

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Example 3.3 Binary confusion matrix for the MNIST dataset
5  Created for EMCH 504 at USC
6  @author: Austin Downey
7  """
8
9  import IPython as IP
10 IP.get_ipython().run_line_magic('reset', '-sf')
11
12 import numpy as np
13 import scipy as sp
14 import matplotlib as mpl
15 import matplotlib.pyplot as plt
16 import sklearn as sk
17 from sklearn import linear_model
18 from sklearn import pipeline
19 from sklearn import datasets
20 from sklearn import metrics
21
22 cc = plt.rcParams['axes.prop_cycle'].by_key()['color']
23 plt.close('all')
24
25
26 %% Load your data
27
28 # Fetch the MNIST dataset from openml
29 mnist = sk.datasets.fetch_openml('mnist_784', as_frame=False, parser='auto')
30 X = mnist['data'] # load the data
31 Y = np.asarray(mnist['target'], dtype=int) # load the target
32
33 # Split the data set up into a training and testing data set
34 X_train = X[0:60000,:]
35 X_test = X[60000:,:]
36 Y_train = Y[0:60000]
37 Y_test = Y[60000:]
38
39 %% Train a Stochastic Gradient Descent classifier
40
41 # Extract a subset for our "5-dector".
42 Y_train_5 = (Y_train == 5)
43 Y_test_5 = (Y_test == 5)
44
45 # build and train the classifier
46 sgd_clf = sk.linear_model.SGDClassifier()
47 sgd_clf.fit(X_train, Y_train_5)
48
49 # we can now test this for the "5" that we plotted earlier.
50 digit_id = 35
51 test_digit = X[digit_id,:]
52 print(sgd_clf.predict([test_digit]))
53
54 %% Build the Confusion Matrices
55
56 # Return the predictions made on each test fold
57 X_train_pred = sgd_clf.predict(X_train)
58
59
60 # build the confusion Matrix
61 print(sk.metrics.confusion_matrix(Y_train_5, X_train_pred))
62
63 # Now let's find all the False positive and false negative
64 confusion_booleans = np.vstack((Y_train_5, X_train_pred)).T
65 FN_index = np.where((confusion_booleans == [True, False]).all(axis=1))[0]
66 FP_index = np.where((confusion_booleans == [False, True]).all(axis=1))[0]
67

```

```

68 # We built a 5-detector, so:
69 # True and True is true positive (TP)
70 # False and False is True Negative (TN)
71 # True and False is false negative (FN)
72 # False and True is false positive (FP)
73
74
75 # from this, we see #0 is a false negative (FN), i.e. its is actually a 5 but
76 # the classifier said it was not. We can plot this digit below
77 digit_id = 0
78 test_digit = X[digit_id,:]
79 digit_resaped = np.reshape(test_digit,(28,28))
80 plt.figure()
81 plt.imshow(digit_resaped,cmap = mpl.cm.binary,interpolation="nearest")
82 plt .title('A "'+str(Y[digit_id])+'" digit from the MNIST dataset')
83 plt.xlabel('pixel column number')
84 plt.ylabel('pixel row number')
85
86 # Lastly, we see that #68 is classified as an false positive (FP), again, we can plot
87 this.
88 digit_id = 161
89 test_digit = X[digit_id,:]
90 digit_resaped = np.reshape(test_digit,(28,28))
91 plt.figure()
92 plt.imshow(digit_resaped,cmap = mpl.cm.binary,interpolation="nearest")
93 plt .title('A "'+str(Y[digit_id])+'" digit from the MNIST dataset')
94 plt.xlabel('pixel column number')
95 plt.ylabel('pixel row number')
96
97

```