from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

import numpy as np

import argparse

import sys

import os

def accuracy( C ):

''' Compute accuracy given Numpy array confusion matrix C. Returns a floating point value '''

print ('TODO')

def recall( C ):

''' Compute recall given Numpy array confusion matrix C. Returns a list of floating point values '''

print ('TODO')

def precision( C ):

''' Compute precision given Numpy array confusion matrix C. Returns a list of floating point values '''

print ('TODO')

def class31(filename):

''' This function performs experiment 3.1

Parameters

filename : string, the name of the npz file from Task 2

Returns:

X\_train: NumPy array, with the selected training features

X\_test: NumPy array, with the selected testing features

y\_train: NumPy array, with the selected training classes

y\_test: NumPy array, with the selected testing classes

i: int, the index of the supposed best classifier

'''

print('TODO Section 3.1')

return (X\_train, X\_test, y\_train, y\_test,iBest)

def class32(X\_train, X\_test, y\_train, y\_test,iBest):

''' This function performs experiment 3.2

Parameters:

X\_train: NumPy array, with the selected training features

X\_test: NumPy array, with the selected testing features

y\_train: NumPy array, with the selected training classes

y\_test: NumPy array, with the selected testing classes

i: int, the index of the supposed best classifier (from task 3.1)

Returns:

X\_1k: numPy array, just 1K rows of X\_train

y\_1k: numPy array, just 1K rows of y\_train

'''

print('TODO Section 3.2')

return (X\_1k, y\_1k)

def class33(X\_train, X\_test, y\_train, y\_test, i, X\_1k, y\_1k):

''' This function performs experiment 3.3

Parameters:

X\_train: NumPy array, with the selected training features

X\_test: NumPy array, with the selected testing features

y\_train: NumPy array, with the selected training classes

y\_test: NumPy array, with the selected testing classes

i: int, the index of the supposed best classifier (from task 3.1)

X\_1k: numPy array, just 1K rows of X\_train (from task 3.2)

y\_1k: numPy array, just 1K rows of y\_train (from task 3.2)

'''

print('TODO Section 3.3')

def class34( filename, i ):

''' This function performs experiment 3.4

Parameters

filename : string, the name of the npz file from Task 2

i: int, the index of the supposed best classifier (from task 3.1)

'''

print('TODO Section 3.4')

if \_\_name\_\_ == "\_\_main\_\_":

parser.add\_argument("-i", "--input", help="the input npz file from Task 2", required=True)

args = parser.parse\_args()

# TODO : complete each classification experiment, in sequence.