Hand Written Digit Prediction - Classification Analysis

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

Import Library

```
import pandas as pd
```

```
import numpy as np
```

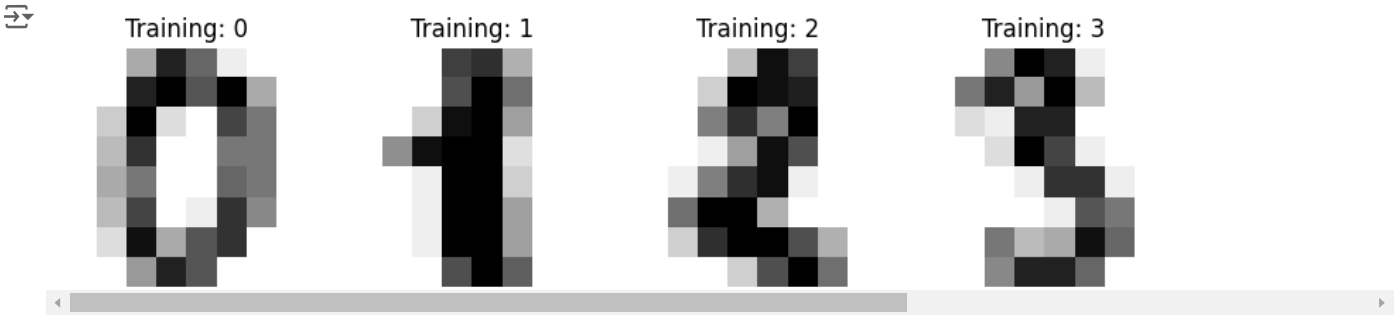
```
import matplotlib.pyplot as plt
```

Import Data

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

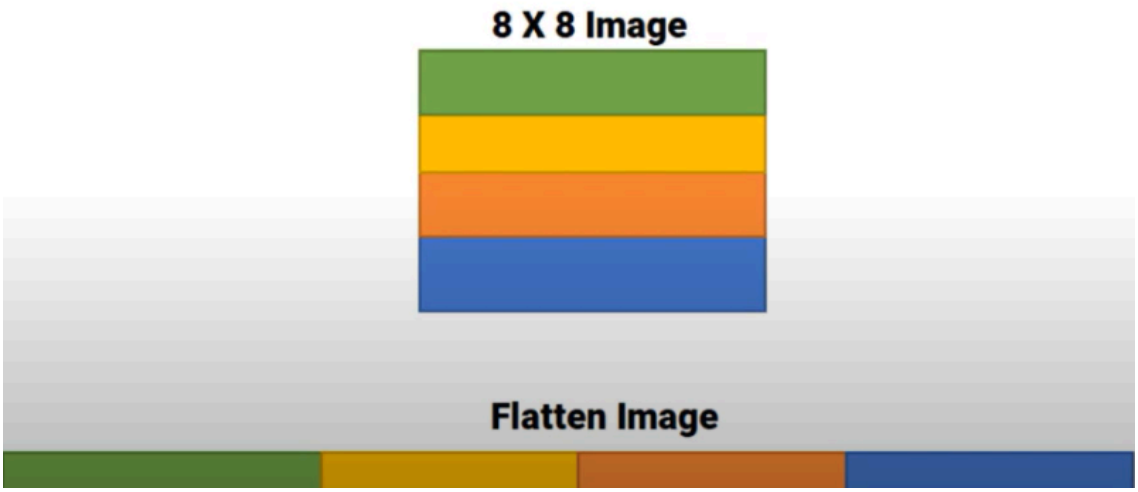
```
df = load_digits()
```

```
_, axes = plt.subplots(nrows=1,ncols=4,figsize=(10,3))
for ax, image, lable 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" % lable)
```



Data Processing

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

Model Accuracy

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

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

```
array([[49,  0,  0,  0,  1,  0,  0,  0,  0,  0],
       [ 0, 59,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 44,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 55,  0,  0,  0,  1,  1,  0],
       [ 0,  0,  0,  0, 56,  0,  0,  2,  0,  1],
       [ 0,  0,  0,  0,  0, 52,  0,  0,  0,  0],
       [ 1,  2,  0,  0,  1,  1, 60,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 53,  0,  0],
       [ 0,  3,  0,  0,  0,  1,  0,  2, 45,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  2, 48]])
```

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

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        50
           1       0.92      1.00      0.96        59
           2       1.00      1.00      1.00        44
           3       1.00      0.96      0.98        57
           4       0.97      0.95      0.96        59
           5       0.96      1.00      0.98        52
           6       1.00      0.92      0.96        65
           7       0.91      1.00      0.95        53
           8       0.94      0.88      0.91        51
           9       0.98      0.96      0.97        50

    accuracy                           0.96       540
   macro avg       0.97      0.97      0.97       540
weighted avg       0.97      0.96      0.96       540
```