Lab 07: Handwritten Digit Recognition Using MLPClassifier

1. Import Necessary imports

In [1]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.io import loadmat

from sklearn.neural_network import MLPClassifier
from sklearn import metrics
```

2. Upload Dataset

In [4]:
```
mnist_raw = loadmat("mnist-original.mat")
```

MNIST is a database. The acronym stands for “Modified National Institute of Standards and Technology.” The MNIST database contains handwritten digits (0 through 9), and can provide a baseline for testing image processing systems.

mnist_raw file is a matlab file . It is a dictionary containing some key value pairs . Lets have a look

In [5]:
```
mnist_raw
```

Out[5]:
```
{'__header__': b'MATLAB 5.0 MAT-file Platform: posix, Created on: Tue Jan 19 16:10:39 2016',
 '__version__': '1.0',
 '__globals__': [],
 'mldata_descr_ordering': array([[array(['label'], dtype='<U5'), array(['data'], dtype='<U4')]],
       dtype=object),
 'data': array([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=uint8),
 'label': array([[0., 0., 0., ..., 9., 9., 9.]])}
```

We are interested in data and labels

In [6]:
```
mnist_raw["data"]
```

Out[6]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

In [7]:
```
mnist_raw["data"].shape
```

Out[7]:
```
(784, 70000)
```

Each image is stored in a column. Each row represnts each pixcel value . Each image is 28 X 28. so there are total 784 pixel values in an image.This format is not consistent.

In [8]:
```
mnist_raw["data"].T.shape
```

Out[8]:
```
(70000, 784)
```

In [9]:
```
mnist_raw["label"]
```

Out[9]:
```
array([[0., 0., 0., ..., 9., 9., 9.]])
```

In [10]:
```
mnist_raw["label"].T.shape
```

Out[10]:
```
(70000, 1)
```

In [11]:
```
X,y = mnist_raw['data'].T, mnist_raw["label"].T
X.shape, y.shape
```

Out[11]:
```
((70000, 784), (70000, 1))
```

3. Train Test Split

In [12]:
```
shuffle_index = np.random.permutation(70000)
X, y = X[shuffle_index], y[shuffle_index]
```

In [13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=101)
```

In [14]:
```
y
```

Out[14]:
```
array([[1.],
       [4.],
       [7.],
       ...,
       [2.],
       [1.],
       [1.]])
```

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

Out[15]:
```
(70000, 784)
```

In [16]:
```
y.shape
```

Out[16]:
```
(70000, 1)
```

4. Dataset Visualization

In [17]:
```
some_digit = X[36000]
```

In [18]:
```
some_digit.shape
```

Out[18]:
```
(784,)
```

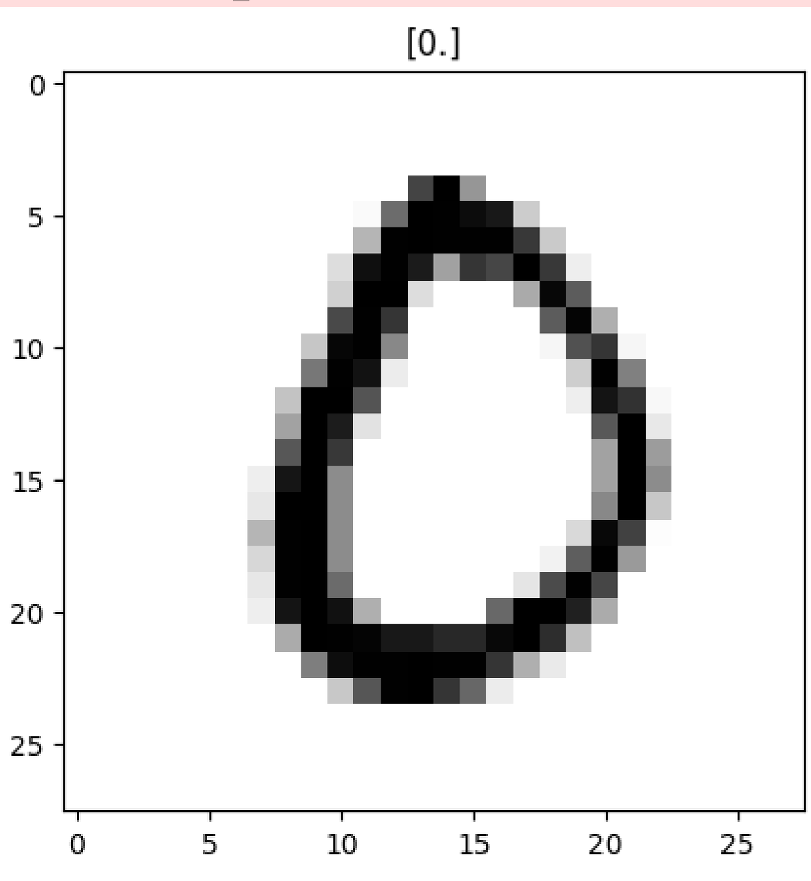
In [19]:
```
some_digit;
```

In [20]:
```
some_digit_image = some_digit.reshape(28, 28)
```

In [21]:
```
some_digit_image;
```

In [22]:
```
plt.imshow(some_digit_image, cmap = plt.cm.binary)
plt.title(y[36000])
plt.show()
```

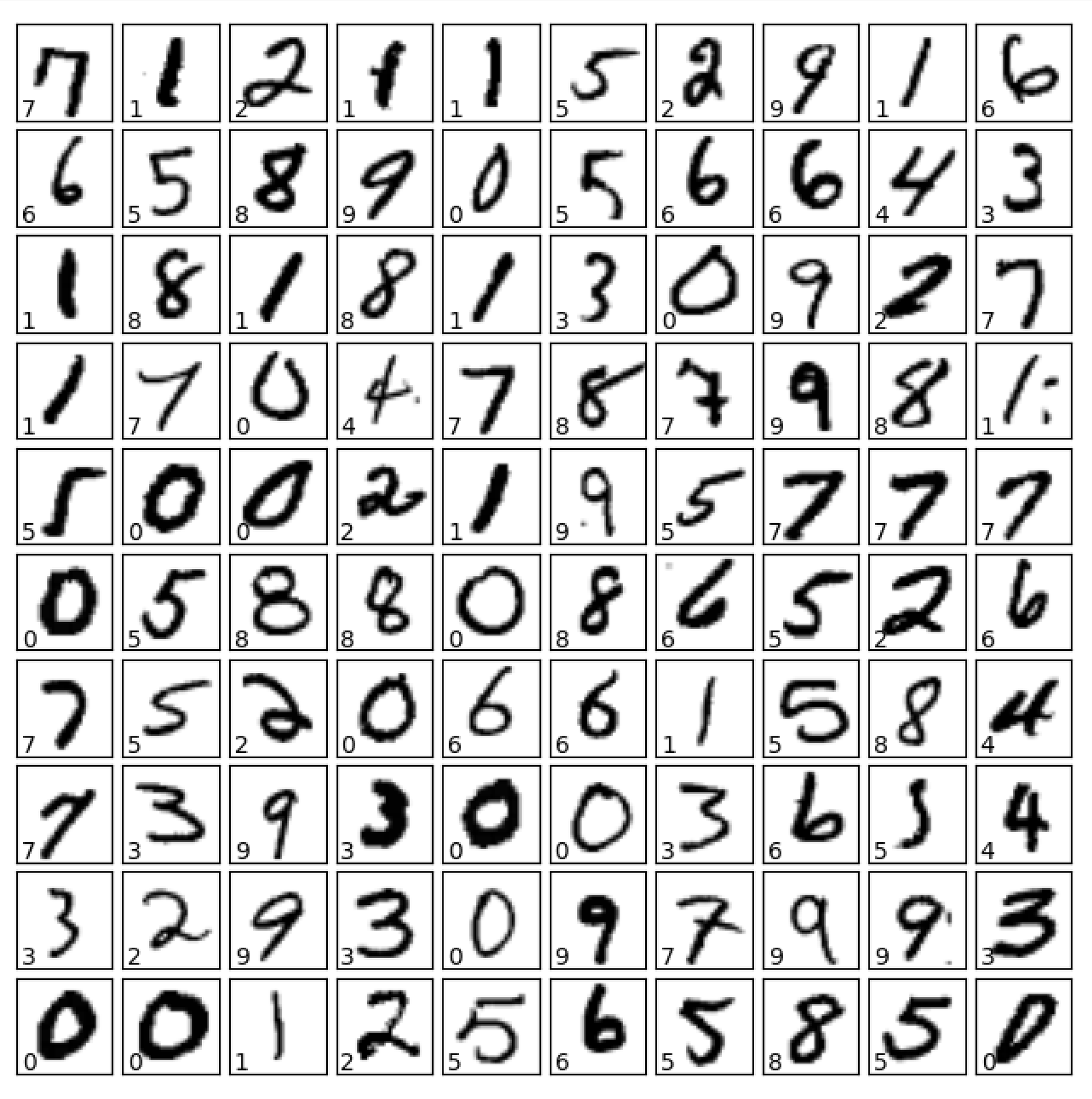
```
C:\Users\Azlan\anaconda3\envs\tensorflow_env\lib\site-packages\matplotlib\text.py:1242: FutureWarning: elementwise comparison failed; returning scalar instea
d, but in the future will perform elementwise comparison
  if s != self._text:
```



View First 100 Training Data Points

In [23]:
```
fig, axes = plt.subplots(10, 10, figsize=(8, 8),
                        subplot_kw={'xticks':[], 'yticks':[]},
                        gridspec_kw=dict(hspace=0.1, wspace=0.1))
for i, ax in enumerate(axes.flat):

    ax.imshow(X_train[i].reshape((28,28)), cmap='binary')
    ax.text(0.05, 0.05, str(int(y_train[i])),
           transform=ax.transAxes, color='black')
plt.show()
```



5. Training MLPClassifier

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

Out[24]:
```
(63000, 784)
```

In [25]:
```
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=10, alpha=1e-4,
                    solver='sgd', verbose=True, tol=1e-5, random_state=1,
                    learning_rate_init=0.001)

mlp.fit(X_train, y_train)
```

```
C:\Users\Azlan\anaconda3\envs\tensorflow_env\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:1118: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
Iteration 1, loss = 1.97830828
Iteration 2, loss = 0.95796046
Iteration 3, loss = 0.70889967
Iteration 4, loss = 0.56051496
Iteration 5, loss = 0.41836474
Iteration 6, loss = 0.35549838
Iteration 7, loss = 0.32619268
Iteration 8, loss = 0.30650824
Iteration 9, loss = 0.29068110
Iteration 10, loss = 0.28206831
C:\Users\Azlan\anaconda3\envs\tensorflow_env\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:702: ConvergenceWarning: Stochastic Optimizer:
Maximum iterations (10) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[25]:
```
MLPClassifier
MLPClassifier(max_iter=10, random_state=1, solver='sgd', tol=1e-05,
              verbose=True)
```

In [26]:
```
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

```
Training set score: 0.923921
Test set score: 0.915143
```

6. Prediction on Test Data

In [27]:
```
yfit = mlp.predict(X_test)
```

In [28]:
```
y_test.shape
```

Out[28]:
```
(7000, 1)
```

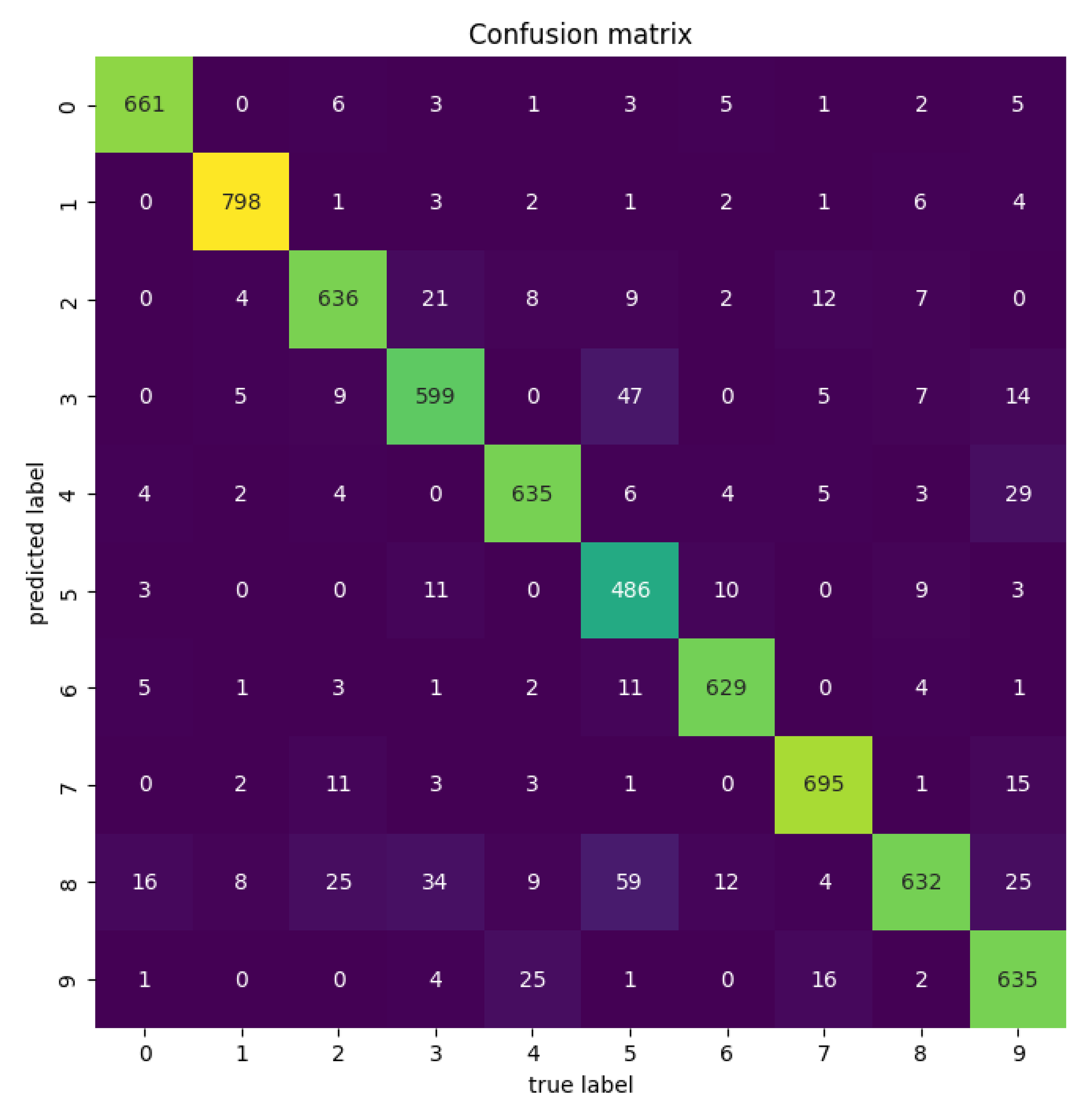
In [29]:
```
tn = ['0','1','2','3','4','5','6','7','8','9']
print(metrics.classification_report(y_test, yfit,
                            target_names=tn))
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       690
           1       0.98      0.97      0.97       820
           2       0.91      0.92      0.91       695
           3       0.87      0.88      0.88       679
           4       0.92      0.93      0.92       685
           5       0.93      0.78      0.85       624
           6       0.96      0.95      0.95       664
           7       0.95      0.94      0.95       739
           8       0.77      0.94      0.84       673
           9       0.93      0.87      0.90       731

    accuracy                           0.92      7000
   macro avg       0.92      0.91      0.91      7000
weighted avg       0.92      0.92      0.92      7000
```

In [30]:
```
mat = metrics.confusion_matrix(y_test, yfit)
fig, ax = plt.subplots(figsize=(8,8))
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False, cmap='viridis',
            xticklabels=tn,
            yticklabels=tn
            )
plt.title('Confusion matrix')
plt.xlabel('true label')
plt.ylabel('predicted label');
```



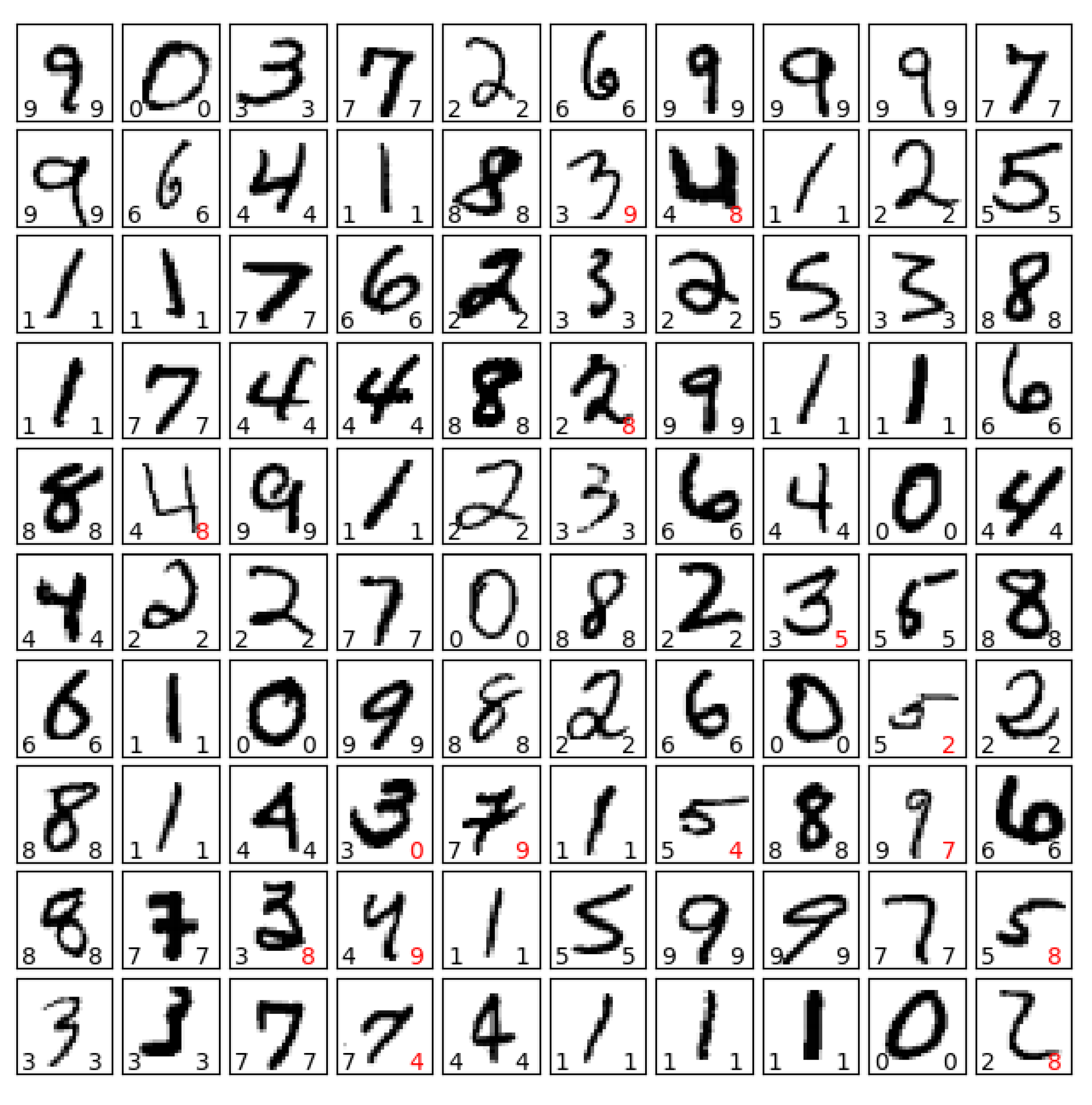
In [31]:
```
from sklearn.metrics import accuracy_score
print("Accuracy", metrics.accuracy_score(y_test, yfit)*100)
```

```
Accuracy 91.51428571428572
```

In [32]:
```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(10, 10, figsize=(8, 8),
                        subplot_kw={'xticks':[], 'yticks':[]},
                        gridspec_kw=dict(hspace=0.1, wspace=0.1))

for i, ax in enumerate(axes.flat):
    ax.imshow(X_test[i].reshape((28,28)), cmap='binary', interpolation='nearest')
    # actual class
    ax.text(0.05, 0.05, str(int(y_test[i])),
            transform=ax.transAxes,
            color='black')
    # predict class
    ax.text(0.75, 0.05, str(int(yfit[i])),
            transform=ax.transAxes,
            color='black' if yfit[i] == y_test[i] else 'red')
plt.show()
```



In [ ]: