```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Example 3.4 Precision and recall accuracy for the MNIST dataset

Developed for Machine Learning for Mechanical Engineers at the University of
South Carolina

@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.pyplot as plt
import sklearn as sk
from sklearn import linear_model
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 = np.asarray(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-detector".
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)

#%% Build the Confusion Matrices

# Return the predictions made with the trained model
X_train_pred = sgd_clf.predict(X_train)

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

TN = confusion_matrix[0,0]
FP = confusion_matrix[0,1]
FN = confusion_matrix[1,0]
TP = confusion_matrix[1,1]

#%% Calculate Precision and Recall

# calculate the Precision and Recall values using the commands discussed in class
accuracy = (TP + TN)/(TP + TN + FP + FN)
precision = TP/(TP+FP)
recall = TP/(TP+FN)
```

```
# of course, SK learn has built-in functions for this.
accuracy_SK = sk.metrics.accuracy_score(Y_train_5, X_train_pred)
precision_SK = sk.metrics.precision_score(Y_train_5, X_train_pred)
recall_SK = sk.metrics.recall_score(Y_train_5, X_train_pred)

# compute the F1 score for the data set
F1 = sk.metrics.f1_score(Y_train_5, X_train_pred)

#%% plot the precision and recall over the threshold domain.

# first, compute the scores for the predictions.
Y_scores= sgd_clf.decision_function(X_train)

#Y_scores= Y3
# now, for the scores and the target training set, calculate the precisions, recalls, threshold values.
precisions, recalls, thresholds = sk.metrics.precision_recall_curve(Y_train_5, Y_scores)

# now, compute the F1 score over the entire threshold range
F1s = 2/(1/precisions + 1/recalls)

# plot the Precision and Recall vs threshold.
plt.figure()
plt.plot(thresholds, precisions[:-1], "--", label="Precision")
plt.plot(thresholds, recalls[:-1], "-", label="Recall")
plt.plot(thresholds, F1s[:-1], ":", label="F1 score")
plt.xlabel("Threshold")
plt.ylabel("normalized precision\nand recall index")
plt.legend(loc=6,framealpha=1)
plt.ylim([-0.05, 1.05])
plt.grid(True)
plt.tight_layout()
```