Master Thesis

Model-Based Approaches for Sleep Stage Classification using Time-Domain EEG Analysis

submitted to the

Institute of
Analysis and Scientific Computing
TU Wien

under the supervision of

Assistant Prof. Dr. Andreas Körner

by

Alexander Edthofer
Matriculation number: 01613479

Vienna, September 2023

Abstract

Classification of different levels of consciousness is of great importance in the diagnosis of sleep disorders. Semi-automated or automated strategies are advised to facilitate and accelerate the procedure. Different methods which achieve a good accuracy have been proposed in research, most of them are based on machine learning algorithms. However, none of them is in widespread use in the clinical or preclinical field so far. For acceptance in the medical sector, a model has to be interpretable, which many of these artificial intelligence-based models lack. Our approach aims to create an explainable model for understanding and analysing electroencephalogram (EEG) data.

Due to the complex structure of the brain, a modelling strategy that is based on rules and equations is not possible in neuroscience. Classical modelling combined with machine learning supports the development of a mathematical and computational framework for classification of sleep stages. For a given EEG signal, features are extracted from the signal that should predict the level of consciousness. The focus lies on entropy-based parameters, such as Permutation Entropy, Entropy of Difference and Kullback-Leibler Divergence, as well as Granger Causality is used. These are explained in every detail and used, in combination with statistical features and a personal parameter of the patient, for the creation of different machine learning models. They are trained, tested and compared regarding their performance and interpretability. The training dataset is the CAP Sleep Database. The implementation and tests are all run on MATLAB.

The results show that linear models, which are much more explainable and easier to interpret, can compete with more complex ones regarding the performance. The desired accuracy cannot be achieved, but it is presented, how enhancements of the models can improve the results. Therefore, this thesis has contributed to show that sleep stage classification does not have to rely on black-box modelling, but can also work with plain EEG-based models.

Kurzfassung

Diagnose von Schlafstörungen ist aufgrund der Regeneration des Körpers während des Schlafs von enormer Wichtigkeit für unsere Gesellschaft. Dies passiert mittels Untersuchungen im Schlaflabor und anschließender Schlafstadienklassifikation, welche manuell ausgeführt bis zu zwei Stunden dauern kann. Daher wird nach semiautomatisierten oder automatisierten Prozessen geforscht, um die Klassifikation zu beschleunigen. In den letzten Jahren ist diesbezüglich vor allem maschinelles Lernen federführend, was jedoch noch nicht in breiter klinischer Verwendung ist. In dieser Arbeit wird daher ein einfaches interpretierbares Modell zur Schlafstadienklassifikation gesucht, das Forschenden, Ärzt:innen wie auch Patient:innen leicht zu erklären ist.

Für Modellierung der Hirnaktivität gibt es aufgrund mangelnder Kenntnisse darüber keine beschreibenden Gleichungen, stattdessen müssen Parameter gefunden werden, die uns die komplexen Vorgänge im Gehirn modellieren. Grundlegende Messung dazu ist das Elektroenzephalogramm. Von diesem Signal werden Features berechnet, diese Arbeit beschäftigt sich vor allem mit Entropie-basierten, wie der Permutation Entropy, der Entropy of Difference und der Kullback-Leibler Divergenz, sowie der Granger Causality. Diese werden mathematisch detailliert erläutert. Abgesehen davon werden noch statistische Parameter sowie das Alter verwendet, um verschiedene Modelle basierend auf Algorithmen des maschinellen Lernens mit der CAP Sleep Database zu trainieren. Die Implementierung erfolgt in MATLAB.

Die Ergebnisse zeigen, dass einfache lineare Modelle mit komplexeren in Bezug auf Genauigkeit mithalten können. Diese sind leichter interpretierbar und daher auch besser Laien zu erklären. Durch Erhöhung der Feature Anzahl kann die Genauigkeit sogar noch verbessert werden, auch wenn die gewünschte Genauigkeit nicht erreicht werden konnte. Somit hat diese Arbeit dazu beigetragen zu zeigen, dass die Klassifikation von Schlafstadien nicht auf Black-Box Modellen beruhen muss, sondern auch mit einfachen EEG-basierten Modellen durchgeführt werden kann.

Acknowledgements

There are many people to whom I owe a big "Thank you!" after submitting this thesis.

First of all, I would like to thank my supervisor Andreas Körner for giving me the opportunity to study this research topic. He not only introduced me to the field, but also connected me with other researchers and supported me all the time, especially in the last months, the weekly meetings were very encouraging.

My colleagues have also been a great help, particularly Clara, Corinna, Iris, Lana and Daniel. Professional discussions and words of encouragement provided knowledge and a pleasant atmosphere.

I would also like to thank Matthias Kreuzer, Tom Fenzl and Michelle Franka from the Technical University of Munich for helpful discussions, especially about the physiology of the brain and the EEG.

To my parents and siblings I would like to express my gratitude, for understanding that despite my physical presence I was sometimes mentally absent.

A big thank you also goes to my friends whom I often annoyed with my constant talking about my Master's thesis. Nevertheless, they provided a lot of mental support and there were also very interesting conversations about sleep behaviour and different applications of machine learning.

Last but not least, I would like to thank my girlfriend Mina. Especially the last few weeks have been extremely exhausting and without her support, hugs, encouragement, proofreading, listening to me talk about the thesis, I probably would not have made it through the summer to hand in on time.

Vienna, September 2023 *Alexander Edthofer*

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, September 2023

Alexander Edthofer

Contents

List of Abbreviations

EEG	Electroencephalogram/electroencephalography
EOG	Electrooculogram/electrooculography
EMG	Electromyogram/electromyography
ECG	Electrocardiogram/electrocardiography
PE	Permutation entropy
EoD	Entropy of difference
KLD	Kullback Leibler divergence
CE	Cross entropy
GC	Granger causality
CAP	Cyclic alternating patterns
LDA	Linear discriminant analysis
SVM	Support vector machine
KNN	K nearest neighbour
R&K	Rechtschaffen and Kales
AASM	American Academy of Sleep Medicine
VAR	Vector autoregressive
PSD	Power spectral density
OLS	Ordinary least squares

1. Introduction

Sleep is the natural rest for humans and animals. We all need it for daily functioning as a counterpart to being awake, when energy is consumed. Biological changes throughout the day are called circadian rhythms, see [60]. They play an important role in our functioning, one of them is prominent as it regulates our sleeping behaviour. This one is influenced by light, which makes us more alert when it is brighter, by body temperature, which is lower at night, and by chemicals such as melatonin and neurotransmitters. The latter are messenger substances that enable neurons to communicate with each other and can be easily affected by caffeine, medicines or drugs. Other addictive substances like alcohol or nicotine affect the sleep behaviour as well. This can lead to sleep deprivation.

Not only physical but also mental natural recovery happens during sleep. If the body does not get enough sleep, it can cause severe impairments in everyday life. We get drowsy and it is hard to concentrate. Low performance in cognitive tasks, judgement, attention and coordination are the result. Especially when being awake for longer than 24 hours, metabolic activity decreases, which results in less energy production of the body. If the deprivation continues, hallucinations as well as mood swings can occur. From this it can be deduced that sleep reduces the risk of suffering from physical and mental diseases as it helps us to maintain emotional and social functioning at an optimal level, see [60, 70]. Hence, it is of great importance to investigate sleeping behaviours for health and, subsequently, for socio-economic reasons, as it effects all people.

Consequently, as sleep deprivation can be caused by a neurological-psychiatric sleep disorder, accurate diagnosis methods of sleep are of great importance for our society. The gold standard for these is polysomnography in a sleep laboratory over night for at least eight hours, a recording of multiple biosignals with multiple channels. Using electroencephalography (EEG), electrooculography (EOG), electromyography (EMG) and electrocardiography (ECG) the activity of the brain, the eyes, the muscles and the heart are measured, see [60, 31], whereas breathing work is recorded as well. The polysomnography is evaluated by a specialist, which can take up to two hours. This not only takes some time, but also requires a lot of financial resources of the health care system. Svetnik et al., see [64], wrote that in 2005 the global market spent around 4.3 billion US$ for sleep stage classification with a rising forecast due to the new invention of different sleep medication. EEG monitoring during sleep deprivation can also help in the diagnosis of epilepsy, since electrical brain activity changes greatly during seizures, see [47].

Automated and semi-automated algorithms for supporting sleep stage classification and analysis of the different biosignals are getting created since over 20 years. Many of these approaches are based on machine learning algorithms. In research, new classification models are created and tested on different sleep datasets to benchmark and compare against to other models. Recent publications by Brandmayr et al. [10] or Van der Donckt et al. [69]

introduce different classification approaches that reach accuracies of up to 87.1%, benchmarking various datasets. Some algorithms are based on supervised learning, a method for which the classified stages for the training dataset are available, and some others on unsupervised learning, a training method for which the output is not known. Even though they perform very well and achieve accuracies of 80% or above, none of them is in widespread use in hospitals or sleep laboratories. A possible explanation is that many of the approaches are difficult to interpret and hardly explainable, see [23], properties that are not very acceptable in the medical field. It is important to reach at least 80%, as this is also approximately the interrater reliability, i.e. the overall agreement between two different sleep scorers, who manually classify a polysomnography recording, according to [18].

This is where our approach comes in as we aim to find a plain interpretable model that is based solely on EEG recordings. Three EEG channels are used to extract 21 different features, which are the permutation entropy (PE) [5], the entropy of difference (EoD) [51], the Kullback-Leibler divergence (KLD) [40], the cross entropy (CE) [50], the Granger causality (GC) [27, 26] and statistical parameters. The PE, EoD, KLD and CE are based on the Shannon Entropy, which is coming from information theory [63]. The predictors have already been used by different researchers in the past in the field of EEG analysis, see [52, 37, 28, 59], but have not yet been combined. We calculate all of the features in the time-domain, such that no fast Fourier transformation has to be applied on the data, with the intention that this speeds up the feature extraction. Including the age, we use 22 different predictors for building a sleep stage classification model. The aim of this thesis is to show that plain linear models, based only on EEG signals and a personal parameter, achieve similar accuracy to more complex ones. This would pave the way for interpretable and explainable sleep scoring models based on time-domain EEG analysis that can be applied in the medical field.

The features are used to train different machine learning algorithms using the cyclic alternating patterns (CAP) Sleep Database [67]. It is an open source database that has been downloaded from PhysioNet [25], which is a data repository that contains freely available data from medical research. We look at linear discriminant analysis (LDA), decision trees, bagged trees, boosted trees with AdaBoost, kernel logistic regression, naive Bayes classifier, weighted K nearest neighbour method (KNN) and linear support vector machines (SVM). They are all supervised learning approaches. AdaBoost is an abbreviation of adaptive boosting, which is an algorithm that optimises the performance of a model. The descriptions of the algorithms mainly follow [50, 39]. Afterwards, a comparison between the models based on the algorithms is given, regarding performance and other properties, and the best ones are used to benchmark the open-source Sleep Telemetry Database of the Sleep-EDF Database Expanded [35], which is available on PhysioNet as well.

This thesis is structured as followed. Firstly, in chapter 2, the electrical activity in brains, as well as the EEG measurement, are described. We also give an insight into (pre-)processing of the recording and coding of the signal such that the mentioned features can be directly computed. Secondly, in chapter 3, sleep and its characteristics are explained. The different stages and how they are distinguished in a polysomnography recording are described. Then, a review of automated and semi-automated approaches of sleep scoring is given as

well. Thirdly, in chapter 4, the already mentioned features are explained in mathematical detail. The underlying CAP Sleep Database that is used for training the models and its properties are described in chapter 5. The parameters and the implementation of the various features are explained as well. The chapter closes with an introduction into all the models that are looked at. In chapter 6, the achieved results are presented and the models are compared. An enhancement of the models to increase the accuracies is given as well. We do not only take a look at the classification accuracies but also at other properties of the best performing models. The best ones are then used for sleep stage classification of the Sleep Telemetry Database to benchmark our created models. Afterwards, in chapter 7 a rough summary and a discussion about the outcomes as well as thoughts of improvement are given.

The implementation was done in MATLAB©, Version 2022b [45]. The programs were executed on a MacBook2017 with an 1.2GHz Dual-Core Intel Core m3 processor and an 8GB 1867MHz LPDDR3 RAM working memory up until the results of chapter 5. The training and testing of the machine learning models described in chapter 6 were performed on a desktop computer with Microsoft Windows 10 as an operating system because of the computational effort that the training took. It runs with an AMD Ryzen7 1700X Eight-Core Processor with working memory of 3400MHz on eight kernels and 16 logistic processors. For creating the different models we used the Classification Learner application of MATLAB.

2. Processing and Coding of Electrical Brain Signals

Every stimulus humans perceive, whether by sight, touch, hearing, smell or taste, is processed by their nervous system. It consists of a peripheral and a central part. The first one is represented by peripheral nerves which are made of nerve fibres. They have different sizes and functions as there are sensory, motor and vegetative nerves. The sensory system connects our senses and transmits different stimuli to the central nervous system. The motor nerves control the movement of our striated muscles, i.e. our skeletal muscles. The functionality of our internal organs is regulated by the vegetative ones. The central nervous system consists of the brain and the spinal cord. It is estimated that 100 billion nerve cells are embedded in the brain. In between two such neurons are glia cells which make up the structure of the brain. In the central nervous system the information received by the sensory nerves is processed and memorized and, if applicable, actions, for example movement, are ordered which are then carried out by the motor nerves. Transmission of information is done by so-called action potentials which are temporary changes in the neuron membrane electrical potential. They travel along axons. The joints between two neurons or between axons and neurons are synapses, see [19, 47, 60].

The information processing in the central nervous system is done by changing the electrical properties of the membrane, i.e. the ionic conductivity, see [16]. The measurement of this brain activity is called EEG. It is of great importance for neurologists in the diagnosis of diseases, but also for anaesthetists in monitoring during surgery. The recording and processing of an EEG is a complex technical procedure. This will be discussed in more detail in the next section. In the second section the encoding of the EEG signal into certain patterns will be explained. These patterns will be used for the analysis of the EEG.

2.1. Signal Recording and Processing of Human Brain Activity

The procedure of an EEG consists of many different steps, several things are needed, see [31]. To summarise the signal recording and processing, a schematic diagram of the process is shown in figure 2.1, the different steps are described in this section.

As already mentioned, information processing is done by changing the neuron membrane electrical property. To measure this, electrodes are used as the link between the human body and the recording system. Between the neuron membrane and the skin, where the electrode is placed, there are still some layers, i.e. the brain itself, the skull and the scalp as well as other structures like veins, arteries, muscles and tissue. All of these attenuate and disturb the signal. Hence, only a significant quantity of neurons active concurrently is capable of producing a signal of sufficient strength for recording, see [60].

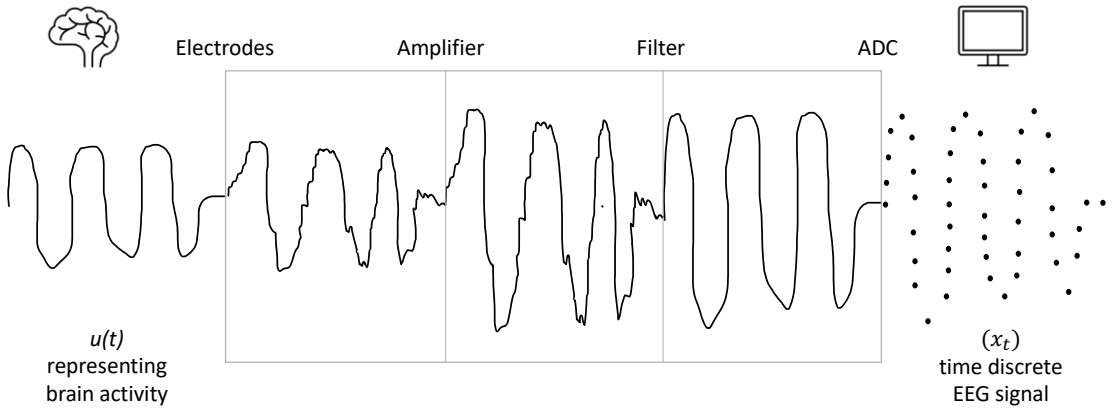

Figure 2.1.: Schematic diagram of the signal recording and processing of an EEG.

Once this happens it is also important to look at where the electrodes are placed. For reasons of comparison and reproducibility the "10-20 system" has been installed as a standardisation for the EEG. It allows a dynamic placement since it does not use a constant measure but specifications given in percent. This way it can be applied not only to adults but also to children, infants or other people with smaller heads. The name comes from the fact that certain anatomical landmarks are taken and then the placement occurs in 10 or 20 percent steps of this distance. The key points of reference are the nasion, the lowest point between the nose and the forehead in the middle between the eyes, the inion, the lower bony protuberance in the middle of the back of the head at the base of the neck muscles, and both preauricular points, the cavities in front of the external auditory canal just below the cheekbone and above the mandibular joint, see figure 2.2. The usual EEG measurement is done with 21 electrodes, for a larger setting there is a definition with 75 electrodes available, including four for a reference signal. In smaller settings the reference for the voltage is usually the average value of all of the electrodes used. There are also cases where the number of electrodes is minimised down to three, for example during surgery. One is for grounding, one for a reference signal and one for the actual measurement. Anaesthetists use it in that setting to monitor the level of consciousness.

As soon as the signal has been measured by the electrodes the processing can start. At first, an amplifier is needed to increase the amplitude of the signal due to the attenuation mentioned above. Raw EEG signals have a value in the range of μ-volt. Secondly, artefacts are detected. These are disturbance voltages, also known as noise, which interfere with the signal. They can be split up in two categories. The first category is the physiological which includes movement, sweating or muscle twitching. Other interfering measurements such as EMG, ECG or EOG are also part of it. The second category comprises the technical related artefacts like power-supply, noise of the amplifier, missing grounding or using false electrodes or faulty cables. In most cases these artefacts result in at least a 50Hz disturbance, see [47].

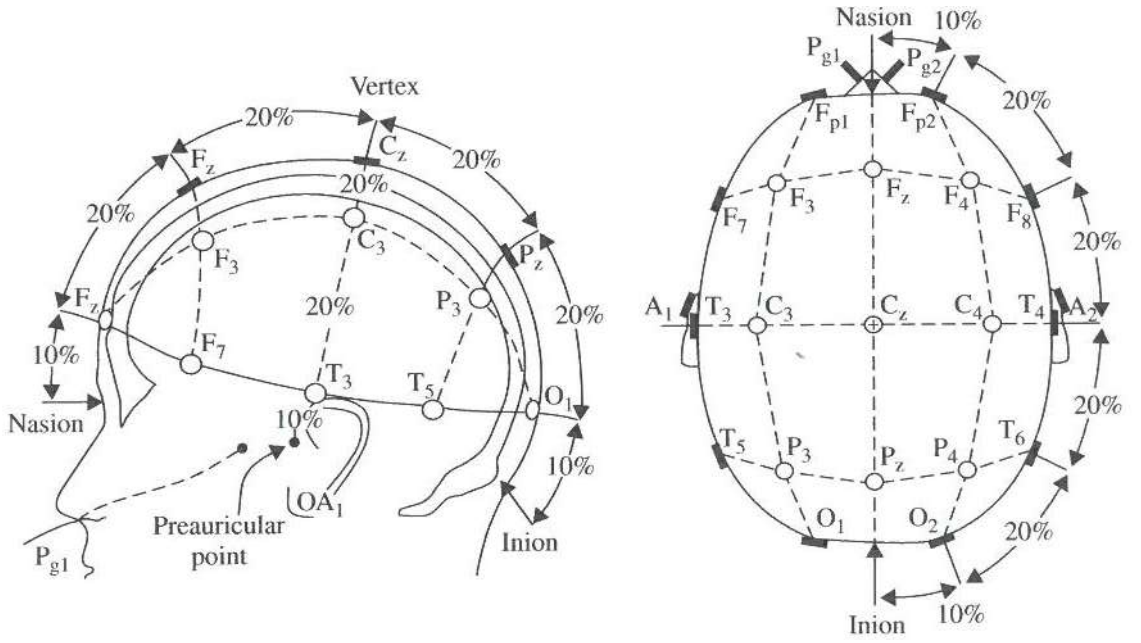

Figure 2.2.: Representation of the "10-20 system" for the electrode placement during an EEG recording, taken from [60]. The right picture also shows the placement of the common setting of 21 electrodes.

Therefore, it is necessary to apply highpass and lowpass filtering. Often a highpass filter of 0.5Hz is used which removes low-frequent noise, for example breathing. For lowpass filtering the disturbance of line voltage alone is around 50Hz, but it is needed to run the technical instruments. Furthermore, as [71] states, muscle activity disturbs EEG frequencies over 20Hz. Even if medication for muscle relaxation is applied to the patient, muscle activity occurs in the forehead.

It is also important to know the frequency of human brain waves. They are divided into five categories: δ (0.5-4Hz), θ (4-8Hz), α (8-13Hz), β (13-30Hz) and γ (above 30Hz, mainly up to 45Hz). The first category, the δ-waves, mainly appear in deep sleep but can also be found during the process of waking up. The θ-waves are associated when humans are in an unconscious state but easy to arouse. This level is called drowsiness. The next higher frequent ones, the α-waves, is the most common category in brain activity. It appears primarily in an awake state when the eyes are closed. The β-waves occur mainly when adults are mentally active, i.e. during thinking, processing or being attentive. Their appearance in specific states of consciousness are further described in section 3.1. Because waves of the last category, the γ-waves, are rare and do not play a major role in determining the level of consciousness, plus all of the disturbances listed above, a 30Hz lowpass filter is applied to the EEG, see [60].

After filtering, the signal must be discretised for further investigation and processing. This is done by a multi-channel analogue-to-digital converter (ADC). The sampling frequency needs to be at least two times the maximum frequency. This result was stated in 1949 by Claude Shannon with the Nyquist-Shannon sampling theorem in [62].

Theorem 2.1.1. *If a function $f(t)$ contains no frequencies higher than W cps, it is completely determined by giving its ordinates at a series of points spaced $\frac{1}{2}$ W seconds apart.*

Proof. The proof can be found in [62]. □

As the frequencies of human brain waves go up to 45Hz, sampling frequencies of at least 100 samples/s should be enough in theory. For practical uses, other sources like [60] suggest a sampling frequency of 200 samples/s, since the effective bandwidth for EEG signals is around 100Hz, depending on the specific purpose for which the recording is being made. After discretisation, the signal can also be filtered further.

2.2. Coding of the Data for Further Investigation

Processed EEG signal can be displayed as a time series $(x_t)_I, I = \{1, ..., n\}$, with a certain length n. Depending on the investigation it is necessary to encode the series in some way to calculate different parameters. These will be discussed in detail in chapter 4, but the encoding in patterns will be introduced here. The parameters, which need the signal in an encoded form, are the PE, as it was defined in 2002 by Christoph Bandt and Bernd Pompe [5], and the EoD, as Pasquale Nardone introduced in [51]. Some other measures are based on these two and will therefore need the same encoding as them. The notation and definition will follow [9, 73]. The PE and the EoD have in common that one first has to choose an order m and a time delay τ. Afterwards the series will be divided into $k := n - (m-1)\tau$ tuples of length m. The time delay tells the index shift we have between two values in the tuple. This means, that for $\tau = 1$ only neighbouring values appear in a tuple, whereas for $\tau > 1$ the index distance is greater.

The influence of the time delay τ is discussed in [54]. It is explained that a higher time delay behaves the same as a lower sampling rate which means that the product of the two values is constant. Hence, it is of great importance to specify both settings such that the parameter is described correctly, for example for reasons of reproducibility. This holds independently of the order m.

As we want to divide the series (x_t) into k tuples of length m, where the index shift between each component is τ, the mapping rule is

$$\begin{aligned}(x_1, ..., x_n) \mapsto ((x_1, &x_{1+\tau}, x_{1+2\tau}, ..., x_{1+(m-1)\tau}),\\ &(x_2, x_{2+\tau}, x_{2+2\tau}, ..., x_{2+(m-1)\tau}), ..., (x_k, x_{k+\tau}, x_{k+2\tau}, ..., x_{k+(m-1)\tau})).\end{aligned} \tag{2.1}$$

Now, we distinguish between the PE and the EoD. Starting with the first parameter, when a certain tuple beginning with the value x_j is taken, we look at the numerical order of the values. Numbers $r_1, r_2, ..., r_m \in J = \{1, 2, ..., m\}$ are introduced such that it holds

$$x_{j+(r_{s_1}-1)\tau} < ... < x_{j+(r_1-1)\tau} < ... < x_{j+(r_{s_m}-1)\tau}.$$

The values $(r_s)_J$ are chosen such that one value of the set J is assigned to exactly one of the values (r_s). It is important to mention that r_1 is associated to the first value of the tuple x_j, r_2 is associated to the second value $x_{j+\tau}$, and so on, until r_m is associated to $x_{j+\tau(m-1)}$. The order shows that r_{s_1} is associated to the lowest value of the tuple and r_{s_m}

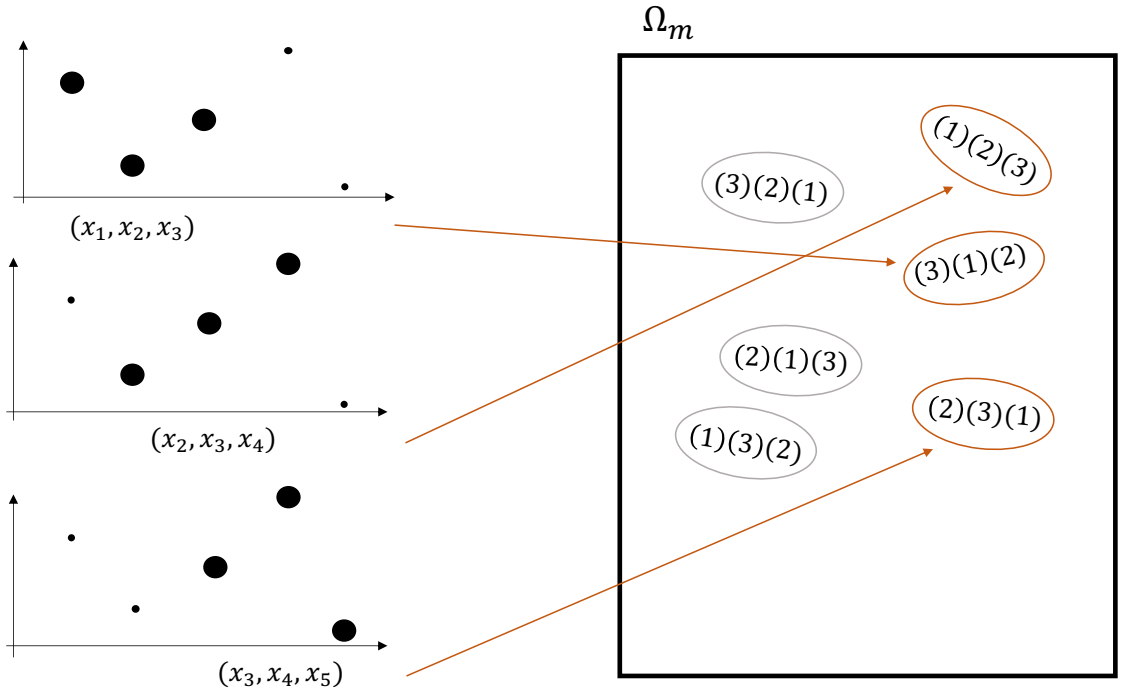

Figure 2.3.: Schematic diagram of the mapping between the tuples of the time discrete signal (x_t) and its encoded permutation variation (r_s) for the PE in Ω_m for $m = 3$ and $\tau = 1$.

to the highest one. If two values within a tuple are equal, then the first appearing one has the lower index, i.e. if $x_{j_1} = x_{j_2}$ with $j_1 < j_2$ then $r_{s_1} < r_{s_2}$ with r_{s_1} associated to x_{j_1} and r_{s_2} associated to x_{j_2}. This allows ties to be bypassed. For our application of EEG analysis, this is very rarely the case.

The numbers $(r_s) \in J$ are a permutation of the pattern $(1)...(m)$. By writing the numbers $(r_1)...(r_m)$ in rounded brackets, patterns with order $m > 9$ can be used as well. The set of all possible patterns depends on the chosen order m and is defined as

$$\Omega_m := \{\omega : \omega \text{ is a permutation of } (1)(2)...(m)\}.$$

The cardinality of this set is $|\Omega_m| = m!$ since this is the number of permutations of positive integers up to m. Two extreme cases are given as an example. If the tuple is strictly monotonously increasing, then the permutation type equals $\omega = (1)(2)...(m-1)(m)$. On the opposite if the tuple is strictly monotonously decreasing, then $\omega = (m)(m-1)...(2)(1)$. A graphical representation of this encoding can be seen in figure 2.3.

The second form of encoding needed is for the EoD. The tuple generation (2.1) is taken again but now we just look at neighbouring values within one tuple. The encoded form will be only built up from " + " and " − ". Starting from the first value x_j of an arbitrary tuple if $x_{j+\tau}$ as the next one is higher, i.e. $x_{j+\tau} > x_j$, then a " + " is added to the encoding. On the opposite, if it is lower, i.e. $x_{j+\tau} < x_j$, a " − " is added. This comparison is carried

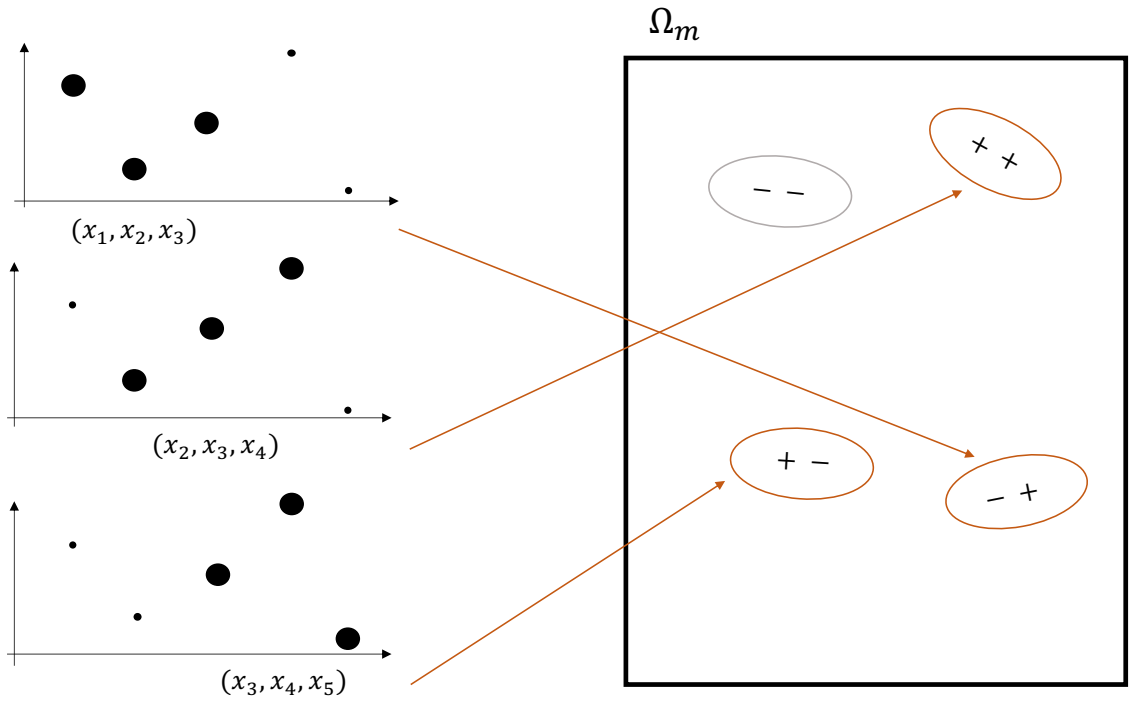

Figure 2.4.: Schematic diagram of the mapping between the tuples of the time discrete signal (x_t) and its encoded type for the EoD in Ω_m for $m = 3$ and $\tau = 1$.

out for the whole tuple. If two neighbouring values within a tuple are the same, then the comparison results in a " + ", like if the first one would be lower. This coincides with our definition for ties of the PE encoding. But again, in our application of EEG analysis, this is very rarely the case.

For a tuple of length m the encoded form has length $m - 1$ since there are that many comparisons. Hence, the number of all possible patterns of " + " and " − " signs of the encoded form is 2^{m-1}, because every position can be taken by the two symbols. By defining the space of all possible patterns again as

$$\Omega_m := \{\omega : \omega \text{ is a string of symbols of length } m-1 \text{ containing " + " or " − "}\},$$

the cardinality is $|\Omega_m| = 2^{m-1}$. Again, for strictly monotonously increasing, the pattern $\omega =$ " + +... + ". On the opposite if the tuple is strictly monotonously decreasing, then $\omega =$ " − −... − ". A graphical representation of this encoding can be seen in figure 2.4.

3. Attributes and Classification of Sleep

Firstly, the main question: What is sleep and how can it be distinguished from wakefulness? The answer is given for example by [49]. On the one hand, wakefulness is typically defined by a state of high alertness and goal-oriented movements which are caused by either external stimuli or internal motivations. On the other hand, sleep is characterised by unconsciousness from which one can be aroused. The last aspect is the important difference compared to a coma, from which one cannot be aroused, see [60]. Apart from that, it is specified mainly by decreased interaction, motor and muscular tone, closed eyes, an increased rate of cell structure synthesis and, on the contrary, a decreased rate of breakdown of cell structures.

The sleep-wake cycle can be categorised in different states, which will be focused on in section 3.1. The usual classification is done manually by specialists following a manual [32] using different biosignal measurements. In research there are many algorithms used for classification and decision support systems, an overview is given in section 3.2. They primarily use EEG measurements as described in chapter 2.

3.1. Sleep Stages and their Characterisation

The specific sleep states and their characteristics will be introduced in this section. The measurement plays an important role in sleep analysis. The sleep-wake cycle can mainly be divided in the three categories wakefulness, rapid eye movement sleep (REM) and non-rapid eye movement sleep (NREM), see [49]. All of them have their own characteristics regarding behaviour, vital parameters and brain waves frequency. Wakefulness is typically identified by high frequency oscillations in the brain waves frequency spectrum with 8Hz upwards, which indicates α-waves. The EEG activity primarily has a low amplitude. Usually, people have their eyes closed as well as increased muscle tone.

When falling asleep further, i.e. NREM sleep, the frequencies of the EEG slow down, usually in the range of θ-waves with a frequency of 0.5 to 4Hz. The amplitude of the EEG activity is high. Sleep spindles with a frequency of 11 to 15Hz also rarely occur but only they just around 0.4s. Therefore, NREM sleep is characterised by a reduced muscle tone and lower frequent brain waves. Furthermore, the autonomous nervous system reacts to this sleep state, i.e. on the one hand the activity of the parasympathetic nervous system increases, while on the other hand the sympathetic activity decreases, which results in a reduced heart rate and blood pressure. The heart rate variability is used to determine this change of autonomous nervous system activity.

REM sleep has a high frequent spectrum, similar to wakefulness, but a decrease in muscle tone as well as saw-tooth like waveforms are specific features for that state. In this state a change of the vital parameters is not identifiable as they are at a similar level as during wakefulness. REM sleep accounts for around 20 to 25% of the total sleep time. It is the

time when people are supposed to dream. Hence, the higher frequent brain activity as well as eye ball movement is reasonable. The last symptom is also the reason why it is called rapid eye movement sleep. During this stage even talking or sudden movements are possible.

The NREM state can be further distinguished in four subcategories, regarding to Rechtschaffen and Kales (R&K) [58]. Stage 1 is drowsiness, when someone is between being awake and asleep. According to [60] around 5 to 10% of the entire sleep time is spent in this stage, where the muscles begin to relax. One can be easily aroused in that time. During growing up the brain wave activity changes. Infants and younger children have more prominent θ-waves in this level of consciousness. Teenagers as well as elderly people only have little appearance of slow activity. Stage 1 in adults is primarily characterised by α-waves. Sleep spindles with a frequency of 18 to 25Hz can also occur.

Stage 2 accounts for 40 to 50 % of the total sleep time. When using an EOG to measure eye movement, one can identify that it stops. The frequencies of the brain waves slow down more and more, but still spindles of higher frequencies appear from time to time. It is the state of light sleep. In stage 3, δ-waves occur with rarely appearing sleep spindles of 12 to 14Hz. In stage 4, the δ-waves are the most prominent wave category. They are also barely detectable in a routine EEG, but require monitoring of at least 24 hours, see [60]. In both of the last stages muscle and eye activity is almost gone. Humans are hard to wake up during these stages but if it happens, one is very disorientated. These are called the stages of deep or very deep sleep.

The classification by R&K was revised by the American Academy of Sleep Medicine (AASM) in 2007, see [32]. They merged stages 3 and 4 and it is now the internationally used and recognised sleep stage classification. The abbreviation of the different stages are listed in table 3.1.

Table 3.1.: Sleep stages and their abbreviations in 1968 and 2007.

Sleep stage	awake	REM	NREM stage 1	NREM stage 2	NREM stage 3	NREM stage 4
R&K (1968)	W	R	S1	S2	S3	S4
AASM (2007)	W	R	N1	N2	N3	

Figure 3.1 shows an example of how the course of the sleep stages can be during one night. The data is taken from the open-source CAP Sleep Database [67]. It contains a polysomnography recording of 108 patients that were monitored over night. Most of them suffer from a sleep disorder, but there was also a control group of 16 healthy subjects included. The sleep stages are given in a thirty second interval. The dataset was published in 2001, therefore the sleep stage classification by R&K was used. For comparable results we merged the stages S4 and S3 by hand to get the new classification by AASM. The subject n1 is our example as one of the 16 healthy probands, i.e. not suffering from a sleep disorder. The woman was 37 years old at the time of the recording.
The sleep stages given on the y-axis are abbreviated as mentioned in table 3.1. The patient n1 fell asleep at around 22:10 which is approximately 30 seconds after the sleep stage documentation began. The night and monitoring lasted for 9 hours and 33 minutes. The

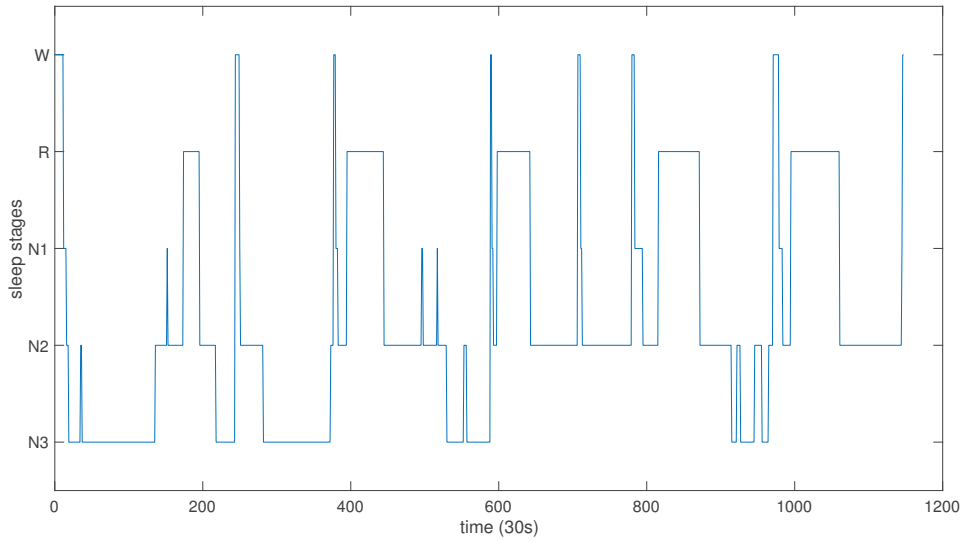

Figure 3.1.: Course of the sleep stages over time using the example of the patient n1.

unit of time on the x-axis is given in 30 seconds. We can see, that the patient fell asleep and woke up for a brief amount of seconds a few times during that night. The REM phases always lasted some time, but the most time was spent in stage N2, as it is characteristic for that stage like mentioned above.

Having a polysomnography recording, the classification is done by specialists following a certain manual. Nowadays the manual of the AASM published in 2007 is usually used, see [32], which is updated every few years. The scoring is done visually by splitting the biosignals in 30 second windows, looking at the different channels during this time and categorising the sleep stages regarding their attributes. For a recording of a whole night this can take some time. Hence, algorithms and decision support systems have been invented and implemented since over 20 years. An overview about different approaches, especially of the last few years, is given in the next section.

3.2. Review of Automated Classification Algorithms

Manual sleep scoring is a very time consuming procedure. The polysomnography recording is split in 30 second windows, also called an epoch, and then examined by a specialist, which can take up to two hours. Hence, semi-automated and automated approaches of sleep scoring are explored. We provide a literature review of various concepts, focusing on EEG analysis. In particular, we take a look at the recent advances of the last three years.

A difficulty that automated sleep scoring faces is the comparison to manual sleep stage classification because of interrater reliability. In 2009, Danker-Hopfe et al. [18] showed in a study that for the R&K rules the overall agreement was 80.6% and 82% for the standards set by AASM. Consequently, procedures that reach this percentage are already quite good.

In biology and medicine a measure for classifier accuracy is Cohen's κ coefficient, see [14], which is also used in many publications in the field of sleep scoring. It uses the proportion of units in which the judges agreed p_0 and the proportion of units for which agreement is expected by chance p_c. The value is then defined as

$$\kappa = \frac{p_0 - p_c}{1 - p_c}. \tag{3.1}$$

The value has a range of $[-1, 1]$ and the higher the value the better agreement. A negative κ indicates worse agreement than random classification. For values between 0 and 0.2 one has slight agreement, then up to 0.4 fair, until 0.6 moderate, then up to 0.8 substantial and finally up until 1 (almost) perfect agreement. Danker-Hopfe showed in her study about the interrater reliability that Cohen's κ coefficient for the R&K rules was 0.68 and for the AASM standards 0.76 which means that these values are to be aimed at.

In [64] two systems which had been tested on clinical data and the underlying algorithms had been published were compared. These were Morpheus by WideMed Technologies Ltd. from Israel [53] and the Somnolyzer24X7 by The Siesta Group Schlafanalyse GmbH from Austria [2]. The authors looked at 164 recordings of 82 different subjects and compared the results between the different automated and semi-automated systems with the manual sleep scoring according to the rules by R&K. The epoch-by-epoch agreement was between 70% and 75.9% with a Cohen's κ between 0.548 and 0.627 with semi-automated classification performing better than the automated one. Hence, the automatisation of this procedure promises good results.

Ever since then, the systems developed and it was already published in 2010 that the agreement between two Somnolyzer24X7 assisted scorings reached 99% with $\kappa = 0.99$ and between semi-automated and manual scoring 81% to 82% with $\kappa = 0.76$ according to the AASM rules, see [3]. The researchers in this field did not want to stop with automated and semi-automated scoring using polysomnography but also less biosignals and wires.

In 2003, Louis et al. published a computer-based sleep scoring algorithm with up to three steps to determine the states active wake, quiet wake, REM sleep and NREM sleep, see [43]. The difference between active and quiet wake is whether the subject is moving or not. It has a very obvious approach and is designed for Wistar rats using only EEG and EMG. The algorithm is a decision-making process and contains three steps. For each step, a threshold is set that has been previously determined using visually scored data. Firstly, the EMG amplitude is looked at and if it is above the threshold, the rat is in active wake, if not another level is reached. Secondly, the EEG amplitudes of the frequency wave categories are considered. If the ratio of $\frac{\delta \cdot \alpha}{\beta \cdot \gamma}$ is above the second threshold, the rat is in the state of NREM sleep, if not we get to the third level. There, if the ratio of the EEG amplitudes of $\frac{\theta^2}{\delta \cdot \alpha}$ is above the third threshold, the rat is in REM sleep state, if not it is simply in quiet wake. With this algorithm an agreement of 87.9% was reached in comparison to manual raters. In this publication no Cohen's κ value was given.

Over the years, different bio- as well as other signals were used to perform sleep stage classification. The broad field stretches from EOG, see [30], EEG, see [44], ECG, see [57], and combinations such as EEG and EMG, see [1], or EEG and EOG, see [74] but also cameras were used during a car drive to determine vigilance states by recording eyes, face and head, see [33]. According to [21] all of the signals contain important information for

sleep scoring. Consequently, by combining them the result improves and we gain more knowledge. However, the goal is to reduce wiring but still get good results in comparison to manual sleep scoring. Performance improvement and optimisation drove the various algorithms and procedures further and at some point machine learning was added in different forms. Faust et al. made a review in 2018 about deep learning for healthcare applications, see [20]. The result was that these machine learning methods will be used more and more, since they improve as more data becomes accessible.

In 2019, Radha et al. addressed the issue of performing sleep stage classification over time of patients at home, see [57]. Polysomnography is done for one or two nights in the hospital, but cannot be done long term since the wiring effects the sleep quality. Therefore, a cheap and ergonomic alternative should be found. They studied heart rate variability in the context of sleep scoring. For this purpose, they used a long-short-term memory network, because according to the authors, long term sleep patterns should also be taken into account. 584 measurements of 292 subjects of different age and health status of the Siesta database [36] were examined. They split the data in four folds, trained the model with three and tested it with the last. The overall accuracy was the mean of these four test runs which results in 77% for a model structure of 128 cells with $\kappa = 0.61$. However, it is described that the performance for people over 50 years declined and this method does not get close to EEG-based sleep stage classification.

EEG-based methods were followed quite a lot over time according to [22]. The methods range from fuzzy logic over hidden Markov models and wavelet transformation to feature extraction in combination with different machine learning algorithms. Especially the most famous machine learning method, i.e. neural networks, was used in different variations. In 2020, a convolutional neural network was proposed by Fernandez-Blanco et al. that achieved very good results, see [22]. They studied the difference of examining two channels instead of one, i.e. Fpz-Cz and Pz-Oz, on a dataset of 61 recordings with healthy people as well as patients under medication because of problems with falling asleep. The authors gained an accuracy of 92.67% in comparison to manual scoring with $\kappa = 0.84$.

Combining the good results with low-cost and wearable measurement devices is done in [13, 41] for example. Casciola et al. applied their convolutional neural network and long-short-term memory model to EEG measurements of twelve subjects between 21 and 61 years old. Using a portable two-channel headband they get an accuracy of 74% for their model. Cohen's κ coefficient was not given in that paper. In comparison, Kwon et al. invented a convolutional neural network model based on two EEG and two EOG channels and trained it with a public dataset of 100 people. Afterwards, they tested the model on four subjects and got an accuracy of 81.52% with $\kappa = 0.734$. The advantage of their approach was that they fabricated four nanomembrane, stretchable electrodes which are easily wearable. With different neural network enhancements, Brandmayr et al. published a single channel EEG model in 2022 whose performance had improved even further, see [10]. In comparison to other deep learning models the authors achieved the best results applied to three databases with up to 87.1% accuracy and $\kappa = 0.823$.

Although many methods with good results have been described in research, none of them have yet manifested in clinical or pre-clinical settings. A possible explanation is the lack of interpretability, a big disadvantage that all the deep learning models show, see [23]. A possible way out is to change the method to explainable and interpretable methods.

In 2023, Van der Donckt et al. proposed with logistic regression and gradient boosted trees, two rather simple models, see [69]. The authors extracted 131 different features with 56 based on the time domain and 75 based on frequency domain per epoch, but looked at eight different windows at the same time. They included the current epoch, but also two in the past and two in the future, as well as overlapping windows of 60 and 90 seconds. Consequently, the amount of features looked at per epoch was 1048. With these models, the authors looked at four different datasets and compared them to other sleep scoring models. The results were an accuracy of up to 86.7% and a Cohen's κ value of up to 0.816 with which the models could not only compete with deep learning models, but also surpass them for some measured data. The publication shows the opportunity of interpretable machine learning models in the field of sleep scoring, which have a better chance of use in the clinical context.

A recent publication in May 2023 by Metzner et al., see [46], point out that also non-machine learning algorithms can still give advances in the field of sleep scoring. They used the general discrimination value, a measure that is based on the intra-class and inter-class distances for class separability, on the frequency domain EEG measurement together with a principle component analysis. The authors showed that one component, that has a strong correlation with the temporal progression of the sleep stages, can model the depth of sleep very well. This result promises development in the understanding of brain activity in sleep.

4. Mathematical Fundamentals of the Sleep Scoring Modelling Framework

Our modelling approach introduced in chapter 5 is based on 22 features. Some of them are more prominent as they are mainly statistical parameters, an overview is of these is given in section 4.6. The ones that are not widely known are described first in the next sections. The underlying theory of many of them is entropy, hence, before describing the used parameters an introduction in the field of entropy in information theory is given in section 4.1. Afterwards, the features PE, section 4.2, and the EoD, section 4.3, are described which are already mentioned in section 2.2, where the encoding of the time-discrete EEG signal is explained as a preprocessing to calculate these parameters. Next, two comparison methods are introduced in sections 4.4 and 4.5, namely the KLD and the GC. The first one describes the difference to a reference model, the second one calculates the gain or loss of information between two signals.

The first feature does not need to be calculated because it is directly dependent on the examined patient. As we look at EEG measurements of patients of different age and health-status we also include age as a feature, since EEG recordings show amplitude and frequency changes in elderly people, see [60]. It is described that on the one hand α frequency is reduced and β wave activity is increased. Also, the amplitudes of the EEG during NREM sleep lower with age. REM sleep is less present, but sleep disruption happens more frequently. Age is also the only feature that is independent of the time series $(x_t)_I$, $I = \{1, ..., n\}$, that represents the EEG.

4.1. Application of Entropy in Information Theory

Many of the used parameters in the model are entropy-based. Therefore, an introduction into the concept of entropy is given, which has its origin in physics. In the field of thermodynamics it plays a very important role, see [68]. In the 19th century, Ludwig Boltzmann described the heat of a fluid by its disordered movement of atoms and molecules. The entropy increases when the number of states that can be taken rises. In the middle of the 19th century, Clausius gave a macroscopic description of the term, contrary to Boltzmann's microscopic view. He introduced the change in the entropy of a system as being dependent on the supplied heat and, consequently, the temperature. He supposed the second principle of thermodynamics, "In a closed system, entropy never decreases". Hence, it can only stay constant or increase.

Since then, the term entropy spread throughout many fields of physics, from astronomy to mechanics. It even found its way to chemistry, biology and social sciences, but every field interprets the term in its own way, as [55] mentions.

In 1948, Claude Shannon introduced another application of entropy in his article "A Mathematical Theory of Communication" [63]. He used it in information theory and described the loss of information when a signal is transmitted. In particular, the Shannon entropy is the rate at which information is produced if discrete information is modelled as a Markov process. Therefore, there are n possible states that occur with certain respective probabilities $p_1, ..., p_n$. Shannon posed the question, "Can we find a measure of how much 'choice' is involved in the selection of the event or of how uncertain we are of the outcome?", see [63], p. 10. As an answer he proposed the Shannon entropy $H(p_1, ..., p_n)$, which should suffice the assumptions:

(1) The entropy H is continuous in the p_i.

(2) If there is an equal probability distribution among the events, i.e. $p_i = \frac{1}{n}$ for all $i = 1, ..., n$, then H is a monotonically increasing function with respect to n.

(3) A choice of n events can be broken down into consecutive decisions. Then, H is the weighted sum of the individual values.

Under these assumptions the following theorem is stated.

Theorem 4.1.1. *Let $K \in \mathbb{R}^+$ be a positive number. The only H satisfying the three above assumptions is of the form*

$$H(p_1, ..., p_n) = -K \sum_{i=1}^{n} p_i \log p_i. \tag{4.1}$$

Proof. Firstly, we consider an equal distribution $p_i = \frac{1}{n}$ for all $i = 1, ..., n$. Furthermore, $A(n) := H(\frac{1}{n}, ...\frac{1}{n})$ is defined. We know from condition (2) that A is a monotonically increasing function in n. Setting $n = s^m$, which means that there are s^m possible choices, with $s, m \in \mathbb{N}$ arbitrarily chosen we get $A(s^m) = mA(s)$ using the decomposition condition (3), i.e. m choices from s possibilities. Now $t, n \in \mathbb{N}$ are chosen, such that

$$s^m \leq t^n < s^{m+1}. \tag{4.2}$$

The inequality can be transformed by taking the logarithm into

$$\frac{m}{n} \leq \frac{\log t}{\log s} \leq \frac{m}{n} + \frac{1}{n}.$$

This holds since the logarithm function is monotonously increasing and all variables are positive. This inequality can be rewritten with an arbitrarily small ε to

$$\left| \frac{m}{n} - \frac{\log t}{\log s} \right| < \varepsilon. \tag{4.3}$$

Analogously to above it also holds that $A(t^n) = nA(t)$. With the monotonic assumption (2) and (4.2) we get the inequality $A(s^m) \leq A(t^n) \leq A(s^{m+1})$ which is equivalent to

$$\frac{m}{n} \leq \frac{A(t)}{A(s)} \leq \frac{m}{n} + \frac{1}{n}.$$

Again, with an arbitrarily small ε it holds

$$\left|\frac{m}{n} - \frac{A(t)}{A(s)}\right| < \varepsilon. \tag{4.4}$$

Hence by adding/subtracting inequality (4.4) from (4.3) one gets

$$\left|\frac{A(t)}{A(s)} - \frac{\log t}{\log s}\right| < 2\varepsilon.$$

This means, that

$$A(t) = K \log t \tag{4.5}$$

with $K \in \mathbb{R}^+$ being positive for the monotonic assumption (2).

Secondly, we assume we have an arbitrarily large number of measured data n_i for n different possible choices. The probability for the j-th choice is

$$p_j = \frac{n_j}{\sum_{i=1}^{n} n_i}$$

with $j \in \{1, ..., n\}$. The entropy regarding uniformly distributed probabilities among all measured points is as stated in (4.5)

$$H\left(\frac{1}{\sum_{i=1}^{n} n_i}, ..., \frac{1}{\sum_{i=1}^{n} n_i}\right) = A\left(\sum_{i=1}^{n} n_i\right) = K \log \sum_{i=1}^{n} n_i.$$

With the decomposition assumption (3) we can break down the choices to the n possibilities and then, after the j-th is chosen, to a uniformly distributed choice of n_j possibilities of probability $\frac{1}{n_j}$. Hence we can again get with equation (4.5),

$$K \log \sum_i n_i = H(p_1, ..., p_n) + \sum_{j=1}^{n} p_j K \log n_j.$$

Using $\sum_{j=1}^{n} p_j = 1$ and transforming the equation, we get

$$\begin{aligned} H(p_1, ..., p_n) &= K\left(\sum_{j=1}^{n} p_j \log \sum_{i=1}^{n} n_i - \sum_{j=1}^{n} p_j \log n_j\right) \\ &= -K \sum_{j=1}^{n} p_j \log \frac{n_j}{\sum_{i=1}^{n} n_i} \\ &= -K \sum_{j=1}^{n} p_j \log p_j. \end{aligned}$$

Should the probabilities p_j be incommeasurable, then they can be approximated by rational numbers. Due to the continuity assumption (1), the above formula still holds and hence the theorem is proven. The positive number K can be chosen arbitrarily and depends on the application. □

In the next sections different modifications of the Shannon Entropy will be discussed.

4.2. Permutation Entropy

The first parameter that we want to use in our model is the PE. It was introduced in 2002 by Christoph Bandt and Bernd Pompe [5] and is based on the Shannon Entropy (4.1). This statistical measurement describes the complexity of time series. The notation is based on [9, 73]. The starting point for the calculation of the PE is the encoded form of the time series of length n that represents the EEG measurement as it is described in section 2.2. A certain order m and time delay τ are chosen and the series is split in $k := n - (m-1)\tau$ tuples of length m according to the rule (2.1). Afterwards, it is determined which tuple corresponds to which permutation type $\omega_j \in \Omega_m := \{\omega \colon \omega \text{ is a permutation of } (1)(2)...(m)\}$ as it is represented in Fig. 2.3 by an example with $m = 3$ and $\tau = 1$. As the cardinality of the set Ω_m is $|\Omega_m| = m!$ there are $m!$ different permutation types.

For the calculation of the PE it is necessary to calculate the probability distribution among all the possible pattern types. The probability of a certain permutation type ω_j is defined as $p_j := P(\omega_j)$, with

$$P(\omega_j) = \frac{\#\{i \colon 1 \leq i \leq k, (x_i, ..., x_{i+(m-1)\tau}) \text{ has type } \omega_j\}}{k}.$$

The modified Shannon Entropy, such that it holds for the setting of the PE, is with $\mathrm{ld} = \log_2$ a logarithm base of two

$$H_{PE}(m) := H_{PE}(p_1, ..., p_{m!}) = -\sum_{j=1}^{m!} p_j \,\mathrm{ld}\, p_j. \tag{4.6}$$

A summand of the overall sum would not be defined if there exists an i with probability $p_i = 0$. To determine this value, we take the limit towards 0 and, as well known, we get

$$\lim_{p_i \to 0} p_i \,\mathrm{ld}\, p_i = 0. \tag{4.7}$$

Therefore, if there is an i with probability $p_i = 0$ we can neglect that summand in the overall sum. Claude Shannon states in [63] that values of the Shannon Entropy H can range from 0 to the logarithm of the number of summands. Accordingly, the range of the PE, as written in equation (4.6), is $[0, \mathrm{ld}(m!)]$. Firstly, keeping in mind that for every probability distribution holds that the sum over all probabilities equals 1, which reads for our setting

$$\sum_{j=1}^{m!} p_j = 1,$$

one can easily conclude that the minimal value 0 of the range is taken if there exists an i with $p_i = 1$. Then, it follows that for all other $l \neq i$ holds $p_l = 0$ and for the PE follows

$$H_{PE}(m) = -\sum_{j=1}^{m!} p_j \,\mathrm{ld}\, p_j = -p_i \,\mathrm{ld}\, p_i = -1 \cdot 0 = 0.$$

Secondly, the functions takes the maximal value of $\mathrm{ld}(m!)$ when the probabilities over all patterns is equally distributed. This means that $p_j = \frac{1}{m!}$ for all $j \in 1, ..., m!$. In that case, for the PE follows

$$H_{PE}(m) = -\sum_{j=1}^{m!} p_j \,\mathrm{ld}\, p_j = -m! \frac{1}{m!} \,\mathrm{ld}\left(\frac{1}{m!}\right) = -\left(\mathrm{ld}\, 1 - \mathrm{ld}(m!)\right) = \mathrm{ld}(m!).$$

Knowing this range, one can define a normalised measurement to make it comparable independent of the order m because otherwise the range would increase with m as well, since the range is $[0, \mathrm{ld}(m!)]$. This can be achieved by dividing the PE as defined in equation (4.6) by $\mathrm{ld}(m!)$ which results in

$$\hat{H}_{PE}(m) := -\frac{1}{\mathrm{ld}(m!)} \sum_{j=1}^{m!} p_j \,\mathrm{ld}\, p_j. \tag{4.8}$$

The range of the normalised measurement $\hat{H}_{PE}$ is $[0, 1]$. Analogously to above the extremal values are once again obtained for the same probability distribution. This means that on the one hand if there is an $i \in \{1, ..., m!\}$ for that holds $p_i = 1$ then $\hat{H}_{PE}(m) = 0$. On the other hand if there is an equal distribution $p_j = \frac{1}{m!}$, then the maximal value $\hat{H}_{PE}(m) = 1$ is obtained. Consequently, large values of $\hat{H}_{PE}$ indicate higher complexity, since a larger number of different patterns appear.

This way one can also define values and ranges for applications. There are already some fields where the PE is used. Yan et al describe a characterisation of rotary machine status in [72]. The advantage of this complexity measure is that it can also deal with non-linear behaviour as it appears during vibration. Especially good results were found for monitoring the status of rolling bearings. It is also already implemented in the topic of EEG analysis in research. This is, among others, explained by Olofsen et al. in [52]. They investigate the change of the PE when analysing time discrete series of EEG signals of humans who took or got γ-amino-butyric acid (GABA)-ergic anaesthetic drugs. The entropy value was high at almost 1 when the EEG signal had mainly high frequencies. On the contrary, low values at around 0.4 were obtained by $\hat{H}_{PE}$ when low frequencies prevail. They highlight the need of an open-source EEG index that is sensitive to eye-blink artefacts and has minimal to no time delay in reacting to changes in the EEG signal. The PE gives promising first results.

The use case in this thesis also addresses the question of how the value can reflect on the level of consciousness in humans, but we focus on the application in the field of sleep stage classification. When referring to the PE, we always speak of the normalised PE as defined in equation (4.8) from now on.

4.3. Entropy of Difference

The EoD is similar to the PE as it is also a modification of the Shannon Entropy. It is a complexity measure of time discrete series as well, but was introduced twelve years later in 2014 by Pasquale Nardone, see [51]. Again, we use the EEG signal of length n in an encoded form described in section 2.2, but this time we use the second introduced form of tuple generation. Hence, for a chosen order m and time-delay τ we have $k := n - (m-1)\tau$

tuples of length m according to the rule (2.1). The encoding in the different types of tuples $\omega_j \in \Omega_m := \{\omega\colon \omega \text{ is a string of symbols of length } m-1 \text{ containing "+" or "−"}\}$ happens as illustrated in Fig. 2.4 with the example with $m = 3$ and $\tau = 1$ by looking at neighbouring values of the tuple and determine if there is an increase, which gives a "+", or a decrease, which gives a "−". As mentioned above, the cardinality of the set Ω_m is $|\Omega_m| = 2^{m-1}$ and therefore there are 2^{m-1} different possible types of encoded patterns.

The EoD is calculated on the basis of the Shannon Entropy by

$$H_{ED}(m) := H_{ED}(p_1, ..., p_{2^{m-1}}) = -\sum_{j=1}^{2^{m-1}} p_j \operatorname{ld} p_j. \tag{4.9}$$

The base of the logarithm is two again. The variables p_j describe the probability distribution among the patterns, i.e. $p_j := P(\omega_j)$, with

$$P(\omega_j) = \frac{\#\{i\colon 1 \le i \le k, (x_i, ..., x_{i+(m-1)\tau}) \text{ has type } \omega_j\}}{k}.$$

As in equation (4.7), if a certain i exists such that $p_i = 0$ which would follow in a not defined summand due to the logarithm, we take the limit towards 0 and neglect the summand in the total sum. To take a look at the range that the EoD can obtain, we look at the two extremal cases of a probability distribution again which are on the one hand when there exists a i with $p_i = 1$ and on the other hand if $p_j = \frac{1}{2^{m-1}}$ for all $j \in \{1, ..., 2^{m-1}\}$. With the constraint

$$\sum_{j=1}^{2^{m-1}} p_j = 1$$

follows for the first case

$$H_{ED}(m) = -\sum_{j=1}^{2^{m-1}} p_j \operatorname{ld} p_j = -p_i \operatorname{ld} p_i = -1 \cdot 0 = 0.$$

For the second case we get

$$\begin{aligned} H_{ED}(m) &= -\sum_{j=1}^{2^{m-1}} p_j \operatorname{ld} p_j = -2^{m-1} \frac{1}{2^{m-1}} \operatorname{ld}\left(\frac{1}{2^{m-1}}\right) \\ &= -\left(\operatorname{ld} 1 - \operatorname{ld}(2^{m-1})\right) = \operatorname{ld}(2^{m-1}) = m - 1. \end{aligned}$$

The last step of the equation follows since the logarithm is of base two. Hence, we get, as Claude Shannon already mentioned in his work [63], that the range of H_{ED} is $[0, m-1]$. This gives the opportunity to define a normalised measure to have a comparison independently of m and also to the PE.

$$\hat{H}_{ED}(m) := -\frac{1}{m-1} \sum_{j=1}^{2^{m-1}} p_j \operatorname{ld} p_j \tag{4.10}$$

The measure $\hat{H}_{ED}$ has a range of $[0, 1]$, which can easily be verified by looking at the calculations of the extremal values above, but dividing by the factor $m - 1$. Like for the PE, higher values indicate large complexity. The normalised EoD as defined in equation 4.10 is from now on the meant measure when speaking about the EoD.

The EoD has already been used in the field of EEG analysis for sleep or wake behaviour in rats, see [37]. The authors used different settings to investigate different behaviours, i.e. order $m = 5$ and time-delay $\tau = 1$ for fast dynamics in the EEG and $\tau = 6$ for slow dynamics. They differentiated between two groups, one having Alzheimer's disease and one control group. The calculation showed increased values of the EoD for awake rats of the control group in comparison to when they were sleeping. In the group of rats with Alzheimer's disease this difference was not found, but they showed lower EoD values in the wake state than the control group, indicating lower complexity during this state for that group. However, one has to keep in mind that this experiment was done with rats and one cannot conclude for our race. The application of the EoD in this work will focus on sleep stage classification of humans.

4.4. Kullback Leibler Divergence and Cross Entropy

The concept of the next feature has been invented decades ago by Solomon Kullback and Richard Leibler, see [40]. In 1951, they were concerned about the problem of measuring divergence between populations using statistics. This involves the difference of information between two groups of people or animals. Using Shannon's theory about measuring information, which is described in section 4.1 and [63], they defined a measure to determine the loss of information when using one signal to predict another one.

Firstly, the KLD was introduced for continuous problems but also extended for applications on time-discrete series. Following the notation of [40, 50, 28], the measure is defined as

$$KL(p, q) := \sum_{j=1}^{n} p_j \log\left(\frac{p_j}{q_j}\right), \tag{4.11}$$

with $p = (p_j)_J$ and $q = (q_j)_J, J = \{1, ..., n\}$ describing the probability distributions of two time-discrete signals of length n. The second argument q is the distribution of the signal that is used to predict the signal whose distribution is the first argument p. The number of summands n is the number of possible outcomes, which in our case depends on whether we are looking at the KLD associated with the PE, which is described in section 4.2, or associated with the EoD, described in section 4.3. Depending on the order m, in the first case it holds $n = m!$, since there are that many possible patterns after encoding for the PE. In the second case $n = 2^{m-1}$ as for the EoD there are that many patterns that can be achieved. In literature the KLD is also often described as relative entropy of p with respect to q, see [17].

The lower boundary of the KLD is 0. This can be proven with Jensen's inequality, as it is done in [17, 50].

Theorem 4.4.1 (Jensen's inequality)**.** *For any convex function f, we have that*

$$f\left(\sum_{j=1}^{n}\lambda_j x_j\right) \leq \sum_{j=1}^{n}\lambda_j f(x_j), \tag{4.12}$$

where $\lambda_j \geq 0$ and $\sum_{j=1}^{n}\lambda_j = 1$.

Proof. The proof can be found in [17]. □

A function f is convex if for every $k, l \in \{1, ..., n\}$ and $0 \leq \lambda \leq 1$ it holds that

$$f(\lambda x_k + (1-\lambda)x_l) \leq \lambda f(x_k) + (1-\lambda)f(x_l).$$

It is called strictly convex if this equation holds only for $\lambda = 0$ or $\lambda = 1$. A function f is called (strictly) concave if $-f$ is (strictly) convex. Knowing that, we can prove the lower boundary assumption stated above.

Theorem 4.4.2 (Information inequality)**.** *Let p, q be two probability distributions, i.e. $\sum_{j=1}^{n} p_j = 1$ and $\sum_{j=1}^{n} q_j = 1$. Then*

$$KL(p, q) \geq 0$$

with equality if and only if $p_j = q_j$ for all $j \in \{1, ..., n\}$.

Proof. Following [17], let $A = \{j : p_j > 0\}$ be the support of p. With the conventions based on continuity arguments and the limit towards 0, as analogously stated in equation (4.7), it holds that

$$\lim_{p_j \to 0} p_j \log\frac{p_j}{q_j} = 0, \qquad \lim_{q_j \to 0} 0 \log\frac{0}{q_j} = 0 \qquad \text{and} \qquad \lim_{q_j \to 0} p_j \log\frac{p_j}{q_j} = \infty.$$

Consequently, since the logarithm function is concave it follows that

$$\begin{aligned} -KL(p, q) &= -\sum_{j\in A} p_j \log\frac{p_j}{q_j} = \sum_{j\in A} p_j \log\frac{q_j}{p_j} \\ &\underset{(4.12)}{\leq} \log\sum_{j\in A} p_j \frac{q_j}{p_j} = \log\sum_{j\in A} q_j \\ &\leq \log\sum_{j=1}^{n} q_j = \log 1 = 0. \end{aligned}$$

Since the logarithm function is moreover strictly concave, the Jensen inequality gives an equal sign if and only if $\frac{q_j}{p_j} = c$ for some $c \in \mathbb{R}$ and all $j \in \{1, ..., n\}$. Consequently, when looking at the next inequality it only holds equality if and only if

$$\log\sum_{j\in A} q_j = \log\sum_{j\in A} c p_j = \log c \sum_{j\in A} p_j = \log c \overset{!}{=} 0.$$

Hence, $c = 1$ and therefore $KL(p, q) = 0$ if and only if $p_j = q_j$ for all $j \in \{1, ..., n\}$. □

There is no upper boundary for this measure because if there exists an $i \in \{1, ..., n\}$ such that $q_i = 0$ and $p_i \neq 0$, the corresponding summand in the definition of the KLD in equation (4.11) would tend toward ∞ and hence, the whole sum would equal ∞, see [17]. Consequently, no normalisation can be defined since no maximal value can be given.

The KLD can be computed with the PE and the EoD as underlying coding separately, which means that the base of the logarithm is two. This results in the following two definitions.

$$KL_{PE}(p, q) = \sum_{j=1}^{m!} p_j \operatorname{ld}\left(\frac{p_j}{q_j}\right) \tag{4.13}$$

$$KL_{ED}(p, q) = \sum_{j=1}^{2^{m-1}} p_j \operatorname{ld}\left(\frac{p_j}{q_j}\right) \tag{4.14}$$

Based on an entropy measure and the KLD the CE can be defined, see [50]. It is determined as the sum of both of them. Hence, we have two new measures, dependent on the underlying entropy method, either the PE or the EoD. To shorten the definitions of the CE, the sum is now seen over all $j \in \{1, ..., m!\}$ or $j \in \{1, ..., 2^{m-1}\}$ analogously to equations (4.13) or (4.14) depending on the underlying entropy method. As the entropy methods defined in lines (4.6) and (4.9), respectively, depend on the order m as a symbol of the size of the probability distribution, but with differing definitions in literature for the measures, we set $H(p) = H(m)$ for our following definition of the CE.

$$\begin{aligned} H(p, q) &:= H(p) + KL(p, q) \\ &= -\sum_j p_j \operatorname{ld} p_j + \sum_j p_j \operatorname{ld}\frac{p_j}{q_j} \\ &= \sum_j p_j(-\operatorname{ld} p_j + \operatorname{ld} p_j - \operatorname{ld} q_j) \\ &= -\sum_j p_j \operatorname{ld} q_j \end{aligned}$$

The interpretation of the CE is, according to [50], that it is the mean number of bits that are required to encode information of a source modelled by p, when the distribution q is used as basis for the encoding. When $p = q$ we get the PE or EoD $H(p, p) = H(p)$ which, accordingly, is the number of bits when using the same model for prediction. The KLD as the difference between the CE and the entropy measure is then the mean number of extra bits required for the information encoding, when using q instead of p for the prediction. Since there is no upper boundary for the KLD we also do not normalise the CE. Therefore, the CE is computed as the sum of the KLD and the entropy measure before eventual normalisation.

The KLD has been applied in the field of EEG analysis already in 2009 by Gupta et al. [28]. A modification of the measure was introduced, since a distance measure was needed. As the KLD is a non-symmetric measure, one has to redefine it such that it is symmetric by

$$KL_{dist}(p, q) := \frac{KL(p, q) + KL(q, p)}{2}.$$

The abbreviation "dist" specifies that this measure is now a distance measure in the mathematical sense. The authors use the KL_{dist} in combination with different machine learning algorithms for classification of different mental tasks like multiplication, letter composing, counting and rotating an imaginary figure as well as a baseline task where nothing was done. They compare the results with the same algorithm, but with the euclidean distance measure and find better classification. In 2017, the KLD was applied on an EEG for determining seizure/non-seizure episodes of patients who suffer from epilepsy by Quintero-Rincón et al. [56]. They also found that the measure is potentially useful for this application.

4.5. Granger Causality

The next parameter that will be focused on is the GC. It was introduced by Clive W.J. Granger in 1963 as a parameter for the field of econometrics, to face the problem of deciding if causality happens there and if feedback is present, see [27, 26]. Later on, the theory was enhanced and specified even more. Definitions of the GC exist in the time-domain as well as the frequency-domain. However, since the analysis model of the EEG as a time series is based only in the time-domain, the corresponding measure will be described here.

For the definition we follow the notation of Barnett and Seth [6]. Firstly, we have to give an introduction in vector autoregressive (VAR) theory. Let $n, m \in \mathbb{N}$. We take a look at finite sequences of measurements or observations of a stochastic process of the form $(X_t)_I \in (\mathbb{R}^m)^I$ with $I = \{1, ..., n\}$ and simply write X for these time series. They are also called multivariate processes. For two such time series X and Y we now want to investigate whether and how the series Y influences a process X, and vice versa. To do this, first consider the following VAR model of order p, VAR(p) for short, of the form

$$X_t = \sum_{i=1}^{p} A_i' X_{t-i} + \varepsilon_{X_t}. \tag{4.15}$$

It is assumed that each value X_t of the time series results from the p previous values X_{t-i}, which are already known and weighted with the regression coefficients $A_i' \in \mathbb{R}$. The p previous values X_{t-i} are then used to calculate the lag value. In this context, X_{t-i} is also called the i-th lag value of X_t. The error term ε_{X_t}, also called the residual, indicates the deviation of the modelled value from the actual value X_t. In the next step, consider the model

$$X_t = \sum_{i=1}^{p} A_i X_{t-i} + \sum_{i=1}^{p} B_i Y_{t-i} + \varepsilon_{XY_t}. \tag{4.16}$$

In this model, not only lag values of X are used to model the value X_t, but also those of Y. The coefficients $A_1, ..., A_p$ in this model generally do not match the coefficients $A_1', ..., A_p'$ of the other model. If this is would be the case, i.e. for all $i = 1, ..., p$ hold $B_i = 0$ it would mean that the lag values of Y do not contribute anything to the modelling of X. Hence, there would be no conditional dependence of X on the past of Y.

The definition of the GC is based on the Maximum likelihood theory that suggests an analysis method of parametric data modelling. For models as given in equations (4.15) and

(4.16) the log-likelihood ratio statistic is used. The test statistic can be used to test the null hypothesis, as mentioned above, that $B_i = 0$ for all $i = 1, ..., p$. This means that there would be zero causality. GC is therefore a measure that tests whether X can be modelled significantly better with lag values of Y than without.

The GC is then defined using the log-likelihood ratio of the absolute value of the covariances of the residuals ε_{X_t} and ε_{XY_t}, see [7], as

$$\mathcal{F}_{Y\to X} := \ln\left(\frac{|\mathrm{Cov}(\varepsilon_{X_t})|}{|\mathrm{Cov}(\varepsilon_{XY_t})|}\right). \tag{4.17}$$

Since we only take a look at univariate time series, the covariance equals the variance, i.e. $\mathrm{Cov}(X) = s^2$, which for a time series X is defined by the mean value $\bar{x}$ as

$$\bar{x} := \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad s^2 := \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

The Multivariate Granger Causality Toolbox, as presented in [6], is a MATLAB toolbox using the GC in different varieties for analysing time series in general. However, it is already intended for the use in the field of neuroscience. The method is used during drug-induced anaesthesia for example. As it is shown in [7] there is an increase of the GC during loss of consciousness. Causal interactivity in the system was presented as well. The measure has also been applied in sleep scoring. Saghayan gives an example of an automated sleep stage classification method in [59] using the GC in combination with a SVM with a Gaussian kernel as a machine learning approach which is a non-linear classifier. With the GC in different varieties they used 30 features in total by calculating the measure between all electrodes and all possible directions. The model was applied to EEG measurements of ten healthy probands and trained and tested in a ten fold classification. The authors gained an accuracy of 72.7% and $\kappa = 0.65$.

The GC accounts for two features in our model as we use multiple channels.

4.6. Statistical Features

After the description of all these measures we now have eight features for the model up until now, i.e. age, the PE, the EoD, the KLD as well as the CE with respect to both of the entropies and the GC. The other features that are used are statistical ones and are presented in this section. The notation of the parameters is taken from [48]. For all values a time series $(x_t)_I, I = \{1, ..., n\}$, of length n is considered.

Firstly, the central measures are explained. Starting with the arithmetic mean or arithmetic average $\bar{x}$ is defined as the sum of all values divided by the length of the series.

$$\bar{x} := \frac{1}{n}\sum_{i=1}^{n} x_i$$

Since this parameter takes all values with the same weight into account, it is very sensitive for outliers, which means that if the extremal values, i.e. minimum or maximum, are very low or high, respectively, they have a big impact on the arithmetic mean. There is a

modification of the arithmetic average, which is the trimmed arithmetic mean. This cuts a fixed percentage of outliers off the series and calculates the parameter with the reduced amount of values. This measure was not included in the list of features. With the geometric and the harmonic average there are also two other kinds of mean. The geometric one $\bar{x}_g$ is defined as the n-th root of the product of all the values of the series.

$$\bar{x}_g := \sqrt[n]{\prod_{i=1}^{n} x_i}$$

However, since the time series $(x_t)_I$ can also include negative values and n can be an even number, it could be the case that this parameter is not defined in $\mathbb{R}$ as it could be a complex number. That is the reason why we did not include the geometric mean as a feature in the model. The harmonic mean $\bar{x}_h$ is defined as

$$\bar{x}_h := \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}.$$

Hence, the reciprocal of this measure is the arithmetic mean of the reciprocals. There are also central measures which are more robust, i.e. less sensitive, to outliers. The mode x_{mod} is very easy to determine because it is the value in the series with the highest frequency. For the definition of the median or central value $\tilde{x}$ we first have to arrange the series according to the size of the values. Then, we get a series which we will label with $x_{[1]}, ..., x_{[n]}$. The median is then defined, dependent on the fact if n is even or odd, as

$$\tilde{x} := \begin{cases} x_{\left[\frac{n+1}{2}\right]}, & \text{if } n \text{ is odd,} \\ \frac{1}{2}\left(x_{\left[\frac{n}{2}\right]} + x_{\left[\frac{n}{2}+1\right]}\right), & \text{if } n \text{ is even.} \end{cases} \tag{4.18}$$

This means that the median is literally the central value if n is odd or the arithmetic mean between the two central values if n is even.

Secondly, we also look at measures of dispersion. The range R of a time series is given by the subtraction of the maximal value and the minimal value. Using the ordered series which was labelled above as $x_{[1]}, ..., x_{[n]}$ we get

$$R := x_{[n]} - x_{[1]}.$$

As we already discussed with the mean, the range is also very sensitive to outliers. The variance s^2 is widely known and given by

$$s^2 := \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

Because of its definition it is also often called mean squared deviation. The standard deviation s is based on the variance and the square root of it. Hence, it is, on the contrary to the variance as a squared measure of dispersion, a linear one.

$$s := \sqrt{s^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Both of these parameters take the arithmetic mean into account. A parameter that calculates the dispersion of the median is the mean absolute deviation d. It is defined as

$$d := \frac{1}{n} \sum_{i=1}^{n} |x_i - \tilde{x}| .$$

The median $\tilde{x}$, as presented in equation (4.18), has the important property that 50% of the values of the time series are higher or equal and 50 % are lower or equal than that parameter. This can be generalized by a p-quantile x_p. It is defined, such that $p \cdot 100$ % of the values are lower or equal and $(1-p) \cdot 100$ % of the values are higher or equal than x_p. Hence, the median is nothing less than the quantile $x_{0.5}$. For the definition, the floor operation $\lfloor . \rfloor$ is used which gives the first integer that is lower or equal. Again, we use the ordered sequence $x_{[1]}, ..., x_{[n]}$.

$$x_p := \begin{cases} x_{[\lfloor np \rfloor + 1]}, & \text{if } np \text{ is not an integer,} \\ \frac{1}{2}\left(x_{[np]} + x_{[np+1]}\right), & \text{if } np \text{ is an integer.} \end{cases}$$

Other special cases of the p-quantile are the 0.25-quantile and 0.75-quantile which are also called the lower and upper quartile respectively. These are used as features for the model. Another measure, the interquartile range Q, is similarly defined to the range but with the difference of the quartiles.

$$Q := x_{0.75} - x_{0.25}$$

The last two statistical measures are the shape parameters skewness $\hat{\gamma}_1$ and kurtosis $\hat{\gamma}_2$. They are defined as

$$\hat{\gamma}_1 := \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^3 ,$$

$$\hat{\gamma}_2 := \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^4 .$$

The skewness describes if it is more likely to be on the right or on the left side of the arithmetic mean. The kurtosis stands for the concentration around the average and consequently, how strongly the edges are occupied.

This gives us a total of 22 features that are used as the basis for our model approach for EEG analysis, which is described in the next chapter.

5. Building a Classification Model

After introducing the features, a machine learning model for sleep scoring is built. Important properties of this model should be, as described in section 3.2, that it is interpretable and still able to achieve good results. It should be valid not only for healthy subjects but also hold for patients suffering from sleep or other neurological disorders. The examined dataset is the CAP Sleep Database published in 2001, see [67], which is presented in section 5.1. Furthermore, the data processing, which was done for the application, is presented in this section as well. In the following section 5.2, the implementation and settings of the features presented in chapter 4 is explained. The decision process of which model is chosen is difficult, the different machine learning algorithms and their properties are described in section 5.3. The achieved results are further discussed in chapter 6.

5.1. CAP Sleep Database and Data Preprocessing

The CAP Sleep Database [67] contains 92 polysomnography recordings of patients with various sleep disorders and 16 of patients without any sleep related or neurological disorders, made at the Sleep Disorders Centre of the Ospedale Maggiore of Parma in Italy. The purpose of this publication was to provide a large number of examples where not only the sleep stages but also the CAP stages are annotated. CAP are cyclic EEG patterns of brain activity followed by deactivation. It is a sign of restlessness during sleep and an omen of sleep disorders as a very high number of CAP phases are examined in various sleep disorders, such as insomnia (ins), periodic leg movements (plm), sleep-disordered breathing (sdb), REM behaviour disorder (rbd) or nocturnal frontal lobe epilepsy (nfle). In EEGs of patients suffering from narcolepsy (narco), CAP phases are also often present. Recordings of patients diagnosed with each of the listed disorders are included in the database, as well as recordings of two patients suffering from bruxism (brux) and 16 healthy subjects (n). The latter are called the control group and did not take any medication that affects the central nervous system. The abbreviations in brackets are the anonymised patient IDs which, in combination with numbering, provide a unique allocation of the recording to a subject, for example nfle13 or n1 who was mentioned in section 3.1 for the sleep course over night in figure 3.1. In table 5.1, the database with its 108 recordings is broken down into each disorder and its abbreviation as well as the number of available patients.

The database includes at least two EEG channels, two EOG channels, two EMG channels, ECG and respiratory signals and the age of the subjects. Additional EEG channels are often given as well. As we only want to use EEG measurements for our model we only look at this information. The placement of the electrodes was done as usual according to the 10-20 system, see figure 2.2. The sleep stage classification was done by expert neurologists educated by the already mentioned sleep disorders centre according to the R&K rules. For this purpose, the software REMlogic^{TM} by Embla® was used. The classifications are given

Table 5.1.: Breakdown of the CAP database regarding its disorders.

Disorder	Abbreviation	Number of Recordings
No pathology (control group)	n	16
Bruxism	brux	2
Insomnia	ins	9
Narcolepsy	narco	5
Nocturnal frontal lobe epilepsy	nfle	40
Periodic leg movements	plm	10
REM behaviour disorder	rbd	22
Sleep-disordered breathing	sdb	4

among other formats in a .txt file, the authors of the publication provided a MATLAB script for reading these as well. The polysomnography recordings are given in a .edf format which was imported in MATLAB using the `import_edf` function, alongside other functions needed, written by Sebastian Berger et al. and published in [9] with adaptions to our specific needs. The European data format is the standard format in this field. This way we could import the EEG channels and the information of the recordings in MATLAB, the other measurements were excluded since we do not need them for our model.

For our purpose of building a sleep scoring model, we want to use the modern classification rules according to the AASM. Hence, we merge S4 and S3 to N3 and get the sleep stages W, R, N1, N2 and N3 as explained in section 3.1.

Even though the original intention of the publication of the dataset was to have a large amount of recordings with CAP annotations, we will only be using the sleep scoring of the dataset. This data is very useful for our intention of building a sleep scoring model that is valid for all subjects independent of their health status, as it contains many datasets of patients suffering from sleep or other neurological disorders. As it is not our focus to diagnose sleep disorders, we treat every recording the same way in the model creation process. The CAP annotations are therefore irrelevant for us. Another advantage of the dataset is that the selection of patients is very heterogeneous as it includes people aged from 14 to 82 years old and 66 male and 42 female individuals. The exact breakdown of the subjects' ages and genders is given in the publication.

The publication does not mention anything about the settings and the preprocessing of the recordings. Hence, we had to investigate these ourselves. We noticed a large difference in the setup between the subjects regarding the filter settings and the sampling frequency, when only looking at the insomnia patients and the control group. For the first problem, we looked at the frequency spectrum of the EEG signals using the `fft` function of MATLAB. It computes the discrete Fourier transform using fast Fourier transformation. With the MATLAB function `fftshift` we could shift the zero-frequency component to the centre such that the plot of the frequency spectrum is interpretable and symmetric around this component. We found out that some of the recordings seemed to have a small highpass filter while others did not have any, for example n4. This is shown in figure 5.1, where

on the left hand side the unfiltered signal of the channel Fp2-F4 is shown. On the x-axis the frequency in Hertz and on the y-axis the power per frequency in decibel per Hertz are given. One can see that the frequency spectrum shows an outlier in the zero-frequency component in the middle. As described in section 2.1 such low frequent noise happens for example due to breathing, that is why a highpass filter is necessary. On the right hand side of the figure the same EEG signal is pictured but filtered with a 0.5Hz highpass and a 30Hz lowpass filter. Here we do not see the outlier anymore.

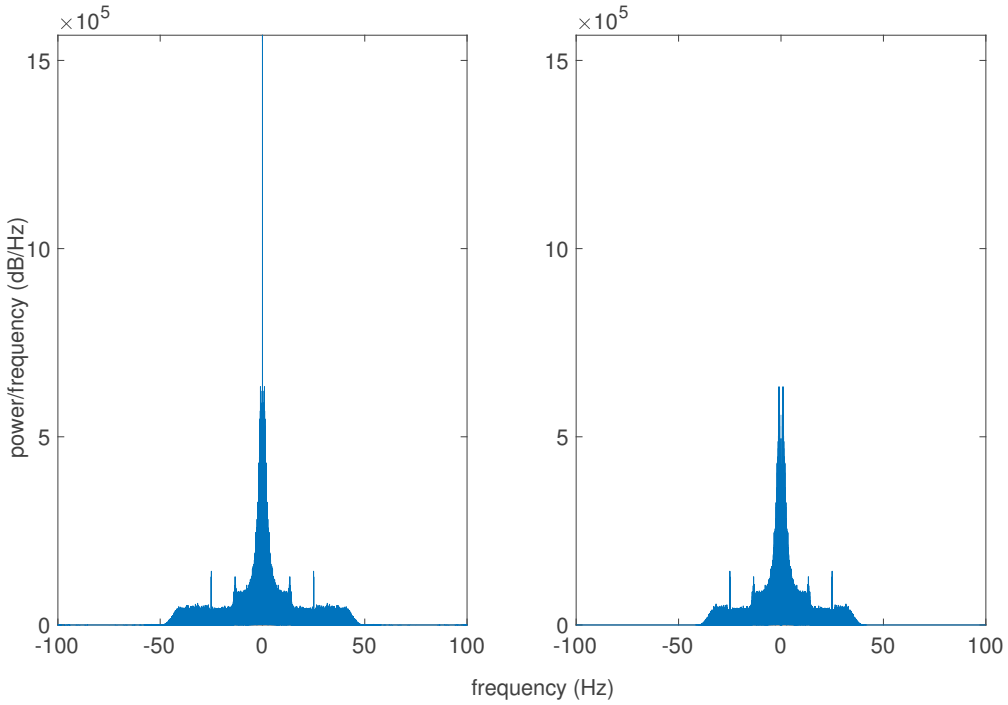

Figure 5.1.: The figure shows the frequency spectrum of the EEG signal of the channel Fp2-F4 using the example of subject n4. On the left hand side no additional filter is used, on the right hand side a lowpass filter of 30Hz and a highpass filter of 0.5Hz are used. It illustrates the necessity of a highpass filter.

Apart from the missing highpass filter we also discovered different lowpass filters applied to the recordings. Most of them showed a lowpass filter of maximal 50Hz, but some seemed to do not have any. As an example, the frequency spectrum of subject ins2 is shown in figure 5.2. Again, on the left hand side the unfiltered spectrum of the EEG signal of the channel Fp2-F4 is shown, with the frequency in Hertz on the x-axis and the power per frequency in decibel per Hertz on the y-axis. Even frequencies up to around 100Hz still have a visible magnitude, bearing in mind that the scale of the y-axis is times 10^5. Due to reasons of noise, as explained in section 2.1, a lowpass filter of 30Hz and a highpass filter of 0.5Hz are applied to the signal. The corresponding frequency spectrum is shown on the right hand side of the figure. With these two filters we have a consistent processing of the EEG recordings. The MATLAB pre-implemented functions `lowpass` and `highpass` are used.

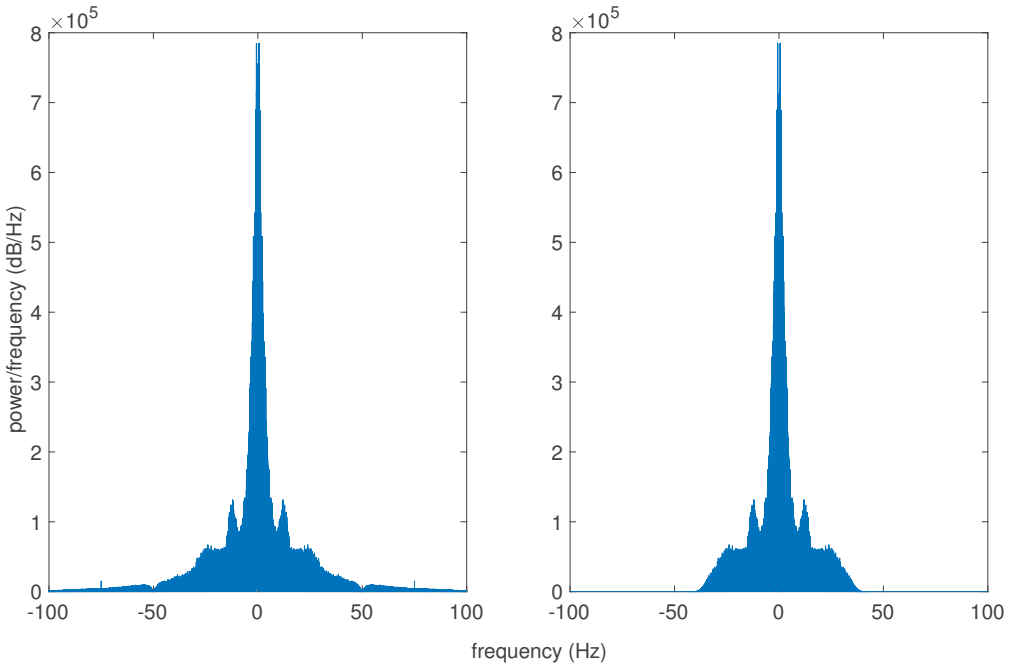

Figure 5.2.: The figure shows the frequency spectrum of the EEG signal of the channel Fp2-F4 using the example of subject ins2. On the left hand side no additional filter is used, on the right hand side a lowpass filter of 30Hz and a highpass filter of 0.5Hz are used. It illustrates the necessity of a lowpass filter.

The sampling frequency was the second important parameter regarding the signal recording and processing setup for which we faced a huge difference in all datasets. It ranges from 100Hz to 512Hz. This problem can be easily solved by simply resampling the EEG recordings. Our set parameter is 200Hz, as [60] suggests. For this, we used the function `par_resample`, which was published by Sebastian Berger et al. in [9]. It is based on the MATLAB function `resample` but uses multiple CPU scores for the computation, if it is run on GNU Octave with the parallel package. As we run our implementation in MATLAB, it simply uses the `resample` function as it is.

After the preprocessing of the EEG signals we interpret them as time series. We had to cut each of the time series into suitable lengths, as the annotation of the sleep scoring was not always given for the full recording but sometimes started later or finished earlier. This way, we had for a certain time period of 30s, where the sleep stage was given, a corresponding EEG signal as direct comparison.

Unfortunately, when importing the data in MATLAB we had to exclude four datasets. Three of them were not possible to load, which were brux1, n13 and n14. The first one was a missing sample and the others were excess samples. The fourth problematic data base was nfle27 because the file that contains the sleep stage annotations is a part of brux2 as this patient ID was written on top of the .txt file.

5.2. Implementation of the Feature Extraction

The filtered, resampled and cut recordings are the input for the model that we want to build. Another input are the ages of the examined subjects. We use up to three EEG channels always depending on the availability of the signals. The main channel is the one for which most of the features described in chapter 4 are calculated. As brain activity during sleep results mostly in electrical activity in the frontal region, see [12, 61], channels of electrodes that are placed there should be the main signal. Hence, the channel Fp2-F4 or Fp1-F3 are used whenever possible. If both are available, Fp2-F4 is preferred, if none of them is available, the next frontal will be used. Two other EEG signals are taken as well as an input for the calculation of the GC. For this, we use an EEG channel that is placed on the back of the head in the parietal region, i.e. P4-O2 or P3-O1, and a channel that is placed in the centre of the head, i.e. C4-A1 or C3-A2. The GC is then calculated two times. By defining the signals as F for the main and P and C for the other two EEG signals, we compute $\mathcal{F}_{P\to F}$ and $\mathcal{F}_{C\to F}$ following the notation of section 4.5. The signals are cut into windows of 6000 datapoints corresponding to 30s which is the time period that the sleep scoring also uses for the annotation of the stages. This correspondence is needed for the training of the model and as a comparison when it is tested.

The possible outputs of our model are the five sleep stages W, R, N1, N2 or N3. This connection between input and output is illustrated in figure 5.3. A closer look at the model in between will be given hereafter.

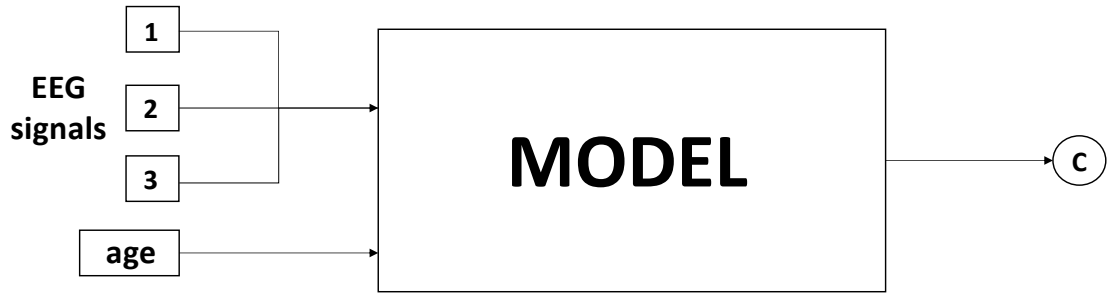

Figure 5.3.: Rough model structure for visualising input and output. The output C ∈ {W, R, N1, N2, N3} can take one of the five sleep stages.

After the definition of input and output, the parameters for the different features will be explained in detail. We need to set the order m and the time delay τ for the PE and the EoD, the corresponding KLDs and consequently for the CE, choose a model signal to predict the main EEG signal for the KLDs and define the order of the underlying VAR model for the GC. The regression coefficients for the VAR model in equations (4.15) and (4.16) will be estimated throughout the calculation. For the age and the statistical features we do not have to set any parameters since we already defined that we only use the lower quartile $x_{0.25}$ and the upper quartile $x_{0.75}$ of the p-quantiles.

For the PE as well as for the EoD we choose an order $m = 3$ and a time delay $\tau = 1$. To make this decision, we looked at different combinations of the values for the patients

ins1-ins9 and n1-n9 and calculated the correlation between the PE, the EoD and the KLDs with the sleep stages. For this, we had to assign each sleep stage a certain value, which are listed in table 5.2, and then used the `corrcoef` MATLAB function. We tested the parameters for m from 3 to 7 and for τ from 1 to 5. The results were that for the order we get a very similar correlation between the values $m = 3$ to $m = 7$. Hence, we chose $m = 3$ due to minimal computational effort. For the time delay the best correlation values were received for $\tau = 1$. For higher time delays lower correlation values occurred which can be explained using the findings of [54]. Popov et al. described that the product of the time delay and the sampling frequency is constant, i.e. higher time delays behave the same as lower sampling rates. A lower sampling frequency gives a worse resolution of the EEG signal which can lead to a wrong signal imaging, see [60]. This led us to the decision of choosing $\tau = 1$.

Table 5.2.: Assigned values for each sleep stage.

Sleep stage	awake	REM	NREM stage 1	NREM stage 2	NREM stage 3
Abbreviation	W	R	N1	N2	N3
Value	5	4	3	2	1

After defining the order and the time delay, the EEG signal can be divided and coded into the patterns following section 2.2. Firstly, looking at the encoding of the PE we introduce an algorithm that is based on the Lehmer code for further symbolisation, following [8]. This gives a simpler numerical representation of a pattern type using the mapping

$$(x_1, x_2, ..., x_m) \mapsto \sum_{i=1}^{m-1} \left((m-i)! \sum_{j=i+1}^{m} [x_i > x_j] \right).$$

The operation $\sum_{j=i+1}^{m} [x_i > x_j]$ on the right hand side is the right inversion count and gives the number of elements placed on the right of x_i that are higher than x_i. The function `symbolise` gives us the numerical representation. It was written once again by Sebastian Berger et al., published in [9, 8], but has been adapted and enhanced. Dependent on the order m, this function calls another one, which in the case of $m = 3$ is `encode_pe_dim_3`.

After this procedure our EEG signal represented by the time series $(x_t)_I, I = \{1, ..., n\}$, of length n, which was divided into $k := n - (m-1)\tau$ tuples of length m, is encoded into a numerical representation vector $y \in \{1, ..., m!\}^k$. This way, it is very easy to determine the probability distribution among all the possible pattern types and to compute the PE according to equation (4.8) afterwards.

Analogously, we could gain a simple numerical representation of the EoD patterns according to the mapping

$$(x_1, x_2, ..., x_m) \mapsto \sum_{i=1}^{m-1} \left(2^{m-i} [x_i > x_{i+1}] \right).$$

In comparison to the mapping for the PE, the term on the right hand side has a simpler structure because we only compare neighbouring values to determine the pattern and do

not need to compare the values of the whole tuple. For encoding the tuple, we wrote a function `symbolise_eod` that calls dependent on the order another one which is named for $m = 3$ `encode_eod_dim_3`.

As before, we have a numerical representation of our time series $(x_t)_I, I = \{1, ..., n\}$, and with $k := n - (m-1)\tau$ a vector $y \in \{1, ..., 2^{m-1}\}^k$ is received as the output, containing the pattern types of the k tuples we divided the series into. The probability distribution is calculated and used for the computation of the EoD, according to equation (4.10).

Next, a reference signal for predicting the main signal with the KLD is searched. For this we take a look at different types of noise, especially white, pink and brown noise, which are defined according to their spectral density $\frac{1}{f^\beta}$, see [34]. Noise signals in general are produced by stochastic processes. The variable β takes for white, pink and brown noise the values $\beta \in \{0, 1, 2\}$ respectively, i.e. white noise has an equally distributed frequency spectrum, pink shows a decrease, the higher the frequency the lower the power density, and brown an even larger decrease. Figure 5.4 shows the power spectral density (PSD) of the three kinds of noise and illustrates the difference between them. The signals are created in MATLAB using the `dsp.ColoredNoise` function, for the PSD we used the function `fft`, similar to above.

White noise has an almost constant frequency spectrum, meaning that high frequencies occur as often as low ones, which results in a flat curve for the course. Pink and brown

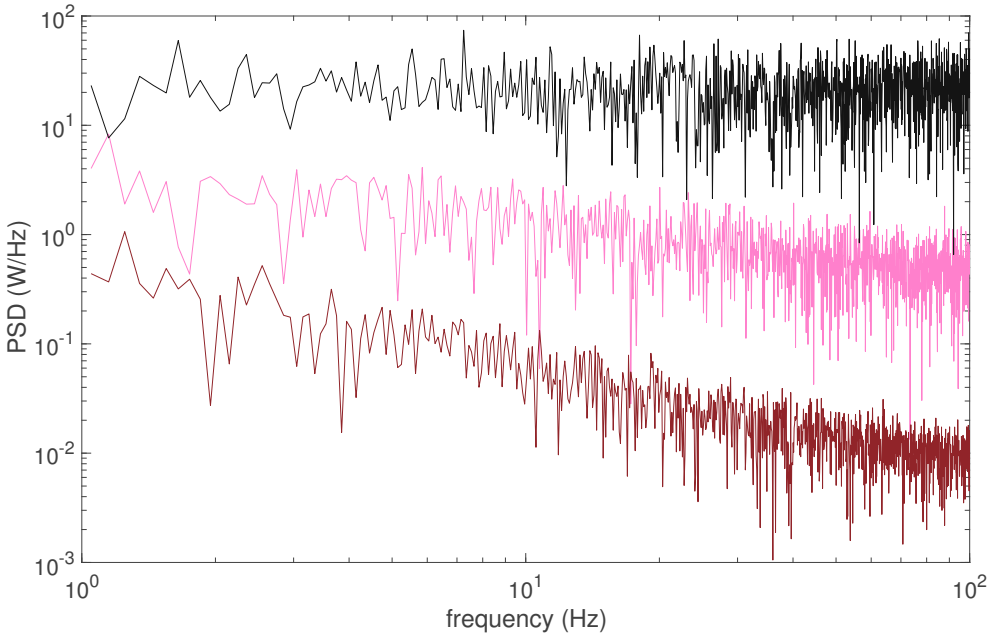

Figure 5.4.: PSD of white, pink and brown noise starting at 1Hz with white depicted in black and pink and brown in their respective colours. A sampling frequency of 200Hz and a duration of 10s are chosen. The `loglog` plot shows the equal distribution of the frequencies for the white noise and a decrease in power when the frequencies increase for pink and even more for brown noise.

noise, on the contrary, already resemble waves as low frequencies prevail more and more with reduced occurrence of high frequencies. These two kinds of noise look more rhythmic. Examples of the courses of the different signals are given in figure 5.5, which show the described properties.

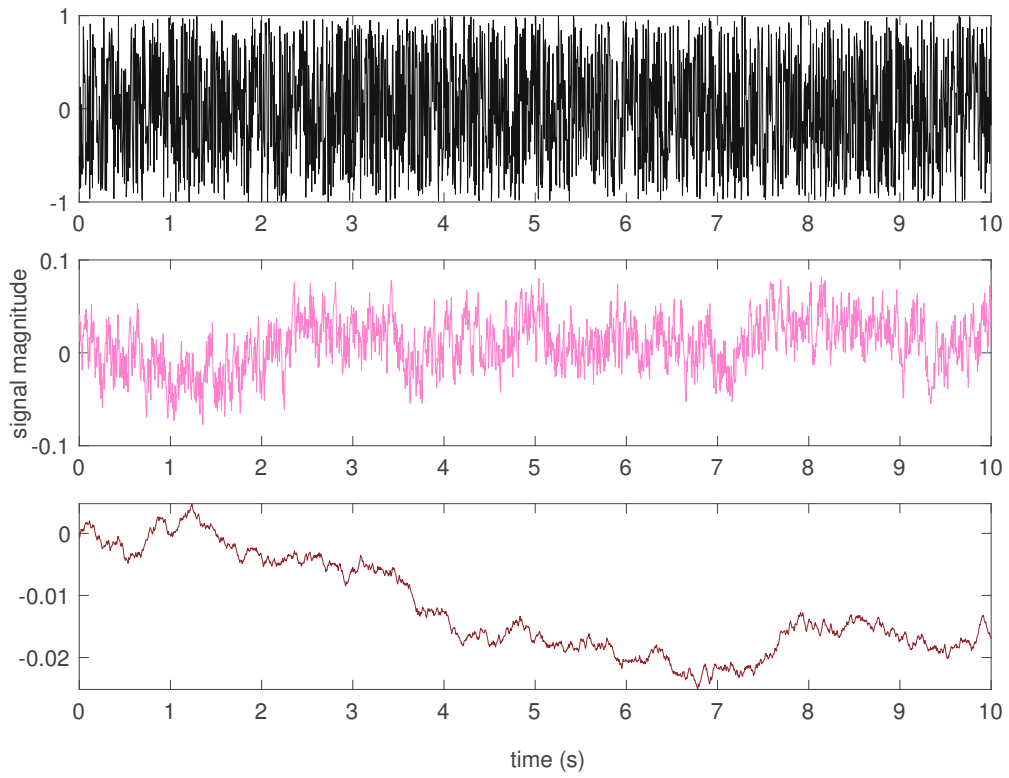

Figure 5.5.: Example for white (top), pink (middle) and brown (bottom) noise signals with white depicted in black and pink and brown in their respective colours. A sampling frequency of 200Hz and a duration of 10s are chosen. The magnitude of each signal is illustrated on the y-axis over the time given in seconds on the x-axis. White resembles a complete random signal without any structure, but pink and brown, on the contrary, show wave-like behaviour.

In 2001, it has been already described that pink noise very often appears in biosystems, e.g. instabilities in solid state physics, lasers or fluid instabilities, see [65, 29]. The dynamic processes in these systems are self-organizing and stochastic and seek the lowest available energy as equilibrium. Such processes are associated with pink noise. Human brain activity during wakefulness can also be compared to pink noise as it is a good approximation, see [15, 11]. The brain activity shows a steeper decrease for higher frequencies during an unconscious state, see [15], and hence resembles brown noise more.

Christoph Bandt defined in [4] a distance between the PE and white noise. Patients suffering from insomnia and healthy subjects of the control group of the CAP Sleep Database

were examined using this measure on one EEG channel. The results were, that it is a good estimate of sleep depth and patients suffering from insomnia had a smaller distance than healthy people had.

These results motivate to take white noise as the reference model signal for the KLD because it measures the divergence of the examined signal to the reference one. As white noise has PE values of almost one, see [4], it also shows the highest possible values. Another reason to choose white noise is that since the KLD is always nonnegative, as shown in section 4.4, the parameter does not measure in which direction the divergence is directed. For example, if pink noise would be the reference model and the KLD has a certain value, we could not interpret if it would tend more towards white or towards brown noise. Hence, the choice for the reference signal q for the KLDs in definitions (4.13) and (4.14) is white noise.

For the generation of white noise, we used the MATLAB function `dsp.ColoredNoise`. The reference signal should have the same settings as the EEG signal. Therefore, we created the noise with the same length as the EEG signal, such that the resampling frequency of 200Hz is maintained. Afterwards, the noise signal is filtered the same way as the EEG signals with a lowpass filter of 30Hz and a highpass filter of 0.5Hz.

The calculation of the CE is done with the PE and the EoD of equations (4.6) and (4.9), respectively, i.e. without any normalisation. This is done because the CE is the sum of the entropy measure and the KLD. Since the KLD has no normalisation, we also do not want to take it into account for the entropy measure.

Up until now, only the main EEG channel has been used. For the GC, we also use the other two channels, however they are only used for this measure. Firstly, we have to define the order p of the underlying VAR model. The Multivariate Granger Causality Toolbox, written by Lionel Barnett and Anil K. Seth, see [6], was used for the computations. The main function we use for the calculation of the time-domain GC $\mathcal{F}_{P\to F}$ is `GCCA_tsdata_to_mvgc`. The second GC value using the EEG channel placed at the centre of the head is computed analogously. The abbreviation GCCA stands for Granger causal connectivity analysis. The arguments of the function are the time series representing the two EEG channels of the 30 second windows, the indices of the examined channel F and the signal, that could possibly be an influence P, as well as the order of the VAR model p. To take a look at what orders are possible, we tested the datasets of the subjects n1 to n9 and ins1 to ins9 and discovered that the VAR model can only be generated for the orders $p \in \{1, 2\}$. For higher orders, the model could no longer be created as it was unable to estimate parameters using the 6000 data points of the two EEG channels, respectively. Consequently, we tested the function once for $p = 1$ and once for $p = 2$ and compared the results.

The main function calls the function `tsdata_to_var` of the toolbox two times, once for the reduced model in equation (4.15) and once for the full one in equation (4.16), which fits the regression coefficients A_i' and A_i, B_i and the residuals ε_{X_t} and ε_{XY_t}. The ordinary least squares (OLS) algorithm is applied, which estimates the coefficients as well es the residuals. With these, the covariance of the residuals can be estimated as well, and thus, the GC $\mathcal{F}_{P\to F}$ can be directly computed with these values according to equation (4.17). The algorithm is explained in detail in appendix A. There could also be the possibility of calculating the GC more accurately by computing an autocovariance sequence using

the estimations received by the function `tsdata_to_var` with the function `var_to_autocov` and afterwards calculating the time-domain GC using `autocov_to_mvgc`. These functions require more computational effort. Due to efficiency reasons, this method was waived and consequently the GC is computed with the estimated covariance of the residuals that have been obtained by the function `tsdata_to_var`.

The results for the GC for the different orders p are listed in table 5.3. We examined the time-domain GCs $\mathcal{F}_{P\to F}$ and $\mathcal{F}_{C\to F}$ for the channels P4-O2 and P3-O1 as well as C4-A1 and C3-A2, respectively, whenever available. According to the channel, the GC in the table will be written as $\mathcal{F}_{X\to F}$ with $X \in \{P, C\}$. The mean of the absolute values of all possible calculations is given depending on the availability of the channel for the different patients' databases. We compare absolute values because we want to get the highest possible measures.

Table 5.3.: Mean of the absolute values of the GC for different orders p and different channels.

	P4-O2	P3-O1	C4-A1	C3-A2
$\mathcal{F}_{X\to F}$, $p = 1$	0.205	0.279	0.264	0.472
$\mathcal{F}_{X\to F}$, $p = 2$	0.215	0.195	0.182	0.43

When comparing these results we conclude to choose the VAR model order $p = 1$ as it shows higher values in more channels. For the central channel, we either take C4-A1 or C3-A2, if both are available then C3-A2 is preferred because of the higher value. However, for these two channels it was often the case that either one or the other was given, but never both. Out of the parietal channels we prefer to take P3-O1, if not, then P4-O2 is taken. The latter one is present in almost all recordings. The GC was also calculated for the other available channels, but did not show higher measures in the most frequently occurring EEG signals, in rarely occurring channels the value was sometimes higher, but not significantly. Consequently, these channels were our choice of interest for this feature.

As we already described, the used channels are either Fp2-F4 or Fp1-F3, C3-A2 or C4-A1 and P3-O1 or P4-O2. 97 out of the 104 used datasets had always given at least one of the two searched signals. In two of the recordings, nfle25 and nfle33, the signals are not labelled with the measure between two electrodes, as in the other patients, but with the measure of one electrode. From them, the signals Fp2, C3 and O1 are taken. The datasets of subjects n12 and n15 did not contain the correct frontal and parietal channel, for n12 we take C4-A1 as the frontal, C3-A2 as the central and O2-A1 as the parietal signal, for n15 F3-A2 is used for the frontal, C3-A2 for the central and O2-A1 for the parietal signal. The biggest problem regarding the channel selection we encountered with the datasets n6, n7 and n9. Only the channels C3-A2 and O2-A1 were available. There was no point in excluding them as well, otherwise the size of the control group would have been very small. Hence, we took the first signal as the frontal one and the second as the central and the parietal.

After defining all the parameters for our features and the channels for the EEG signals to be examined, the feature extraction can be done and the process of selecting a machine learning model, that classifies the data, begins.

5.3. Selection of the Model

It can be very difficult to choose a machine learning algorithm that fits the appropriate properties, i.e. being interpretable but still being able to receive a good accuracy. We follow a suggestion given in [39] to start with a few models, that are flexible but less interpretable as they have a higher probability of achieving the maximal accuracy. Then, we also use models that are not very complex until we reach a plain model, that is the easiest to interpret. This way, the performance can be analysed by comparing the achieved accuracy. Other properties, such as training time or prediction speed, can be important as well depending on the final application of the model. If, in the end, different algorithms have similar results, these properties are crucial. For us, the most important one is interpretability. The data is split into test and training data, with the latter one needed for building the model, more details on how this is done are given in chapter 6.

This guide led us to choose eight different machine learning algorithms for classification as the basis for the models. We run all of them separately and compare afterwards, so the order they are given has no meaning. The different models are a decision tree, LDA, a linear SVM, weighted KNN, bagged trees, boosted trees with AdaBoost, kernel logistic regression and naive Bayes classifier. The explanation and notation of the algorithms are based on [50, 39]. The implementation was done in MATLAB, using the Classification Learner application of the machine learning package. It can train different models, test them on given data and export them afterwards to the regular MATLAB workspace.

A decision tree is a method, that splits the input, i.e. the features, into smaller parts to minimise the misclassification error. Let the input be denoted by the matrix $X \in \mathbb{R}^{N \times D}$ with N being the number of samples and D the number of features. In our case, the number of samples equals the number of the examined 30s windows in our training data. The output $y \in \{1, ..., 5\}^N$ is given as well, which are the scored sleep stages for the various windows. It is needed for building the model, which is called growing a tree for this method. The output and the input are summarised as dataset $\mathcal{D}$. Each node in the tree has to decide whether a feature j of a sample $i, x_{ij} \in X$, is lower or higher than a certain threshold t. This is done by a split function, that gives back the best feature and corresponding threshold for that node.

$$(j^*, t^*) = \underset{j \in \{1,...,D\}, t \in \mathcal{T}_j}{\arg\min} (\text{cost}(\{X_i, y_i : x_{ij} \leq t\}) + \text{cost}(\{X_i, y_i : x_{ij} > t\}))$$

The vector X_i denotes the i-th row of the matrix X, i.e. the i-th sample, and $\mathcal{T}_j$ the possible thresholds, which can be determined by taking all the values that feature j obtains. The dividing also needs a cost function, we use the Gini index. Firstly, we calculate estimates for the class conditional probabilities for the classes $c \in \{1, ..., 5\}$ according to

$$\hat{\pi}_c = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}(y_i = c), \tag{5.1}$$

with $\mathbb{I} \in \{0, 1\}$ determining if the output y_i is the sleep stage c or not. The Gini index defined by

$$\sum_{c=1}^{5} \hat{\pi}_c (1 - \hat{\pi}_c) = \sum_{c=1}^{5} \hat{\pi}_c - \sum_{c=1}^{5} \hat{\pi}_c^2 = 1 - \sum_{c=1}^{5} \hat{\pi}_c^2$$

represents the expected error rate. The tree grows following the procedure 5.1 given as pseudocode, see [50].

Algorithm 5.1: Recursive algorithm for growing a decision tree for classification depicted from [50].

```
function fitTree(node, 𝒟, depth)
node.prediction = class label distribution;
(j*, t*, 𝒟_L, 𝒟_R) = split(𝒟);
if not worthSplitting(depth, cost, 𝒟_L, 𝒟_R) then
  return node;
else
  node.left = fitTree(node, 𝒟_L, depth+1);
  node.right = fitTree(node, 𝒟_R, depth+1);
  return node;
end
```

In MATLAB we set an abort criterion, such that the maximal number of splits is 100. This reduces the risk of overfitting. A major advantage of decision trees is that they are very interpretable, as they can basically be torn down to if-then queries. On the other side, two disadvantages are that they do not yield good classification accuracies compared to other algorithms and they are unstable, i.e. very sensitive to changes in the input data.

To overcome the last drawback, the method of bagging, also known as bootstrap aggregating, is introduced. It is an ensemble technique that takes a certain number of learners and handles the prediction of each one of them as a vote. The class with the highest number of votes has the highest probability for an accurate prediction and is therefore chosen by the algorithm. Bagged trees take decision trees as the underlying learners, for our model we take 30 and choose the maximal number of splits as 4185, as the predefined settings of MATLAB are for this classification model. Unfortunately, when using multiple trees the property of interpretability is lost.

Boosted trees are another modification in which several decision trees are combined to create a new model. A certain number of weak learners, in our case 30 decision trees, are the basis. They are called that way because they perform just a little better than a complete random classifier. This is achieved by setting the maximal number of splits to a rather low value, in our case it is 20. A boosting algorithm is used for model building procedure. We use AdaBoost, an abbreviation of adaptive boosting, which adds one weak learner that predicts the misclassified samples better than the model before in every step. The basic concept is that weights are given in each step, such that the misclassified samples receive a higher weight and correctly classified ones less in this iteration.

The original algorithm was invented for a binary classification problem, the enhanced one, that is used by MATLAB, is called AdaBoost.M2 and is explained as a pseudocode in algorithm 5.2. The procedure was introduced in 1997 by Yoav Freund and Robert E. Schapire, see [24]. The input is the dataset $\mathcal{D}$ that comprises of the feature matrix of all the samples $X \in \mathbb{R}^{N \times D}$ and the output $y \in \{1, ..., k\}^N$ with N being the number of samples, D the number of features and k the number of classes, in our case $k = 5$. The other inputs are the weak learners, WeakLearn, and the number of iterations T, which equals the number

of learners. We use 30 of them. The variables for the weights are $w_{i,c}^t$ with t specifying the current iteration, i the sample number and c the class, i.e. the sleep stage. A uniform distribution is used for initialisation.

In every iteration the procedure first calculates a label weight function q_t and a distribution D_t which is needed for the weak learner. The classifier then tries to minimise the pseudo-loss ε_t, which is defined in step 3. It is a modification of the misclassification error,

Algorithm 5.2: Boosting algorithm AdaBoost.M2 for an ensemble method to improve weak classifiers to a better performing model depicted from [24].

function AdaBoost.M2($\mathcal{D}$, WeakLearn, T)
Initialize weight vector $w_{i,c}^1 = \frac{1}{N(k-1)}$ **for** $i = 1, ..., N, c \in \{1, ..., k\} \backslash \{y_i\}$.
for $t = 1, 2, ..., T$

1. Set $W_i^t = \sum_{c \neq y_i} w_{i,c}^t$;
   Define the label weight function

$$q_t(i, c) = \frac{w_{i,c}^t}{W_i^t}$$

   for $c \neq y_i$ and the distribution

$$D_t(i) = \frac{W_i^t}{\sum_{i=1}^{N} W_i^t}.$$

2. Call WeakLearn provided with the distribution D_t and label weight function q_t;
   get back a hypothesis: $h_t \colon X \times \{1, ..., k\} \to [0, 1]$.
3. Calculate pseudo-loss of h_t:

$$\varepsilon_t = \frac{1}{2} \sum_{i=1}^{N} D_t(i) \left(1 - h_t(X_i, y_i) + \sum_{c \neq y_i} q_t(i, c) h_t(X_i, c) \right).$$

4. Set $\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$.
5. Set the new weights vector to

$$w_{i,c}^{t+1} = w_{i,c}^t \beta_t^{1/2(1 + h_t(X_i, y_i) - h_t(X_i, c))}$$

   for $i = 1, ..., N, c \in \{1, ..., k\} \backslash \{y_i\}$.

end
return

$$h(\tilde{x}) = \arg \max_{c \in \{1, ..., k\}} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) h_t(\tilde{x}, c);$$

as it examines the weighted problem. Typically it has low values in the first iterations and rises afterwards. Once the weights have been updated, the next iteration can begin. After all weak learners are included, i.e. after T iterations, the final hypothesis function h is returned, which represents the trained model. It calculates for a given input $\tilde{x}$ for each class $c \in \{1, ..., k\}$ the weighted mean of the hypothesis values of the weak learners and then chooses the class with the highest one.

Next, LDA will be introduced. It is based on the multivariate normal distribution, whose probability density function is defined for a dimension of d as

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right),$$

with $\mu = \mathbb{E}(x) \in \mathbb{R}^D$ denoting the mean vector and Σ the $D \times D$ covariance matrix of x. The variable D defines the number of features again, $|\Sigma|$ refers to the determinant of the matrix Σ. The argument of the exponential function is also called Mahalanobis distance between the data vector x and its mean vector. In general, Gaussian discriminant analysis as an application of the distribution given above is a generative classifier that defines the class conditional densities as

$$p(x|y = c, \theta) = \mathcal{N}(x|\mu_c, \Sigma_c). \tag{5.2}$$

They are used for the calculation of the probability that for a given feature vector x the class $c \in \{1, ..., k\}$ is predicted. The variable θ comprises of the parameters of the model, k defines the number of classes again. According to Bayes' rule the conditional probability that we would like to determine is

$$p(y = c|x, \theta) = \frac{p(y = c|\theta)p(x|y = c, \theta)}{\sum_{c'=1}^{k} p(y = c'|\theta)p(x|y = c', \theta)}. \tag{5.3}$$

The prior probabilities $p(y = c|\theta)$ are computed as the sampling probability of the training dataset of N samples of each class analogously to equation (5.1) and denoted as $\hat{\pi}_c$.

$$\hat{\pi}_c = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(y_i = c) \tag{5.4}$$

Plugging this, as well as the probability density function (5.2), in, we get

$$p(y = c|x, \theta) = \frac{\hat{\pi}_c(2\pi)^{-\frac{d}{2}}|\Sigma_c|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu_c)^T\Sigma_c^{-1}(x-\mu_c)\right)}{\sum_{c'=1}^{k} \hat{\pi}_{c'}(2\pi)^{-\frac{d}{2}}|\Sigma_{c'}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu_{c'})^T\Sigma_{c'}^{-1}(x-\mu_{c'})\right)}.$$

This formula gives a quadratic function of x and the algorithm is therefore called quadratic discriminant analysis. Assuming that the covariance matrices are shared or tied, i.e. all of

them are equal $\Sigma_c = \Sigma$, we get LDA.

$$\begin{aligned} p(y=c|x,\theta) &= \frac{\hat{\pi}_c(2\pi)^{-\frac{d}{2}}|\Sigma|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(x-\mu_c)^T\Sigma^{-1}(x-\mu_c)\right)}{\sum_{c'=1}^{k}\hat{\pi}_{c'}(2\pi)^{-\frac{d}{2}}|\Sigma|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(x-\mu_{c'})^T\Sigma^{-1}(x-\mu_{c'})\right)} \\ &= \frac{\hat{\pi}_c\exp\left(-\frac{1}{2}x^T\Sigma^{-1}x+\mu_c^T\Sigma^{-1}x-\frac{1}{2}\mu_c\Sigma^{-1}\mu_c\right)}{\sum_{c'=1}^{k}\hat{\pi}_{c'}\exp\left(-\frac{1}{2}x^T\Sigma^{-1}x+\mu_{c'}^T\Sigma^{-1}x-\frac{1}{2}\mu_{c'}\Sigma^{-1}\mu_{c'}\right)} \\ &= \frac{\exp\left(\mu_c^T\Sigma^{-1}x-\frac{1}{2}\mu_c\Sigma^{-1}\mu_c+\ln(\hat{\pi}_c)\right)}{\sum_{c'=1}^{k}\exp\left(\mu_{c'}^T\Sigma^{-1}x-\frac{1}{2}\mu_{c'}\Sigma^{-1}\mu_{c'}+\ln(\hat{\pi}_{c'})\right)} \end{aligned}$$

Defining $\gamma_c := -\frac{1}{2}\mu_c^T\Sigma^{-1}\mu_c + \ln(\hat{\pi}_c)$ and $\beta_c = \Sigma^{-1}\mu_c$ we can shorten the definition to

$$p(y=c|x,\theta) = \frac{\exp\left(\beta_c^T x+\gamma_c\right)}{\sum_{c'=1}^{k}\exp\left(\beta_{c'}^T x+\gamma_{c'}\right)}.$$

It is called linear, because, since the term $\exp\left(-\frac{1}{2}x^T\Sigma^{-1}x\right)$ is truncated from the numerator and denominator, a linear function remains, if the logarithm of the definition is taken. Linear models generally have the property to be interpretable. When all appearing parameters for the posterior probabilities $p(y=c|x,\theta)$ are determined, the model is trained and can test new data given by the feature vector x by calculating the posterior probabilities of each class and choose the class with the highest value.

Naive Bayes is very similar to LDA and if Σ_c in equation (5.2) is a diagonal matrix they are even equivalent. The machine learning algorithm takes Bayes' rule of equation (5.3) as well, but assumes that the features are conditionally independent. This results in a change of the calculation of the class conditional densities, which were defined for the LDA, as given in equation (5.2), to

$$p(x|y=c,\theta) = \prod_{j=1}^{D} p(x_j|y=c,\theta_{jc}).$$

The parameters comprised by θ_{jc} are the mean μ_{jc} and the variance σ_{jc} of a certain feature j of items a class c, the variable D defines the number of the features. As in our case all of them are real values for which we can set $p(x|y=c,\theta) = \prod_{j=1}^{D}\mathcal{N}(x_j|\mu_{jc},\sigma_{jc})$ and compute the probabilities using the normal distribution. The prior probabilities $p(y=c|\theta) =: \hat{\pi}_c$ are again calculated analogously to equation (5.4) as the probability of how often class c occurs in the training dataset. After the training process, a test can be run like before by determining all the posterior probabilities $p(y=c|x,\theta)$ for a given x and predicting the class with the highest one.

The assumption of the conditional independence is strict and it is very unlikely that this is the case in reality. That is why the method is called naive. Nevertheless, models based

on this algorithm do not have a high risk of overfitting and are quite interpretable because of their plain structure, while still performing well to some extent.

Both LDA and naive Bayes are generative classifiers, as they start looking at a joint model of the form $p(y, x)$ and then deriving $p(y|x)$. Another technique is given by logistic regression for example. Even though the name is misleading, it is a discriminative classifier. A discriminative approach tries to fit the model $p(y|x)$ directly. At the end of the section we will discuss the differences as well as the advantages and disadvantages between the two creation processes. The logistic regression model for multiple classes takes the form

$$p(y = c|x, W) = \frac{\exp(W_c^T x)}{\sum_{c'=1}^{k} \exp(W_{c'}^T x)}.$$

The weight matrix $W \in \mathbb{R}^{D\times k}$ stores the coefficients of the classes $c \in \{1, ..., k\}$ for D features. The vector W_c represents the c-th column of the matrix W. To get a procedure for determining the weights, we need some definitions and corresponding notation. Let $\mu_{ic} = p(y_i = c|X_i, W)$ for the i-th out of N training samples and summarised to the vector $\mu_i = (\mu_{i1}, ..., \mu_{ik})^T$. For $y_{ic} = \mathbb{I}(y_i = c)$ the vector y_i comprises of $(y_{i1}, ..., y_{ik})^T$. As it is suggested in [38], for reasons of identifiability, the last column of W is set to a zero vector, i.e. $W_k = 0$. Afterwards, we set the vector w as a column vector of length $D \cdot (k-1)$ with the values of the matrix W of the columns 1 to $k-1$ stacked, i.e. $w = (W_1; ...; W_{k-1})$. Hence, we also shorten the vectors μ_i and y_i by 1 to a length of $k-1$. Logistic regression uses a negative log-likelihood function as basis, consequently we have for this reason

$$\begin{aligned} f(w) &= -\log\left(\prod_{i=1}^{N}\prod_{c=1}^{k} \mu_{ic}^{y_{ic}}\right) = -\sum_{i=1}^{N}\sum_{c=1}^{k} y_{ic}\log(\mu_{ic}) \\ &= -\sum_{i=1}^{N}\left[\left(\sum_{c=1}^{k-1} y_{ic}W_c^T X_i\right) - \log\left(\sum_{c'=1}^{k-1}\exp(W_{c'}^T X_i)\right)\right]. \end{aligned}$$

We will need the gradient g and the Hessian matrix H of this function. For this, we use the kronecker product of matrices $\otimes$. The gradient calculates to

$$g(W) = \nabla f(w) = \sum_{i=1}^{N}(\mu_i - y_i) \otimes X_i$$

which results in a column vector of length $D \cdot (k-1)$. The Hessian is a $D(k-1) \times D(k-1)$ matrix given by

$$H(W) = \nabla^2 f(w) = \sum_{i=1}^{N}(\mathrm{diag}(\mu_i) - \mu_i\mu_i^T) \otimes (X_i X_i^T).$$

The operator diag gives back a $(k-1) \times (k-1)$ diagonal matrix with the values of its argument on the main diagonal. We then consider the perturbed problem with the objective function $f'(W) = -\log p(\mathcal{D}|w) - \log p(W)$ with $p(W) = \prod_{c=1}^{k}\mathcal{N}(W_c|0, V_0)$ using ℓ^2

regularisation which we want to minimise. The regularisation strength is chosen by MATLAB automatically. The normal distribution is taken of a mean vector of 0 and a variance matrix V_0. The function, the gradient and the Hessian matrix then read as

$$f'(W) = f(W) + \frac{1}{2}\sum_{c=1}^{k} W_c V_0^{-1} W_c,$$
$$g'(W) = g(W) + V_0^{-1}\left(\sum_{c=1}^{k} W_c\right),$$
$$H'(W) = H(W) + I \otimes V_0^{-1}.$$

The matrix I denotes the unit matrix. These definitions are needed for the algorithm that estimates our parameters of the logistic regression model. The basic idea is to use Newton's algorithm

$$\theta_{k+1} = \theta_k - \eta_k H_k^{-1} g_k$$

with a learning rate η_k and a parameter set θ_k, which includes the weight matrix W to find the optimal fix point of the parameters. An iteration limit of 1000 is set. For determining the Hessian, the L-BFGS algorithm is used, which stands for limited memory Broyden, Fletcher, Goldfarb and Shanno. The four last names belong to the people who created the procedure. It is a quasi Newton method that approximates the Hessian matrix H_k or its inverse using only a diagonal and a low rank matrix.

As we use kernel logistic regression, we first do a random feature expansion using the Fastfood scheme, see [42] for further information. It maps the input $x \in \mathbb{R}^D$ to a higher dimensional space. The parameters for this procedure are set automatically by MATLAB. In the high dimensional space the logistic regression model is applied, which minimises the objective function f'. Even though the overall model is called kernel logistic regression, no Gaussian kernel, a method that is not introduced in this work, is applied to the model. The presented procedure defines an equivalent and this is the way how it is implemented in MATLAB.

The model that is introduced next is KNN. It is a plain approach that tests new data by looking at a certain number of neighbours K of the points used to train the model, which we summarised as $\mathcal{D}$. We set $K = 10$ in our model. For the decision of the nearest neighbours, a distance metric has to be chosen, in our case it is the Euclidean distance

$$d(\tilde{x}, X_i) = \left(\sum_{j=1}^{D} (\tilde{x}_j - x_{ij})^2\right)^{\frac{1}{2}}.$$

The distance is calculated for feature vector that is tested $\tilde{x}$ and each of the N training sample vectors X_i. Each of them is of length D which is the number of predictors. This defines a neighbourhood $N_K(\tilde{x}, \mathcal{D})$ of size K around $\tilde{x}$. The probability for a certain class c is then given by

$$p(y = c|\tilde{x}, \mathcal{D}, K) = \frac{1}{K}\sum_{i \in N_K(\tilde{x}, \mathcal{D})} \mathbb{I}(y_i = c). \tag{5.5}$$

The model predicts the class with the highest probability. We use a modification of this algorithm, which is called weighted KNN. The distance measure affects not only the choice of neighbourhood, but also the strength of that neighbour's "vote". The distance weight in our setting is set to squared inverse, which means that equation (5.5) changes to

$$p(y = c|\tilde{x}, \mathcal{D}, K) = \sum_{i \in N_K(\tilde{x}, \mathcal{D})} \frac{1}{d(\tilde{x}, X_i)^2} \mathbb{I}(y_i = c).$$

The factor $\frac{1}{K}$ was neglected as well, since we cannot speak about probabilities anymore because they do not sum up to 1. This procedure changes the model in a way that the more distant a neighbour is the less of an influence it has on the prediction. As the distance measure is of great importance for the original procedure and even more for the modification, the scales of the features are highly important. Predictors, whose values do not have a wide range, contribute less to the result. Consequently, in literature it is found that standardisation of the features is recommended. MATLAB does this for each numeric feature by centring by the mean and scaling by the standard deviation, respectively.

Properties of KNN models are very easy to explain to other people and they show good performances. However, when using a high number of features these characteristics diminish, as interpretation for this approach can only happen on a local level, since only neighbours are being looked at. This gets complex for many predictors. The classification accuracy also reduces with an increasing number of feature vectors, even though for weighted KNN this is not as bad as for the original algorithm.

The last machine learning approach, that will be introduced, are linear SVMs. They are only used for binary classification, but extensions can also make classification of multiple classes possible. As we have $k = 5$ different classes, a one-versus-one approach is used that trains $\frac{k(k-1)}{2}$ linear SVMs $f_{c,c'}$, that can distinguish between two classes c, c'. The prediction is then made according to the most "votes" of the classifiers.

Binary classification with linear SVMs is a similar approach to logistic regression, but changes the underlying function from negative log-likelihood to the hinge loss function $\max(0, 1 - yf(x))$. The class labels are denoted as $y \in \{-1, 1\}$, as only two available classes remain. The input is the feature vector $x \in \mathbb{R}^D$. The confidence in choosing label 1 is given by the function $f(x) = w^T x + b$ with a weight vector $w \in \mathbb{R}^D$ and a bias $b \in \mathbb{R}$. The objective function for determining the weights is

$$\min_{w,b} \left(\frac{1}{2} ||w||^2 + C \sum_{i=1}^{N} \max\left(0, 1 - y_i f(x_i)\right) \right).$$

The constant C is determined throughout the training process and specifies the number of tolerable errors during the training. The solution is given by

$$\hat{w} = \sum_{i=1}^{N} \alpha_i X_i$$

with a sparse α vector. Many of the components are 0 because the hinge loss function often obtains the value 0. When for $i \in \{1, ..., N\}$ the corresponding α_i is greater than 0, X_i is called a support vector. These are the ones that are not classified correctly. After

the training process, the prediction for a certain feature vector x of the binary classifier is given by

$$\hat{y}(x) = \text{sgn}(f(x)) = \text{sgn}\left(\hat{w}^T x + \hat{b}\right)$$

with sgn $\in \{-1, 1\}$ denoting the signum function. Our MATLAB procedure standardises the feature data as well, analogously like for KNN.

This procedure is, in the given linear case, easy to interpret, however, especially in the case of multiple classes the training time of the model is very high, as it is presented in section 6.2.

After introducing all the various examined models, a short comparison between discriminative versus generative approaches and corresponding properties is given. An overview about which of the presented models belongs to which category is given in table 5.4. Generative models are generally easier and faster to fit. They are able to fit each class separately, which can be important when adding a new one, and they can handle unlabelled training data as well as missing features better. The last characteristics are not very important for our application. On the contrary, due to sometimes very strong assumptions in generative models, like for naive Bayes, discriminative ones are often better calibrated regarding their probability estimates, i.e. probabilities of 0 or 1 only happen rarely. Another advantage for discriminative models is that they can preprocess features, for example using expansion.

Table 5.4.: Breakdown which of the examined models is generative and which is discriminative.

Model	Generative or discriminative
LDA	Generative
Naive Bayes classifier	Generative
Weighted KNN	Generative
Kernel logistic regression	Discriminative
Decision tree	Discriminative
Bagged trees	Discriminative
Boosted trees with AdaBoost	Discriminative
Linear SVM	Discriminative

As all the models have now been introduced, we can apply the algorithms on the CAP Sleep Database and compare the results to gain a classification model for our sleep scoring problem.

6. Results and Benchmarking

The CAP Sleep Database is used for training and testing the different models. As explained in section 5.1, we use 104 datasets. To get a comparison, we split the data into five folds and always train the algorithms with four folds and then test with the fifth one. The given accuracies and results in section 6.1 always represent the mean of these five test runs. For the first application of our models, we use the 22 features, as explained in chapter 4. Van der Donckt et al. suggest in [69] to use a higher number of predictors, i.e. not lower than 40, as a low number of features limits the research contribution. Accordingly, we expanded our feature set, which is presented in section 6.2. Afterwards, the best performing, trained models are used for benchmarking the Sleep Telemetry Database of the Sleep EDF Database Expanded [35]. In section 6.3 the corresponding results are given.

6.1. Performance of Different Models for 22 Features

For finding a well performing sleep scoring model, we investigate different machine learning algorithms, which are described in section 5.3. The eight models are a decision tree, LDA, a linear SVM, weighted KNN, bagged trees, boosted trees with AdaBoost, kernel logistic regression and naive Bayes classifier. Before we can train the different models, we first have to determine the five folds we split the database into. The algorithms are then always trained with four folds and tested with the fifth one, for comparison, the mean is taken from the five runs. As we want to build a model for sleep stage classification, independent of the health status regarding sleep disorders, we split the data base in a way such that each sleep disorder is equally distributed among all folds. This way, each fold is as heterogeneous as possible. We also distributed the seven data sets as widely as possible, where we faced the problem of having to choose different EEG signals. With these assumptions the folds as presented in appendix B are determined. The datasets, for which we had to take different channels than usual, are underlined.

After the extraction of all the features for the 104 patients, we created two tables according to the folds with training data and test data. They contain the 22 features for each 30s window of each patient, where 21 of them change for almost every window, the age only changes with the patient. The tables also contain the sleep stages of the 30s windows.

Firstly, the training data is handed over to the Classification Learner application of MATLAB. We have to choose which column of the table is the response variable, in our case this is the column of the sleep stages, and which ones are the predictors, which are the 22 other columns. One can also select to not work with all of them but only choose some. We use all of the 22 features. There is also the option to set aside a test dataset of the imported table. This is not done here, since an additional table is already set aside with the test data. The application also gives an option for validation, which protects against overfitting, see [50]. We use cross-validation with five folds, meaning that we split

the training data in five folds again, train them with four and test them against the fifth one, like it is later done with the test data. Thereafter, the average validation accuracy is the mean of the five runs. However, these folds are chosen randomly, therefore we have a high probability of training the model with data of a patient and to then do the validation test with different data from the same patient. This makes the model biased towards that patient, which is something we want to avoid in the overall model. Consequently, we still perform our tests later with the test dataset. But still, cross-validation gives a rough estimate of the accuracy of the model. Even though it is not recommended for large datasets, due to computational effort, we choose it for our model building process, as we want better estimates for the accuracy. Another variant would be the holdout validation, which only tests one validation fold against the other training folds. This approach is faster for larger datasets. The validation procedure is of great importance, if no test set is available, but we create our own set by splitting the database. After a validation process, the final model is trained with the whole table of the training data.

Secondly, the different models are chosen with the settings as described in section 5.3 and then trained. The validation accuracies are listed in table 6.1. Some models were trained quickly, some of them needed more time, for example the linear SVM or bagged trees. A short comparison of the exact times is given in section 6.2. After having the model trained with four folds, it is tested against the fifth fold. The test accuracy results are also presented in table 6.1.

Table 6.1.: Test and validation accuracies of the eight examined models given in percent. In this case the feature extraction comprised of 22 predictors.

Model	Validation accuracy	Test accuracy
LDA	61.9%	61.5%
Naive Bayes classifier	44.5%	44.1%
Weighted KNN	69.9%	62.5%
Kernel logistic regression	53.1%	49.4%
Decision tree	65.1%	60.4%
Bagged trees	73.6%	62.8%
Boosted trees with AdaBoost	63.6%	60.3%
Linear SVM	64.8%	63.7%

As we can see, the models perform very differently. The validation and test accuracies for most models are over 3%, for LDA, naive Bayes classifier and linear SVM they are lower. This strengthens our hypothesis to split the CAP Sleep Database according to the patients and sleep disorders to gain a sleep scoring model for all humans, independent of their health status. It seems that in the cases of a high validation accuracy, if the model is trained with data of a certain patient, and then tests with different data of the same person, it classifies better, than for a complete different subject. Therefore, we only compare the models based on their test accuracies. Linear SVM, bagged trees, weighted KNN and LDA are the four best in descending order. We want to investigate these further by looking at the confusion matrices of the models.

A confusion matrix lists the observations of predicted and true classes in a table, see [50, 39]. True classes are written in the rows and the predicted classes in the columns. This way observations of truly predicted classes can be found on the main diagonal. In the off-diagonals, the misclassified stages are written, divided into which class was incorrectly predicted. With the confusion matrix, one can directly compute some important rates. On the one hand, the true positive rate (TPR) of each class is the number of true predictions divided by the corresponding rowsum of the matrix. This value is also called sensitivity. The false negative rate is 1 - TPR for each class, also known as false alarm rate. On the other hand, the positive predictive values (PPV) is the number of true predictions divided by the corresponding columnsum of the matrix. The false discovery rate is therefore 1 - PPV. To obtain the confusion matrices of the four models, we take the correctly and incorrectly classified observations of each test run and sum them up. The result for each model is presented in figure 6.1, the plots are done with the MATLAB function `confusionchart`. Blue coloured cells show correctly predicted sleep stages, red coloured incorrectly ones. The darker the colour, the more correct or false observations were done.

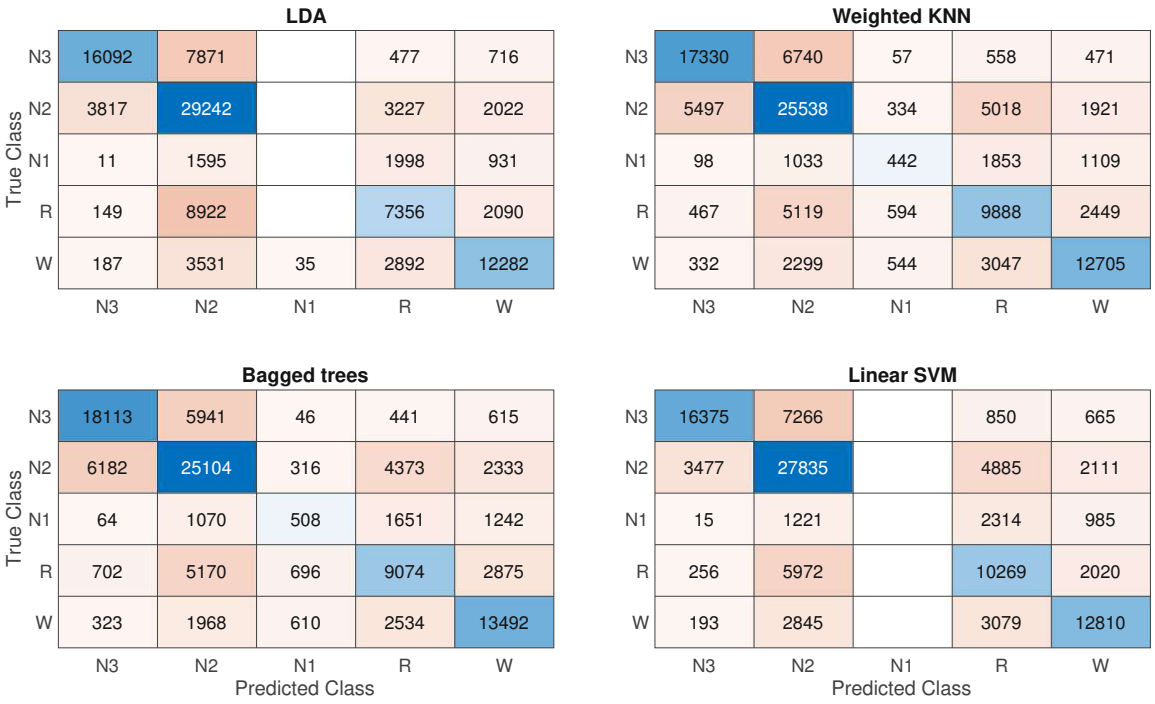

Figure 6.1.: Confusion matrices of the different models trained with four folds and tested against the fifth one with 22 features. Blue coloured cells show correctly predicted classes, red coloured misclassified ones. White coloured cells represent no observation.

Taking the rowsum over each of the matrices we can see that for the 104 examined datasets of the CAP Sleep Database, there are 105443 windows of a length of 30s for which we know the sleep stage annotations. In total, the stage N3 appeared 25156 times for all patients, N2 38308 times, N1 4535 times, R 18517 times and W 18927 times. What all four models

have in common is that many misclassifications happen between the stages N3 and N2 and N2 and R. Incorrect predictions between R and W also occur quite often. Linear SVM and LDA rarely or not at all attempt to predict the class N1.

We do not only want to compare the overall accuracy of the models but to also calculate Cohen's κ coefficient, see [14], which was introduced in section 3.2. According to [66], the calculation can be modified using the confusion matrix. For the adaption to fit our case we define the number of correctly predicted sleep stages, which is the sum of the main diagonal, i.e. the trace, as $T = \sum_{i=1}^{5} M_{ii}$. The matrix M denotes the confusion matrix. The rowsum of M is a vector of length 5 that contains the true number of sleep stages, which will be denoted by SC. The columnsum analogously comprises of the predicted number of sleep stages, written as SP. As defined in equation (3.1) the adaption to our problem reads as

$$\kappa = \frac{p_0 - p_c}{1 - p_c} = \frac{\frac{T}{N} - \sum_{c=1}^{5} \frac{SC_c}{N}\frac{SP_c}{N}}{1 - \sum_{c=1}^{5} \frac{SC_c}{N}\frac{SP_c}{N}}$$

with $N = 105443$ being the number of 30s windows. Cohen's κ coefficient has the advantage that it also considers the number of predictions that happen by chance, which is represented by the term $\sum_{c=1}^{5} \frac{SC_c}{N}\frac{SP_c}{N}$. As with the overall accuracy, a higher value indicates better accuracy. For comparison, the overall accuracy OA is defined as

$$OA = \frac{T}{N} \cdot 100\%.$$

The κ values are listed in table 6.2. It is visible that the linear SVM continues to be the best model for this value.

Table 6.2.: Cohen's κ values for the four better performing models that were trained with 22 features.

Model	Cohen's κ coefficient
LDA	0.4674
Weighted KNN	0.4929
Bagged trees	0.4991
Linear SVM	0.5052

To optimise our models and to achieve a higher accuracy and κ value, we take more features into account and hope that we can get a better classification of stage N1 as it has never been predicted by the linear SVM. This is done in section 6.2.

6.2. Model Improvement using 64 Features

It is written in [69] that if the number of features is below 40, the research contribution is limited. As we want to expand our number of predictors to optimise the performance of our models, as described in section 6.1, the number of features is almost tripled. Out of the 22 predictors, that had been used before, the age is the only one that cannot be used a second time. Except for the GC, that makes up two of the features, all of the others had been simply calculated of the frontal channel, up until this moment. These 19 features are now also computed of the central and of the parietal signals. Before, the GC had been calculated regarding the influence of the central channel C on the frontal F $\mathcal{F}_{C\to F}$ and the parietal P on F $\mathcal{F}_{P\to F}$. Now, we also compute every other possible combination of the GC, i.e. $\mathcal{F}_{F\to C}$, $\mathcal{F}_{F\to P}$, $\mathcal{F}_{C\to P}$ and $\mathcal{F}_{P\to C}$, which gives us four more predictors. This way, we were able to remain in the time-domain, resulting in a total of $3 \cdot 19 + 6 + 1 = 64$ features.

By first extracting the new amount of features, we followed the same procedure outlined in the previous section and examined the accuracy of the tests to compare the eight models. For this, we also used the same test folds, as described in appendix B, to compare the different approaches regarding the number of predictors. The validation folds for cross-validation are randomly chosen by the Classification Learner application. The test and validation accuracies for each of the algorithms are listed in table 6.3.

Table 6.3.: Test and validation accuracies of the eight examined models given in percent. In this case the feature extraction comprised of 64 predictors.

Model	Validation accuracy	Test accuracy
LDA	65.2%	63.7%
Naive Bayes classifier	46%	45.4%
Weighted KNN	76.4%	65.8%
Kernel logistic regression	58.9%	54.5%
Decision tree	67.2%	62.1%
Bagged trees	77.7%	67.2%
Boosted trees with AdaBoost	66.6%	63.1%
Linear SVM	68.4%	66.1%

The four best models regarding the test accuracy are LDA, weighted KNN, bagged trees and linear SVM, as discovered before, but the order has changed. Bagged trees are now the best classifier. When comparing to the results achieved with 22 features, we can see that the test accuracy increases for all the models up until a raise of five percentage points. The ratio between validation and test accuracy stays approximately the same. To have a deeper insight into the performance, we want to take a closer look at the confusion matrices of the four best models and later on determine the Cohen's κ value as well. The confusion matrices are pictured in figure 6.2, the plots are again created using the MATLAB function `confusionchart`.

As we look at the same data, the number of 30s windows stays the same with 105443. We achieve similar results as before, but the number of correct predictions is higher for each model and class, as expected. The feature expansion led to the desired hope for the LDA

Figure 6.2.: Confusion matrices of the different models trained with four folds and tested against the fifth one with 64 features. Blue coloured cells show correctly predicted classes, red coloured misclassified ones. White coloured cells represent no observation.

that the model also tries to predict the sleep stage N1. This is not the case for the linear SVM. However, the increase in correct classifications also resulted in a rise in the overall accuracy of the algorithm. Apart from that, we still see a high number of misclassifications between the stages N3 and N2 and N2 and R. When comparing the Cohen's κ coefficient, listed in table 6.4, we see an increase in all four models. Again it holds, that the higher the overall accuracy the higher the κ value.

Table 6.4.: Cohen's κ values for the four better performing models that were trained with 64 features.

Model	Cohen's κ coefficient
LDA	0.5007
Weighted KNN	0.539
Bagged trees	0.5595
Linear SVM	0.5399

When the two approaches with 22 and 64 features are compared, it is obvious that for each model the higher number of predictors performs better in terms of accuracy. Although the training time increases with more features, the primary aim is to develop a model that predicts with maximum accuracy. In order to achieve the desired results, the longer period

of training can be considered to be of little importance, since properties like interpretability, accuracy and prediction speed matter more. The latter one is an estimate given by MATLAB of how fast new data is tested on the basis of the prediction speed during validation. However, training time is still included in the comparison of the final models that are trained with the whole dataset containing the 104 subjects.

After all of the investigations, we decided to train all four of the finalists, i.e. bagged trees, linear SVM, weighted KNN and LDA. A comparison between them is given in table 6.5.

Table 6.5.: Comparison of linear SVM, bagged trees, weighted KNN and LDA trained with 104 datasets of the CAP Sleep Database and 64 features extracted. The properties are the accuracies of the five fold test of table 6.3, of a five fold cross-validation, if it is easy to interpret, the training time and the prediction speed.

	LDA	Weighted KNN	Bagged trees	Linear SVM
Five fold test	63.7%	65.8%	67.2%	66.1%
Five fold cross-validation	64.9%	76.3%	77.5%	68.2%
Easy to interpret	Yes	No	No	Yes
Training time	6.85s	429.25s	861.68s	2308.9s
Prediction speed	$\sim$ 190000 obs/s	$\sim$ 1100 obs/s	$\sim$ 39000 obs/s	$\sim$ 110000 obs/s

As we can see, the accuracies for the five fold cross-validation of the entire database used are similar to the mean of cross-validation when trained with the specified four folds which is listed in table 6.3. In terms of interpretability, we already discussed some points in section 5.3, linear models like linear SVM and LDA are easier to interpret than the other ones. The training time between the models differs a lot. Especially the linear SVM needed much more time than the rest with almost 39 minutes. This long training time has already been hinted in section 5.3, as SVMs cannot create a model for classification of multiple classes directly. In our case, the linear SVM model is comprised of $\frac{5 \cdot 4}{2} = 10$ linear SVMs, which always only classify a binary case. Bagged trees follow with a training time of 15 minutes and weighted KNN needed approximately half of that time. LDA is trained very quickly, needing only a few seconds. The five fold cross-validation also increases training time, holdout validation would speed up this process. In terms of prediction speed, LDA and linear SVM are outstanding, whereas weighted KNN is much slower. The unit of this property is given in observations per second.

6.3. Benchmarking Sleep-EDF Sleep Telemetry Database

The four models obtained from the previous section, i.e. LDA, weighted KNN, bagged trees and linear SVM trained with the whole database of 104 subjects, are now tested to benchmark a new set of data. The Sleep-EDF Database Expanded [35] is freely available and was downloaded from PhysioNet [25]. It comprises of 197 polysomnography recordings split into two separate databases. The Sleep Casette is a set of 153 recordings of 78 subjects between the ages of 25 to 101, who did not take any sleep-related medication, recorded between 1987 and 1991. The Sleep Telemetry Database contains 44 recordings of 22 patients with minor problems with falling asleep who were between the ages of 18 and 79. The subjects were recorded on two consecutive nights in 1994, one night being given the sleep medication temazepam and the other night a placebo. With Fpz-Cz and Pz-Oz only two EEG signals were recorded with a sampling rate of 100Hz. The sleep stages are annotated based on the R&K rules, and therefore we had to manually merge the sleep stages S3 and S4 to our needs according to the AASM rules again.

The second database, i.e. the Sleep Telemetry Database, is used for testing our models. As only two EEG channels are included, we choose Fpz-Cz as the frontal and Pz-Oz as the central and parietal one analogously to the subjects n6, n7 and n9 of the CAP Sleep Database, as described in section 5.2. The signals go through the same preprocessing, as needed, by resampling to 200Hz, lowpass filtering with 30Hz and highpass filtering with 0.5Hz. Afterwards, the 64 features are extracted and the sleep stages are predicted. In figure 6.3, the confusion matrices of the four models are pictured, which show the true and predicted classes of the summarised 44 datasets. The plots are created using the MATLAB function `confusionchart`.

With the confusion matrices, the overall accuracy and the Cohen's κ coefficients are determined. The results are listed in table 6.6 and show slightly worse outcomes than for the CAP Sleep Database. However, the best model LDA reaches an accuracy of 64% and $\kappa = 0.4842$, which is a relatively good achievement because of the limited number of EEG channels and a sampling rate of only 100Hz.

Table 6.6.: Overall accuracy and Cohen's κ coefficients of the four models trained with 64 features for the Sleep Telemetry Database.

Model	Overall accuracy	Cohen's κ coefficient
LDA	64%	0.4842
Weighted KNN	60.7%	0.447
Bagged trees	54.9%	0.3711
Linear SVM	59.1%	0.4261

In early 2023, Van der Donckt et al. gave an overview of benchmarking results of different classification approaches in their publication [69]. With their newly introduced approach, they achieved the best results for this database, with an overall accuracy of 83.6% and $\kappa = 0.765$. It is a gradient boosted trees model with Catboost using 1048 features per 30s window. They used the two given EEG channels, one EOG and one EMG channel for their feature extraction.

Figure 6.3.: Confusion matrices of the different models trained with 64 features for benchmarking the Sleep Telemetry Database of the Sleep-EDF Database Expanded. Blue coloured cells show correctly predicted classes, red coloured misclassified ones. White coloured cells represent no observation.

Now that all the results of our models have been presented, it can be concluded that more complex, difficult to interpret models do not automatically perform better in sleep scoring. Even though bagged trees with 64 features were the best classifier for the CAP Sleep Database, linear SVM followed with only one percentage point less overall accuracy. When the models were applied to the Sleep Telemetry Database, LDA achieved the best performance. This shows that linear sleep scoring models based on EEG analysis can compete with more complex models in terms of accuracy.

7. Discussion

Sleep stage classification is of great importance in the diagnosis of sleep disorders. As every human being needs the right amount of sleep to stay healthy, it has a big impact on the overall health care system. However, the procedure of sleep scoring done manually by a specialist can take some time and, consequently, costs the society money. To speed up this process, semi-automated and automated approaches have been introduced more than 20 years ago. These are used in research and achieve very good results and perform very well, but none of them has yet manifested in widespread use in the clinical setting. Hence, it is important to continue the research.

Our approach to this topic is to find an interpretable machine learning model that is based on features in the time-domain and examines three EEG signals. For this purpose, we present different entropy-based parameters, i.e. PE, EoD, KLD and CE, in every mathematical detail. Especially the first two show a high correlation with the course of the sleep stages and thus are highly promising predictors for sleep scoring. Apart from that the GC is explained in detail as well, as this is a measure of influence between the channels. Statistical parameters and the age of the examined subject as personal information complete our list of features.

With the parameters eight machine learning algorithms are trained which are LDA, naive Bayes classifier, weighted KNN, kernel logistic regression, a decision tree, bagged trees, boosted trees with AdaBoost and a linear SVM. For training and testing, 104 recordings of the CAP Sleep Database [67] are used. The preprocessing of the data is described and the mathematical basis for the models is introduced. We split the patients into five folds and achieve our results by taking the mean of five runs, in which the algorithms are trained with four folds and are then tested against the fifth one. The models were tested with 22 features, mainly applied to one EEG signal, and also with 64 predictors, by applying them to the other two given channels. In both cases these tests show that with LDA and linear SVM, two models that are easy to interpret, can compete with the more complex models, that also performed well. The accuracy was evaluated using the overall accuracy and the Cohen's κ coefficient. For the better performing case of 64 features, bagged trees achieved the highest accuracy with 67.2% and $\kappa = 0.5595$, followed by the linear SVM with 66.1% and $\kappa = 0.5399$, weighted KNN with 65.8% with $\kappa = 0.539$ and LDA with 63.7% and $\kappa = 0.5007$. Similar results were obtained with benchmarking the Sleep Telemetry Database of the Sleep-EDF Database Expanded, although only two EEG channels are available for these recordings, and therefore, the database does not fit the requirements for our models perfectly. Therefore, it can be concluded that plain linear models, that are based only on EEG signals and a personal parameter, achieve similar accuracy to more complex ones. This suggests further research with linear machine learning models for sleep scoring based

on time-domain EEG analysis, as the properties of explainability and interpretability are desired in the medical field.

Even though the results did not show the desired accuracy of at least 80% overall accuracy and $\kappa = 0.7$, which according to [18] are the values of overall agreement between two raters, we could show that black-box modelling does not always have to be the better performing option in machine learning. Linear models can compete with models that cannot be or are hard to interpret. They also outperform the more complex models in other properties, such as prediction speed. Our research demonstrated as well that a feature expansion can improve the performance in terms of accuracy.

Improvements to the models can also be done. From the machine learning perspective, some algorithms can be adapted regarding their hyperparameters, which is called model tuning. An example would be to change the number of neighbours K in a KNN model. Using feature selection, the number of features can also be optimised. Some of them may not contribute to a correct classification and should therefore be neglected anyway. This can result in more interpretable and less time-consuming models. Some algorithms perform even worse, if many non-informative features are present, see [39]. Decision trees and their modifications have the advantage that they are resistant against this deterioration. In the recent publication [69] the authors receive our desired results with simple models by taking two EEG, one EMG and one EOG channel into account. With them they extract 131 features per window and by looking at eight different sized time windows at the same time they have 1048 predictors for the classification of one 30s window. They also include features calculated in the frequency-domain, which can be directly applied to our models without taking other measurements or channels into account. Adding different biosignals or more channels would always result in more wiring. However, this is something that researchers want to minimise as well because it has a negative effect on the sleep quality of the subject.

The results of this thesis provide a starting point for the development of EEG-based sleep scoring models using interpretable machine learning algorithms. This is an important step not only for the classification of sleep stages, but also for the level of consciousness in general, as this knowledge can eventually also be transferred to other medical fields, e.g. vigilance states during anaesthesia. As only time-domain features are used, a fast analysis seems promising, which would be an important property for real-time classification of level of consciousness.

The PE has already been used as a measure of anaesthesia for the EEG in [52]. The authors concluded that PE was a promising predictor but that further research was needed. In this work it has been shown that the use of multiple parameters to measure vigilance states during unconsciousness can produce a good classification model. By calculating the features of several channels instead of one, an increased accuracy is achieved. This means that the PE as well as the other entropy-based features contribute to a better classification.

Bibliography

[1] M. Akin, M. B. Kurt, N. Sezgin, and M. Bayram. Estimating vigilance level by using EEG and EMG signals. *Neural Computing and Applications*, 17:227–236, 2008.

[2] P. Anderer, G. Gruber, S. Parapatics, M. Woertz, T. Miazhynskaia, G. Klösch, B. Saletu, J. Zeitlhofer, M. J. Barbanoj, H. Danker-Hopfe, et al. An E-health solution for automatic sleep classification according to Rechtschaffen and Kales: validation study of the Somnolyzer 24× 7 utilizing the Siesta database. *Neuropsychobiology*, 51(3):115–133, 2005.

[3] P. Anderer, A. Moreau, M. Woertz, M. Ross, G. Gruber, S. Parapatics, E. Loretz, E. Heller, A. Schmidt, M. Boeck, et al. Computer-assisted sleep classification according to the standard of the American Academy of Sleep Medicine: validation study of the AASM version of the Somnolyzer 24× 7. *Neuropsychobiology*, 62(4):250–264, 2010.

[4] C. Bandt. A new kind of permutation entropy used to classify sleep stages from invisible EEG microstructure. *Entropy*, 19(5):197, 2017.

[5] C. Bandt and B. Pompe. Permutation entropy: a natural complexity measure for time series. *Physical review letters*, 88(17):174102, 2002.

[6] L. Barnett and A. K. Seth. Granger causality for state-space models. *Physical Review E*, 91(4):040101, 2015.

[7] A. B. Barrett, M. Murphy, M.-A. Bruno, Q. Noirhomme, M. Boly, S. Laureys, and A. K. Seth. Granger causality analysis of steady-state electroencephalographic signals during propofol-induced anaesthesia. *PloS one*, 7(1):e29072, 2012.

[8] S. Berger, A. Kravtsiv, G. Schneider, and D. Jordan. Teaching ordinal patterns to a computer: Efficient encoding algorithms based on the lehmer code. *Entropy*, 21(10):1023, 2019.

[9] S. Berger, G. Schneider, E. F. Kochs, and D. Jordan. Permutation entropy: too complex a measure for EEG time series? *Entropy*, 19(12):692, 2017.

[10] G. Brandmayr, M. Hartmann, F. Fürbass, G. Matz, M. Samwald, T. Kluge, and G. Dorffner. Relational local electroencephalography representations for sleep scoring. *Neural Networks*, 154:310–322, 2022.

[11] G. Buzsaki. *Rhythms of the Brain*. Oxford university press, 2006.

[12] C. Cajochen, R. Foy, D.-J. Dijk, et al. Frontal predominance of a relative increase in sleep delta and theta EEG activity after sleep loss in humans. *Sleep Res Online*, 2(3):65–69, 1999.

[13] A. A. Casciola, S. K. Carlucci, B. A. Kent, A. M. Punch, M. A. Muszynski, D. Zhou, A. Kazemi, M. S. Mirian, J. Valerio, M. J. McKeown, et al. A deep learning strategy for automatic sleep staging based on two-channel EEG headband data. *Sensors*, 21(10):3316, 2021.

[14] J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[15] M. A. Colombo, M. Napolitani, M. Boly, O. Gosseries, S. Casarotto, M. Rosanova, J.-F. Brichant, P. Boveroux, S. Rex, S. Laureys, et al. The spectral exponent of the resting EEG indexes the presence of consciousness during unresponsiveness induced by propofol, xenon, and ketamine. *NeuroImage*, 189:631–644, 2019.

[16] R. Cooper, J. W. Osselton, and J. C. Shaw. *Elektroenzephalographie : Technik und Methoden*. Fischer, Stuttgart, 1974.

[17] T. M. Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[18] H. Danker-hopfe, P. Anderer, J. Zeitlhofer, M. Boeck, H. Dorn, G. Gruber, E. Heller, E. Loretz, D. Moser, S. Parapatics, et al. Interrater reliability for sleep scoring according to the Rechtschaffen & Kales and the new AASM standard. *Journal of sleep research*, 18(1):74–84, 2009.

[19] G. Eckert and W. Müller. *Zentralnervensystem (ZNS) und peripheres Nervensystem (PNS)*, chapter 14, pages 297–310. John Wiley & Sons, Ltd, 2017.

[20] O. Faust, Y. Hagiwara, T. J. Hong, O. S. Lih, and U. R. Acharya. Deep learning for healthcare applications based on physiological signals: A review. *Computer methods and programs in biomedicine*, 161:1–13, 2018.

[21] O. Faust, H. Razaghi, R. Barika, E. J. Ciaccio, and U. R. Acharya. A review of automated sleep stage scoring based on physiological signals for the new millennia. *Computer Methods and Programs in Biomedicine*, 176:81–91, 2019.

[22] E. Fernandez-Blanco, D. Rivero, and A. Pazos. Convolutional neural networks for sleep stage scoring on a two-channel EEG signal. *Soft Computing*, 24:4067–4079, 2020.

[23] L. Fiorillo, A. Puiatti, M. Papandrea, P.-L. Ratti, P. Favaro, C. Roth, P. Bargiotas, C. L. Bassetti, and F. D. Faraci. Automated sleep scoring: A review of the latest approaches. *Sleep medicine reviews*, 48:101204, 2019.

[24] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[25] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex

physiologic signals. *Circulation*, 101(23):e215–e220, 2000 (June 13). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[26] C. W. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

[27] C. W. J. Granger. Economic processes involving feedback. *Information and control*, 6(1):28–48, 1963.

[28] A. Gupta, S. Parameswaran, and C.-H. Lee. Classification of electroencephalography (EEG) signals for different mental activities using Kullback Leibler (KL) divergence. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1697–1700. IEEE, 2009.

[29] H. Haken. Synergetics: an overview. *Reports on Progress in Physics*, 52(5):515, 1989.

[30] S. Hanke, J. Zeitlhofer, G. Wiest, W. Mayr, and D. Moser. Automated vigilance classification based on EOG signals: preliminary results. In *World Congress on Medical Physics and Biomedical Engineering, September 7-12, 2009, Munich, Germany: Vol. 25/9 Neuroengineering, Neural Systems, Rehabilitation and Prosthetics*, pages 428–431. Springer, 2009.

[31] K. P. Hoffmann and U. Krechel. Geräte und Methoden der Klinischen Neurophysiologie (EEG, EMG/ENG, EP). *Medizintechnik: Verfahren—Systeme—Informationsverarbeitung*, pages 129–168, 2007.

[32] C. Iber, S. Ancoli-Israel, A. L. Chesson, S. F. Quan, et al. *The AASM manual for the scoring of sleep and associated events: rules, terminology and technical specifications*, volume 1. American academy of sleep medicine Westchester, IL, 2007.

[33] Q. Ji and X. Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-time imaging*, 8(5):357–377, 2002.

[34] Y. Jiang, C.-K. Peng, and Y. Xu. Hierarchical entropy analysis for biological signals. *Journal of Computational and Applied Mathematics*, 236(5):728–742, 2011.

[35] B. Kemp, A. Zwinderman, B. Tuk, H. Kamphuisen, and J. Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000.

[36] G. Klosh, B. Kemp, T. Penzel, A. Schlogl, P. Rappelsberger, E. Trenker, G. Gruber, J. Zeithofer, B. Saletu, W. Herrmann, et al. The SIESTA project polygraphic and clinical database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):51–57, 2001.

[37] M. Kreuzer, G. L. Keating, T. Fenzl, L. Härtner, C. G. Sinon, I. Hajjar, V. Ciavatta, D. B. Rye, and P. S. García. Sleep/wake behavior and EEG signatures of the TgF344-AD rat model at the prodromal stage. *International journal of molecular sciences*, 21(23):9290, 2020.

[38] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Learning sparse Bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds. *IEEE. Trans. Pattern. Anal. Mach. Intell*, 32, 2005.

[39] M. Kuhn, K. Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[40] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[41] K. Kwon, S. Kwon, and W.-H. Yeo. Automatic and accurate sleep stage classification via a convolutional deep neural network and nanomembrane electrodes. *Biosensors*, 12(3):155, 2022.

[42] Q. Le, T. Sarlós, A. Smola, et al. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, volume 85, page 8, 2013.

[43] R. P. Louis, J. Lee, and R. Stephenson. Design and validation of a computer-based sleep-scoring algorithm. *Journal of neuroscience methods*, 133(1-2):71–80, 2004.

[44] D. M. Mateos, R. G. Erra, R. Wennberg, and J. L. P. Velazquez. Measures of Entropy and Complexity in altered states of consciousness, 2017.

[45] MATLAB. *Version 9.13.0 (R2022b)*. The MathWorks Inc., Natick, Massachusetts, 2022.

[46] C. Metzner, A. Schilling, M. Traxdorf, H. Schulze, K. Tziridis, and P. Krauss. Extracting continuous sleep depth from EEG data without machine learning. *arXiv preprint arXiv:2301.06755*, 2023.

[47] V. Milnik. *Elektrophysiologie in der Praxis*. Elsevier Health Sciences Germany, 2012.

[48] H.-J. Mittag. *Statistik: eine Einführung mit interaktiven Elementen*. Springer-Verlag, 2017.

[49] E. Murillo-Rodriguez. *Methodological Approaches for Sleep and Vigilance Research*. Academic Press, 2021.

[50] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[51] P. Nardone. Entropy of Difference. *arXiv preprint arXiv:1411.0506*, 2014.

[52] E. Olofsen, J. Sleigh, and A. Dahan. Permutation entropy of the electroencephalogram: a measure of anaesthetic drug effect. *British journal of anaesthesia*, 101(6):810–821, 2008.

[53] S. D. Pittman, M. M. MacDonald, R. B. Fogel, A. Malhotra, K. Todros, B. Levy, A. B. Geva, and D. P. White. Assessment of automated scoring of polysomnographic recordings in a population with suspected sleep-disordered breathing. *Sleep*, 27(7):1394–1403, 2004.

[54] A. Popov, O. Avilov, and O. Kanaykin. Permutation entropy of EEG signals for different sampling rate and time lag combinations. In *2013 Signal Processing Symposium (SPS)*, pages 1–4. IEEE, 2013.

[55] M. Popovic. Researchers in an entropy wonderland: A review of the entropy concept. *arXiv preprint arXiv:1711.07326*, 2017.

[56] A. Quintero-Rincón, M. Pereyra, C. D'Giano, H. Batatia, and M. Risk. A visual EEG epilepsy detection method based on a wavelet statistical representation and the Kullback-Leibler divergence. In *VII Latin American Congress on Biomedical Engineering CLAIB 2016, Bucaramanga, Santander, Colombia, October 26th-28th, 2016*, pages 13–16. Springer, 2017.

[57] M. Radha, P. Fonseca, A. Moreau, M. Ross, A. Cerny, P. Anderer, X. Long, and R. M. Aarts. Sleep stage classification from heart-rate variability using long short-term memory neural networks. *Scientific reports*, 9(1):14149, 2019.

[58] A. Rechtschaffen and A. Kales. *A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects*. United States Government Printing Office, Washington DC, 1968.

[59] M. H. Saghayan, S. Seifpour, and A. Khadem. Automated Sleep Stage Scoring Using Brain Effective Connectivity and EEG Signals. In *2021 7th International Conference on Signal Processing and Intelligent Systems (ICSPIS)*, pages 1–5. IEEE, 2021.

[60] S. Sanei and J. A. Chambers. *EEG signal processing*. John Wiley & Sons, 2013.

[61] L. A. Schmidt, K. A. Cote, D. L. Santesso, and C. E. Milner. Frontal electroencephalogram alpha asymmetry during sleep: stability and its relation to affective style. *Emotion*, 3(4):401, 2003.

[62] C. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[63] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[64] V. Svetnik, J. Ma, K. A. Soper, S. Doran, J. J. Renger, S. Deacon, and K. S. Koblan. Evaluation of automated and semi-automated scoring of polysomnographic recordings from a clinical trial using zolpidem in the treatment of insomnia. *Sleep*, 30(11):1562–1574, 2007.

[65] P. Szendro, G. Vincze, and A. Szasz. Pink-noise behaviour of biosystems. *European Biophysics Journal*, 30:227–231, 2001.

[66] A. J. Tallón-Ballesteros and J. C. Riquelme. Data mining methods applied to a digital forensics task for supervised machine learning. *Computational intelligence in digital forensics: forensic investigation and applications*, pages 413–428, 2014.

[67] M. G. Terzano, L. Parrino, A. Sherieri, R. Chervin, S. Chokroverty, C. Guilleminault, M. Hirshkowitz, M. Mahowald, H. Moldofsky, A. Rosa, et al. Atlas, rules, and recording techniques for the scoring of cyclic alternating pattern (CAP) in human sleep. *Sleep medicine*, 2(6):537–554, 2001.

[68] S. Titz. Welt der Physik: Entropie. `https://www.weltderphysik.de/thema/phaenomene-der-thermodynamik/entropie/`, 23 July 2013.

[69] J. Van Der Donckt, J. Van Der Donckt, E. Deprost, N. Vandenbussche, M. Rademaker, G. Vandewiele, and S. Van Hoecke. Do not sleep on traditional machine learning: Simple and interpretable techniques are competitive to deep learning for sleep scoring. *Biomedical Signal Processing and Control*, 81:104429, 2023.

[70] E. J. W. Van Someren. Brain mechanisms of insomnia: new perspectives on causes and consequences. *Physiological Reviews*, 101(3):995–1046, 2021. PMID: 32790576.

[71] E. M. Whitham, K. J. Pope, S. P. Fitzgibbon, T. Lewis, C. R. Clark, S. Loveless, M. Broberg, A. Wallace, D. DeLosAngeles, P. Lillie, A. Hardy, R. Fronsko, A. Pulbrook, and J. O. Willoughby. Scalp electrical recording during paralysis: Quantitative evidence that EEG frequencies above 20Hz are contaminated by EMG. *Clinical Neurophysiology*, 118(8):1877–1888, 2007.

[72] R. Yan, Y. Liu, and R. X. Gao. Permutation entropy: A nonlinear statistical measure for status characterization of rotary machines. *Mechanical Systems and Signal Processing*, 29:474–484, 2012.

[73] M. Zanin, L. Zunino, O. A. Rosso, and D. Papo. Permutation entropy and its main biomedical and econophysics applications: A review. *Entropy*, 14(8):1553–1577, 2012.

[74] Z. Zhang, S. Wei, G. Zhu, F. Liu, Y. Li, X. Dong, C. Liu, and F. Liu. Efficient sleep classification based on entropy features and a support vector machine classifier. *Physiological Measurement*, 39(11):115005, nov 2018.

List of Figures

List of Tables

A. OLS Algorithm

The OLS algorithm is used during the feature extraction for estimating the regression coefficients and the residuals. It is already pre-implemented in the used MVGC toolbox, written by Lionel Barnett and Anil K. Seth, an explanation is given in [6].

The algorithm takes a general representation of the VAR model into account, by defining the different time-series that occur in the full model of equation (4.16) as

$$U_t = \begin{pmatrix} X_t \\ Y_t \end{pmatrix}. \tag{A.1}$$

Let X_t and Y_t be of length n, the time series U_t is of the same length. The model representation can then be extended to the system

$$\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \sum_{k=1}^{p} \begin{pmatrix} A_{xx,k} & A_{xy,k} \\ A_{yx,k} & A_{yy,k} \end{pmatrix} \begin{pmatrix} X_{t-k} \\ Y_{t-k} \end{pmatrix} + \begin{pmatrix} \varepsilon_{XY_t} \\ \varepsilon_{YX_t} \end{pmatrix},$$

with the first row being equivalent to equation (4.16). Using definition (A.1) and summarizing the regression coefficient matrix to A_k and the residuals vector to ε_t, the system can be rewritten as

$$U_t = \sum_{k=1}^{p} A_k U_{t-k} + \varepsilon_t.$$

For the computation, the time series U_t will first be demeaned, i.e. the mean $\bar{U} = \frac{1}{n}\sum_{t=1}^{n} U_t$ is subtracted from each element of the series. The OLS algorithm uses this representation to get estimated regression coefficients $\hat{A}_k$ and estimated residuals $\hat{\varepsilon}_t$, by solving the overdetermined system

$$\sum_{k=1} \hat{A}_k U_{t-k} = U_t,$$

for $t = p+1, ..., n$, with QR-decomposition. This is done such that the mean squared error

$$E^2 = \frac{1}{n-p} \sum_{t=p+1}^{n} ||\varepsilon_t||^2,$$

with $||.||$ representing the L^2 norm, is minimised. Consequently the covariance of the estimated residuals can be computed using the formula for the unbiased sample covariance, see [6],

$$\widehat{\mathrm{Cov}}(\hat{\varepsilon}_t) = \frac{1}{n-p-1} \sum_{t=p+1}^{n} \hat{\varepsilon}_t \, \hat{\varepsilon}_t^T.$$

The output of the algorithm are the estimated regression coefficients $\hat{A}_k$, the estimated residuals $\hat{\varepsilon}_t$ and the estimated covariance $\widehat{\mathrm{Cov}}(\hat{\varepsilon}_t)$. In our case, only the latter one is returned as we only need this for the calculation of the GC.

B. Folddistribution of the Data

Table B.1.: Distribution of the patients among the folds. The underlined patients are the ones that did not have a recording of at least one of the usually taken channels.

Disorder	Fold1	Fold2	Fold3	Fold4	Fold5
No pathology	n4, n9, n16	n5, n15	n1, n6, n10	n2, n7, n11	n3, n8, n12
Bruxism	brux2				
Insomnia	ins1, ins2	ins3, ins4	ins5, ins6	ins7, ins8	ins9
Narcolepsy	narco1	narco2	narco3	narco4	narco5
Nocturnal frontal lobe epilepsy	nfle33, nfle34-nfle39	nfle25, nfle40, nfle 26-nfle32	nfle17-nfle24	nfle9-nfle16	nfle1-nfle8
Periodic leg movements	plm1, plm2	plm3, plm4,	plm5, plm6	plm7, plm8	plm9, plm10
REM behaviour disorder	rbd1-rbd5	rbd6-rbd10	rbd11-rbd14	rbd15-rbd18	rbd19-rbd22
Sleep-disordered breathing		sdb1	sdb2	sdb3	sdb4