Activity #10

LSTM Classification

Agenda

10.1 Data Preparation

10.2 LSTM Model Training and Testing

10.3 LSTM Performance Measurement

LIBRARIES

1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. from sklearn.preprocessing import StandardScaler
5. from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
6. from sklearn.svm import SVC
7. from sklearn import metrics
8. from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
9. from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
10. import glob
11. from scipy import stats
12. import datetime as dt

10.1 Data Preparation

- **Data Exploration and Cleaning / Transform / Feature Selection /Train-Test-Split**

10.1.1 Data Preparation (9.1.1)

1

- # ------------ Prepare same as 9.1.1 (8.1 (a) – (e))------------------------------
  - # acceleration.txt, heartrate.txt, labeled_sleep.txt
  - # Rounding ACC (Rounding to 1 sec)
  - # ACC Average rounding duplicated time

  - # Rounding Heart Rate (Rounding to 1 sec)
  - # Resampling every 1s with median with ffill

  - # Rounding Sleep Label (Rounding to 1 sec)
  - # Resampling every 1s with median with ffill

  - # After all above steps, we get
    - # df_feature
    - # df_label

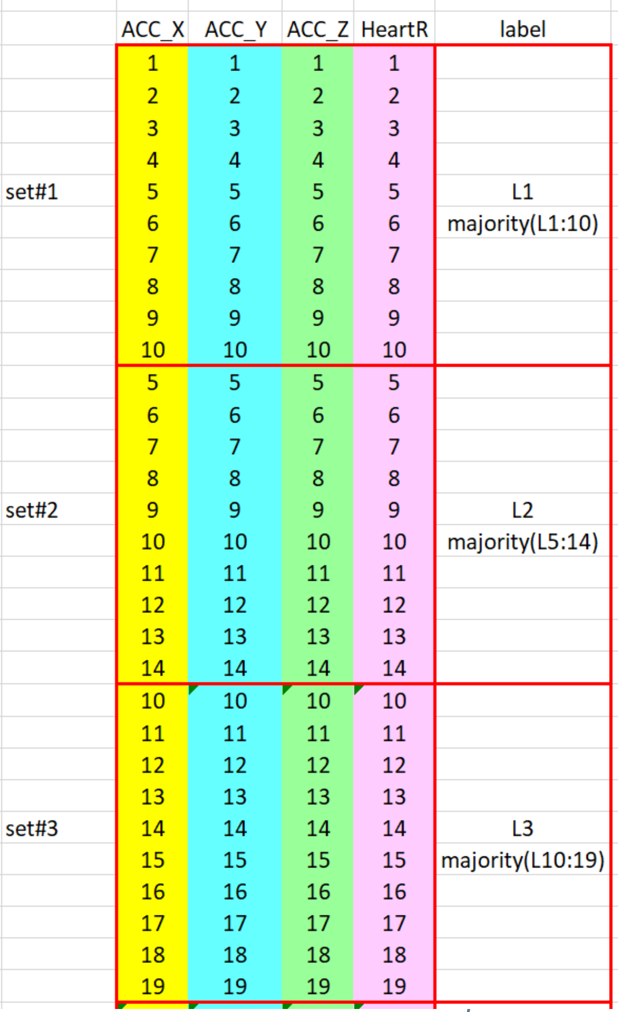
10.1.2 Simple Moving Average (SMA) and Create 2D feature

2

- # ------------Simple Moving Average (SMA) -----------------------------
- #columns=['accX', 'accY', 'accZ', 'heartrate']
- df_feature_SMA['accX'] = df_feature[0].rolling(5, min_periods=1).mean()
- df_feature_SMA['accY'] = df_feature[1].rolling(5, min_periods=1).mean()
- df_feature_SMA['accZ'] = df_feature[2].rolling(5, min_periods=1).mean()
- df_feature_SMA['heartrate'] = df_feature[3].rolling(5, min_periods=1).mean()

3

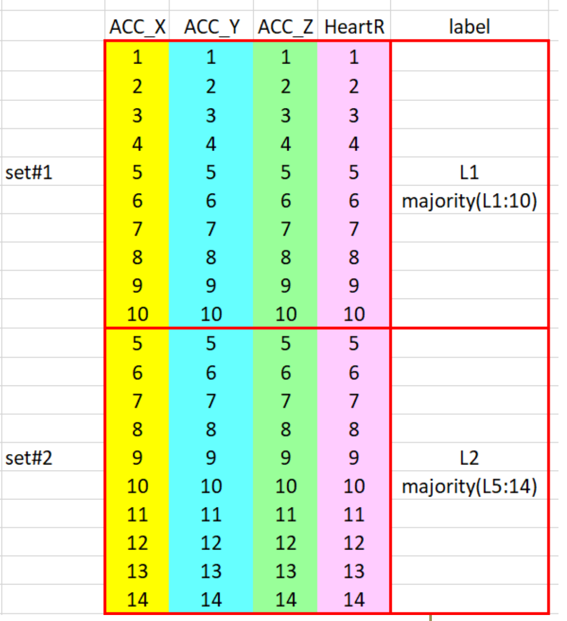
- # ------------ Train-Test-Split 2D features ------------------------------
- # set sliding window parameter
- slidingW = 100 #จำนวน row
- Stride_step = 5
- For t in range(0 , len(df_feature), stride_step)
  - F2d= df_feature(t : t + slidingW))
  - df_feature2D.append(F2d)
  - F2d_T = np.transpose(F2d)
  - df_feature2D_T.append(F2d_T)
  - Labels = stats.mode(df_label (t : t+slidingW , ‘label’))
  - df_label_new.append(Labels)



	ACC_X	ACC_Y	ACC_Z	HeartR	label
	1	1	1	1	
	2	2	2	2	
	3	3	3	3	
	4	4	4	4	
set#1	5	5	5	5	L1
	6	6	6	6	majority(L1:10)
	7	7	7	7	
	8	8	8	8	
	9	9	9	9	
	10	10	10	10	
	5	5	5	5	
	6	6	6	6	
	7	7	7	7	
	8	8	8	8	
set#2	9	9	9	9	L2
	10	10	10	10	majority(L5:14)
	11	11	11	11	
	12	12	12	12	
	13	13	13	13	
	14	14	14	14	
	10	10	10	10	
	11	11	11	11	
	12	12	12	12	
	13	13	13	13	
set#3	14	14	14	14	L3
	15	15	15	15	majority(L10:19)
	16	16	16	16	
	17	17	17	17	
	18	18	18	18	
	19	19	19	19	

10.1.3 Train Test Split

Option#1: no transpose
input_shape=(timesteps, n_features)



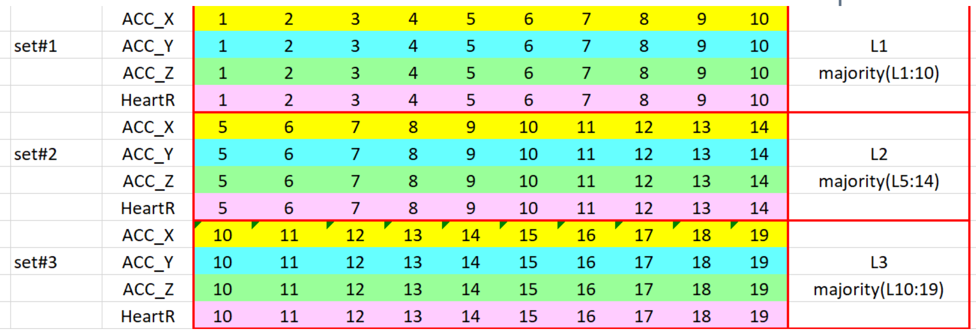
	ACC_X	ACC_Y	ACC_Z	HeartR	label
	1	1	1	1	
	2	2	2	2	
	3	3	3	3	
	4	4	4	4	
set#1	5	5	5	5	L1
	6	6	6	6	majority(L1:10)
	7	7	7	7	
	8	8	8	8	
	9	9	9	9	
	10	10	10	10	
	5	5	5	5	
	6	6	6	6	
	7	7	7	7	
	8	8	8	8	
set#2	9	9	9	9	L2
	10	10	10	10	majority(L5:14)
	11	11	11	11	
	12	12	12	12	
	13	13	13	13	
	14	14	14	14	

4

- # ------------ Train-Test-Split 2D features ---------------------------------
  - x_train, x_test, y_train, y_test = train_test_split(df_feature2D, df_label)

5

- # ------------ Train-Test-Split 2D features ---------------------------------
  - x_train, x_test, y_train, y_test = train_test_split(df_feature2D_T, df_label)

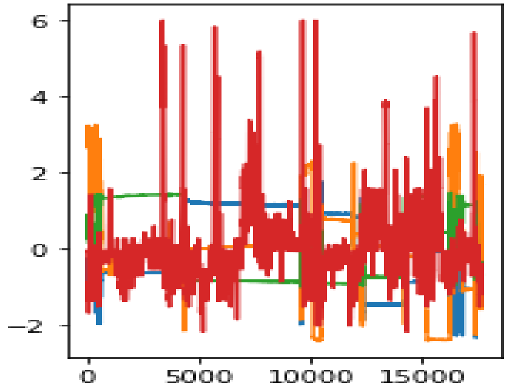


	ACC_X	1	2	3	4	5	6	7	8	9	10	
set#1	ACC_Y	1	2	3	4	5	6	7	8	9	10	L1
	ACC_Z	1	2	3	4	5	6	7	8	9	10	majority(L1:10)
	HeartR	1	2	3	4	5	6	7	8	9	10	
	ACC_X	5	6	7	8	9	10	11	12	13	14	
set#2	ACC_Y	5	6	7	8	9	10	11	12	13	14	L2
	ACC_Z	5	6	7	8	9	10	11	12	13	14	majority(L5:14)
	HeartR	5	6	7	8	9	10	11	12	13	14	
	ACC_X	10	11	12	13	14	15	16	17	18	19	
set#3	ACC_Y	10	11	12	13	14	15	16	17	18	19	L3
	ACC_Z	10	11	12	13	14	15	16	17	18	19	majority(L10:19)
	HeartR	10	11	12	13	14	15	16	17	18	19	

Option#2: with transpose
input_shape=(n_features, timesteps)

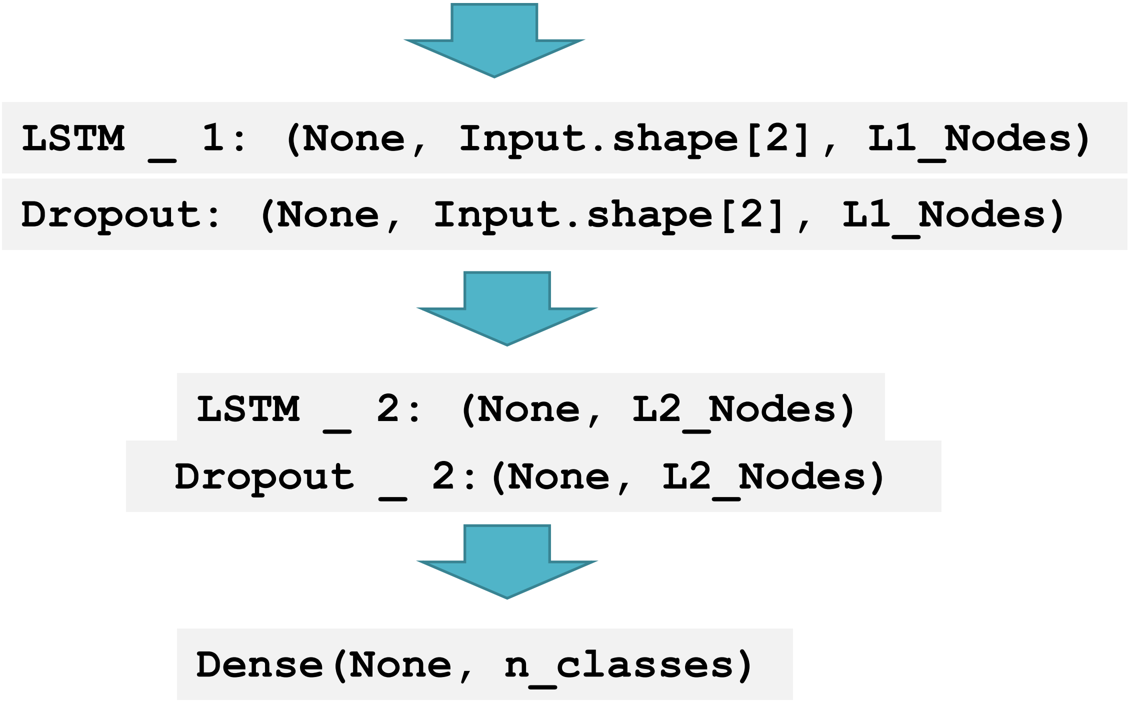
10.2 LSTM Model Training and Testing

LSTM Model Architecture



Option#1: no transpose
input_shape=(timesteps, n_features)

Option#2: with transpose
input_shape=(n_features, timesteps)



LSTM _ 1: (None, Input.shape[2], L1_Nodes)
Dropout: (None, Input.shape[2], L1_Nodes)

LSTM _ 2: (None, L2_Nodes)
Dropout _ 2:(None, L2_Nodes)

Dense(None, n_classes)

10.2.1 LSTM Model Parameters

1

- # ------------ LSTM Architecture parameter ------------------------------
- # Nlayer (LSTM, dense), Nnode, Activation
  - LSTM_L1 = 100 # try 200, 300, 400, 500, 1000
  - LSTM_L2 = 50 # try 50, 100, 150, 200, 250, 300
  - dropRate_L1 = 0.25
  - dropRate_L2 = 0.5
  - D_out = 5
  - Activation = “Softmax”
  - n_classes = 5
  - Input_shape = (inRow, inCol)
    - # try
      - #Option #1:
        - inRow = N_features
        - inCol = Sliding_windows
      - # Option #2
        - inRow = Sliding_windows
        - inCol = Sliding_windows

Option#1: no transpose
input_shape=(timesteps, n_features)

Option#2: with transpose
input_shape=(n_features, timesteps)

10.2.2 LSTM Model Train Test

2

- **# ------------ Create LSTM Model** -------------------------------
  - model = Sequential()
  - model.add(LSTM (LSTM_L1, return_sequences=True,
    - input_shape=Input_shape))
  - model.add(Dropout(**dropRate_L1**))
  - model.add(LSTM(LSTM_L2))
- model.add(Dropout(**dropRate_L12**))
  - model.add(Dense(n_classes, activation='softmax'))
  - model.summary()

10.2.3 LSTM Model Train Test

3

- # ------------ Create Optimizer ------------------------------
  - model.compile(optimizer='adam',
  - loss='categorical_crossentropy',
  - metrics=["acc"])

4

Option#1: no transpose
input_shape=(timesteps, n_features)

Option#2: with transpose
input_shape=(n_features, timesteps)

- # ------ Train CNN using 2D feature--
  - # Training the model
  - EP = 50
  - batch_size = 60 # try 20, 40, 60, 80, 100

- history = model.fit(X_train, y_train, # try Option #1 และ Option #2
  batch_size = batch_size,
  validation_data=(X_test, y_test), epochs=EP)

10.3 LSTM Performance Measurement

10.3 Performnace of LSTM Model

1

- #LSTM prediction for Option #1 and Option #2
  - LSTM_pred = model.predict(X_test)
  - Get classID from max prob(LSTM_pred)
  - df_pred = pd.DataFrame(LSTM_pred)
  - df_class => use dataframe -> idxmax(axis=1)

Option#1: no transpose
input_shape=(timesteps, n_features)

Option#2: with transpose
input_shape=(n_features, timesteps)

2

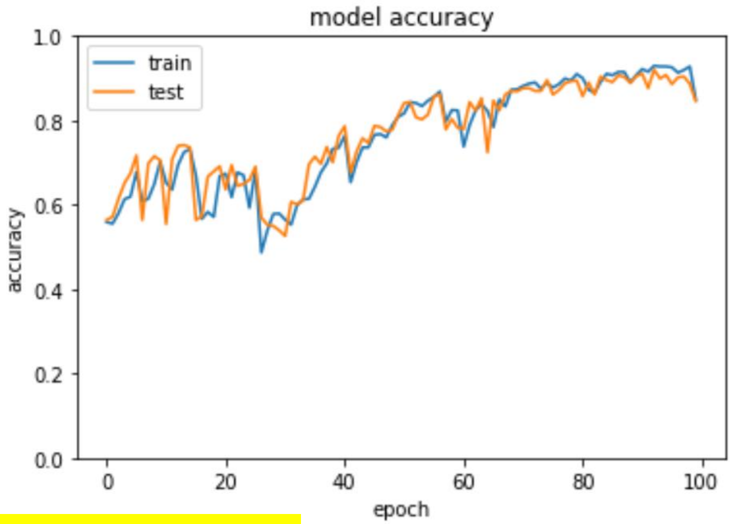
- # ------------ View Confusion Matrix, Classification Report ------------------------------

3

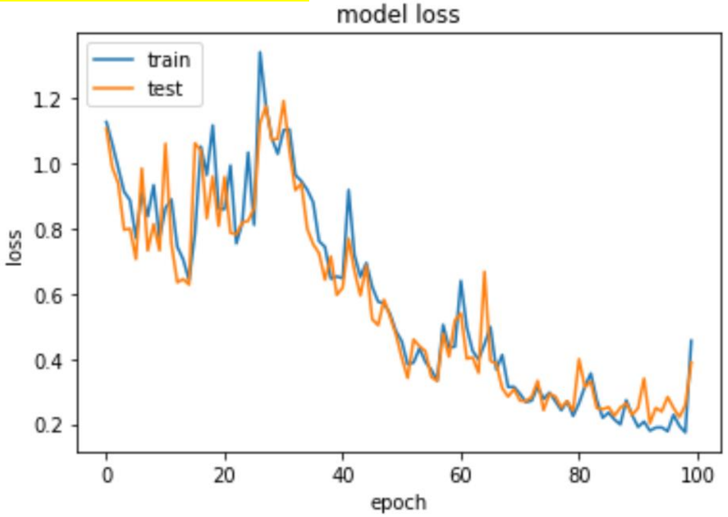
- # ------ View History Graph --
  - # View Accuracy Graph, Loss Graph
    - plt.plot(history.history['acc'])
    - plt.plot(history.history['val_acc'])

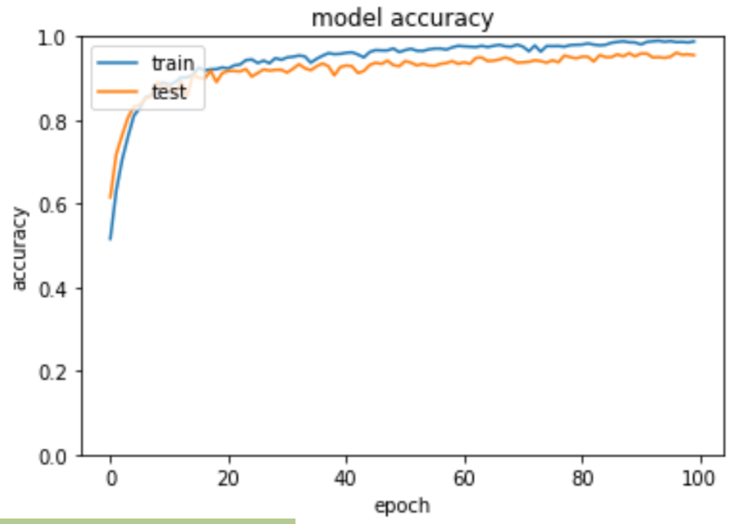
    - plt.plot(history.history['loss'])
    - plt.plot(history.history['val_loss'])

History Graph (Accuracy, Loss)

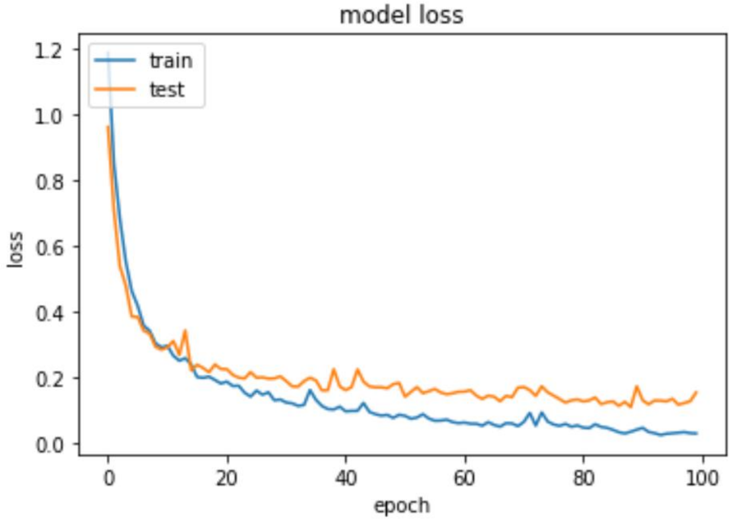


Option#1: no transpose
input_shape=(timesteps, n_features)





Option#2: with transpose
input_shape=(n_features, timesteps)


model loss
loss
epoch
train
test