In [1]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_digits
```

In [2]:
```
digits=load_digits()
```

In [3]:
```
digits
```

Out[3]:
```
{'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'frame': None,
 'feature_names': ['pixel_0_0',
  'pixel_0_1',
  'pixel_0_2',
  'pixel_0_3',
  'pixel_0_4',
  'pixel_0_5',
  'pixel_0_6',
  'pixel_0_7',
  'pixel_1_0',
  'pixel_1_1',
```

In [4]:
```
digits.target
```

Out[4]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

In [5]:
```
digits.target_names
```

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

In [6]:
```
from sklearn.datasets import load_digits
```

In [7]:
```
for image,label in zip(digits.data[0:5], digits.target[0:5]):
    print(image)
    print(label)
```

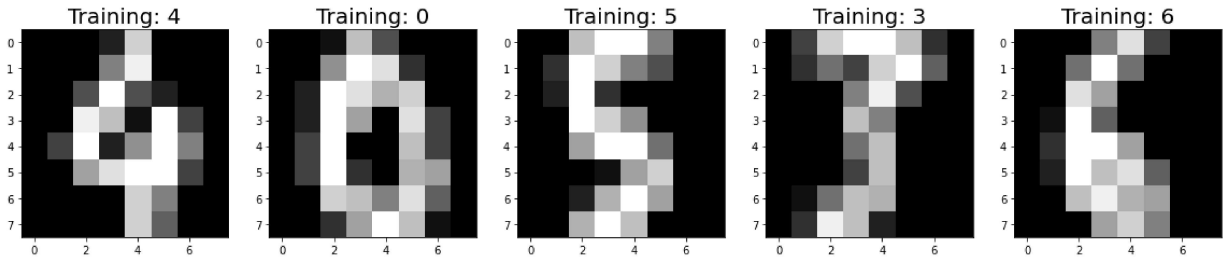
```
[ 0.  0.  5. 13.  9.  1.  0.  0.  0.  0. 13. 15. 10. 15.  5.  0.  0.  3.
 15.  2.  0. 11.  8.  0.  0.  4. 12.  0.  0.  8.  8.  0.  0.  5.  8.  0.
  0.  9.  8.  0.  0.  4. 11.  0.  1. 12.  7.  0.  0.  2. 14.  5. 10. 12.
  0.  0.  0.  0.  6. 13. 10.  0.  0.  0.]
0
[ 0.  0.  0. 12. 13.  5.  0.  0.  0.  0.  0. 11. 16.  9.  0.  0.  0.  0.
  3. 15. 16.  6.  0.  0.  0.  7. 15. 16. 16.  2.  0.  0.  0.  0.  1. 16.
 16.  3.  0.  0.  0.  0.  1. 16. 16.  6.  0.  0.  0.  0.  1. 16. 16.  6.
  0.  0.  0.  0.  0. 11. 16. 10.  0.  0.]
1
[ 0.  0.  0.  4. 15. 12.  0.  0.  0.  0.  3. 16. 15. 14.  0.  0.  0.  0.
  8. 13.  8. 16.  0.  0.  0.  0.  1.  6. 15. 11.  0.  0.  0.  1.  8. 13.
 15.  1.  0.  0.  0.  9. 16. 16.  5.  0.  0.  0.  0.  3. 13. 16. 16. 11.
  5.  0.  0.  0.  0.  3. 11. 16.  9.  0.]
2
[ 0.  0.  7. 15. 13.  1.  0.  0.  0.  8. 13.  6. 15.  4.  0.  0.  0.  2.
  1. 13. 13.  0.  0.  0.  0.  0.  2. 15. 11.  1.  0.  0.  0.  0.  0.  1.
 12. 12.  1.  0.  0.  0.  0.  0.  1. 10.  8.  0.  0.  0.  8.  4.  5. 14.
  9.  0.  0.  0.  7. 13. 13.  9.  0.  0.]
```

In [8]:
```
for i,j in enumerate(['ele1','ele2','ele3']):
    print(i)
    print(j)
```

```
0
ele1
1
ele2
2
ele3
```

In [9]:
```
#Display some of the images and those labels
import matplotlib.pyplot as plt
```

In [10]:
```
plt.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[100:105],digits.target[100:1
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title(f"Training: {label}",fontsize=20)
```



In [11]:
```
X=digits.data
y=digits.target
```

In [12]:
```
X
y
```

Out[12]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

In [13]:
```
X.shape
```

Out[13]:
```
(1797, 64)
```

In [14]:
```
y.shape
```

Out[14]:
```
(1797,)
```

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [17]:
```
X_train
```

Out[17]:
```
array([[ 0.,  0.,  3., ..., 13.,  4.,  0.],
       [ 0.,  0.,  9., ...,  3.,  0.,  0.],
       [ 0.,  0.,  0., ...,  6.,  0.,  0.],
       ...,
       [ 0.,  0.,  9., ..., 16.,  2.,  0.],
       [ 0.,  0.,  1., ...,  0.,  0.,  0.],
       [ 0.,  0.,  1., ...,  1.,  0.,  0.]])
```

In [18]:
```
y_train
```

Out[18]:
```
array([6, 0, 0, ..., 2, 7, 1])
```

In [19]:
```
from sklearn.preprocessing import StandardScaler
```

In [20]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [21]:
```
X_train.shape
```

Out[21]:
```
(1437, 64)
```

In [22]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

In [28]:
```
lda=LinearDiscriminantAnalysis(n_components=9)
X_train=lda.fit_transform(X_train,y_train)
X_test=lda.transform(X_test)
```

In [29]:
```
lda.explained_variance_ratio_
```

Out[29]:
```
array([0.27851663, 0.19023768, 0.17379022, 0.11077975, 0.08773235,
       0.0654018 , 0.04249174, 0.02968284, 0.02136698])
```

In [30]:
```
X_train.shape
```

Out[30]:
```
(1437, 9)
```

In [31]:
```
X_train[0]
```

Out[31]:
```
array([ 3.58237017, -1.82036205, -3.00354787,  2.13332268,  0.26151916,
       -1.5496613 , -0.19386593, -0.50007375,  0.40930703])
```

In [32]:
```
#model_building
```

In [33]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [35]:
```
rf1=RandomForestClassifier(n_estimators=100,random_state=42)
```

In [36]:
```
rf1.fit(X_train,y_train)
```

Out[36]:
```
RandomForestClassifier(random_state=42)
```

In [39]:
```
y_pred1=rf1.predict(X_test)
```

In [40]:
```
y_pred1
```

Out[40]:
```
array([6, 9, 3, 7, 2, 1, 5, 2, 5, 3, 1, 9, 4, 0, 4, 2, 3, 7, 8, 8, 4, 3,
       9, 7, 5, 6, 3, 5, 6, 3, 4, 9, 1, 4, 4, 6, 9, 4, 7, 6, 6, 9, 1, 3,
       6, 1, 3, 0, 6, 5, 5, 1, 9, 5, 6, 0, 9, 0, 0, 1, 0, 4, 5, 2, 4, 5,
       7, 0, 7, 5, 9, 5, 5, 4, 7, 0, 7, 5, 5, 9, 9, 0, 2, 3, 8, 0, 6, 4,
       4, 9, 1, 2, 8, 3, 5, 2, 9, 4, 4, 4, 4, 3, 5, 3, 1, 3, 5, 9, 4, 2,
       7, 7, 4, 4, 1, 9, 2, 7, 8, 7, 2, 6, 9, 4, 0, 7, 2, 7, 5, 8, 7, 5,
       7, 9, 0, 6, 6, 4, 2, 8, 0, 9, 4, 6, 9, 9, 6, 9, 0, 5, 5, 6, 6, 0,
       6, 4, 3, 9, 3, 7, 7, 2, 9, 0, 4, 5, 3, 6, 5, 9, 9, 8, 4, 2, 1, 3,
       7, 7, 2, 2, 3, 9, 8, 0, 3, 2, 3, 5, 6, 9, 9, 4, 1, 5, 4, 2, 3, 6,
       4, 8, 5, 9, 5, 7, 8, 9, 4, 8, 1, 5, 4, 4, 9, 6, 1, 8, 6, 0, 4, 5,
       2, 7, 4, 6, 4, 5, 6, 0, 3, 2, 3, 6, 7, 1, 5, 1, 4, 7, 6, 9, 1, 5,
       5, 1, 4, 2, 8, 8, 9, 8, 7, 6, 2, 2, 2, 3, 4, 8, 8, 3, 6, 0, 3, 7,
       7, 0, 1, 0, 4, 5, 1, 5, 3, 6, 0, 4, 1, 0, 0, 3, 6, 5, 9, 7, 3, 5,
       5, 9, 9, 8, 5, 3, 3, 2, 0, 5, 8, 3, 4, 0, 2, 4, 6, 4, 3, 4, 5, 0,
       5, 2, 1, 3, 1, 4, 1, 1, 7, 0, 1, 5, 2, 1, 2, 8, 7, 0, 6, 4, 8, 8,
       5, 1, 8, 4, 5, 8, 7, 9, 8, 6, 0, 6, 2, 0, 7, 9, 8, 9, 5, 2, 7, 7,
       1, 8, 7, 4, 3, 8, 3, 5])
```

In [41]:
```
y_test
```

Out[41]:
```
array([6, 9, 3, 7, 2, 1, 5, 2, 5, 2, 1, 9, 4, 0, 4, 2, 3, 7, 8, 8, 4, 3,
       9, 7, 5, 6, 3, 5, 6, 3, 4, 9, 1, 4, 4, 6, 9, 4, 7, 6, 6, 9, 1, 3,
       6, 1, 3, 0, 6, 5, 5, 1, 9, 5, 6, 0, 9, 0, 0, 1, 0, 4, 5, 2, 4, 5,
       7, 0, 7, 5, 9, 5, 5, 4, 7, 0, 4, 5, 5, 9, 9, 0, 2, 3, 8, 0, 6, 4,
       4, 9, 1, 2, 8, 3, 5, 2, 9, 0, 4, 4, 4, 3, 5, 3, 1, 3, 5, 9, 4, 2,
       7, 7, 4, 4, 1, 9, 2, 7, 8, 7, 2, 6, 9, 4, 0, 7, 2, 7, 5, 8, 7, 5,
       7, 7, 0, 6, 6, 4, 2, 8, 0, 9, 4, 6, 9, 9, 6, 9, 0, 3, 5, 6, 6, 0,
       6, 4, 3, 9, 3, 9, 7, 2, 9, 0, 4, 5, 3, 6, 5, 9, 9, 8, 4, 2, 1, 3,
       7, 7, 2, 2, 3, 9, 8, 0, 3, 2, 2, 5, 6, 9, 9, 4, 1, 5, 4, 2, 3, 6,
       4, 8, 5, 9, 5, 7, 8, 9, 4, 8, 1, 5, 4, 4, 9, 6, 1, 8, 6, 0, 4, 5,
       2, 7, 4, 6, 4, 5, 6, 0, 3, 2, 3, 6, 7, 1, 5, 1, 4, 7, 6, 8, 8, 5,
       5, 1, 6, 2, 8, 8, 9, 9, 7, 6, 2, 2, 2, 3, 4, 8, 8, 3, 6, 0, 9, 7,
       7, 0, 1, 0, 4, 5, 1, 5, 3, 6, 0, 4, 1, 0, 0, 3, 6, 5, 9, 7, 3, 5,
       5, 9, 9, 8, 5, 3, 3, 2, 0, 5, 8, 3, 4, 0, 2, 4, 6, 4, 3, 4, 5, 0,
       5, 2, 1, 3, 1, 4, 1, 1, 7, 0, 1, 5, 2, 1, 2, 8, 7, 0, 6, 4, 8, 8,
       5, 1, 8, 4, 5, 8, 7, 9, 8, 5, 0, 6, 2, 0, 7, 9, 8, 9, 5, 2, 7, 7,
       1, 8, 7, 4, 3, 8, 3, 5])
```

In [42]:
```
#accuracy_score
```

In [48]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [46]:
```
accuracy_score(y_test,y_pred1)
```

Out[46]:
```
0.9638888888888889
```

In [52]:
```
y=confusion_matrix(y_test,y_pred1)
```

In [50]:
```
c=classification_report(y_test,y_pred1)
```

In [51]:
```
print(c)
```

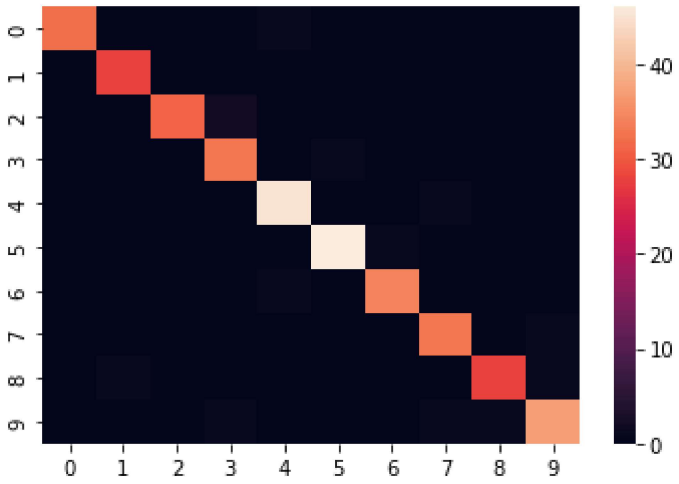
```
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        33
           1       0.97      1.00      0.98        28
           2       1.00      0.94      0.97        33
           3       0.92      0.97      0.94        34
           4       0.96      0.98      0.97        46
           5       0.98      0.98      0.98        47
           6       0.97      0.97      0.97        35
           7       0.94      0.97      0.96        34
           8       0.97      0.93      0.95        30
           9       0.95      0.93      0.94        40

    accuracy                           0.96       360
   macro avg       0.96      0.96      0.96       360
weighted avg       0.96      0.96      0.96       360
```

In [53]:
```
import seaborn as sns
sns.heatmap(y)
```

Out[53]: <AxesSubplot:>



In [54]:
```
def get_misclassified_index(y_pred,y_test):
    misclassification=[]#help us out to get the misclassified index value
    for index,(predicted,actual) in enumerate(zip(y_pred,y_test)):
        if predicted!=actual:
            misclassification.append(index)

    return misclassification
```

In [57]:
```
misclassification = get_misclassified_index(y_pred1,y_test)
```

In [58]:
```
misclassification
```

Out[58]: [9, 76, 97, 133, 149, 159, 186, 239, 240, 244, 249, 262, 339]

In [ ]: