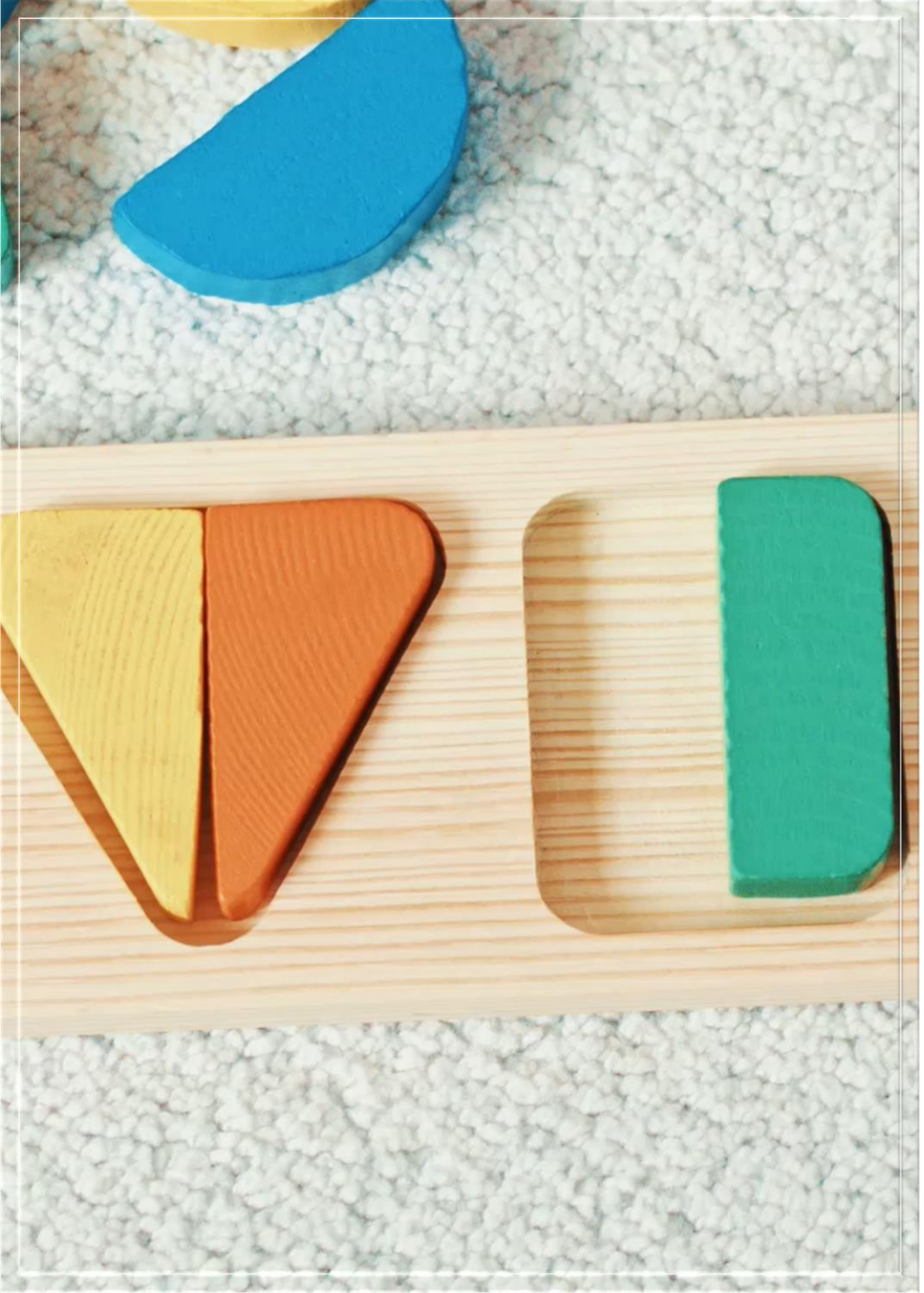
ACTIVITY #8

CLASSIFICATION MODEL

SVM

Time series Data



AGENDA

8.1	Data Preparation
8.2	Model Training and Testing
8.3	Hyperparameter tuning

8.1

DATA PREPARATION

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

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

8.1 (a) Load and Prepare Data

1

- # Load data from csv 3 files
- # acceleration.txt, heartrate.txt, labeled_sleep.txt
  - ACC = read_csv(acceleration.txt, sep = ' ',names=['timedelta', 'accX', 'accY', 'accZ'])
  - HeartR = read_csv(heartrate.txt, sep = ',',names=['timedelta', 'heartrate'])
  - SleepL = read_csv(labeled_sleep.txt, sep = ' ',names=['timedelta', 'sleep'])

2

- # Check 'timedelta' max(), min() of ACC, HeartR, SleepL (ช่วงเวลาที่มีข้อมูลใกล้กัน)
  - Ex
    - ACC_max_date = ACC['timedelta'].max()
    - ACC_min_date = ACC['timedelta'].min()
    - หา start_timedelta, end_timedelta

```
ACC start:  -124489.16105 ACC end:  17643.046417
HeartR start:  -355241.73971 HeartR end:  34491.1535499
SleepL start:  0 SleepL end:  28530
```

3

- # select only intersected timedelta (ACC, HeartR, SleepL) (ช่วงเวลาที่มีข้อมูลใกล้กัน)
- Ex
  - ACC_new = ACC[(ACC['timedelta'] > start_timedelta) &(df_acc['timedelta'] < end_timedelta)]

8.1 (b) Load and Prepare Data (ACC)

4

- # ---------- Rounding ACC (Rounding to 1 sec) ----------
  - # Convert to datetime and round to second,
    - ACC_new['timedelta'] = pd.DataFrame(pd.to_timedelta(ACC_new['timedelta'], timedelta_unit).round('1s'))

5

- # Average rounding duplicated time
- # df_acc_X = ACC_new.groupby('timedelta')['accX'].mean().reset_index()
- df_acc_Y = ACC_new.groupby('timedelta')['accY'].mean().reset_index()
- df_acc_Z = ACC_new.groupby('timedelta')['accZ'].mean().reset_index()

6

- # acc_X, acc_Y, acc_Z
- Ex
  - pd.concat([df_acc_X, df_acc_Y, df_acc_Z], axis=1)

Before / After

convert datetime and round and average to 1s

```
------ Before convert datetime and round and average to 1s ------------
       timedelta      accX      accY      accZ
0 -124489.161050  0.017487 -0.586700 -0.805771
1 -124489.116395  0.018982 -0.589676 -0.809158
2 -124489.115548  0.020966 -0.580887 -0.815048
3 -124489.114691  0.019485 -0.580872 -0.813583
4 -124489.097700  0.016998 -0.587204 -0.806259
------ After convert datetime and round and average to 1s ------------
        timedelta      accX      accY      accZ
0 0 days 00:00:00 -0.243203  0.895372  0.367591
1 0 days 00:00:01 -0.240757  0.873826  0.415446
2 0 days 00:00:02 -0.244620  0.883943  0.387026
3 0 days 00:00:03 -0.248036  0.902427  0.347812
4 0 days 00:00:04 -0.241778  0.912946  0.321502
```

8.1 (c) Load and Prepare Data (Heart rate)

1

- # ---------- Rounding Heart Rate (Rounding to 1 sec) ----------
  - HeartR_new['timedelta'] = pd.DataFrame(pd.to_timedelta(HeartR_new['timedelta'], timedelta_unit).round('1s'))

2

- # Resampling every 1s with median with ffill
- resample_rule = '1s'
- HeartR_new2 = HeartR_new.set_index('timedelta').resample(resample_rule,).median().ffill()

8.1 (d) Load and Prepare Data (Sleep Label)

1

- # ---------- Rounding Sleep Label (Rounding to 1 sec) ----------
  - SleepL_new['timedelta'] = pd.DataFrame(pd.to_timedelta(SleepL_new['timedelta'], timedelta_unit).round('1s'))

2

- # Resampling every 1s with median with ffill
- resample_rule = '1s'
- SleepL_new2 = SleepL_new.set_index('timedelta').resample(resample_rule,).median().ffill()

8.1 (e) Merge Data and Standardized data

1

- # ----------Merge All Data ----------
  - df = []
  - df = pd.merge_asof(ACC_new2, HeartR_new2, on='timedelta')
  - df = pd.merge_asof(df, df_SleepL_new2, on = 'timedelta')

2

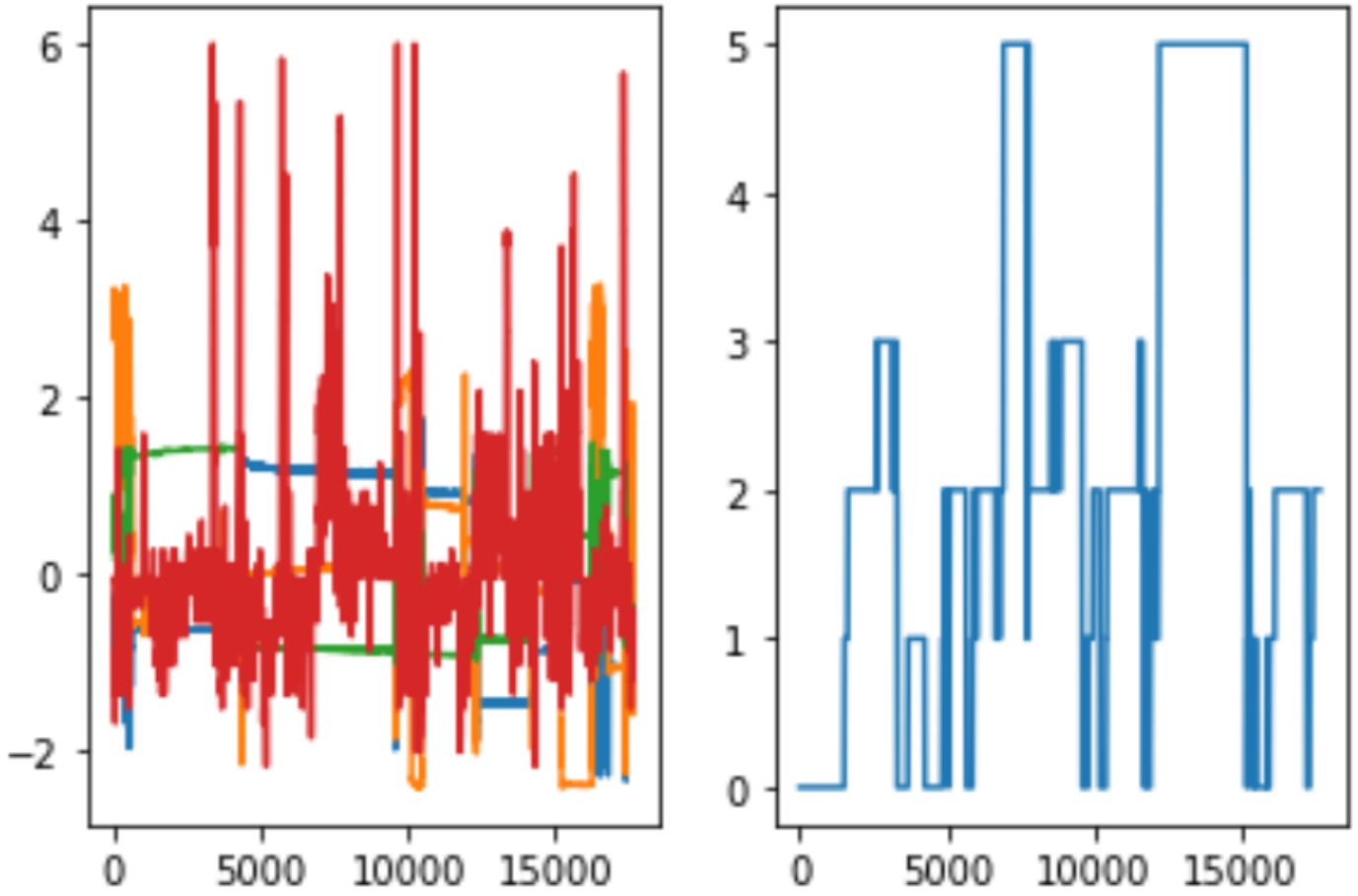
- # Fill NA
- # Heart rate
  - Fillna() # using median()
- # Sleep Label
  - Fillna() # with 0
- # Drop column
  - drop('timedelta')

3

- # Standardized data
  - feature_columns = ['accX', 'accY', 'accZ', 'heartrate']
  - label_columns = ['sleep']
  - df_feature = df[feature_columns] <= standardized data of df_feature
  - df_label = df[label_columns]

4

- # Visualize signals
  - df_feature.plot(), df_label.plot()



8.1(e)

8.2

MODEL PREPARATION

SVM

CONFUSION MATRIX / ACCURACY
PRECISION / RECALL / F1

8.2 (a) SVM Model Training and Testing

1

- **# Train Test Split**

2

- # Model Traing Parameter
  - # Create SVC model
  - c_val = 100 , gmm =0.1, d = 2

3

- # Model initialize
  - svc_lin = SVC(kernel='linear', C=c_val)
  - svc_rbf = SVC(kernel='rbf', C=c_val, gamma=gmm)
  - svc_poly = SVC(kernel='poly', C=c_val, degree = d)

4

- # Model Training
  - svc_rbf_pred = svc_rbf.fit(x_train, y_train)
  - svc_poly = svc_poly.fit(x_train, y_train)

5

- # Model Testing (Predict)
- svc_rbf_pred = svc_rbf.predict(x_test)
- svc_poly_pred = svc_poly.predict(x_test)

8.2 (b) SVM Prediction Report

1

- **# Model Confusion Matrix of SVC_rbf, SVC_poly**
  - Ex
    - confusion_matrix(y_test,svc_rbf_pred))

2

- # Model Classification Report of SVC_rbf, SVC_poly
  - Ex
    - classification_report(y_test,svc_rbf_pred))

3

```
----- Confusion matrix of SVC Rbf -----
[[ 908   73  104   34    9]
 [  73  411  127    0   10]
 [  50   32 1764   98   25]
 [   0    1   79  367    1]
 [   1    2   86    1 1038]]
----- Confusion matrix of SVC Poly -----
[[ 514   84  472   39   19]
 [ 103  295  218    0    5]
 [  64   55 1730   45   75]
 [   2    1  256  188    1]
 [  13    0  127    0  988]]
```

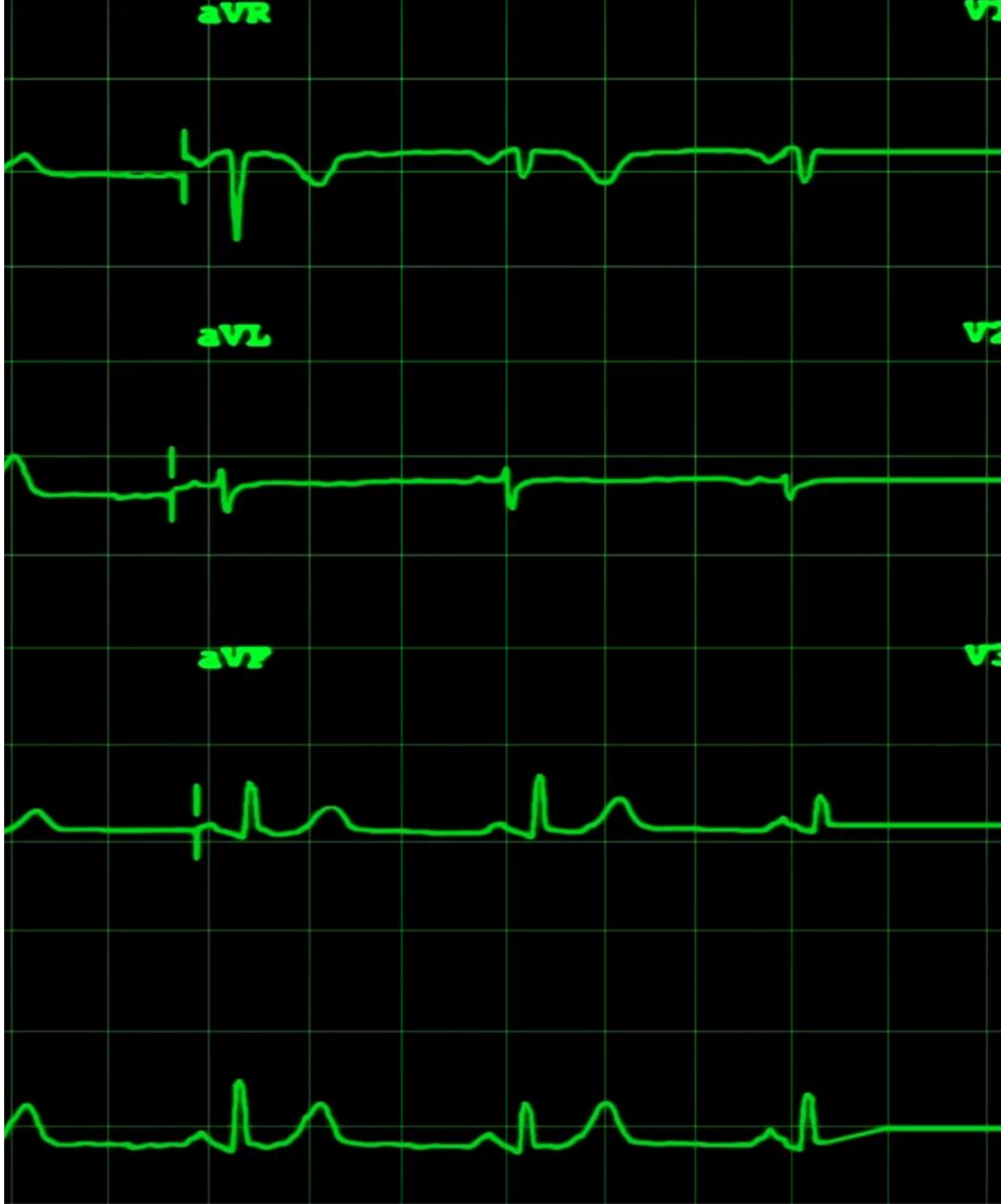
```
----- Classification Report of SVC Rbf -----
              precision    recall  f1-score   support

         0.0       0.88      0.80      0.84      1128
         1.0       0.79      0.66      0.72       621
         2.0       0.82      0.90      0.85      1969
         3.0       0.73      0.82      0.77       448
         5.0       0.96      0.92      0.94      1128

    accuracy                           0.85      5294
   macro avg       0.84      0.82      0.83      5294
weighted avg       0.85      0.85      0.85      5294
```

```
----- Classification Report of SVC Poly -----
              precision    recall  f1-score   support

         0.0       0.74      0.46      0.56      1128
         1.0       0.68      0.48      0.56       621
         2.0       0.62      0.88      0.73      1969
         3.0       0.69      0.42      0.52       448
         5.0       0.91      0.88      0.89      1128

    accuracy                           0.70      5294
   macro avg       0.73      0.62      0.65      5294
weighted avg       0.72      0.70      0.69      5294
```

8.3

HYPERPARAMETER TUNING (GRIDSEARCHCV())

SVM



8.3 Hyperparameter Tuning (GridsearchCV)

1

- #Create Model Parameter Dictionary for SVC
  - C_list = [0.1, 1.0, 10.0, 100.0, 200.0, 500.0]
  - Gamma_list = [0.01, 0.1, 1.0, 10]
  - d_list = [2, 3]

2

- # Perform GridsearchCV() for each classification model
  - grid = GridSearchCV(model, n_jobs, , verbose, scoring = 'accuracy', cv = 2, param_grid)
  - grid_result = grid.fit(x_train, y_train)

3

- # Show best search results
  - Print()
- # Show and Display Mean, std, params
  - Print()
  - Bar()