In [1]:
```
import os
```

In [2]:
```
import numpy as np
import seaborn as sns
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
```

In [4]:
```
from sklearn.datasets import fetch_openml
X,y=fetch_openml ('mnist_784', version=1,return_X_y=True)
```

```
C:\Users\User\anaconda3\Lib\site-packages\sklearn\datasets\_openml.py:1002: F
utureWarning: The default value of `parser` will change from `'liac-arff'` to
`'auto'` in 1.4. You can set `parser='auto'` to silence this warning. Therefo
re, an `ImportError` will be raised from 1.4 if the dataset is dense and pand
as is not installed. Note that the pandas parser may return different data ty
pes. See the Notes Section in fetch_openml's API doc for details.
  warn(
```

In [5]:
```
X.shape
```

Out[5]: (70000, 784)

In [18]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0
```

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

Out[21]: (56000, 784)

In [22]:
```
X_test.shape
```

Out[22]: (14000, 784)

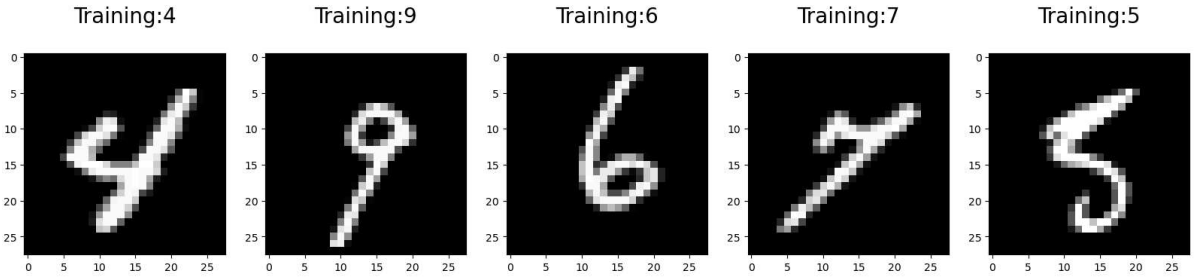
In [23]:
```
y_train.shape
```

Out[23]: (56000,)

In [24]:
```
y_test.shape
```

Out[24]: (14000,)

In [43]:
```
plt.figure(figsize=(20,4))
for index in range(5):
    plt.subplot (1,5,index+1)
    plt.imshow(X_train.to_numpy()[index].reshape((28,28)),cmap=plt.cm.gray)
    plt.title('Training:%i\n' %int(y_train.to_numpy()[index]),fontsize=20)
```



In [45]:
```
from sklearn.linear_model import LogisticRegression
logmodel=LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
C:\Users\User\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:4
60: ConvergenceWarning: lbfgs failed 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 (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 (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[45]:
```
LogisticRegression
LogisticRegression()
```

In [46]:
```
prediction=logmodel.predict(X_test)
from sklearn.metrics import accuracy_score
score=accuracy_score(y_test,prediction)
print(score)
```

```
0.9172142857142858
```

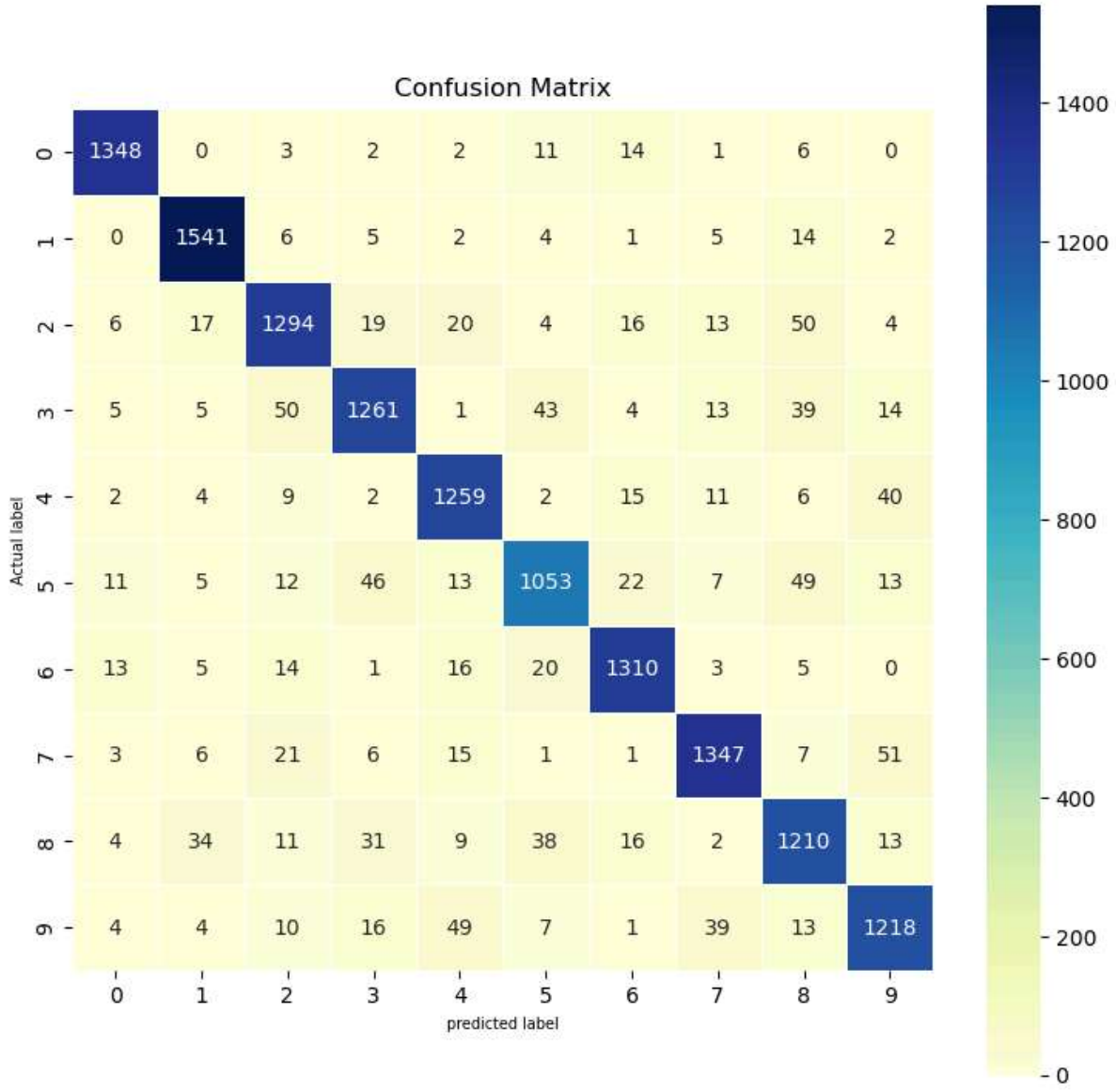
In [48]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,prediction))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      1387
           1       0.95      0.98      0.96      1580
           2       0.90      0.90      0.90      1443
           3       0.91      0.88      0.89      1435
           4       0.91      0.93      0.92      1350
           5       0.89      0.86      0.87      1231
           6       0.94      0.94      0.94      1387
           7       0.93      0.92      0.93      1458
           8       0.86      0.88      0.87      1368
           9       0.90      0.89      0.90      1361

    accuracy                           0.92     14000
   macro avg       0.92      0.92      0.92     14000
weighted avg       0.92      0.92      0.92     14000
```

In [49]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,prediction)
print(cm)
```

```
[[1348    0    3    2    2   11   14    1    6    0]
 [   0 1541    6    5    2    4    1    5   14    2]
 [   6   17 1294   19   20    4   16   13   50    4]
 [   5    5   50 1261    1   43    4   13   39   14]
 [   2    4    9    2 1259    2   15   11    6   40]
 [  11    5   12   46   13 1053   22    7   49   13]
 [  13    5   14    1   16   20 1310    3    5    0]
 [   3    6   21    6   15    1    1 1347    7   51]
 [   4   34   11   31    9   38   16    2 1210   13]
 [   4    4   10   16   49    7    1   39   13 1218]]
```

In [51]:
```
plt.figure(figsize=(9,9))
plt.title('Confusion Matrix')
sns.heatmap(cm,annot=True,fmt='g',linewidth=0.5,cmap='YlGnBu',square=True)
plt.xlabel('predicted label',fontsize=7)
plt.ylabel('Actual label',fontsize=7)
plt.show()
```



In [ ]:
```
```