**เอกสารอธิบายการทดลองที่ 1 ตอนที่ 1**

**พื้นฐานการอ่านไฟล์ข้อมูล การแก้ปัญหาข้อมูลหาย การปรับช่วงค่าของข้อมูล   
การปรับลดมิติข้อมูล และแสดงผลข้อมูลในเชิงกราฟ**

**ตอนที่ 1:** การทดลองอ่านไฟล์ข้อมูล การแก้ปัญหา ข้อมูลหาย และการปรับช่วงค่าของข้อมูล แสดงข้อมูลเชิง

กราฟ และ การจัดเตรียมรูปแบบข้อมูลเพื่อนำเข้าโมเดล

* 1. ขั้นตอนการทดลองในการนำเข้าข้อมูล แก้ปัญหาข้อผิดพลาดของ และการปรับช่วงค่าของข้อมูล
* Import Library : นำเข้า library ทั้ง 4 ตัว ได้แก่ numpy, pandas, matplotlib.pyplot และ seaborn โดย Axes3D ใช้สำหรับการ plot ที่เป็น 3D

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from mpl\_toolkits.mplot3d import Axes3D

%matplotlib inline

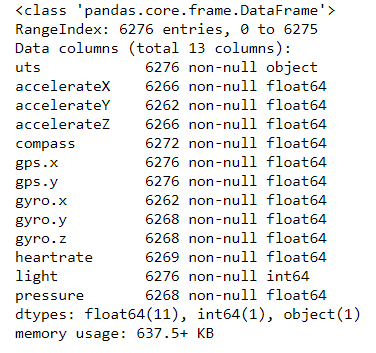
%config InlineBackend.figure\_format = 'retina'  #ความละเอียด 2เท่า

* Read Data File : อ่านไฟล์ csv ชื่อ watch\_test2\_sample จาก path

df = pd.read\_csv('watch\_test2\_sample.csv')

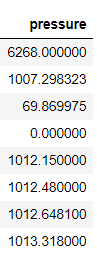
จากนั้นแสดงรายละเอียดพื้นฐาน Dataset เช่น data type, จำนวน row หรือ columnด้วยคำสั่งด้านล่าง

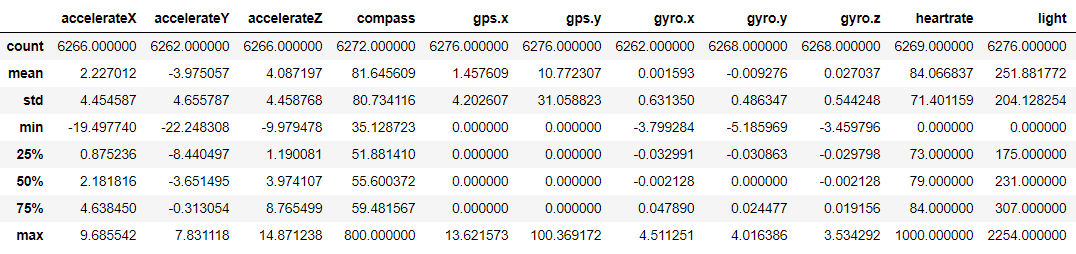
df.info()



และแสดงรายละเอียดทางสถิติของข้อมูลในแต่ละ column เช่น count mean std min และ max เป็นต้น โดยใช้โค้ด และได้ผลลัพธ์ตามรูปด้านล่าง

df.describe()





* Data Cleaning

เริ่มต้นด้วยการลบข้อมูลแถวที่มีค่าซ้ำหรือเหมือนกันด้วยคำสั่งด้านล่าง ซึ่งผลล์ที่ได้ทำให้ข้อมูลมีจำนวนแถวเหลืออยู่ 271 ข้อมูล จากเริ่มต้น 6276 ข้อมูล

df.drop\_duplicates(inplace=True)

จากนั้นทำการแทนค่า null value หรือค่าโดด

* เริ่มต้นโดย ลบ null value ใน column gyro.x, gyro.y และ gyro.z
* ลบ null value ใน column accelerateX, accelerateY, accelerateZ และ compass
* แทน null value ด้วยค่าเฉลี่ยของแต่ละ column โดยแทนใน column accelerateX, accelerateY, accelerateZ, heartrate, light และ pressure
* แทนค่า 0 ด้วยค่าเฉลี่ยของแต่ละ column โดยแทนใน column heartrate และ pressure
* แทนค่า 0 ด้วยข้อมูลของ row ก่อนหน้า ใน column gps.x และ gps.y
* และสุดท้ายแทนค่าที่มีค่าโดดเกินค่าที่กำหนดด้วยค่าเฉลี่ยของแต่ละ column   
  เมื่อ heartrate > 150 , light> 1000 และ compass > 400

df.dropna(subset=['gyro.x', 'gyro.y', 'gyro.z'], inplace=True)

df.dropna(subset=['accelerateX', 'accelerateY', 'accelerateZ', 'compass'], thresh=2, inplace=True)

df.fillna(df.mean()['accelerateX':'accelerateZ'], inplace=True)

df.fillna(df.mean()['heartrate':'pressure'], inplace=True)

df.replace({

    'heartrate': {0: df['heartrate'].mean()},

    'pressure': {0: df['pressure'].mean()},

}, inplace=True)

df[["gps.x", "gps.y"]]= df[["gps.x", "gps.y"]].replace(0, method='ffill')

df['heartrate'] = df['heartrate'].apply(lambda row: df['heartrate'].mean() if row > 150 else row)

df['light'] = df['light'].apply(lambda row: df['light'].mean() if row > 1000 else row)

df['compass'] = df['compass'].apply(lambda row: df['compass'].mean() if row > 400 else row)

จากนั้นจะทำการแยกช่วงเวลาที่ช่วงเวลาห่างกันเยอะเป็น 3 ช่วง ใช้สำหรับการทำ resample โดยทำการแยกจากเลข index

time\_1 = df[:33]

time\_2 = df[33:239]

time\_3 = df[239:]

เมื่อทำการแบ่งข้อมูลเป็น 3 ช่วงแล้วจะทำการ resample ดังฟังก์ชันด้านล่าง และสุดท้ายจะนำข้อมูลที่ได้จากการเข้าฟังก์ชันมาต่อกัน โดยจะมีข้อมูลอยู่ที่ 317 ข้อมูล

def resample\_df(df):

    df['uts'] = pd.to\_datetime(df['uts'])

    df.set\_index('uts', inplace=True)

    df = df.resample('20s').sum()

    print(df.shape)

    return df

resample\_1 = resample\_df(time\_1.reset\_index(drop=True))

resample\_2 = resample\_df(time\_2.reset\_index(drop=True))

resample\_3 = resample\_df(time\_3.reset\_index(drop=True))

df = pd.concat([resample\_1, resample\_2, resample\_3])

และสุดท้ายสำหรับการทำ data cleaning จะทำการแทรกข้อมูลในช่วงเวลาที่ขาดหายไปด้วย interpolate และจัดการลดสัญญาณรบกวนในข้อมูลด้วยการทำ Moving Average ด้วย rolling

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

df.rolling(2).mean()

* Data Normalization

ทำการ Normalize ข้อมูลด้วยเทคนิค Max-Min Normalization ดังโค้ดด้านล่าง

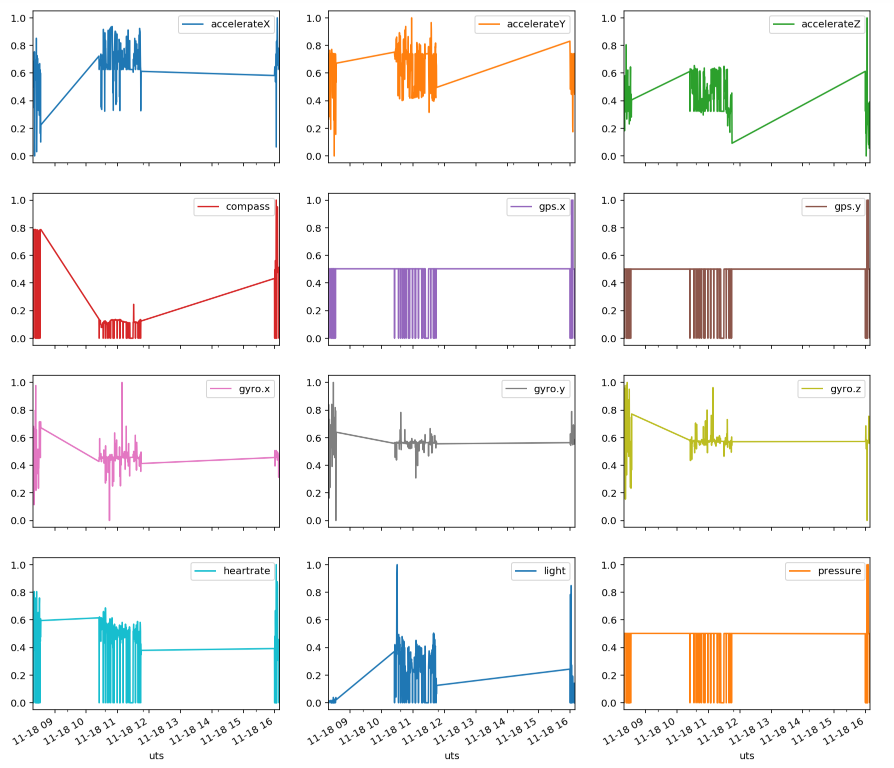
for column in df.loc[:, 'accelerateX': ].columns :

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

* 1. ขั้นตอนการแสดงข้อมูลเชิงกราฟ

- แสดงกราฟข้อมูลแต่ละ feature (Column) ด้วย Line Plot เพื่อดูค่าที่แท้จริง : ใช้คำสั่งที่มีอยู่ใน pandas โดยการแบ่งเป็น layout shape 5\*3 ขนาด 15\*18 ได้ผลลัพธ์ดังรูปด้านล่าง

df[df.columns].plot(subplots=True, layout=(5, 3), figsize=(15,18))



- แสดงกราฟข้อมูลความสัมพันธ์ระหว่างคู่ features ด้วย 3D Scatter Plot เพื่อดูความสัมพันธ์ของข้อมูลเชิง 3 มิติ

fig = plt.figure(figsize=(20, 10))

ax = fig.add\_subplot(1, 2, 1, projection='3d')

ax.scatter(df.accelerateX, df.accelerateY, df.accelerateZ, c='cyan', s=20, edgecolor='k')

ax.set\_xlabel('x')

ax.set\_xlabel('y')

ax.set\_xlabel('z')

ax.view\_init(30, -60)

ax = fig.add\_subplot(1, 2, 2, projection='3d')

ax.scatter(df['gyro.x'], df['gyro.y'], df['gyro.z'],c='violet', s=20,edgecolor='k')

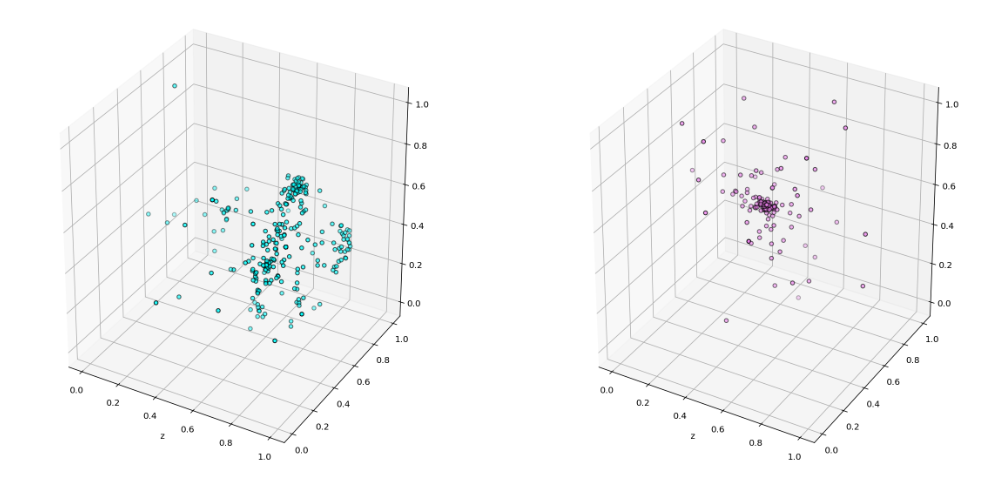
ax.set\_xlabel('x')

ax.set\_xlabel('y')

ax.set\_xlabel('z')

ax.view\_init(30, -60)

plt.show()



- แสดงข้อมูลเชิงพิกัด Geolocation ของ ข้อมูล (gps.x, gps.y) :

เริ่มต้นด้วยการ Export ภาพแผนที่จาก https://www.openstreetmap.org/export#map=6/12.780/102.986   
จากนั้นจะทำการอ่านไฟล์ภาพแผนที่ด้วย imread และทำการ scatter plot ลงบนแผนที่

map\_im = plt.imread('map.png')

fig, ax = plt.subplots(figsize=(15, 10))

# plot ตำแหน่งพิกัด GPS

ax.scatter(before\_resample['gps.x'], before\_resample['gps.y'], zorder=1, alpha=0.5, c='r', s=20)

ax.set\_title('Plotting Spatial Data on Map')

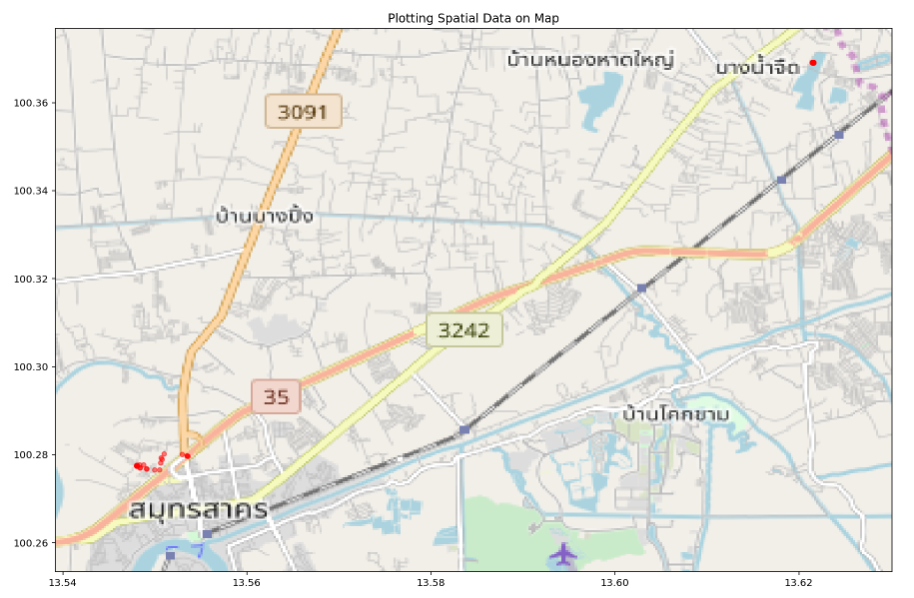
# lat\_min, lat\_max, long\_min, long\_max

BBox =[13.5392, 13.6301, 100.2535, 100.3768]

ax.set\_xlim(BBox[0], BBox[1])

ax.set\_ylim(BBox[2], BBox[3])

ax.imshow(map\_im, zorder=0, extent=BBox, aspect='auto')



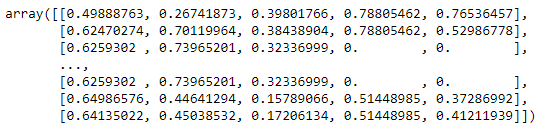
* 1. ขั้นตอนการจัดเตรียมข้อมูลเพื่อนำเข้าโมเดล
* ทำการจัดข้อมูล 5 Features [accelerateX, accelerateY, accelerateZ, compass, heartrate]

อะเรย์ชุดที่ 1: เป็นการจัดเรียงข้อมูล โดยต้องการ row: single time sample / column: 5 features

array\_1 = np.array(df[['accelerateX', 'accelerateY', 'accelerateZ', 'compass', 'heartrate']])

array\_1.shape





อะเรย์ชุดที่ 2: เป็นการจัดเรียงข้อมูล time series ในรูปของ อะเรย์ 3 มิติ : จะทำการสร้างฟังก์ชัน process\_create\_WindowTimeSeries สำหรับจัดเรียงข้อมูลรูปแบบ time series ดังโค้ดด้านล่าง

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

จากนั้นทำการกำหนด time\_step, time\_stride, ชื่อ column เพื่อใช้สำหรับเป็นพารามิเตอร์ในฟังก์ชัน process\_create\_WindowTimeSeries

time\_step = 3

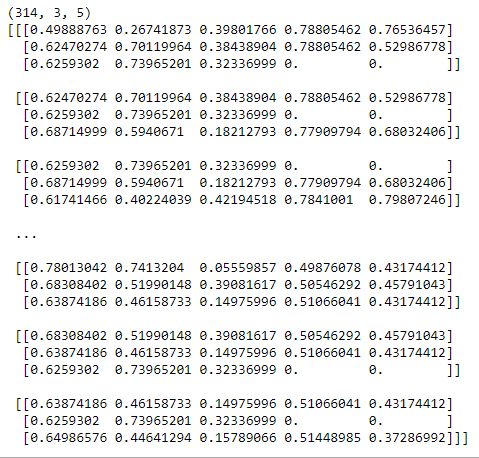
time\_stride = 1

col\_name = ['accelerateX', 'accelerateY', 'accelerateZ', 'compass', 'heartrate']

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

print(time\_series.shape)

print(time\_series)

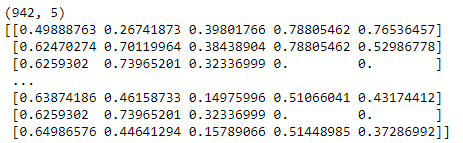


สุดท้ายทำการปรับอะเรย์ 3 มิติที่ได้ ให้อยู่ในรูปของ 2 มิติขนาด

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

print(time\_series\_2d.shape)

print(time\_series\_2d)



* แสดงภาพของอะเรย์ชุดที่ 1 และชุดที่ 2

# แสดงภาพของอะเรย์ชุดที่ 1

fig, ax = plt.subplots(figsize=(15, 10))

ax.set\_title('Data')

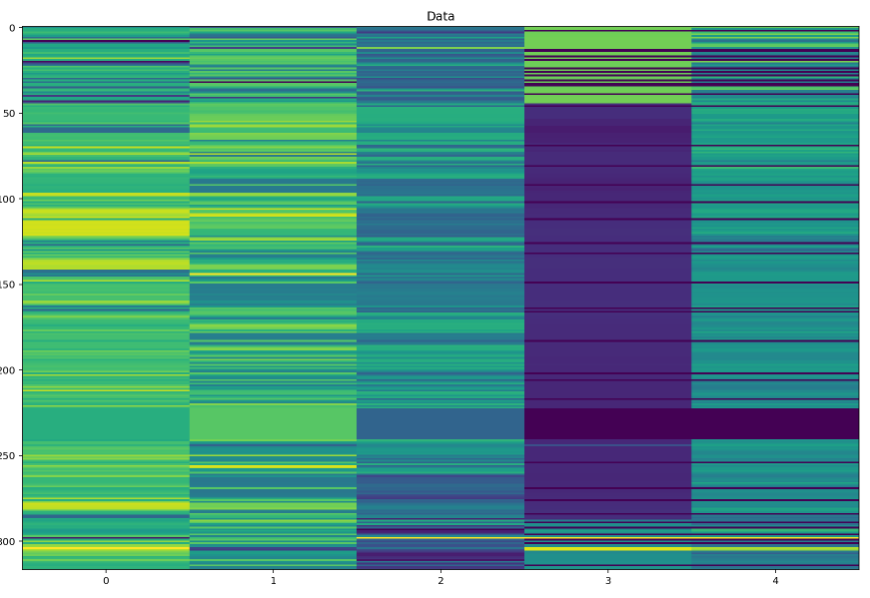
ax.imshow(array\_1, aspect='auto')

# แสดงภาพของอะเรย์ชุดที่ 2 (TimeSeries)

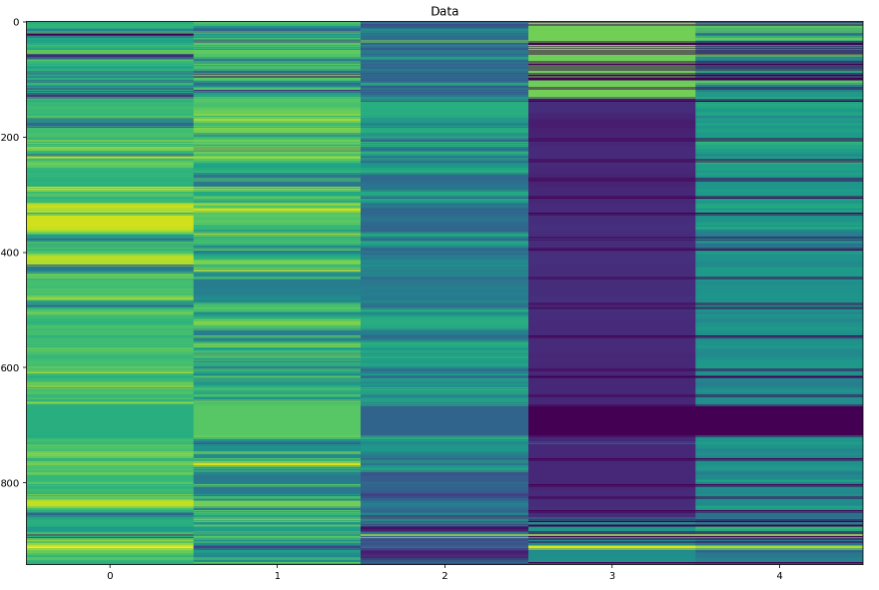
fig, ax = plt.subplots(figsize=(15, 10))

ax.set\_title('Data')

ax.imshow(time\_series\_2d, aspect='auto')



ภาพของอะเรย์ชุดที่ 1



ภาพของอะเรย์ชุดที่ 2