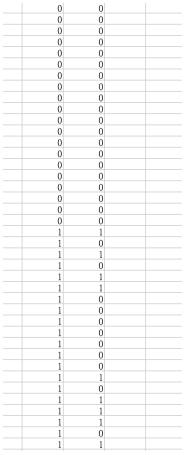
B10601002 廖品捷 github: https://github.com/JamesLiao714/IOT_data_analysis

Part C:

(a)上次用 sliding window 搭配 knn 進行分析（window_size = 3），預測出來最好的情況大概在 73%左右，如果訓練資料雜訊太多會造成 test accuracy 更低，重點是要跑很久，mac 感覺要爆炸了，這次直接使用 classifier(extreme gradient boosting)直接對資料進行分，結果成功率進步不少（code 放在 github (a)資料夾裡）。我是將所有資料(train)存起來餵給 classifier 建一個 model，然後再把新的 motion sensor data 丟給 model predict，用.score 查看正確率後 print 出一個 pred_outptut 檔來對照預測結果與實際 motion（已經過 labelencoder）

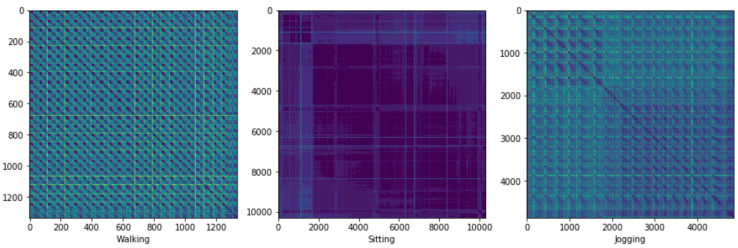
```
Using TensorFlow backend.
[[-0.29339302  0.30572653 -0.46498013]
 [-0.8981955   0.3636732   0.5878544 ]
 [-0.00661314  0.07286644 -0.74147034]
 ...
 [ 0.05728316 -0.16935015 -0.0191412 ]
 [ 0.05728316 -0.16935015 -0.0191412 ]
 [ 0.05728316 -0.16935015 -0.0191412 ]]
['W' 'W' 'W' ... 'X' 'X' 'X']
training score: 0.7906678243799313
test score: 0.8366510780946075
[Finished in 6.1s]
```

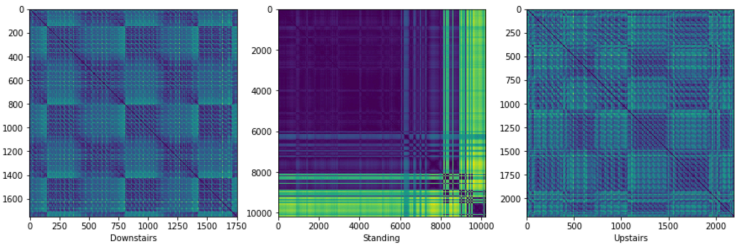


(b)(c)一開始我用 sub1 的所有 motion 的資料建了一個 model(用的是 xgboost 的 classifier, 將其切成 test 0.1 跟 train 0.9，因為資料乾淨結果相當高，都在 90%上下)，將 model 存在 xgb_model.sav 裡接著再去對其他使用者進行 cross-people study 但結果不盡理想，在分析 sub1 的動態時準確率相當高，但其他人的準確率大概只在 30-40%左右，經過觀察後，發現應該是和測試者的身高體種和走路習慣有關。

```
Using TensorFlow backend.
motion prediction using p1's model:
participant1-> test result0.9582905379381178
participant2-> test result0.32052166380596414
participant3-> test result0.4027304845491024
participant4-> test result0.43120607855028004
participant5-> test result0.3518734579117495
participant6-> test result0.3934552051353614
participant7-> test result0.3740126640120112
participant8-> test result0.32171851071187446
participant9-> test result0.45171170540464856
participant10-> test result0.1455448524984949
participant11-> test result0.370184847871551
participant12-> test result0.3008650291783161
participant13-> test result0.4109722955403054
participant14-> test result0.2531555933223574
participant15-> test result0.3089850781420857
participant16-> test result0.3101181650029809
participant17-> test result0.33493581087163976
participant18-> test result0.2919214112851866
participant19-> test result0.08266966879317295
participant20-> test result0.23222671477526155
participant21-> test result0.2904746706899553
participant22-> test result0.30124223602484473
participant23-> test result0.45555026028301193
participant24-> test result0.23025193608469757
[Finished in 54.5s]
```

為了提高 cross_people 的準確性，我參考了 kaggle kermel 的做法。實作方式是透過 cnn 的方式將讀入資料轉成圖片餵給 model，透過[3*3]sliding window 卷積取得特徵值和 pooling 等步驟後進行預測，結果正確率相當高。雖然這個做法跟我的作法很像，只是 train 的 data 是全部測試者的資料，但也讓我發現原來 cnn 不只能處理圖形，也可以拿來處理連續時間資料！





將 data 轉為圖片(sub1)
參考程式中的
recurrent_plot

```
model = Sequential()

model.add(Convolution2D(32, (3, 3), activation='relu', input_shape=(1,32,32), data_format='channels_first'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Convolution2D(32, (3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
#model.add(LeakyReLU(alpha=0.03))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

CNN model

reference:

https://www.kaggle.com/tigurius/recuplots-and-cnns-for-time-series-classification