In [4]:
```
import numpy  as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
```

In [7]:
```
digits = load_digits()
print("Shape of data images",digits.data.shape)
print("Label data shape",digits.target.shape)
```

```
Shape of data images (1797, 64)
Label data shape (1797,)
```

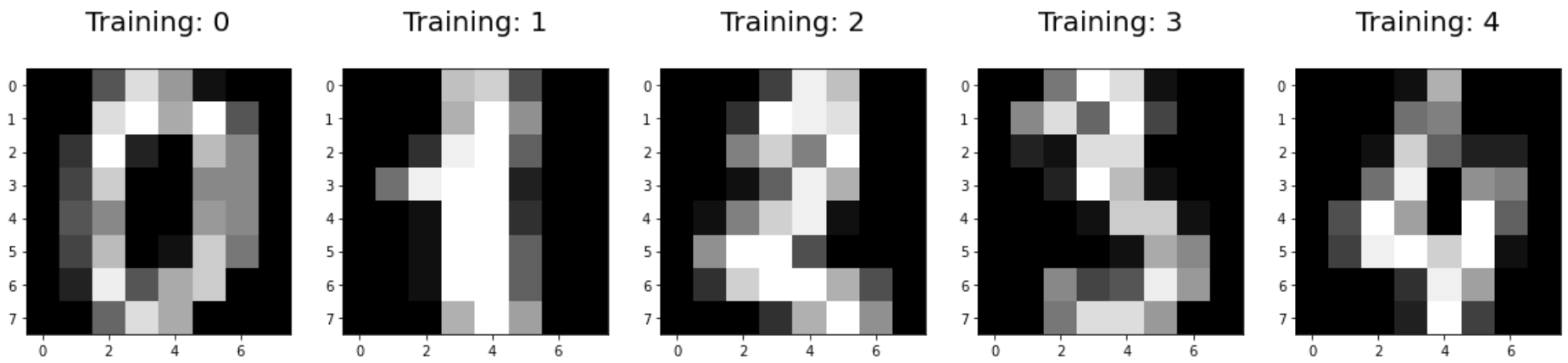
In [8]:
```
digits.data
```

Out[8]:
```
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.]])
```

In [10]:
```
digits.target
```

Out[10]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

In [11]:
```
plt.figure(figsize=(20,5))
for index, (image,label) in enumerate(zip(digits.data[0:5],digits.target[0:5])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title("Training: %i\n"%label,fontsize=20)
```



In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(digits.data,digits.target,test_size=0.3,random_state=3)
x_train.shape,x_test.shape
```

Out[12]:
```
((1257, 64), (540, 64))
```

In [16]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\Users\U.R Computer\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[16]:
```
LogisticRegression()
```

In [27]:
```
scr = lr.score(x_test,y_test)*100
scr
```

Out[27]:
```
95.37037037037037
```

In [ ]:
```

```

In [19]:
```
y_pred = lr.predict(x_test)
y_pred[0:10]
```

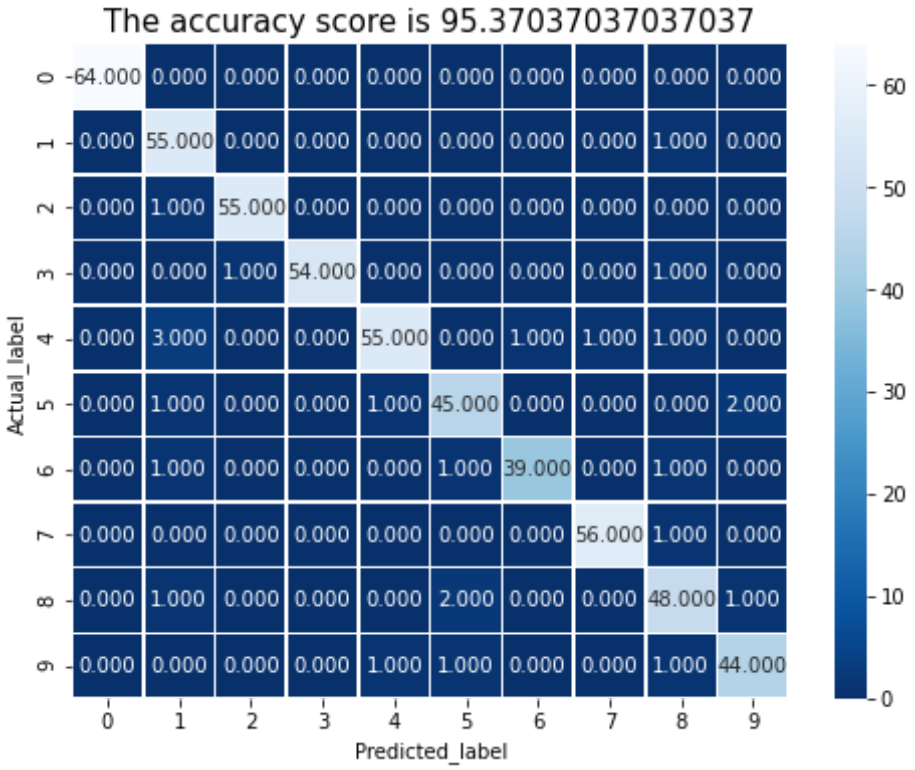
Out[19]:
```
array([0, 4, 1, 2, 0, 0, 8, 7, 6, 6])
```

In [20]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[20]:
```
array([[64,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 55,  0,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  1, 55,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  1, 54,  0,  0,  0,  0,  1,  0],
       [ 0,  3,  0,  0, 55,  0,  1,  1,  1,  0],
       [ 0,  1,  0,  0,  1, 45,  0,  0,  0,  2],
       [ 0,  1,  0,  0,  0,  1, 39,  0,  1,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 56,  1,  0],
       [ 0,  1,  0,  0,  0,  2,  0,  0, 48,  1],
       [ 0,  0,  0,  0,  1,  1,  0,  0,  1, 44]], dtype=int64)
```

In [35]:
```
plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,linewidths=0.5,fmt=".3f",cmap="Blues_r")
plt.ylabel("Actual_label")
plt.xlabel("Predicted_label")
title = f"The accuracy score is {scr}"
plt.title(title,size=15)
```

Out[35]:
```
Text(0.5, 1.0, 'The accuracy score is 95.37037037037037')
```

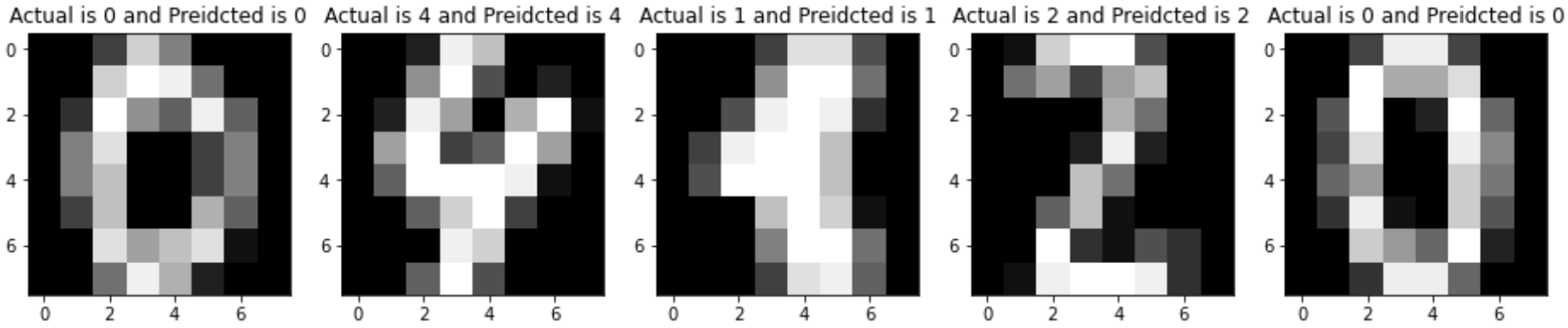


In [36]:
```
index = 0
classifiedIndex = []
for actual, predicted in zip(y_test,y_pred):
    if actual==predicted:
        classifiedIndex.append(index)
    index += 1
```

In [47]:
```
print(len(classifiedIndex),len(y_test))
```

```
515 540
```

In [59]:
```
plt.figure(figsize=(16,12))
for plotindex, wrong in enumerate(classifiedIndex[0:5]):
    plt.subplot(1,5,plotindex + 1)
    plt.imshow(np.reshape(x_test[wrong],(8,8)),cmap=plt.cm.gray)
    plt.title(f"Actual is {y_test[wrong]} and Preidcted is {y_pred[wrong]}")
```



In [ ]: