**เอกสารอธิบายการทดลองที่ 3 ตอนที่ 1**

**การทดลองเตรียมข้อมูล ปรับค่าข้อมูล และจัดแบ่งชุด Train, Test เพื่อสอนโมเดล**

**ตอนที่ 1:** การทดลองเตรียมข้อมูล ปรับค่าข้อมูล และจัดแบ่งชุด Train, Test เพื่อสอนโมเดล

* 1. Import Lib (numpy, pandas, matplotlib, sklearn, keras)
* Import Library : นำเข้า library ที่สำคญต่าง ๆ เช่น numpy, pandas, matplotlib.pyplot, model layer หรือ optimizer ของ scikit-learn, seaborn, talos สำหรับค้นหา parameters ที่ดีที่สุด เป็นต้น

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from collections import Counter

from sklearn.preprocessing import OneHotEncoder

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D,   
MaxPooling2D, BatchNormalization

from tensorflow.keras.optimizers import SGD, Adam

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from tensorflow.keras.layers import LSTM

import talos as ta

plt.style.use('ggplot')

* 1. โหลดข้อมูล Timeseries Dataset file

- ทำการโหลดข้อมูลชุดที่ 1 โดยเลือกชุด 46343, 781756 และ 1066528 ซึ่ง feature ที่ใช้ประกอบด้วย acceleration, heartrate และ labeled\_sleep

num\_name = ['46343', '781756', '1066528', '844359']

fea\_name = [ 'acceleration', 'heartrate', 'labeled\_sleep']

df\_list = list()

merge\_fea\_col = pd.DataFrame()

for n in num\_name :

  fea\_col = list()

  for idx\_fea,f in enumerate(fea\_name, 0):

    fea\_df = pd.read\_csv(root\_dir + '/Sleep\_MiNiSet/' + n + '\_' + f + '.txt', sep=" ", header=None)

    if len(fea\_df.columns) < 2 :

      fea\_df = pd.read\_csv(root\_dir + '/Sleep\_MiNiSet/' + n + '\_' + f + '.txt', sep=",", header=None)

    fea\_df['uts\_' + f] = pd.to\_datetime(fea\_df[0], unit='s', origin=pd.Timestamp('2019-10-04'))

    if len(fea\_df.columns) <= 3 :

      fea\_df.columns = ['time\_sec\_'+f , f, 'uts\_' + f]

    else :

      fea\_df.columns = ['time\_sec\_'+f , f+'\_x', f+'\_y', f+'\_z', 'uts\_' + f]

    fea\_col.append(fea\_df)

  limit\_sec = fea\_col[len(fea\_name) - 1].iloc[-1:, 0].values[0]

  for num\_df in range(len(fea\_name)) :

    fea\_col[num\_df].set\_index('uts\_' + fea\_name[num\_df], inplace=True)

    fea\_col[num\_df] = fea\_col[num\_df][(fea\_col[num\_df]['time\_sec\_' + fea\_name[num\_df]] >= 0 ) & ( fea\_col[num\_df]['time\_sec\_' + fea\_name[num\_df]] <= limit\_sec + 30)]

    fea\_col[num\_df] = fea\_col[num\_df].resample('30s').mean()

    fea\_col[num\_df].drop(fea\_col[num\_df].columns[0], axis=1, inplace=True)

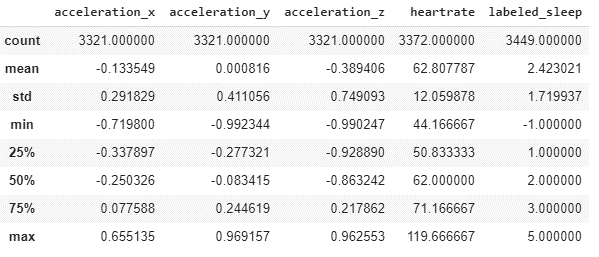
  df\_list.append(pd.concat([fea\_col[0], fea\_col[1], fea\_col[2]], axis=1))

df\_all = pd.concat([df\_list[0], df\_list[1], df\_list[2]], sort=False)

df\_all.columns = ['acceleration\_x', 'acceleration\_y', 'acceleration\_z', 'heartrate', 'labeled\_sleep']

df\_all.describe()

โดยได้ผลลัพธ์ดังนี้ จะพบว่ามีบาง column ที่มีค่า null value อีกทั้ง label มีค่าที่ไม่มีอยู่ในช่วง เช่น ค่า -1 เป็นต้น



* 1. Preprocess data

- ทำการ reset index และกำหนดชื่อ column ใหม่

df\_all.reset\_index(inplace=True)

df\_all.columns = ['uts', 'acceleration\_x', 'acceleration\_y',   
'acceleration\_z', 'heartrate', 'labeled\_sleep']

- ลบข้อมูลที่ซ้ำออก และลบคลาสที่ไม่อยู่ในช่วง 0-5 ในที่นี้คือ -1

df\_all.drop\_duplicates(inplace=True)

df\_all = df\_all[df\_all['labeled\_sleep'] >= 0]

- เติมค่า null value ด้วยค่ากึ่งกลาง

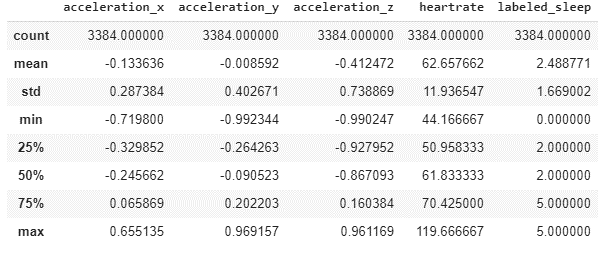
df\_all.fillna(df\_all. median()['acceleration\_x': 'heartrate'], inplace=True)

- แทรกข้อมูลในช่วงเวลาที่ขาดหายไป และจัดการลดสัญญาณรบกวนในข้อมูลด้วยการทำ Moving Average

df\_all.interpolate(method='slinear', inplace=True)

df\_all.rolling(2).mean()

- เมื่อทำการคลีนข้อมูลแล้วจำได้ผลลัพธ์ดังนี้



- รูปภาพเปรียบเทียบข้อมูลก่อน Cleaning data

before\_clean\_df.set\_index('uts', inplace=True)

before\_clean\_df[before\_clean\_df.columns].plot(subplots=True, layout=(2, 3), figsize=(15,10))

g = sns.PairGrid(before\_clean\_df, hue='labeled\_sleep')

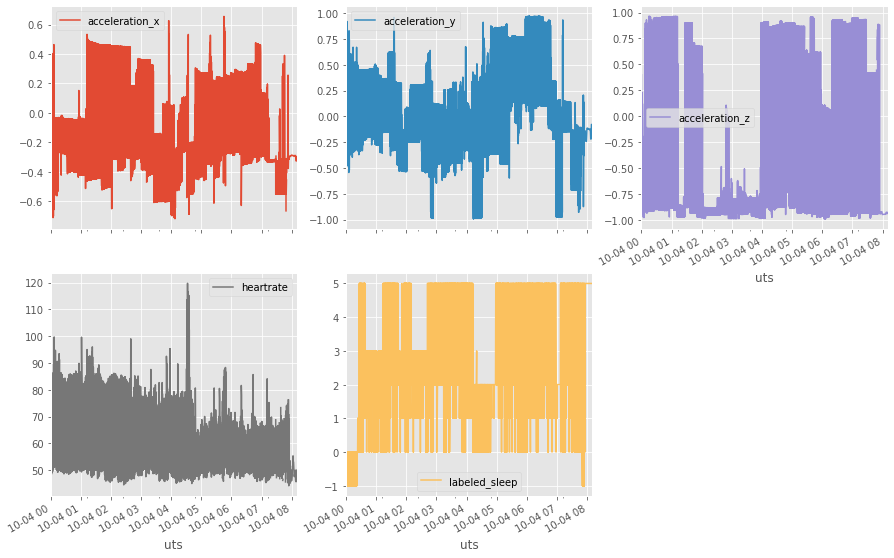
g = g.map\_offdiag(plt.scatter)

g = g.add\_legend()

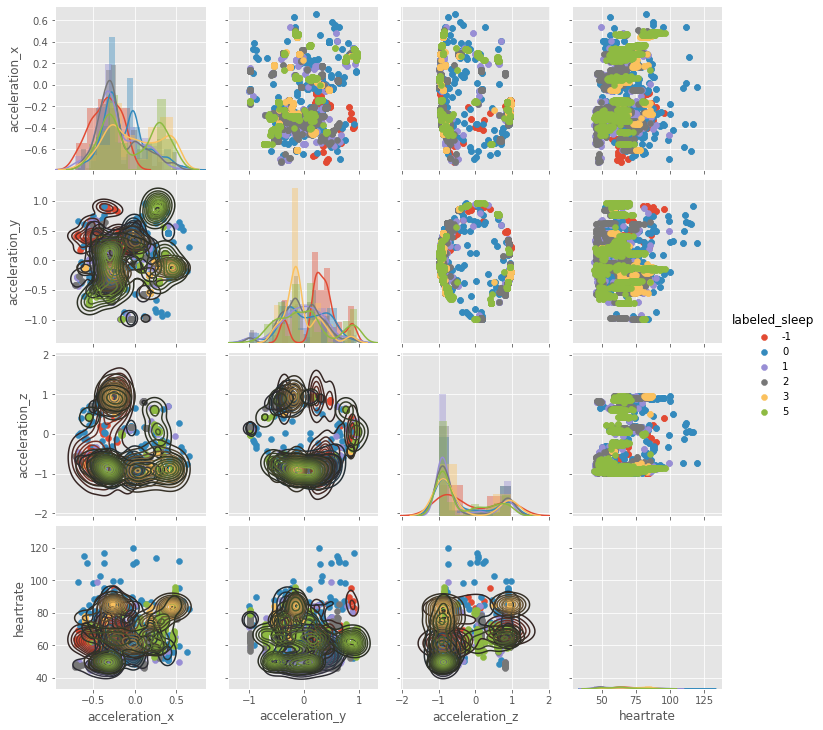
g.map\_upper(plt.scatter)

g.map\_lower(sns.kdeplot)

g.map\_diag(sns.distplot)



ภาพ Feature (Y-axis) – Time (X-axis) ก่อน Cleaning



ภาพ Statistical Plot ก่อน Cleaning

- รูปภาพเปรียบเทียบข้อมูลหลัง Cleaning data

df\_all[df\_all.columns].plot(subplots=True, layout=(2, 3),figsize=(15,10))

g = sns.PairGrid(df\_all, hue='labeled\_sleep')

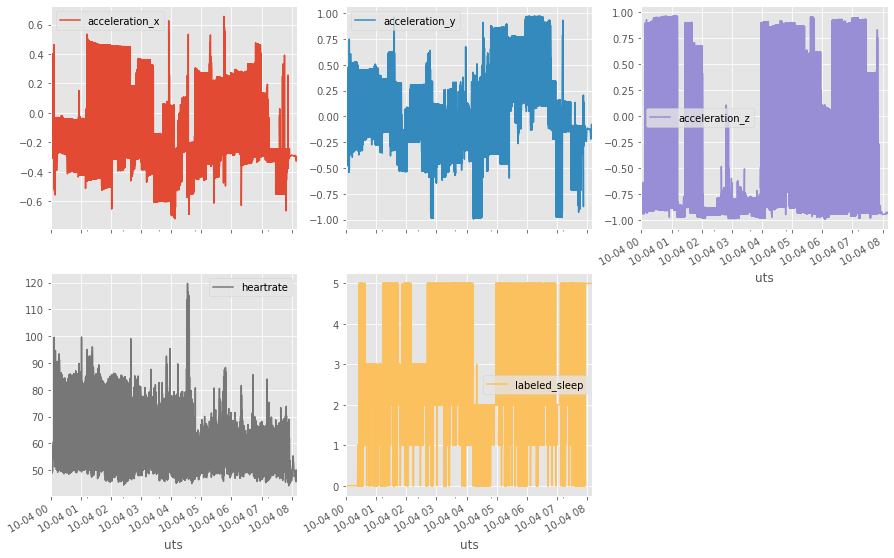
g = g.map\_offdiag(plt.scatter)

g = g.add\_legend()

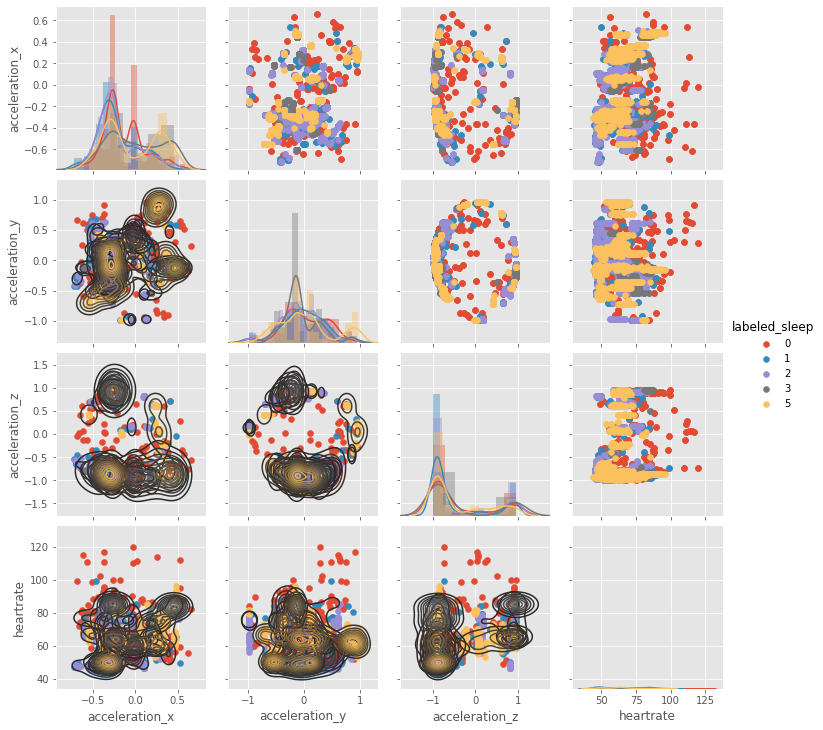
g.map\_upper(plt.scatter)

g.map\_lower(sns.kdeplot)

g.map\_diag(sns.distplot)



ภาพ Feature (Y-axis) – Time (X-axis) หลัง Cleaning data

ภาพ Statistical Plot หลัง Cleaning data

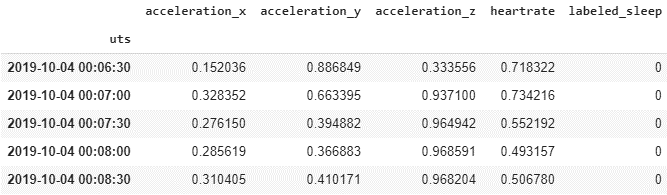
- ทำการ Normalize ข้อมูล(acceleration\_x, acceleration\_y, acceleration\_z, heartrate)ด้วยเทคนิค Max-Min Norm

for column in df\_all.loc[:, 'acceleration\_x':'heartrate'].columns :

    df\_all[column] = df\_all.apply(lambda row: (row[column] - df\_all[column].min()) / (df\_all[column].max() - df\_all[column].min()), axis=1)

df\_all

ตัวอย่างชุดข้อมูลที่ได้ดังนี้



- สร้างฟังก์ชันสำหรับสร้าง TimeSeries

def process\_create\_WindowTimeSeries(df, activity\_start, activity\_len, time\_window, n\_feature, step\_stride):

    df\_series = df

    segments = []

    # วนลูปโดยให้ i มีค่าตั้งแต่ row ที่ 0 ถึง ( จำนวน row - time\_window) โดยนับทีละ step\_stride

    for i in range(0, len(df\_series) - time\_window, step\_stride):

        # เก็บค่าของ row ที่ i ถึง i + time\_window

        df\_series\_feature = df\_series.iloc[i: i + time\_window]

        segments.append(np.array(df\_series\_feature))

    # ทำการ reshape ให้มีขนาด  ( #ชุด time\_series, #time\_step, #features )

    reshaped\_segments = np.asarray(segments).reshape(-1, time\_window, n\_feature)

    return reshaped\_segments

- ทำการเปลี่ยนข้อมูลให้อยู่ในรูปแบบของ Time Series โดยกำหนด time step เท่ากับ 3 และtime stride เท่ากับ 1

time\_step = 3

time\_stride = 1

col\_name = ['acceleration\_x', 'acceleration\_y', 'acceleration\_z', 'heartrate', 'labeled\_sleep']

time\_series = process\_create\_WindowTimeSeries(df\_all[col\_name], 0, len(col\_name), time\_step, len(col\_name), time\_stride)

time\_series\_2d = time\_series.reshape(time\_series.shape[0]\*time\_series.shape[1],time\_series.shape[2])

* 1. Prepare Label Ground Truth (y) สำหรับแต่ละชุด timeseries ด้วย majority vote และ ทำOne-Hot Encoding

- สร้าง X สำหรับแปลงข้อมูล feature ให้อยู่ในรูป array

col\_name = ['acceleration\_x', 'acceleration\_y', 'acceleration\_z', 'heartrate']

X = process\_create\_WindowTimeSeries(df\_all[col\_name], 0, len(col\_name), time\_step, len(col\_name), time\_stride)

X\_2d = X.reshape(X.shape[0]\*X.shape[1], X.shape[2])  # convert to 2D

- สร้างฟังชันสำหรับการทำ Majority vote โดยแบ่งเป็นฟังชันสำหรับหา Majority vote ของโมเดล CNN และฟังชันสำหรับหา Majority vote ของโมเดล LSTM

def majority\_3d(time\_series=time\_series):

  mj\_list = list()

  for t in range(time\_series.shape[0]) :

    mj\_vote = list()

    for r in range(time\_series.shape[1]):

      mj\_vote.append( time\_series[t][r][time\_series.shape[2] - 1] )

    mj\_counter = Counter(mj\_vote)

    mj\_list.append(mj\_counter.most\_common(1)[0][0])

  return mj\_list

def majority\_2d(time\_series=time\_series\_2d):

  mj\_list = list()

  for r in range(time\_series.shape[0]) :

    mj\_list.append(time\_series[r][-1])

  return mj\_list

- จากนั้นเรียกใช้งานฟังชันด้านบน

# Init y 1D

y = np.array(majority\_3d())

y = y.reshape(-1, 1)

# Onehot

enc = OneHotEncoder()

enc.fit(y)

y = enc.transform(y).toarray()

print(y.shape)

print('=======================================================')

# Init y 2D

y\_2d = np.array(majority\_2d())

y\_2d = y\_2d.reshape(-1, 1)

# Onehot

enc = OneHotEncoder()

enc.fit(y\_2d)

y\_2d = enc.transform(y\_2d).toarray()

print(y\_2d.shape)

* 1. Prepare training, validation, and test data (Train\_test\_split())

- ทำการแบ่งข้อมูลออกเป็น 3 ชุดคือ Train data, Valid data และ Test data โดยแบ่งเป็นข้อมูลสำหรับโมเดล CNN และ LSTM แยกกัน

X\_t, X\_test, y\_t, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)

X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X\_t, y\_t, test\_size=0.2, random\_state=42, stratify=y\_t)

X\_train = np.expand\_dims(X\_train, axis=-1)

X\_test = np.expand\_dims(X\_test, axis=-1)

X\_valid = np.expand\_dims(X\_valid, axis=-1)

print(X\_train.shape)

print(X\_test.shape)

print(X\_valid.shape)

print('=============================')

X\_2d\_train  = X\_train.reshape(-1, 3, 4)

X\_2d\_valid  = X\_valid.reshape(-1, 3, 4)

X\_2d\_test  = X\_test.reshape(-1, 3, 4)

print(X\_2d\_train.shape)

print(X\_2d\_test.shape)

print(X\_2d\_valid.shape)

ได้ผลลัพธ์ดังรูปด้านล่าง โดยแสดงเป็นขนาดของข้อมูล train, test และ valid ตามลำดับ

