**HW1\_2019\_part-B**

(b)

由於我在搜集資料時，一直發生中斷，不然就是在拿到資料在查看時，才發現測量的sensor的數值，太不合理，所以我只拿其中算是比較合理可用的data來使用。設定sensor為每一秒紀錄一次。

2019-10-26\_22-35-35-Riding

|  |  |
| --- | --- |
| Accelerometer X | Accelerometer Y |
|  |  |
| Accelerometer Z | Compass X |
|  |  |
| Compass Y | Compass Z |
|  |  |
| GPS Latitude | GPS Longitude |
|  |  |
| GPS Altitude |  |
|  |  |

2019-10-27\_18-41-01-Riding

|  |  |
| --- | --- |
| Accelerometer X | Accelerometer Y |
|  |  |
| Accelerometer Z | Compass X |
|  |  |
| Compass Y | Compass Z |
|  |  |
| GPS Latitude | GPS Longitude |
|  |  |
| GPS Altitude |  |
|  |  |

2019-10-27\_16-06-15-Walk

|  |  |
| --- | --- |
| Accelerometer X | Accelerometer Y |
|  |  |
| Accelerometer Z | Compass X |
|  |  |
| Compass Y | Compass Z |
|  |  |
| GPS Latitude | GPS Longitude |
|  |  |
| GPS Altitude |  |
|  |  |

2019-10-27\_21-42-51-Study

|  |  |
| --- | --- |
| Accelerometer X | Accelerometer Y |
|  |  |
| Accelerometer Z | Compass X |
|  |  |
| Compass Y | Compass Z |
|  |  |
| GPS Latitude | GPS Longitude |
|  |  |
| GPS Altitude |  |
|  |  |

2019-10-27\_20-37-27-Study

|  |  |
| --- | --- |
| Accelerometer X | Accelerometer Y |
|  |  |
| Accelerometer Z | Compass X |
|  |  |
| Compass Y | Compass Z |
|  |  |
| GPS Latitude | GPS Longitude |
|  |  |
| GPS Altitude |  |
|  |  |

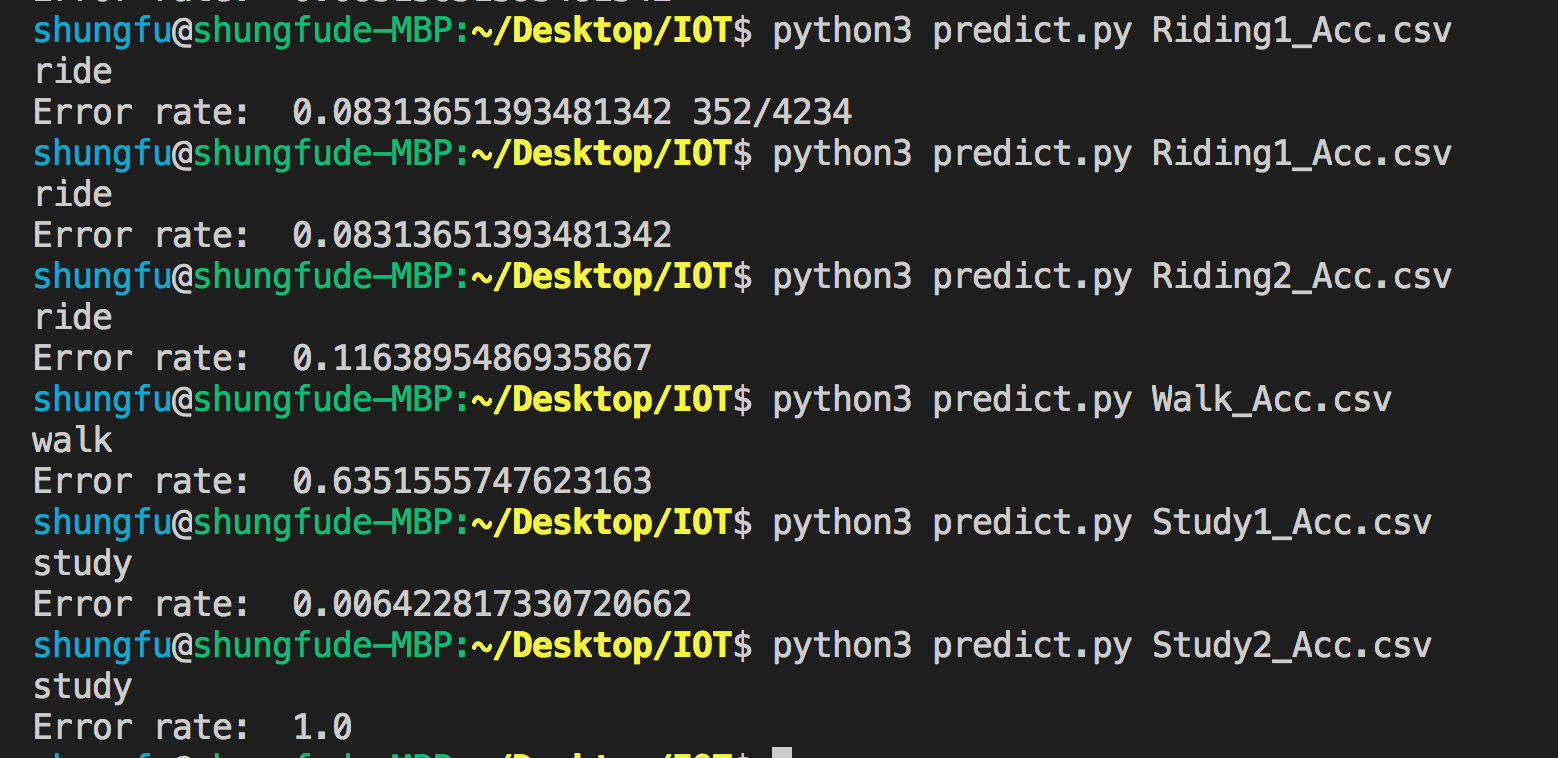
(c)

這邊我只使用我認為較有判斷價值的加速度。我先將Ride, Study和walk的csv各自建模型。建立模型方法是使用sliding window計算其標準差，window size為50，然後再取所有標準差的平均值。由於算完後的值有加速度的Ｘ,Y,Z三種，而Ｙ軸的加速度是，Ride,Study和walk中差異最大的，所以我採用Ｙ作為這個模型的判定基準數值。

之後讀入要做判斷的csv檔，我們用一樣的方法，算輸入檔案的標準差平均值，然後再利用我們先前模型建立出的判定基準，來判斷說兩者相差最少的，就為該類別。

結果：

用來測試的資料中，有未參與建模的dataset ,Riding2和Studing2。測試後，結果都為正確，並顯示內部運算的error rate。



雖然說結果和每一欄的判斷結果不同的error rate 高達1.0，但是由於我是用sliding window計算標準差，之後運算平均值。再拿來判斷，所以我依然能夠預測正確。

Python code:

<https://hackmd.io/@L5UnFuBVQ12dUMp2E__zOw/Hy-gNCu_r>

(f)

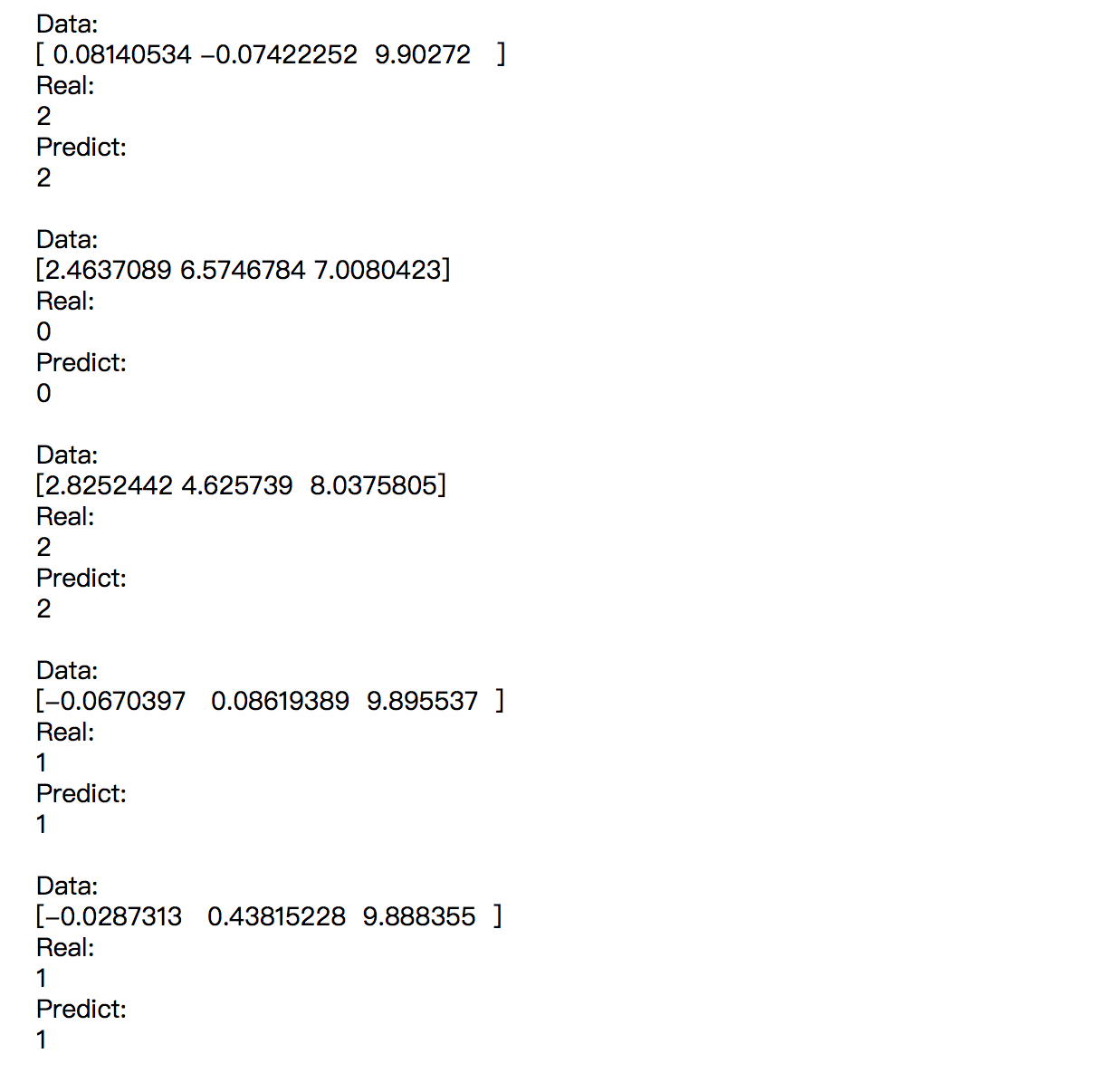
由於這邊需要將有標上label的檔案分群，所以我使用knn演算法來實作。knn屬於監督式的學習，是一種classification。再開始訓練模型之前，我們需要把原本分散的csv結合成一個，並為他們標上label。（這邊我標記0: riding motorcycle, 1: walking, 2: studying）

我們把上面結合好的資料做打亂，這樣我們訓練的結果才能更好。再把標好的label和原本的資料及切開，分為data和target。這樣在訓練的時候，我們的判斷結果才不會被視為一種attribute。接著再將data和target依照你的喜好比例，把他們都切成test和train。

做好上述切割後，我們就可以匯入模型了。這邊我直接使用sklearn寫好的套件。from sklearn.neighbors import KNeighborsClassifier。把test data和test target丟入fit function裡面，模型就做好了。如果你想要看看正確率如何的話，你可以使用score function，這邊的參數就要放你剛剛切割的train data和train target丟入，如果你還是丟test的話就會失去參考性了，因為你的模型和測試資料太過吻合了。



（準確率）



(測試判斷)

Python code:

<https://hackmd.io/@L5UnFuBVQ12dUMp2E__zOw/S1d-YpK9r>