# a01_3

October 3, 2025

[1]:
```
# ---
# jupyter:
#   jupytext:
#     text_representation:
#       extension: .py
#       format_name: percent
#       format_version: '1.3'
#       jupytext_version: 1.16.7
# ---

# %% [markdown]
# # 3 Experiments on MNIST Digits Data

# %%
import sklearn
import sklearn.metrics
# %load_ext autoreload
# %autoreload 2

from a01_helper import *
from a01_functions import nb_train, nb_predict
```

[2]:
```
# %%
# Let's train the model on the digits data and predict
model_nb2 = nb_train(X, y, alpha=2)
pred_nb2 = nb_predict(model_nb2, Xtest)
yhat = pred_nb2["yhat"]
logprob = pred_nb2["logprob"]
```

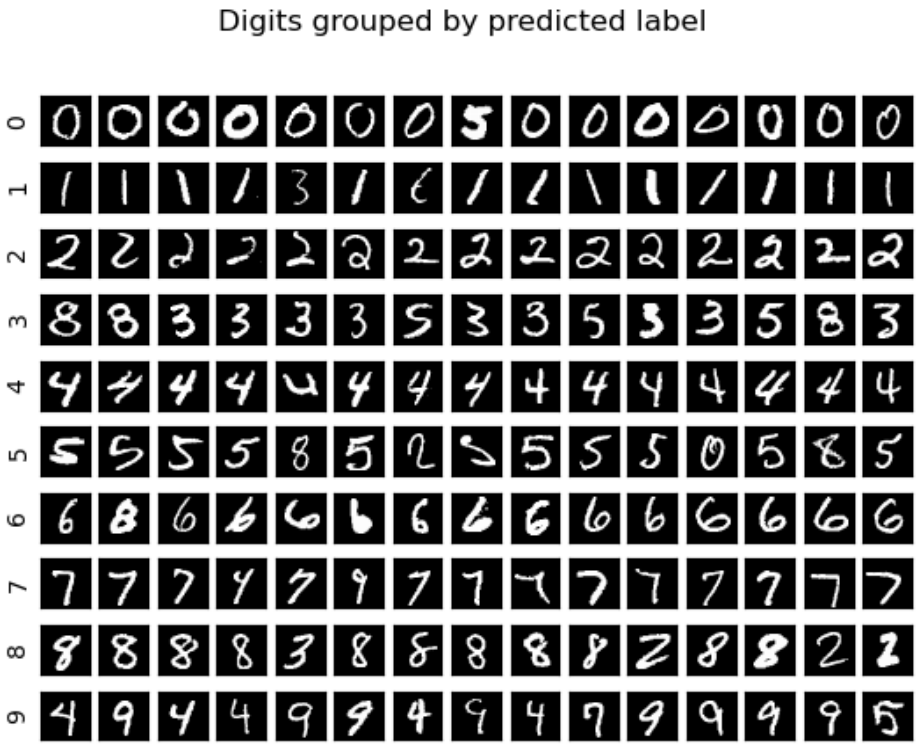
[3]:
```
# %%
# Accuracy
sklearn.metrics.accuracy_score(ytest, yhat)
```

[3]:
```
0.8363
```

[4]:
```
# %%
# show some digits grouped by prediction; can you spot errors?
nextplot()
```

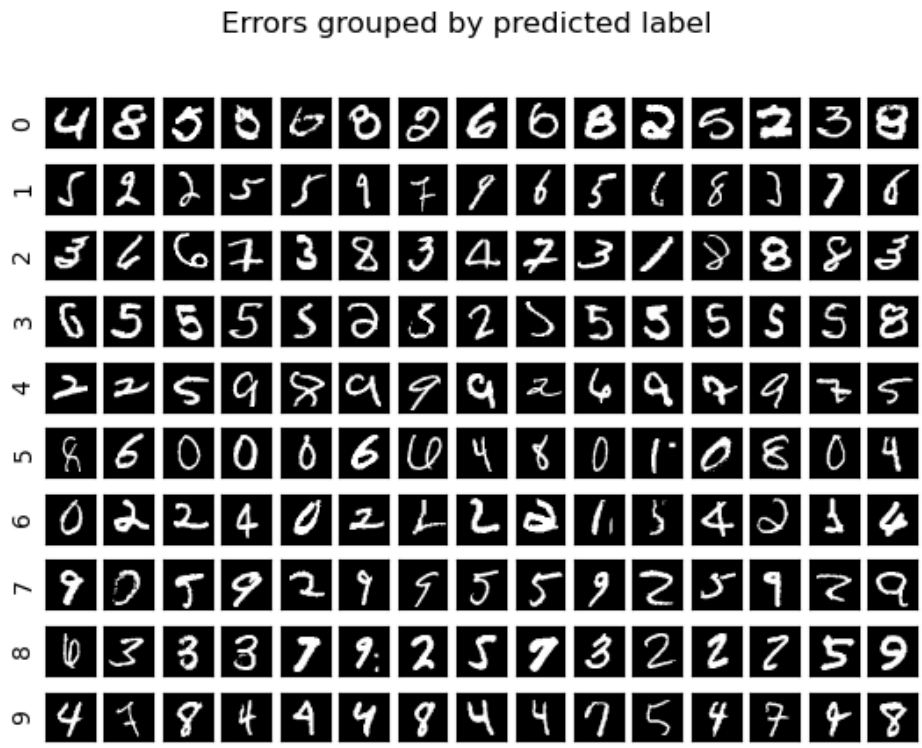
```
showdigits(Xtest, yhat)
plt.suptitle("Digits grouped by predicted label")
```

[4]: Text(0.5, 0.98, 'Digits grouped by predicted label')



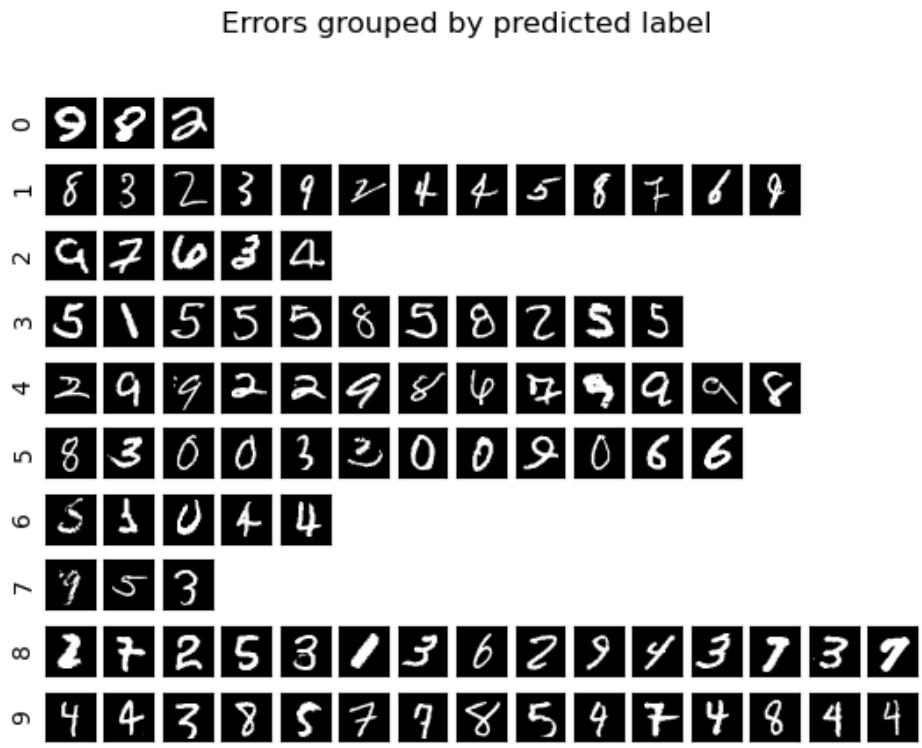

[5]:
```
# %%
# do the same, but this time show wrong predicitions only
perror = ytest != yhat
nextplot()
showdigits(Xtest[perror, :], yhat[perror])
plt.suptitle("Errors grouped by predicted label")
```

[5]: Text(0.5, 0.98, 'Errors grouped by predicted label')




[6]:
```
# %%
# do the same, but this time on a sample of wrong preditions to see
# error proportions
ierror_s = np.random.choice(np.where(perror)[0], 100, replace=False)
nextplot()
showdigits(Xtest[ierror_s, :], yhat[ierror_s])
plt.suptitle("Errors grouped by predicted label")
```

[6]:
```
Text(0.5, 0.98, 'Errors grouped by predicted label')
```



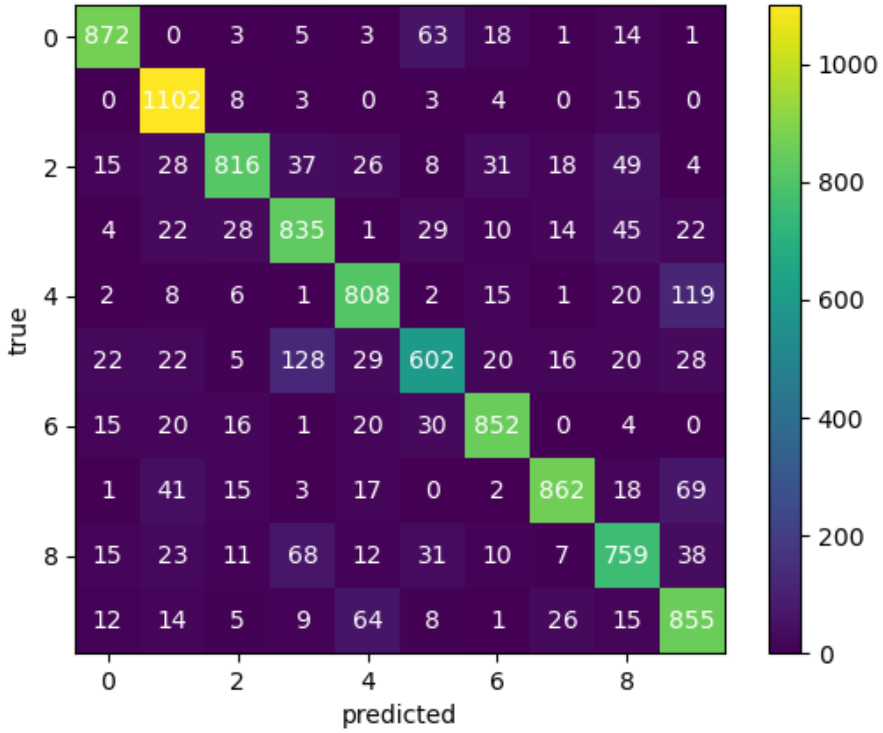

[7]:
```
# %%
# now let's look at this in more detail
print(sklearn.metrics.classification_report(ytest, yhat))
print(sklearn.metrics.confusion_matrix(ytest, yhat))  # true x predicted
```

```
              precision    recall  f1-score   support

           0       0.91      0.89      0.90       980
           1       0.86      0.97      0.91      1135
           2       0.89      0.79      0.84      1032
           3       0.77      0.83      0.80      1010
           4       0.82      0.82      0.82       982
           5       0.78      0.67      0.72       892
           6       0.88      0.89      0.89       958
           7       0.91      0.84      0.87      1028
           8       0.79      0.78      0.79       974
           9       0.75      0.85      0.80      1009

    accuracy                           0.84     10000
   macro avg       0.84      0.83      0.83     10000
weighted avg       0.84      0.84      0.84     10000
```

```
[[ 872    0    3    5    3   63   18    1   14    1]
 [   0 1102    8    3    0    3    4    0   15    0]
 [  15   28  816   37   26    8   31   18   49    4]
 [   4   22   28  835    1   29   10   14   45   22]
 [   2    8    6    1  808    2   15    1   20  119]
 [  22   22    5  128   29  602   20   16   20   28]
 [  15   20   16    1   20   30  852    0    4    0]
 [   1   41   15    3   17    0    2  862   18   69]
 [  15   23   11   68   12   31   10    7  759   38]
 [  12   14    5    9   64    8    1   26   15  855]]
```

[8]:
```
# %%
# plot the confusion matrix
nextplot()
M = sklearn.metrics.confusion_matrix(ytest, yhat)
plt.imshow(M, origin="upper")
for ij, v in np.ndenumerate(M):
    i, j = ij
    plt.text(j, i, str(v), color="white", ha="center", va="center")
plt.xlabel("predicted")
plt.ylabel("true")
plt.colorbar()
```

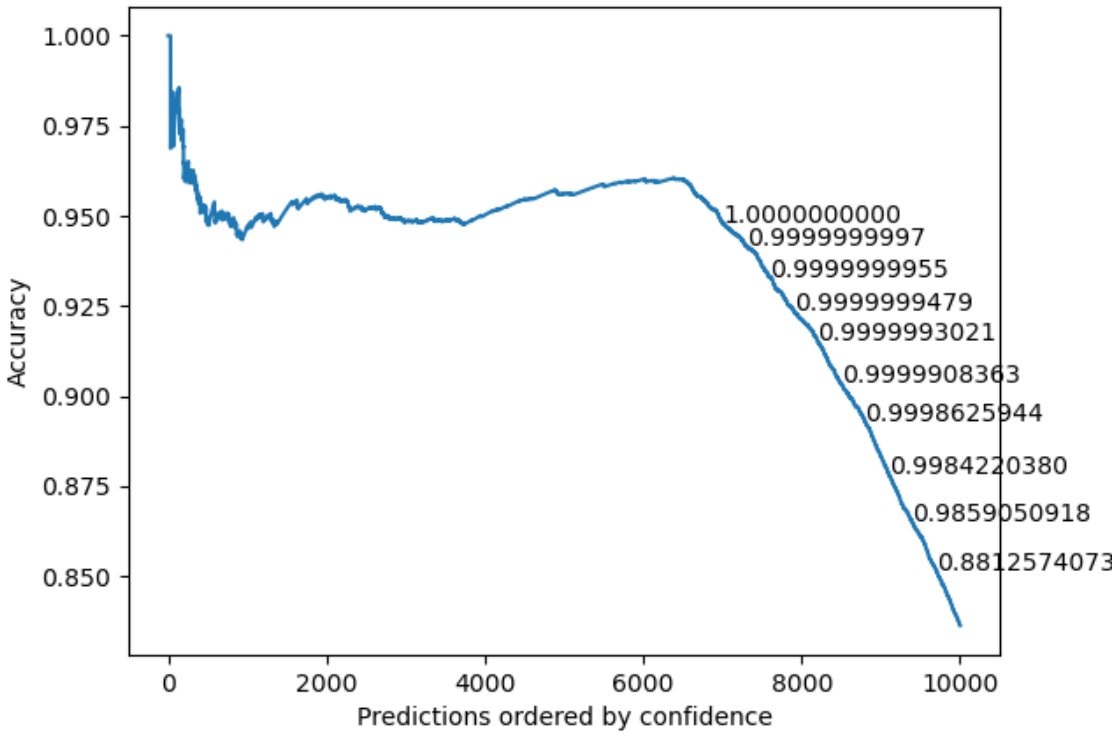
[8]:
```
<matplotlib.colorbar.Colorbar at 0x3467e32c0>
```




[9]:
```
# %%
# cumulative accuracy for predictions ordered by confidence (labels show␣
 ↪predicted
# confidence)
order = np.argsort(logprob)[::-1]
accuracies = np.cumsum(ytest[order] == yhat[order]) / (np.arange(len(yhat)) + 1)
nextplot()
plt.plot(accuracies)
plt.xlabel("Predictions ordered by confidence")
plt.ylabel("Accuracy")
for x in np.linspace(0.7, 1, 10, endpoint=False):
    index = int(x * (accuracies.size - 1))
    print(np.exp(logprob[order][index]))
    plt.text(index, accuracies[index], "{:.10f}".format(np.
 ↪exp(logprob[order][index])))
```

```
0.9999999999822649
0.9999999996949782
0.9999999955447265
0.9999999478873192
0.999999302093004
```

```
0.9999908362580441
0.9998625944161882
0.9984220379937704
0.9859050917808865
0.8812574072791101
```




[10]:
```
# %%
# Accuracy for predictions grouped by confidence (labels show
# predicted confidence). Make the plot large (or reduce number of bins) to see
# the labels.
bins = (np.linspace(0, 1, 50) * len(yhat)).astype(int)
mean_accuracy = [
    np.mean(ytest[order][bins[i] : bins[i + 1]] == yhat[order][bins[i] : bins[i␣
 ↪+ 1]])
    for i in range(len(bins) - 1)
]
nextplot()
plt.bar(np.arange(len(mean_accuracy)), mean_accuracy)
plt.xticks(
    np.arange(len(mean_accuracy)),
    [
```

```
        "{:.10f}".format(x)
        for x in np.exp(logprob[order][np.append(bins[1:-1], len(yhat) - 1)])
    ],
)
plt.gcf().autofmt_xdate()
plt.xlabel("Confidence bin")
plt.ylabel("Accuracy")
```

[10]: Text(0, 0.5, 'Accuracy')