```
#Handling the import

import numpy as np
from numpy import linalg as LA

import matplotlib.pyplot as plt
from matplotlib.offsetbox import AnnotationBbox, OffsetImage
#%matplotlib inline
from numpy.linalg import norm
import warnings
warnings.filterwarnings("ignore")
import random

#Set dataset
random.seed(17)
Trainsetx = []
Trainsety = []
for x in range(1,2001):
    Trainlst = np.random.normal(0,1,10)
    Trainsetx.append(Trainlst)
    if(sum(np.square(Trainlst)) > 9.34):
        Trainsety.append(1)
    else:
        Trainsety.append(-1)

Testsetx = []
Testsety = []

for x in range(1,10001):
    Testslst = np.random.normal(0,1,10)
    Testsetx.append(Testslst)
    if(sum(np.square(Testslst)) > 9.34):
        Testsety.append(1)
    else:
        Testsety.append(-1)

def genlst(datx,daty,k,p,Weight,sumwei):
    decpoint = datx[p][k]
    predictionlst = []
    sig = 1

    for x in range(0,np.shape(Trainsetx)[0]):
        if(datx[x][k] > decpoint):
            predictionlst.append(1)
        else:
            predictionlst.append(-1)

    weiavg = Weightpred(Weight,Trainsety,predlst,sumwei)
    if(weiavg > 0.5):
        sig = -1
        predictionlst = list(map(lambda x: -1 * x, predictionlst))

    return predictionlst, sig, weiavg


#weighted prediction
def Weightpred(Weight,daty,lst,sumwei):
    return sum(list(map(lambda a,b,c: (a != b)*c/sumwei, daty ,lst,Weight)))


def predacc(daty,lst):
    return sum(list(map(lambda a,b: a == b, daty ,lst)))/(np.shape(daty)[0])

def eva(adawe, Testx,Trainx):
    sumpred = np.zeros(np.shape(Testx)[0])
    for x in range(0,np.shape(adawe)[0]):
        cureval = adawe[x]
        a = cureval[0]
        k = cureval[1]
        adaweight = cureval[2]
        sign = cureval[3]

        decpoint = Trainx[a][k]
        predictionlst = []

        for x in range(0,np.shape(Testx)[0]):
            if(sign == 1):
                if(Testx[x][k] > decpoint):
                    predictionlst.append(1)
                else:
                    predictionlst.append(-1)
            else:
                if(Testx[x][k] <= decpoint):
                    predictionlst.append(1)
                else:
                    predictionlst.append(-1)

        sumpred = np.add(sumpred,predictionlst)


    return np.sign(sumpred)

 #Ada boost
Itera = 25
Weig = np.full((1,2000),1/2000)[0]
```

```
sumwei = 1
adapara = []

trainacc = []
testacc = []

random.seed(17)
for it in range(0, Itera):
    print(it)

    #Accuracy error
    acc_score = 1

    #Best score
    best_lst = []
    best_k = 0
    best_a = 0
    best_sign = 0

    #generate the best weak classflier
    for k in range(0,10):
        print(k)
        for a in range (0,2000):
            predlst,sig,locwei = genlst(Trainsetx,Trainsety,k,a,Weig,sumwei)
            if(locwei < acc_score):
                acc_score = locwei
                best_lst = predlst
                best_k = k
                best_a = a
                best_sign = sig



    #boostlst = genlst2(Trainsetx,randomk,randoma)
    #acc_score = Weightpred(Weig,Trainsety,boostlst)

    #Printout weak class
    print("{} {} : Weighted Accuracy {}".format(best_a, best_k ,acc_score))
    ada_weig =  np.log((1-acc_score)/acc_score)
    Weig =  list(map(lambda a,b: a * np.exp(-(ada_weig)*b), Weig ,list(map(lambda 
a,b: a != b, Trainsety ,best_lst))))
    sumwei = sum(Weig)


    adapara.append([best_a,best_k,ada_weig,best_sign])
    print("Ada wei {}:".format(ada_weig))

    PredTrainlst = eva(adapara,Trainsetx,Trainsetx)
    PredTrainres = predacc(PredTrainlst,Trainsety)

    print("Training Accuracy {}:".format(PredTrainres))


    PredTestlst = eva(adapara,Testsetx,Trainsetx)
    PredTestres = predacc(PredTestlst,Testsety)

    print("Testing Accuracy {}:".format(PredTestres))


    trainacc.append(1-PredTrainres)
    testacc.append(1-PredTestres)
    print("______________________________________________________________")

plt.plot(list(range(1,np.shape(trainacc)[0]+1)),trainacc)
plt.plot(list(range(1,np.shape(trainacc)[0]+1)),testing)
plt.show()
```