Homework 1 – part C

環境建置：

使用 jupyter notebook 和 Visual Studio Code。

組員：

B10615056 黃暉翔　B10615045 陳尚富　B10615046 柯元豪

(A) .

這邊使用 hw1-partB 收集到的資料，來對其使用 deep learning method（multiple hidden layer）。Model summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                128
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                528
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 51
=================================================================
Total params: 707
Trainable params: 707
Non-trainable params: 0
```

Input layer: Activation function 使用 relu，input_dim 為 3 因為只有三個特徵。

Hidden layer: 使用了兩層的 dropout 和一層 dense layer 使用 relu 作為 activation function。

Output layer: dense layer，因為是分類所以選用 softmax 作為輸出層的 activation function。

方法流程：

先將收集到分散的各種類 csv 結合成一個 csv file。但是由於 label 為字串，若是直接轉為數字對於 Ann 或是 deep learning 而言，會更像是二分類法或是 regression，所以要先將 label 轉為 category（類似 one hot encoding）。

	X	Y	Z	label
0	0.256187	2.966506	9.409500	Walk
1	-2.669617	4.477294	9.488511	Walk
2	1.357554	-0.193936	9.093455	Walk
3	0.095771	0.088588	9.893144	Walk
4	0.114925	0.088588	9.890749	Walk
...	...	...	...	...
28540	-0.052674	0.098165	9.921875	Ride
28541	-0.011971	0.088588	9.921875	Ride
28542	-0.052674	0.098165	9.941029	Ride
28543	-0.035914	0.105348	9.919480	Ride
28544	-0.033520	0.076617	9.902720	Ride

→

	0	1	2
13634	1.0	0.0	0.0
28041	1.0	0.0	0.0
4883	0.0	0.0	1.0
2480	0.0	0.0	1.0
16605	0.0	1.0	0.0
...	...	...	...
12333	0.0	1.0	0.0
12890	1.0	0.0	0.0
17503	0.0	1.0	0.0
23337	0.0	1.0	0.0
25493	1.0	0.0	0.0

再來應用 sliding window 將所有的 data 取平均值，window size 這邊定義為 100。然後再將一些 nan 的資料砍掉，前處理就完成了。再來只需要套用上面介紹的 deep learning model 就完成了。

(B) .

由於是將(A)的實作方式，套用在 kaggle 上的資料，因此只在底下列出不同處。

- Attribute 不只有加速度的 X,Y,Z，而 act 也不只三種，一樣不需要的 drop 掉

```
data = pd.read_csv('0.csv')
data = data.drop(['index','id','weight','height','age','gender'],axis = 1)
data
```

	attitude.roll	attitude.pitch	attitude.yaw	userAcceleration.x	userAcceleration.y	userAcceleration.z	act

- 在 ANN 的部分 dim 也要做對應的更改，因為特徵變成六種

```
def ANN_model():
    # Using Relu activation, and Adam optimizer, 100 epochs
    model = Sequential()

    model.add(Dense(32,input_dim=6,activation='relu',kernel_initializer='uniform'))

    model.add(Dropout(0.2))
    model.add(Dense(16,activation='relu'))
    model.add(Dropout(0.2))

    model.add(Dense(6,activation='softmax'))
```

其餘步驟與實作方法皆與(A)相同

(C)．

先將subject.csv 和 device motion data 結合做出time series的資料，依照每個人的資料做出24個csv檔，讀取前20個當training data，用rolling函數做sliding windows後使用ann model訓練。

讀取後面4個一樣做sliding windows 當testing data 後驗證模型準確度。

```
Epoch 46/50
1176364/1176364 [==============================] - 2s 2us/step - loss: 0.7627 - accuracy: 0.7084
Epoch 47/50
1176364/1176364 [==============================] - 2s 2us/step - loss: 0.7603 - accuracy: 0.7096
Epoch 48/50
1176364/1176364 [==============================] - 2s 2us/step - loss: 0.7593 - accuracy: 0.7098
Epoch 49/50
1176364/1176364 [==============================] - 2s 2us/step - loss: 0.7564 - accuracy: 0.7112
Epoch 50/50
1176364/1176364 [==============================] - 2s 2us/step - loss: 0.7544 - accuracy: 0.7121
229325/229325 [==============================] - 0s 1us/step
training 準確率: 0.640872124713834
```