```
import matplotlib.pyplot as plt
from pandas import read_csv

########################   CONTROL PARAMETERS
##############################
zz = 2970    # length of dataset
ds = 1       # downsampling
div = 20     # no. of vertical divisions
Catch = 100  # no of least countinuous max sensors instance to be caught
as an anomaly
Cont = 5    # sequences of data to be used to calculate hidden markov
model probability
##################    training data       ############################
train_filename = "000_Et_H_CO_n.csv"

#################  testing data with 1.25 times
############################
#test_filename = "000_Et_H_CO_n_7_1.30_1950_2150.csv"
#test_filename = "000_Et_H_CO_n_7_1.50_1950_2150.csv"
#test_filename = "032_Et_H_CO_n_3_1.25_1050_1230.csv"
#test_filename = "032_Et_H_CO_n_3_1.50_1050_1230.csv"
#test_filename = "066_Et_H_CO_n_3_1.25_2200_2450.csv"
#test_filename = "066_Et_H_CO_n_3_1.50_2200_2450.csv"
#test_filename = "098_Et_H_CO_n_2_1.30_1350_1550.csv"
#test_filename = "098_Et_H_CO_n_2_1.50_1350_1550.csv"


# train sensors : collectionn of individual sensor inputs into packets of
ds data each
train_dataset = read_csv(train_filename)
train_array = train_dataset.values

a,b,c,d,e,f,g,h = [],[],[],[],[],[],[],[]
train_sensors = [a,b,c,d,e,f,g,h]
count = 3
sum =0
n=1
for i in train_sensors:
    while(n<zz and count<11):
        for k in range(ds*(n-1),ds*n):
            sum += train_array[k][count]
        i.append(sum/ds)
        sum=0
        n+=1
    count+=1
    n=1

plt.plot(a)
plt.plot(b)
plt.plot(c)
plt.plot(d)
plt.plot(e)
plt.plot(f)
plt.plot(g)
plt.plot(h)
plt.legend(['1', '2', '3', '4','5','6','7','8'], loc='upper left')
plt.show()
print("Training data plot")
```

```
# test_sensors : collectionn of individual sensor inputs into packets of
ds data each
test_dataset = read_csv(test_filename)
test_array = test_dataset.values

aa,bb,cc,dd,ee,ff,gg,hh = [],[],[],[],[],[],[],[]
test_sensors = [aa,bb,cc,dd,ee,ff,gg,hh]
count = 3
sum =0
n=1
for i in test_sensors:
    while(n<zz and count<11):
        for k in range(ds*(n-1),ds*n):
            sum += test_array[k][count]
        i.append(sum/ds)
        sum=0
        n+=1
    count+=1
    n=1

plt.plot(aa)
plt.plot(bb)
plt.plot(cc)
plt.plot(dd)
plt.plot(ee)
plt.plot(ff)
plt.plot(gg)
plt.plot(hh)
plt.legend(['1', '2', '3', '4','5','6','7','8'], loc='upper left')
plt.show()
print("Testing data plot")

######################           min max fitting
###################
minimum,maximum,width = [],[],[]
for i in range(8):
    j = train_sensors[i]
    k = test_sensors[i]

    m = min(min(j),min(k))      ###changes
    n = max(max(j),max(k))

    minimum.append(m)
    maximum.append(n)
    w = (n - m)/div
    width.append(w)

###################          Vertical disribution
######################

m,n,o,p,q,r,s,t = [],[],[],[],[],[],[],[]
train_values = [m,n,o,p,q,r,s,t]

mm,nn,oo,pp,qr,rr,ss,tt = [],[],[],[],[],[],[],[]
test_values = [mm,nn,oo,pp,qr,rr,ss,tt]

for i in range(8):
    j = train_values[i]
    k = test_values[i]
    w = width[i]
```

```
    m = minimum[i]
    for l in train_sensors[i]:
        for n in range(div+1):
            if l >= (m + n*w) and l < (m + (n+1)*w):
                j.append(n)
                break                    ###changes
    for l in test_sensors[i]:
        for n in range(div+1):
            if l >= (m + n*w) and l < (m + (n+1)*w):
                k.append(n)
                break

plt.plot(train_values[0])
plt.plot(train_values[1])
plt.plot(train_values[2])
plt.plot(train_values[3])
plt.plot(train_values[4])
plt.plot(train_values[5])
plt.plot(train_values[6])
plt.plot(train_values[7])
plt.legend(['1', '2', '3', '4','5','6','7','8'], loc='upper left')
plt.show()
print("Vertical Assignment of Training data plot")

plt.plot(test_values[0])
plt.plot(test_values[1])
plt.plot(test_values[2])
plt.plot(test_values[3])
plt.plot(test_values[4])
plt.plot(test_values[5])
plt.plot(test_values[6])
plt.plot(test_values[7])
plt.legend(['1', '2', '3', '4','5','6','7','8'], loc='upper left')
plt.show()
print("Vertical Assignment of Testing data plot")

####################       Variation in data streams
######################
var_stream = []

for l in range(zz-10):
    diff_array = []  # difference of sensor values b/w training & testing
per instance
    for i in range(8):
        j = train_values[i]
        k = test_values[i]
        diff_array+=[abs(j[l] - k[l])]   ###changes
    var_stream += [diff_array]

ga,gb,gc,gd,ge,gf,gg,gh = [],[],[],[],[],[],[],[]
gaps = [ga,gb,gc,gd,ge,gf,gg,gh]

for i in range(8):
    for j in var_stream:
        gaps[i].append(j[i])

plt.plot(ga)
plt.plot(gb)
plt.plot(gc)
plt.plot(gd)
```

```
plt.plot(ge)
plt.plot(gf)
plt.plot(gg)
plt.plot(gh)
plt.legend(['1', '2', '3', '4','5','6','7','8'], loc='upper left')
plt.show()
print("Training And Testing Differnce")

######################    *********    Markov Model   *********
#########################
#print(var_stream)
###################    Emmision matrix probability assignment
###################
emm_prob = []
sensor_max_list = []
for i in var_stream:
    g_sum=0
    for j in i:
        g_sum += j
    emm_prob.append(8*max(i) - g_sum)   #prob of being an anomaly
    for k in range(8):
        if max(i) == i[k]:
            sensor_max_list.append(k)
            break
#### normalisation
z = max(emm_prob)
for i in range(len(emm_prob)):
    emm_prob[i] = emm_prob[i]/z

##############  Observation data with "sensor_max_list" and "emm_prob"
combined
sequence = []
Observation = []
obs = []
seq = []
for i in range(len(sensor_max_list)):
    obs.append([sensor_max_list[i],emm_prob[i]])
    seq.append(i)
    if i+1 == len(sensor_max_list) or sensor_max_list[i] !=
sensor_max_list[i+1]:
        Observation +=[obs]
        sequence += [seq]
        obs = []
        seq = []

#print(Observation)
#print(sequence)   #tracks
###############    transition matrix and initial state matrix
pi = 0.5
A = [1,0.5]

###############    probability of observation for each block(100 instance
each) of data
P = []
k = Cont
for i in Observation:
    if len(i)>Catch:   #significant deviation considered an anomaly
        Ps = []
        n = len(i)//k   # n = len(i)-Catch    #cont=5 Block of data
        Ps.append( i[0][0] )
```

```
        for c in range(n):
            p = ( i[c*k][1] * pi )
            for j in range(1,k):              #cont
                p = p * ( i[c*k+j][1]) * A[0]
            Ps.append( p )
        P += [Ps]
#print(P)
###################   ************  Final Selection  ***********
###################
A_count        = []     # Location Track of Anomaly
A_intensity    = []     # No of Countinious Anomalous Blocks
A_sensor     = []     # Anomalous Sensor Number (0-7)
A_prob       = []     # Probability of anomalous sequence
for i in P:
    A_sensor.append(i[0])
    A_intensity.append(len(i))
    c = len(i)
    s = 0
    for j in range(1,c):
        s += i[j]
    A_prob += [ s/(c-1) ]

for i in range(len(sequence)):
    k = sequence[i][0]+2     # margin
    if len(sequence[i])>Catch:
        A_count += [k]

######################           Print it out
##################################

print("Seq.   Track   Sensor no.   Intensity   Probability of Anomaly")
for i in range(len(A_sensor)):
    print(" %-5d %-7d %-12d %-14d %-12.10f"%

(i+1,A_count[i],A_sensor[i],A_intensity[i],A_prob[i]))
```