Scuola universitaria professionale
della Svizzera italiana
SUPSI

University of Applied Sciences and Arts of Southern Switzerland
Department of Innovative Technologies

---

Applied Case Studies of Machine Learning and Deep Learning in Key Areas II

# FINAL PROJECT - SPINDLE CLASSIFICATION

Davide Gamba
davide.gamba@student.supsi.ch

Nathan Margni
nathan.margni@student.supsi.ch

Federico Weithaler
federico.weithaler@student.supsi.ch

Professor: Dr. Francesca D. Faraci
SUPSI, Lugano Switzerland

28/07/2023

# Table of Contents

# 1. Introduction

## 1.1 Context and objective

This project focuses on building a classification (detection) model that can extract spindle sleep patterns from a dataset of sleep feature measurements, specifically a whole night Polysomnography. Spindle sleep is a type of non-REM sleep that has been linked to memory consolidation and cognitive processing.

"A sleep spindle is a train of distinct waves with frequency in the range of 11 to 16 Hz and with duration $\geq$ of 0.5 s, usually maximal in amplitude using central EEG derivations. Spindles are a distinctive pattern of sleep stage NREM2, and can also be present in NREM3; their identification assists the sleep scoring procedure."[1]

Our primary goal is to develop a robust machine learning model that accurately identifies spindle sleep from the available sleep feature measurements that we extract using a variety of machine learning techniques, including supervised learning and feature engineering. We are working with multiple files to form single dataset for our model, as we believe that this approach allows us to build a precise and accurate data structure that it's containing just the information that we need during training, testing and evaluation.

## 1.2 Data extraction

The dataset that we are using is available to be downloaded, at the following link:

Dataset download (Zenodo.org)

Note: The first excerpt, (EDF file) did not work properly because of its metadata. Therefore, the metadata from the second excerpt was copied onto the first one to make it functional and then the frequency info was modified manually with python.

Excerpts of eight Polysomnographic recordings, belonging to patients with various sleep pathologies, are saved as EDF files. The labeling is composed of annotations from two experts in the field, in order to build the model. Additionally, there is an auto-computed labeling available. The first expert analyzed only 1000 seconds (16 minutes and 40 seconds) out of the total 30 minutes of data for all eight patients. On the other hand, the second expert analyzed the entire 30 minutes of data, but only for the first six patients. Therefore, we merge the three scorings using a union over the labels to reduce the possibility of losing important information. We developed a function that accepts three different settings to adjust its rigidity when merging the data. The first setting assigns a positive label only if the positive label covers the entire window (0.5 seconds or 100 data points). The second setting considers the window to be 50% covered, and the last setting (the least restrictive) considers at least one data point as positive.

While the paper[1] ignores the last two subjects, we have decided to keep them and use the data from the first expert and the auto-computed data to form the labeling. All three scorings are considered for the first six patients, while only the labels from the first expert and the auto-features are considered for the last two patients. We then load all the data, perform exploratory data analysis, address unbalanced features through resampling, conduct feature engineering, and resize the data in order to prepare it for training.

The pipeline for extracting information from the files is structured as follows: signals are loaded from the "excerpt{n}.edf" files. Annotations are read from the ".txt" annotation files named "Visual scoring 1," "Visual scoring 2," and "Automatic detection." The data is then pre-processed and resampled to match the sampling frequency of 200 Hz using the SciPy resampling method and a static list of given frequencies. The channels of the EEG electrodes include:

- FP1 and FP2: Frontopolar (forehead) regions, respectively, left and right.
- CZ and CZ2: Central regions (usually around the top of the head), possibly duplicates or from different montages

| Subject | Scored by expert 1 | Scored by expert 2 | Scored by automatic detection | Union |
|---|---|---|---|---|
| ***1*** | 30 | 119 | 159 | **214** |
| ***2*** | 35 | 53 | 94 | **124** |
| ***3*** | 4 | 48 | 45 | **70** |
| ***4*** | 53 | 25 | 44 | **105** |
| ***5*** | 29 | 86 | 69 | **126** |
| ***6*** | 40 | 88 | 78 | **137** |
| ***7*** | 10 | 0 | 23 | **29** |
| ***8*** | 39 | 0 | 106 | **129** |

- O1 and O2: Occipital (back of the head) regions, left and right, respectively

The table shows the positive labelling done by the different sources. Using the first method (the most restrictive one) we achieve a similar count to the one showcased in the paper [1] of reference.

After several experiments, we noticed that Patient 1 was particularly problematic. The inclusion of this particular patient in our cross-validation process and, more broadly, in the training of our models, yielded notably inferior performances and diminished scores. After a visual inspection (shown in the figure below), we realized that the signals of Patient 1 were very different from those of all other patients. These discernible dissimilarities prompted us to contemplate the omission of Patient 1 from our dataset, which, remarkably, resulted in a substantial enhancement in the model's performance, with the f1-score experiencing a noteworthy increment of at least 10 percentage points.

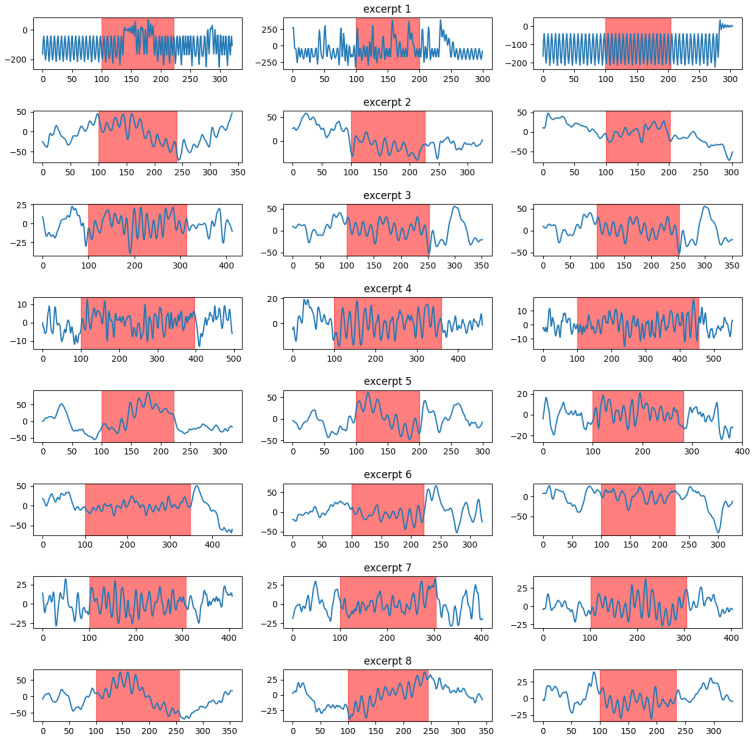



Figure 1: Sleep Spindles by Patient

# 2. Feature Extraction

To compute most of the features we take in consideration the values of the whole signal, for example for following features: maximum, minimum, variance, standard deviation, etc. For other features focused on the sliding window, we have prepared two time window, respectively of, 0.5 seconds, and a wider sample, so 1.75 seconds of data before and after are added to the original window to achieve a total window of 4 seconds.

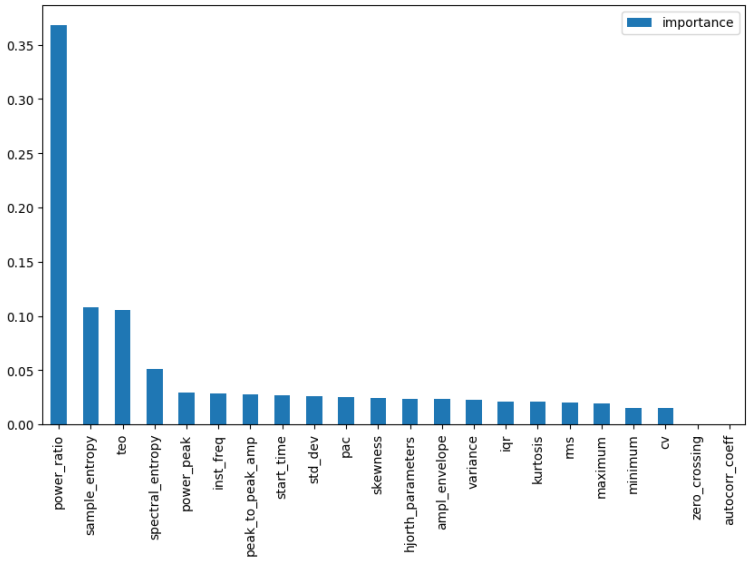
## 2.1 Features

A function manages the computation of all the features:

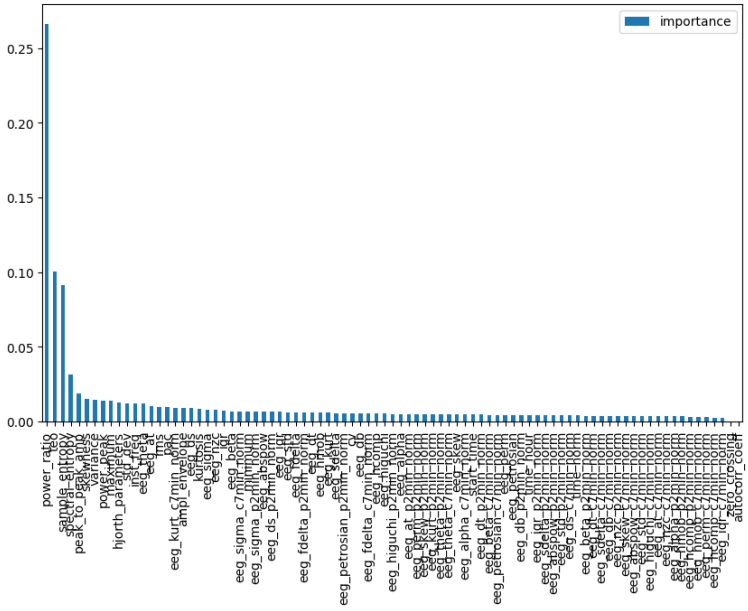
- Sample entropy: Measures signal complexity based on patterns of self-similarity.
- Maximum: The highest value of the signal.
- Minimum: The lowest value of the signal.
- Variance: Measures the spread of the signal.
- Std dev: The standard deviation of the signal.
- Kurtosis: Measures the "peakedness" of the distribution.
- Skewness: Measures the asymmetry of the distribution.
- Power peak: The frequency with the highest power in the signal.
- Power ratio: The ratio of the power of a specific frequency band to the total power of the signal.
- Inst freq: The instantaneous frequency of the signal.
- Ampl envelope: The amplitude envelope of the signal, which describes how the amplitude of the signal changes over time.
- PAC (Phase-Amplitude Coupling): A measure of the coupling between the phase of a low-frequency signal and the amplitude of a high-frequency signal.
- IQR (Interquartile range): The range between the first and third quarterlies of the signal distribution.
- Zero crossing: The number of times the signal crosses the horizontal axis.
- Spectral entropy: Measures signal complexity in the frequency domain.
- Autocorr coeff: Measures the similarity between observations to detect non-randomness and repeating patterns.
- Hjorth parameters: Descriptors used in EEG analysis to measure signal power, frequency, and similarity to a pure sine wave.
- Activity: A measure of the signal power.
- TEO (Teager Energy Operator): Nonlinear energy tracking operator used in speech and audio signal processing.
- Peak to peak amp: Measures signal variability as the difference between the maximum and minimum amplitude.
- RMS (Root Mean Square): Stastistical measure of the power of a varying quantity.
- CV (Coefficient of Variation): Provides a normalized measure of the variability of a signal relative to its mean amplitude.

## 2.2 Features Importance

Here you can find the feature importance plots calculated by our Random Forest Classifiers:




(a) Feature Importance




(b) Feature importance with YASA features

Figure 2: Reward and Success with low learning rate over more than 4M steps

## 2.3 Experiment: Yasa Sleep Staging Features

Considering that sleep spindles often occur during the N2 phase of sleep [2], we thought that augmenting the models with features extracted by YASA [3] for its SleepStaging classification could prove advantageous. These newly computed features were subsequently merged with the aforementioned features. Despite the Random Forest Classifier designating many of YASA's computed features as highly important, the overall performance of the model exhibited a marked decline. In the subsequent sections, we will provide a detailed presentation of the results. Notably, we refrained from conducting feature selection from the YASA features. While this could potentially

serve as a prospective improvement, the considerable degradation in results dissuaded us from pursuing this avenue.

# 3. Model buildings (without yasa features)

## 3.1 Sequential NN

We first tried to classify the target label using a Neural Network with imbalanced data, and weights on the labels with the correct proportion. A simple sequential architecture was set up and the model trained.

Table 1: Layers for Sequential NN

| |
|---|
| Dense: input, 64, relu |
| Dense: layer, 32, relu |
| Dense: output, 1, sigmoid |

The results of this model are obviously poor, since the data is not balanced and the model is really simple. We will improve in the following models our accuracy, starting with smote oversampling to balance the label.

Table 2: Scores for NN model

| **Metric** | **Score** |
|---|---|
| Accuracy | 0.237 |
| Precision | 0.089 |
| Recall | 0.395 |
| F1 Score | 0.131 |

From this point onwards all the models will be trained on a dataset, on which we performed an oversampling with the RandomOverSampler from Imbalanced-Learn, obtaining a 50/50 balance of the label.

## 3.2 Logistic Regression

To train the logistic regression model, as per the neural network, we scale the data. We then train the model with the LogisticRegression() from sci-kit learn library.

## 3.3 KNN (k-nearest neighbors)

Here also scaling was performed prior to the training with the same function that already performs cross validation (see chapter Cross Validation). For this K-nearest neighbors classifier we use also a model from the sci-kit learn library: "KNeighborsClassifier()".

## 3.4 Random forest

This model is the only one that doesn't need scaling, therefor we simply set the flag to False and the cross validation function runs with original structured data. We train with a Random Forest Classifier, also from the sci-kit library.

## 3.5 Sequential NN with SMOTE

To use the data with this model we first need to transform it with a standard scaler, as we did with previous models. This time we use SMOTE as scaler. Cross validation is performed to prevent data leakage in the testing procedure. The structure of the sequential neural network is the following:

Table 3: Layers for balanced Sequential NN

| |
|---|
| Dense: input, 256, relu |
| Dense: layer, 128, relu |
| Dense: layer, 64, relu |
| Dense: output, 1, sigmoid |

### 3.6 SVC (support vector machine)

SVC is particularly useful when working with high-dimensional data, as it is effective in separating data points into classes by maximizing the margin between the decision boundary and the closest data points. In this case it's not really beneficial for this reason, but we wanted to try it.

### 3.7 Model buildings (YASA included)

Models are also available with the same architecture, with YASA features appended to the dataset used in training. Sequential NN, Logistic Regression, KNN, and Random Forest are trained on this merged dataframe.

## 4. Results and Conclusions

### 4.1 Evaluation and result

#### 4.1.1 Cross validation

A point that we decided to include to try and improve our testing approach, is cross-validation. The dataset is split into multiple folds, and the model is trained and evaluated on each fold in turn. The performance of the model is then averaged across all folds to obtain a more reliable estimate of its generalization performance. With this different approach we can prevent any data leakage during the testing phase. An appropriate function is defined to train each model with the option to scale the data before training. The scaling is performed using the Standard Scaler of sci-kit learn.

#### 4.1.2 Result's table

Following the table with the results for each model, with the appropriate cross validation scores.

Table 4: Results for ML Models

| Model | Accuracy | Precision | Recall | F1-Score | ROC AUC |
|---|---|---|---|---|---|
| Sequential NN (not balanced) | 0.900 | 0.165 | 0.362 | 0.211 | - |
| Sequential NN with SMOTE | **0.983** | 0.302 | 0.303 | 0.293 | - |
| Logistic Regression | 0.976 | 0.640 | 0.337 | 0.416 | 0.666 |
| KNN | 0.946 | 0.306 | 0.653 | 0.392 | **0.805** |
| Random Forest | 0.977 | **0.676** | 0.391 | **0.447** | 0.693 |
| SVC | 0.943 | 0.314 | **0.655** | 0.381 | 0.803 |

Table 5: Results for ML Models with yasa features

| Model | Accuracy | Precision | Recall | F1-Score | ROC AUC |
|---|---|---|---|---|---|
| Sequential NN with SMOTE | 0.894 | 0.145 | - | 0.289 | - |
| Random Forest | 0.975 | 0.643 | 0.153 | 0.212 | 0.575 |

We used the patient 8 to test the best model, that patient was removed from the cross validation data, and the model that we consider to be the best (from the CV scores) finally trained on the whole data, excluding the patient 8 to which we then tested this final model.

Table 6: Random Forest and Logistic Regression on test patient 8

| Model | Accuracy | Precision | Recall | F1-Score | ROC AUC |
|---|---|---|---|---|---|
| Logistic Regression | 0.847 | 0.169 | 0.829 | 0.280 | 0.838 |
| Random Forest | 0.965 | 0.666 | 0.093 | 0.163 | 0.545 |
| SVC | 0.952 | 0.358 | 0.411 | 0.383 | 0.692 |

From these scores we can conclude that the models are probably overfitting on the patients, especially random forest that perform very badly on this particular patient. SVC instead performed relatively well, meaning that, even if with the CV had slightly worse scores, it is more robust when testing on different patients.

## 4.2 Conclusions and improvements on the original paper

Our focus was to perform a solid features extraction to build reliable models. We successfully computed many features that helped the model achieve a better results. Nonetheless, even though the overall results seem really good, looking at the accuracy, in reality they are not very promising, as we can see from the precision and f1 scores. Probably, over-fitting is present from the training set (and therefore the single patients). Also, a low recall score usually means that an imbalanced dataset is used. We achieved results for what concerns data cleaning and extraction. We are satisfied with our approach to the pre-processing. We took a different path, in respect to the reference paper, in choosing how and why we keep the subjects. We decided to remove completely the first person, as it was particularly different from the others. To take into account multiple-scorer ground truths, we tried to achieve a similar number of positive labels as described in the paper, but we also added a third opinion, the auto-computed labels. Another difference from the paper is how we split the data for training and testing. We made sure to split the data so that data from a patient present in the training set, would not be present in the testing set and vice-versa. This ensures that there is no data leakage and results are not biased. Some more hyperparameter tuning could be performed to improve the results.

### 4.2.1 Student's duty

We organized the work strictly to optimize the work that an individual can do in parallel to the other members. While one person was working on processing the signals, dummy data-sets and allocations were created to start working at multiple stages of the project all at once. During the last phase of the project, we worked together to "cross validate" all the different parts that we built individually. We helped each other during all the steps of the development, but mainly the tasks were divided as follows:

- Signal Processing: Federico Weithaler
- Feature Extraction and Selection: Davide Gamba
- Models and CV: Nathan Margni

This documentation has been edited by all of the members of the team.

# References

[1] S. Scafa, L. Fiorillo, M. Lucchini, *et al.*, "Personalized sleep spindle detection in whole night polysomnography," vol. 2020, Jul. 2020, pp. 1047–1050. DOI: 10.1109/EMBC44109.2020.9176136.

[2] Y. Wei, G. P. Krishnan, M. Komarov, and M. Bazhenov, *Differential roles of sleep spindles and sleep slow oscillations in memory consolidation*, Jul. 2018. DOI: 10.1371/journal.pcbi.1006322. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6053241/.

[3] R. V. PhD, *A simple and efficient sleep spindles detector*. [Online]. Available: https://raphaelvallat.com/spindles.html.