پروژه درس شناسایی الگو:

روش مورد استفاده DTW

استاد:

جناب آقای دکتر مهدی امیری

نام و نام خانوادگی:

فاطمه صالح نیا

نفیسه جعفری

اسفند ماه ۹۷


بسم الله الرحمن الرحيم

فهرست مطالب

چکیده

تشخیص الگو در سری های زمانی، کاربرد های زیادی در دنیای امروز دارد. به همین منظور امروزه روش های تشخیص الگو در سری های زمانی بسیار مورد توجه قرار گرفته اند. روش انتخابی در این پروژه پیچش زمانی پویا (dtw) است. مزیت این روش نسبت به روش های شبکه عصبی این است که اگر رکورد ها تغییر کنند اصلا نیاز به صرف زمان دوباره برای آموزش شبکه نیست؛ چون مدل در زمان اجرا ایجاد خواهد شد و نسبت به الگوریتم های شبکه عصبی سرعت بسیار بالاتری در زمان آموزش دارد. پیچیدگی زمانی این الگوریتم به صورت پایه بسیار بالاست؛ به این منظور از الگوریتم fast dtw برای سرعت بالاتر استفاده شده است.

برای آموزش الگو با استفاده از میانگین گیری مقادیر از ۴۰۰۰۰ رکورد استفاده شده است. برای کلاس بندی رکورد ها از ۳ کلاس استفاده است. برای قسمت تست از ۱۰ فایل استفاده شد. دقت حدود ۷۰ درصد به دست آمد.

فصل اول

فصل اول: سری های زمانی

۱-۱ سری های زمانی چیست؟

در علوم مختلف، به یک توالی یا دنباله از متغیرهای تصادفی که در فاصله های زمانی ثابت نمونه برداری شده باشند، اصطلاحاً سری زمانی یا پیشامد تصادفی در مقطع زمان می‌گویند. به عبارت دیگر منظور از یک سری زمانی مجموعه‌ای از داده‌های آماری است که در فواصل زمانی مساوی و منظمی جمع‌آوری شده باشند. روش‌های آماری که این گونه داده‌های آماری را مورد استفاده قرار می‌دهد مدل های تحلیل سری زمانی نامیده می‌شود. مانند فروش فصلی یک شرکت طی سه سال گذشته. یک سری زمانی مجموعه مشاهدات تصادفی ای است که بر اساس زمان مرتب شده باشند. مثال‌های آن در اقتصاد و حتی رشته‌های مهندسی دیده می‌شود. [۱]

۱-۲ روش های تحلیل برای سری زمانی

روش‌های تحلیل سری زمانی به دو دسته تقسیم می‌شوند: روش‌های دامنه فرکانس و روش‌های دامنه زمان. دسته اول شامل تحلیل طیفی و تحلیل موجک و دسته دوم شامل تحلیل‌های خودهمبستگی و همبستگی متقابل است.

افزون بر این می‌توان روش‌های تحلیل سری زمانی را به دو دسته پارامتری و ناپارامتری تقسیم کرد. در روش‌های پارامتری چنین انگاشته می‌شود که فرایند مانای احتمالاتی دارای ساختاری مشخص است که می‌توان آن را با تعداد اندکی پارامتر (از جمله با استفاده از مدل خودهمبسته یا میانگین متحرک) توصیف کرد. در این روش‌ها هدف تخمین پارامترهای مدلی است که فرایند احتمالاتی را توصیف می‌کند. در مقابل، روش‌های ناپارامتری صریحاً کوواریانس یا طیف فرایند را بدون در نظر گرفتن ساختاری مشخص برای آن تخمین می‌زنند. همچنین می‌توان روش‌های تحلیل سری زمانی را به دسته روش‌های خطی و غیر خطی یا روش‌های تک‌متغیره و چندمتغیره تقسیم کرد.

۱-۳ کاربرد ها

تحلیل سری های زمانی در زمینه آمار، اقتصاد ، زلزله شناسی، هواشناسی، و ژئوفیزیک، هدف اصلی تلقی می شود. همچنین در زمینه پردازش سیگنال، مهندسی کنترل و مهندسی ارتباطات، برای تشخیص و

[1] https://fa.wikipedia.org/wiki/%D8%B3%D8%B1%DB%8C_%D8%B2%D9%85%D8%A7%D9%86%DB%8C

برآورد سیگنال استفاده می شود. زمینه دیگر کاربرد تحلیل سری زمانی در داده کاوی، تشخیص الگو و تجزیه و تحلیل روش های یادگیری ماشین برای خوشه بندی، طبقه بندی، پرس و جو با محتوا، تشخیص ناهنجاری و همچنین پیش بینی می باشد.

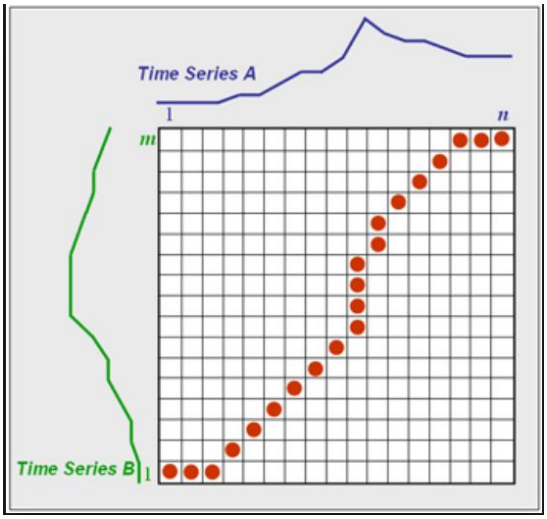
- سری زمانی در اقتصاد، مانند قیمت سهام در روزهای متوالی، صادرات در ماه‌های متوالی، متوسط درآمد در ماه‌های متوالی ...
- سری زمانی فیزیک، بویژه در علوم مربوط به آثار جوی، علوم دریایی، فیزیک زمین (ژئو فیزیک).
- سری‌های زمانی بازاریابی، تجزیه و تحلیل ارقام فروش در هفته یا ماه‌ها متوالی یک مسئلهٔ مهم در تجارت است.
- سری‌های زمانی جمعیت نگاری، اندازه‌گیری سالانه جمعیت با هدف پیش بینی تغییرات جمعیت در مدت زمان ده تا بیست سال آینده.
- فرایندهای دوتایی، سری‌هایی که مشاهدات یکی از دو مقدار که معمولاً با ۰ و ۱ نشان می‌دهند را اختیار کند، که بخصوص در نظریه ارتباطات اتفاق می‌افتد را فرایند دوتایی می‌نامند.
- فرایندهای نقطه ای، نوعی سری زمانی که پیشامدهای رخداده به طور تصادفی در زمان رخداده، زمان‌های رخ دادن تصادفات قطارها.

فصل دوم

۲- فصل دوم الگوریتم DTW

۲-۱ پیچش زمانی پویا

در سری‌های زمانی تاب دادن زمان هوشمند یا کش و قوس زمانی پویا[۲] الگوریتمی برای اندازه‌گیری شباهت بین دو دنباله زمانیست که ممکن است در سرعت یا زمان متفاوت باشند. برای نمونه، DTW می تواند شباهت بین دو الگوی راه رفتن را بیابد حتی اگر سرعت یا شتاب راه رفتنشان در بازه‌های زمانی یکسان نباشد. DTW دنباله‌های زمانی داده‌های صوتی، ویدئویی و تصویری را تحلیل کرده است. در واقع هر داده‌ای که بتواند به صورت دنباله پشت سر هم اطلاعات به دست بیاید، DTW می‌تواند آنرا تحلیل کند. یک کاربرد مهم این الگوریتم بازشناسی گفتار یکسان افراد مختلف با سرعت گفتار مختلف است. و از کاربردهای دیگر DTW می‌توان به تشخیص صدا، تشخیص دستخط و امضا اشاره کرد. همچنین استفاده‌های از آن در تشخیص تطبیق اشکال هندسی جزئی دیده شده‌است. در مجموع، DTW روشی است که بهینه‌ترین تطبیق بین دو دنباله زمانی با محدودیت‌های معین را پیدا می‌کند (برای مثال:سری زمانی). دنباله‌ها به صورت غیر خطی در محور زمان «کش و قوس پیدا می‌کنند» تا معیاری برای شباهت آنها مستقل از برخی تغییرات غیر خطی در محور زمان به دست‌آورد. این روش تنظیم دنباله بسیاری از اوقات در دسته بندی سری زمانی مورد استفاده قرار می‌گیرد. اگرچه DTW معیاری همانند فاصله بین دو دنباله داده شده می‌یابد، تضمین نمی‌کند که نامساوی مثلث در مورد آن برقرار باشد.



تصویر ۱: تصویر مسیر بهینه

[2].Dynamic time warping(DTW)

۲-۲ پیاده سازی

این مثال پیاده‌سازی الگوریتم DTW را برای دو دنباله رشته‌ای s و t از نشانه‌های مجزا ترسیم می‌کند. برای نشانه‌های x، y و d(x,y) برابر فاصله بین نشانه‌ها است. برای مثال d(x,y) = |x−y|

```
int DTWDistance(s: array [1..n], t: array [1..m]) {
    DTW := array [0..n, 0..m]

    for i := 1 to n
        DTW[i, 0] := infinity
    for i := 1 to m
        DTW[0, i] := infinity
    DTW[0, 0] := ۰

    for i := 1 to n
        for j := 1 to m
            cost:= d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],// الحاق
                                        DTW[i  , j-1],// حذف
                                        DTW[i-1, j-1])// تطبیق

    return DTW[n, m]
}
```

تصویر ۲: پیاده سازی الگوریتم dtw

۲-۳ محاسبه سریع

محاسبه الگوریتم DTW در حالت کلی از مرتبه زمانی O(N^2) است. از جمله تکنیک‌های سریع تر برای محاسبه DTW می‌توان به SparseDTW و FastDTW اشاره نمود. یکی از کارهای معمول در این زمینه، به دست آوردن سری‌های زمانی مشابه است که می‌تواند با استفاده از کران پایین‌هایی چون LB_Keogh و LB_Improved تسریع گردد. در یک بررسی، Wang گزارش نموده است که به کمک کران پایین‌های LB_Improved نتایج بهتری از کران پایین LB_Keoghبه دست آورده و نشان داده است که تکنیک‌های دیگر بهینه نیستند.

۲-۴ دنباله متوسط

میانگین گیری از الگوریتم کش و قوس زمانی پویا هم ارز مسئله دنباله میانگین مجموعه‌ای از دنباله هاست. دنباله میانگین، دنباله‌ای است که مجموع مربع‌های فواصل آن از دنباله‌های مجموعه داده شده کمینه باشد. NLAAF روش دقیق حل این مسئله است. برای بیش از دو دنباله، مسئله مربوط به یکی از تنظیم‌های چندگانه است و به روشی هیوریستیک نیاز دارد. در حال حاضر روش مورد استفاده برای به دست آوردن میانگین مجموعه‌ای از دنباله‌ها به صورت پایدار به وسیله DBA،DTW نام دارد. COMASA

با استفاده از DBA به عنوان یک پردازش بهینه‌سازی محلی، جستجو برای دنباله بهینه را به طور بهینه اتفاقی می‌کند.

فصل سوم

۳- فصل سوم تحلیل کد و نمایش نتایج

۳-۱ خواندن دیتاست

دیتاست مورد استفاده در این پروژه فایل bin بوده است. برای استفاده راحت تر ما این فایل ها را به فایل csv تبدیل کردیم.

تبدیل فایل اصلی (**FlyerEu_Tp70_MinSp20_334d_Dsr.bin**) به csv

```
import struct
import datetime
import numpy as np
import csv
# load the dataset
class BinaryReaderEOFException(Exception):
    def __init__(self):
        pass
    def __str__(self):
        return 'Not Enough Info In File...'
class BinaryReader:
    # Map well-known type names into struct format characters.
    typeNames = {

        'uint64' :'Q',
        'double' :'d'

    }

    def __init__(self, fileName):
        self.file = open(fileName, 'rb')

    def read(self, typeName):
        typeFormat = BinaryReader.typeNames[typeName.lower()]
        typeSize = struct.calcsize(typeFormat)
        value = self.file.read(typeSize)
        if typeSize != len(value):
            raise BinaryReaderEOFException
        return struct.unpack(typeFormat, value)[0]

    def __del__(self):
        self.file.close()

def binfile_operation(filename):
```

```
binaryReader = BinaryReader(filename)
row=[]
row_final_csv=[]
 # counting classes  0.65<grand_t<=1
try:
    csv2 = open("FlyerEu_Tp70_MinSp20_334d_Dsr.csv", "w")
    csv_final = open("final_DB_as.csv", "w")
    for i in range(25,40000,25):
        grand_t_final = 0
        grand_t_sum = 0
        sensor1_sum = 0
        sensor1_final = 0
        sensor2_sum = 0
        sensor2_final = 0
        time_sum = 0
        date_median = np.array([],dtype=str)
        counter = i - 25
        while(counter < i + 25):
            grand_t = (binaryReader.read('double'))
            #print(grand_t)
            sensor1 = (binaryReader.read('double'))
            sensor1_sum = sensor1_sum + sensor1
            #print(sensor1)
            sensor2 = (binaryReader.read('double'))
            sensor2_sum = sensor2_sum + sensor2
            #print(sensor2)
            time = (binaryReader.read('uint64'))
            time_sum = time_sum + time
            #print(time)
            if (time == ''):
                break
            secs = int(time / 10.0 ** 7)
            #print(secs)
            delta =datetime.timedelta(seconds=secs)
            datetime.timedelta(733940, 34260)
            ts = datetime.datetime(1,1,1) + delta
            #print(ts)
            fmt = '%Y/%m/%d-%H:%M:%S.%d'
            now_str = ts.strftime(fmt)
            date_median = np.append(date_median,now_str)
            # print(time)
            if (grand_t<=0.35999):
                grand_t=2
                # grand_t_count2 = grand_t_count2 + 1
            elif(0.36<grand_t<=0.65999):
                grand_t=3
                #grand_t_count3 = grand_t_count3 + 1
            elif(0.66<grand_t<=1):
                grand_t=1
                #grand_t_count1 = grand_t_count1 + 1
```

```
                row.append(str(grand_t) + "," + str(sensor1)+","
+ str(sensor2)+ "," + str(time) +"," + now_str +"\n")
                counter=counter+1
                # grand_t_sum += grand_t
                print(row)
                # print(time)
                csv2.write(str(grand_t) + "," + str(sensor1)+","
+ str(sensor2)+ "," + str(time) +"," + now_str +"\n")
                grand_t_final = grand_t

            sensor1_final = sensor1_sum / 50
            sensor2_final = sensor2_sum / 50
            time_final = time_sum / 50
            row_final_csv.append(str(grand_t_final) + "," +
str(sensor1_final)+"," + str(sensor2_final)+ "," +
str(time_final) +"," + date_median[25] +"\n")
            csv_final.write(str(grand_t_final) + "," +
str(sensor1_final)+"," + str(sensor2_final)+ "," +
str(time_final) +"," + date_median[25] +"\n")
        return row

    except BinaryReaderEOFException:
        # One of our attempts to read a field went beyond the end
of the file.
        print ("Error: File seems to be corrupted.")
b3=BinaryReader('FlyerEu_Tp70_MinSp20_334d_Dsr.bin')
b3.read('double')
binfile_operation('FlyerEu_Tp70_MinSp20_334d_Dsr.bin')
print('done')
```

کد تبدیل فایل تست به CSV:

```
import struct
import datetime
import csv

# load the dataset
class BinaryReaderEOFException(Exception):
    def __init__(self):
        pass
    def __str__(self):
        return 'Not enough bytes in file to satisfy read request'

class BinaryReader:
    # Map well-known type names into struct format characters.
    typeNames = {
        'int8'   :'b',
        'uint8'  :'B',
        'int16'  :'h',
```

```
            'uint16' :'H',
            'int32'  :'i',
            'uint32' :'I',
            'int64'  :'q',
            'uint64' :'Q',
            'float'  :'f',
            'double' :'d',

            'char'   :'s'}

        def __init__(self, fileName):
            self.file = open(fileName, 'rb')

        def read(self, typeName):
            typeFormat = BinaryReader.typeNames[typeName.lower()]
            typeSize = struct.calcsize(typeFormat)
            value = self.file.read(typeSize)
            if typeSize != len(value):
                raise BinaryReaderEOFException
            return struct.unpack(typeFormat, value)[0]

        def __del__(self):
            self.file.close()

    def readbinfile(filename):
        binaryReader = BinaryReader(filename)

        row=[]
        counter=0
        try:
            file_csv = open("FlyerZz2_Dsr_test.csv", "w")

            while (counter<800):
                    f1 = (binaryReader.read('double'))
                    #print(f1)
                    f2 = (binaryReader.read('double'))
                    #print(f2)
                    f3 = (binaryReader.read('double'))
                    #print(f3)
                    f4 = (binaryReader.read('uint64'))
                    #print(f4)
                    if (f4 == ''):
                        break
                    secs = int(f4 / 10000000)
                    #print(secs)
                    delta =datetime.timedelta(seconds=secs)
                    datetime.timedelta(733940, 34260)
                    ts = datetime.datetime(1,1,1) + delta
                    #print(ts)
                    fmt = '%Y/%m/%d-%H:%M:%S.%d'
```

```
                now_str = ts.strftime(fmt)
                if (f1<=0.35999):
                    f1=2
                elif(0.36<f1<=0.65999):
                    f1=3
                elif(0.66<f1<=1):
                    f1=1

                row.append(str(f1) + "," + str(f2)+"," + str(f3)+
","  + str(f4) +"," + now_str +"\n")
                counter=counter+1
                print(row)
                file_csv.write(str(f1) + "," + str(f2) + "," +
str(f3) + "," + str(f4) + "," + now_str + "\n")

        return row,counter

# print(rowsc[1],conter)
    except BinaryReaderEOFException:
        # One of our attempts to read a field went beyond the end
of the file.
        print ("Error: File corrupted.")

res=BinaryReader('FlyerZz2_Tp7_MinSp2_Dsr.bin')
res.read('double')
readbinfile('FlyerZz2_Tp7_MinSp2_Dsr.bin')
```

ما برای تحلیل این سری زمانی از الگوریتم dtw استفاده کردیم. این الگوریتم به طور کلاسیک مشکل پیچیدگی زمانی دارد و بسیار کند اجرا می شود. ما در این کد از کتابخانه fastdtw استفاده کردیم که این مشکل را تا حد زیادی کاهش می دهد. این الگوریتم برای حل مشکل ابتدا رزولوشن را کاهش داده سپس خود الگوریتم dtw را اجرا می کند؛ سپس رزولوشن را به مقدار اولیه برمی گرداند.

```
import numpy as np
from scipy.spatial.distance import euclidean

from fastdtw import fastdtw
from dtaidistance import dtw
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from dtaidistance import dtw_visualisation as dtwvis

predict_array = []
test_array = []
def dtw_fuction(df,test,ind):
```

```
    win_lenth = [int((len(df) * 2) / 10)]
    mean_y_test = []
    mean_total = []
    print('====================================')

    win_test = []
    win_per = [] # for period size (20, 40, 80)

    for per in range(0, len(win_lenth), 1):
        win_min = 0
        kj = win_lenth[per]

        dis_array = []
        path_array = []
        win = []
        half_len_win = int(win_lenth[per] / 2)
        for i in range(half_len_win, len(df), half_len_win):
            counter = i - half_len_win
            while (counter < kj):
                win.append(df.iloc[counter:kj])
                counter = counter + 1
                kj = kj + 1
                if kj >= len(df):
                    break;

        print('finish win creation...please wait...')

        print('test file ' + str(ind) + ' compare starting...')

        for ii in range(0, len(win), 1):
            distance, path = fastdtw(win[ii], test, radius=1,
dist=None)


            dis_array.append([distance])
            path_array.append([path])
            minimum_dist = min(dis_array)
            win_min = win[dis_array.index(minimum_dist)]
        print('minimum window:', win_min)
        print('minimum distance:', minimum_dist)
        print('====================================')


        win_per.append(win_min['grand'].iloc[-1])
        predict_array.append(win_per)
    print('Predicted Label:', win_per)
    mean_y_test.append(test['grand'].iloc[-1])
        print('True Label', int(mean_y_test[0]))
    test_array.append(int(mean_y_test[0]))
    print('finished!')
```

```
df = pd.read_csv('final_DB_as.csv', delimiter=',', header=None,
names=['grand', 's1', 's2'], usecols=[0, 1, 2])
print(df)
df.columns = ['grand', 's1', 's2']
for ind in range(1,11,1):
    test = pd.read_csv('FlyerZz2_Dsr_test'+str(ind)+'.csv',
delimiter=',', names=['grand', 's1', 's2'], usecols=[0, 1, 2])
    test.columns = ['grand', 's1', 's2']
    dtw_fuction(df,test,ind)
df.plot(y='grand');
test.plot(y='grand');
plt.legend(loc='best')
```

در نهایت، به وسیله کتابخانه classification_report، می توان گزارش نتایج بدست آمده را در خروجی نمایش داد. علاوه بر این، از طریق تابع accuracy_score نیز مجدداً دقت برنامه محاسبه شد که در خروجی نیز نمایش داده شده است.

```
report = classification_report(test_array, predict_array)
print(report)
from sklearn.metrics import accuracy_score
accuracy=accuracy_score(test_array,predict_array)
print('ACCURACY is: ',accuracy)
```

خروجی نهایی برنامه، در چند بخش ارائه شده است.

بخش اول، نمایش داده ها به صورت میانگین گیری و برچسب دسته براساس آخرین نقطه تعیین شده است. در برنامه،۱۰ مجموعه تست با اندازه های مختلف استفاده شده اند. در خروجی برنامه، به ترتیب برای هر فایل داده تست، مقایسه با بخش های ۲۰ درصدی از داده اصلی انجام شده و براساس الگوریتم استفاده شده، کمترین فاصله و سایز پنجره ای که کمترین فاصله را دارد، نمایش می دهد. در نهایت نیز برچسب پیش بینی شده و برچسب اصلی دسته در خروجی نمایش داده شده اند.

نتیجه نهایی بدست آمده به عنوان دقت برنامه ۷۰ درصد است.

```
======== Report Project Result ====================
              precision    recall  f1-score   support

           1       1.00      0.67      0.80         6
           2       1.00      0.50      0.67         2
           3       0.40      1.00      0.57         2

   micro avg       0.70      0.70      0.70        10
   macro avg       0.80      0.72      0.68        10
weighted avg       0.88      0.70      0.73        10

ACCURACY is:  0.7
```

منابع و مراجع

1. https://fa.wikipedia.org/
2. https://www.kaggle.com/uciml/indian-liver-patient-records
3. https://www.mathworks.com/help/stats/one-way-anova.html
4. https://www.mathworks.com/help/stats/analysis-of-variance-and-covariance.html

پیوست یک:

در این بخش خروجی برنامه درج شده است.

C:\Python36\python.exe E:/final_dtw_project/fast_DTW.py

	grand	s1	s2
0	3	1.060816	1.061437
1	3	1.062158	1.062386
2	3	1.062534	1.062774
3	3	1.062714	1.062923
4	3	1.062434	1.062669
5	3	1.062234	1.062693
6	3	1.062448	1.062784
7	3	1.062287	1.062783
8	2	1.062670	1.063001
9	2	1.062611	1.062818
10	1	1.062290	1.062633
11	1	1.062072	1.062295
12	1	1.061886	1.062149
13	1	1.061622	1.061831
14	1	1.061535	1.061735
15	1	1.061499	1.061699
16	1	1.061649	1.061849
17	1	1.061606	1.061806
18	1	1.061591	1.061791
19	1	1.061561	1.061761
20	1	1.061540	1.061740
21	1	1.061726	1.061926

22	1	1.061905	1.062105
23	1	1.062180	1.062380
24	1	1.062099	1.062303
25	1	1.062115	1.062323
26	1	1.062293	1.062493
27	1	1.062396	1.062596
28	3	1.062399	1.062599
29	3	1.063022	1.063222
...	...	...	...
1569	1	1.060114	1.060314
1570	1	1.060124	1.060324
1571	1	1.060113	1.060313
1572	1	1.060115	1.060315
1573	1	1.060187	1.060387
1574	1	1.060142	1.060342
1575	1	1.060123	1.060323
1576	1	1.060159	1.060359
1577	1	1.060113	1.060313
1578	1	1.060093	1.060293
1579	1	1.060083	1.060283
1580	1	1.060080	1.060280
1581	1	1.060068	1.060268
1582	1	1.060093	1.060293
1583	1	1.060097	1.060297
1584	1	1.060090	1.060290
1585	1	1.060140	1.060340
1586	1	1.060122	1.060322

1587 1 1.060153 1.060353

1588 1 1.060144 1.060344

1589 1 1.060127 1.060327

1590 1 1.060129 1.060329

1591 1 1.060126 1.060326

1592 1 1.060134 1.060334

1593 1 1.060134 1.060334

1594 1 1.060130 1.060330

1595 1 1.060129 1.060329

1596 1 1.060136 1.060336

1597 1 1.060140 1.060340

1598 1 1.060134 1.060334

[1599 rows x 3 columns]

==========

finish win creation...please wait...

test file 1 compare starting...

minimum window: grand s1 s2

1431 2 1.060269 1.060469

1432 2 1.060264 1.060464

1433 2 1.060283 1.060483

1434 2 1.060321 1.060521

1435 2 1.060280 1.060480

1436 2 1.060323 1.060523

1437 2 1.060437 1.060637

1438 2 1.060439 1.060639

1439	2	1.060470	1.060670
1440	2	1.060464	1.060664
1441	2	1.060461	1.060661
1442	2	1.060459	1.060659
1443	2	1.060452	1.060652
1444	2	1.060451	1.060651
1445	2	1.060470	1.060670
1446	2	1.060461	1.060661
1447	2	1.060465	1.060665
1448	2	1.060434	1.060634
1449	2	1.060423	1.060623
1450	2	1.060409	1.060609
1451	2	1.060302	1.060502
1452	2	1.060225	1.060425
1453	2	1.060254	1.060454
1454	2	1.060240	1.060440
1455	2	1.060265	1.060465
1456	2	1.060191	1.060391
1457	3	1.060060	1.060260
1458	3	1.060058	1.060258
1459	2	1.060168	1.060368
1460	2	1.060415	1.060615
...	...	...	...
1569	1	1.060114	1.060314
1570	1	1.060124	1.060324
1571	1	1.060113	1.060313
1572	1	1.060115	1.060315

1573 1 1.060187 1.060387

1574 1 1.060142 1.060342

1575 1 1.060123 1.060323

1576 1 1.060159 1.060359

1577 1 1.060113 1.060313

1578 1 1.060093 1.060293

1579 1 1.060083 1.060283

1580 1 1.060080 1.060280

1581 1 1.060068 1.060268

1582 1 1.060093 1.060293

1583 1 1.060097 1.060297

1584 1 1.060090 1.060290

1585 1 1.060140 1.060340

1586 1 1.060122 1.060322

1587 1 1.060153 1.060353

1588 1 1.060144 1.060344

1589 1 1.060127 1.060327

1590 1 1.060129 1.060329

1591 1 1.060126 1.060326

1592 1 1.060134 1.060334

1593 1 1.060134 1.060334

1594 1 1.060130 1.060330

1595 1 1.060129 1.060329

1596 1 1.060136 1.060336

1597 1 1.060140 1.060340

1598 1 1.060134 1.060334

[168 rows x 3 columns]

minimum distance: [53.038860799999995]

==========

Predicted Label: [1]

True Label 1

finished!

==========

finish win creation...please wait...

test file 2 compare starting...

minimum window: grand s1 s2

660 1 1.059986 1.060186

661 1 1.059625 1.059825

662 1 1.059674 1.059874

663 3 1.059594 1.059794

664 2 1.059715 1.059915

665 3 1.059728 1.059928

666 1 1.059510 1.059710

667 1 1.059508 1.059708

668 1 1.059484 1.059684

669 1 1.059508 1.059708

670 1 1.059706 1.059906

671 1 1.059683 1.059883

672 1 1.059779 1.059979

673 1 1.059770 1.059970

674 3 1.059929 1.060129

675 3 1.059996 1.060196

676 3 1.059996 1.060196

677 3 1.059976 1.060176

678 3 1.059951 1.060151

679 3 1.059969 1.060169

680 3 1.060046 1.060246

681 2 1.060206 1.060406

682 2 1.060370 1.060570

683 2 1.060435 1.060635

684 2 1.060305 1.060505

685 2 1.060127 1.060327

686 2 1.060211 1.060411

687 2 1.060168 1.060368

688 2 1.060030 1.060230

689 3 1.059658 1.059858

.. ..

949 2 1.059258 1.059458

950 2 1.059442 1.059642

951 2 1.059436 1.059636

952 2 1.059412 1.059612

953 2 1.059403 1.059603

954 3 1.059179 1.059379

955 3 1.059236 1.059436

956 3 1.059045 1.059245

957 3 1.059019 1.059219

958 2 1.059041 1.059241

959 3 1.059035 1.059235

960 1 1.058811 1.059011

961 1 1.058643 1.058843

962 1 1.058835 1.059035

963 1 1.058852 1.059052

964 3 1.059126 1.059326

965 3 1.059279 1.059479

966 1 1.059062 1.059262

967 3 1.059185 1.059385

968 3 1.059298 1.059498

969 3 1.059293 1.059493

970 3 1.059188 1.059388

971 3 1.059230 1.059430

972 3 1.059248 1.059448

973 1 1.059187 1.059387

974 1 1.059064 1.059264

975 1 1.059099 1.059299

976 3 1.059203 1.059403

977 3 1.059235 1.059435

978 3 1.059287 1.059487

[319 rows x 3 columns]

minimum distance: [79.23509559999998]

==========

Predicted Label: [3]

True Label 1

finished!

==========

finish win creation...please wait...

test file 3 compare starting...

minimum window: grand s1 s2

347 1 1.060990 1.061190

348 1 1.060967 1.061167

349 1 1.060950 1.061150

350 1 1.060799 1.060999

351 1 1.060696 1.060896

352 1 1.060609 1.060809

353 1 1.060623 1.060823

354 1 1.060770 1.060970

355 1 1.060744 1.060944

356 1 1.060712 1.060912

357 1 1.060753 1.060953

358 1 1.060781 1.060981

359 1 1.060990 1.061190

360 1 1.061021 1.061221

361 1 1.061016 1.061216

362 1 1.061021 1.061221

363 1 1.061142 1.061342

364 1 1.061122 1.061322

365 1 1.061222 1.061422

366 1 1.061185 1.061385

367 1 1.061290 1.061490

368 1 1.061392 1.061592

369 1 1.061338 1.061538

370	1	1.061323	1.061523
371	1	1.061433	1.061633
372	1	1.061337	1.061537
373	1	1.061274	1.061474
374	1	1.061300	1.061500
375	1	1.061270	1.061470
376	1	1.061239	1.061439
..	...	...	...
636	1	1.059306	1.059506
637	1	1.059166	1.059366
638	1	1.059186	1.059386
639	3	1.059345	1.059545
640	1	1.059285	1.059485
641	1	1.059131	1.059331
642	1	1.059199	1.059399
643	1	1.059134	1.059334
644	3	1.059296	1.059496
645	3	1.059539	1.059739
646	2	1.059577	1.059777
647	3	1.059617	1.059817
648	3	1.059418	1.059618
649	3	1.059278	1.059478
650	3	1.059357	1.059557
651	1	1.059185	1.059385
652	1	1.059107	1.059307
653	1	1.059307	1.059507
654	1	1.059276	1.059476

655 1 1.059433 1.059633

656 3 1.059581 1.059781

657 1 1.059644 1.059844

658 3 1.059664 1.059864

659 2 1.059896 1.060096

660 1 1.059986 1.060186

661 1 1.059625 1.059825

662 1 1.059674 1.059874

663 3 1.059594 1.059794

664 2 1.059715 1.059915

665 3 1.059728 1.059928

[319 rows x 3 columns]

minimum distance: [16.09485000000002]

==========

Predicted Label: [3]

True Label 3

finished!

==========

finish win creation...please wait...

test file 4 compare starting...

minimum window: grand s1 s2

209 1 1.061186 1.061386

210 1 1.061244 1.061444

211 1 1.061216 1.061416

212 1 1.061386 1.061586

213 1 1.061527 1.061727

214 1 1.061588 1.061788

215 1 1.061743 1.061943

216 1 1.061590 1.061790

217 1 1.061594 1.061794

218 1 1.061631 1.061831

219 1 1.061706 1.061906

220 1 1.061539 1.061739

221 1 1.061536 1.061736

222 1 1.061732 1.061932

223 3 1.061881 1.062081

224 1 1.061962 1.062162

225 1 1.061781 1.061981

226 1 1.061850 1.062050

227 1 1.061888 1.062088

228 3 1.061914 1.062114

229 1 1.062037 1.062237

230 3 1.062034 1.062234

231 1 1.062004 1.062204

232 1 1.061969 1.062169

233 3 1.061980 1.062180

234 3 1.062162 1.062362

235 3 1.062364 1.062564

236 3 1.062332 1.062532

237 2 1.062382 1.062582

238 3 1.062426 1.062626

.. ..

498 2 1.062645 1.062845

499 2 1.062838 1.063038

500 2 1.062687 1.062887

501 2 1.062620 1.062820

502 2 1.062535 1.062735

503 2 1.062546 1.062746

504 2 1.062308 1.062508

505 2 1.062207 1.062407

506 2 1.062178 1.062378

507 2 1.062105 1.062305

508 2 1.061980 1.062180

509 2 1.061983 1.062183

510 2 1.062099 1.062299

511 2 1.062142 1.062342

512 2 1.062143 1.062343

513 2 1.062131 1.062331

514 2 1.062094 1.062294

515 2 1.061870 1.062070

516 2 1.061654 1.061854

517 2 1.061832 1.062032

518 2 1.061707 1.061907

519 2 1.061939 1.062139

520 2 1.062122 1.062322

521 2 1.061962 1.062162

522 2 1.061738 1.061938

523 2 1.061710 1.061910

524 2 1.061720 1.061920

525 2 1.061733 1.061933

526 2 1.061601 1.061801

527 2 1.061553 1.061753

[319 rows x 3 columns]

minimum distance: [26.23014480000004]

==========

Predicted Label: [2]

True Label 2

finished!

==========

finish win creation...please wait...

test file 5 compare starting...

minimum window: grand s1 s2

7 3 1.062287 1.062783

8 2 1.062670 1.063001

9 2 1.062611 1.062818

10 1 1.062290 1.062633

11 1 1.062072 1.062295

12 1 1.061886 1.062149

13 1 1.061622 1.061831

14 1 1.061535 1.061735

15 1 1.061499 1.061699

16 1 1.061649 1.061849

17 1 1.061606 1.061806

18 1 1.061591 1.061791

19	1	1.061561	1.061761
20	1	1.061540	1.061740
21	1	1.061726	1.061926
22	1	1.061905	1.062105
23	1	1.062180	1.062380
24	1	1.062099	1.062303
25	1	1.062115	1.062323
26	1	1.062293	1.062493
27	1	1.062396	1.062596
28	3	1.062399	1.062599
29	3	1.063022	1.063222
30	3	1.062849	1.063049
31	3	1.062824	1.063024
32	3	1.062881	1.063081
33	3	1.062928	1.063128
34	3	1.062956	1.063156
35	3	1.062967	1.063167
36	3	1.062955	1.063155
..	...	...	...
296	2	1.061817	1.062017
297	2	1.061717	1.061917
298	2	1.061491	1.061691
299	3	1.061263	1.061463
300	3	1.061055	1.061255
301	3	1.060957	1.061157
302	3	1.060892	1.061092
303	3	1.060900	1.061100

304 3 1.060842 1.061042

305 1 1.060841 1.061041

306 3 1.060767 1.060967

307 1 1.060823 1.061023

308 1 1.060726 1.060926

309 1 1.060671 1.060871

310 1 1.060758 1.060958

311 3 1.060813 1.061013

312 3 1.060893 1.061093

313 3 1.060890 1.061090

314 1 1.060772 1.060972

315 3 1.060816 1.061016

316 3 1.060862 1.061062

317 3 1.060811 1.061011

318 1 1.060749 1.060949

319 3 1.060876 1.061076

320 3 1.060952 1.061152

321 3 1.060910 1.061110

322 3 1.060893 1.061093

323 3 1.060849 1.061049

324 3 1.060840 1.061040

325 3 1.060835 1.061035

[319 rows x 3 columns]

minimum distance: [31.053036200000015]

==========

Predicted Label: [3]

True Label 2

finished!

==========

finish win creation...please wait...

test file 6 compare starting...

minimum window: grand s1 s2

98 1 1.062660 1.062860

99 1 1.062525 1.062725

100 1 1.062508 1.062708

101 1 1.062629 1.062829

102 3 1.062732 1.062932

103 3 1.062942 1.063142

104 3 1.062964 1.063164

105 3 1.062924 1.063124

106 3 1.062958 1.063158

107 1 1.062806 1.063006

108 1 1.062718 1.062918

109 3 1.062943 1.063143

110 3 1.062948 1.063148

111 2 1.063166 1.063366

112 3 1.063093 1.063293

113 1 1.062914 1.063114

114 3 1.062810 1.063010

115 1 1.062894 1.063094

116 1 1.062826 1.063026

117 3 1.062947 1.063147

118	3	1.062960	1.063160
119	3	1.062893	1.063093
120	3	1.062853	1.063053
121	3	1.062958	1.063158
122	3	1.062923	1.063123
123	3	1.062935	1.063135
124	1	1.062818	1.063018
125	1	1.062911	1.063111
126	3	1.062984	1.063184
127	2	1.063179	1.063379
..	...	...	...
387	1	1.061720	1.061920
388	1	1.061721	1.061921
389	1	1.061715	1.061915
390	1	1.061612	1.061812
391	1	1.061621	1.061821
392	1	1.061667	1.061867
393	1	1.061740	1.061940
394	1	1.061931	1.062131
395	1	1.061836	1.062036
396	1	1.061793	1.061993
397	1	1.061756	1.061956
398	1	1.061778	1.061978
399	1	1.061742	1.061942
400	1	1.061717	1.061917
401	1	1.061879	1.062079
402	1	1.061818	1.062018

403 1 1.061733 1.061933

404 1 1.061735 1.061935

405 1 1.061721 1.061921

406 1 1.061864 1.062064

407 3 1.061966 1.062166

408 1 1.061840 1.062040

409 1 1.061799 1.061999

410 1 1.061820 1.062020

411 1 1.061820 1.062020

412 1 1.061773 1.061973

413 1 1.061865 1.062065

414 1 1.061803 1.062003

415 1 1.061733 1.061933

416 1 1.061772 1.061972

[319 rows x 3 columns]

minimum distance: [10.010794399999991]

==========

Predicted Label: [1]

True Label 1

finished!

==========

finish win creation...please wait...

test file 7 compare starting...

minimum window: grand s1 s2

132 3 1.063049 1.063249

133 2 1.063210 1.063410

134 2 1.063196 1.063396

135 2 1.063031 1.063231

136 2 1.062864 1.063064

137 2 1.062880 1.063080

138 2 1.062890 1.063090

139 2 1.062817 1.063017

140 2 1.062790 1.062990

141 2 1.062725 1.062925

142 2 1.062572 1.062772

143 2 1.062586 1.062786

144 2 1.062690 1.062890

145 2 1.062727 1.062927

146 2 1.062913 1.063113

147 2 1.062969 1.063169

148 2 1.062913 1.063113

149 2 1.062795 1.062995

150 2 1.062706 1.062906

151 2 1.062677 1.062877

152 2 1.062760 1.062960

153 2 1.062730 1.062930

154 2 1.062637 1.062837

155 2 1.062604 1.062804

156 2 1.062694 1.062894

157 2 1.062622 1.062822

158 2 1.062647 1.062847

159 2 1.062571 1.062771

160	2	1.062434	1.062634
161	2	1.062507	1.062707
..	...	...	...
421	1	1.061724	1.061924
422	3	1.061828	1.062028
423	1	1.061858	1.062058
424	1	1.061761	1.061961
425	1	1.061817	1.062017
426	1	1.061838	1.062038
427	1	1.061765	1.061965
428	1	1.061765	1.061965
429	1	1.061808	1.062008
430	1	1.061808	1.062008
431	1	1.061830	1.062030
432	1	1.061772	1.061972
433	1	1.061751	1.061951
434	3	1.061952	1.062152
435	3	1.062131	1.062331
436	3	1.062192	1.062392
437	3	1.062127	1.062327
438	1	1.061987	1.062187
439	1	1.061837	1.062037
440	1	1.061901	1.062101
441	3	1.062011	1.062211
442	1	1.062009	1.062209
443	3	1.061982	1.062182
444	3	1.062104	1.062304

445 1 1.062016 1.062216

446 3 1.062124 1.062324

447 3 1.062347 1.062547

448 3 1.062236 1.062436

449 3 1.062255 1.062455

450 3 1.062118 1.062318

[319 rows x 3 columns]

minimum distance: [36.62529300000003]

==========

Predicted Label: [3]

True Label 1

finished!

==========

finish win creation...please wait...

test file 8 compare starting...

minimum window: grand s1 s2

238 3 1.062426 1.062626

239 3 1.062328 1.062528

240 2 1.062520 1.062720

241 2 1.062615 1.062815

242 2 1.062471 1.062671

243 2 1.062384 1.062584

244 2 1.062453 1.062653

245 2 1.062379 1.062579

246 2 1.062324 1.062524

247 2 1.062282 1.062482

248 2 1.062226 1.062426

249 2 1.062163 1.062363

250 2 1.062322 1.062522

251 2 1.062249 1.062449

252 2 1.062111 1.062311

253 2 1.062097 1.062297

254 2 1.062255 1.062455

255 2 1.062340 1.062540

256 2 1.062364 1.062564

257 2 1.062330 1.062530

258 2 1.062303 1.062503

259 2 1.062278 1.062478

260 2 1.062200 1.062400

261 2 1.062211 1.062411

262 2 1.062305 1.062505

263 2 1.062276 1.062476

264 2 1.062115 1.062315

265 2 1.062160 1.062360

266 2 1.062143 1.062343

267 2 1.062093 1.062293

.. ..

527 2 1.061553 1.061753

528 3 1.061386 1.061586

529 3 1.061317 1.061517

530 3 1.061228 1.061428

531 2 1.061332 1.061532

532	3	1.061375	1.061575
533	3	1.061204	1.061404
534	3	1.061082	1.061282
535	3	1.061042	1.061242
536	3	1.060930	1.061130
537	3	1.060808	1.061008
538	3	1.060894	1.061094
539	3	1.061001	1.061201
540	3	1.061019	1.061219
541	3	1.060899	1.061099
542	3	1.060923	1.061123
543	3	1.060974	1.061174
544	3	1.061174	1.061374
545	3	1.061373	1.061573
546	3	1.061192	1.061392
547	3	1.061105	1.061305
548	3	1.060927	1.061127
549	3	1.060822	1.061022
550	3	1.060930	1.061130
551	3	1.060960	1.061160
552	3	1.061223	1.061423
553	3	1.061254	1.061454
554	3	1.061267	1.061467
555	3	1.061202	1.061402
556	3	1.060912	1.061112

[319 rows x 3 columns]

minimum distance: [74.83209519999997]

==========

Predicted Label: [3]

True Label 3

finished!

==========

finish win creation...please wait...

test file 9 compare starting...

minimum window: grand s1 s2

113 1 1.062914 1.063114

114 3 1.062810 1.063010

115 1 1.062894 1.063094

116 1 1.062826 1.063026

117 3 1.062947 1.063147

118 3 1.062960 1.063160

119 3 1.062893 1.063093

120 3 1.062853 1.063053

121 3 1.062958 1.063158

122 3 1.062923 1.063123

123 3 1.062935 1.063135

124 1 1.062818 1.063018

125 1 1.062911 1.063111

126 3 1.062984 1.063184

127 2 1.063179 1.063379

128 3 1.063141 1.063341

129 2 1.063129 1.063329

130 2 1.063267 1.063467

131 1 1.063096 1.063296

132 3 1.063049 1.063249

133 2 1.063210 1.063410

134 2 1.063196 1.063396

135 2 1.063031 1.063231

136 2 1.062864 1.063064

137 2 1.062880 1.063080

138 2 1.062890 1.063090

139 2 1.062817 1.063017

140 2 1.062790 1.062990

141 2 1.062725 1.062925

142 2 1.062572 1.062772

.. ..

402 1 1.061818 1.062018

403 1 1.061733 1.061933

404 1 1.061735 1.061935

405 1 1.061721 1.061921

406 1 1.061864 1.062064

407 3 1.061966 1.062166

408 1 1.061840 1.062040

409 1 1.061799 1.061999

410 1 1.061820 1.062020

411 1 1.061820 1.062020

412 1 1.061773 1.061973

413 1 1.061865 1.062065

414 1 1.061803 1.062003

415 1 1.061733 1.061933

416 1 1.061772 1.061972

417 1 1.061787 1.061987

418 1 1.061760 1.061960

419 1 1.061763 1.061963

420 1 1.061777 1.061977

421 1 1.061724 1.061924

422 3 1.061828 1.062028

423 1 1.061858 1.062058

424 1 1.061761 1.061961

425 1 1.061817 1.062017

426 1 1.061838 1.062038

427 1 1.061765 1.061965

428 1 1.061765 1.061965

429 1 1.061808 1.062008

430 1 1.061808 1.062008

431 1 1.061830 1.062030

[319 rows x 3 columns]

minimum distance: [82.8652796000001]

==========

Predicted Label: [1]

True Label 1

finished!

==========

finish win creation...please wait...

test file 10 compare starting...

minimum window: grand s1 s2

113 1 1.062914 1.063114

114 3 1.062810 1.063010

115 1 1.062894 1.063094

116 1 1.062826 1.063026

117 3 1.062947 1.063147

118 3 1.062960 1.063160

119 3 1.062893 1.063093

120 3 1.062853 1.063053

121 3 1.062958 1.063158

122 3 1.062923 1.063123

123 3 1.062935 1.063135

124 1 1.062818 1.063018

125 1 1.062911 1.063111

126 3 1.062984 1.063184

127 2 1.063179 1.063379

128 3 1.063141 1.063341

129 2 1.063129 1.063329

130 2 1.063267 1.063467

131 1 1.063096 1.063296

132 3 1.063049 1.063249

133 2 1.063210 1.063410

134 2 1.063196 1.063396

135 2 1.063031 1.063231

136 2 1.062864 1.063064

137 2 1.062880 1.063080

138	2	1.062890	1.063090
139	2	1.062817	1.063017
140	2	1.062790	1.062990
141	2	1.062725	1.062925
142	2	1.062572	1.062772
..	...	...	...
402	1	1.061818	1.062018
403	1	1.061733	1.061933
404	1	1.061735	1.061935
405	1	1.061721	1.061921
406	1	1.061864	1.062064
407	3	1.061966	1.062166
408	1	1.061840	1.062040
409	1	1.061799	1.061999
410	1	1.061820	1.062020
411	1	1.061820	1.062020
412	1	1.061773	1.061973
413	1	1.061865	1.062065
414	1	1.061803	1.062003
415	1	1.061733	1.061933
416	1	1.061772	1.061972
417	1	1.061787	1.061987
418	1	1.061760	1.061960
419	1	1.061763	1.061963
420	1	1.061777	1.061977
421	1	1.061724	1.061924
422	3	1.061828	1.062028

```
423     1  1.061858  1.062058
424     1  1.061761  1.061961
425     1  1.061817  1.062017
426     1  1.061838  1.062038
427     1  1.061765  1.061965
428     1  1.061765  1.061965
429     1  1.061808  1.062008
430     1  1.061808  1.062008
431     1  1.061830  1.062030

[319 rows x 3 columns]
minimum distance: [82.8652796000001]
==================================
Predicted Label: [1]
True Label 1
finished!
======== Report Project Result =====================
              precision    recall  f1-score   support

           1       1.00      0.67      0.80         6
           2       1.00      0.50      0.67         2
           3       0.40      1.00      0.57         2
   micro avg       0.70      0.70      0.70        10
   macro avg       0.80      0.72      0.68        10
weighted avg       0.88      0.70      0.73        10
ACCURACY is:  0.7
```

Process finished with exit code 0

1090.9236645698547