        58
           6       1.00      0.98      0.99        47
           7       0.93      1.00      0.97        56
           8       0.97      0.90      0.94        41
           9       0.93      0.96      0.94        53

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