```python
# -*- coding: utf-8 -*-
"""
Created on Tue Dec 28 23:04:52 2021

@author: Utilisateur
"""
import os
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
os.chdir("C:/Users/Utilisateur/Documents/GitHub/PAr_135/Files")

def sub_bool(a,b):
    if a :
        if b :
            return 0
        else :
            return 1
    else :
        if b :
            return 1
        else :
            return 0


def norm_1(matrix_1,matrix_2):
    distance = 0
    dimensions=np.shape(matrix_1)
    if dimensions!=np.shape(matrix_2):
        return ('Pas la même taille')
    for i in range(dimensions[0]) :
        for j in range(dimensions[1]):
            distance+=sub_bool(matrix_1[i,j],matrix_2[i,j])
    return distance

def norm_1_KNN(matrix_1,matrix_2):
    distance = 0
    dimensions=np.shape(matrix_1)[0]
    if dimensions!=np.shape(matrix_2)[0]:
        return (np.inf())
    for i in range(dimensions) :
            distance+=sub_bool(matrix_1[i],matrix_2[i])
    return distance


###creating training and testing data set

nb_class=1000
size_fault_matrix=(int(55),int(108))
numb_coeff=int(size_fault_matrix[0]*size_fault_matrix[1])
rate_train=0.8
"""
trainset=[]
testset=[]
label=[]

for i in range (nb_class):
    file_name='intermittent_fault_file/intermittent_fault_array'+str(i)+'.npz'
    Matrix_list=np.load(file_name)['matrix']

    number_intermittent=np.shape(Matrix_list)[2]
    sample_size=int(number_intermittent*rate_train)
    sampple_size=int(number_intermittent*(1-rate_train))
    Matrix_fault_test=np.zeros((sample_size,numb_coeff),dtype=bool)
    Matrix_fault_train=np.zeros((sample_size,numb_coeff),dtype=bool)
    Matrix_remodeled=np.reshape(Matrix_list,(number_intermittent,numb_coeff))
    for j in range(sample_size):
        label+=[i+1]
        Matrix_fault_train[j,:]=Matrix_remodeled[j,:]
    for j in range(sampple_size):
        Matrix_fault_test[j,:]=Matrix_remodeled[number_intermittent-j-1,:]
```

```
    trainset.append(Matrix_fault_train)
    testset.append(Matrix_fault_test)
label=np.array(label,dtype=int)
print('bug')
matrix_trainset=np.concatenate(trainset,axis=0)
matrix_testset=np.concatenate(testset,axis=0)
np.savez_compressed('trainset_KNN',matrix=matrix_trainset,label=label)
np.savez_compressed('testset_KNN',matrix=matrix_testset,label=label)

print('datasets are ready')


trainset=np.load('trainset_KNN.npz')['matrix']
label=np.load('trainset_KNN.npz')['label']
model=KNeighborsClassifier(n_neighbors = 5,algorithm='auto')
model.fit(trainset,label)

print('boring part is over ?')
X_test=np.load('testset_KNN.npz')['matrix']
y_test=np.load('testset_KNN.npz')['label']
#result=model.predict(X_test)[0:5]
#true_result=Y_test[0:5]
"""

"""
totalset=[]
tottalset=[]
label=[]
labbel=[]
for i in range (nb_class):
    file_name='intermittent_fault_file/intermittent_fault_array'+str(i)+'.npz'
    Matrix_list=np.load(file_name)['matrix']
    number_intermittent=np.shape(Matrix_list)[2]
    Matrix_remodeled=np.reshape(Matrix_list,(number_intermittent,numb_coeff))
    Matrix_remodeled_dtypebool=np.zeros((number_intermittent,numb_coeff),dtype=bool)
    Matrix_remodeled_dtypebool[:,:]=Matrix_remodeled[:,:]
    label+=[(i+1)//50]*number_intermittent
    totalset.append(Matrix_remodeled_dtypebool)

matrix_totalset=np.concatenate(totalset,axis=0)
np.savez_compressed('totalset_KNN',matrix=matrix_totalset,label=label)


print('dataset is ready')
"""

#Allows for less modulation of the training and testing set
#but it is quicker to code, these lines are more like a reminder
#that we can use the train_test_split function


totalset=np.load('totalset_KNN.npz')['matrix']
label=np.load('totalset_KNN.npz')['label']
,
X_train, X_test, y_train, y_test = train_test_split(totalset, label, test_size=0.2, random_state=1)
model=KNeighborsClassifier(n_neighbors = 5,algorithm='auto',metric=norm_1_KNN)
model.fit(X_train,y_train)




test=model.score(X_test,y_test)

# with only 6 class it took quite some time, 80% training set so approximately 80
# training matrix per class --> give a 32% success rate
# plus 1/6 of test matrix were training matrix so normally 16% of test matrix
# are already known
# after test it was because the norm takes too much time to calculate, i don't know why though
# might have the same effect on the metaclass creation

#anyway even with pre-created norms, it still takes too much time
"""
totalset=np.load('totalset_KNN.npz')['matrix']
label=np.load('totalset_KNN.npz')['label']
X_train, X_test, y_train, y_test = train_test_split(totalset, label, test_size=0.01, random_state=1)
gnb = BernoulliNB()
```

```
y_pred = gnb.fit(X_train, y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d" % (X_test.shape[0], (y_test != y_pred).sum()))
"""
# approximately 5000 training samples per class

# with Multinomial model,Number of mislabeled points out of a total 286 points : 270
# so an approximate 10% success rate

# with Gaussian model, Number of mislabeled points out of a total 286 points : 269
# pretty much the same success rate

# pretty much the same with Bernoulli model : 269 mislabeled points
```