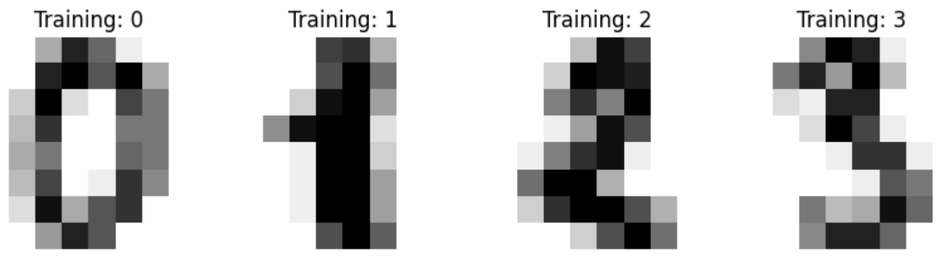
In [3]:
```
##HAND-WRITTEN DIGIT PREDICTION
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:
```
#import data
from sklearn.datasets import load_digits
df=load_digits()
```

In [6]:
```
_, 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)
```



In [8]:
```
#Data Preprocessing
df.images.shape
```

Out[8]:
```
(1797, 8, 8)
```

In [9]:
```
df.images[0]
```

Out[9]:
```
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.]])
```

In [10]:
```
df.images[0].shape
```

Out[10]:
```
(8, 8)
```

In [11]:
```
len(df.images)
```

Out[11]:
```
1797
```

In [12]:
```
n_samples=len(df.images)
data=df.images.reshape((n_samples,-1))
```

In [13]:
```
data[0]
```

Out[13]:
```
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.])
```

In [14]:
```
data[0].shape
```

Out[14]:
```
(64,)
```

In [15]:
```
data.shape
```

Out[15]:
```
(1797, 64)
```

In [16]:
```
#Scaling Image Data
data.min()
```

Out[16]:
```
0.0
```

In [17]:
```
data.max()
```

Out[17]:
```
16.0
```

In [18]:
```
data=data/16
```

In [19]:
```
data.min()
```

Out[19]:
```
0.0
```

In [20]:
```
data.max()
```

Out[20]:
```
1.0
```

In [21]:
```
data[0]
```

Out[21]:
```
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.    ])
```

In [22]:
```
#Train Test Split Data
from sklearn.model_selection import train_test_split
```

In [24]:
```
X_train,X_test,y_train,y_test=train_test_split(data,df.target,test_size=0.3)
```

In [25]:
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[25]:
```
((1257, 64), (540, 64), (1257,), (540,))
```

In [26]:
```
#Random Forest Model
from sklearn.ensemble import RandomForestClassifier
```

In [27]:
```
rf=RandomForestClassifier()
```

In [28]:
```
rf.fit(X_train,y_train)
```

Out[28]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [29]:
```
#Predict Test Data
y_pred=rf.predict(X_test)
```

In [30]:
```
y_pred
```

Out[30]:
```
array([8, 8, 3, 0, 5, 8, 9, 6, 9, 1, 5, 2, 5, 5, 3, 2, 5, 3, 8, 2, 0, 8,
       1, 3, 9, 6, 0, 1, 5, 6, 0, 3, 0, 7, 3, 9, 6, 1, 1, 3, 5, 7, 0, 0,
       6, 5, 3, 5, 6, 9, 3, 4, 6, 6, 2, 2, 1, 3, 2, 5, 3, 2, 0, 2, 6, 7,
       5, 8, 9, 7, 7, 4, 4, 2, 4, 1, 4, 8, 4, 3, 0, 9, 0, 7, 3, 0, 7, 8,
       2, 5, 5, 0, 9, 1, 5, 4, 6, 0, 8, 2, 8, 3, 3, 5, 3, 1, 5, 0, 8, 6,
       3, 6, 7, 5, 0, 4, 6, 9, 9, 7, 0, 0, 9, 3, 9, 9, 5, 3, 5, 9, 5, 7,
       7, 9, 1, 4, 0, 5, 6, 1, 1, 5, 7, 2, 5, 7, 3, 1, 6, 8, 7, 5, 1, 7,
       9, 1, 7, 2, 4, 0, 3, 5, 7, 1, 0, 9, 3, 8, 4, 7, 4, 6, 4, 3, 5, 8,
       6, 7, 5, 6, 6, 0, 7, 4, 3, 4, 4, 8, 5, 6, 1, 1, 3, 0, 6, 4, 0, 1,
       9, 2, 9, 5, 4, 9, 7, 2, 8, 9, 5, 7, 4, 9, 2, 0, 7, 8, 0, 2, 7, 7,
       3, 7, 8, 7, 2, 1, 0, 0, 9, 8, 1, 9, 9, 6, 0, 9, 8, 5, 2, 1, 2, 5,
       3, 1, 3, 7, 1, 4, 3, 9, 8, 3, 3, 0, 1, 7, 3, 9, 6, 1, 9, 6, 4, 3,
       9, 8, 0, 4, 6, 5, 4, 3, 6, 6, 8, 3, 6, 5, 3, 9, 0, 8, 1, 5, 6, 4,
       9, 1, 4, 1, 7, 4, 5, 5, 5, 9, 1, 7, 9, 0, 9, 2, 4, 8, 0, 0, 1, 7,
       8, 3, 1, 2, 3, 3, 9, 0, 3, 3, 3, 1, 7, 1, 9, 1, 7, 1, 1, 6, 9, 7,
       4, 5, 3, 7, 2, 8, 1, 8, 3, 1, 6, 0, 1, 6, 3, 4, 5, 5, 8, 4, 4, 3,
       7, 3, 0, 4, 9, 9, 8, 6, 2, 7, 7, 1, 7, 2, 2, 0, 7, 5, 2, 8, 2, 0,
       3, 7, 7, 9, 7, 6, 5, 6, 3, 4, 6, 9, 4, 8, 1, 7, 3, 0, 3, 9, 9, 0,
       7, 3, 6, 9, 7, 4, 5, 2, 1, 1, 0, 2, 6, 3, 8, 5, 0, 9, 3, 2, 8, 9,
       1, 4, 9, 0, 2, 9, 7, 4, 9, 8, 7, 7, 9, 2, 9, 6, 7, 8, 6, 9, 2, 3,
       5, 4, 1, 0, 7, 7, 0, 3, 3, 2, 8, 5, 8, 6, 4, 5, 3, 6, 6, 5, 7, 1,
       3, 1, 2, 2, 5, 3, 7, 9, 1, 7, 8, 9, 6, 2, 8, 3, 8, 0, 3, 7, 1, 9,
       5, 0, 2, 8, 9, 7, 6, 0, 0, 2, 6, 8, 5, 1, 8, 3, 9, 8, 1, 6, 1, 5,
       8, 5, 3, 3, 3, 0, 5, 2, 7, 1, 5, 8, 5, 9, 3, 2, 1, 9, 9, 9, 6, 2,
       3, 8, 3, 7, 8, 9, 0, 2, 7, 5, 7, 8])
```

In [31]:
```
#Model Accuracy
from sklearn.metrics import confusion_matrix,classification_report
```

In [32]:
```
confusion_matrix(y_test,y_pred)
```

Out[32]:
```
array([[52,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 54,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 43,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  1, 69,  0,  1,  0,  1,  1,  0],
       [ 0,  1,  0,  0, 39,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 56,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1, 48,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 58,  0,  1],
       [ 0,  0,  0,  0,  0,  0,  0,  1, 48,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  1,  1, 62]], dtype=int64)
```

In [33]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        52
           1       0.98      1.00      0.99        54
           2       0.98      1.00      0.99        43
           3       1.00      0.95      0.97        73
           4       1.00      0.95      0.97        41
           5       0.97      1.00      0.98        56
           6       1.00      0.98      0.99        49
           7       0.94      0.98      0.96        59
           8       0.96      0.98      0.97        49
           9       0.98      0.97      0.98        64

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

```
weighted avg       0.98      0.98      0.98       540
```

In [ ]: