OBJECTIVE: to develop a Handwritten digit prediction system that is used to recognize human handwritten digits.

DATA SOURCE:

Import Library

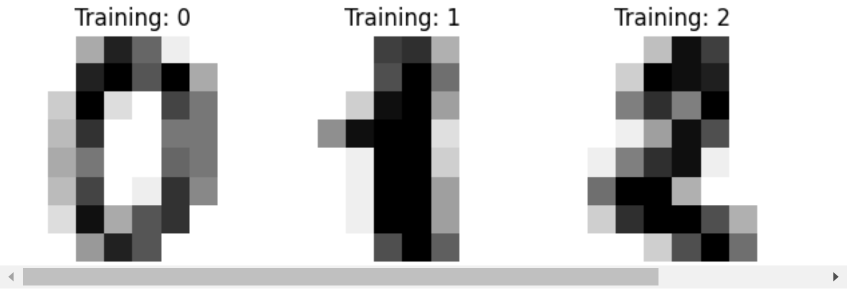
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Import Data

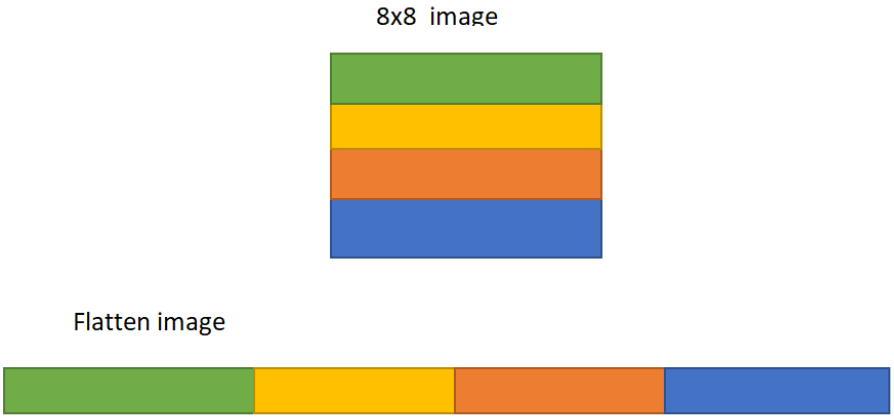
```
from sklearn.datasets import load_digits
```

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

Model Accuracy

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

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

```
array([[50,  0,  0,  0,  2,  0,  0,  0,  0,  0],
       [ 0, 52,  0,  0,  0,  1,  0,  0,  0,  0],
       [ 0,  1, 53,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 52,  0,  1,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 62,  0,  0,  2,  0,  0],
       [ 0,  0,  0,  0,  1, 45,  0,  0,  0,  0],
       [ 0,  1,  0,  0,  1,  0, 52,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 40,  0,  0],
       [ 0,  2,  1,  0,  0,  0,  0,  1, 54,  0],
       [ 0,  0,  0,  5,  0,  1,  0,  2,  0, 57]])
```

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

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        52
           1       0.93      0.98      0.95        53
           2       0.98      0.96      0.97        55
           3       0.90      0.98      0.94        53
           4       0.94      0.97      0.95        64
           5       0.94      0.98      0.96        46
           6       1.00      0.96      0.98        54
           7       0.89      1.00      0.94        40
           8       1.00      0.93      0.96        58
           9       1.00      0.88      0.93        65

    accuracy                           0.96       540
```

```
   macro avg       0.96      0.96      0.96       540
weighted avg       0.96      0.96      0.96       540
```

✓ 0s completed at 10:42 AM ● ×