**WESAD Preprocessing & Exploratory Data Analysis**

This code processes and merges raw WESAD data into a clean format for later machine-learning or statistical analysis. The WESAD dataset contains physiological signals from chest and wrist sensors. The core tasks are:

1. Merging subject data (for multiple participants).
2. Resampling chest-derived labels to match wrist sensor frequencies.
3. Filtering out unwanted data based on specific label and temperature criteria.

**Overview of the Data**

* **Chest Sensors**: ACC (3-axis accelerometer), ECG, EMG, EDA, TEMP, and RESP.
* **Wrist Sensors**: Accelerometer (3-axis), BVP (Blood Volume Pulse), EDA, and TEMP.
* **Sampling Frequencies**:
  + Chest: 700 Hz
  + Wrist ACC: 32 Hz
  + Wrist BVP: 64 Hz
  + Wrist EDA / TEMP: 4 Hz

The dataset also provides **label information** (stress, amusement, meditation, etc.) at the chest sampling rate (700 Hz). In order to align these labels with wrist signals, the label array must be downsampled from 700 Hz to the target sensor frequency.

**Code Structure**

**1. Global Configuration**

* **DATA\_PATH**: Root directory containing the WESAD data folders.
* **chest\_columns**: Column names for the merged chest data.
* **all\_columns**: Column names for a hypothetical combined chest+Wrist data (not necessarily used in final output, but helpful for reference).
* **ids**: Subject IDs present in the WESAD dataset that you want to process.
* **Sampling frequencies**: Defined for each sensor modality (sf\_BVP, sf\_EDA, sf\_ACC, etc.).

**2. map\_chest\_labels(label\_array, output\_length, chest\_sf, target\_sf)**

A helper function that **resamples** the chest labels to match the given sensor’s frequency (i.e., wrist ACC, BVP, EDA/TEMP). Since chest data is at 700 Hz, and wrist sensors can be at 32 Hz, 64 Hz, or 4 Hz, this function uses the **mode** of the label array over the appropriate time slice.

* **Parameters**:
  + label\_array: Original label array at 700 Hz.
  + output\_length: Desired number of labels (e.g., the length of the wrist data).
  + chest\_sf: Chest sampling frequency (700 Hz).
  + target\_sf: Wrist sensor frequency.
* **Returns**:  
  A 1D NumPy array of labels of length output\_length, with each label determined by the most frequent label over the corresponding chest frames.

**3. pkl\_to\_np\_wrist(filename, subject\_id)**

Reads one subject’s **wrist data** and label data (from the .pkl file). It splits the wrist data into three parts:

1. **Accelerometer** (ACC)
2. **Blood Volume Pulse** (BVP)
3. **Electrodermal Activity** (EDA) & **Temperature** (TEMP)

Each signal type has a different sampling frequency, so labels need to be resampled via map\_chest\_labels(...).

* **Parameters**:
  + filename: Path to the subject’s WESAD .pkl file.
  + subject\_id: Numeric ID for the participant.
* **Returns**:  
  Three NumPy arrays:
  + [sid, w\_acc\_x, w\_acc\_y, w\_acc\_z, label]
  + [sid, bvp, label]
  + [sid, w\_eda, w\_temp, label]

**4. merge\_wrist\_data()**

Loops over **all** subject IDs, calling pkl\_to\_np\_wrist(...) for each subject. Then concatenates the arrays from each subject into three large arrays:

1. **Wrist ACC** (across all subjects)
2. **Wrist BVP** (across all subjects)
3. **Wrist EDA/TEMP** (across all subjects)

Finally, it saves these three arrays as pickle files:

* data/subj\_merged\_acc\_w.pkl
* data/subj\_merged\_bvp\_w.pkl
* data/subj\_merged\_eda\_temp\_w.pkl

**5. pkl\_to\_np\_chest(filename, subject\_id)**

Reads one subject’s **chest data** (ACC, ECG, EMG, EDA, TEMP, RESP) and labels, concatenates them into a single array, and returns that.

* **Parameters**:
  + filename: Path to the subject’s WESAD .pkl file.
  + subject\_id: Numeric ID for the participant.
* **Returns**: A 2D NumPy array of shape (N, 10), where each row is [sid, acc1, acc2, acc3, ecg, emg, eda, temp, resp, label].

**6. merge\_chest\_data()**

Similar to merge\_wrist\_data(), but for chest sensors. It iterates over all subject IDs:

1. Calls pkl\_to\_np\_chest(...) for each subject.
2. Concatenates all returned arrays into a single array.
3. Saves the merged data in data/merged\_chest.pkl.

**7. filter\_chest\_data()**

Loads the merged chest data, applies the following filters:

* Keep only rows with label ∈ {1, 2, 3}.
* Keep only rows with temp > 0 (removing outliers or bad measurements).

Finally, it saves the filtered dataset as data/merged\_chest\_fltr.pkl.

**8. preprocess()**

A convenience function that **runs all** the steps in sequence:

1. merge\_wrist\_data()
2. merge\_chest\_data()
3. filter\_chest\_data()

When you call preprocess(), it executes the entire pipeline in the correct order.

**Usage**

1. **Install Requirements**  
   Make sure you have installed pandas, numpy, scipy, and matplotlib (if you plan to visualize).
2. **Edit Paths**  
   Update DATA\_PATH in the code if your WESAD data is in a different directory.
3. **Run the Pipeline**

python preprocessing.py

1. **Output**
   * Merged wrist data in three pickle files:  
     subj\_merged\_acc\_w.pkl, subj\_merged\_bvp\_w.pkl, subj\_merged\_eda\_temp\_w.pkl
   * Merged chest data: merged\_chest.pkl
   * Filtered chest data: merged\_chest\_fltr.pkl

**Notes & Considerations**

* **Labeling**: The original labels are provided at the chest sampling frequency (700 Hz). By taking the mode over each time window, we assume that within a small window, the subject remains in the same label/state.
* **Filtering**: The choices to keep labels {1, 2, 3} and temp > 0 are based on the typical usage in WESAD (1 = baseline, 2 = stress, 3 = amusement, etc.) and to remove corrupted temperature data.
* **Further Processing**: You may still need to perform additional preprocessing (e.g., normalization, artifact removal, segmentation) depending on your downstream task.

By following this pipeline, you’ll have clean, consistently-labeled chest and wrist data ready for machine learning or statistical analysis.