```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Example 3.3 Binary confusion matirx for the MINST dataset
Created for EMCH 504 at USC
@author: Austin Downey
"""

import IPython as IP
IP.get_ipython().run_line_magic('reset', '-sf')

import numpy as np
import scipy as sp
import matplotlib as mpl
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn import linear_model
from sklearn import pipeline
from sklearn import datasets
from sklearn import metrics

cc = plt.rcParams['axes.prop_cycle'].by_key()['color']
plt.close('all')


#%% Load your data

# Fetch the MNIST dataset from openml
mnist = sk.datasets.fetch_openml('mnist_784',as_frame=False,parser='auto')
X = mnist['data']    # load the data
Y = np.asarray(mnist['target'],dtype=int)  # load the target

# Split the data set up into a training and testing data set
X_train = X[0:60000,:]
X_test = X[60000:,:]
Y_train = Y[0:60000]
Y_test = Y[60000:]

#%% Train a Stochastic Gradient Descent classifier

# Extract a subset for our "5-dector".
Y_train_5 = (Y_train == 5)
Y_test_5 = (Y_test == 5)

# build and train the classifier
sgd_clf = sk.linear_model.SGDClassifier()
sgd_clf.fit(X_train, Y_train_5)

# we can now test this for the "5" that we plotted earlier.
digit_id = 35
test_digit = X[digit_id,:]
print(sgd_clf.predict([test_digit]))

#%% Build the Confusion Matrices

# Return the predictions made on each test fold
X_train_pred = sgd_clf.predict(X_train)


# build the confusion Matrix
print(sk.metrics.confusion_matrix(Y_train_5, X_train_pred))

# Now let's find all the False positive and false negative
confusion_booleans = np.vstack((Y_train_5, X_train_pred)).T
FN_index = np.where((confusion_booleans == [True,False]).all(axis=1))[0]
FP_index = np.where((confusion_booleans == [False,True]).all(axis=1))[0]

```

```
# We built a 5-detector, so:
# True and True is true positive (TP)
# False and False is True Negative (TN)
# True and False is false negative (FN)
# False and True is false positive (FP)


# from this, we see #0 is a false negative (FN), i.e. its is actually a 5 but
# the classifier said it was not. We can plot this digit below
digit_id = 0
test_digit = X[digit_id,:]
digit_reshaped = np.reshape(test_digit,(28,28))
plt.figure()
plt.imshow(digit_reshaped,cmap = mpl.cm.binary,interpolation="nearest")
plt .title('A "'+str(Y[digit_id])+'" digit from the MNIST dataset')
plt.xlabel('pixel column number')
plt.ylabel('pixel row number')

# Lastly, we see that #68 is classified as an false positive (FP), again, we can plot
this.
digit_id = 161
test_digit = X[digit_id,:]
digit_reshaped = np.reshape(test_digit,(28,28))
plt.figure()
plt.imshow(digit_reshaped,cmap = mpl.cm.binary,interpolation="nearest")
plt .title('A "'+str(Y[digit_id])+'" digit from the MNIST dataset')
plt.xlabel('pixel column number')
plt.ylabel('pixel row number')
```