เอกสารอธิบายการทดลองที่ 3 ตอนที่ 1

การทดลองเตรียมข้อมูล ปรับค่าข้อมูล และจัดแบ่งชุด Train, Test เพื่อสอนโมเดล

ตอนที่ 1: การทดลองเตรียมข้อมูล ปรับค่าข้อมูล และจัดแบ่งชุด Train, Test เพื่อสอนโมเดล

1.1 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')
```

1.2 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=" ", head
er=None)
    if len(fea_df.columns) < 2 :
      fea_df = pd.read_csv(root_dir + '/Sleep_MiNiSet/' + n + '_' + f + '.txt', sep=",", he
ader=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', 'label
ed_sleep']
df_all.describe()
```

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

	acceleration_x	acceleration_y	acceleration_z	heartrate	labeled_sleep
count	3321.000000	3321.000000	3321.000000	3372.000000	3449.000000
mean	-0.133549	0.000816	-0.389406	62.807787	2.423021
std	0.291829	0.411056	0.749093	12.059878	1.719937
min	-0.719800	-0.992344	-0.990247	44.166667	-1.000000
25%	-0.337897	-0.277321	-0.928890	50.833333	1.000000
50%	-0.250326	-0.083415	-0.863242	62.000000	2.000000
75%	0.077588	0.244619	0.217862	71.166667	3.000000
max	0.655135	0.969157	0.962553	119.666667	5.000000

1.3 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()
```

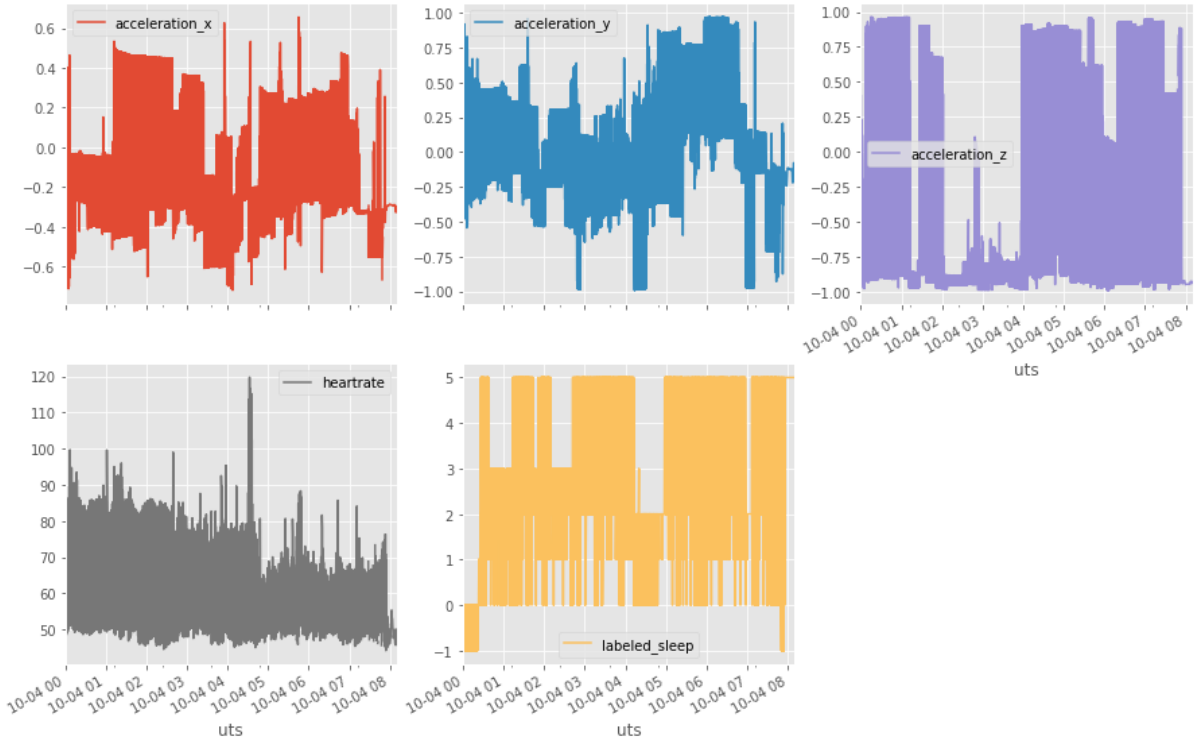
- เมื่อทำการคลีนข้อมูลแล้วจำได้ผลลัพธ์ดังนี้

	acceleration_x	acceleration_y	acceleration_z	heartrate	labeled_sleep
count	3384.000000	3384.000000	3384.000000	3384.000000	3384.000000
mean	-0.133636	-0.008592	-0.412472	62.657662	2.488771
std	0.287384	0.402671	0.738869	11.936547	1.669002
min	-0.719800	-0.992344	-0.990247	44.166667	0.000000
25%	-0.329852	-0.264263	-0.927952	50.958333	2.000000
50%	-0.245662	-0.090523	-0.867093	61.833333	2.000000
75%	0.065869	0.202203	0.160384	70.425000	5.000000
max	0.655135	0.969157	0.961169	119.666667	5.000000

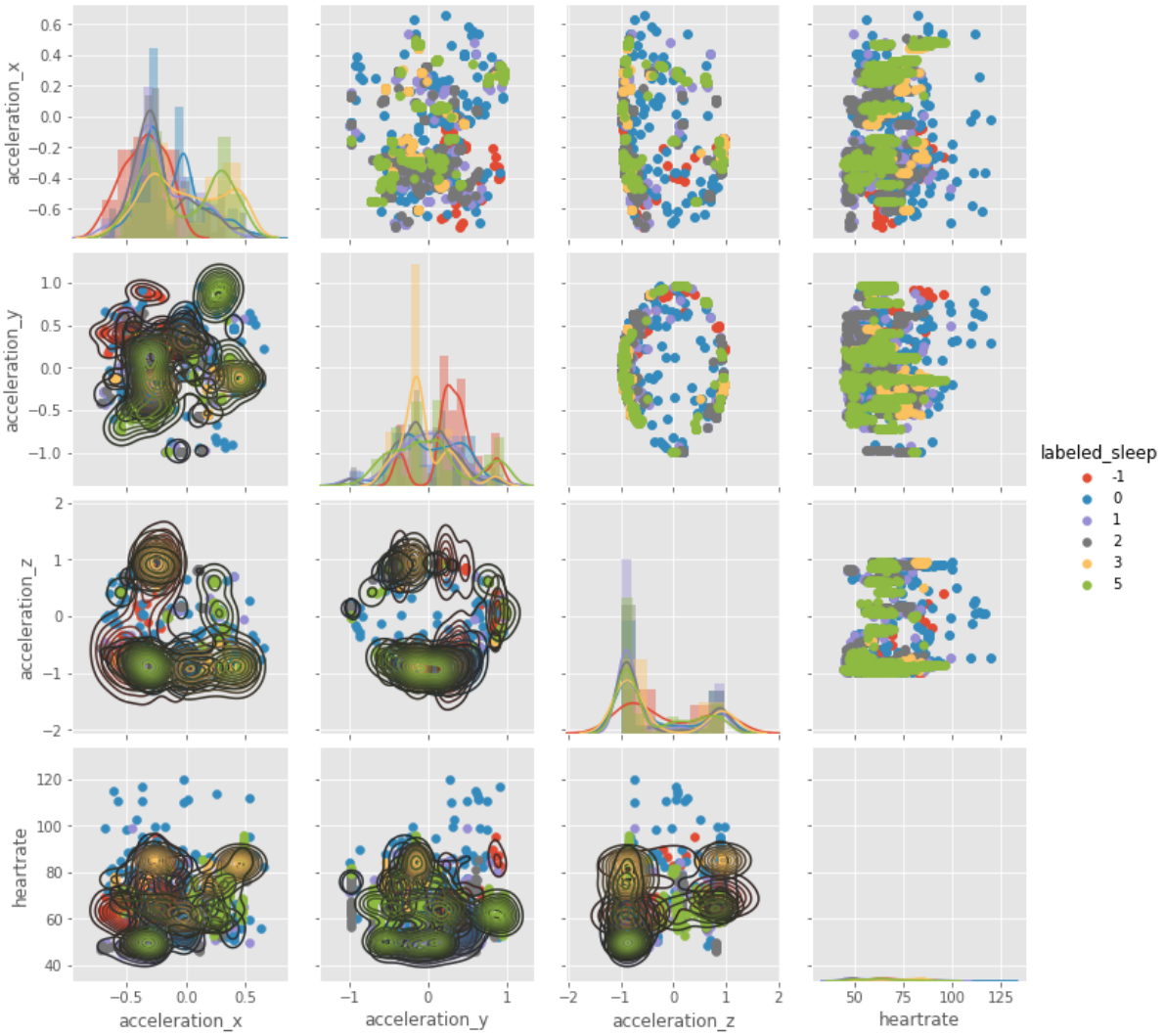
- รูปภาพเปรียบเทียบข้อมูลก่อน Cleaning data

```
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

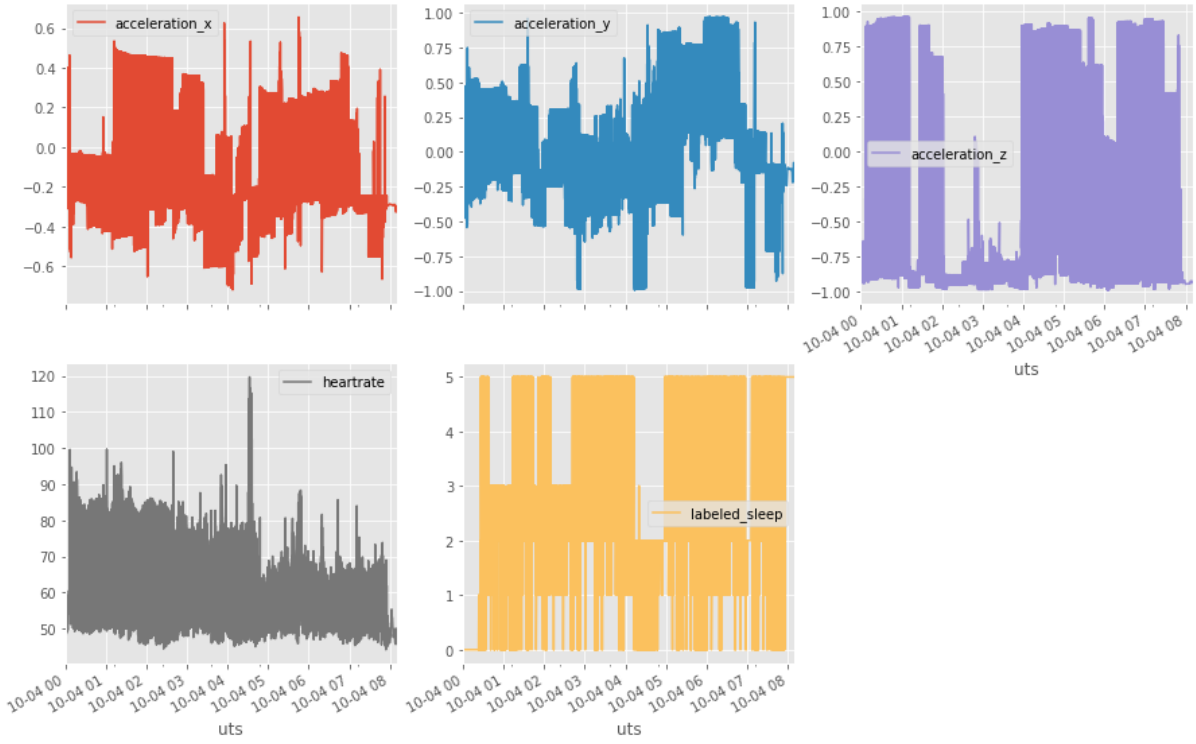


ภาพ Statistical Plot ก่อน Cleaning

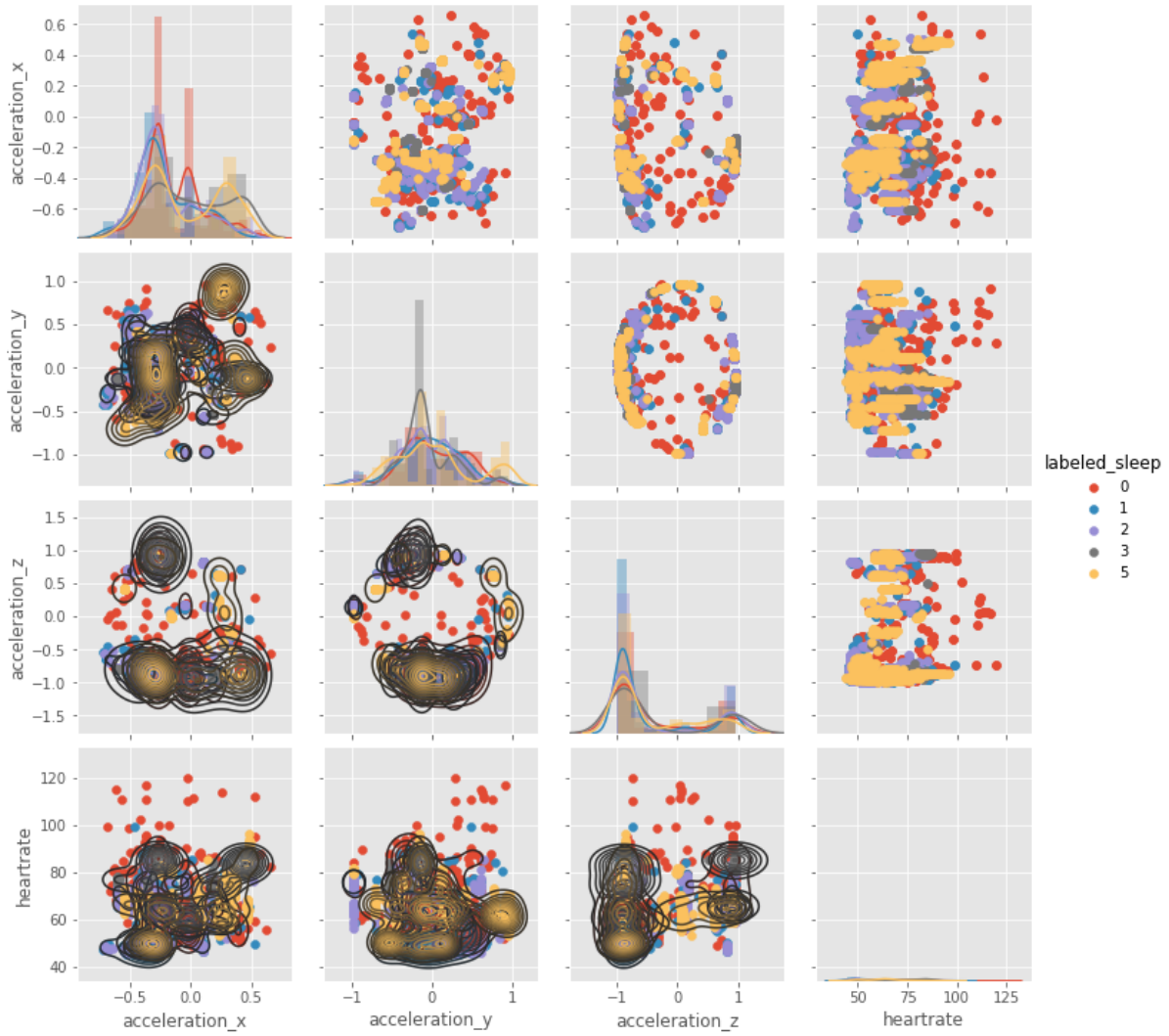
- รูปภาพเปรียบเทียบข้อมูลหลัง Cleaning data

```
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

uts	acceleration_x	acceleration_y	acceleration_z	heartrate	labeled_sleep
2019-10-04 00:06:30	0.152036	0.886849	0.333556	0.718322	0
2019-10-04 00:07:00	0.328352	0.663395	0.937100	0.734216	0
2019-10-04 00:07:30	0.276150	0.394882	0.964942	0.552192	0
2019-10-04 00:08:00	0.285619	0.366883	0.968591	0.493157	0
2019-10-04 00:08:30	0.310405	0.410171	0.968204	0.506780	0

- สร้างฟังก์ชันสำหรับสร้าง TimeSeries

```
def process_create_WindowTimeSeries(df, activity_start, activity_len, time_windo
w, 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 และ time 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],t
ime_series.shape[2])
```

1.4 Prepare Label Ground Truth (y) สำหรับแต่ละชุด timeseries ด้วย majority vote และ ทำ One-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_ste
p, 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)
```

1.5 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=4
2, stratify=y)
X_train, X_valid, y_train, y_valid = train_test_split(X_t, y_t, test_size=0.2, r
andom_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

```
(2163, 3, 4, 1)
(677, 3, 4, 1)
(541, 3, 4, 1)
==========================
(2163, 3, 4)
(677, 3, 4)
(541, 3, 4)
```