เอกสารอธิบายการทดลองที่ 1 ตอนที่ 1

พื้นฐานการอ่านไฟล์ข้อมูล การแก้ปัญหาข้อมูลหาย การปรับช่วงค่าของข้อมูล การปรับลดมิติข้อมูล และแสดงผลข้อมูลในเชิงกราฟ

ตอนที่ 1: การทดลองอ่านไฟล์ข้อมูล การแก้ปัญหา ข้อมูลหาย และการปรับช่วงค่าของข้อมูล แสดงข้อมูลเชิงกราฟ และ การจัดเตรียมรูปแบบข้อมูลเพื่อนำเข้าโมเดล

1.1 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 column ด้วยคำสั่งด้านล่าง

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6276 entries, 0 to 6275
Data columns (total 13 columns):
uts            6276 non-null object
accelerateX    6266 non-null float64
accelerateY    6262 non-null float64
accelerateZ    6266 non-null float64
compass        6272 non-null float64
gps.x          6276 non-null float64
gps.y          6276 non-null float64
gyro.x         6262 non-null float64
gyro.y         6268 non-null float64
gyro.z         6268 non-null float64
heartrate      6269 non-null float64
light          6276 non-null int64
pressure       6268 non-null float64
dtypes: float64(11), int64(1), object(1)
memory usage: 637.5+ KB
```

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

```
df.describe()
```

	accelerateX	accelerateY	accelerateZ	compass	gps.x	gps.y	gyro.x	gyro.y	gyro.z	heartrate	light	pressure
count	6266.000000	6262.000000	6266.000000	6272.000000	6276.000000	6276.000000	6262.000000	6268.000000	6268.000000	6269.000000	6276.000000	6268.000000
mean	2.227012	-3.975057	4.087197	81.645609	1.457609	10.772307	0.001593	-0.009276	0.027037	84.066837	251.881772	1007.298323
std	4.454587	4.655787	4.458768	80.734116	4.202607	31.058823	0.631350	0.486347	0.544248	71.401159	204.128254	69.869975
min	-19.497740	-22.248308	-9.979478	35.128723	0.000000	0.000000	-3.799284	-5.185969	-3.459796	0.000000	0.000000	0.000000
25%	0.875236	-8.440497	1.190081	51.881410	0.000000	0.000000	-0.032991	-0.030863	-0.029798	73.000000	175.000000	1012.150000
50%	2.181816	-3.651495	3.974107	55.600372	0.000000	0.000000	-0.002128	0.000000	-0.002128	79.000000	231.000000	1012.480000
75%	4.638450	-0.313054	8.765499	59.481567	0.000000	0.000000	0.047890	0.024477	0.019156	84.000000	307.000000	1012.648100
max	9.685542	7.831118	14.871238	800.000000	13.621573	100.369172	4.511251	4.016386	3.534292	1000.000000	2254.000000	1013.318000

- 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 r
ow)
        df['compass'] = df['compass'].apply(lambda row: df['compass'].mean() if row > 400 e
lse 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

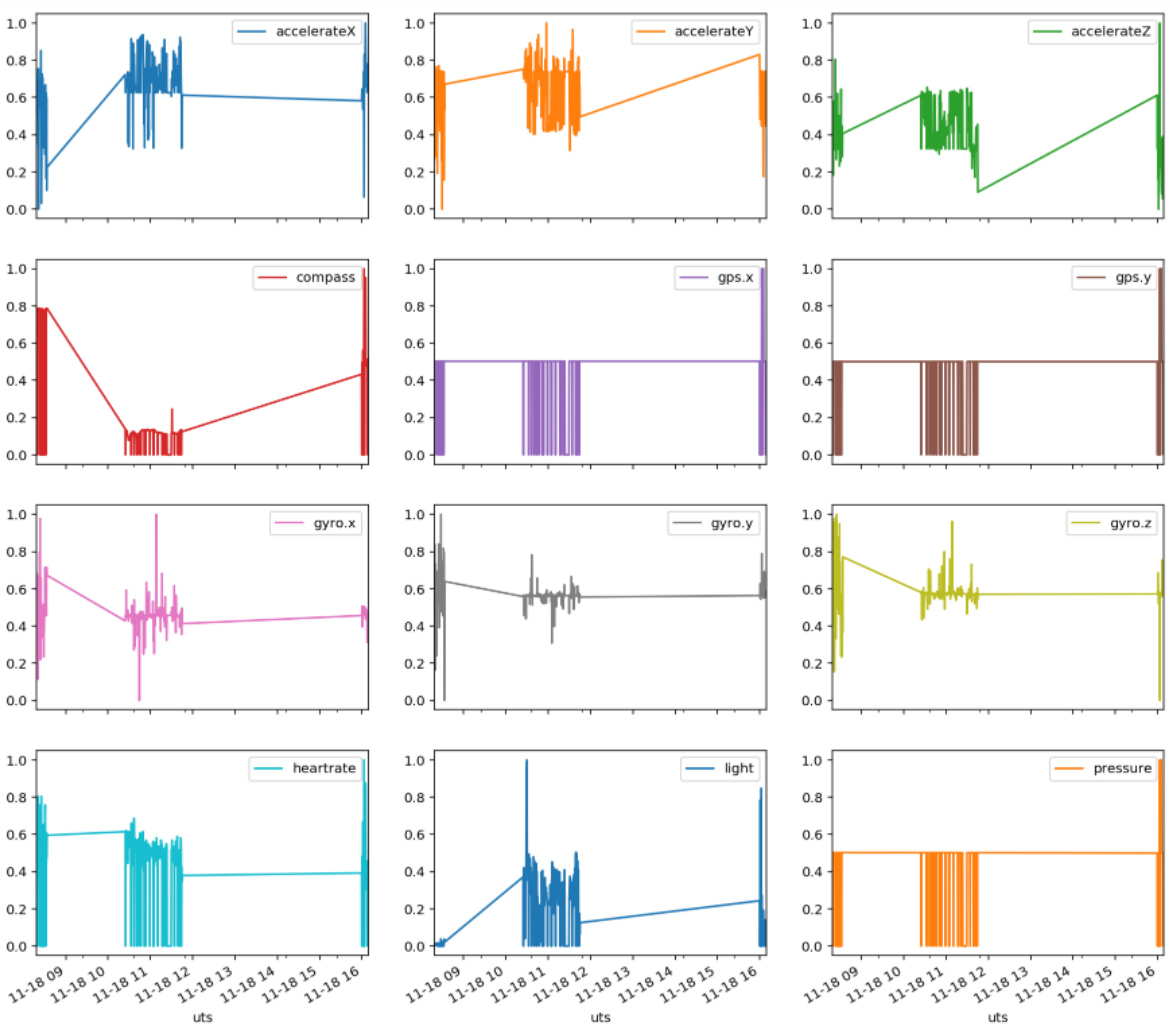
ทำการ Normalize ข้อมูลด้วยเทคนิค Max-Min Normalization ดังโค้ดด้านล่าง

```
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)
```

1.2 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))
```

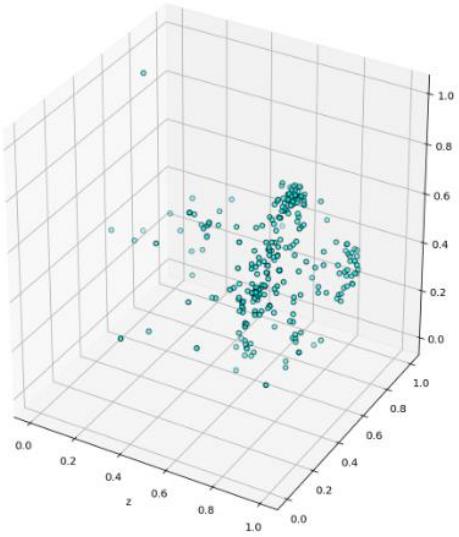


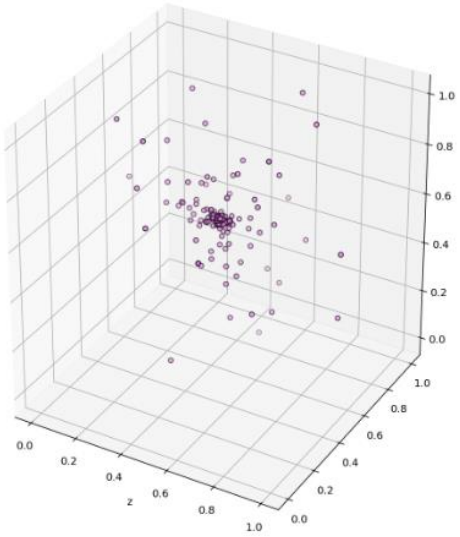
- แสดงกราฟข้อมูลความสัมพันธ์ระหว่างคู่ features ด้วย 3D Scatter Plot เพื่อดูความสัมพันธ์ของข้อมูลเชิง 3 มิติ

```
        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, edgecolo
r='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) :

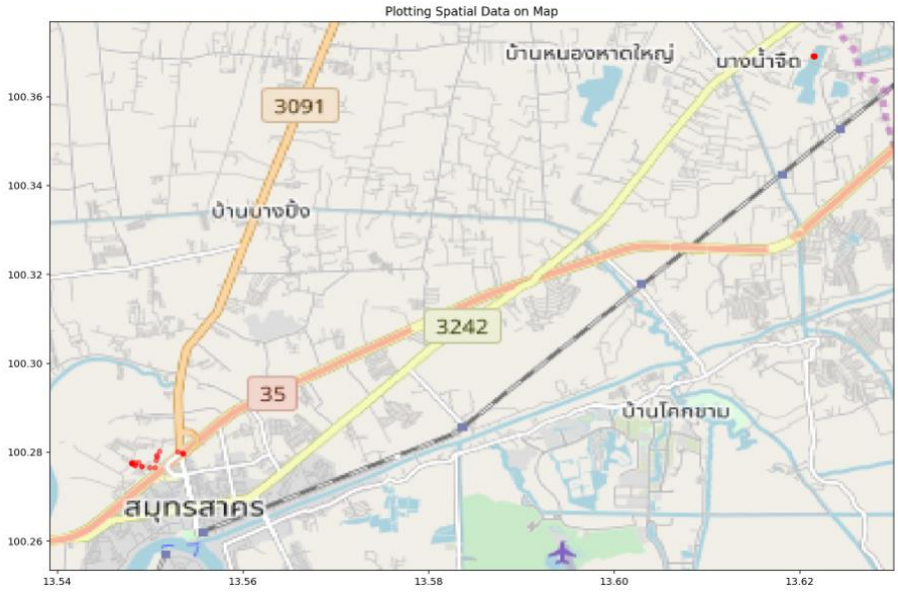
เริ่มต้นด้วยการ Export ภาพแผนที่จาก https://www.openstreetmap.org/export#map=6/12.780/102.986 จากนั้นจะทำการอ่านไฟล์ภาพแผนที่ด้วย imread และทำการ scatter plot ลงบนแผนที่

```
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')
```



1.3 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
```

```
(317, 5)
```

```
array([[0.49888763, 0.26741873, 0.39801766, 0.78805462, 0.76536457],
       [0.62470274, 0.70119964, 0.38438904, 0.78805462, 0.52986778],
       [0.6259302 , 0.73965201, 0.32336999, 0.        , 0.        ],
       ...,
       [0.6259302 , 0.73965201, 0.32336999, 0.        , 0.        ],
       [0.64986576, 0.44641294, 0.15789066, 0.51448985, 0.37286992],
       [0.64135022, 0.45038532, 0.17206134, 0.51448985, 0.41211939]])
```

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)
```

```
(314, 3, 5)
[[[0.49888763 0.26741873 0.39801766 0.78805462 0.76536457]
  [0.62470274 0.70119964 0.38438904 0.78805462 0.52986778]
  [0.6259302  0.73965201 0.32336999 0.         0.        ]]

 [[0.62470274 0.70119964 0.38438904 0.78805462 0.52986778]
  [0.6259302  0.73965201 0.32336999 0.         0.        ]
  [0.68714999 0.5940671  0.18212793 0.77909794 0.68032406]]

 [[0.6259302  0.73965201 0.32336999 0.         0.        ]
  [0.68714999 0.5940671  0.18212793 0.77909794 0.68032406]
  [0.61741466 0.40224039 0.42194518 0.7841001  0.79807246]]

 ...

 [[0.78013042 0.7413204  0.05559857 0.49876078 0.43174412]
  [0.68308402 0.51990148 0.39081617 0.50546292 0.45791043]
  [0.63874186 0.46158733 0.14975996 0.51066041 0.43174412]]

 [[0.68308402 0.51990148 0.39081617 0.50546292 0.45791043]
  [0.63874186 0.46158733 0.14975996 0.51066041 0.43174412]
  [0.6259302  0.73965201 0.32336999 0.         0.        ]]

 [[0.63874186 0.46158733 0.14975996 0.51066041 0.43174412]
  [0.6259302  0.73965201 0.32336999 0.         0.        ]
  [0.64986576 0.44641294 0.15789066 0.51448985 0.37286992]]]
```

สุดท้ายทำการปรับอะเรย์ 3 มิติที่ได้ ให้อยู่ในรูปของ 2 มิติขนาด

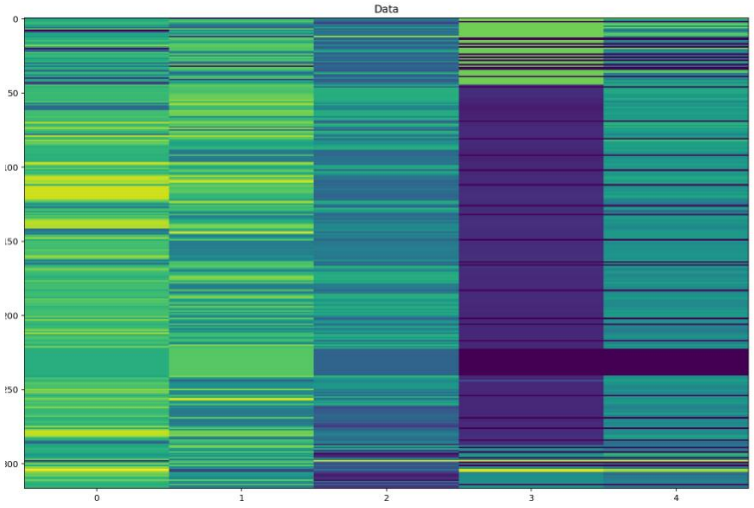
```
time_series_2d = time_series.reshape(time_series.shape[0]*time_series.shape[1],time_series.shape[2])
print(time_series_2d.shape)
print(time_series_2d)
```

```
(942, 5)
[[0.49888763 0.26741873 0.39801766 0.78805462 0.76536457]
 [0.62470274 0.70119964 0.38438904 0.78805462 0.52986778]
 [0.6259302  0.73965201 0.32336999 0.         0.        ]
 ...
 [0.63874186 0.46158733 0.14975996 0.51066041 0.43174412]
 [0.6259302  0.73965201 0.32336999 0.         0.        ]
 [0.64986576 0.44641294 0.15789066 0.51448985 0.37286992]]
```

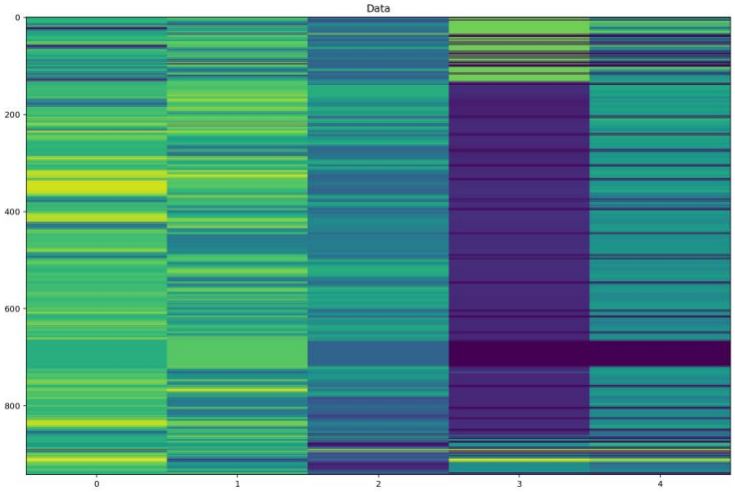
- แสดงภาพของอะเรย์ชุดที่ 1 และชุดที่ 2

```
# 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