Sleep Stage Classification from Wrist ACC + Heart Rate (PhysioNet Sleep-Accel / Apple Watch)

Chouaib (github.com/ChouaibB)

January 27, 2026

Contents

1. Data source: Sleep-Accel (PhysioNet; Apple Watch + PSG) . . . 2
2. Data loading and check . . . 3
3. Epoch alignment (30s) — assign raw samples to PSG epochs . . . 4
4. Feature extraction . . . 6
   4.1 Minimal per-epoch feature summaries (ACC magnitude + HR) . . . 6
   4.2 Per-subject missingness (% of epochs) . . . 7
   4.4 Causal history features (past-only rolling summaries) . . . 7
   4.5 Feature relevance and visualization . . . 8
5. Modeling . . . 10
6. Results & reporting . . . 11
   6.1 Performance summary . . . 12
   6.2 Confusion matrices . . . 13

1. Data source: Sleep-Accel (PhysioNet; Apple Watch + PSG)

Dataset: *Motion and heart rate from a wrist-worn wearable and labeled sleep from polysomnography* (PhysioNet, v1.0.0)

- **PhysioNet dataset page (download + description):**
  - https://physionet.org/content/sleep-accel/1.0.0/
- **DOI (v1.0.0):**
  - https://doi.org/10.13026/hmhs-py35
- **Local path (expected):** download + unzip into `./data/sleep_accel/` (the `data/` directory is not committed to git)
  - expected: `heart_rate/`, `motion/`, `labels/`, `steps/`, plus `LICENSE.txt`
- **License (for files):** Open Data Commons Attribution License v1.0 (**ODC-By 1.0**)

Citations (as requested by PhysioNet):
- Walch, O. (2019). *Motion and heart rate from a wrist-worn wearable and labeled sleep from polysomnography* (version 1.0.0). PhysioNet. https://doi.org/10.13026/hmhs-py35
- Walch, O., Huang, Y., Forger, D., Goldstein, C. (2019). *Sleep stage prediction with raw acceleration and photoplethysmography heart rate data derived from a consumer wearable device.* SLEEP. https://doi.org/10.1093/sleep/zsz180
- Goldberger, A. L., et al. (2000). *PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals.* Circulation.

Files used(and expected columns)

We will use the following folders/files from the PhysioNet release:

- **Motion (ACC):** `motion/[subject]_acceleration.txt`
  Columns per line: `t_sec, ax_g, ay_g, az_g`
  where `t_sec` is seconds since PSG start, and accelerations are in **g**.
- **Heart rate (HR):** `heart_rate/[subject]_heartrate.txt`
  Columns per line: `t_sec, hr_bpm`
  where `hr_bpm` is heart rate in **beats per minute**.
- **PSG sleep labels:** `labels/[subject]_labeled_sleep.txt`
  Columns per line: `t_sec, stage` with stage codes:
  `Wake=0, N1=1, N2=2, N3=3, REM=5` (we drop unscored/invalid epochs if present).

  Note: The dataset also includes `steps/` files, but we won't use them in the first version.

Notebook intention

Goal: build a **reproducible sleep-staging pipeline** from **wrist ACC + HR** aligned to **PSG-scored 30-second epochs**, with **leakage-aware, subject-wise evaluation**.

What we do:

1. **Define the modeling unit as the PSG epoch (30s)** and build one feature row per epoch.

2. **Align** wrist **ACC** and **HR** to each labeled 30s epoch (aggregate samples falling in `[t, t+30s)`), and attach the PSG stage label at `t`.

3. **Extract simple, readable features** per epoch:
   - ACC: magnitude and axis statistics + activity intensity proxies
   - HR: summary statistics + missingness indicators

4. Add **causal context (history) features** using past-only rolling summaries over recent epochs (e.g., last few minutes) to capture local sleep continuity without using future information.

5. Train and compare a small set of classical models using **subject-wise cross-validation** (GroupKFold) and report robust staging metrics (macro-F1, balanced accuracy, confusion matrices, per-subject performance).

Note on extra "pre-PSG" wearable data:

This dataset includes wearable streams that may start **before PSG time zero** (e.g., steps for days prior, HR for hours prior, motion shortly before). For the main staging pipeline we **restrict to the PSG-labeled interval** and only aggregate sensor data within labeled 30s epochs. Pre-PSG data can be used in extensions as **subject-level context** (e.g., prior-days activity summaries computed strictly from `t < 0`), but coverage varies across subjects and adds preprocessing complexity.

2. Data loading and check

We load per-subject **wrist accelerometer (ACC)**, **heart rate (HR)**, and **PSG sleep labels** from the PhysioNet Sleep-Accel folder structure.

Key conventions: - All timestamps are de-identified and expressed as **seconds since PSG start** (`t=0` is the PSG start time). - For the main staging pipeline, we **restrict to the PSG-labeled interval** and only aggregate sensor samples that fall inside each labeled 30s epoch.

```
Found 31 subjects (from labels/). Example IDs: ['1066528', '1360686', '1449548',
'1455390', '1818471']

Loaded ALL data:
  labels: (27211, 3) | subjects=31
  hr    : (254425, 3)     | subjects=31
  acc   : (51819151, 5)    | subjects=31

Example subject 1066528:
  labels_ex: (952, 3) | t range: [0.0, 28530.0]
  hr_ex    : (16617, 3)     | t range: [-355241.7, 34491.2]
  acc_ex   : (1281000, 5)    | t range: [-21684.8, 28626.5]
```

	stage_code	stage_name	count
0	0	Wake	185
1	1	N1	97
2	2	N2	299

```
3           3         N3     62
4           5        REM    309
```

3. Epoch alignment (30s) — assign raw samples to PSG epochs

We use PSG labels as the **epoch grid** (30s cadence), and assign raw HR and ACC samples to each epoch.

Output: one row per epoch with:
- `epoch_id`, `t_start_sec`, `t_end_sec`, `stage_code`
- raw per-epoch sequences: HR (`t_sec`, `hr_bpm`) and ACC (`t_sec`, `ax_g`, `ay_g`, `az_g`)

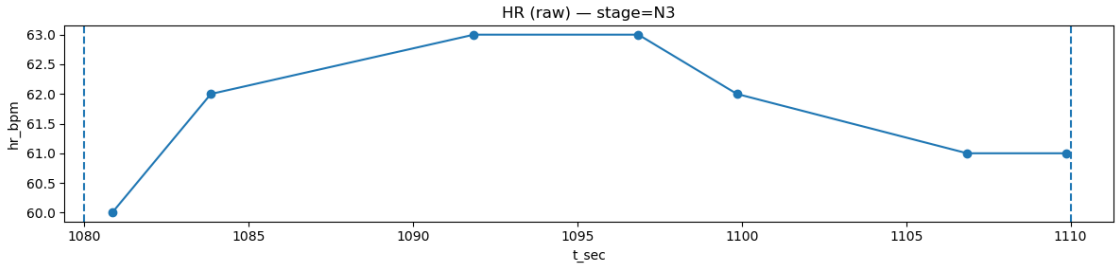
Note on sampling rates and interpolation:
ACC is high-rate, HR is sparse/irregular, and labels are per 30s epoch. We **do not resample or interpolate** signals to a common grid. Instead, we compute **per-epoch summary features** (e.g. counts, mean/std/percentiles, simple trends) directly from the raw samples assigned to each epoch at the next section.

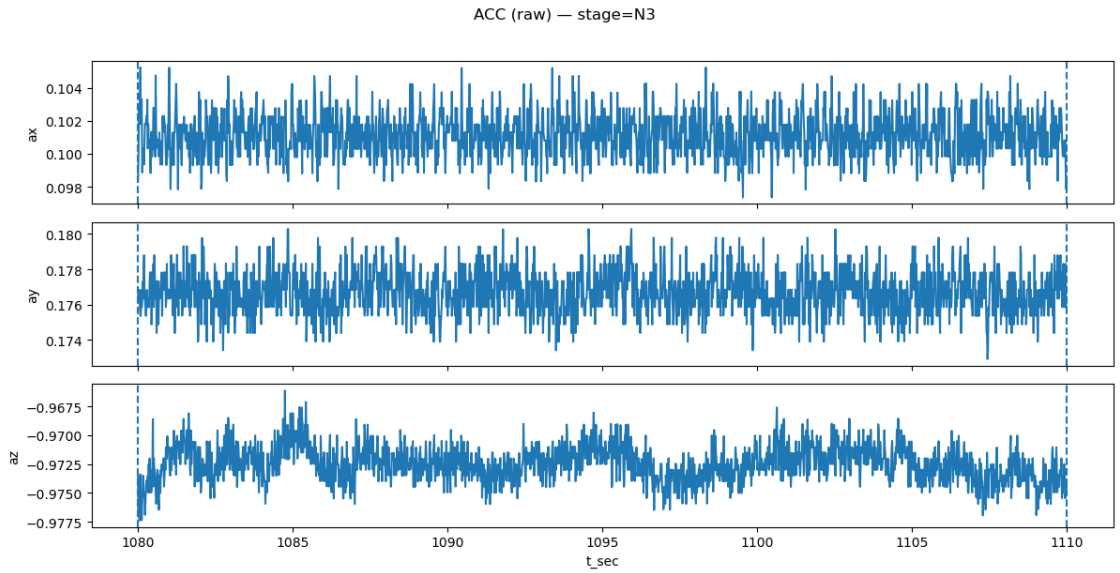
`epoch_raw: (26417, 11)`

```
  subject_id  t_start_sec  stage_code  t_end_sec  epoch_id  \
0    1066528          0.0           0       30.0         0
1    3509524          0.0           0       30.0         1
2    4018081          0.0           0       30.0         2
3    4426783          0.0           0       30.0         3
4    5132496          0.0           0       30.0         4

                                              hr_t_sec  \
0  [6.38561010361, 6.38561010361, 6.38561010361, …
1  [2.97270989418, 6.97270989418, 11.9727399349, …
2  [1.20449995995, 5.2044699192, 9.2044699192, 18…
3  [3.28572010994, 7.28574991226, 13.2857499123, …
4  [6.48358011246, 8.48358011246, 12.4835801125, …

                                                hr_bpm  \
0  [52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 51.0, 51…
1         [82.0, 84.0, 86.0, 86.0, 86.0, 83.0, 76.0]
2               [69.0, 66.0, 62.0, 57.0, 58.0, 61.0]
3               [71.0, 71.0, 72.0, 61.0, 60.0, 60.0]
4                [113.0, 115.0, 112.0, 117.0, 113.0]

                                             acc_t_sec  \
0  [0.0159480571747, 0.0360059738159, 0.055885076…
1  [0.00270414352417, 0.0179829597473, 0.03272795…
2  [0.0129339694977, 0.0286469459534, 0.042945861…
3  [0.0104320049286, 0.0303421020508, 0.050279140…
4  [0.0128321647644, 0.0269389152527, 0.042412996…

                                               acc_x  \
0  [0.4039307, 0.4039154, 0.4049072, 0.4083557, 0…
```

```
1  [0.0323334, 0.0274353, 0.02388, 0.0252228, 0.0…
2  [-0.4547424, -0.4547424, -0.455246, -0.4581757…
3  [-0.7835388, -0.7860413, -0.7800598, -0.783889…
4  [-0.1996918, -0.4198914, -0.3826447, -0.204345…

                                                acc_y  \
0  [0.4490051, 0.4480286, 0.4465485, 0.447525, 0…
1  [-0.4576111, -0.4629974, -0.4717712, -0.465408…
2  [0.5541077, 0.5541077, 0.5560608, 0.5536194, 0…
3  [-0.21698, -0.2355804, -0.2243347, -0.2145691,…
4  [0.3026581, 0.1375885, 0.0600433, 0.2941742, 0…

                                                acc_z
0  [-0.7968597, -0.7953949, -0.7958527, -0.796768…
1  [-0.8885193, -0.8871613, -0.8794403, -0.874557…
2  [-0.6973724, -0.6978607, -0.6973877, -0.696945…
3  [-0.5306091, -0.5212555, -0.5314789, -0.546798…
4  [-0.9453125, -0.8695831, -0.9243622, -0.813476…
```

Example plot for one epoch: subject_id=759667 | epoch_id=756 | window=[1080.0,1110.0) | stage=N3





4. Feature extraction

We convert each PSG-aligned 30-second epoch into a single feature vector.
To keep the notebook **simple and robust** (and to avoid a "feature zoo", especially once we add **causal history features**), we use a **small, standard set of summary statistics**.

Signals and features

Accelerometer (ACC):
- We collapse tri-axial ACC into **magnitude** `|a| = sqrt(ax² + ay² + az²)`.
- Per epoch we compute: `count`, `mean`, `std`, `median`, `IQR`, `p10`, `p90`, `min`, `max` - Dynamic proxy `std(diff(|a|))` (often useful for "movement bursts")

Heart rate (HR):
- Per epoch we compute: `count`, `mean`, `std`, `IQR`, `min`, `max`

Time
- `time_since_start_hours` (simple time context within the night)

Note on sampling and reliability ACC is high-rate while HR is sparse/irregular; labels are per 30s epoch. We **do not resample or interpolate**.
Summary statistics are computed directly from the raw samples assigned to each epoch, and `count` acts as a lightweight **signal reliability** indicator (few/no samples → less reliable features).

4.1 Minimal per-epoch feature summaries (ACC magnitude + HR)

```
epoch_features shape: (26417, 22)
```

	subject_id	epoch_id	t_start_sec	t_end_sec	stage_code	\
0	1066528	0	0.0	30.0	0	
1	3509524	1	0.0	30.0	0	
2	4018081	2	0.0	30.0	0	
3	4426783	3	0.0	30.0	0	
4	5132496	4	0.0	30.0	0	

	time_since_start_hours	hr_mean	hr_std	hr_iqr	hr_min	…	\
0	0.0	52.166667	0.897527	0.00	51.0	…	
1	0.0	83.285714	3.325842	3.50	76.0	…	
2	0.0	62.166667	4.219663	6.25	57.0	…	
3	0.0	65.833333	5.520165	10.75	60.0	…	
4	0.0	114.000000	1.788854	2.00	112.0	…	

	acc_mag_count	acc_mag_mean	acc_mag_std	acc_mag_median	acc_mag_iqr	\
0	1500	1.000029	0.001947	1.000037	0.002023	
1	1848	0.993591	0.034502	0.992176	0.010523	
2	1918	0.999597	0.001879	0.999510	0.002095	
3	1716	1.000140	0.081830	0.991798	0.012816	
4	1861	1.019645	0.174913	0.998606	0.020979	

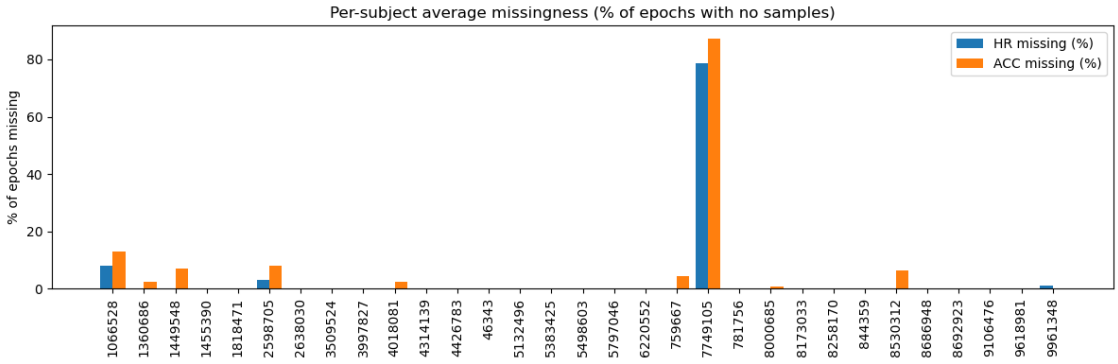
	acc_mag_p10	acc_mag_p90	acc_mag_min	acc_mag_max	acc_mag_diff_std
0	0.997970	1.002068	0.977874	1.017645	0.002779
1	0.967860	1.019650	0.780282	1.235724	0.019853
2	0.997544	1.001872	0.990996	1.008555	0.001812
3	0.972162	1.025978	0.512724	2.283005	0.063535
4	0.932608	1.136894	0.288624	3.711552	0.122539

[5 rows x 22 columns]

4.2 Per-subject missingness (% of epochs)

We quantify how often each signal is missing at the epoch level: - **HR missing**: `hr_count == 0` - **ACC missing**: `acc_mag_count == 0`

We plot the **percentage of missing epochs per subject** for HR and ACC to spot subjects with low signal coverage.



4.4 Causal history features (past-only rolling summaries)

Sleep is a continuous process, so we add lightweight **history context** to each epoch using only **past information**.

For each subject, we compute histiry (rolling means) for the main **level** and **variability** signals **`hr_mean`, `hr_std`, `acc_mag_mean`, `acc_mag_std`**, over the previous:
- **5 epochs** (~2.5 minutes)
- **20 epochs** (~10 minutes)

To keep this strictly causal, we use `shift(1)` so the rolling window for epoch t uses epochs `< t` only.

```
Added history features: ['hr_mean_roll5', 'hr_std_roll5', 'acc_mag_mean_roll5',
'acc_mag_std_roll5', 'hr_mean_roll20', 'hr_std_roll20', 'acc_mag_mean_roll20',
'acc_mag_std_roll20']
epoch_features shape with history features: (26417, 30)
```

```
  subject_id  epoch_id  t_start_sec  t_end_sec  stage_code  \
0    1066528         0          0.0       30.0           0
1    1066528        11         30.0       60.0           0
2    1066528        22         60.0       90.0           0
3    1066528        33         90.0      120.0           0
4    1066528        44        120.0      150.0           0

   time_since_start_hours    hr_mean    hr_std  hr_iqr  hr_min  …  \
0                0.000000  52.166667  0.897527     0.0    51.0  …
1                0.008333  50.833333  1.950783     4.0    49.0  …
2                0.016667  52.400000  1.356466     1.0    50.0  …
3                0.025000  53.833333  0.372678     0.0    53.0  …
4                0.033333  60.500000  4.500000     8.0    56.0  …

   acc_mag_max  acc_mag_diff_std  hr_mean_roll5  hr_std_roll5  \
0     1.017645          0.002779            NaN           NaN
1     1.004546          0.001883      52.166667      0.897527
2     1.005295          0.001951      51.500000      1.424155
3     1.008659          0.002374      51.800000      1.401592
4     1.006992          0.002160      52.308333      1.144364

   acc_mag_mean_roll5  acc_mag_std_roll5  hr_mean_roll20  hr_std_roll20  \
0                 NaN                NaN             NaN            NaN
1            1.000029           0.001947       52.166667       0.897527
2            1.000041           0.001725       51.500000       1.424155
3            1.000067           0.001666       51.800000       1.401592
4            1.000057           0.001751       52.308333       1.144364

   acc_mag_mean_roll20  acc_mag_std_roll20
0                  NaN                 NaN
1             1.000029            0.001947
2             1.000041            0.001725
3             1.000067            0.001666
4             1.000057            0.001751

[5 rows x 30 columns]
```

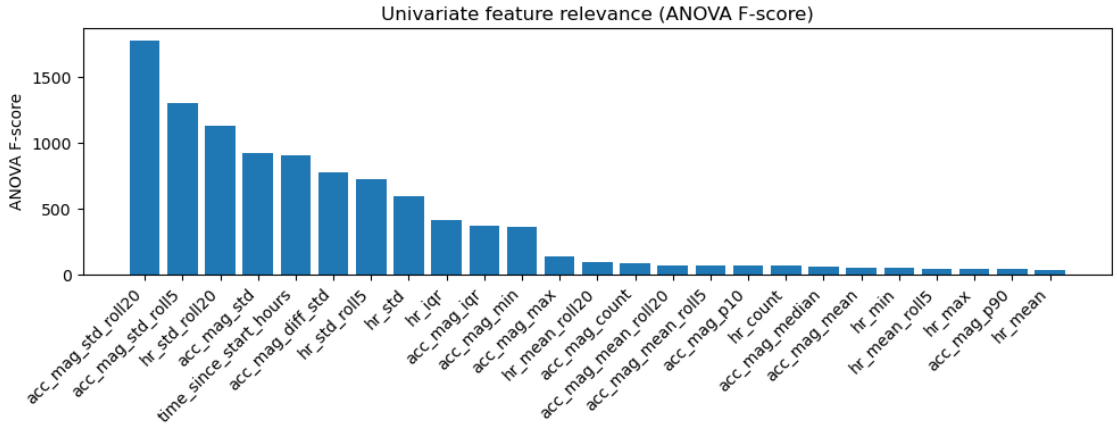
4.5 Feature relevance and visualization

Before modeling, we do two quick sanity checks:

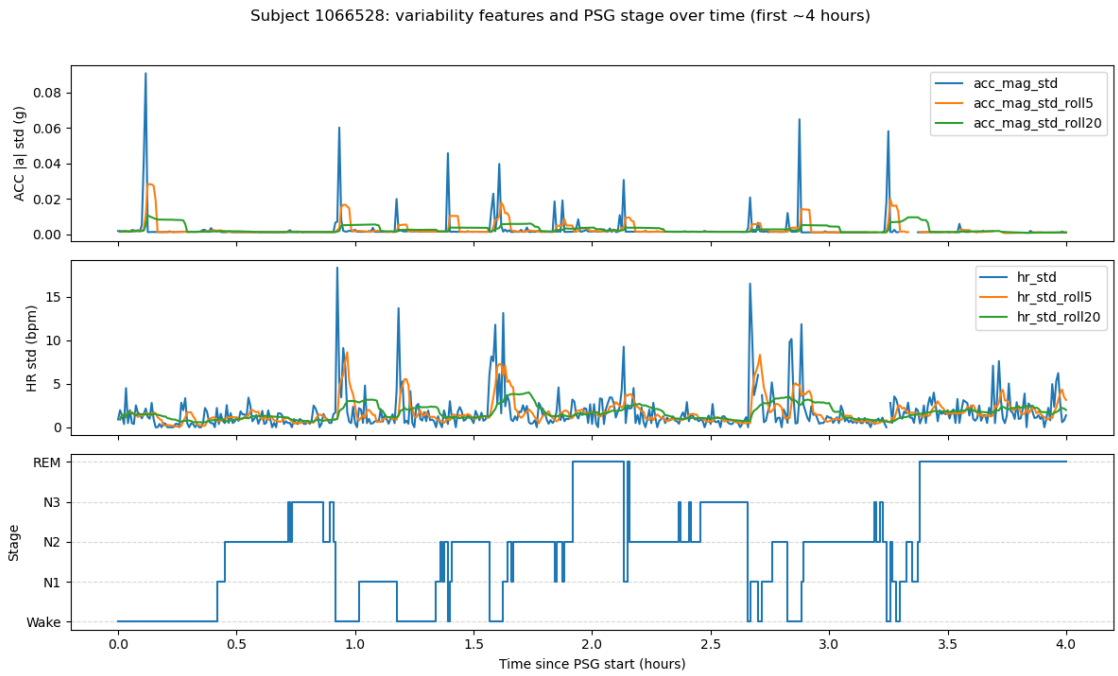
1) **Univariate feature relevance (ANOVA F-score)**: how strongly each feature differs across sleep stages (one feature at a time).

2) **Single-subject timeline**: how a couple of key features evolve over time alongside the PSG stage labels.

These visualizations are descriptive (not "proof" of separability) and help validate the pipeline before ML.



	feature	F_score	p_value
24	acc_mag_std_roll20	1781.548135	0.000000e+00
23	acc_mag_std_roll5	1300.868806	0.000000e+00
20	hr_std_roll20	1136.074761	0.000000e+00
7	acc_mag_std	926.236595	0.000000e+00
16	time_since_start_hours	909.576551	0.000000e+00
14	acc_mag_diff_std	781.981964	0.000000e+00
19	hr_std_roll5	727.759319	0.000000e+00
1	hr_std	595.518177	0.000000e+00
2	hr_iqr	420.137748	0.000000e+00
9	acc_mag_iqr	375.699593	2.950669e-315
12	acc_mag_min	366.777206	6.272101e-308
13	acc_mag_max	144.082104	4.462149e-122
18	hr_mean_roll20	98.912676	1.034501e-83
15	acc_mag_count	93.162596	8.167400e-79
22	acc_mag_mean_roll20	76.561146	1.162256e-64
21	acc_mag_mean_roll5	75.370355	1.205480e-63
10	acc_mag_p10	73.628140	3.694619e-62
5	hr_count	73.305709	6.961063e-62
8	acc_mag_median	59.866863	2.053812e-50
6	acc_mag_mean	57.542436	1.980793e-48
3	hr_min	52.692967	2.732684e-44
17	hr_mean_roll5	50.168934	3.899823e-42
4	hr_max	50.130534	4.205529e-42
11	acc_mag_p90	42.451121	1.502108e-35
0	hr_mean	41.436253	1.102355e-34



5. Modeling

We evaluate two label granularities using **subject-wise 5-fold GroupKFold** (leakage-aware sleep staging using the PSG-aligned epoch table):

Target
- **5-class:** Wake / N1 / N2 / N3 / REM
- **3-class:** Wake / NREM (N1+N2+N3) / REM

Features per-epoch summaries + causal history features (past-only rolling means).

Validation subject-wise 5-fold GroupKFold (`group = subject_id`) to avoid subject leakage.

Models (no tuning):
- **Logistic Regression (L2, class_weight="balanced")** (baseline)
- **HistGradientBoostingClassifier** (strong classical baseline)

We also inspect the **label distribution** to understand class imbalance.

```
5-class | CV: LR (L2, balanced): 100%|
                                    | 5/5
[00:03<00:00,  1.37it/s]
5-class | CV: HGB: 100%|
                                    | 5/5
[00:06<00:00,  1.24s/it]

5-class | Test-fold label distribution (% within each test fold):
```

```
   fold   Wake    N1     N2     N3    REM  n_test_epochs  n_test_subjects
0     0   9.82  6.70  47.28  13.97  22.22           5854                7
1     1   8.98  6.72  46.31  14.37  23.62           5254                6
2     2   8.23  6.59  48.17  14.06  22.94           5248                6
3     3   6.70  8.98  52.37   8.43  23.52           5043                6
4     4  12.20  5.52  51.49  11.82  18.97           5018                6

5-class | CV summary (mean ± std over folds, in %):

       run              model  macro_f1_mean  macro_f1_std  bal_acc_mean  \
1  5-class  LR (L2, balanced)          44.19          4.64         50.96
0  5-class                HGB          43.62          4.54         42.40

   bal_acc_std
1         5.59
0         4.76

3-class | CV: LR (L2, balanced): 100%|
                                       | 5/5
[00:00<00:00,  5.75it/s]
3-class | CV: HGB: 100%|
                                       | 5/5
[00:03<00:00,  1.26it/s]

3-class | Test-fold label distribution (% within each test fold):


   fold   Wake   NREM    REM  n_test_epochs  n_test_subjects
0     0   9.82  67.95  22.22           5854                7
1     1   8.98  67.40  23.62           5254                6
2     2   8.23  68.83  22.94           5248                6
3     3   6.70  69.78  23.52           5043                6
4     4  12.20  68.83  18.97           5018                6

3-class | CV summary (mean ± std over folds, in %):

       run              model  macro_f1_mean  macro_f1_std  bal_acc_mean  \
0  3-class                HGB          55.71          4.60         54.12
1  3-class  LR (L2, balanced)          54.73          4.89         61.06

   bal_acc_std
0         4.37
1         6.75
```

6. Results & reporting

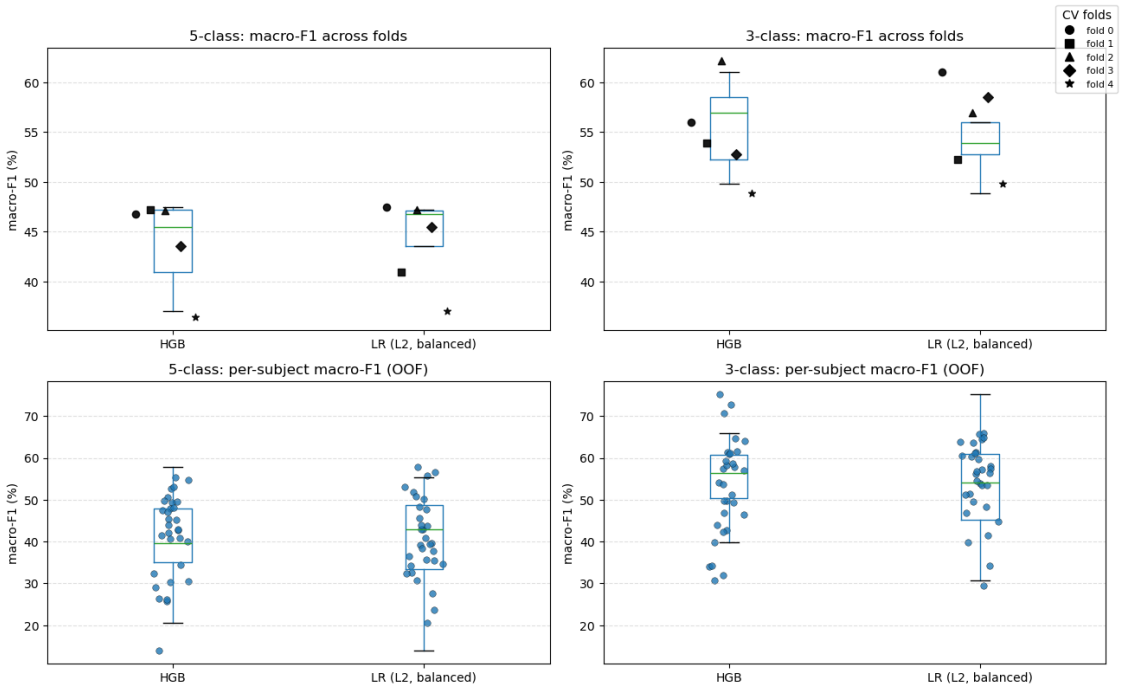
This section analyzes the **out-of-fold (OOF)** predictions produced by the subject-wise CV loop.

We report results for: - **5-class staging** (Wake / N1 / N2 / N3 / REM) - **3-class staging** (Wake / NREM / REM)

6.1 Performance summary

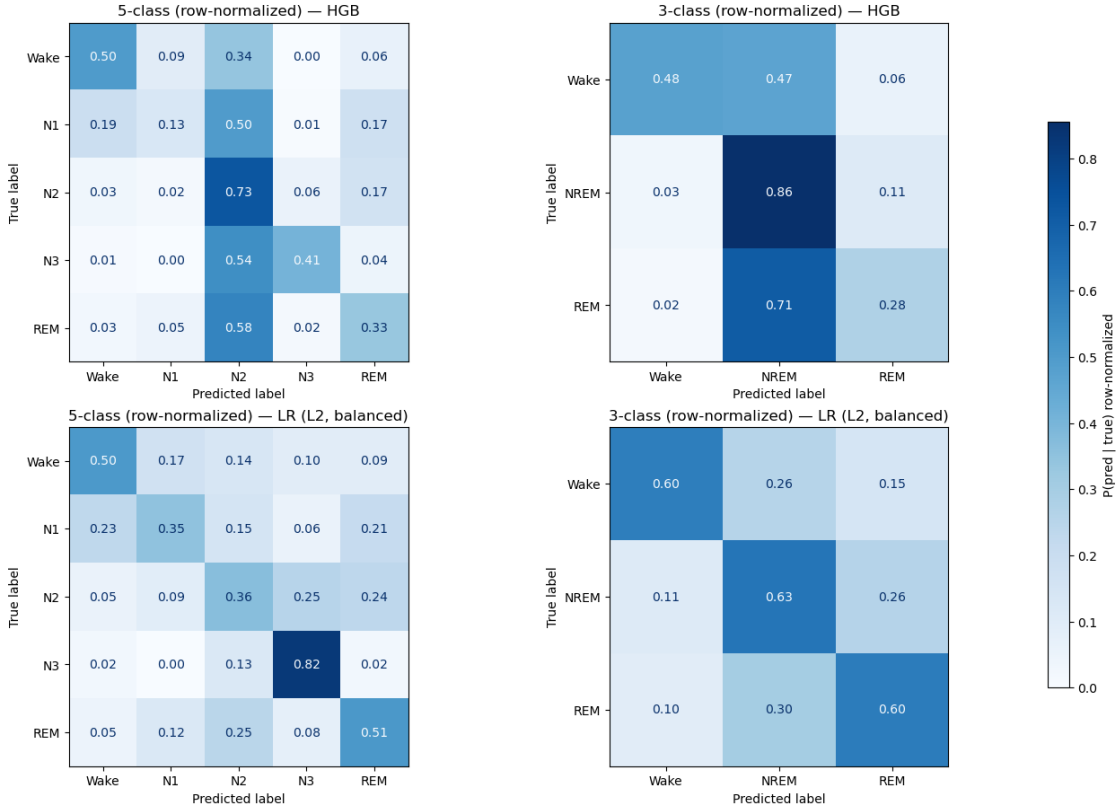
We summarize performance as:

- **Across folds:** macro-F1 variability across the 5 subject-wise folds.
- **Across subjects:** per-subject macro-F1 computed from out-of-fold predictions.



- **3-class (Wake/NREM/REM) is substantially easier than 5-class**, with macro-F1 shifting up for both models.
- **HGB is slightly better and more consistent across folds**, especially for 3-class; LR remains competitive.
- **Fold-to-fold variance is moderate** (a few harder folds appear as outliers), suggesting reasonable stability under subject-wise CV.
- **Per-subject variance is large** in both settings: some subjects are consistently much harder/easier, which is typical in wearable sleep due to signal quality and individual differences.
- **Takeaway:** for a wearable-style pipeline, **3-class staging** is a strong, realistic target; **5-class staging** remains challenging with lightweight wrist ACC/HR features.

6.2 Confusion matrices



- **5-class is difficult:** most errors happen between neighboring sleep stages. In particular, **N1 is rarely recognized cleanly** and is often confused with **N2** (and sometimes Wake/REM).
- **N2 dominates the 5-class predictions:** both models tend to "collapse" uncertain epochs into **N2**, and **N3/REM** are frequently pulled toward N2 as well (more so for HGB here).
- **3-class is much clearer:** the main diagonal is stronger for both models, confirming that wrist signals separate **Wake vs NREM vs REM** better than fine-grained staging.
- **Key remaining confusion in 3-class: REM NREM** is the hardest boundary (especially for HGB), while LR appears **more balanced** between NREM and REM here.
- **Overall:** these matrices explain why **macro-F1 improves a lot** when moving from 5-class to 3-class: fewer ambiguous boundaries and less reliance on N1/N2/N3 separation.