```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Example 3.6 Multiclass Stochastic Gradient Descent (SDG) for the MINST data set

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

@author: austin_downey
"""

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

import numpy as np
import scipy as sp
import pandas as pd
from scipy import fftpack, signal # have to add
import matplotlib as mpl
import matplotlib.pyplot as plt
import sklearn as sk
import time as time
from sklearn import linear_model
from sklearn import pipeline
from sklearn import datasets
from sklearn import multiclass

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)
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 Multiclass Stochastic Gradient Descent classifiers

# SK learn has a  Multiclass and multilabel module as sk.multiclass. You can use
# this module to do one-vs-the-rest or one-vs-one classification.

# here we test a one-vs-rest classifier that uses Stochastic Gradient Descent
tt_1 = time.time()
ovr_clf = sk.multiclass.OneVsRestClassifier(sk.linear_model.SGDClassifier())
ovr_clf.fit(X_train, Y_train)
print('One-vs-Rest took '+str(time.time()-tt_1 )+' seconds to train and execute')

# here we test a one-vs-one classifier that uses Stochastic Gradient Descent
tt_1 = time.time()
ovo_clf = sk.multiclass.OneVsOneClassifier(sk.linear_model.SGDClassifier())
ovo_clf.fit(X_train, Y_train)
print('One-vs-one took '+str(time.time()-tt_1 )+' seconds to train and execute')

# Moreover, Scikit-Learn detects when you try to use a binary classification algorithm for
# a multiclass classification task, and it automatically runs OvA (except for SVM classifiers for which it uses OvO).
tt_1 = time.time()
```

```
multi_sgd_clf = sk.linear_model.SGDClassifier()
multi_sgd_clf.fit(X_train, Y_train) # y_train, not y_train_5
print('SK learns automated selection (OvA) took '+str(time.time()-tt_1 )+' seconds to train and execute')

```