# Child Sleep State Detection

Wojciech Neuman, Enrico Maria Marinelli, Tomasz Makowski

## 1 Introduction

The problem we tried to bring to light is the sleep state detection. In combination with our goal, we took part in a Web-based challenge published by the Child Mind Insitute on Kaggle website, an online platform hosting Machine Learning competitions organized by users or larger organizations, such as Child Mind Institute, that has also provided important money prizes for top-scorers teams. Further detailed information about the problem and the ways that were tried to solve it are described in details in the following sections.

### 1.1 Competition rules and limitations

Being a Code competition, the full test set was hidden at the time of the challenge. Other rules and limitations was to disallow Internet connection for the Notebooks and having a pre-defined amount of memory and time for each submission set at 9-hours. The submission consisted in a *.csv* file compiled with the test dataset **wakeup** and **onset** predictions.

The competition lasted until Dec 6, 2023, unfortunately we had only few days to work on, with respect to the opening date Sep 5, 2023.

## 2 Dataset

### 2.1 General Description

The dataset includes approximately 500 multi-day recordings of wrist-worn accelerometer data, each annotated with two event types: **onset**, representing the beginning of sleep, and **wakeup**, denoting the end of sleep. We define a sleep phase as activity characterized by concrete instructions, detailed on the competition website.

Each data series provides continuous multi-day events for a unique experimental subject, with the length varying from a few days to over 3 weeks. A significant portion of the dataset posed challenges due to the removal of the accelerometer device from the patients. During these periods, the input data exhibited suspiciously little variation over an extended period, and events were not annotated for these intervals. It was paramount for us to abstain from making event predictions during these periods to avoid a high false-positive rate.

### 2.2 Files and Field Descriptions

The training dataset consisted of two files `train_series.parquet` and `train_events.csv`. In `train_series.parquet`, each series represents a continuous recording of accelerometer data for a specific individual over multiple days. Key attributes include:

- `series_id`: A unique identifier for each accelerometer series.
- `step`: Index of a recorded data in the time series.
- `timestamp`: Datetime value.
- `anglez`: The angle of the arm relative to the vertical axis of the body.
- `enmo`: Represents the Euclidean Norm Minus One of all accelerometer signals. Negative values are rounded to zero.

`Anglez` and `ENMO` values are commonly used metrics in sleep detection. They are computed and described by the GGIR package being an R-package to process multi-day raw accelerometer data for physical activity and sleep research.

The `train_events.csv` file contains sleep logs associated with the series in the training

set, recording onset and wake events. Relevant details include:

- `night`: Index of 'onset'/'wakeup' pair. Each night may have at most one pair of events.
- `event`: Indicates the type of event, whether it's an 'onset' or a 'wakeup'.
- `series_id, step, timestamp`: Columns corresponding to the `train_series.parquet` file.

### 2.3 Visualize the Data

Below, we present the visualization of the **enmo** and **anglez** values for one of the patients. On the first 2 figures we can observe that the *onset* and *wakeup* events are evenly distributed—the accelerometer was worn during the whole recording.

However, most of the series have missing events, visualization of one of these series is below, on the third and forth figure.

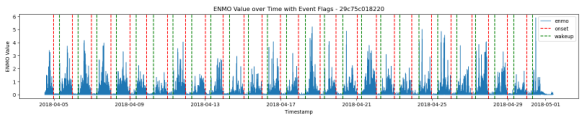

**Figure 1:** Visualization of **enmo** for a well-recorded series

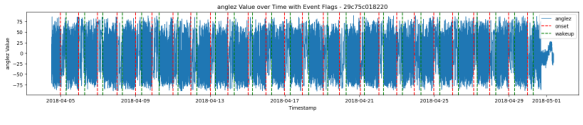

**Figure 2:** Visualization of **anglez** for a well-recorded series

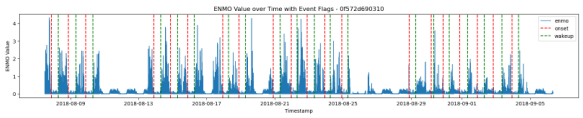

**Figure 3:** Visualization of **enmo** for a series with missing events

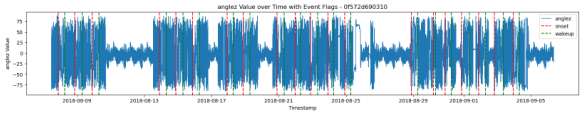

**Figure 4:** Visualization of **anglez** for a series with missing events

## 3 Data preparation

### 3.1 Selection of series

As mentioned in the previous section, a considerable portion of the series in the dataset exhibits incomplete recordings, with identical values not consistently registered. Recognizing that such incomplete series might negatively impact prediction efficiency, a strategy was devised to curate a more useful subset of series for training purposes. The approach involves two key steps:

1. **Data Cleaning:** In the first step, the data in each series is systematically pruned, starting from the first occurrence of a NaN value onward. This ensures that the series under consideration maintains consistency and reliability in the recorded values.
2. **Selection Criteria:** To further refine the dataset, only series with a duration of at least the last 5 days are retained. This selection criterion aims to enhance the visibility of patterns within the data, as shorter series may not allow good pattern recognition.

As a result, we successfully distilled the dataset from an initial count of **277** series to a refined subset of **147**, which served as the foundational dataset for training our models.

### 3.2 Feature Engineering

In the given data there where only two important features that changed as the time went by. The models trained only on these features (anglez, enmo) returned highly unsatisfied results - the model could not detect the change of sleeping state. To enhance the model's ability to understand patterns and relationships in the data, there was need of creating the new features created based on the values of two crucial, given features.

Let's break down the key operations done during feature engineering phase of code:

- Time-related Features:

  The timestamp column is converted to datetime format, allowing for easy manipulation of time-related information. Two new columns, hour and half_hour, are created. The hour column extracts the hour information from the timestamp, while half_hour represents the time in half-hour intervals.

- Anglez and Enmo Interaction:

  A new feature, anglez_times_enmo, is introduced. It represents the absolute value of the 'anglez' column multiplied by the 'enmo' column. This interaction feature captures a combined effect of these two variables.

- Temporal Differences:

  anglez_diff and enmo_diff represent the differences between consecutive samples for 'anglez' and 'enmo', respectively. These differences provide information about how these values change over time.

- Rolling Statistics:

  Rolling mean, max, and standard deviation are computed for both 'anglez' and 'enmo' features, with a specified rolling window of periods. These rolling statistics capture trends and variations in the data over the specified time window. For example, anglez_mean_rolling represents the mean of 'anglez' within the rolling window.

- NaN Handling:

  For the temporal differences and rolling statistics, missing values (NaN) introduced by the diff and rolling operations are handled using forward-fill (ffill) and backward-fill (bfill) methods. This ensures that each sample has a value for these new features.

The purpose of these operations is to create informative features that capture relevant patterns and dynamics in the data, making it easier for machine learning models to learn and generalize well on the task of sleep state detection. Temporal differences and rolling statistics provide insights into how signals change over time, while the interaction feature (anglez_times_enmo) offers a combined measure of two important variables. These engineered features contribute to a more comprehensive representation of the underlying patterns in the data, ultimately improving the model's performance.

### 3.3 Fixed window sequences

Has also been attempted an approach different from the creation of new features. A simpler idea was to have a **fixed window** of time-series array. The final resulting array would have been with three shapes `(number_of_sequences, window_size, 2)`[1]. We define as sequence a continuous series of "awake" or "sleeping" labeled samples. In the function used for building were discarded entire sequences smaller than the window and part of sequences greater than the window threshold, keeping only the first `window_size` samples[2]. Hence every sequence had only one label, the one represented by every sample in the sequence.

Although this data management was not a good-fitting with the classification and submission requested in the rules of the challenge. For this reason has been abandoned in favour of the Feature Engineering approach.

## 4 Model creation and training

### 4.1 Ensemble Models

In this study, we employed ensemble methods to address this challenging problem. The ensemble model was crafted by combining various classifiers, namely the LightGBM classifier, Gradient Boosting Classifier, Random Forest Classifier, and XGBoost Classifier. This ensemble approach leverages the diverse strengths of individual models to enhance overall predictive performance.

To optimize the performance of LightGBM, Random Forest and XGBoost classifiers, a thorough hyperparameter tuning process was conducted using grid search. This process systematically explored various combinations of hyperparameters to identify the optimal set that maximizes the models' accuracy and robustness. For Random Forest, hyperparameters such as maximum depth, minimum samples per leaf,

[1] 2 corresponds to the features *enmo* and *anglez* for each sample.

[2] The dataset could have been augmented with a moving window to gain all the possible information, but there were not this necessity.

and the number of estimators were fine-tuned. Similarly, for XGBoost, parameters including maximum depth, minimum child weight, subsample ratio, column subsample ratio, learning rate, and the number of estimators were optimized.

After the training of the model the feature importance analysis was performed. The analysis helped to highlight the most important features, which can be seen on the plot below.

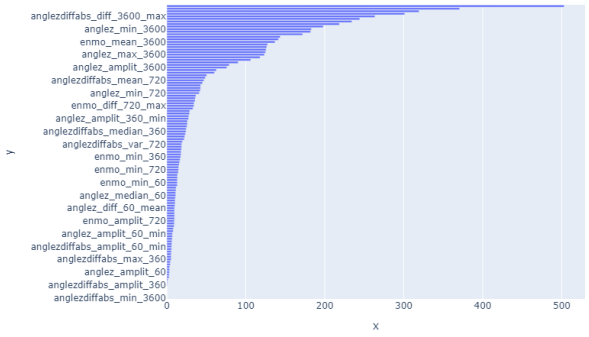


**Figure 5:** Feature importances

Notably, certain features, such as 'anglezdiffabs_min_3600' and 'anglezdiffabs_amplit_720,' demonstrated negligible contributions. These insights provide a deeper understanding of the model's decision-making process and guide further refinement.

In conclusion, the ensemble approach, coupled with hyperparameter tuning, offers a robust solution for accurately detecting sleep-related events. The combination of LightGBM, Gradient Boosting, Random Forest, and XGBoost classifiers leverages the strengths of each model to create a powerful and accurate ensemble, showcasing its effectiveness in real-world problem.

### 4.2 LSTM-Convolution Neural Network

The problem has been tackled completely differently with a Deep Learning model. The model used presents a very complex structure: is composed of two main blocks, identical in the form but varying in the number of neurons, and a final block for extracting the output. The main blocks are structured with a **1D Convolutional** layer, Batch-Normalization and Activation layers, then 3 **residual blocks** and an Attention layer followed by an **LSTM** layer. This scheme is repeated twice with the first main block twice the size of Conv1D and LSTM neurons.

The last block is composed by a Conv1D layer, larger in terms of neurons, a GAP layer and the dense layer for the output.

Has been used a Binary Cross Entropy loss with AdamW as optimizer.

This model was completely fitting with the data in Section 3.3, but for the reasons previously described, has only been tested with the other method.

## 5 Results

### 5.1 Evaluation

The metric which was used to evaluate the submissions was based on the average precision of detected events. It was averaged over timestamp error tolerance thresholds, averaged over event classes.

Each detection is matched with the real event within error tolerances. There were given the tolerance thresholds which were in minutes:

**{1, 3, 5, 7.5, 10, 12.5, 15, 20, 25, 30}**

or in steps:

**{12, 36, 60, 90, 120, 150, 180. 240, 300, 360}**

Each event gets a score after being compared with the ground-truth event. The result is given based on the fact in which set does the difference fall. To make things more clear, it is presented in the table 1 below.

The implementation of metric can be find here.

The evaluation process comprises three main steps: Assignment, Scoring, and Reduction.

**Assignment:**
During assignment, predicted events are aligned with ground-truth events. This is done within specific criteria, considering event type, tolerance, and series ID. Ground-truth events

Table 1: Scoring explanation

| Difference (in minutes) | Score |
|---|---|
| [0; 1) | 1 |
| [1; 3) | 0.9 |
| [3; 5) | 0.8 |
| [5; 7.5) | 0.7 |
| [7.5; 10) | 0.6 |
| [10; 12.5) | 0.5 |
| [12.5; 15) | 0.4 |
| [15; 20) | 0.3 |
| [20; 25) | 0.2 |
| [25;30) | 0.1 |
| >=30 | 0 |

are matched with the highest-confidence, unmatched predictions within the allowed tolerance. It's important to note that not all ground-truths may find a corresponding prediction, and vice versa. Nonetheless, these unmatched instances are still considered in the subsequent scoring.

**Scoring:**
The scoring phase involves computing an Average Precision (AP) score for each combination of event, tolerance, and series ID. For each set of predictions and ground-truths within the same event-tolerance-series_id group, the AP score is determined. This score is calculated based on precision and recall metrics, utilizing a precision-recall curve generated by varying confidence score thresholds. True Positive (TP) and False Positive (FP) are assigned to matched predictions above the threshold, while unmatched predictions and ground-truths are labeled as False Positive (FP) and False Negative (FN), respectively.

**Reduction:**
In the reduction step, the multiple AP scores obtained are averaged to produce a singular overall score. This involves averaging scores first over tolerance and then over the event.

Submission File: To participate, a file that predicts each event in a series, specifying the event type, the step where it occurred, and a confidence score for that prediction had to be submitted. The file format includes a header and follows the structure:

Table 2: Example submission file

| **row_id** | **series_id** | **step** | **event** | **score** |
|---|---|---|---|---|
| 0 | 038441c925bb | 100 | onset | 0 |
| 1 | 038441c925bb | 105 | wakeup | 0 |
| 2 | 03d92c9f6f8a | 80 | onset | 0.5 |
| 3 | 03d92c9f6f8a | 110 | wakeup | 0.5 |
| 4 | 0402a003dae9 | 90 | onset | 1 |
| 5 | 0402a003dae9 | 120 | wakeup | 1 |

## 5.2 Model Performances

In the table given below there are presented the performances of our models.

Table 3: Performance of models

| **Model** | **Private score** | **Public score** |
|---|---|---|
| GradientBoostingClassifier | 0.533 | 0.514 |
| RandomForestClassifier | 0.527 | 0.499 |
| XGBClassifier | 0.534 | 0.502 |
| LGBMClassifier | 0.518 | 0.493 |
| EnsembleAvgProbaClassifier | 0.557 | 0.535 |
| LSTM-Conv | n/a | n/a |

In the first row of the table we can see the model which performance was measured. In the second column the results obtained on the shared subset of test set and in the last column the final score performed on the whole test dataset, which was uploaded after the competition had ended.

All of the models present similar performance. The score is around 0.5. That means that the model trained by us detects the moment of falling asleep and waking-up with the average error of around 10 minutes.

In the final analysis, the EnsembleAvgProbaClassifier stood out as the best performer among our models. It excelled in the tricky task of spotting when people fall asleep or wake up, showing an average error of around 10 minutes. This suggests the ensemble model is really good at picking up on subtle changes in sleep patterns. That is why it was our final choice as the submission to the Detect Sleep States competition created by Child Mind Institute.

# 6 References


1. **Child Mind Institute**.
   https://childmind.org/

2. **Detect sleep states competition**
   https://www.kaggle.com/competitions/child-mind-institute-detect-sleep-states/overview

3. **GGIR package**
   https://cran.r-project.org/web/packages/GGIR/vignettes/GGIR.html#4_Inspecting_the_results