Automated Sleep Staging using the data from the Dreem Headband by Beacon Biosignals
-
Machine Learning

Raphaël Romand-Ferroni
CentraleSupélec
raphael.romandferroni@student-cs.fr

Clément Simon
CentraleSupélec
clement.simon@student-cs.fr

Abstract

Automated sleep staging is a critical task in sleep medicine that involves classifying sleep stages based on biosignals. In this study, we present a comprehensive approach to automated sleep staging using biosignal data. Our method involves preprocessing the data and extracting features such as time-domain statistical features, spectral entropy, fractal dimensions, and wavelet features. These are used as inputs for machine and deep learning models, which are then combined as a bigger model through ensemble learning. The performance of the models is evaluated using various metrics, and post-processing techniques such as Hidden Markov Models and convolution band-pass filters are explored to further refine our predictions. Our approach achieved a promising F1 score of 70%, demonstrating the potential of our method to accurately classify sleep stages. This work has been widely nurtured by an article on the subject from the work of Khald Ali I. Aboalayon and al. published in 2016 ***Sleep Stage Classification Using EEG Signal Analysis: A Comprehensive Survey and New Investigation*** *[2].*

Figure 1. The Dreem Headband with sensors locations

1. Introduction

In this study, we present our approach to automated sleep staging using a dataset of 5 EEG and 3 Accelerometer records with corresponding targets representing 5 sleep stages. Here is a description of the different stages to predict:

- **Wake**: Prominence of the alpha rhythm, fast rhythm between 8 and 12 Hz (Subject awake with eyes closed).
- **N1**: Rythm between 4 and 7Hz, slow eyes movement
- **N2**: Appartition of spindles, short burst (0.5s - 2 s) between 11 and 16Hz and K-complexes
- **N3** (Slow wave sleep): Majority of slow waves (between 0.5 to 4Hz)
- **REM sleep**: Sawtooth waves (2 to 6Hz), and rolling symmetrical eyes movement

As detailed on **Beacon Biosignals**'s website [1], the headband measures EEG signals thanks to five sensors differently located around the brain as depicted in Figure 1: 2 frontal sensors to measure frontal brain activity (A), 1 ground sensor on the frontal band (B) and 2 occipital sensors to monitor occipital brain activity (C). Additionally, 3D accelerometers located in F, E and G (see Figure 1) measure movements, head position, and respiratory rate during sleep, which could greatly help our classification task.

In this report, we will delve into the five main steps that we followed to build our pipeline:

- Data description
- Pre-processing
- Feature extraction
- Comprehensive description of the final model
- Scoring and cross-validation method
- Post-processing
- Presentation of the results

2. Data description

For the competition, ten labeled nights recorded with the Dreem Headband 1 were provided by Beacon Biosignals [1]. Seven nights are used for training and three for evaluation. Each sleep stage is defined over 30-second periods which we will call a Sleep Epoch. A night of 8 hours contains 960 sleep epochs but this can be varying across the different records provided. During sleep, the brain oscillates between the five sleep stages previously described. Each stage corresponds to particular electric patterns and specific brain waves that we can easily visualize in Figure 2. These different patterns lead us to consider a first EEG decomposition into five frequency bands, namely δ, θ, α, β and γ (see Table 2) that will help us to gain valuable information for signals and improve our classification task.

Bands	Frequencies (Hz)
δ	$0-4$
θ	$4-8$
α	$8-13$
β	$13-22$
γ	>30

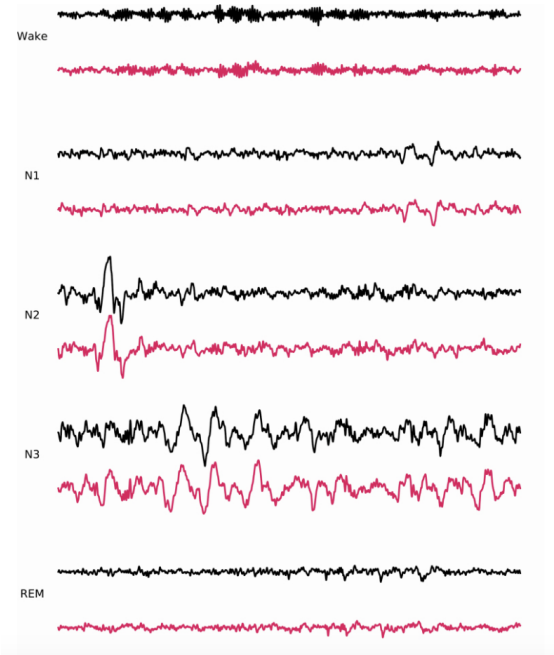

Figure 2. Sleep stages EEG patterns

3. Pre-processing

In the pre-processing phase of our automated sleep staging system, we implemented signal clipping as a crucial step to remove excessively large portions of the EEG and accelerometer signals, ensuring data integrity and mitigating the impact of extreme values on subsequent analyses. The EEG and accelerometer signals were subjected to a **signal clipping process** to address potential outliers and extreme values that could adversely affect the accuracy and reliability of the subsequent feature extraction and modeling stages. The signal clipping process involved setting upper and lower limits for the magnitude of the signals, effectively truncating any values that exceeded these limits. For the EEG data, a clipping limit of 200 Hz was applied (see Figure 3), while for the accelerometer data, a limit of 1.5 Hz was applied. These limits were determined based on domain knowledge and the characteristics of the physiological signals, ensuring that the clipped signals retained their essential information while removing extreme values that could introduce noise or bias into the subsequent analyses.

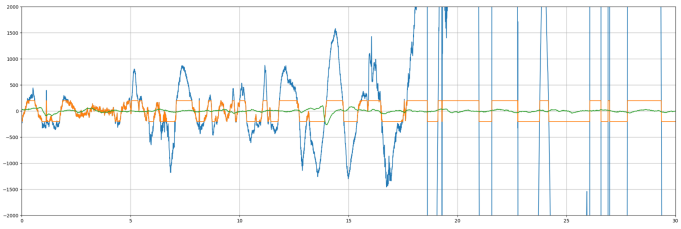

Figure 3. Clipping EEG data to 200 Hz

One other challenging issue to address is the highly class-imbalanced nature of the training set as depicted in Figure 4. It's clear here that Wake and N1 stages are under-represented in the data. We will have to address this issue by using a stratified sampling method and adding weights to the model to balance the dataset.

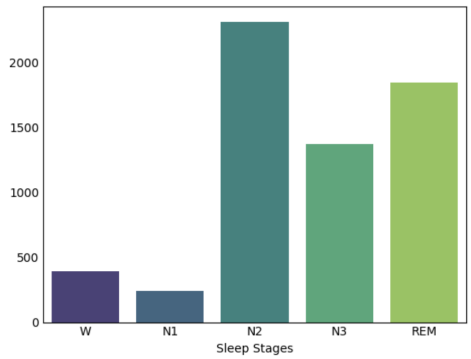

Figure 4. Class distribution across training records

4. Feature Extraction

In the context of automated sleep staging, feature extraction plays a pivotal role in capturing relevant information from EEG and accelerometer data, to characterize different sleep stages and facilitate the subsequent classification process. Our feature extraction process encompassed a diverse range of techniques and computations aimed at capturing meaningful attributes from the input signals. A main part of the feature extraction was provided by the state-of-the-art method used in the paper by Khald Ali I. Aboalayon and al. [2] but also with the great help of this article from 2019, [5]. Most features could be found already implemented in various Python/C++ libraries like PyEEG [3] and the TorchEEG library, built on the famous Deep Learning PyTorch architecture.

4.1. Features

Here are some of our most important features in terms of capturing the essence of what quantitatively defines a sleep stage, according to our device measurements.

Our protocol consisted of extracting (scalar) features to build the tabular data X (features), y (labels 0 to 4):

1. Per captor, e.g. the third EEG captor.

2. Per captor, on a specific power band (as seen above)

3. Combining different captors, e.g. finding the mean 3d distance of the (x, y, z) accelerometers (the 3d norm)

4.1.1 Time Domain Statistical Features

The standard statistics - mean, variance, standard deviation, skewness, kurtosis, median, and quantiles - provide valuable insights into the tendency, the degree of dispersion, the asymmetry, and the data peaks of the signals.

4.1.2 Frequency Domain

As we have seen above in Figure 2, the frequency content of an EEG epoch characterizes pretty well a sleep stage. A common extracted feature in such problems is frequency-domain features that are well known for their ability to capture the dynamics and the underlying statistical moments of the EEG signals. The first method to convert EEG time domain signals to a representation in the frequency domain is to compute its **Power Spectral Density** (PSD) in each frequency band listed above using Short-Time Fourier Transform (STFT). This will help us capture the distribution of signal power and valuable information about signal frequency content and dynamics.

Another famous decomposition method that we used is **Wavelet Transforms** (WT). Such a method is suitable for non-stationary signal analysis like EEG data. Like the PSD, WT breaks down the input EEG signal by shifting and scaling wavelets over the different frequency bands defined. The question remained to know what was the most informative wavelet family to use in the realm of EEG signals. Such a question is answered in the article of Gandhi et al. (2011) [4]. A combination of Haar, Daubechies, Coiflets, and Biorthogonal seemed to provide the best classification results.

Additionally, several entropy-based features were extracted like **Spectral entropy**, **SVD entropy**, **Shannon Entropy** and **Fisher information** to quantify the information content of the signals, offering insights into signal irregularity and complexity.

4.1.3 Fractal Dimensions

Petrosian and **Higuchi** fractal dimensions helped us capturing the irregularity and self-similarity of the signals, providing insights into signal complexity and dynamics.

4.1.4 Additional Features

The **Hurst exponent** characterizes long-range dependencies, memory, and self-similarity in the signals, offering insights into signal dynamics and complexity.
As presented in the paper [2], we used the three **Hjorth** parameters (activity, mobility, and complexity) introduced by Bo Hjorth in 1970 [6], commonly used in the analysis of electroencephalography signals for feature extraction.
Additionally, we extracted the first-order coefficient of an Autoregressive (AR) model fitted onto our captors to capture some interesting auto-correlation factors.

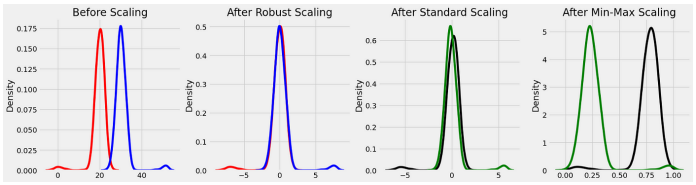

Figure 5. Comparison of known re-scaling techniques. See: Medium article

$$X_i' = \frac{X_i - \mu}{\sigma} = \frac{X_i - X_{\text{mean}}}{X_{std}}$$

StandardScaler

Figure 6. StandardScaler

$$X_i' = \frac{X_i - X_{\min}}{X_{max} - X_{min}}$$

Figure 7. MinMaxScaler

$$X_i' = \frac{X_i - X_{\text{med}}}{IQR} = \frac{X_i - X_{\text{med}}}{X_{75} - X_{25}}$$

Figure 8. RobustScaler

In the end, we also looked into "chaos theory" metrics like Lyapunov exponents to quantify the explosiveness of our signals and identify peculiar patterns in the sleep stages. However, these features were more computationally intensive.

4.2. Rescaling

Furthermore, most of the models that we implemented and that will be discussed further benefited from rescaling the pipeline. We tried StandardScaler 6, MinMaxScaler 7, and RobustScaler 8 methods from the scikit-learn library.

After trial and error, not all models benefited from rescaling, especially tree-based models like RandomForest or XGBoost.

4.3. Dimension reduction

As a result of this long extraction process, our final dataset contains **692 features** (128 for each EEG sensor and 23 for each accelerometer, plus 13 additional features).

In this particular case, dimensionality reduction and feature selection to find the most discriminative features and reduce the complexity of the model would have been very relevant.
We tried to perform dimension reduction by applying the most popular **PCA** (Principal Component Analysis) algorithm. The inner idea behind PCA is to express a high dimensional space into a lower one that will help decrease the complexity of time and space. After a few tests, it didn't improve the score of the model as depicted in Figure 9, so we simply kept all the features extracted. One issue lies in the very **non-linear** nature of most features we have implemented. Re-normalization using StandardScaler techniques from the scikit-learn library before applying PCA dimension reduction did not help select the most relevant and complementary features. Most Machine Learning models we used revolved around decision trees and were suited to handle a large number of features, as long as rank was preserved.

We attempted to visualize the most valuable features of our models. As depicted in Figure 10, the two most important fea-

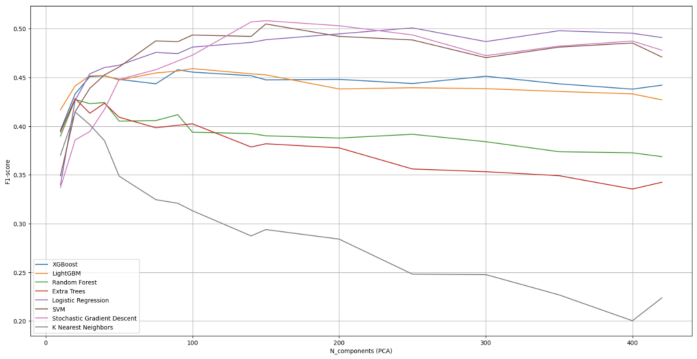

Figure 9. PCA number of kept features impact on model F1-score

tures for the XGBoost model were *EEG 3 Petrosian fractal dimension* and *EEG 3 bandbinpower beta* e.g the power spectral density in the β range and the pretosian fractal dimension of the ground sensor on the frontal band.

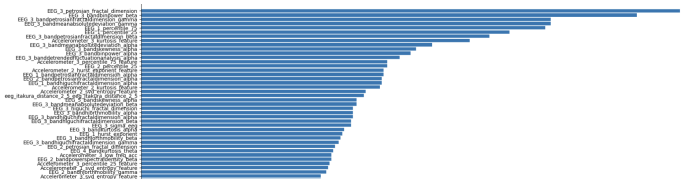

Figure 10. Feature Importance for XGBoost

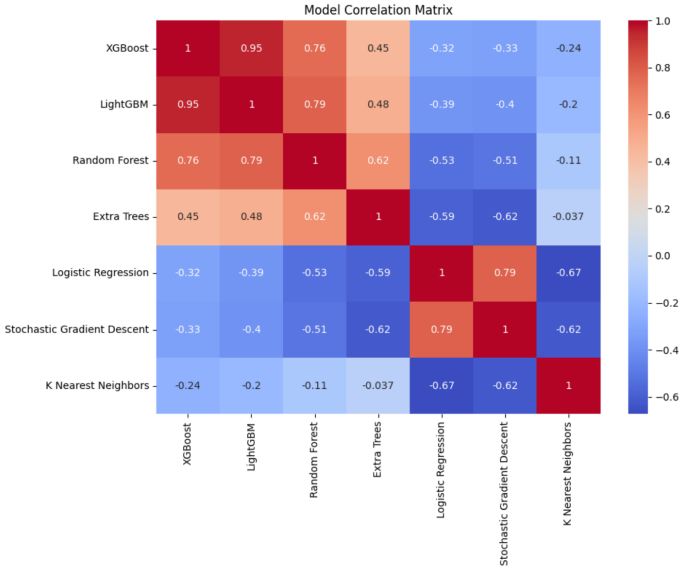

Figure 11. Base Model Correlation

5. Final model architecture

The main idea behind the final architecture of the model that we delivered, is to get the most out of the diversity of our predictors via a method known as **Ensemble Learning**. Our goal was to create a **Stacking Classifier** that combines intelligently the predictions of multiple base models. A simple VotingClassifier (known as "Majority Voting") was not sufficient because over-confident models seemed to already agree when the features speaked for themselves, but eclipsed the wiser ones in some regimes, like N1 stage (see 17). The predictions of the base models were used to feed a final meta-model. To avoid training the

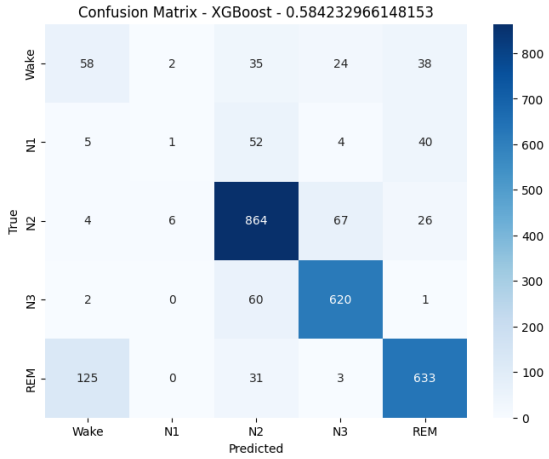

Figure 12. A confusion matrix example

supervised meta-model on the same training set as the base models (resulting in training leakage and overfitting), we use astute cross-validation. This was performed using a 2, 5 or 10-fold cross-validation, where the base models iteratively give predictions that are used as features to train the meta-model for future inferences. This technique leverages the bias/variance of each model to reduce the variance of the final prediction. Instead of using fixed decision weights per model, we aimed to intelligently assign weights based on the performance of each model. The architecture of this Stacking classifier has been schematized on Figure 13.

5.1. Base models

The base models are used to predict the sleep stages from our dataset enriched with data pre-processing and feature extraction. We used in total eight different base models, including seven Machine learning models - **XGBoost**, **LightGBM**, **Random Forest**, **Extra Trees**, **Logistic Regression**, **Stochastic Gradient Descent**, and **K Nearest Neighbors** and one deep learning model - **LSTM**.

For the Deep learning model, we used the library Keras to create the deep neural network architecture. An architecture of two LSTM layers and a dense output layer was used. We trained the model on the extracted features and evaluated its performance using the same metrics as the machine learning models. To avoid overfitting We used a learning rate scheduler and an early stopping with a patience parameter of 15 epochs without improvement on the validation loss.

Overall, while the deep learning models did not outperform the machine learning models, they still showed promise for automated sleep staging. Further exploration of architecture, as well as data preprocessing improvements, could potentially lead to even better performance and reduced overfitting.

5.2. Meta model and final output

With the predictions stacked of each of the eight base models, we trained a final or so-called *meta* model on these predictions. Each individual base model not only predicts the class label for the input but also provides probability scores or confidence levels for each class. These probability scores represent the estimated likelihood of the input belonging to each class. The meta-model works like a smart "soft" Voting Classifier where the final predic-

tion is made by considering a flexible (i.e. not "static") weighted average of the probability scores for each class across all base models by leveraging the above model (anti-)correlations. The class with the highest aggregated probability score is chosen as the final predicted class by soft voting. After several trial and error iterations, we found that a **Random Forest** model served as an efficient meta-model for the last part of the Stacking Classifier. 2 (resp. 5) stratified folds worked as the best combination to maximize the performance of the base models on 50% (resp. 80%) of the training set, and that of the meta-model on the other 50% (resp. 20%) validation set. Very surprisingly, over-training the meta-model to fully leverage the models' diversity (thus under-training the base models) provides comparable results as the other way around. For that exact reason, 10 folds under-trained the meta-model and largely degraded the overall performance.

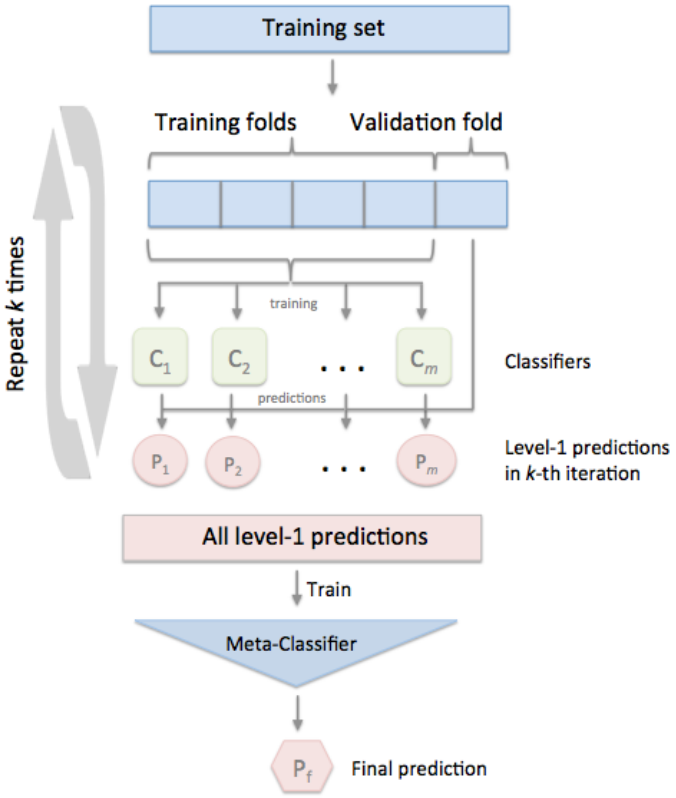

Figure 13. Staking Classifier architecture

The meta model hyperparameters tuning and evaluation was performed using the same method as the individual models. Our final submission with this architecture achieved a promising F1-score of 67.5%. This demonstrated the potential of ensemble learning to improve the accuracy of automated sleep staging.

Overall, the Stacking Classifier proved to be an effective technique for combining the strengths of multiple models and producing more reliable predictions. Further exploration and optimization of the Stacking Classifier could lead to even better performance in automated sleep staging.

6. Cross-Validation

To ensure robust evaluation of our machine learning and deep learning models, we employed 5-fold and 10-fold Stratified Cross-Validation method. The process of K-Fold splitting has been schematized in Figure 14.

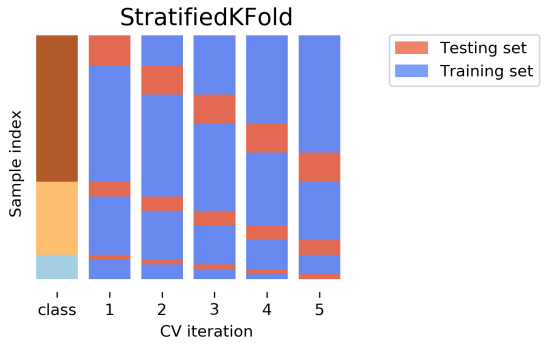

Figure 14. (Stratified) Cross Validation F1-scoring

Stratifying the splits ensures that we keep the same class distribution in every fold as the original dataset. We utilized the StratifiedKFold function from the scikit-learn library.

As a last step of the competition, we fine-tuned the hyperparameters of each model. Optimal hyperparameters are summarized in the table below. The **class weight** argument corresponds to the weights added to each model to ensure the balance between the classes. The metrics used for evaluation included overall accuracy, precision, recall, F1 score, Cohen's kappa coefficient, and confusion matrices. Across these 5 different folds, these metrics provided a robust estimation of the models' performance and helped in identifying any potential issues such as overfitting or underfitting.

Model	**Hyperparameters**
XGBoost	`max_depth=5, learning_rate=0.05, n_estimators=600, subsample=0.7, sample_weight, tree_method='hist', device='cuda'`
LightGBM	`learning_rate=0.05, max_depth=3, n_estimators=400, class_weight`
Random Forest	`class_weight`
Extra Trees	`n_estimators=50, class_weight`
Logistic Regression	`C=0.1, max_iter=1000, class_weight`
SGD Classifier	`alpha=0.001, max_iter=1000, loss='log_loss', penalty='l1', class_weight`
KNN	`n_neighbors=7, weights='uniform'`

This allowed us to obtain a more reliable estimate of the models' performance by averaging the evaluation metrics across the 5 folds. This approach provided valuable insights into the generalization capabilities of the models and their effectiveness in automated sleep staging. This is especially important, as Kaggle

submissions only benchmark 50% of the final testing data. Resorting to fine-tuning at the very last moment, protected us against covariate shift, a pitfall where one could expect a significant shift in the feature's distribution of the final Kaggle scoring dataset.

7. Post-Processing

At this stage, we obtain relevant predictions on 30-second measurements. However, there seems to be one characteristic of our data that we did not exploit. Sleep data is intrinsically context-based and thus can be viewed as a time series. Looking at doctor consensus-based hypnograms, there seem to be (almost) impossible states' transitions. For instance, changing from state N3 (deeply asleep) to suddenly awake is indeed very common. However, transitioning from awake to deeply asleep (N3) is non-realistic, counter-intuitive, and scarce in the data (see 16). Yet, our model seems to be unconscious of this, with its scope limited to 30 second epochs:

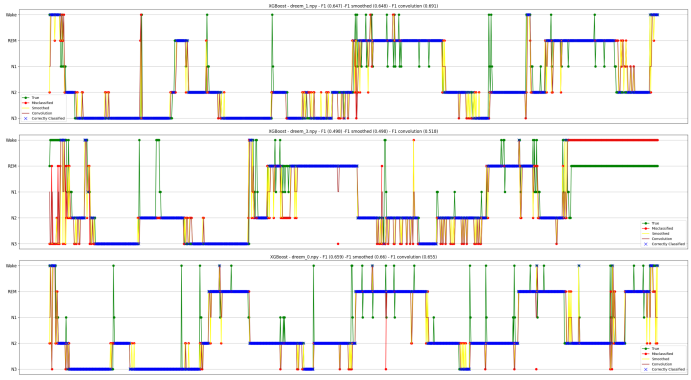

Figure 15. Night per night temporal prediction comparison

To further enhance this fact, a discrete Markov Chain representation of the transition between each sleep stage on the training set yields:

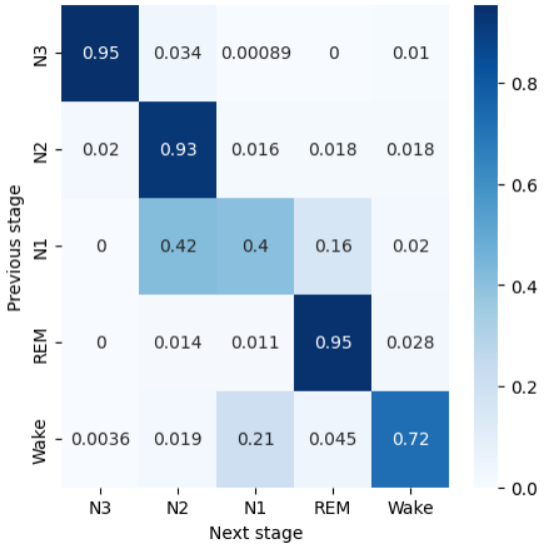

Figure 16. Aggregated transition matrix from the training set

As we can see, and for human-intelligible reasons, some transitions never happen in real life. These are:

1. REM to N3
2. N1 to N3
3. N3 to REM

Therefore, we manually applied thresholds to the output probabilities of the ensemble model to ensure that the predictions aligned with physiological expectations and known sleep stage patterns. By setting appropriate thresholds, we aimed to refine the predictions and mitigate potential inconsistencies or outliers in the automated sleep staging process.

Secondly, we aimed to train a case-by-case smart corrective model to align our prediction with the transition matrix above. We implemented a Hidden Markov Model (HMM), an unsupervised machine learning model fed with the estimated target transition matrix to further refine the predictions generated by the ensemble of machine learning and deep learning models. The HMM leveraged the temporal dependencies between consecutive sleep stage predictions, which are inherent in sleep data characterized by sequential epochs. This approach allowed us to incorporate the temporal context of sleep stage transitions, enhancing the coherence and consistency of the final sleep stage predictions. Nevertheless, some hyper-parameter fine-tuning is still needed to improve this method, and vastly underfits our output data.

As a last try, we tried to smooth out the jumps by relying on discrete convolution low-frequency band-pass filters, a method commonly found in signal processing to avoid too sudden, 'high frequency' jumps in our predictions. With a well-calibrated window of 2 (i.e. a range of plus-minus 1 minute inspection). This turned out to be the most efficient and low-cost method to increase our score by an additional 2 to 5% margin. Note, however, that this method breaks the causality of our predictions, as it uses a bit of the future to predict the present, making this system slightly unrealistic in a production live environment, yet very convenient for a Kaggle competition.

Overall, the post-processing phase played a crucial role in refining the predictions generated by the ensemble of models, ultimately contributing to the reliability and interpretability of our automated sleep staging system.

8. Results

Our automated sleep staging system achieved promising results, with an overall F1 score of 67.5% using the Stacking Classifier architecture that we built. The individual machine learning models achieved F1 scores ranging from 60% to 67%, with XGBoost and LightGBM performing the best among the machine learning models.

The LSTM deep learning model achieved generalization F1 scores of 62%. While our tries on deep learning models did not outperform the machine learning model ones, these models still showed promise for automated sleep staging as a complementary approach, only beneficial to theoretically boost ensemble methods. The use of 5-fold Stratified Cross-Validation enhanced the robustness of our model evaluations and contributed to the overall reliability of our automated sleep staging system. The evaluation metrics used during cross-validation, including F1-Score, precision, recall, accuracy, Cohen's kappa coefficient, and

confusion matrices, provided us with a thorough understanding of the models' performance across different folds.

Overall, our automated sleep staging system demonstrated the potential of machine learning and deep learning techniques for accurately and efficiently classifying sleep stages. Further exploration and optimization of the models could lead to even better performance and contribute to the development of more reliable and accessible sleep staging systems.

References

[1] The dreem headband - beacon biosignals. https://beacon.bio/dreem-headband/. Accessed: 2024-01-12. 1, 2

[2] Khald Ali I. Aboalayon, Miad Faezipour, Wafaa S. Almuhammadi, and Saeid Moslehpour. Sleep stage classification using eeg signal analysis: A comprehensive survey and new investigation, 2016. 1, 2, 3

[3] Sylvain Baillet, Forrest Sheng Bao, Xin Liu, and Christina Zhang. Pyeeg: An open source python module for eeg/meg feature extraction. *Computational Intelligence and Neuroscience*, 2011:406391, 2011. 2

[4] Tapan Gandhi, Bijaya Panigrahi, and Sneh Anand. A comparative study of wavelet families for eeg signal classification. *Neurocomputing*, 74:3051–3057, 10 2011. 3

[5] Philip Gouverneur, Frédéric Li, Wacław Adamczyk, Tibor Szikszay, Kerstin Luedtke, and Marcin Grzegorzek. Comparison of feature extraction methods for physiological signals for heat-based pain recognition, 07 2021. 2

[6] Bo Hjorth. Eeg analysis based on time domain properties, 1970. 3

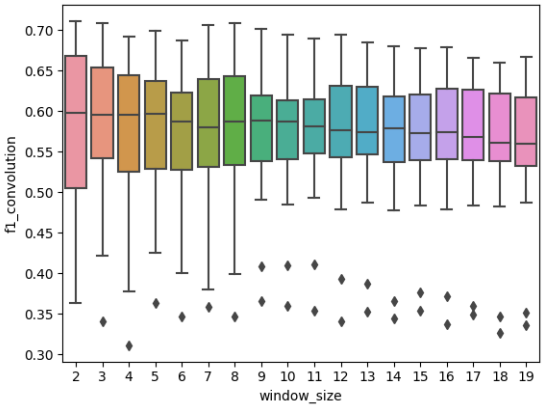

Figure 18. Convolution smoothing F1-score for each model against the range of the rolling window

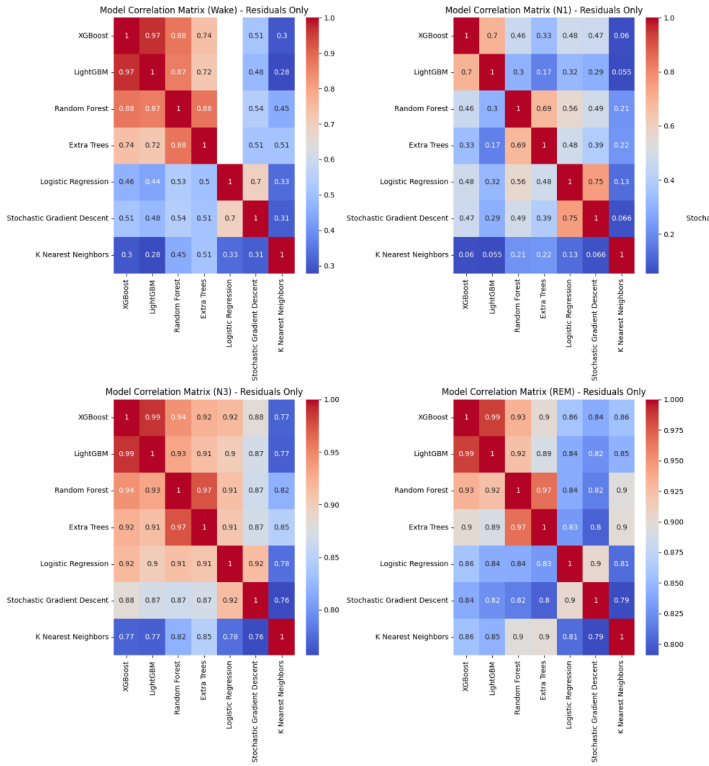

Figure 17. Decomposition of the confusion matrix on miss-predicted stages

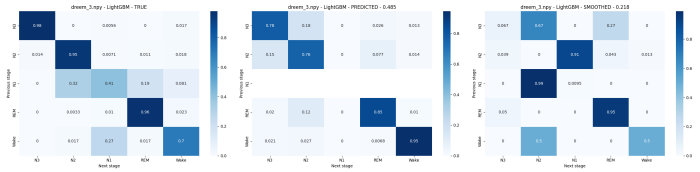

Figure 19. Transition matrix for the fitted LGBM model on a given night (Left: theoretical, Middle: prediction, Right: HMM smoothed prediction)

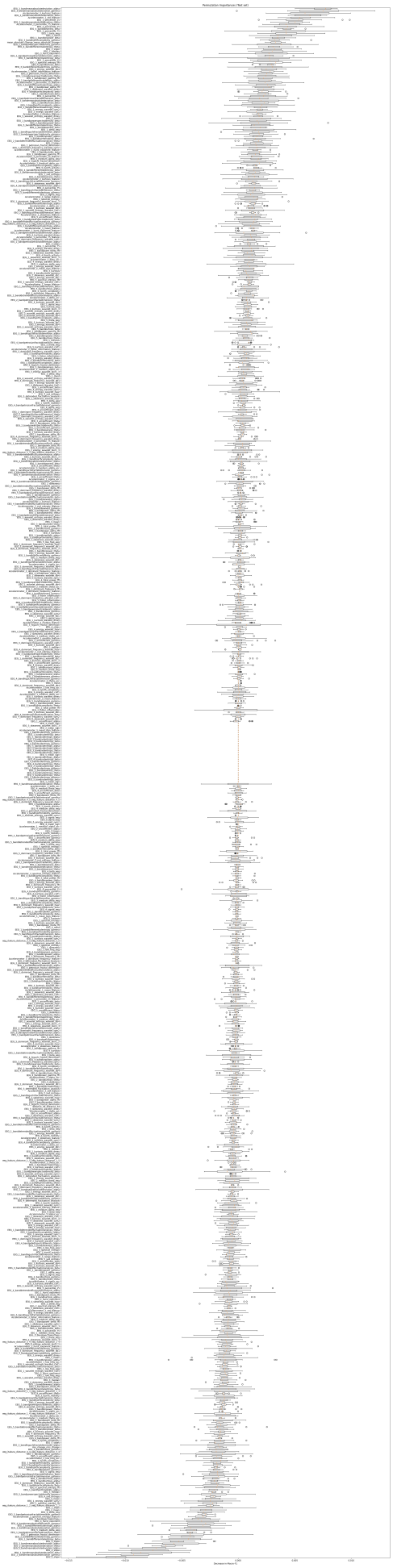

Figure 20. Random Forest Permutation Plot - 20 permutations