

## Scaler–Sampler–Cost Interactions in Classical Sleep Staging: Utility-Based Trade-offs for N1 on Sleep-EDF

Ahmet Sertol Köksal\*<sup>1</sup>

<sup>1</sup>Department of Computer Engineering, Yozgat Bozok University, Turkey

\*(ahmet.koksal@bozok.edu.tr) Email of the corresponding author

(Received: 05 February 2026, Accepted: 15 February 2026)

(7th International Conference on Engineering, Natural and Social Sciences ICENSOS 2026, February 06-07, 2026)

**ATIF/REFERENCE:** Köksal, A. S. (2026). Scaler–Sampler–Cost Interactions in Classical Sleep Staging: Utility-Based Trade-offs for N1 on Sleep-EDF, *International Journal of Advanced Natural Sciences and Engineering Researches*, 10(2), 61-73.

**Abstract** – Automated sleep staging is still difficult. Because of its transitional nature and overlap with adjacent stages, stage N1 shows the lowest reliability. In this work, we examine whether N1 performance under leakage-safe evaluation can be enhanced by cost-sensitive learning. We also look at how it affects SMOTE-family resampling and feature scaling. We use a multilayer perceptron with five stages and 30-s epochs from Sleep-EDF Expanded dataset. 28 handcrafted features from Fpz-Cz EEG channel are used to train the model. Three scalars— z-score, robust, and min-max—are assessed. Additionally, we assess three cost formulations (Inverse-frequency, Effective-number, and Log-scaled) and three resampling techniques (SVMSMOTE, SMOTETOMEK, and SMOTE with AllKNN). Every experiment is carried out using subject-wise 5-fold cross-validation. The best-performing global option, according to the results, is inverse-frequency sample weighting without resampling. Compared to SMOTE-based methods, it achieves competitive N1 improvements. On the other hand, SVM-based resampling with effective-number weighting is most advantageous for min-max scaling. Additionally, a within-cell added-value analysis indicates that cost-sensitive learning yields its greatest benefits when resampling is not used. Once SMOTE-family sampling is used, it provides little additional benefit. Reducing misclassifications into W/N2/REM is the main source of N1 improvements, according to confusion-matrix inspection. All things considered, the study offers useful advice for choosing N1-oriented configurations. Under realistic, leakage-safe evaluation, it also takes into consideration trade-offs in non-target stages.

*Keywords* – Automated sleep staging, Cost-sensitive learning, Feature scaling, N1 detection, SMOTE, Subject-wise CV

### I. INTRODUCTION

Sleep staging via polysomnography (PSG) is a core component of clinical sleep assessment and is widely used in sleep research. To reduce reliance on full PSG, automated methods based on EEG-derived features—often using a single channel—have improved; however, robust performance across subjects and stages remains difficult. Among sleep stages, N1 is often the hardest to classify because it lies at the boundary between wakefulness and N2. In addition, sleep-stage distributions are typically imbalanced, and N1 is frequently the least represented stage in many datasets [1–5].

A key methodological concern in sleep staging research is evaluation realism. Specifically, when samples from the same subject are included in both training and testing splits, cross-subject generalization

may be exaggerated. Models can exploit subject-specific signatures, leading to inflated performance estimates that do not generalize to unseen subjects. For leakage-safe evaluation and clinically meaningful generalization, subject-wise cross-validation is therefore crucial [6–8].

Even under leakage-safe protocols, improving N1 sensitivity (recall) or the N1-F1-score can reduce performance on other stages, most notably wake (W) and N2. Many studies report a single aggregate metric (e.g., overall accuracy) or focus on a single targeted metric (e.g., N1 F1) without quantifying how the remaining classes are affected. Failing to report these class-wise trade-offs is problematic for applications that require stage-specific reliability, such as detecting micro-arousals, transitional sleep, or sleep fragmentation patterns [9–11].

Rather than transition- or sequence-aware modeling, we concentrate on imbalance-aware elements in classical sleep-staging pipelines and their interactions in this work. Resampling and cost-sensitive learning are two popular methods to address class imbalance and enhance minority-stage performance. Random oversampling and synthetic sample generation, like SMOTE, are examples of resampling. These techniques are sometimes combined with cleaning procedures to reduce label noise and class overlap. Through class- or sample-weighting schemes, cost-sensitive learning increases the importance of minority classes. Although these approaches are frequently examined independently, their combined impact may vary depending on preprocessing choices [12,13].

The selection of feature scaling is another aspect of these pipelines that receives little attention. Scaling modifies neighborhood geometry and, consequently, the synthetic sample distribution for distance-based resampling techniques like SMOTE. Scaling also affects optimization and the relative importance of feature dimensions for gradient-based classifiers like multilayer perceptrons (MLPs). Despite this, scaling decisions are not routinely assessed in conjunction with resampling and weighting in many empirical studies [14–16].

Inspired by these gaps, we build on our earlier work [17], in which we used leakage-safe subject-wise cross-validation to systematically analyze scaler  $\times$  oversampling interactions on Sleep-EDF (Sleep Cassette). That study showed that SMOTE-based oversampling can increase N1-F1, but the magnitude and cost of that gain—especially decreases in W and N2 performance—were highly sensitive to the scaler choice. To enable principled selection of trade-offs, we also introduced a utility-style summary that jointly captures N1 improvement and the cost paid by other classes. Although [17] focused on resampling and scaling, it remains unclear whether cost-sensitive learning can improve N1 without the same trade-offs, and whether combining cost-sensitive learning with the best scaler–sampler settings reduces the overall cost.

In this work, we examine cost-sensitive learning for sleep staging using the same dataset and leakage-safe experimental setup as in [17]. We compare three weighting formulations using a classical MLP with handcrafted EEG features to determine whether cost-sensitive learning, either by itself or in conjunction with specific SMOTE-based samplers, produces a better N1-versus-cost trade-off than resampling alone [18,19].

We report overall performance (accuracy, macro-F1, Cohen’s kappa) along with class-wise F1 scores to make these trade-offs clear. We also use confusion matrices to examine stage-specific shifts. To highlight useful “best practice” configurations for different objectives (e.g., prioritizing N1 vs. preserving overall agreement), we apply a utility-based ranking with multiple baselines [20,21].

Our contributions are threefold:

- Using subject-wise 5-fold cross-validation, we provide a leakage-safe, controlled assessment of cost formulations intended to enhance N1 on Sleep-EDF.
- Beyond SMOTE-based oversampling, we examine the scaler  $\times$  sampler  $\times$  cost interaction and elucidate when cost-sensitive learning is helpful (or not).

To help choose configurations that increase N1 while minimizing degradation in other stages, we introduce utility-based “cost of improvement” reporting.

## II. MATERIALS AND METHODS

### A. Dataset

All experiments were conducted on the Sleep-EDF Expanded (Sleep Cassette) database [22]. It contains recordings from 78 individuals: 75 subjects contributed two nights each, while 3 subjects contributed one night. EEG signals are provided as 30-s epochs and sampled at 100 Hz. Sleep stages in Sleep-EDF were originally scored according to the Rechtschaffen & Kales (R&K) standard [23]. To align with current practice, we mapped the annotations to AASM-style labels [24], merging N3 and N4 into a single N3 class. Epochs labeled “?” and “M” (when present) were excluded. Moreover, to reduce the influence of prolonged wake periods—typically around sleep onset and morning awakening—we limited W to the first and last 30 minutes of each recording. The resulting post-processing class composition is shown in Fig. 1. Sleep-EDF provides two EEG channels. We used Fpz–Cz throughout, as it is the most commonly adopted channel in the Sleep-EDF literature.

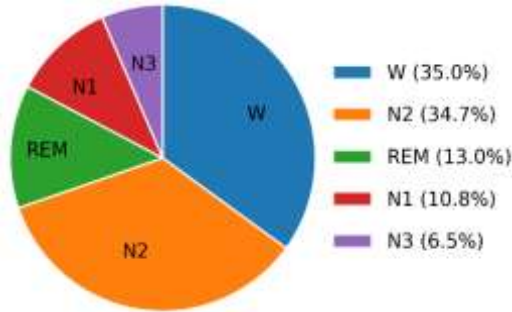


Fig. 1. Post-processing class distribution of Sleep-EDF under AASM labels.

### B. Hand-crafted Feature Representation

We use a small set of hand-crafted descriptors to represent each 30-s epoch rather than feeding raw EEG into a sequence model. This design facilitates feature-level interpretability and keeps the pipeline compatible with classical machine-learning classifiers [25].

No additional bandpass filtering, notch filtering, or signal resampling was applied, since Sleep-EDF is already provided at 100 Hz. Features were computed from non-overlapping 30-s epochs using a rectangular window. For each epoch, we calculate a 28-dimensional feature vector from the Fpz–Cz channel. The feature set includes (i) distributional/statistical descriptors (mean, median, standard deviation, variance, minimum, maximum, peak-to-peak amplitude, skewness, kurtosis, mean absolute deviation, and signal energy); (ii) time-domain dynamics and complexity measures (Hjorth activity, mobility, complexity; Petrosian and Higuchi fractal dimensions; detrended fluctuation analysis; and Lempel–Ziv complexity); (iii) spectral features (total power, band powers in delta/theta/alpha/beta/gamma, median frequency, and peak frequency); and (iv) time–frequency descriptors derived from wavelets (wavelet energy and wavelet entropy). This fixed representation is used as the model input in all experiments.

### C. Leakage-Safe Evaluation Protocol

To estimate cross-subject generalization under realistic conditions and to prevent optimistic bias caused by subject-specific patterns, we employ a leakage-safe, subject-wise 5-fold cross-validation design. Each epoch  $i$  is described by a hand-crafted feature vector  $x_i$ , a sleep-stage label  $y_i \in \{W, N1, N2, N3, REM\}$ , and a subject identifier  $s_i$ . Outer folds are constructed using a stratified, group-wise splitting strategy in which subjects form the groups and fold assignment is stratified over stage labels to keep class proportions comparable, while guaranteeing that no subject appears in both training and test splits. Within each fold, all epochs assigned to a split are stacked into a feature matrix  $X \in \mathbb{R}^{N \times D}$  with an associated label vector  $y \in \mathbb{R}^N$  and subject-index vector  $s \in \mathbb{R}^N$ , where  $N$  denotes the number of epochs in the corresponding split and  $D = 28$  is the feature dimensionality.

All operations that depend on the data distribution are executed within the outer fold under a strict train-only rule. Concretely, the scaler  $g$  (Section II.D) is fit using only the outer-training features and then applied to both the training and test sets. When resampling  $r$  is enabled (Section II.E), it is applied only to

the training partition and implemented in a subject-aware manner. This restriction is motivated by the large inter-subject variability in EEG: generating synthetic samples by pooling all subjects may lead to overly optimistic behavior and can introduce indirect leakage by blending subject-specific signatures across the feature space. For cost-sensitive learning, we compute class counts  $n_{f,c}$  for each class  $c$  from the original (pre-resampling) training labels in fold  $f$ . These counts parameterize the chosen cost formulation  $\phi(\cdot)$  (Section II.F) to produce per-sample weights  $w$ . Finally, an MLP classifier  $h$  (Section II.G) is trained on the resampled training data using the corresponding sample weights, and performance is assessed exclusively on held-out test subjects. The per-fold workflow is summarized in Algorithm 1.

Algorithm 1. Leakage-safe subject-wise evaluation pipeline (per outer fold).

```

Input:  $D = \{(x_i, y_i, s_i)\}$  // (features, labels, subject IDs)
       scaler  $g$ ; sampler  $r$ ; cost scheme  $\phi$ ; classifier  $h$  (MLP)
Output: fold metrics  $\{\text{metrics}_f\}$ , confusion matrices  $\{\text{CM}_f\}$ , mean across folds

for  $f = 1..5$  do
  ( $S_{\text{train}}, S_{\text{test}}$ )  $\leftarrow$  StratifiedGroupSplit(subject IDs, labels,  $f$ )
  ( $X_{\text{train}}, y_{\text{train}}$ )  $\leftarrow$  SelectBySubjects( $D, S_{\text{train}}$ )
  ( $X_{\text{test}}, y_{\text{test}}$ )  $\leftarrow$  SelectBySubjects( $D, S_{\text{test}}$ )

   $g_f \leftarrow$  FitScaler( $g, X_{\text{train}}$ ) // fit on train only
   $X_{\text{train}} \leftarrow$  Transform( $g_f, X_{\text{train}}$ )
   $X_{\text{test}} \leftarrow$  Transform( $g_f, X_{\text{test}}$ )

   $n \leftarrow$  ClassCounts( $y_{\text{train}}$ ) // counts BEFORE resampling
  ( $X_{\text{res}}, y_{\text{res}}$ )  $\leftarrow$  ResampleTrainOnly( $r, X_{\text{train}}, y_{\text{train}}$ )

   $w \leftarrow$  ComputeSampleWeights( $\phi, y_{\text{res}}, n$ ) // see Section II.F
   $h_f \leftarrow$  FitMLP( $h, X_{\text{res}}, y_{\text{res}}, \text{sample\_weight} = w$ )

   $\hat{y} \leftarrow$  Predict( $h_f, X_{\text{test}}$ )
   $\text{metrics}_f, \text{CM}_f \leftarrow$  Evaluate( $\hat{y}, y_{\text{test}}$ )
end for

return  $\text{mean}_f(\text{metrics}_f)$  and  $\{\text{CM}_f\}$ 

```

#### D. Feature Scaling

We test three simple scaling options: z-score standardization, robust scaling based on the median and interquartile range, and min–max normalization. We include scaling as a first-class experimental factor for a practical reason: it can change how “close” two epochs look in feature space. This matters directly for SMOTE-style methods, which rely on nearest neighbors. It also matters for the MLP, where feature magnitude can affect the stability and speed of optimization. Each scaling transform is applied under the leakage-safe protocol in Section II.C. In every outer fold, the scaler is fit on the training split only and then used to transform both training and test features.

#### E. Resampling (Sampler) Strategies

We study four resampling conditions: no resampling, SVMSMOTE, SMOTE-Tomek, and SMOTE with AllKNN. Whenever resampling is enabled, it is applied only to the training split and implemented in a subject-aware manner. Concretely, the sampler is run separately on each subject’s training epochs and the resulting resampled subsets are then concatenated (no cross-subject mixing).

We use a consistent class-wise target within each fold for SVMSMOTE and SMOTE with AllKNN. Each minority class is upsampled until it reaches 70% of the majority-class size, based on the fold’s initial (pre-resampling) training distribution. SMOTE with AllKNN is implemented as a two-step procedure (SMOTE followed by AllKNN cleaning), whereas SMOTE-Tomek is applied in its integrated form. Consequently, only SMOTE-Tomek is not constrained to the same fixed 70% target.

#### F. Cost-Sensitive Learning via Sample Weights

We implement cost-sensitive learning by assigning per-sample weights during MLP training (via `sample_weight`). In each outer fold, we first compute class counts  $\{n_c\}_{c=1}^K$  from the original (pre-resampling) training labels, where  $K = 5$  and  $N = \sum_{c=1}^K n_c$ . Using these counts, we define a class-weight vector  $\{w_c\}$ . Each training instance  $i$  is then weighted according to its class label, i.e.,  $w_i = w_{y_i}$ . We compare four weighting settings: none ( $w_i = 1$ ), inverse-frequency, effective-number (class-balanced) [19], and log-scaled weighting. The class-weighting rules are:

$$\text{Inverse-frequency (balanced): } w_c = \frac{N}{K n_c}.$$

$$\text{Effective-number weighting (class-balanced): } w_c = \frac{1-\beta}{1-\beta n_c}, \beta \in (0,1).$$

$$\text{Log-scaled weighting: } w_c = \frac{1}{\log(k+n_c)}, k > 1.$$

We set  $\beta = 0.999$  and  $k = 1.1$  in all experiments. Sample weights are used only in training (never in scaling, resampling, or evaluation) and are applied within each fold following the leakage-safe protocol.

### G. Classifier and Training Setup

We train a multilayer perceptron (MLP) classifier with a fixed architecture and optimization setup across all experimental conditions. The network uses a single hidden layer with 100 units and ReLU activation. Training is performed with stochastic gradient descent using an  $\ell_2$  penalty of  $\alpha = 0.001$ , an initial learning rate of  $10^{-3}$ , and a maximum of 1000 iterations. We do not use early stopping. All runs use a fixed random seed (42).

The MLP hyperparameters are kept constant in this study. Rather than re-running an exhaustive hyperparameter search, we reuse a configuration identified as well-performing in our earlier work and fix it to isolate the effects of the pipeline components under investigation (scaling, resampling, and cost weighting), rather than to maximize absolute performance. When cost-sensitive learning is enabled, the model is fit with per-sample weights as described in Section II.F.

### H. Experimental Design

We use a full-factorial design to study the interaction between pipeline components: 3 feature scalers  $\times$  4 resampling settings  $\times$  4 cost-weighting schemes. Combined with 5 outer cross-validation folds, this yields  $3 \times 4 \times 4 \times 5 = 240$  runs. For each run, we store the fold outputs (metrics and confusion matrices) and later aggregate them across folds to support comparative ranking and trade-off analysis.

#### i. Evaluation Metrics and Utility-Based Trade-off Analysis

We report accuracy, macro-F1, and Cohen's  $\kappa$  as global performance indicators.  $F1_c$  denotes the class-specific F1-score computed for class  $c$ , and  $F1_{N1}$  refers to the F1-score of the N1 stage. To make class-specific behavior explicit, we report per-class F1 scores and the confusion matrix for each fold.

To quantify trade-offs relative to a chosen baseline  $b$ , we define the class-wise change in F1 as:

$$\Delta F1_c^{(b)} = F1_c(\text{config}) - F1_c(\text{baseline } b).$$

Here, 'config' denotes the evaluated pipeline configuration, and 'baseline  $b$ ' refers to the corresponding reference setting used for comparison. We then summarize the "cost of improvement" using a utility function that prioritizes N1 while accounting for changes in the remaining classes:

$$U_b(\alpha) = 2 \Delta F1_{N1}^{(b)} + \alpha \sum_{c \in \{W, N2, N3, \text{REM}\}} \Delta F1_c^{(b)}.$$

We evaluate  $\alpha \in \{0.5, 1, 2\}$  and use  $\alpha = 1$  for primary reporting.

Utility scores are computed under three baseline definitions:

- $U_0$ (global baseline): Standard scaling + no resampling + no cost weighting.
- $U_1$ (within-scaler baseline): same scaler + no resampling + no cost weighting.
- $U_2$ (within-cell baseline): same scaler + same sampler + cost weighting disabled.

All metrics, confusion matrices, and utility values are computed at the fold level and averaged across the five outer folds.

### J. Implementation and Reproducibility

All experiments were implemented in Python (v3.12.3) in a Windows/conda environment using scikit-learn (v1.7.1). Randomness was controlled by fixing `random_state = 42` wherever applicable. Fold-level results were saved in a consistent format to enable cross-fold aggregation and utility-based comparisons.

## III. RESULTS

We report accuracy, macro-F1, Cohen’s  $\kappa$ , and class-wise F1 scores, with a specific focus on F1(N1). To quantify the cost of improvement, we summarize how gains in N1 relate to concurrent changes in  $\{W, N2, N3, REM\}$  and use the proposed utility views  $(U_0, U_1, U_2)$  to rank configurations under different preference settings ( $\alpha$ ). Across the 240 runs, we observe that cost-sensitive learning alone often achieves N1 improvements comparable to SMOTE-family resampling, while combining resampling with cost weighting provides limited additional benefit in most settings. For readability, we refer to the scaling methods as z-score (STD), robust scaling (ROB), and min-max scaling (MMX).

### A. Effect of Feature Scaling under the Baseline Setting

The three scaling techniques under the baseline configuration (ORI = no resampling; cost = unweighted) are compared in Table 1. In terms of accuracy, macro-F1, and Cohen’s  $\kappa$ , STD and ROB perform almost identically, and they exhibit the same baseline level for F1(N1). In contrast, MMX shows a lower baseline, with a clear reduction in F1(N1) together with lower macro-F1 and  $\kappa$ . Stage-wise, W and N2 remain stronger but are still lower than under STD/ROB, whereas the largest MMX gaps are observed in N1 and REM.

Table 1. Baseline performance (mean  $\pm$  std over 5 folds) for each scaling method under the baseline configuration.

Scaler	Accuracy	Macro-F1	Cohen’s $\kappa$	F1(N1)	F1(N2)	F1(N3)	F1(REM)	F1(W)
STD	0.75 $\pm$ 0.02	0.65 $\pm$ 0.02	0.65 $\pm$ 0.02	0.22 $\pm$ 0.04	0.81 $\pm$ 0.02	0.80 $\pm$ 0.03	0.56 $\pm$ 0.04	0.87 $\pm$ 0.02
ROB	0.75 $\pm$ 0.01	0.65 $\pm$ 0.02	0.65 $\pm$ 0.02	0.22 $\pm$ 0.05	0.81 $\pm$ 0.02	0.79 $\pm$ 0.03	0.56 $\pm$ 0.03	0.87 $\pm$ 0.02
MMX	0.73 $\pm$ 0.02	0.62 $\pm$ 0.03	0.61 $\pm$ 0.03	0.17 $\pm$ 0.05	0.78 $\pm$ 0.03	0.79 $\pm$ 0.04	0.50 $\pm$ 0.06	0.85 $\pm$ 0.02

### A. Global Ranking under $U_0(\alpha = 1)$ : Best Overall Trade-offs

In subsequent results, sampling strategies are abbreviated as ORI (no resampling), SVM (SVMSMOTE), TOM (SMOTE-Tomek), and AKNN (SMOTE with AllKNN), while cost-weighting schemes are abbreviated as NONE (no cost weighting), INV (inverse-frequency), EFF (effective-number), and LOG (log-scaled). Table 2 lists the top-10 configurations ranked by the global utility  $U_0(\alpha = 1)$  against the global baseline (STD–ORI–NONE). The top configuration is STD–ORI–INV ( $U_0 = 0.11$ ). It reaches F1(N1) = 0.34 while keeping overall performance stable ( $\kappa = 0.60$ , macro-F1 = 0.65, Acc = 0.70). The runner-up, ROB–ORI–INV, is very close ( $U_0 = 0.10$ , F1(N1) = 0.34). This puts an ORI + cost option at the top under  $U_0$ .

Table 2. Top-10 configurations ranked by the global utility  $U_0(\alpha = 1)$ .

Rank	Scaler	Sampler	Cost	$U_0(\alpha = 1)$	F1(N1)	F1(W)	F1(N2)	$\kappa$	Macro-F1	Acc
1	STD	ORI	INV	0.11	0.34	0.84	0.76	0.60	0.65	0.70
2	ROB	ORI	INV	0.10	0.34	0.84	0.76	0.60	0.65	0.70
3	STD	TOM	NONE	0.10	0.34	0.85	0.76	0.60	0.65	0.70
4	STD	AKNN	EFF	0.09	0.34	0.85	0.75	0.60	0.65	0.70
5	ROB	TOM	LOG	0.08	0.34	0.85	0.75	0.60	0.64	0.70
6	ROB	AKNN	EFF	0.08	0.34	0.85	0.75	0.60	0.65	0.70
7	ROB	AKNN	NONE	0.08	0.34	0.85	0.75	0.60	0.64	0.70

8	ROB	SVM	NONE	0.10	0.32	0.85	0.78	0.61	0.65	0.71
9	ROB	TOM	EFF	0.09	0.33	0.86	0.76	0.61	0.65	0.70
10	ROB	SVM	LOG	0.09	0.33	0.85	0.77	0.61	0.65	0.71

Resampling-only settings are also competitive. STD-TOM-NONE matches the same N1 score ( $F1(N1) = 0.34$ ) with  $U_0 = 0.10$  and a slightly higher  $F1(W) = 0.85$  (Table 2). Several sampling+cost entries sit just below the top. For instance, STD-AKNN-EFF and ROB-AKNN-EFF again reach  $F1(N1) = 0.34$ , but they do not exceed the best ORI+cost utilities ( $U_0 = 0.09$  and  $0.08$ , respectively). SVM-based variants can improve accuracy a bit (e.g., ROB-SVM-NONE,  $Acc = 0.71$ ), yet they lag in N1 ( $F1(N1) = 0.32$ ). Overall, the top list supports a simple point: cost weighting alone can match the N1 gain of SMOTE-family resampling, and combining the two rarely adds much on top in this regime.

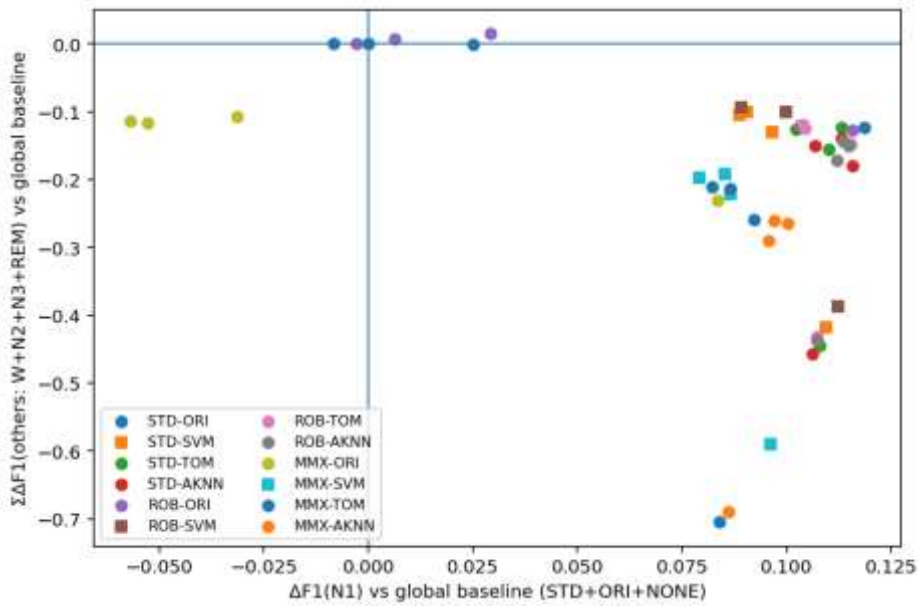


Fig. 2. Global trade-off under  $U_0$ :  $\Delta F1(N1)$  versus the aggregate change in the remaining stages relative to the baseline.

Fig. 2 provides the trade-off view. It plots  $\Delta F1(N1)$  versus the aggregate change in the remaining stages  $\{W, N2, N3, REM\}$  relative to the baseline. Each point corresponds to one (scaler, sampler, cost) configuration (mean over folds); thus, for each (scaler, sampler) pair, four points reflect the four cost-weighting schemes. Most points with strong N1 gains fall below zero on the y-axis, showing a measurable cost in the other classes. The top-ranked configurations sit in the area with positive  $\Delta F1(N1)$  and a relatively small drop in the others.

*B. Within-Scaler Selection under  $U_1(\alpha = 1)$ : Best Choices per Scaler*

Table 3 summarizes the best-performing configurations under the within-scaler utility  $U_1(\alpha = 1)$ , where each scaling method is ranked relative to its own baseline (ORI, NONE) within that scaler. Under STD, the top within-scaler choice is STD-ORI-INV ( $U_1 = 0.11$ ), with  $F1(N1) = 0.34$ . ROB shows the same pattern: ROB-ORI-INV is also ranked first ( $U_1 = 0.11$ ,  $F1(N1) = 0.34$ ), confirming that inverse-frequency weighting remains the most favorable within-scaler option under both STD and ROB.

Table 3. Best six configurations ranked by  $U_1(\alpha = 1)$ , reporting the top two candidates within each scaler.

Scaler	Best Sampler	Best Cost	$U_1(\alpha = 1)$	$F1(N1)$	$F1(W)$	$F1(N2)$	$\kappa$	Macro-F1	Acc
MMX	SVM	EFF	0.20	0.31	0.83	0.75	0.59	0.63	0.69
MMX	SVM	NONE	0.18	0.30	0.83	0.75	0.58	0.63	0.69
STD	ORI	INV	0.11	0.34	0.84	0.76	0.60	0.65	0.70

ROB	ORI	INV	0.11	0.34	0.84	0.76	0.60	0.65	0.70
ROB	SVM	NONE	0.10	0.32	0.85	0.78	0.61	0.65	0.71
STD	TOM	NONE	0.10	0.34	0.85	0.76	0.60	0.65	0.70

In contrast, the best MMX choices differ and are driven by SVM-based resampling. The highest MMX entry is MMX-SVM-EFF ( $U_1 = 0.20$ ), followed by MMX-SVM-NONE ( $U_1 = 0.18$ ). While these MMX winners improve N1 relative to the MMX baseline, their absolute F1(N1) values (0.31 and 0.30) remain below the STD/ROB within-scaler winners (0.34) (Table 3). Overall, Table 3 highlights a clear scaler dependence: STD/ROB favor cost weighting without resampling (ORI+INV), whereas MMX favors SVM-based resampling (SVM+EFF/NONE).

C. Added Value of Cost-Sensitive Learning Within Each Scaler-Sampler Cell ( $U_2$ )

Table 4 quantifies the incremental contribution of cost-sensitive learning within each fixed (scaler, sampler) cell using  $U_2(\alpha = 1)$ , i.e., relative to the cost=NONE configuration in the same cell. The strongest added value appears in the ORI cells. For STD, inverse-frequency weighting is the best option (STD-ORI-INV,  $U_2 = 0.11$ ), driven by  $\Delta F1(N1) = +0.12$  with a corresponding  $\Delta \Sigma F1(\text{others}) = -0.12$ . ROB-ORI-INV shows the same pattern ( $U_2 = 0.11$ ,  $\Delta F1(N1) = +0.12$ ,  $\Delta \Sigma F1(\text{others}) = -0.13$ ). Under MMX, the ORI cell exhibits the largest cost effect overall: MMX-ORI-INV reaches  $U_2 = 0.16$  with  $\Delta F1(N1) = +0.14$  and  $\Delta \Sigma F1(\text{others}) = -0.11$  (Table 4).

Table 4. Best cost method and added value within each (scaler, sampler) cell ( $U_2, \alpha = 1$ ).

Scaler	Sampler	Best Cost	$U_2(\alpha = 1)$	$\Delta F1(N1)$	$\Delta \Sigma F1(\text{others})$
STD	ORI	INV	0.11	0.12	-0.12
STD	SVM	EFF	-0.01	0	0
STD	TOM	EFF	-0.02	-0.01	0
STD	AKNN	EFF	0.02	0.01	0.01
ROB	ORI	INV	0.11	0.12	-0.13
ROB	SVM	LOG	-0.01	0	-0.02
ROB	TOM	EFF	0	0	0
ROB	AKNN	EFF	0	0	0.01
MMX	ORI	INV	0.16	0.14	-0.11
MMX	SVM	EFF	0.02	0.01	0.01
MMX	TOM	EFF	-0.01	0	0
MMX	AKNN	EFF	0	0	0

In contrast, once resampling is already applied, adding cost-sensitive learning yields little further benefit in most cells. Under STD, the best cost choice within SVM and TOM cells is EFF, yet the net changes are negligible or slightly negative ( $U_2 = -0.01$  for SVM;  $U_2 = -0.02$  for TOM). The only STD resampling cell with a clear positive net effect is AKNN, where EFF gives  $U_2 = 0.02$  with small gains ( $\Delta F1(N1) = +0.01$ ,  $\Delta \Sigma F1(\text{others}) = +0.01$ ). For ROB, resampling cells are essentially flat ( $U_2 \approx 0$ ), with near-zero deltas across SVM/TOM/AKNN. A similar pattern holds for MMX, where the resampling cells show modest or near-zero increments (e.g., MMX-SVM-EFF:  $U_2 = 0.02$ ) and the remaining resampling cells remain close to zero. Overall, Table 4 supports a consistent conclusion: cost weighting delivers its primary added value when applied without resampling (ORI), whereas its incremental benefit is limited once SMOTE-family sampling is already in place.

D. Sensitivity of the Global Ranking to the Utility Weight  $\alpha$

Table 5 reports the top-5 configurations under  $U_0$  for three utility weights ( $\alpha = 0.5, 1.0, 2.0$ ), and Table 6 summarizes the overlap between the corresponding top-5 sets. The  $\alpha = 0.5$  and  $\alpha = 1.0$  lists overlap in

three configurations: STD-ORI-INV, ROB-ORI-INV, and STD-TOM-NONE ( $3/5 = 60\%$ ). The remaining two entries differ. At  $\alpha = 0.5$ , the list includes STD-AKNN-EFF and ROB-TOM-LOG, whereas at  $\alpha = 1.0$  it includes ROB-SVM-NONE and ROB-TOM-EFF.

At  $\alpha = 2.0$ , the composition changes more markedly and stability drops (only 1/5 overlap with  $\alpha = 1.0$ ; 0/5 with  $\alpha = 0.5$ ). The top-ranked entries are ORI-based, cost-weighted solutions (ROB-ORI-LOG, STD-ORI-LOG, ROB-ORI-EFF) and the unweighted baseline (STD-ORI-NONE) also appears in the top-5. Notably,  $U_0$  values compress toward zero at  $\alpha = 2.0$  (0.09 to 0.00), indicating that placing more weight on the non-N1 stages favors more conservative trade-offs, even when N1 gains are smaller.

Table 5. Sensitivity of the global ranking to the utility weight  $\alpha$ : top-5 configurations under  $U_0$  for  $\alpha \in \{0.5, 1, 2\}$ .

Rank	( $\alpha = 0.5$ ) (Config)	( $U_0$ )	( $\alpha = 1.0$ ) (Config)	( $U_0$ )	( $\alpha = 2.0$ ) (Config)	( $U_0$ )
1	STD-ORI-INV	0.18	STD-ORI-INV	0.11	ROB-ORI-LOG	0.09
2	ROB-ORI-INV	0.17	ROB-ORI-INV	0.10	STD-ORI-LOG	0.05
3	STD-TOM-NONE	0.16	STD-TOM-NONE	0.10	ROB-ORI-EFF	0.03
4	STD-AKNN-EFF	0.16	ROB-SVM-NONE	0.10	STD-ORI-NONE	0
5	ROB-TOM-LOG	0.16	ROB-TOM-EFF	0.09	ROB-SVM-NONE	0

Table 6. Ranking stability across  $\alpha$  (top-5 overlap).

Set comparison	Overlap (count)	Overlap (%)
Top-5: ( $\alpha = 0.5$ ) vs ( $\alpha = 1.0$ )	3	60%
Top-5: ( $\alpha = 1.0$ ) vs ( $\alpha = 2.0$ )	1	20%
Top-5: ( $\alpha = 0.5$ ) vs ( $\alpha = 2.0$ )	0	0%

### E. Confusion-Matrix Analysis of Stage-Specific Error Shifts

Figure 3 shows row-normalized confusion matrices for five settings: STD-ORI-NONE (baseline), STD-ORI-INV (cost-only), STD-TOM-NONE (sampler-only), STD-AKNN-EFF (hybrid), and MMX-SVM-EFF (MMX-based). In the baseline (Fig. 3a), N1 recall is only 17.2%. Most N1 epochs are predicted as N2 (33.8%), W (26.0%), or REM (22.7%). By comparison, W and N2 are much more stable (89.9% and 84.4% recall).

With STD-ORI-INV (Fig. 3b), N1 recall jumps to 45.0%. This does not come for free. W recall falls to 78.6% and N2 recall to 67.5%. The shift is visible in the rows: W→N1 increases from 4.0% to 14.0%, and N2→N1 from 2.6% to 12.4%. There is also a clear REM→N1 increase. In other words, part of the N1 gain is obtained by pulling borderline REM epochs into N1.

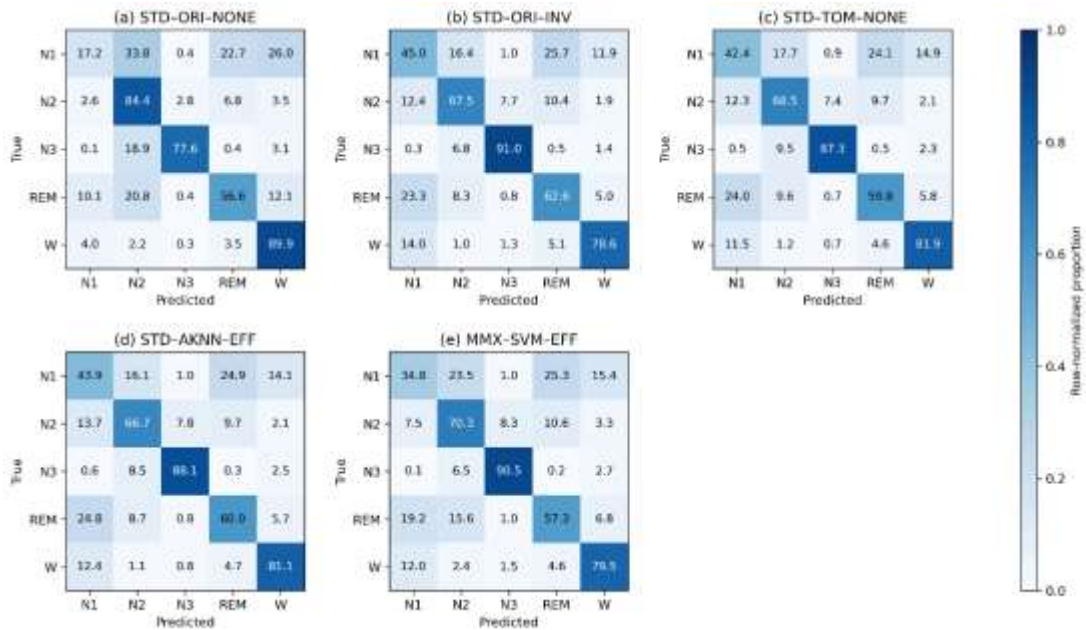


Fig. 3. Row-normalized confusion matrices for the baseline, cost-only, sampler-only, hybrid, and MMX-based settings.

A similar picture appears with sampling. STD-TOM-NONE (Fig. 3c) raises N1 recall to 42.4%. STD-AKNN-EFF (Fig. 3d) is close (43.9%). In both cases, the N1 gain again coincides with lower recalls in the stronger classes, especially W and N2 (Figs. 3c–d). The REM row also contributes to the N1 increase, although the pattern is milder than in the cost-only setting.

The MMX case is different. In MMX-SVM-EFF (Fig. 3e), N1 recall reaches 34.8%, but the bigger change is N3. N3 recall rises to 90.5%, up from 77.6% in the baseline. At the same time, W and N2 recalls drop to 79.5% and 70.3%. This matters for interpretation. Gains in N3 can partly offset losses in W/N2/REM when results are aggregated. So, the “cost of improving N1” is not carried by one class only. It is redistributed, and sometimes partly masked by a strong improvement in N3.

#### IV. DISCUSSION

Because a single headline score can conceal the outcomes of individual stages, we present results in layers. Improving N1 often shifts errors into nearby stages, mainly W and N2. This is important if tracking fragmentation or transitions is the aim. Three straightforward questions are addressed by the three utility views.  $U_0$  shows the best overall trade-off.  $U_1$  picks the best option within a scaler. After resampling is implemented,  $U_2$  determines whether cost weighting is still beneficial. Together, they make the N1 gain—and its side-effects—visible.

In this pipeline, scaling is important. It alters both the neighborhoods that SMOTE-style samplers use and what the MLP “sees.” Robustness to outliers has little effect here because STD and ROB behave nearly identically in the baseline. MMX is weaker at first, especially on N1. Due to its proximity to other stages in this dataset, N1 is the first stage to suffer when all features are mapped into  $[0,1]$ .

The nearest neighbors determined by feature distances are used in neighbor-based resampling. The set of nearest neighbors can change when the scaling technique is modified. Global rankings may be deceptive for MMX since it begins with a weaker baseline. At baseline, STD and ROB are close; later gaps mostly come from the scaler-sampler and scaler-cost pairings. For MMX, we therefore focus on within-scaler choices ( $U_1$ ) to judge what cost weighting adds.

Two distinct trends in the  $U_0$  ranking are displayed in Table 2. In this pipeline, inverse cost weighting on ORI can reach N1 gains similar to SMOTE-style resampling. However, once you look at the remaining errors, the same N1 may appear differently. Some settings raise N1 by shifting mistakes from nearby stages; others keep those shifts smaller, even if N1 improves less. For this reason, rather than selecting a single “best” N1 score, we compare options within each scaler after summarizing performance using utility.

Table 2 shows that similar N1 can come from different settings. Resampling tends to shift mistakes between stages. Other settings improve agreement and  $\kappa$ , but with lower N1. The top ten list is not about a single winner. It simply shows the trade-offs. What you pick depends on the goal: pushing N1, or keeping the overall error pattern more stable.

Fig. 2 shows this is not a single lucky configuration. There are many ways to increase N1, but only some of them keep the rest of the confusion pattern from getting worse. That is why we do not rely on a single global list. Next, we choose settings within each scaler ( $U_1$ ). Then we fix the sampler and check one thing ( $U_2$ ): does cost weighting still help, or is it redundant once resampling is in place?

Table 3 shows a clear MMX-specific result. MMX improves most when it is paired with SVM-based resampling, and the best MMX option shifts accordingly to MMX–SVM–EFF. Within MMX, this resampling route is consistently stronger than relying on cost weighting alone. For STD and ROB, the within-scaler picture is simpler. ORI–INV is still the best choice for N1 here. This contrast is why  $U_1$  is useful. It keeps the recommendation tied to the chosen scaler, instead of forcing one “best” recipe for everyone.

Table 4 separates two cases. When we do not resample (ORI), cost weighting is the main lever for N1, and inverse weighting is the most consistent choice. The gain is not free. N1 improves mainly by shifting errors from nearby stages, most visibly W and N2, and this shows up as a drop in aggregate performance elsewhere.

Once resampling is in place, cost weighting usually adds little. In most resampling cells, the deltas are close to zero, which suggests that resampling already captures most of the rebalancing needed in this feature space. One exception is the STD setting with ALLKNN and effective-number weighting, where a small positive delta appears. Overall, the results indicate that cost weighting acts mainly as an alternative to resampling rather than as a generally synergistic addition.

Across Tables 5–6, the main conclusions are stable for moderate  $\alpha$  values (0.5–1.0). The top sets overlap strongly and keep the same core options, including ORI–INV under STD/ROB and a small set of competitive resampling alternatives. This suggests the ranking is not overly sensitive to  $\alpha$  within the range that balances N1 gains against collateral loss.

The confusion matrices explain what sits behind the utility trade-offs. At baseline, most N1 errors go to W, N2, and sometimes REM, which fits N1 behaving like a transition stage. ORI–INV and the strongest resampling settings improve N1 mainly by recovering part of these borderline epochs. The gain is not free: W and N2 shift in parallel, which is exactly the collateral effect captured by the utility terms and reflected in the  $\alpha$  sensitivity.

Under MMX–SVM–EFF, the redistribution looks different. N1 recall increases together with a strong N3 recall gain, while W and N2 recall drop. This matches Table 3, where the best MMX setting comes from the SVM-based resampling route. Because the confusion matrices are row-normalized, they only show recall, so we read these shifts together with the F1 results. The practical message is that “best” depends not only on how much N1 improves, but also on where the recovered N1 comes from and which shifts are acceptable for the intended use.

At  $\alpha = 2.0$ ,  $\alpha$  behaves less like a technical detail and more like a preference setting. With a strong penalty on non-target changes, conservative ORI options move up, including log-weighted variants and even the baseline. In other words, when collateral shifts across stages are heavily discouraged, smaller N1 gains can become preferable. It reflects what is prioritized: pushing N1 aggressively, or preserving overall agreement and the reliability of dominant stages. This is why a single  $\alpha$  can be misleading. Tables 5–6 help by showing how conclusions change as that preference becomes more conservative.

This study is limited to a single dataset, a single EEG channel, and handcrafted features with a fixed MLP. This controlled setup helps isolate scaler–sampler–cost effects under leakage-safe evaluation, but the results may not fully carry over to multi-channel PSG, different feature representations, or end-to-end temporal deep models. We also report fold-averaged metrics and use a restricted set of cost schemes and SMOTE-family samplers, so other imbalance or robustness strategies were out of scope. Future work should test whether the same trade-offs hold across additional cohorts and recording conditions, extend

the analysis to multimodal inputs, and add explicit temporal context—especially around  $W \leftrightarrow N1$  and  $N1 \leftrightarrow N2$  transitions—to better address boundary-driven N1 errors.

## V. CONCLUSION

In this study, we tested cost-sensitive learning for N1 on Sleep-EDF (Sleep Cassette) with leakage-safe, subject-wise 5-fold evaluation and a fixed handcrafted-feature MLP pipeline. In the global ranking ( $U_0$ ), inverse-frequency weighting on the original data reached the best N1 trade-offs and was competitive with SMOTE-family resampling. The within-scaler view ( $U_1$ ) showed that the best choice depends on the scaler: STD/ROB favor ORI-INV, while MMX works best with SVM-based resampling and effective-number weighting. The within-cell view ( $U_2$ ) showed that cost weighting helps most when no resampling is used, and adds little once SMOTE-family sampling is applied. Overall, improving N1 shifts errors in other stages, and the utility summaries make those shifts visible.

## REFERENCES

- [1] D. J. Levendowski, D. Popovic, C. Berka, and P. R. Westbrook, “Retrospective cross-validation of automated sleep staging using electroocular recording in patients with and without sleep disordered breathing,” *Int. Arch. Med.*, vol. 5, no. 1, pp. 1–9, 2012.
- [2] B. P. Lucey, J. S. Mcleland, C. D. Toedebusch, J. Boyd, J. C. Morris, E. C. Landsness, and D. M. Holtzman, “Comparison of a single-channel EEG sleep study to polysomnography,” *J. Sleep Res.*, vol. 25, no. 6, pp. 625–635, 2016.
- [3] A. Viola, L. Thiesse, L. Staner, G. Fuchs, T. Roth, J. Y. Schaffhauser, and R. Luthringer, “0951 Inter-Scorer Reliability between Somno-Art Software and 5 Sleep Centers,” *Sleep*, vol. 46, p. A419, 2023.
- [4] W. Li, T. Liu, B. Xu, and A. Song, “SleepFC: Feature Pyramid and Cross-Scale Context Learning for Sleep Staging,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 32, pp. 2198–2208, 2024.
- [5] Z. Xiong, Y. Gou, Y. Zhou, Y. Yang, Y. Peng, Y. Gong, and X. Zeng, “KAN-SleepNet: A deep learning model combining Kolmogorov–Arnold Networks and bidirectional LSTM for automated sleep staging using EEG signals,” *Digital Health*, vol. 11, Art. no. 20552076251398440, 2025.
- [6] H. Phan and K. Mikkelsen, “Automatic sleep staging of EEG signals: recent development, challenges, and future directions,” *Physiol. Meas.*, vol. 43, no. 4, Art. no. 04TR01, 2022.
- [7] S. K. Satapathy, B. Brahma, B. Panda, P. Barsocchi, and A. K. Bhoi, “Machine learning-empowered sleep staging classification using multi-modality signals,” *BMC Med. Inform. Decis. Mak.*, vol. 24, no. 1, Art. no. 119, 2024.
- [8] F. Del Pup, A. Zanola, L. F. Tshimanga, A. Bertoldo, L. Finos, and M. Atzori, “The role of data partitioning on the performance of EEG-based deep learning models in supervised cross-subject analysis: a preliminary study,” *Comput. Biol. Med.*, vol. 196, Art. no. 110608, 2025.
- [9] V. Birrer, M. Elgendi, O. Lamercy, and C. Menon, “Evaluating reliability in wearable devices for sleep staging,” *npj Digit. Med.*, vol. 7, no. 1, 2024.
- [10] M. Rusanen, G. Jouan, R. Huttunen, S. Nikkonen, S. Sigurðardóttir, J. Töyräs, and H. Korkalainen, “Retrospective validation of automatic sleep analysis with grey areas model for human-in-the-loop scoring approach,” *J. Sleep Res.*, vol. 34, no. 3, Art. no. e14362, 2025.
- [11] D. M. Rapoport, “Non-contact sleep monitoring: are we there yet?,” *J. Clin. Sleep Med.*, vol. 15, no. 7, pp. 935–936, 2019.
- [12] M. Khushi, K. Shaukat, T. M. Alam, I. A. Hameed, S. Uddin, S. Luo, and M. C. Reyes, “A comparative performance analysis of data resampling methods on imbalance medical data,” *IEEE Access*, vol. 9, pp. 109960–109975, 2021.
- [13] K. Wang, Q. Xue, and J. J. Lu, “Risky driver recognition with class imbalance data and automated machine learning framework,” *Int. J. Environ. Res. Public Health*, vol. 18, no. 14, Art. no. 7534, 2021.
- [14] V. Asha, M. T. Vasumathi, A. Prasad, Y. AP, and M. Sivani, “Evaluation of ML Models using SMOTE and Feature Scaling for Intrusion Detection System (IDS),” in *Proc. 2025 Int. Conf. Visual Analytics and Data Visualization (ICVADV)*, Mar. 2025, pp. 243–249.
- [15] N. C. Yang and K. L. Sung, “Non-intrusive load classification and recognition using soft-voting ensemble learning algorithm with decision tree, K-Nearest neighbor algorithm and multilayer perceptron,” *IEEE Access*, vol. 11, pp. 94506–94520, 2023.
- [16] M. Muntasir Nishat, F. Faisal, I. Jahan Ratul, A. Al-Monsur, A. M. Ar-Rafi, S. M. Nasrullah, and M. R. H. Khan, “A comprehensive investigation of the performances of different machine learning classifiers with SMOTE-ENN oversampling technique and hyperparameter optimization for imbalanced heart failure dataset,” *Sci. Program.*, vol. 2022, Art. no. 3649406, 2022.
- [17] A. S. Köksal, “Effect of Feature Scaling on the N1 Trade-off in SMOTE-Based Oversampling Methods: A Study on Sleep-EDF,” in *Proc. 4th Int. Conf. on Pioneer and Innovative Studies (ICPIS 2026)*, Konya, Turkey, Jan. 18–19, 2026, pp. 239–247.

- [18] H. Bei, Y. Wang, Z. Ren, S. Jiang, K. Li, and W. Wang, "A Statistical Approach to Cost-Sensitive AdaBoost for Imbalanced Data Classification," *Math. Probl. Eng.*, vol. 2021, no. 1, Art. no. 3165589, 2021.
- [19] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 9268–9277.
- [20] H. Phan, A. Mertins, and M. Baumert, "Pediatric automatic sleep staging: a comparative study of state-of-the-art deep learning methods," *IEEE Trans. Biomed. Eng.*, vol. 69, no. 12, pp. 3612–3622, 2022.
- [21] G. R. Liu, Y. L. Lo, J. Malik, Y. C. Sheu, and H. T. Wu, "Diffuse to fuse EEG spectra–intrinsic geometry of sleep dynamics for classification," *Biomed. Signal Process. Control*, vol. 55, Art. no. 101576, 2020.
- [22] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.
- [23] E. A. Wolpert, "A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects," *Archives of General Psychiatry*, 20(2), 246–247, 1969.
- [24] R. B. Berry, R. Brooks, C. E. Gamaldo, S. M. Harding, R. M. Lloyd, C. L. Marcus, and B. V. Vaughn (Eds.), *The AASM manual for the scoring of sleep and associated events: Rules, terminology and technical specifications (Version 2.4)*. Darien, IL: American Academy of Sleep Medicine, 2017.
- [25] S. K. Satapathy, "Machine learning approaches with automated sleep staging system based on two-layer heterogeneous ensemble learning stacking model," *International Journal of Computing and Digital Systems*, 2022.