UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Dipartimento di Psicologia Generale

Master Degree in Physics

Final Dissertation

Machine Learning approaches in Neuroscience: behavioral and sleep classification

Thesis supervisor

Prof. Samir Suweis

Thesis co-supervisor

Prof. Marco Zorzi

Candidate

Riccardo Cusinato

Academic Year 2020/2021

A Renata,
questa volta e per sempre.

Acknowledgements

I would like to deeply thank my supervisor at Roche, Dr. Roger Redondo, for believing in me for this project. Without his mentorship and advice this work would not have been possible.
I would also like to thank all the people in the group, Simon, Philipp and Marie. A special appreciation goes to my colleague and friend Simon, for his constant support throughout my journey, for everything he taught me and for our many discussions. I hope to continue and have many more in the future.

Finally, I want to express my gratitude to my academic supervisors, Prof. Samir Suweis and Prof. Marco Zorzi, for having accepted to guide me in a different field and for their precious feedback and advice.

Abstract

The understanding of sleep is of paramount importance from a scientific and clinical point of view. Brain disorders, such as Autism and Alzheimer, show disrupted sleep patterns that contribute to the progression of the disease. To obtain efficacious drugs, an important step is to study and test them in rodents, with the hope to extrapolate the findings to humans. In such preclinical studies, it is fundamental to correctly identify and classify sleep phases in order to compare them to the ones found in humans. However, very few works have been carried out in this regard. This Master thesis work is aimed at critically studying this aspect through an approach based on Machine Learning techniques applied to EEG and accelerometer signals. This work will lay the foundation to investigate the differences between wildtype and transgenic mice with the purpose of characterizing the sleep impairment biomarkers of the disease and its trajectory throughout the rodent's life.

Contents

CHAPTER 1

Introduction

Sleep is one of the most ubiquitous processes found in nature, as virtually all creatures from the animal kingdom experience it and need it for their survival. It is still not very clear what the precise function of sleep is, except the fact that there are many and are all very important for health maintenance and survival of the organisms.

Apart from its intrinsic scientific value, the clinical value of sleep studies resides in the fact that sleep disorders affect a very large number of people and their consequences can heavily impair the subjects afflicted. For example, scientists estimate that in a given year insomnia alone afflicts up to a half of the population [1].

In addition to disorders that directly affect sleep quality and quantity, alterations in sleep patterns accompany many neurological and non-neurological pathologies. Examples are Alzheimer's disease (AD) and Autism. AD is a neurodegenerative disease that represents the most common cause of dementia and is accompanied by progressive learning, memory and related cognitive impairments. In the next few years, the number of affected people is expected to grow, mainly for the increase of life expectancy [2]. Moreover, it is one of the most financially costly diseases [3]. On the opposite side of the onset timeline, Autism is a developmental disorder found in children that display strong impairments in communication and social skills, as information processing and synapse connections are disrupted [4]. It is estimated that 1.5% of children in developed countries suffer from autism [5].
These are just two examples of very different brain disorders united by one common factor: patients who suffer from them show disrupted sleep patterns. The spectrum of disorders that display alterations and disruption of sleep is very wide. It is still unclear whether altered sleep appears as one of the consequences of these disorders or whether it may actually play the role of a causal factor [6].

Given the impact of sleep on physical and mental health and its role in many different pathologies, it is very important to study it thoroughly from a scientific and clinical perspective. This thesis work aims at advancing the tools available for the accurate analysis of behavioral activity and sleep in rodents.
In the following, we give an introduction on the origins of the electrical activity of the brain and on how to measure it to obtain quantitative insights on sleep and on brain activity in general.

1.1 Emergent oscillations in the brain

The brain is a very difficult organ to study, as it physically interacts very little with the external word. It is very difficult to see, as it is placed inside the cranial cavity. Even if one could see a working brain, it would not be able to see any appreciable activity, as it does not emit any light (or radiation in general), nor it does produce any mechanical movement. Indeed, its function remained a mystery for many centuries and nowadays there are still many open questions to answer. There are also brains of simpler organisms that are more easily accessible, e.g. small insects or fishes, but of course, they lack the higher complexity found in mammals' brains.

Brain cells, neurons, communicate with each other through electrical signals called action potentials (APs). Action potentials are measured by voltage changes occurring across the cellular membrane. These voltage changes are in turn generated by the flow of charged ions, the main of which are K^+ and Na^+, inside and outside the cellular membrane.

The biophysical mechanism of the generation of APs is well understood and it has been investigated by Hodgkin and Huxley, earning them the Nobel Prize in 1963 [8]. The Hodgkin-Huxley (HH) model considers the cell membrane and its components as an equivalent circuit and explains the generation of APs in terms of changes in the conductances of voltage-gated ion channels. In particular, the resting membrane potential of an average neuron is approximately $-70\ mV$, with the inside more negative than the outside. As it can be seen from Figure 1.1 on the left, an AP mainly possesses two components: an initial depolarizing phase, in which the membrane potential increases, and a successive depolarizing phase, in which the membrane potential returns to the resting state value.
The depolarizing phase starts when the membrane potential rises above a certain threshold, for example after the injection of some source current, and the voltage-dependent conductances of the Na^+ channels increase, opening the channels. Na^+ ions then flow

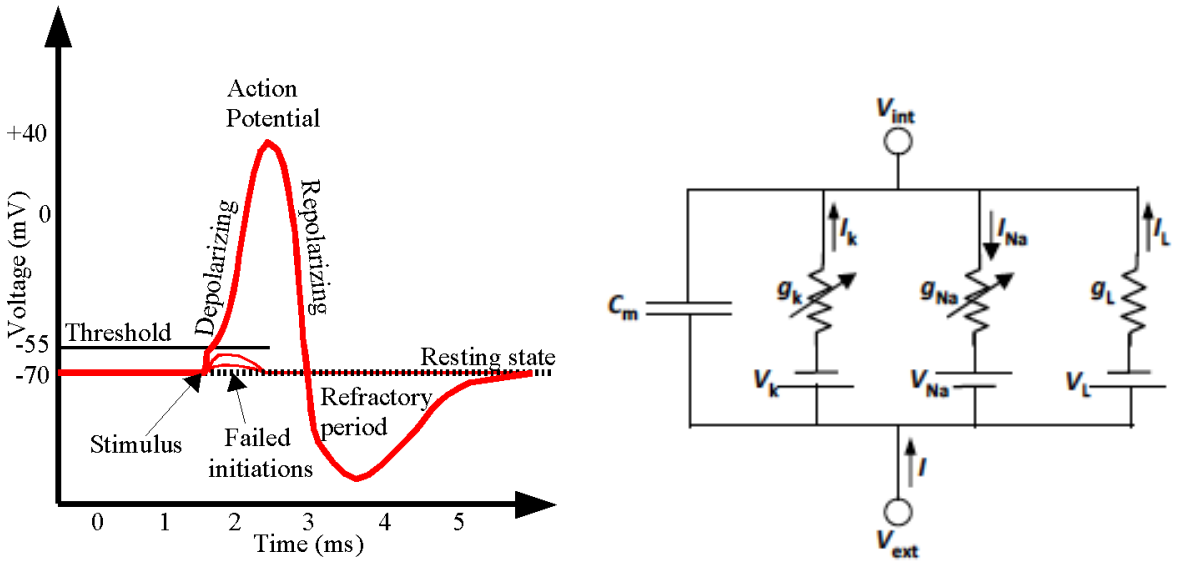

Figure 1.1: **Action potentials**. The Hodgkin-Huxley model quantitatively describes the behavior of an action potential. **Left:** Qualitative behavior of the membrane potential of a neuron during an action potential. Image taken from [7]. **Right:** Equivalent circuit of a neuron's membrane used in the Hodgkin-Huxley model.

inside the cell and depolarize it. This process continues until the repolarizing phase takes place. Two distinct processes drive this phase. On one hand, the conductances of the Na^+ channels start to decrease again as the cell depolarizes and the flux of Na^+ ions starts to slow down; on the other hand, the high membrane potential triggers the opening of the voltage-dependent K^+ channels, which let the ions flow outside the cell and counterbalance the previous Na^+ current. Not only the K^+ current brings the cell again to the resting potential, but it also hyperpolarizes it, meaning that the membrane potential becomes more negative than the resting state one. This hyperpolarization only lasts for a couple of milliseconds, creating the so-called refractory period. During the refractory period, no other AP can be initiated in the same neuron.

APs are the primary targets for experimental techniques like intracellular recordings and extracellular Single-unit recordings (SIR), which are able to pick up the activity of single neurons.

Neurons transmit their APs through their axons and these eventually land on the dendrites of other neurons through synapses. Synapses form the terminal part of a neuron's axon and the beginning of another neuron's dendrites (Figure 1.2). Chemical synapses, which constitute the vast majority of synapses, do not physically connect two neurons.

Rather, chemicals, called neurotransmitters, are released from an axon terminal and land on receptors on the dendrites. These neurotransmitters, in turn, mediate excitatory currents that flow inside the postsynaptic membranes, which are usually constituted by Na^+ and Ca^{2+} ions [9].

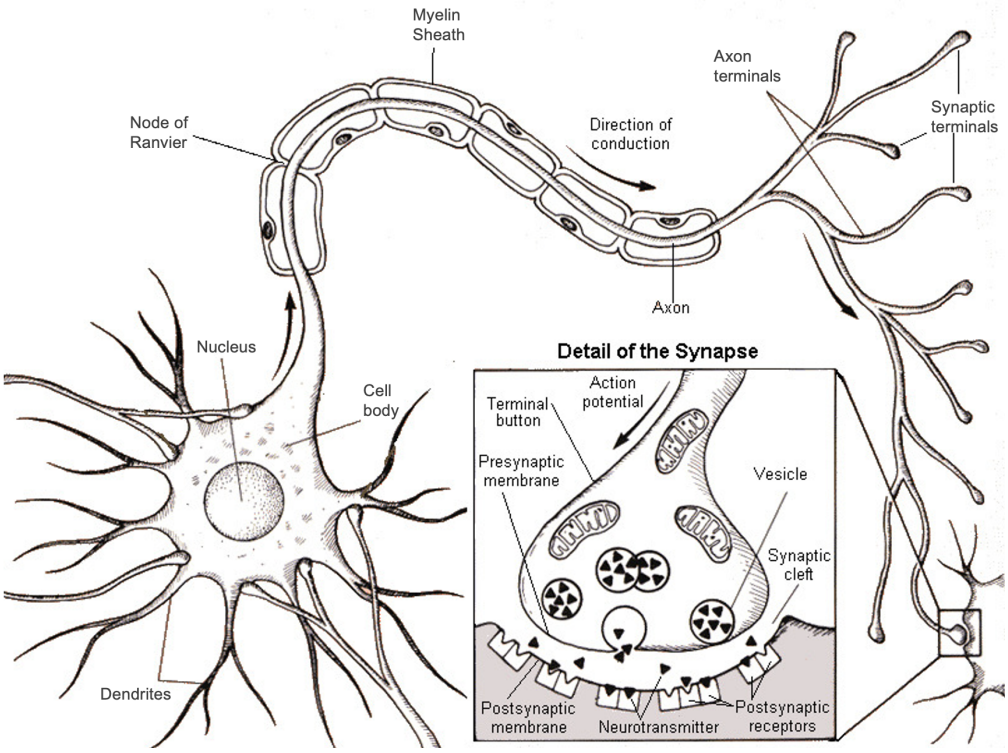

Figure 1.2: **Pictorial representation of a neuron**. Highlighted are the main components. The magnification shows the details of a synapse and how neurotransmitters propagate in the synaptic cleft. Image adapted from [10].

From a more physical perspective, neurons are thus elementary computational units forming complex networks wired through axons, dendrites and synapses [11][12][13]. If we want to understand how the brain works, we need to look the brain as a complex system, i.e., having a systemic view of the brain and considering not only the single neuronal activity but also their concerted interactions. In fact, brain tasks, such as encoding and decoding information from the external world and elaborating behavioural responses, are developed from the collective activity of neuronal populations. A remarkable signature of these coordinated dynamics are sustained oscillations of neural activity, i.e. widespread neuronal collective phenomena observed in brains of different species, e.g., rhythmic patterns of spiking neurons in the central nervous system [14]. These oscillations can be observed and measured, e.g. through electroencephalogram (EEG) [15] (see following section) or

through Local Field potential recording [16].

Such recording techniques that measure the extracellular potential, i.e. the electrical potential measured from a volume of brain tissue, capture the activity of populations of neurons, in which their number can go from tens to thousands of cells.
The activity picked up by these more "coarse grained" techniques does not usually reflect the single APs of the neurons of the population, but it rather comes from other processes [9]. This is easily understandable if one thinks that the resulting electrical field picked up by the electrodes is a linear superposition of the fields generated by all the events involving the movement of ions and that no such individual current has enough strength to be picked up individually by extracellular volume measurements. If a process is very quick (as are APs that last few milliseconds), the synchronization needed between the different neurons to create a detectable signal becomes impossible to reach. On the other hand, slow processes allow hundreds or even thousands of neurons to build a field strong enough to be detected. The strength of the resulting processes roughly follows a power law of the type $P \sim f^{-n}$, with $n = [1, 2]$, between the amplitude of a certain frequency component and the frequency itself, such that slower oscillations have higher amplitudes than faster ones [9].

The processes accounting for the major components of the signals detected are thus "slow" events. In particular, synaptic activity is the major source of extracellular potential. Due to its chemical nature, it is also a very slow event compared to the fast electrical APs. The injection of Ca^{2+} or Na^{+} currents in the neurons following a synaptic event causes the creation of a sink and an opposing return current creates a source at another location. This configuration effectively transforms the neuron into an electrical dipole, which creates an electrical potential decreasing as $1/r^2$, where r is the distance between the neuron and the recording location. Other notable sources of extracellular potential are the relatively long-lasting $(10 - 100\ ms)\ Ca^{2+}$ spikes and ephaptic effects [9].

1.2 How to measure neural activity: EEG

Many different techniques allow registering the activity of populations of neurons. The most widely used, at least in the clinical setting, is the Electroencephalogram (EEG), invented in 1924 by Hans Berger, a German psychiatrist.

EEG uses metal electrodes placed on the scalp to record the activity of the underlying populations of neurons. Due to the quick decay of electrical signals, the recorded neurons are mainly belonging to cortex regions. Since each electrode covers an area of $\sim 1\ cm^2$,

the resulting signal represents the smoothed activity of thousands of neurons. Moreover, the scalp and the other tissues between the electrode and the source further decrease the strength of the resulting field. Nevertheless, very useful and different information can be extracted from the EEG. Other techniques allow monitoring the population activity of the brain, such as Local field potential (LFP), which is similar to EEG, but uses electrodes implanted directly in the nervous tissue and allows to monitor almost every neuron of the population, given a sufficient density of sites.

The EEG possesses a very high temporal resolution but a poor spatial resolution, because it picks up only the average behavior of many neurons. As said in the previous section, the main processes making up the electrical potential picked up by the EEG are the microscopic synaptic and membrane currents. Volume conduction refers to the transmission of the electric field generated by these currents to the recording electrodes. In a first approximation, i.e. when the frequency of the involved process is below ~ 1000 Hz, the problem can be treated using electrostatic equations instead of electromagnetic ones [17]. Therefore, with this approximation we can rewrite Ohm's law as

$$\nabla \cdot (\sigma(\vec{r}) \nabla V_e(\vec{r}, t)) = -C(\vec{r}, t) \tag{1.1}$$

where $\sigma(\vec{r})$ is the conductivity of the medium, $V_e(\vec{r}, t)$ is the electrostatic potential and $C(\vec{r}, t) \equiv \nabla \cdot \vec{j}$ is called current source density (CSD) and is the current per unit volume entering or exiting the extracellular medium at a given position. All the quantities are defined at a certain time t and position $\vec{r}$ in space. If the milieu is isotropic, i.e. if the conductivity does not depend on the position, the equation can be further simplified and resembles the usual Poisson equation for point charges

$$\nabla^2 V_e(\vec{r}, t) = -\frac{1}{\sigma} C(\vec{r}, t) \Longrightarrow V_e(\vec{r}, t) = \frac{1}{4\pi\sigma} \int_V d^3\vec{r}' \frac{C(\vec{r}', t)}{|\vec{r} - \vec{r}'|} \tag{1.2}$$

With Equation 1.2 one can calculate the signal that is picked up by an electrode given the knowledge of the microscopic CSD $C(\vec{r}, t)$. This is called the forward problem. What is typically encountered, instead, is the inverse problem, where the signal at the electrode is known and one would like to reconstruct the microscopic current that generated it [9]. The inverse problem is ill-posed, as many solutions exist given the signal measured and typically different constraints are needed to find a solution, in addition to many recording sites and sophisticated algorithms [18].

A technique very similar to the EEG is the Electrocorticogram (ECoG), in which the

electrodes are placed directly on the surface of the brain, bypassing the attenuation provoked by the scalp and all the tissues in-between. For this reason, the ECoG signals are stronger than the EEG ones, but this technique is also more invasive and it is not usually adopted in experimental settings.
Throughout the thesis, we use the term EEG to indicate the signals recorded in mice, though the recordings were actually ECoGs.

1.3 Brain oscillations

As we have seen, the network of neurons in the brain form a complex system that gives rise to many emergent behaviors [14]. The most prominent of these behaviors in this context is the appearance of spontaneous oscillations. These oscillations can be characterized by the mechanisms that create them and by their frequency.

The EEG signal is usually a very complex signal with a low signal-to-noise ratio. Raw signals, such as the ones in Figure 1.3, need proper treatment, filtering and analysis in order to extract useful information.

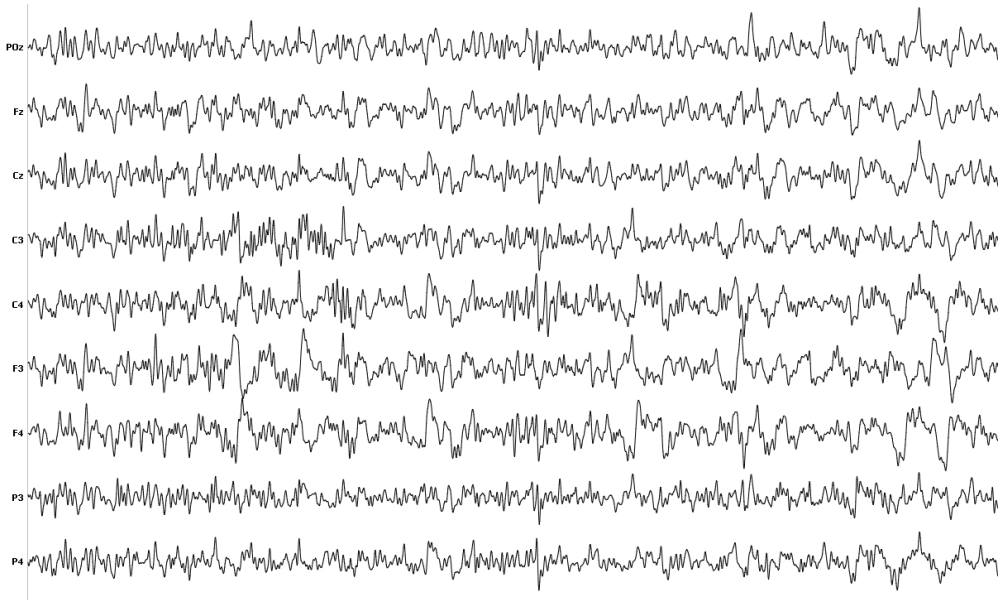

Figure 1.3: **Typical EEG signal**. Raw EEG signals are usually constituted by many channels (electrodes) and display oscillations and complex phenomena. Image taken from [19].

The most widely used methods for EEG analysis are spectral and time-frequency decomposition [20], which work with the Fourier spectrum of the signals. We shall describe these

methods more in detail in section 2.3.

Given the use of frequency components to describe EEG signals, neuroscientists have tried to classify rhythms in different frequency bands, majorly following clinical considerations, associated with different tasks and brain states [21]. The borders of these bands are often arbitrarily drawn, as they are not based on the distinct mechanisms generating them, which are mostly not known [14]. A useful classification of frequency bands could follow a logarithmic scale (Figure 1.4) [22].

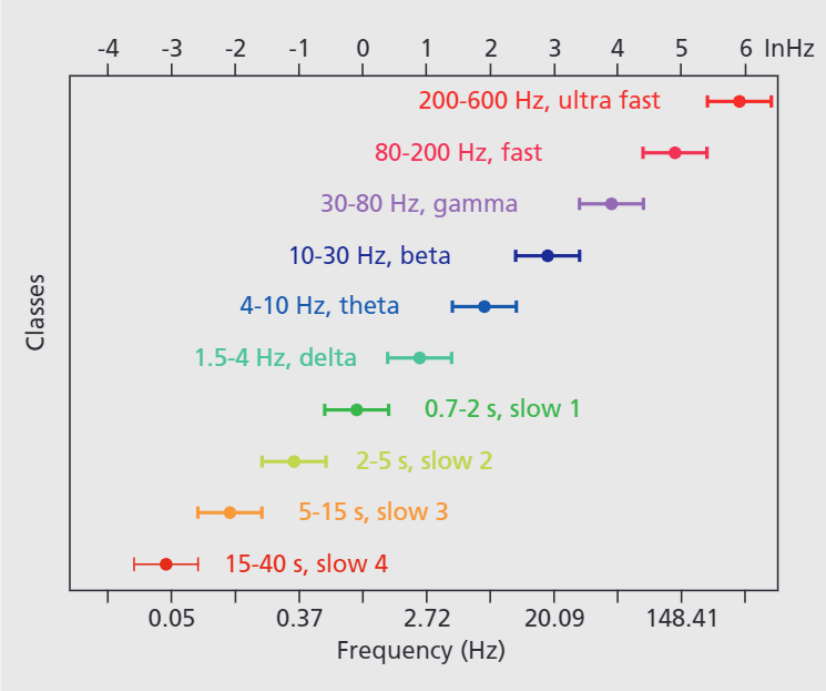

Figure 1.4: **Classes of neural oscillations**. Different oscillatory bands can be ordered on a logarithmic scale, showing the consistency with the classical bands definitions. Image taken from [22].

As a result of these features, EEG signals are time series that possess a stochastic behavior. Like so, their spectrum comprises many different frequency components and can be, in general, very complicated. Usually, higher frequencies are associated with "higher" tasks, i.e. more cognitive tasks such as learning or memory; lower frequencies, on the other hand, are more associated with resting state and sleep.

Another important characteristic of brain oscillations is that they are quite conserved across mammal species and oscillatory bands roughly share the same boundaries. There are also differences, however. For example, the main one is that humans have very slow theta oscillations $(1 - 4\ Hz)$, compared to rodents $(6 - 12\ Hz)$ [22].

1.4 Sleep

Sleep, as a biological process, plays a major role in every living organism [23][24], including mammals [25]. In humans, sleep quality and quantity affect physical [26] and mental [27] health, as well as cognitive processes [28].

Sleep in mammals is divided into two main stages called Rapid Eye Movements (REM) and non-REM (NREM) sleep. REM sleep is characterized by muscles atonia, bursts of rapid eye movements and "desynchronized" EEG patterns that resemble the ones of wakefulness. Sleep has also been defined behaviorally according to the body posture of the animal and the absence of movement [29][30]. An important requirement successively introduced is the presence of an homeostatic regulation of sleep [31].

The regulation of sleep, the timing of its onset and its alternating patterns with wakefulness depend on three distinct systems [32]:

1. The circadian rhythm, which regulates the wakefulness and sleepiness levels and other biological processes following a 24-hour cycle. The circadian rhythm is controlled by the Suprachiasmatic nucleus (SCN), located in the hypothalamus. The SCN primarily regulates the sleep-wake cycle based on light inputs received from the environment.
2. The homeostatic regulation, meaning that there exist biological feedbacks that try to maintain constant the level of sleep an organism gets. This translates into the fact that the amount of sleep depends on the previous time spent awake and the more sleep deprived an organism is, the more "sleep pressure" it has [33]. Sleep pressure can be reliably quantified by using slow-wave activity (SWA), i.e. the power in the EEG signals between $1 - 4Hz$ [34].
3. The ultradian system, which regulates the succession of NREM and REM stages within sleep and their relative duration. NREM and REM stages alternate between each other during sleep, forming sleep cycles that last from the beginning of a REM interval to the beginning of the next. These sleep cycles are a very conserved characteristic that has been found in many mammals, but the duration of each cycle and the proportion of time spent in REM sleep varies greatly between different species [35]. For example, sleep cycles in humans last about 90 minutes, while only around 10 minutes in mice [25].

Figure 1.5 illustrates a model of how sleep pressure is related to the circadian and homeostatic systems. For the homeostatic mechanism, sleep pressure accumulates depending on

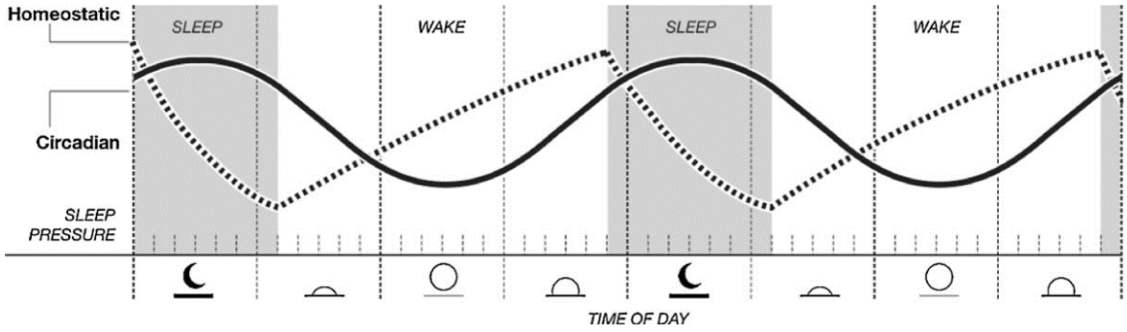

Figure 1.5: **Circadian and homeostatic processes**. These two processes influence the sleep pressure in a rhythmic way throughout the day.

the amount of time spent awake, reaches its maximum right before sleep and drops during the night. The circadian cycle, instead, is a periodic regulation that depends only on the time of the day and not on the previous amount of sleep. It is known that these two systems can work independently, but also contribute to the sleep pressure in complicated manners. It remains still unclear, however, whether these two systems can directly influence each other [36].

The work of this thesis does not address questions of circadian or homeostatic sleep. Instead, this work focuses on the ultradian system and aims at developing better detection algorithms for NREM and REM phases, as we shall see in chapter 4.

1.4.1 Sleep evaluation in humans

The evaluation of human sleep is carried out in sleep laboratories that possess specific instruments that allow for quantitative measurements of physiological quantities. The American Academy of Sleep and Medicine (AASM) recommends the use of the EEG to measure brain activity, of the Electromyogram (EMG) to measure muscles' activity and of the Electrooculogram (EOG) to measure eye movements [37]. Additional recordings may include cardiac and respiratory activities. The overall recording is called polysomnography (PSG).

Human sleep is divided into different stages based on specific characteristics of the PSG signals. The AASM distinguishes four different stages of sleep: N1, N2, N3 and REM sleep [37], with the first three stages being subdivisions of NREM sleep. The approach for scoring sleep is to divide the recordings into epochs of 30 seconds and assign a stage to each.

A summary of the rules used to score sleep in healthy adult subjects and of the charac-

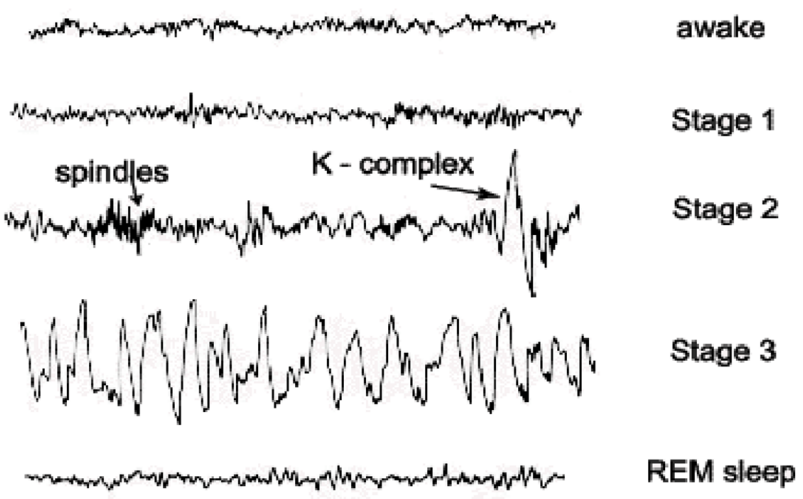

Figure 1.6: **Wave patterns of different sleep stages**. Highlighted are the spindles and K-complexes of the N2 stage and the high-amplitude slow-waves of N3. Image taken from [38].

teristics of each stage is presented below and visually displayed in Figure 1.6 [37]:

1. During N1 the EEG activity "slows down" from the alpha rhythms ($8 - 13$ Hz) of Wakefulness to low amplitude $4 - 7Hz$ activity. The activity in the EMG diminishes with respect to wakefulness and the EOG presents slow rolling eye movements. This stage usually occurs at the transitions from wakefulness to sleep and accounts for $3 - 8\%$ of total sleep time (TST).
2. The N2 stage presents characteristic waves called K-complexes and spindles. The first ones are single slow waves ($1 - 2$ Hz) with high amplitude, while the second ones are brief ($>= 0.5s$) bursts of activity with frequency $11 - 16Hz$. This stage has the highest presence in human sleep, usually accounting for $45 - 55\%$ of TST.
3. The N3 stage is also called slow wave sleep (SWS) because it is characterized by high-amplitude, low-frequency delta oscillations in the EEG. The EMG shows very little activity. It is the "deepest" stage of sleep and accounts for 15-20
4. REM sleep features a complete paralyzation of the body, so the signal of the EMG is flat. The EOG shows irregular eye movements that give the name to this stage. The EEG patterns during REM sleep resemble the ones of N1 and those of wakefulness, with low-amplitude, mixed-frequency activity. REM sleep usually accounts for 20–25% of sleep time.

As already mentioned, sleep stages follow a cyclic behavior. The time course of the different sleep stages during the night can be visualized through a hypnogram, an example of which is shown in Figure 1.7. The content and duration of cycles change throughout the night. In the first cycle, sleep gets progressively deeper, descending from N1 to N3, then going back to N2 and entering REM sleep. The first cycles are usually longer and contain a higher amount of deep N3 sleep, due to the homeostatic regulation. Later cycles, instead, contain a higher amount of REM sleep.

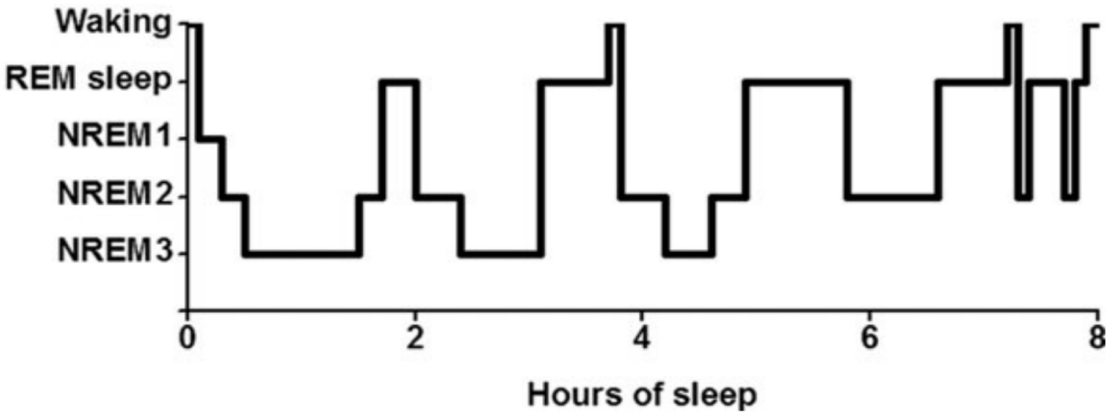

Figure 1.7: **Hypnogram**. An hypnogram shows the succession of the different sleep stages during the night. Image taken from [33].

1.4.2 Sleep in rodents

Rodents constitute a precious tool for scientific and clinical research, including sleep research. Importantly, it has been shown that the generators of sleep-wake transitions and the organizers of brain activity within a wake-sleep state are conserved enough between humans and mice to fulfill back and forward translation goals [39].

However, sleep in other mammals is not studied as extensively and in such detail as in humans. NREM sleep in mice is very often regarded as a continuous and uniform stage [35]. Different studies regarding automatic staging in mice divide sleep in just NREM and REM sleep [40][41][42], sometimes even using the words NREM and slow wave sleep (SWS) interchangeably [43][44].

Although some past animal studies have divided NREM sleep into a "light" and a "deep" stage mostly based on EEG amplitude and slow waves criteria [45][46], no general consensus as it exists in humans has been reached. Nevertheless, sleep is a very conserved

process between mammals, so it is natural to think that rodents would also show consistent differences in their NREM sleep. Recently, Lacroix et al. [47] proposed that mice possess three stages of NREM sleep as well, which can be matched quite well with the three phases present in humans.

The approach of the cited studies has relied more on the definition of different brain states adapted to the data. Differently from them, in the present work we try to address the problem of identifying further stages within NREM in mice starting from data, by means of unsupervised models. With the right features, these models would allow to find consistent patterns within data and to test if these patterns really correspond to different stages, as we shall discuss in chapter 4.

CHAPTER 2

Machine Learning and Signal Processing

In this chapter, we present and explain the computational models and tools used for the analysis of the time series. The major framework employed is the one of Machine Learning (ML), which is first outlined in detail in the next section, to then discuss the particular algorithms used in the analyses. In addition, the signal processing techniques common to the different parts of the thesis are presented. The different functions and algorithms presented in this thesis were implemented using the *Python* package *Scikit-learn* [48].

2.1 Machine learning framework

Artificial intelligence (AI) is a vibrant field nowadays, with new branches of research developing quickly and with many fields of application. Although it may seem recent, its roots are quite far in the past, with the first research on artificial intelligence being conducted in the '50s. Generally speaking, AI algorithms are made by setting a set of rules that allow them to identify different patterns in the given task. This procedure can be considered proper AI, in the sense that a machine is programmed with some rules, acquiring the "intelligence" of applying those rules in the context and getting the results of the computations. This approach has the advantage of exploiting the great computational capabilities of computers with a set of fairly simple and logical rules applied sequentially to even very vast amounts of data, thus making the analysis fast and reliable. However, in many real world cases, being able to build a complete model with related logical rules in order to perform specific tasks or respond to environmental input is far from trivial. Another complementary approach is

instead to use directly the data to program the AI. In fact, all the major achievements we are experiencing today and the so called AI revolution have been made possible by the availability of "Big Data" through which it is possible to train different types of ML algorithms.

ML algorithms, in fact, constitute a subclass of AI and the tasks they are better suited for are the ones that are very difficult to define by a set of logical rules, for example that depend on the context or simply would require too much computational power to be carried on in a rules-based manner. Famous examples are image recognition tasks, translating from a language to another and learning to play games more complicated than chess, such as go, poker or Jeopardy. As we have explained above, in ML the algorithms learn directly from data. As posed by Mitchell [49], learning from examples, that is, from experience, means that the algorithm gets progressively better with experience, as measured by some performance metric. The examples that the algorithm experiences are collected in a dataset, called $\boldsymbol{X}$. In this chapter we indicate vectors and matrices with bold symbols. Each example of $\boldsymbol{X}$, called $\boldsymbol{x}_n$, is usually a collection of observables called features, which can often be represented as vectors in $\mathbb{R}^m$. Examples are physical quantities related to the weather, such as pressure, temperature and humidity, or the height of people. The entire dataset can then be represented as a matrix with N rows and M columns, where N is the number of examples and M the number of features. Extracting the right features for a specific problem can be very time consuming and requires domain-specific knowledge.

Two very broad types of ML algorithms can be identified based on the type of dataset they experience.

The first one concerns supervised learning algorithms. In supervised algorithms, the dataset is composed of examples $\boldsymbol{x}_n$ with associated labels y_n, that can collectively be indicated as $\boldsymbol{Y}$. These labels are often referred to as the "ground truth", because they can be provided by a "teacher" that shows the machine what should be achieved at the end of the learning process. A typical example of supervised tasks is image recognition. In this domain, it is quite easy for a human to assign an image to a certain category. The purpose of a supervised algorithm would be to find how to correctly associate each input with the corresponding provided label. In this case, where labels are categorical, one talks of classification tasks. We deal with a supervised classification task in chapter 3, since our aim is to correctly classify different behaviors of mice based on accelerometer signals. On the other hand, if labels are real numbers, the task is called regression. An example of a regression task would be predicting the temperature based on data about the pressure and the humidity.

The other type of algorithms is called unsupervised learning algorithms. In an unsupervised

setting no labels are provided and the dataset is composed of just samples $\boldsymbol{x}_n$. Typically, the goal of an unsupervised algorithm is to find different representations of the input variables, while still preserving much of the information contained in the original features. The new variables formed from the original ones are often called latent (or hidden) variables. The concept of latent variables is extremely important for us, as we employed latent models in chapter 4. Typical tasks concerning unsupervised learning are density estimation and clustering. In particular, the goal of clustering algorithms is to partition the dataset into different clusters, such that the distance between samples in the same cluster is small compared to the distance with samples in other clusters. Accordingly, samples with similar features are found in the same cluster and different clusters represent subsets with different characteristics.

The majority of ML algorithms are described by the same building blocks: a dataset, a loss function, an optimization procedure and a model [50]. A model here is referred to as a class of parameterized functions. The different classes of functions have different representational powers, meaning that more complicated models can represent a wider range of relationships in the dataset with respect to simpler ones. We already briefly described the datasets and in the following sections we describe the cost function and the optimization procedure, putting an emphasis on the connection between ML and statistics.

2.1.1 Loss function

The parameters of a ML model are found by minimizing a function, called loss function, $J(\boldsymbol{\theta})$. Loss functions can be best understood in the statistical framework of Maximum Likelihood Estimation. The likelihood function is made by the product of the probabilities of each sample in the dataset. In the case of supervised learning, each sample is described by the conditional probability $p(y_n|\boldsymbol{x}_n)$, with y_n being the label associated with the sample $\boldsymbol{x}_n$. In the unsupervised setting, instead, only probabilities $p(\boldsymbol{x}_n)$ over the samples are considered. If the samples are independent and identically distributed (i.i.d.), the Likelihood function is the product of the probabilities of the single samples:

$$\mathscr{L}(\boldsymbol{\theta}) = \prod_{n=1}^{N} p_{model}(y_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \tag{2.1}$$

Where $\boldsymbol{\theta}$ is the set of parameters describing the model and $p_{model}(y_n|\boldsymbol{x}_n;\boldsymbol{\theta})$ is the conditional probability of the n^{th} label given the corresponding sample. The subscript "model" indicates

that the probability distribution is described by the chosen model and is parametrized by $\boldsymbol{\theta}$. It is common practice to work with the logarithm of the Likelihood function, because $\mathscr{L}$ usually takes very low values, being a product of many small numbers. Taking the logarithm has also the advantage of transforming the product into a summation, such that:

$$log(\mathscr{L}(\boldsymbol{\theta})) = \sum_{n=1}^{N} log(p_{model}(y_n|\boldsymbol{x}_n;\boldsymbol{\theta})). \tag{2.2}$$

In the framework of the Maximum Likelihood, the best parameters $\boldsymbol{\theta}_{MLE}$ are found by maximizing the log likelihood function:

$$\boldsymbol{\theta}_{MLE} = \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} log(p_{model}(y_n|\boldsymbol{x}_n;\boldsymbol{\theta})) \longleftrightarrow \arg\max_{\boldsymbol{\theta}} \mathbb{E}[log(p_{model}(y|\boldsymbol{x};\boldsymbol{\theta}))] \tag{2.3}$$

The second expression is an expectation value over the whole population obtained by dividing the first expression by the number samples and letting it tend to infinity. The maximum likelihood estimation possesses nice statistical properties that make it very suited for parameter estimation. In particular, it is the best estimator in terms of rate of convergence in the asymptotic limit of infinite number of samples and it is a consistent estimator [50]. The property of consistency means that, in the asymptotic limit, the estimated value approaches the true value of the parameters. Therefore, in the context of maximum likelihood estimation, the learning process of a machine learning algorithm can be described as the maximization of the log-likelihood function. Or, equivalently, as the minimization of the negative log-likelihood. When the loss function is taken as the negative log-likelihood, it is called cross-entropy:

$$J(\boldsymbol{\theta}) = -\mathbb{E}[log(p_{model}(y|\boldsymbol{x};\boldsymbol{\theta}))]. \tag{2.4}$$

A very important example for us is the cross-entropy loss function used in classification tasks, where the output is an integer number in the set $\{0, ..., C-1\}$, where C is the number of classes. In this case, one usually uses the softmax function to define the conditional probability for each class:

$$\hat{y}_i = p_{\boldsymbol{\theta}}(y = i|\boldsymbol{x}) = \frac{exp(z_i)}{\sum_{j=0}^{C-1} exp(z_j)} \tag{2.5}$$

This function ensures that the outputs are indeed probabilities and sum up to 1. The z_i are called logits and are in general non-linear functions of the inputs that depend on the

parameters $\boldsymbol{\theta}$.
The likelihood function of a dataset with N samples is then written as:

$$\mathscr{L}(\boldsymbol{\theta}) = \prod_{n=1}^{N} \prod_{i=0}^{C-1} \hat{y}_{n,i}^{t_{n,i}} \tag{2.6}$$

where we introduced a vector for each sample, $\boldsymbol{t}_n$, whose components are all zeros except a 1 in the position corresponding to the class of that sample. Then, the cross-entropy loss function is constructed as:

$$J(\boldsymbol{\theta}) = -log(\mathscr{L}(\boldsymbol{\theta})) = -\sum_{n=1}^{N} \sum_{i=0}^{C-1} t_{n,i} log(\hat{y}_{n,i}) \tag{2.7}$$

There is a very useful advantage in using this function because the log and the exponential in the softmax function "cancel out", allowing to avoid numerical problems in software implementation.

Another very nice property of the log-likelihood function is that minimizing it is equivalent to maximizing the Kullback-Leibler divergence $D_{KL}[p(\boldsymbol{x};\boldsymbol{\theta}^*)||p(\boldsymbol{x};\boldsymbol{\theta})]$ [51]. $\boldsymbol{\theta}^*$ denotes the set of true parameters we want to find and the Kullback-Leibler divergence between two discrete probability distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ is defined as:

$$D_{KL}[p||q] = \sum_{i} p(\boldsymbol{x}_i) log\left(\frac{p(\boldsymbol{x}_i)}{q(\boldsymbol{x}_i)}\right) \tag{2.8}$$

2.1.2 Generalization and Regularization

The generalization properties of ML algorithms are very important, because the purpose of the learning process is to acquire the ability to perform a task on previously unseen samples. The generalization capacity of an algorithm is probed by dividing the entire dataset in two subsets: the training set and the test set. The train set is used to adjust the parameters of the model and is the one that drives the learning. It is usually bigger than the test set, comprising $70 - 80\%$ of the total samples. The test set, on the other hand, is never used during the training process, but it is employed at the end of it to compute the generalization performance on unseen samples. The rationale behind this splitting is to ensure that the examples on which the performance of the algorithm is computed do not take part in the optimization of the parameters, which would otherwise result in overestimation

of the performance.

The error achieved at the end of the training procedure is not sufficient to determine the goodness of the final performance. This can be understood by introducing the concepts of bias, variance and model complexity.
The bias measures how much of the actual complexity of the task can be captured by the model, in the limit of having an infinite number of samples. A model with a high bias is said to be underfitting. In this context, the variance represents the expectation value of the variance of the values of the parameters when the learning is carried out using different samples taken from the same population. The variance of a model measures how the best estimate of the parameters varies when using different samples as training set, thus offering a way to measure how robust the model is. A model with a high variance is said to be overfitting. For i.i.d. samples with mean squared error loss function, the test error of the model can be exactly decomposed into bias, variance and a noise source due to the intrinsic stochastic nature of the process [52]. The complexity (or capacity) of a model can be thought of as its power to represent different functions and to adapt to the data. A higher complexity is usually accompanied by low bias and high variance, causing overfitting; while lower complexity may cause underfitting by not capturing more subtle correlations present in the dataset.

Three "regions of errors" are highlighted in Figure 2.1, as a function of the model complexity. What is important to keep in mind is that the error reached on the training set alone is not informative. Instead, one always needs to compare this error with the generalization error of the model on a held-out set. When the model complexity is low, both the training and the test error are high and these errors are dominated by the bias term. This is the region of underfitting. On the other hand, when the complexity is too high, the model enters in an overfitting region, where the training error is much smaller than the test one and the variance contributes more. There is a "sweet spot" in-between when the generalization error reaches its minimum and the model complexity is optimal for the task.

Determining the complexity analytically is basically impossible in any practical application and thus is usually controlled by the so-called hyperparameters. All the parameters of the model that are not directly trained by minimizing the loss function are called hyperparameters. Different ML algorithms have different hyperparameters and the their type can be very different. Of particular importance in the discussion about model complexity are the hyperparameters that control the regularization strength.

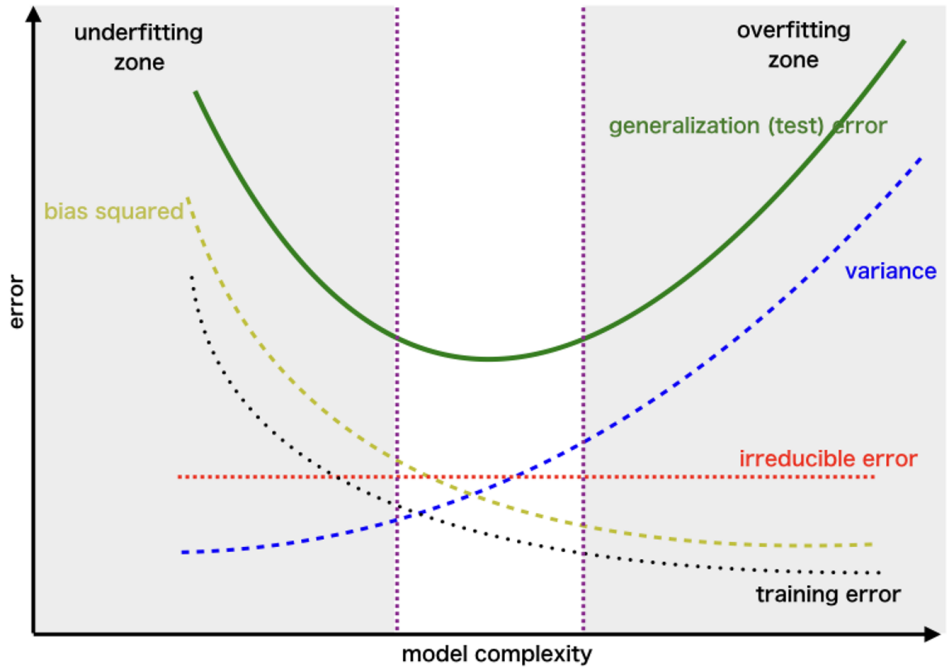

Figure 2.1: **Bias-variance tradeoff**. The image shows different sources of test error when increasing the model complexity. When model complexity is low, the error is dominated by the bias components, while at high complexity the major contribution to the error is due to the variance. There is always an irreducible error due to the intrinsic stochastic nature of the phenomenon. Image taken from [53].

Regularization of a ML algorithm is introduced as an additional term in the loss function, which penalizes models with a high complexity. Two examples of how this can be achieved in practice are the L_1 and L_2 regularizations, also called Lasso and Ridge. The regularized loss function becomes:

$$E^{reg}(\boldsymbol{\theta}) = E(\boldsymbol{\theta}) + \lambda \sum_i |\theta_i|^{1,2} \tag{2.9}$$

λ is the hyperparameter controlling the regularization strength, i.e. how important is the regularization term relative to the unregularized loss function. The exponents $1, 2$ refer to the Lasso and Ridge regularizations, respectively, and correspond to taking the absolute or the squared value of the parameters.

To determine the best values of the hyperparameters, another dataset, different from the training and the test ones, is needed [54]. The reason lies in the fact that, to obtain an unbiased measure of the performance of the model, no samples used to train the model should be used to find the best hyperparameters, and vice versa. In modern practice, one usually uses k-fold cross validation (CV) [52], where the validation set is not kept fixed but

rather, a part of the training set is used to actually train the model and another part is used as validation set. The training set is split in k approximately equal parts and each time the model is trained on $k-1$ folds and then performance measures are computed on the left-out fold; the process is complete once every fold has been used as validation set. The use of CV allows the calculation of statistics about the expected generalization error and provides a way for choosing the best hyperparameters values. For the latter task, the Grid Search CV is usually adopted. This technique consists in constructing an array of values for every hyperparameter to optimize and repeatedly perform the training procedure for every combination of hyperparameters, each time reinitializing the model parameters (e.g., the regression weights). The CV part means that performance metrics are evaluated using k-fold cross validation. Then, the final model is learned on the entire training set using the optimal set of hyperparameters found with the Grid Search procedure. As we shall see, we used the CV and Grid Search procedures in the supervised classification task in chapter 3.

2.1.3 Optimization

In some cases, the loss function is convex with respect to the parameters of the model and it is possible to find a global minimum in one single step. In most cases, however, the shape of the loss function is so complicated that an iterative procedure is needed to update the parameters until the convergence to a local minimum is reached. The most widely used optimization procedure in ML is called Gradient Descent (GD). The idea of GD is to follow the negative gradient of the loss function in order to reach local minima, because the gradient of a function points in the direction of higher values of that function.

Since loss functions can generally be written as a sum over the data points, so can the gradient:

$$\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}) = \sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}} e(\boldsymbol{x}_n, \boldsymbol{\theta}) \tag{2.10}$$

Then, the parameters are updated with the following rule:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \epsilon \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}) \tag{2.11}$$

ϵ is the main hyperparameter regarding the learning and it is called learning rate. It is very important to choose an appropriate value of the learning rate: if it is too small, the convergence is very slow, while if it is too big the algorithm may not converge at all [55]. The reason why these behaviors arise is that the approximation that the GD algorithm uses,

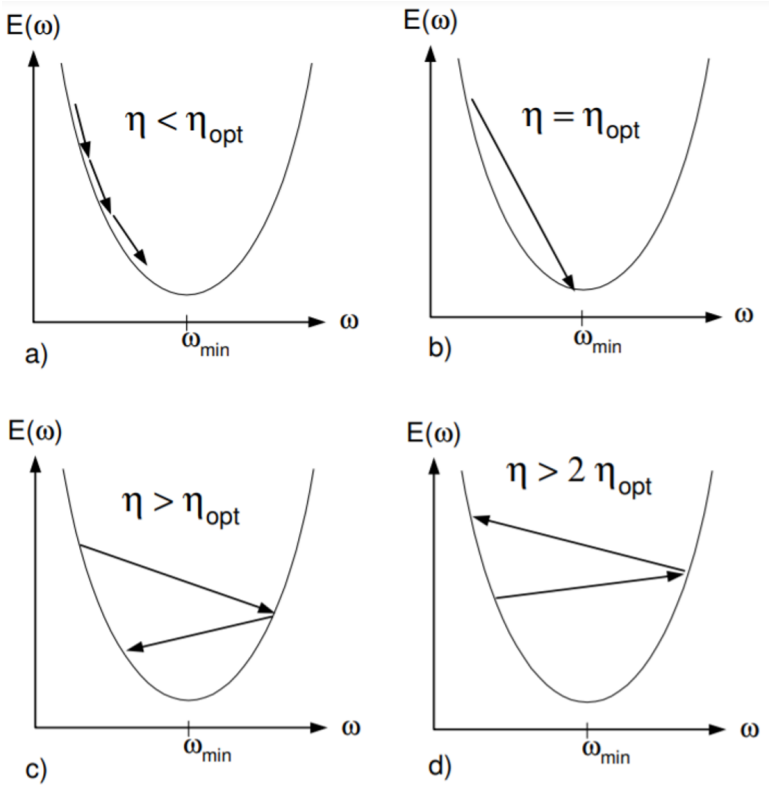

Figure 2.2: **Different regimes of learning**. Successful learning in ML algorithms critically depends on the value of the learning rate. Values too low slow down the learning, while this may be prevented with values too high. Image taken from [55].

namely the use of the gradient to minimize the loss function, is only locally valid. For this reason, the learning rate is usually a small number.

One way to avoid these behaviors is to choose an optimal learning rate at each iteration and for each parameter, based on the information on the curvature of the function. This method is known as Newton's method and is equivalent to a second-order expansion, in which the Hessian matrix is computed. Although this method is more precise, it is rarely used, because it is quite computationally costly. Other methods are usually preferred, for example the addition of a momentum term, inspired by the viscosity of physical systems [56].

A second problem of the GD algorithm is that it often gets stuck in "high" local minima, meaning points with a much higher value than the global minimum. This problem can be alleviated by using a modified version of the GD, called Stochastic GD (SGD). In SGD, the gradient for each optimization step is computed using just one random sample at a time, without replacement, and keeping the update rule the same. The advantage in

introducing stochasticity in the algorithm is that it makes it easier to escape from local minima. Additionally, the computational cost of SGD is much less than the one of GD and SGD usually converges faster [50]. The only drawback of SGD is that using one sample at a time to update the gradient of the loss function can result in instability in the updates of the parameters. There are two main strategies to mitigate this: the first one is to decrease the value of the learning rate as the training proceeds, such that later updates have less influence on the parameters; the other widely used technique is to use mini-batches instead of a single sample at a time. The size of the mini-batches depends on the task and the available data, but their use combines the advantages of GD and SGD algorithms. The majority of the algorithms, including the ones presented in this thesis, work with mini-batch SGD.

2.1.4 Metrics

In this section we present the performance metrics that we used to evaluate the goodness of the outputs produced by the different algorithms, both supervised and unsupervised. Figure 2.3 shows an example of confusion matrix for a binary classification. The number of

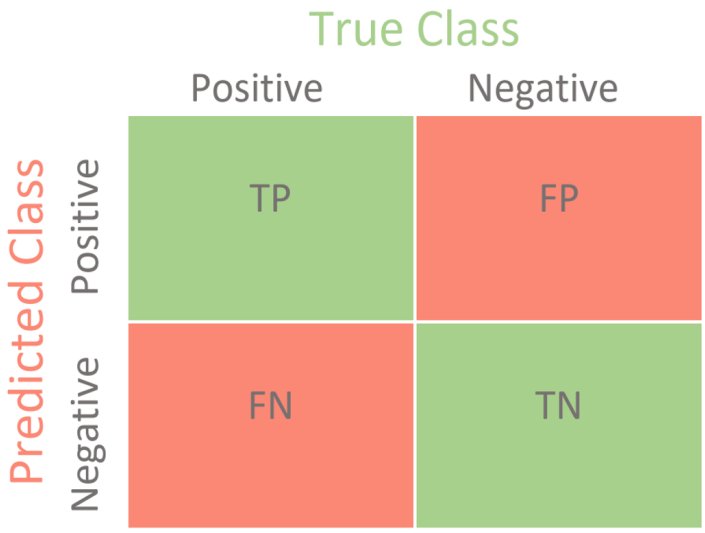

Figure 2.3: **Confusion matrix**. A confusion matrix displays the numerical (or fractional) values of TP, FP, TN, FN. On the rows there are the true labels, while on the columns the predicted ones. It can be very useful to summarize the performance of classification tasks. Image taken from [57].

true positives (TP) and true negatives (TN) identify the samples that have been correctly classified, while the false positives (FP) are the samples incorrectly predicted as positives and the false negatives (FN) the samples incorrectly predicted as negatives. Ideally, one would like to obtain zero FP and FN. To assess the performance of a classifier, we used several metrics for the tasks in chapter 3 and chapter 4:

- Balanced-accuracy $= \frac{1}{2} \cdot (\mathrm{TP}/(\mathrm{TP}+\mathrm{FN}) + \mathrm{TN}/(\mathrm{TN}+\mathrm{FP}))$
- Precision (P) $= \mathrm{TP} \;/\; (\mathrm{TP} + \mathrm{FP})$
- Recall (R) $=$ Sensitivity $= \mathrm{TP} \;/\; (\mathrm{TP} + \mathrm{FN})$
- F1-score $= 2\ (\mathrm{P} \cdot \mathrm{R}) \;/\; (\mathrm{P} + \mathrm{R})$

The balanced accuracy is an average of the recall of the positive class (also called sensitivity) and the recall of the negative class (also called specificity) and is very useful in the case of imbalanced classes. Intuitively, P measures how precise is the classifier in identifying as positives only the samples that are actually positives, while R measures how many actual positives the classifier can identify. The F1-score is the harmonic mean of P and R and is used when one wants both high precision and recall and the classes are imbalanced. In most of the tasks presented in this thesis, we sought high P and R in the identification of the classes and thus used the F1-score as metrics.

To compute these metrics in a multi-class situation, it is sufficient to consider each time one class as positive and the other ones as negative. This scheme is called One-vs-Rest classification. In this way one obtains a set of metrics for each class. To get a single value out of the different classes, averaging is needed. In the particular case, in chapter 3 we used "macro" averages, which means that we took an average value of the metrics for the different classes irrespective of class imbalances.

2.2 Machine Learning Algorithms

In this part we briefly describe the algorithms that were used in the data analysis.

2.2.1 Logistic regression

Despite its name, logistic regression (LR) is a supervised learning algorithm that works with classification tasks. We first describe the LR in the case of a binary classification task and then generalize to the case of multiple classes.

If the classification is binary, the labels can assume the values $\{0, 1\}$. 1 is called the positive class and 0 is the negative class. A logistic model describes the probability of a sample to belong to the positive class by using a logistic function:

$$\hat{y} = P_{\boldsymbol{\theta}}(y = 1|\boldsymbol{x}) = \sigma(z) = \frac{1}{1 + e^{-z}} \qquad \text{with } z = \boldsymbol{w}^T \cdot \boldsymbol{x} + b \tag{2.12}$$

The variable z is called logit and, importantly, is a linear function of the inputs. The

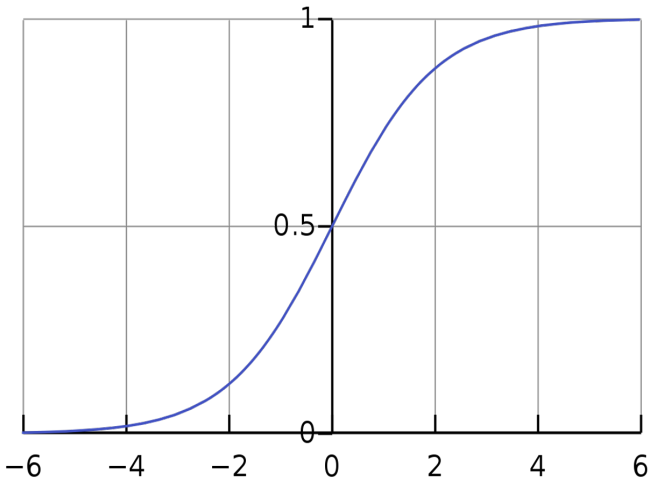

Figure 2.4: **Logistic function**. The logistic (or sigmoid) function $\sigma(z)$ as a function of z.

logistic function only assumes values in the interval $(0, 1)$, making it very suited to represent probabilities. A value of 1 means certainty of belonging to the positive class, while a value of zero translates into certainty of belonging to the negative class, since $P_{\boldsymbol{\theta}}(y = 0|\boldsymbol{x}) = 1 - P_{\boldsymbol{\theta}}(y = 1|\boldsymbol{x})$.

It can be shown that with many conditional distributions, such those belonging to the exponential family, the expression of the logistic model for the posterior probability is exact [58]. The LR algorithm learns the parameters of the model, namely the weights $\boldsymbol{w}$ and the bias b, collectively called $\boldsymbol{\theta}$. It does this by minimizing the cross-entropy loss function, already discussed in subsection 2.1.2.

Let's call again the model's probability prediction for sample n $\hat{y}_n$ and the label associated with it y_n.

$$J(\boldsymbol{\theta}) = -log\mathscr{L}(\boldsymbol{\theta}) = -\sum_{n=1}^{N}[y_n log(\hat{y}_n) + (1 - y_n)log(1 - \hat{y}_n)] \tag{2.13}$$

Due to the presence of an exponential in the analytical form of $\hat{y}_n$, a closed-form equation for the optimal parameters does not exist. Therefore, the optimal parameters $\boldsymbol{w}, b$ are found by iterative minimization, as described in subsection 2.1.3.

In the form presented, the LR algorithm is not a classifier. To make it such, one has to decide a probability threshold above which a sample is classified in the positive class. This value is often chosen to be 0.5 and one can imagine that a hyperplane of constant probability divides the two classes in the feature space. A problem that could arise in this way is that many different hyperplanes could exist that optimally separate the two classes. For this reason, the LR algorithm is also often used in its regularized form, which is equivalent to a Bayesian treatment of the parameters [58]. The regularization term is introduced in the loss function as an L_1 and/or L_2 norm of the weights.

The generalization to multiple classes is straightforward and resembles the steps described in subsection 2.1.2 with the multiclass cross-entropy. In particular, the multiclass LR uses the softmax function instead of the sigmoid and the multiclass cross-entropy as loss function. Both the sigmoid and the softmax functions lead to very good numerical properties of the algorithms, in that the exponentials in them are "cancelled" by the logarithm in the cross entropy loss, leading to linear terms.

2.2.2 Random Forest

Random Forests (RF) [59] are an ensemble method constructed starting from decision trees for supervised classification and regression tasks. A classification decision tree partitions the feature space into different regions in a hierarchical fashion, by splitting the data into smaller subsets, and assigns a class label to each region. An example of classification tree is shown in Figure 2.5.

Starting from the root, at each node the dataset is split based on the feature that best separates the data in the different classes. A common measure to quantify the "goodness" of separation is the Gini impurity. The Gini impurity of a node measures how uniformly samples of different classes in the node are distributed, giving its maximum value (0.5) when the classes are uniformly represented and its minimum (0) when all samples belong to one class.

$$Gini(D_n) = 1 - \sum_{k=1}^{K} p_k^2 \tag{2.14}$$

where D_n represents the node D containing n samples and p_k is the probability that a sample in the node belongs to class k. Thus, the Gini impurity criterion encourages the

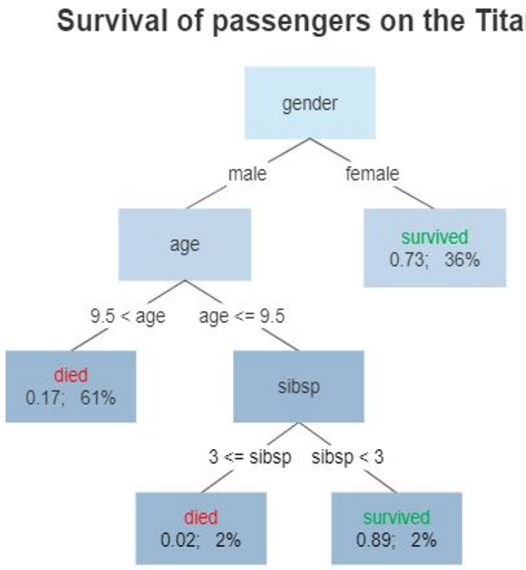

Figure 2.5: **A decision tree**. A tree showing survival of passengers on the Titanic, a classification task with two classes. The first node is called root, while the final nodes leaves. The figures under the leaves show the probability of survival and the percentage of observations in the leaf. Image taken from [60].

formation of nodes with the majority of samples belonging to one class. Taking as reference Figure 2.5, starting from the root, a tree is created by iteratively splitting the dataset and choosing the attribute that gives the lowest Gini impurity.

Decision trees are very interpretable and easy to construct, but they tend to overfit and to be not robust to outliers. For this reason, they are said to be weak classifiers. Weak classifiers can be combined into one model, forming an ensemble. It turns out that these ensemble models are more powerful, i.e. have less variance, the more the single components are uncorrelated between each other [61]. A RF is built from an ensemble of decision trees that are randomized in two ways:

1. The dataset on which each tree is trained is bootstrapped, i.e. sampled with replacement, from the original dataset. In this way, about a third of samples are left out and they constitute the "out-of-bag" samples [59]. This technique is called Bagging [62].
2. At each split, only a random subset of features is considered as a candidate for splitting a node.

This randomization is the distinguished characteristic of RF and it makes the algorithm very

effective in reducing overfitting and learning complex structures of the dataset.

2.2.3 Neural Networks

Artificial Neural Networks (ANN or just NN) have seen a popular rise recently, mainly because of the applications in deep learning. But the ideas of creating artificial networks of neurons date back to the '40s [63] and to the '50s, when the perceptron model, a simple binary classifier, was created [64].

They are very versatile models, as they can handle regression and classification problems, as well as supervised and unsupervised settings. We used them in the supervised classification task that is presented in chapter 3. Nowadays, common NN models have an architecture

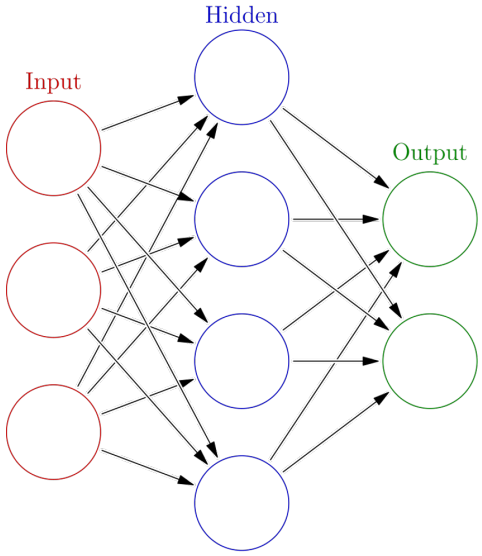

Figure 2.6: **Architecture of a 1-layer artificial neural network** In an artificial neural network the neurons of subsequent layers are connected through different weights, mimicking real synaptic connections. Image taken from [65].

similar to the one shown in Figure 2.6. This type of architecture is called feed-forward, because the information flows from the inputs to the outputs, unidirectionally. The nodes are the "neurons" that process the incoming information and the edges are called "weights", in analogy with synaptic connections in the biological brain. The parameters that a NN model has to learn are the connection weights between pairs of neurons. The neurons are organized in layers, with the first layer being the one that receives the features of the samples and usually does not apply any transformation to them. The final layer, instead, is the output layer that returns the result of the computations done by the intermediate

layers. The particular form of the output layer depends on the kind of task, for example for a classification problem it usually is a logistic or softmax layer. The layers between the input and output layers are called hidden layers and each can have a variable number neurons, or "width".

All neurons in the hidden layers receive as input a weighted sum of the outputs of the neurons in the previous layer, weighted depending on the connection weights, and outputs a non-linear transformation of the input:

$$o_i = g\left(\sum_j w_{ij} o_j\right) \tag{2.15}$$

where we used o_i to indicate the output of a neuron in a layer, o_j the outputs of the neurons in the previous layer and w_{ij} are all the weights between the neurons of the two layers. The non-linear function g can assume various analytical forms and it is called the activation function. In this thesis we used two types of activation functions: the $ReLU$ (Rectified Linear Unit) and the $tanh$ functions. The $ReLU$ has a form $g(x) = max(0, x)$ and allows to have a more efficient training [66].

NNs are trained with an algorithm called backpropagation [50], which allows to compute the derivative of the loss function w.r.t. all the weights in the NN and to use these derivatives in the SGD algorithms. We used a modified version of SGD for training the NN, called Adam [67], which allows for more efficient training by exploiting the second moments of the gradients.

2.2.4 K-means algorithm

The K-means algorithm [68] is perhaps the most used unsupervised algorithm for clustering. In K-means the number of clusters, K, is given a priori and the clusters themselves are described by a set of K vectors $\boldsymbol{\mu}_k$ with $k = 1, ..., K$, which represent the centers of the clusters. For each data point $\boldsymbol{x}_n$ a binary vector, $\boldsymbol{r}_n$, can be introduced, such that $r_{nk} = 1$ if $\boldsymbol{x}_n$ is assigned to cluster k, while $r_{nj} = 0$ if $j \neq k$.

The cost function of the K-means algorithm is:

$$J(\{\boldsymbol{\mu}_k\}, \{\boldsymbol{r}_n\}) = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}|\boldsymbol{x}_n - \boldsymbol{\mu}_k|^2 \tag{2.16}$$

The goal of the learning procedure is to find the sets of $\{\boldsymbol{r}_n\}$ and $\{\boldsymbol{\mu}_k\}$ that minimize J. In

the K-means algorithm the cost function is minimized in an iterative two-steps procedure, which is a particular case of a more general and powerful algorithm called Expectaction-Maximazation (EM) [69].

Initially, the K clusters' centers $\{\boldsymbol{\mu}_k\}$ are randomly initialized. Then, one iteration of the EM algorithm looks like this:

1. In the E-step, each data point is assigned to the nearest center, thus computing $\boldsymbol{r}_n$ for all data.
2. In the M-step, the cost function is minimized with respect to $\boldsymbol{\mu}_k$, yielding:

$$\frac{dJ}{d\boldsymbol{\mu}_k} = 2\sum_{n=1}^{N} r_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) = 0 \longrightarrow \boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n}{\sum_{n=1}^{N} r_{nk}}$$

   i.e. the mean value of all the points assigned to the k^{th} cluster.

These two steps are repeated iteratively, until some convergence criterion is reached, for example if J reaches a minimum threshold or if the parameters do not change for a certain number of iterations. The K-means algorithm is assured to converge, but it often happens that it converges to a local minimum instead of the global one. For this reason, it is common practice to re-run the K-means algorithm several times with different initializations and choose the best one in terms of minimum value of the cost function reached. It is also common to use the K-means^{++} [70] algorithm as initialization scheme, which aims to choose the initial centers in a way as to maximize the distance between each other, such that the whole dataset is well represented by these initial centers.

K-means is very fast and often very effective. On the other hand, it has a reduced representational power, because it is exact only with gaussian-shaped clusters having the same covariance matrix and produces a bad partition when the different clusters have very different sizes, shapes and densities. Moreover, it hard-assigns points to different clusters, whereas a probabilistic approach would be more appropriate in some cases.

2.2.5 Gaussian mixture model

A mixture model is in general a probability density model that can be written as $p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}|\boldsymbol{\phi}_k)$. The index k describes each component of the mixture, to which a specific set of parameters $\boldsymbol{\phi}_k$ is associated. π_k are the "weights", called mixing coefficients in this context, associated with each component, such that $\sum(\pi_k) = 1$. The specific form of $p(\boldsymbol{x}|\boldsymbol{\phi})$ determines the nature of the mixture.

If $p(\boldsymbol{x}|\boldsymbol{\phi})$ is a Gaussian distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, the model takes the name of Gaussian mixture model (GMM). A GMM is an unsupervised algorithm that can be used for density estimation or for clustering, as we shall see in chapter 4. The parameters of the GMM are the sets of $\{\boldsymbol{\mu}_k\}$, $\{\boldsymbol{\Sigma}_k\}$ and $\{\pi_k\}$.

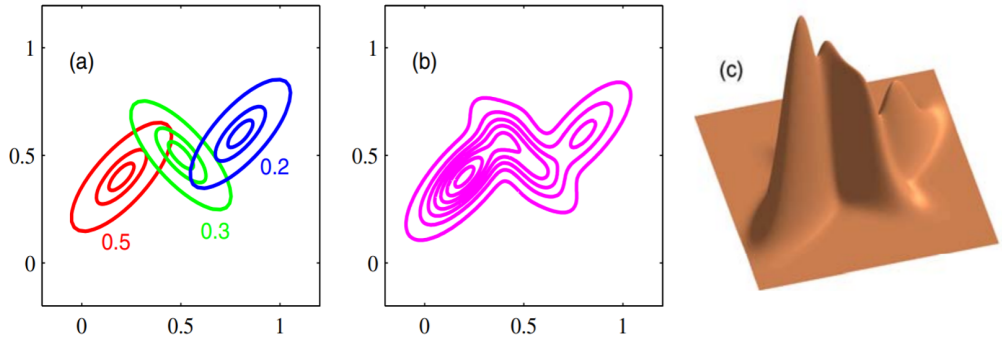

Figure 2.7: **Mixture of Gaussians**. A complicated distribution can arise from simple components. **Left**: the three Gaussian components. **Center**: resulting overall distribution. **Right**: same distribution visualized in 3D space. Image taken from [58].

The power of the GMM lies in the fact that it can represent complicated distributions by means of simple "building blocks". Figure 2.7 shows an example of three bivariate Gaussian distributions, with the single components on the left and the overall distribution on the right, which results quite complicated.

A natural way of expressing the GMM is by means of latent variables. For an observed random vector $\boldsymbol{x}$, a one-hot latent vector $\boldsymbol{z}$ is associated, such that $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$. Also the mixing coefficients can be expressed in terms of the latent vector, being $p(z_k = 1) = \pi_k$. With this formalism, the probability of the observed random variables, $p(\boldsymbol{x})$, can be rewritten by marginalizing the latent variables:

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}) \cdot p(\boldsymbol{x}|\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.17}$$

Given that $\boldsymbol{z}$ is a one-hot vector, it is possible to write the distributions as:

$$p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k} p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k} \tag{2.18}$$

for the equality to hold.

An important quantity is the posterior probability associated with the k^{th} component:

$$p(z_k = 1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|z_k = 1) \cdot p(z_k = 1)}{\sum_j \pi_j \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \equiv \gamma(z_k) \tag{2.19}$$

$\gamma(z_k)$ is called responsibility and represents the probability that the observed point belongs to the component k. For a dataset of N i.i.d. observations, the log-likelihood function is written as:

$$log(\mathscr{L}) = \sum_{n=1}^{N} log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{2.20}$$

Also the parameters of the GMM are optimized with the EM algorithm.

1. In the E-step, the responsibilities $\gamma(z_{nk})$ for each datapoint are calculated using Equation 2.19, given the current values of $\{\boldsymbol{\mu}_k\}$,$\{\boldsymbol{\Sigma}_k\}$ and $\{\pi_k\}$.
2. In the M-step, the log-likelihood is maximized w.r.t. The parameters, yielding:

$$\begin{cases} \boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n \\ \boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})^T \\ \pi_k^{new} = \frac{N_k}{N} \end{cases} \tag{2.21}$$

where $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$ and $\gamma(z_{nk})$ are the ones calculated during the E-step.

2.3 Signal processing

In this thesis work we used different types of time series signals. As mentioned in section 1.3 for the EEG, the raw signals recorded need to be processed with various techniques in order to increase the signal-to-noise ratio and to extract useful features for the classification tasks. In the following we give a brief overview of the signal processing techniques we used, that primarily deal with spectral analysis and ML preprocessing.

2.3.1 Filters

We used finite impulse response (FIR) filters to separate the original time series into different frequency bands, creating bandpass filters. This type of filters are plateau-shaped

in the frequency domain and are similar to a wavelet in the time domain, as can be seen in an example in Figure 2.8.

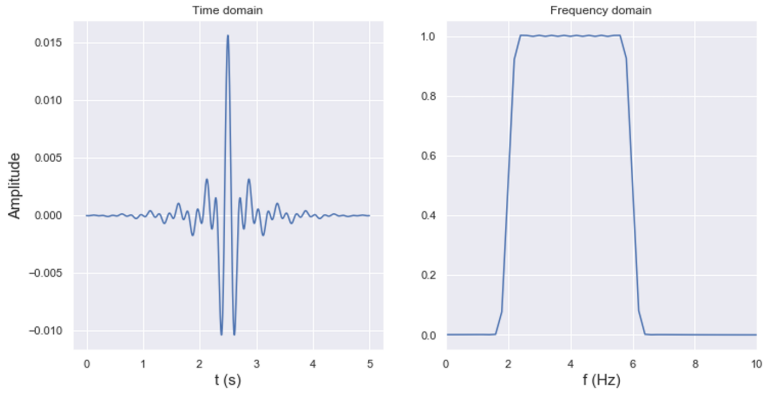

Figure 2.8: **FIR bandpass filter**. Representations in the time (left) and frequency (right) domains of a $2-6Hz$ bandpass filter. In the frequency domain the filter has a plateau shape and tapers down to zero near the boundaries. However, the profile is not sharp in order to avoid discontinuities in the profile that would otherwise create artifacts.

The filter has a gain of 1 in the region of the selected band and tapers quickly to zero outside that region, hence the name bandpass.

The width of the filter in the two domains determines the tradeoff between the temporal and the frequency precision. Since for our analyses we were more interested in having a good frequency window, we used filters more extended in time $(4-5s)$, to ensure a good representation in the frequency space.

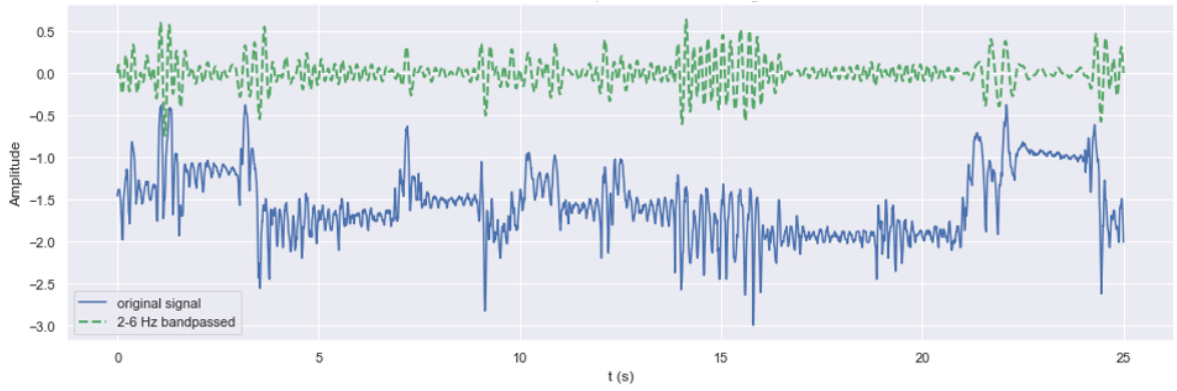

Figure 2.9: **Effects of a bandpass filter**. The green bandpassed signal retains just the frequency components specified by the bandpass and thus is centered around 0, because it has no offsets. The lower signal is slightly displaced to allow better visualisation.

When a filter is constructed, it can be applied to the signal through convolution, obtain-

ing a filtered signal containing just the selected frequency components, as in Figure 2.9. To implement the different filters in *Python*, we used the *Scipy* functions *firwin* and *filtfilt*.

2.3.2 Hilbert transform

To extract power information from a real signal, it is useful to work with its analytical representation, that is, a signal with also an imaginary part.

The Hilbert transform is a method that allows to compute the phase quadrature of a real signal and the creation of the analytical signal. The Hilbert transform is a linear operator that takes a real function as input and returns another real function as output. It has a very simple representation in the frequency domain. Given a function $u(t)$ and its Fourier transform $\hat{X}(f)$, the Hilbert transform multiplies every component of the Fourier spectrum by $-i \cdot sgn(f)$. Given the Euler representation $i = e^{i\pi/2}$, this is equivalent to a $\pm 90°$ rotation of the complex Fourier coefficients, for the negatives and positive frequencies respectively [20]. The analytical signal of $u(t)$ is then $u_a(t) = u(t) + i \cdot \mathcal{H}[u](t)$, i.e. the real part is the original signal and the imaginary part is its Hilbert transform [71]. To grasp the meaning of the Hilbert transform, it can be useful to say that the Hilbert transform of a cosine, $u(t) = cos(\omega t)$, is a sine wave, making $u_a(t) = e^{i\omega t}$. Thus, the intuitive effect of the rotations imposed by the Hilbert transform on the Fourier coefficients is to convert all the real cosine components into imaginary sine components, allowing to build a representation in the complex plane.

It turns out that the same result for the analytical signal can be obtained by zeroing out the negative-frequency components of the Fourier spectrum of the signal and doubling the positive-frequency ones. Once the analytical signal is obtained, one can then compute the instantaneous power and phase, by exploiting the analytical nature of the signal:

$$P(t) = \sqrt{Re[u_a(t)]^2 + Im[u_a(t)]^2} \tag{2.22}$$

$$\theta(t) = tan^{-1}\left(\frac{Im[u_a(t)]}{Re[u_a(t)]}\right) \tag{2.23}$$

2.3.3 Feature scaling

Feature scaling is not directly related to a transformation of the time series signals, but to the features used for the different tasks. The goal of feature scaling is to make the learning of the algorithm easier and more efficient.

There are various types of scaling, but in this thesis we used two of them.

1. Standard scaling (or standardization). It consists in rescaling the features to have mean zero and variance 1:
$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x} - mean(\boldsymbol{x})}{std(\boldsymbol{x})}$$
Standardization is widely applied in all those cases where the distribution of features is approximately Gaussian.
2. Log-scaling:
$$\hat{\boldsymbol{x}} = log(\boldsymbol{x})$$
It is very useful in those cases where the distribution is skewed. With this scaling the values near zero get "spread" in a bigger interval, while high values get shrunk. log-scaling is widely used when dealing with power features for example, because their values are usually very low and concentrated towards zero.

2.3.4 Gaussian smoothing

We also used Gaussian smoothing to smooth the features time series in the different tasks. Gaussian smoothing refers to the procedure of convolving a time series with a Gaussian kernel, such that each point in the transformed time series is a weighted mean of its neighbouring points in the original signal, with the weights decaying like a Gaussian when moving away from the point. An example of Gaussian smoothing applied to a signal is shown in Figure 2.10.

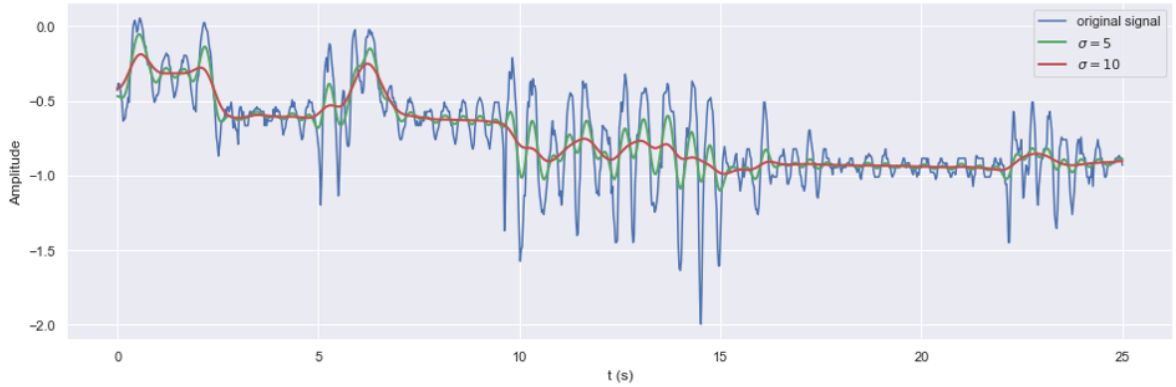

Figure 2.10: **Effects of Gaussian smoothing**. The result of the convolution between the signal and a Gaussian kernel is a smoothed version of the original signal, without higher frequency components.

It can be demonstrated that a Gaussian kernel in the time domain is Gaussian also in the frequency domain and the result of the convolution is similar to the action of a lowpass filter, thus making the resulting signal a "smoothed" (filtered) version of the original one [20].

The parameter of the Gaussian smoothing is the standard deviation, σ, of the kernel. σ is related to the number of points that are used at each step of the convolution by $w = 2\lfloor 4\sigma + 0.5 \rfloor + 1$, where w indicates the "width" of the kernel expressed in points. For example, for $\sigma = 0.5$ and $\sigma = 0.8$ the Gaussian kernel is made of 5 and 7 points, respectively.

CHAPTER 3

Supervised behavioral classification

In the following chapter, we show how we used the different analysis methods described in chapter 2 for a behavioral classification task. The final goal is to exploit the measurements of acceleration recorded by an accelerometer placed on the mice's head to classify automatically the accelerometer signals into different, predefined behaviors.

3.1 Experiments and Datasets

The datasets used for the supervised classification task came from $N = 2$ different $C57BL/6$ mice. They were males with an age at the time of the recording of about 60 weeks, with a $KI\ ApoE4$ genetic modification, meaning that they contained the human $ApoE$ gene. A total of $n = 3$ sessions were recorded at non-specific times during the day, lasting about three hours each. Therefore, a total of about ten hours of recording was available for the data analysis. Mice were recorded in an antistatic arena during spontaneous behavior. A wireless TSE Neurologger [72] device was used to continuously record both cerebral and accelerometer signals. The Neurologger has dimensions $22 \times 15 \times 5$ mm and a weight of about 2 grams, including the batteries. It possesses an integrated 3-axis accelerometer with a measurement range of $\pm 2g$ and an 8-bit ADC converter, meaning that the resolution of the accelerometer is $1/64g \approx 0.016g$. The sampling rate of the accelerometer is $102.4Hz$, that is, 512 points every 5 seconds.

The setup consisted also of an Ethovision [73] tracking camera mounted above the arena. The camera allowed us to replay the videos of the experiments and mark the timestamps

of the different behaviors as they happened. For this purpose, we made use of the software BORIS [74]. This software allows to create projects where one can upload the video files, define the behaviors of interest and mark the events of the corresponding behavior while watching the video.

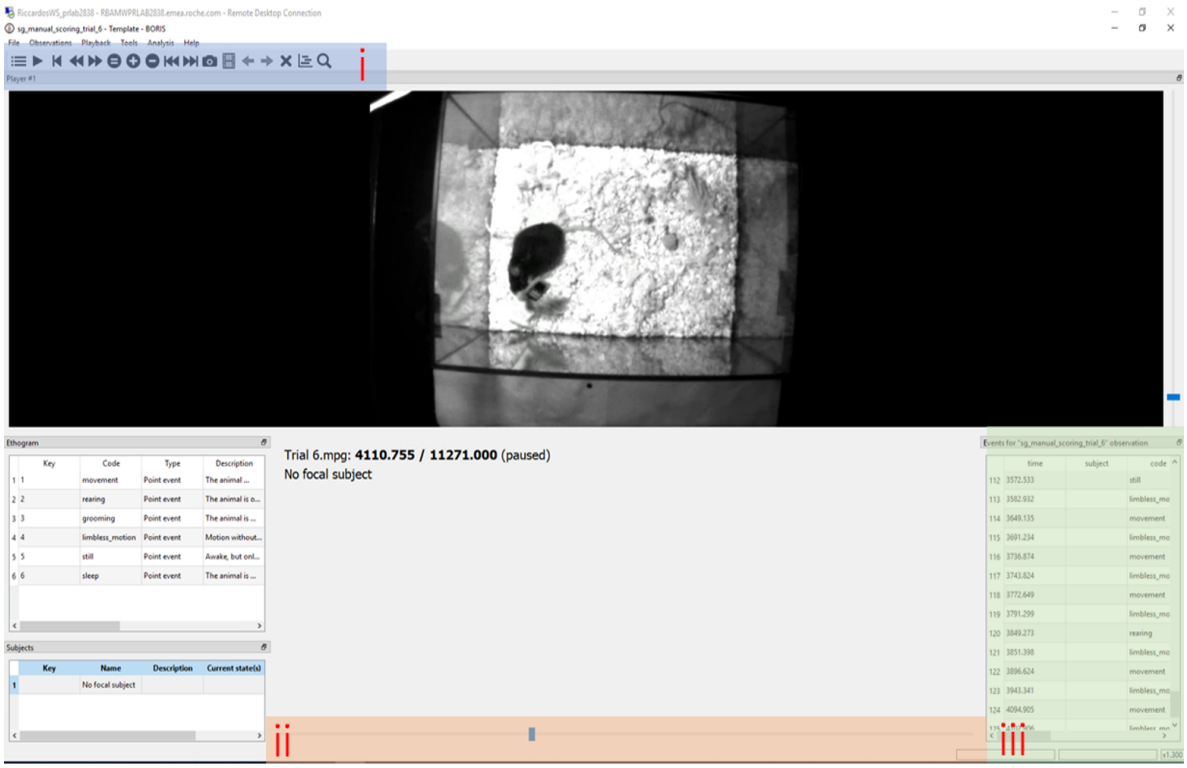

Figure 3.1: **The interface of the BORIS software.** The software BORIS was used to manually score the behaviors of mice and translate them into lables. Different colors indicate different panels. The video recording is visible at the center of the interface.

The recorded videos are played at the center of the interface (Figure 3.1), from which one can see the different behaviors, while at the sides there are various panels of interest: in the panel (i) various commands are present to control the reproduction of the video, the slider in panel (ii) allows to skip to different parts of the video and in panel (iii) the timestamps and the associated behaviors are displayed and saved. The timestamp of the beginning of a behavioral event is saved when the corresponding key is pressed. The process of translating these timestamps into usable labels for the algorithms was automatized by considering the difference of the timestamps of two successive events as the duration of the behavioral interval and taking the behavior of the first event as the label of the interval.

We were interested in four different behaviors, that we called activity, sleep, grooming and rearing. The last two behaviors are stereotypical behaviors and can be of great interest and importance for those disease models in which the activity can be impaired. For scoring the different behaviors by visual inspection, we stuck up to different rules and scored the corresponding behavior only when these rules were respected. Below is a list of the criteria used for scoring the different behaviors.

1. Sleep:
    - Associated with immobility.
    - Intervals less than 10 seconds should be disregarded.
    - If the mouse is leaned on the glass for a brief period of time, even if for longer than 10 seconds, it does not count as sleep.
    - If undecided if the mouse is sleeping while touching the glass, do not consider it as sleep.
2. Rearing:
    - The mouse stands upright on the posterior paws, either in the middle of the cage or by leaning on the glass.
3. Grooming:
    - The intervals should last at least 10 seconds.
    - The mouse should have a "ball-like" shape, with the head bended downwards.
    - The mouse is actively shaking the head and moving the paws.
    - No movement of the bedding should be present, to be sure the mouse is not digging.

Whatever did not fit with any of the above indications was considered as "activity". Therefore, activity does not have a precise definition, but rather, is considered as a variegate behavior in which the mouse is not performing the other three. It could include digging, sniffing, nesting, etc.

3.2 Accelerometer working principles

In this section we illustrate the working principles of the accelerometer, in order to better understand and describe the recorded signals.
The accelerometer works on basic physical principles to sense static and dynamic linear

accelerations. The static component is due to the constant gravitational pulling of the Earth, while the dynamic one is due to external forces applied on the accelerometer. In our case, the dynamic acceleration component represents the external force applied by the mouse head on the accelerometer when moving. Both these types of acceleration yield important information and can be exploited to characterize movements in different kinds of behaviors.

Modern accelerometers are called MEMS, which stands for Micro ElectroMechanical Systems [75]. As the name suggests, they are based on miniaturized mechanical parts, integrated with electronic components [76]. These mechanical parts have a dimension of the order of the micrometers and are exploited to make the sensors small enough to be integrated in many electronic devices. The core part of an accelerometer is made by a combination

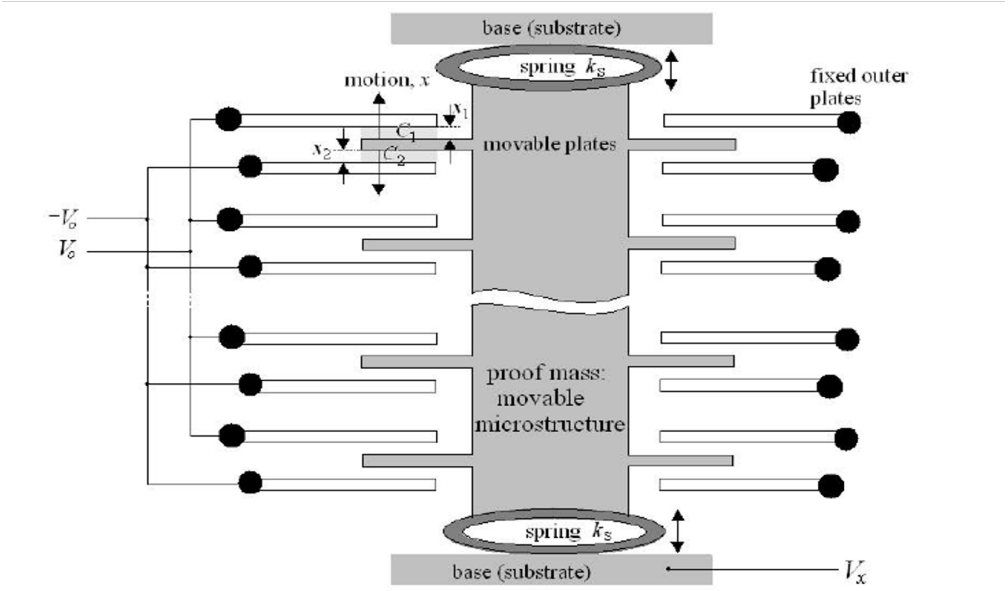

Figure 3.2: **Main components of an accelerometer**. A MEMS accelerometer is made of mechanical components, like masses and springs, and electrical ones, as circuits and capacitances. Image taken from [77].

of a movable mass with plates attached to its sides and fixed plates connected to a circuit acting as capacitors and kept at constant voltage V_0, as can be seen in Figure 3.2. The mass is movable because it is attached to a fixed substrate through springs. The capacitances are created between each plate of the movable mass and the corresponding negative and positive fixed plates, examples being C_1 and C_2 in Figure 3.2.

The whole system exploits a change in capacitance due to the movement of the plates

to produce an electrical signal. The presence of many identical repeating structures ensures that this change in capacitance is big enough to be detected, since the individual changes are very small.

A signal is registered whenever a force with a component perpendicular to the direction of the plates acts on the device. For example, let us suppose that a constant downward acceleration is acting on the accelerometer. Then, the movable mass and plates are pushed downward. Referencing to Figure 3.2, this makes the distances of the movable plates from the fixed plates different, in particular decreasing x_2 and increasing x_1. This, in turn, causes a change in the total capacitance of the circuit, because the capacitance depends on the distance between the plates via the relationship $C = \epsilon A/d$, with ϵ being the dielectric constant, A the area of the plates and d their distance [77]. If d is the initial distance between the plates, after a displacement Δx the new distances are $x_1 = d + \Delta x$ and $x_2 = d - \Delta x$, thus the capacitances:

$$\begin{cases} C_1 = \dfrac{\epsilon A}{x_1} = \dfrac{\epsilon A}{d + \Delta x} \\ C_2 = \dfrac{\epsilon A}{x_2} = \dfrac{\epsilon A}{d - \Delta x} \end{cases} \implies C_2 - C_1 = \Delta C = 2\frac{\epsilon A \Delta x}{d^2 + \Delta x^2} \tag{3.1}$$

The resulting second order equation is

$$\Delta x^2 \Delta C + 2\epsilon A \Delta x - d^2 \Delta C = 0$$

and it can be solved to find the value of the displacement corresponding to a measured capacitance change of ΔC. A simplified solution can be found by assuming that the displacement Δx is very small, thus omitting the term in Δx^2.

$$\Delta x \approx \frac{d^2}{2\epsilon A}\Delta C \tag{3.2}$$

Therefore the total displacement of the movable mass is directly proportional to the change in capacitance registered by the circuit [77].

We now need to link these quantities to the acceleration experienced by the mass and the corresponding voltage outputted by the system. The first is readily obtained from the displacement by considering that the acceleration on the springs is the usual $a = k\Delta x/m$, with k the elastic constant of the springs and m the mass of the movable mass. For what concerns the output voltage V, this is measured as the voltage of the substrate and the

movable mass. For the principle of conservation of charge, it must hold true that

$$(V_0 - V)C_1 = (V_0 + V)C_2 \implies V = \frac{C_2 - C_1}{C_2 + C_1}V_0 = \frac{\Delta x}{d}V_0 \tag{3.3}$$

The last relation is obtained by using again Equation 3.1.
Putting everything together, one obtains that the output voltage of the system is proportional to the acceleration sensed by the accelerometer:

$$V = \frac{mV_0}{kd}a \tag{3.4}$$

Since the construction variables are known, it is possible to invert the equation and use a measure of voltage to compute the corresponding acceleration.

To sense any kind of acceleration, three-axis accelerometers have three different masses with plates perpendicular to each other [78], forming an orthogonal basis for the space, as in Figure 3.3 on the left.

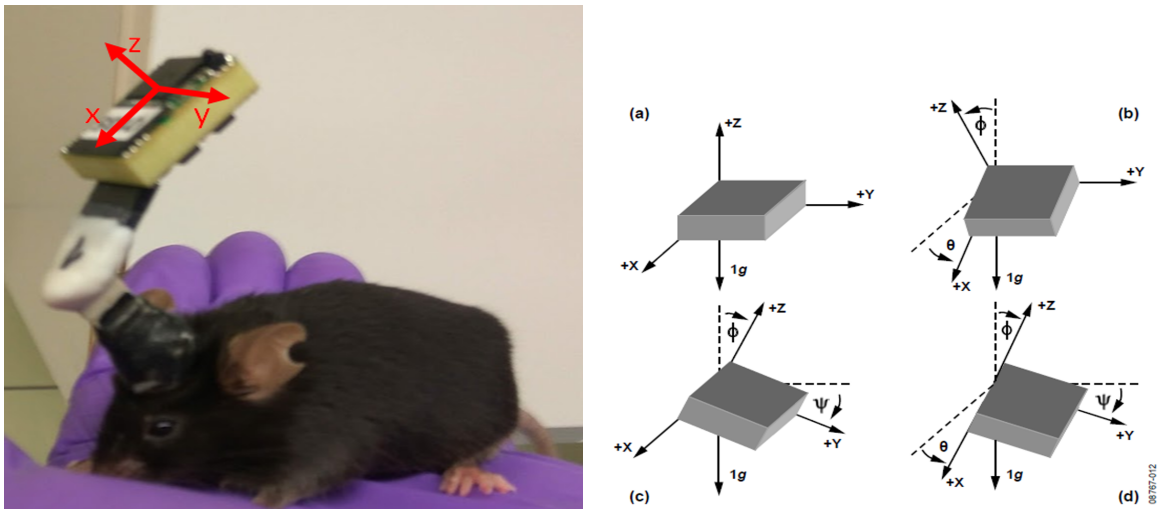

Figure 3.3: . **Left:** Neurologger headstage mounted on the mouse's head. The red arrows indicate the directions sensed by the accelerometer, denoted by x,y and z. **Right:** Different rotational angles of the accelerometer in space. Image taken from [79].

One important information that can be extracted from the acceleration measured on three perpendicular axes is the orientation of the accelerometer in space.
Taking as a reference the axes in Figure 3.3 on the right, and comparing them with the ones in Figure 3.3 on the left, the z axis is antiparallel to the gravity vector when the Neurologger lays on a plane, i.e. when the acceleration components on x and y are zero.

To extract the values of the angles as depicted in the figure, it is necessary to know the value of the acceleration along the three different components. We are interested in the angles θ and ψ. The former describes rotational displacements from the x axis and is insensible to rotations around it; on the other hand, ψ describes rotational displacements from the y axis, thus measuring rotations around the x and z axes. We are interested only in these two angles because, as depicted in Figure 3.3, the position of the headstage is constrained and these two angles are already able to sufficiently describe the majority of movements performed by the mice.

The two angles can be calculated by identifying the component of the gravity vector parallel to the axis and the one laying on the plane perpendicular to the axis. The former is the static acceleration registered on the x-axis and the y-axis for θ and ψ, respectively. The latter, instead, is calculated with Pythagora's theorem and corresponds to $\sqrt{a_y^2 + a_z^2}$ for θ and $\sqrt{a_x^2 + a_z^2}$ for ψ. Then, the values of the angles are found by taking the ratio between the two components and applying the inverse tangent function:

$$\begin{aligned} \theta &= \tan^{-1}\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \\ \psi &= \tan^{-1}\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right) \end{aligned} \tag{3.5}$$

The computational implementation of the inverse tangent in *Numpy* allows to have angles in the interval $[-\pi, \pi]$, as it selects the right quadrant based on the value of the numerator of the fraction.

3.3 Results

In this section we proceed to present the results of the behavioral analysis with the application of supervised ML algorithms. The first subsection is dedicated to the problem of separating activity from sleep, while the second subsection deals with more complex signals of grooming and rearing. We put some emphasis on the first analysis because it also outlines the procedure that we adopted in the other analyses.

This procedure is in general composed of the following steps:

1. Each signal's time series was divided into epochs of equal length.
2. For each epoch, different features relevant for the task were extracted.

3. These features were pooled together to form a design matrix, upon which the algorithms were trained.

Of course, the features extracted and the algorithms used changed from task to task, but this procedure formed the "backbone" of every analysis.

As outlined in section 3.1, we used $n = 3$ different sessions coming from $N = 2$ mice. Throughout the analyses in this chapter, we mixed the datasets before the random training/test splitting, regardless of the identity of the animal.

3.3.1 Classification of activity vs sleep

Activity and sleep are represented in the accelerometer signals by quite different patterns, so the task of separating them was a relatively easy one. Nevertheless, we expose this analysis in quite detail to better outline the procedure adopted throughout the other, more complex, analyses. Activity in this task incorporates also grooming and rearing, forming a unique class.

As a first step, we wanted to visually observe the differences of the accelerometer signals when activity or sleep was detected.

Figure 3.4: **Accelerometer signals during activity and sleep**. Each row indicates a different accelerometer axis. The two columns represent an example of activity epoch and one of sleep.

On the basis of the example signals in Figure 3.4, it is quite clear that we had to search for features that quantify the "degree of movement" present in the signals. One such feature is the power of the signal, which we extracted using the Hilbert transform described in subsection 2.3.2, using a bandpass filter from 0.5 to 15 Hz. We called this feature P and

we applied a log-scaling before using it, in order to reduce the skewness of the resulting distribution.

Another feature that can be very helpful in this context is the quantification of the maximum "displacement" registered by the three channels. In section 3.2, we discussed that the accelerometer measures both the Earth's gravitational pull and the acceleration of the mouse's head. If only the latter was measured, the acceleration on the three axes would have an average value of zero, because mice moved their head in different directions. The effect of the presence of the constant gravity vector is to add an "offset", i.e. a static component, to the signals that varies depending on the orientation of the mouse's head. Thus, even if the accelerometer does not directly measure the actual distance traveled by the head, we can get a measure of how much the mouse has moved by computing the difference between the "highest" acceleration and the "lowest" one. This feature can be useful in situations where the signal is not oscillating much, having relatively low power, but still varies from a lower value to a higher one or vice versa. For example, in Figure 3.4, on the activity signals around 1316 seconds, the signal has low power, but still exhibits an appreciable displacement which can be picked up by this feature. We decided to call it Δ and we computed it by taking the difference between the maximum and minimum values of the signals in each channel and then taking the maximum between the three channels:

$$\Delta = max(max(x) - min(x), max(y) - min(y), max(z) - min(z)) \tag{3.6}$$

The signals of each session were divided into epochs of 5 seconds each. For each epoch we then extracted the power P and the displacement Δ. After pooling all the epochs together, we ended up with 7302 samples and we split them into 70% training set and 30% test set, randomly and independently from the subject. The final shape of the design matrix for the training set was $(5111, 2)$ and it contained quite balanced classes, with 56.8% of activity and 43.2% of sleep. In the test set the classes were distributed as 55.8% of activity and 44.2% of sleep. It is very important that the proportions of the classes are quite the same between the training and the test set, since the former should represent as good as possible the entire population.

We finally applied a standard scaling to bring the two features on the same scale.

As can be seen in Figure 3.5, the standard scaling preserves the shape of the distributions while making the two features comparable in magnitude. As we expected, the overall distribution for the two features is bimodal and the class-conditional distributions $P(\text{feature}|\text{class})$

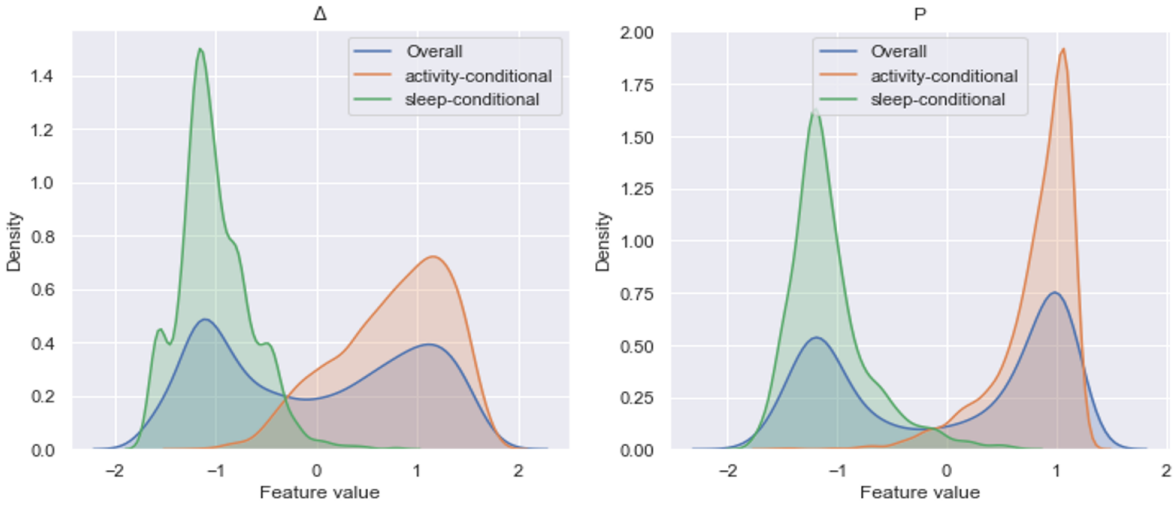

Figure 3.5: **Distributions of features after the scaling**. Overall and class-conditional distributions of the two features used to classify activity and sleep after standard scaling. The class-conditional distributions are approximately Gaussian.

are quite separated, especially for the power feature. Some overlap still remains, though, and it is difficult to say a-priori if it is due a poor choice of features or to the fact that the two classes are intrinsically non-separable.

From Figure 3.6, it is possible to see that there is some overlap in the region where both scaled features are zero, but the number of points in the overlapping region is much less than the number of points in the two very dense green and red regions. There are also some horizontal "stripes" in the green cluster, probably due to the sensitivity of the accelerometer, that introduces "quantized" values of the Δ feature.

We also found that the overlapping region contained many epochs that were at the transition between activity and sleep or vice versa. This happened because, when we scored the videos, as discussed in section 3.1, we marked the behavioral timestamps when they happened, without taking into account the successive split into epochs. Then, some epochs contained a mixture of the two states. To improve the classification, we also computed a weight for each epoch, expressed as the percentage of that epoch belonging to the assigned label. For example, if an epoch of 5 seconds contained 3 seconds of sleep and 2 seconds of activity, the assigned weight would be 0.6, since its features represent sleep just for that percentage. The majority of epochs had weight 1 and only a minority that included the transitions had lower values. With the introduction of the weights, the classification performance depends less on the transition epochs and the shapes of the more dense clusters

Figure 3.6: **Scatter plot of the training epochs**. All the training epochs are represented in the plane, in which each point corresponds to an epoch and the two axes correspond to the two features. The colors designate the two classes.

are better modelled.

From Figure 3.5, we knew that the class-conditional probabilities are approximately Gaussian and we decided to use the Logistic Regression algorithm described in section 2.2, knowing that this choice is well justified and produces a posterior probability distribution that aligns well with the true generating one. To assign labels from the probability outputs of the LR model, we used the conventional probability threshold of 0.5.

We then performed a Grid Search CV with 5 folds, as described in subsection 2.1.2, in order to find the best hyperparameters values. The most important hyperparameter to set for the LR classification is the regularization parameter, λ. We decided to use an L_2, or Ridge, regularization term, which takes into account the squared magnitude of the weights. The LR algorithm was optimized using the SGD algorithm presented in subsection 2.1.3, with a decreasing learning rate going as $\sim 1/n$, with n being the number of complete iterations of the dataset performed. We used the balanced accuracy metric to measure the performance of the classifier, to avoid any possible bias towards the majority class.

Table 3.1 shows the values of the balanced accuracy we obtained for the different values of λ. The best value is found for $\lambda = 10^{-4}$ and, although it is a quite small value, it is

Rank	λ	Balanced Accuracy
1	10^{-4}	0.9733
2	10^{-3}	0.9732
3	10^{-5}	0.9711

Table 3.1: Best regularization parameter λ found using the grid search CV procedure.

located in the middle of the grid, sign that it is an optimum of the hyperparameter.

We then proceeded to train again the algorithm using the whole training set and the optimal hyperparameter found. To assess the final performance of the classifier, we computed the confusion matrices, both for the training and the test set.

As explained in subsection 2.1.4, the anti-diagonal terms of the confusion matrix contain

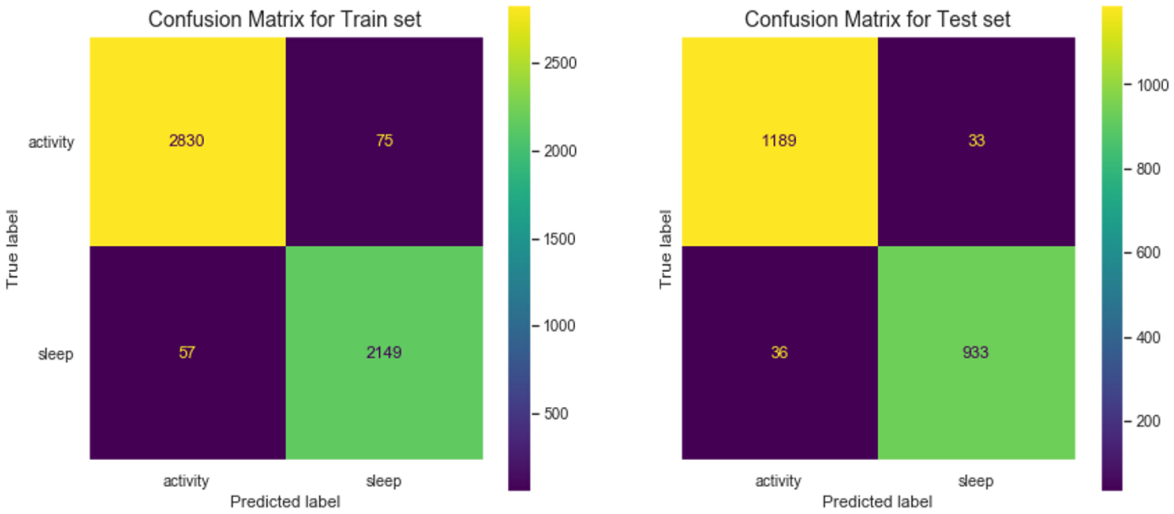

Figure 3.7: **Confusion matrices for the activity/inactivity classification**. Confusion matrices summarize the results of the classification. **Left:** confusion matrix of the training set. **Right:** confusion matrix of the test set.

the number of misclassified samples, i.e. activity epochs classified as sleep and vice versa on the upper right corner and lower left corner, respectively. The output of a classification can generally be considered good when the off-diagonal terms are much lower than the diagonal ones, because this means that the majority of the samples has been correctly classified. This is definitely the case for both matrices in Figure 3.7, but quantitative measures are needed to assess the "goodness" of the final classification. We used the metrics described in subsection 2.1.4 and computed them for both confusion matrices.

The values of precision, recall and F1-score on the training and the test sets give different

Metric	Activity		Sleep	
	Train	Test	Train	Test
Precision	0.974	0.973	0.970	0.966
Recall	0.981	0.972	0.971	0.972
F1-score	0.977	0.973	0.970	0.969

Table 3.2: Summary results of the metrics in the training and test sets for the classification of activity and sleep.

information about the properties of the trained classifier.
The metrics on the training set give a measure of the bias error, i.e. the error we would have obtained by having infinite samples, while the metrics on the test set relate to the variance error, i.e. how good the generalization properties of the algorithm are.

The balanced accuracy on the training set is 0.973 and all the different metrics are above 0.97, which can be considered a quite high value and shows the robustness of the classification.
What emerges, also, from these values is that, with these dataset and algorithm, the best possible error achievable is about $2-3\%$. This represents the maximum capacity (complexity) of the algorithm trained with the two features Δ and P.
The balanced accuracy on the test set is instead 0.968. All the metrics are very similar to the ones of the training set, except 1% less in the sleep precision and activity recall. The similarity between these metrics and the ones of the training set let us conclude that the generalization properties of the algorithm are very good. This is probably due to the fact that we just used two very representative features and a fairly simple algorithm, preventing it from severe overfitting. This can be considered a very good result, also in light of the fact that the complete procedure of feature extraction and subsequent classification is very fast.

To have a better understanding of which epochs were misclassified, we plotted the test samples on top of the decision boundary of the algorithm.
In Figure 3.8, every point in the light red portion of the plane is classified as activity by the algorithm, while every point in the light green portion is classified as sleep. The red points in the light green portion of the plan are activity epochs that were incorrectly classified as sleep, while the vice versa is true for the green triangles in the light red region. As we expected, the major source of misclassification happens around the center of the plane, where the two classes are not well defined.

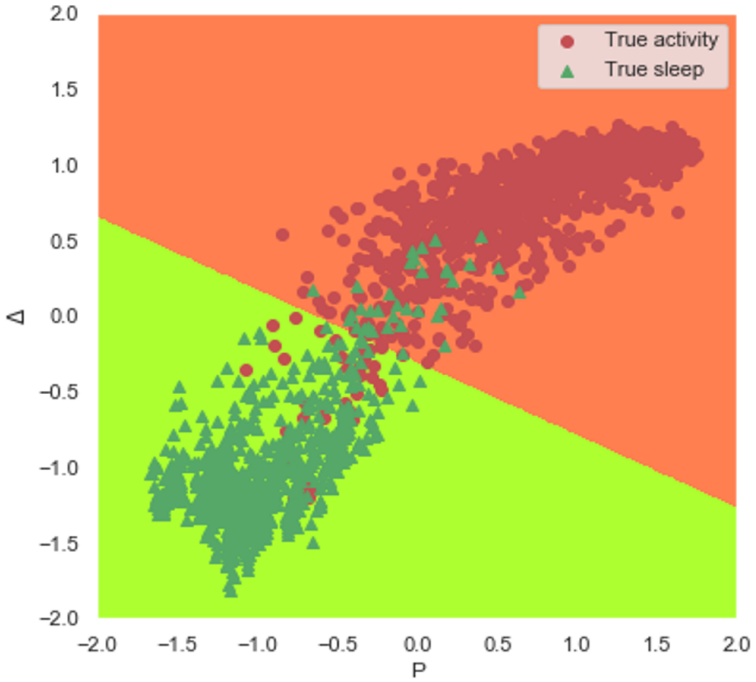

Figure 3.8: **Decision boundary learned by the classifier**. The points represent test samples, colored by the true label. Colored surfaces represent decision regions learned by the algorithm, colored based on the prediction.

To get a sense of what these misclassified epochs look like, we took three examples of each case and plotted the raw signals.
In the left column of Figure 3.9, there are three examples of activity epochs that were classified as sleep. We notice that the signals of the three axes of the accelerometer are quite flat, with only a small amount of activity and oscillations. These types of epochs probably correspond to short twitches during sleep or to the beginning of sleep.
On the other hand, the right column of Figure 3.9 shows three examples of sleep epochs classified as activity. Here, instead, the activity in the accelerometer signals is very high and not flat as one would expect during sleep. These epochs probably correspond to short awakenings during sleep or to transitions from sleep to wake or vice versa, as is pretty evident from the second plot, where the signal is initially flat and then starts to oscillate.

Therefore, already from this qualitative analysis we can conclude that the misclassified epochs belong to gray areas, where it is difficult to assign a definite label. Indeed, this analysis also shows that the "ground truth" provided with the labels is not errorless, on the contrary, it may contain misclassification itself. From this perspective, it is a good thing that the algorithm did not reach 100% accuracy, because this would have meant that it just

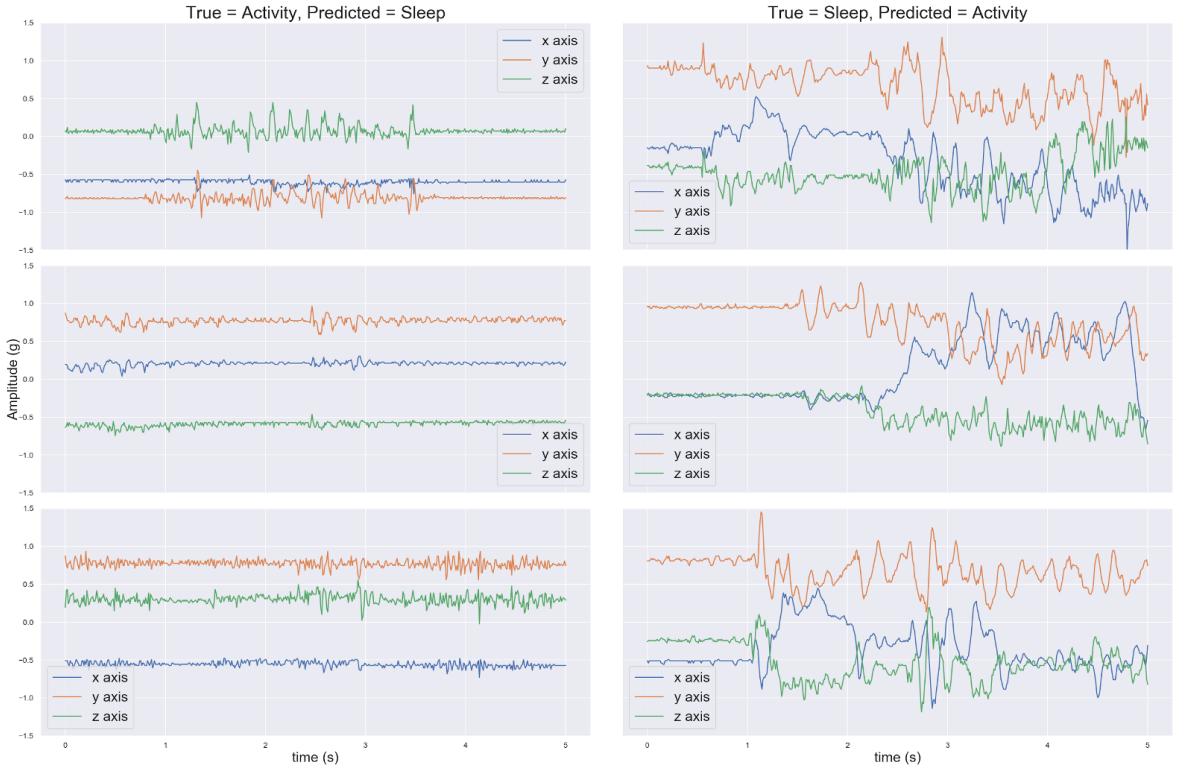

Figure 3.9: **Examples of misclassified epochs**. Each of the six plots represents an epoch that has been classified with the wrong label and contains the three accelerometer channels.

learned to replicate the labels provided, including the examples in Figure 3.9, which are not correctly labelled.

3.3.2 Classification of more complex behaviors

We now turn our attention to the classification of more complex and stereotypical behaviors, such as grooming and rearing. We already defined the characteristics of these behaviors in section 3.1.

Figure 3.10 left highlights that grooming can often exhibit strong oscillations, accompanied by some occasional interruptions and twitches. Rearing, Figure 3.10 right, exhibits instead strong and sudden variations in the values of the signals, particularly visible in the first axis, which is the most affected given the way the accelerometer is placed on the mice's head. We also notice that rearing events can be quite noisy and present strong activity in the accelerometer's channels.

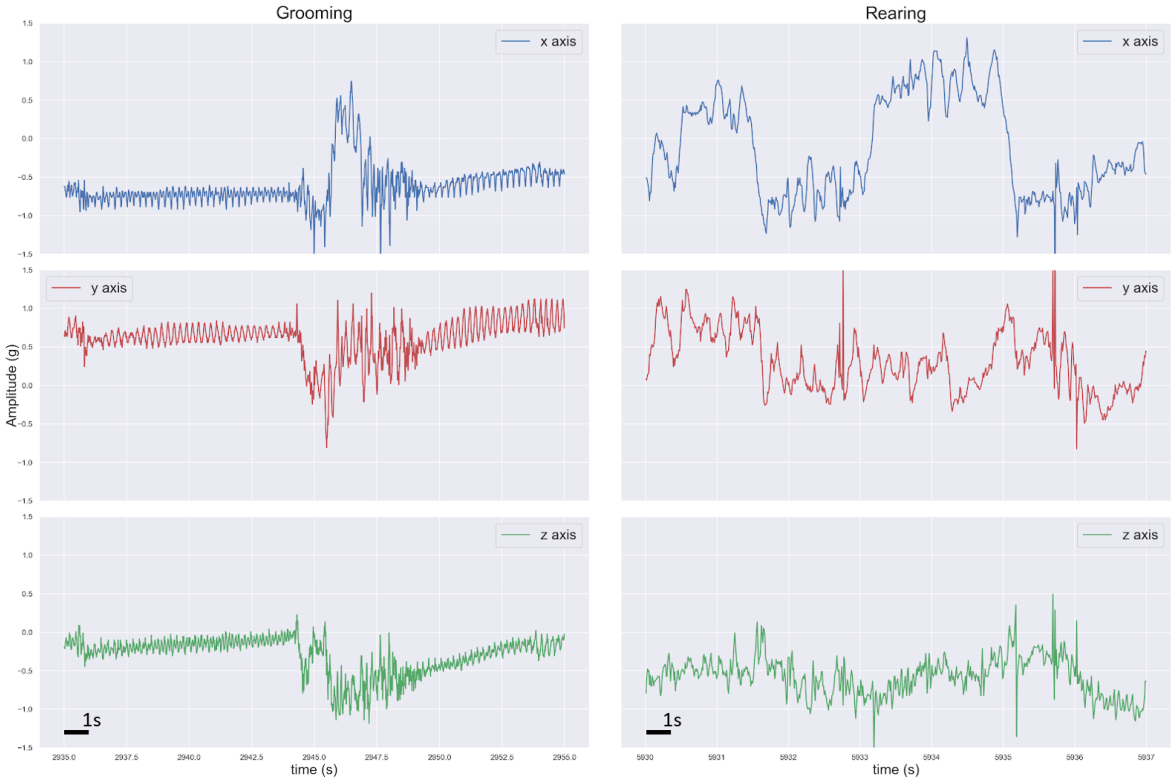

Figure 3.10: **Example accelerometer signals during rearing and grooming**. Each row of the plots represent a different axis of the accelerometer and the two columns display the signals during grooming (left) and rearing (right).

These two additional behaviors are quite challenging to classify, for various reasons. On one hand, a number of grooming events actually shows reduced activity and strength in cleaning, rubbing, licking etc. This translates into accelerometer signals that are quite flat and do not possess the characteristic oscillation previously mentioned.
On the other hand, rearing events are very difficult to characterize for at least two reasons. The first one is that they do not qualitatively differ much from events in which mice simply raise their head, even without standing on their paws. The second reason is given by the duration of these events. While grooming, general activity and sleep are always types of behaviors that last in time at least 20 or 30 seconds, if not minutes, rearing is instead a very rapid type of event that lasts not more than 3 seconds. This made the precise marking of these events from the video quite difficult, such that the intervals produced often had imprecise timestamps, affecting the resulting quality of the labels. Another problem caused by the rearing events being very short is that, even if their absolute number is high, their total duration is quite low and they are very fragmented, resulting in a relatively low number

of epochs.

To mitigate these problems we adopted two solutions.

First, we used overlapping epochs, instead of non-overlapping ones. Overlapping epochs have been previously applied to ML problems, leading to improved results [80]. The use of overlapping windows allows for a better localization in time for the rearing events, allowing to have the maximum possible number of rearing epochs, without losing any. The parameter to consider when creating overlapping epochs is the amount of overlap between consecutive epochs. It depends on the precise problem, and different values have already been used [81][82]. It is clear that a value too high might result in the epochs being too similar, while a value too low might result in the procedure being useless. For this task, we decided to use epochs of 4 seconds with an overlap of 2 seconds.

The second way we adopted to mitigate the problem of having a small number of rearing epochs was by augmenting them. Data augmentation is a general procedure in ML with the main purposes of increasing the amount of data when this is too low and to make the algorithms more robust and less prone to overfitting [83]. We decided to use a fairly simple method for data augmentation for time series, adopted in [84]. This method consisted in the addition of Gaussian noise to the original features of the epochs:

$$\mathbf{x_{new}} = \mathbf{x} + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim N(0, \mathbb{I}_m) \tag{3.7}$$

$\mathbf{x_{new}}$ is a new sample augmented from $\mathbf{x}$, σ is the noise strength and ϵ is a random vector normally distributed, with mean zero and covariance matrix equal to the $m \times m$ identity matrix, with m being the number of features. The important point here is that the noise is directly added to the scaled features, not the signals themselves. For this data augmentation technique, one has to choose how many times to repeat the augmentation, M, and the noise strength ϵ. It is critical to choose these values correctly, otherwise the whole learning process can be compromised. Also importantly, the data augmentation process was carried out just on the samples of the training set, in order to not use samples augmented from the test ones in the training process.

In this task, we used several additional features, relative to the previous classification. We selected these features based on what is currently used in activity recognition problems, on the specific characteristics of the behaviors and on empirical considerations. We used again the two features used in the previous analysis, Δ and P. Moreover, we also defined three different frequency bands for which we calculated the power, using again the Hilbert

transform in combination with FIR bandpass filters. The chosen frequency bands were $1-3Hz$, $3-7Hz$ and $7-15Hz$. The choice of these specific values for the bands was made to cover the frequency spectrum in a logarithmic manner and based on some suggestions that grooming might involve the activity at a frequency around $6Hz$ [44].
We also used statistical descriptors of the signals, namely, the mean and standard deviation of each axis and the three cross-correlations between the axes. The cross-correlation between two stationary time series x_t and y_t is defined as:

$$corr(x_t, y_t; \Delta t) = \gamma(x_t, y_t; \Delta t) = \mathbb{E}[(x_t - \mu_x) \cdot (y_{t+\Delta t} - \mu_y)] \tag{3.8}$$

and it is a function of Δt, which is called lag. A lag of zero means that the correlation is computed with values of time series at the same time point. The cross-correlation can be very useful to quantify how much the changes in the different axes are coordinated or independent. We used the cross-correlations values between each pair of axes with zero lag. Finally, we also computed the orientation of the headstage in the space, as described in section 3.2, by means of the angles θ and ψ. The information provided by the angles can be very useful to identify patterns in the position of the mice's head, for example in behaviors like grooming and sleep. To extract only the static component from the accelerometer signals, we used a FIR low-pass filter with a cutoff frequency at $1Hz$.

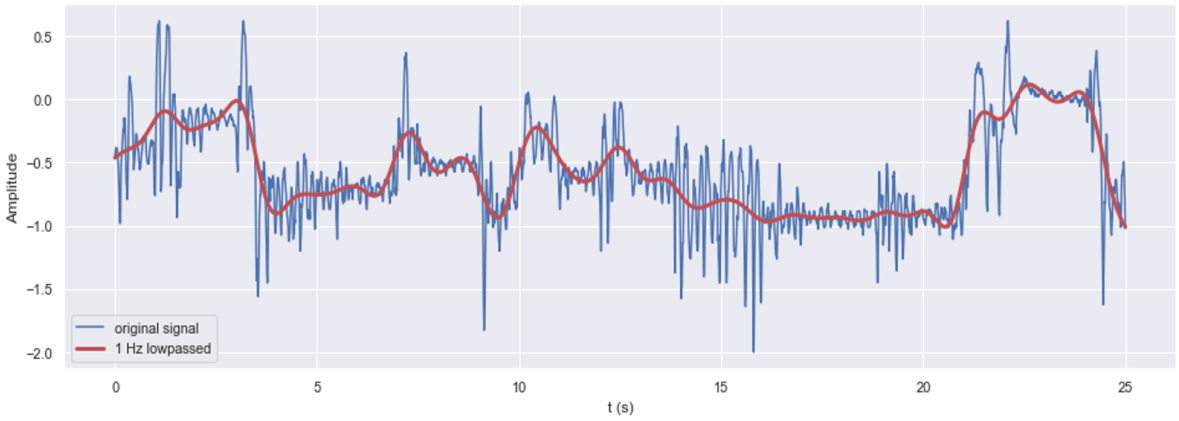

Figure 3.11: **Low-pass filter used to extract angles**. Example of the effect of the 1 Hz low-pass filter. Blue is the original signal, red is the low-passed one, representing the static, gravitational, component.

In Figure 3.11, we plotted an example of signal in one axis and the resulting low-passed signal, which represents a sort of "average" around which additional oscillations happen.

The final values of the angles in an epoch were computed as the mean values of the angles computed for each timepoint within the epoch.

The total number of features extracted for this task was 18. Visualizing them in the features space is impossible, but it is possible to get a sense of how a subset of them is distributed by using a pairplot. In Figure 3.12 we represent a subset of just five features.

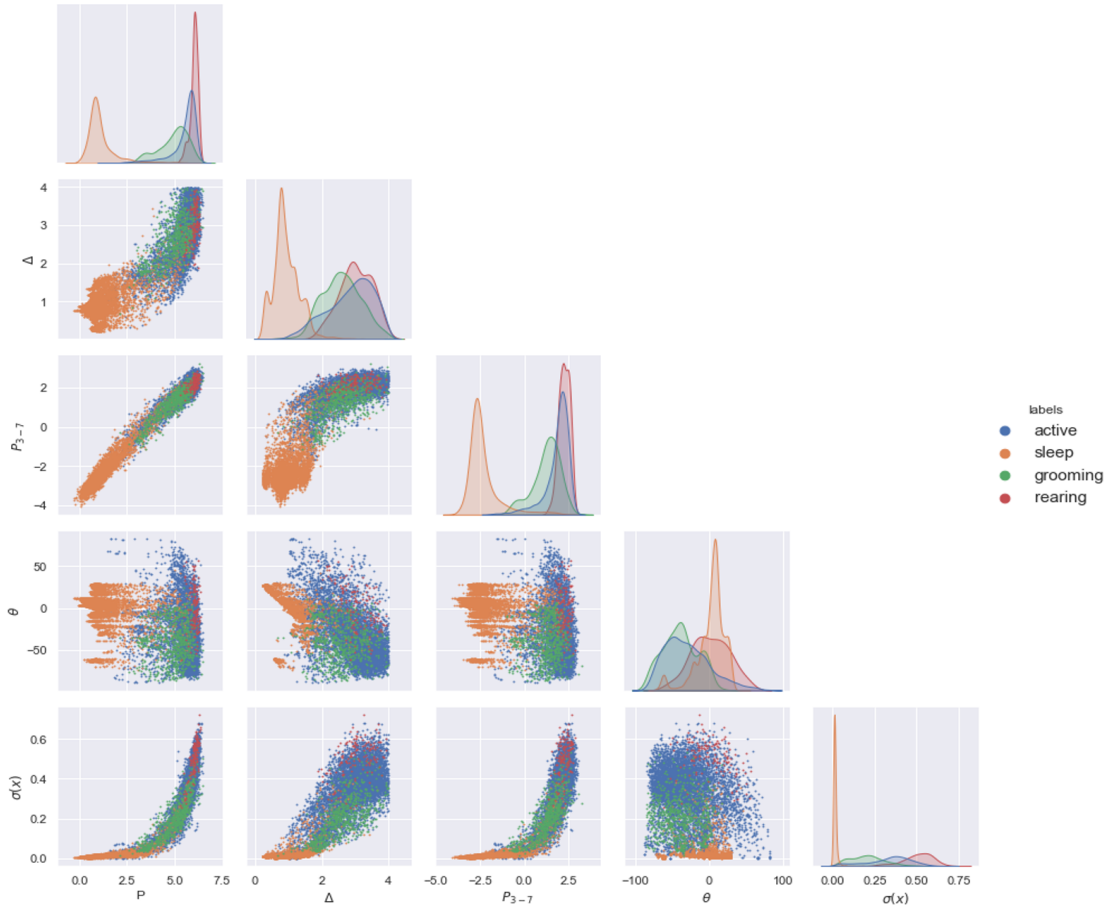

Figure 3.12: **Pairplot showing relations between features**. Represented are a sample of five features, namely, the total power P, the displacement Δ, the power in $3-7$ Hz, the θ angle and the standard deviation of the x-axis, $\sigma(x)$. The diagonal plots display KDEs of the distributions of features, while off-diagonal scatter plots show the pairwise relationships between features. Different colors represent different classes.

In the class-conditional distributions on the diagonal, the one representing sleep is always very different from the other ones, except in the case of θ. Also, discriminative features for grooming seem to be the total power, the power in the window $3-7Hz$ and the standard deviation of the fist axis, while for rearing are θ and again the standard deviation. The

low proportion of rearing epochs can be noticed by the low amount of red points in the off-diagonal scatter plots.

We then extracted a total of 18253 overlapping epochs and we used again a random $70/30$ splitting to create training and test datasets. The proportions of the different classes in the training set are in Table 3.3.

Behavior	Proportion %
Activity	49.6
Sleep	43.7
Grooming	5.6
Rearing	1.0

Table 3.3: Proportions of the four different classes in the non-augmented train dataset.

The proportions of sleep and activity are high, but grooming and even more rearing have very low proportions.

We started by applying the data augmentation procedure previously described to the rearing epochs. The effects of two augmentations with different parameters M and σ are plotted in Figure 3.13.

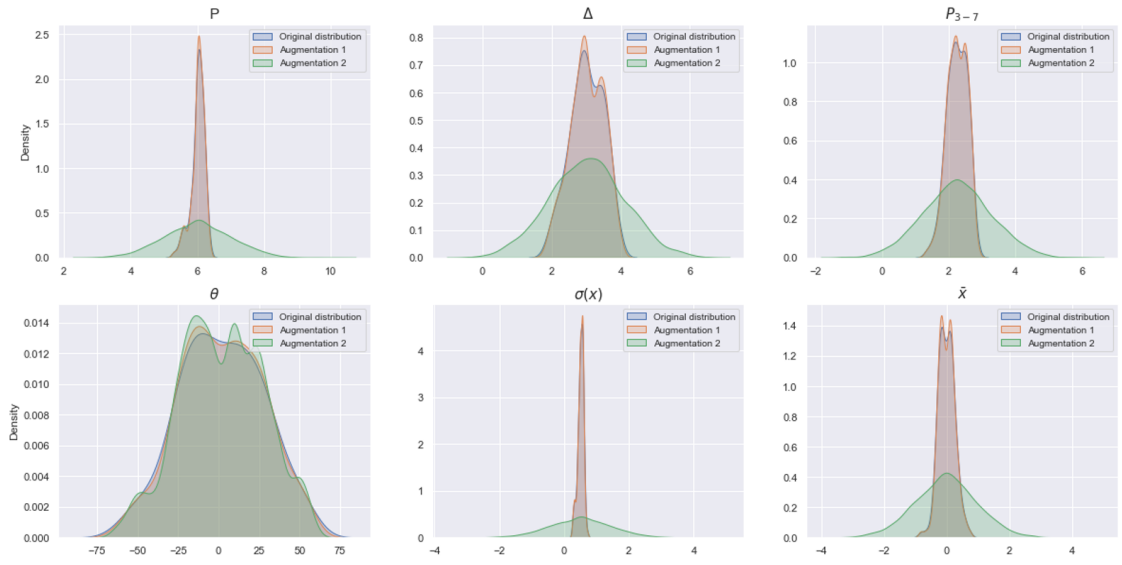

Figure 3.13: **Effect of different augmentation parameters on the features distributions**. Distributions of six features before the augmentation (blue), with small augmentation parameters $M = 2$ and $\sigma = 10^{-4}$ (orange) and with high augmentation parameters $M = 30$ and $\sigma = 1$ (green).

The features are the same ones plotted in Figure 3.12, with the addition of the mean of the x-axis. The parameters used for the first augmentation were $\sigma = 10^{-4}$ and $M = 2$, while for the second augmentation we used $\sigma = 1$ and $M = 30$. It is clear that the second augmentation distorts the original distributions much more, since the two parameters are higher and the distributions have spread due to the high noise and high number of samples. It is thus crucial to choose parameters with high enough values as to make the augmentation effective, but not so high to cause a distortion of the original distribution, as in Figure 3.13.

The choice of these parameters was done with the idea of keeping the augmented distribution as similar as possible to the original one. A way of measuring the "distance" between two distributions is the Kullback-Leibler divergence, defined between the reference distribution p and the test distribution q, which we already presented in subsection 2.1.1:

$$D_{KL}(p||q) = \int p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) d\boldsymbol{x} = \sum p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) \tag{3.9}$$

It is the expectation value over p of the logarithm of p/q. In our case, p represented the original reference distribution, while q the augmented one with different augmentation parameters. The KL divergence is zero when the two distributions are the same and increases its value as the two distributions are more and more different.

For the numerical calculation of the KL divergence we made an approximation. In principle, the integral, or summation, in Equation 3.9 should be computed for the distribution in the 18-dimensional feature space. An integral with this number of dimensions is intractable analytically, but also numerically in our case, because we have just sparse samples of the distribution in space and not the analytical expression itself.

To ease the burden of this computation, we made a "mean-field" approximation, by considering each feature independent and treating the overall distributions as a product of univariate distributions:

$$p_{MF}(\boldsymbol{x}) = \prod_{i=1}^{18} p_i(x_i) \tag{3.10}$$

where $\boldsymbol{x}$ denotes the 18-dimensional random vector of the features and $p_i(x_i)$ the probability distribution of each univariate feature i. The mean-field approximation is better as the different features are more uncorrelated. This is certainly true for some pairs of the features we used, but may not be true for other pairs, such as the power features. With the mean-field

assumption, the KL divergence can be greatly simplified:

$$D_{KL} = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) = \sum_{\boldsymbol{x}} \prod_i p_i(x_i) \sum_i \log\left(\frac{p_i(x_i)}{q_i(x_i)}\right) = \tag{3.11}$$

$$= \sum_{\boldsymbol{x}} \prod_{j \neq i} p_j(x_j) \sum_i p_i(x_i) \log\left(\frac{p_i(x_i)}{q_i(x_i)}\right) = \sum_i \sum_{x_i} p_i(x_i) \log\left(\frac{p_i(x_i)}{q_i(x_i)}\right) = \tag{3.12}$$

$$= \sum_i D_{KL}[p_i||q_i] \tag{3.13}$$

The total KL divergence is thus just the sum of the KL divergences over the univariate distributions of the single features.

We constructed a grid of values from $\sigma = [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 5 \cdot 10^{-1}, 10^{0}, 10^{1}]$ and $M = [5, 10, 20, 30, 100]$ and computed the values of $D_{KL}(p||q)$ for each pair of parameters, for a total of 7x5 $= 35$ different values. $D_{KL}(p||q)$ was computed with the approximation in Equation 3.13, by constructing a normalized empirical histogram for each feature and comparing the values of the frequencies in each bin for p and q.

σ	M=5	M=10	M=20	M=30	M=100
10^{-4}	1.2	1.3	1.2	1.3	1.3
10^{-3}	1.2	1.5	1.4	1.3	1.4
10^{-2}	1.9	2.1	1.9	1.9	2.0
10^{-1}	31.6	33.2	35.5	37.3	37.6
$5 \cdot 10^{-1}$	125.4	150.6	170.5	181.7	189.4
10^{0}	175.3	215.0	243.5	259.8	281.2
10^{1}	258.6	336.8	413.4	456.7	555.7

Table 3.4: KL values after rearing epochs augmentation with different values of augmentation parameters.

Table 3.4 shows the value of $D_{KL}(p||q)$ for each pair of parameters. It is not immediately clear what the best parameters should be, because absolute values are difficult to interpret. Thus, in order to decide, we made some considerations about relative values. By considering a fixed value of M, say $M = 5$, we notice that $D_{KL}(p||q)$ remains pretty low for the lower values of σ, but suddenly, at $\sigma = 0.1$, it increases by more than an order of magnitude. This behavior does not repeat for any other successive steps. Even if this may seem counterintuitive, we chose the value of $\sigma = 0.1$. The rationale behind this choice was that the three lower values of sigma keep $D_{KL}(p||q)$ too low, carrying the risk of overfitting. At $\sigma = 0.1$, instead, a clear change in the distributions happens and our hope was that this

value is sufficiently high to avoid overfitting, but sufficiently low to not distort much the distribution. Then, keeping σ fixed at 0.1, $D_{KL}(p||q)$ increases by increasing M, but reaches a sort of "plateau" for the value $M = 30$. With this heuristic procedure, we decided to use the values of $\sigma = 0.1$ and $M = 30$ as augmentation parameters for the rearing samples.
We then repeated the same procedure for the grooming samples and we chose again $\sigma = 0.1$ and $M = 5$ (Table 3.5).

σ	M=1	M=3	M=5	M=10	M=20	M=30
10^{-4}	0.05	0.08	0.1	0.1	0.1	0.1
10^{-3}	0.06	0.07	0.1	0.1	0.1	0.1
10^{-2}	0.2	0.2	0.2	0.2	0.3	0.3
10^{-1}	4.5	7.5	8.5	9.3	10.1	10.0
$5 \cdot 10^{-1}$	26.5	47.1	56.5	66.0	71.6	74.3
10^{0}	37.8	68.4	83.5	99.8	112.0	116.0
10^{1}	58.0	113.2	143.7	186.0	224.6	245.4

Table 3.5: KL values after grooming epochs augmentation with different values of augmentation parameters.

After augmenting the rearing and grooming samples, the training set contained the proportions in Table 3.6 with a total of 20532 epochs.

Behavior	Proportion %
Activity	30.8
Sleep	27.2
Grooming	21.2
Rearing	20.8

Table 3.6: Proportions of the four different classes in the augmented train dataset.

We then applied two different algorithms to classify the behaviors starting from the augmented dataset.
The first one was the Random Forest classifier, presented in subsection 2.2.2. We followed the same procedure illustrated in the previous section, namely, we performed a Grid Search CV to find the best hyperparameters of the model. The hyperparameters we optimized for the RF were the number of trees used to build the forest and the number of randomly chosen features used at each decision node. By default, we set the minimum samples in a leaf to

5. We tracked the performance by using the F1-score, as we wanted a balance between the precision and the recall. Since the task is a multiclass problem, the F1-score was computed by macro-averaging, i.e. in a One-vs-All scheme considering each label as the positive class and all the others the negative class, and finally averaging all the F1 values.
The best results were reached with $max\_features = 10$ and $n\_estimators = 500$. We

Rank	Max features	n estimators	F1 macro score
1	10	500	0.943
2	11	250	0.942
3	10	250	0.942
4	9	500	0.942

Table 3.7: Best hyperparameters for the RF algorithm.

retrained the model for the entire training set and computed the confusion matrices on the training and test sets.

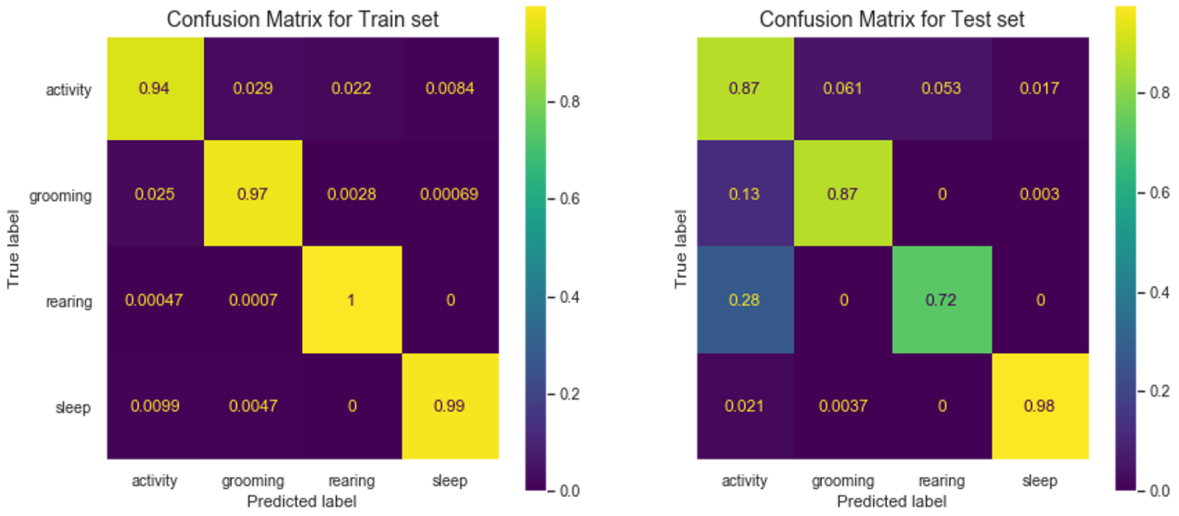

Figure 3.14: **Confusion matrices for the RF classifier**. Training (left) and test (right) confusion matrices. The normalization is done by rows.

The values in Figure 3.14 are normalized by the total number of labels per each class, i.e. each row sums up to 1. We plotted the confusion matrices in this way because of the high number of samples in the dataset.
The overall balanced accuracy for the test set is 0.91, while the average F1-score is just 0.74. The performance on the training set is excellent, but a major problem is that it drastically

Metric	Activity		Grooming		Rearing		Sleep	
	Train	Test	Train	Test	Train	Test	Train	Test
Precision	0.944	0.872	0.990	0.877	1.0	0.728	0.991	0.982
Recall	0.993	0.961	0.953	0.638	0.974	0.239	0.990	0.985
F1-score	0.960	0.913	0.973	0.739	0.981	0.357	0.990	0.983

Table 3.8: Summary results of the metrics in the training and test sets for the classification using RF algorithm.

Metric	Train	Test
Accuracy	0.981	0.914
Precision	0.980	0.863
Recall	0.983	0.705
F1 score	0.985	0.744

Table 3.9: Macro average values of the metrics with RF computed over the different classes.

drops on the test set, indicating that the model has poor generalization properties. From Table 3.8, the main critical issues arise from the very low values of recall for the grooming and even more the rearing classes. A low recall means that many grooming and rearing examples are not identified as such, but rather, as activity (Figure 3.14). Even if we selected a quite high value for the noise strength in the augmentation procedure, this behavior could be due to an overfitting of the training examples. There are also some positive signs, though. Firstly, the performance on the sleep class remains very high and as high as in the training set. Secondly, the precision and the F1-score of the grooming class amply reach above-chance values, indicating that meaningful features about this class have been learned.

Rank	Hidden layers sizes	λ	Activation function	F1 macro score
1	$(70, 50, 30)$	10^{-3}	ReLU	0.946
2	$(70, 50)$	10^{-3}	tanh	0.944
3	$(70, 50)$	10^{-2}	ReLU	0.943
4	$(70, 50, 30)$	10^{-4}	ReLU	0.943

Table 3.10: Best hyperparameters for the NN algorithm.

The second algorithm we applied was the feed-forward NN, also called Multi Layer Perceptron. We wanted to exploit the hierarchical structure of the NN and its ability to extract meaningful features at different levels of the architecture, as suggested previously in

the literature [80]. The hyperparameters searched in the GridSearchCV this time were three: the depth of the network and the number of neurons per each layer, the L_2 regularization parameter λ (i.e., weight decay) and the type of activation function.
The architecture that achieved the best score was indeed the deepest one, coupled with a relatively strong regularization parameter, to prevent the overfit (Table 3.10).

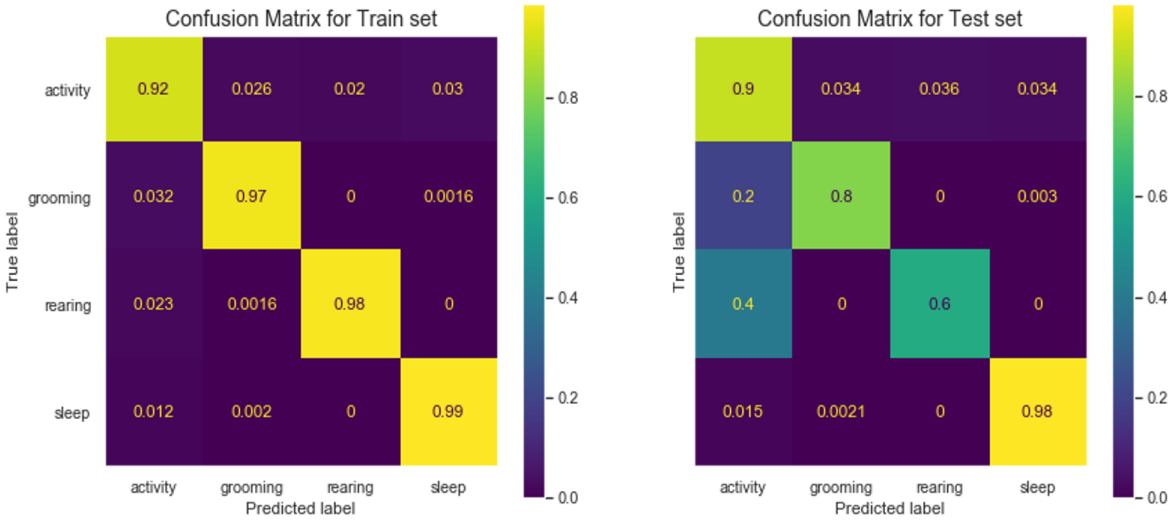

Figure 3.15: **Confusion matrices for the NN classifier**. Training (left) and test (right) confusion matrices. The normalization is done by rows.

Metric	Activity		Grooming		Rearing		Sleep	
	Train	Test	Train	Test	Train	Test	Train	Test
Precision	0.924	0.901	0.970	0.802	0.981	0.607	0.992	0.982
Recall	0.953	0.951	0.963	0.748	0.974	0.279	0.976	0.965
F1-score	0.940	0.923	0.963	0.774	0.981	0.377	0.984	0.973

Table 3.11: Summary results of the metrics in the training and test sets for the classification using NN algorithm.

The situation is very similar to the one with the RF classifier, with Figure 3.15, Table 3.11 and Table 3.12 showing a drop in performance on the test set. The same considerations we made for the RF model remain valid also for the NN model. Nevertheless, the metrics on the test set are a little bit better for the NN. In particular, grooming has a higher recall and F1-score. Also the average F1-score is higher by 2%. Unfortunately, the performance on the rearing class is still poor and not much different from the one observed with the RF model.

Metric	Train	Test
Accuracy	0.961	0.936
Precision	0.962	0.826
Recall	0.961	0.735
F1 score	0.960	0.763

Table 3.12: Macro averages of the metrics with RF computed over the different classes.

For these reasons, we chose the NN as the best model, being more balanced and also less prone to overfitting, while still reaching very promising results on the grooming class. We checked how the "prototypes" of the four behaviors, as identified by the NN, look like in the accelerometer signals. With "prototypes" we mean the epochs that yielded the higher probability of being in each class.

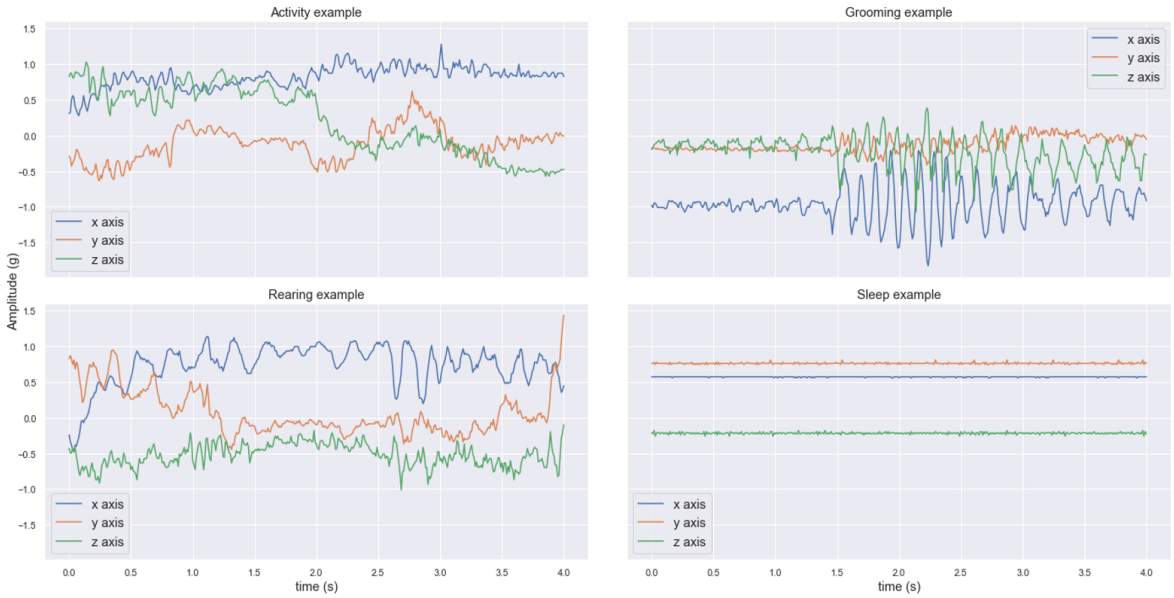

Figure 3.16: **"Prototypes" of the different behaviors as identified by the NN**. Each "prototype" is identified as being the epoch with the highest probability of belonging to each of the four classes. All three axes of the accelerometer are superimposed.

Figure 3.16 shows that the most "representative" grooming epoch is characterized by a strong oscillation at a frequency of approximately $7Hz$, in accordance with what was previously found [44]. Moreover, the profile of the sleep epoch in Figure 3.16 is very flat, as we expected. On the other hand, the rearing epoch is not very clearly represented. Indeed, there is a rise in the x-axis signal and similar decrease in the y-axis, but there are also many

oscillations which make the signal not very interpretable. This noise and non-interpretability of the signal may be at the base of the difficulty of classifying it from the accelerometer. This supports again the idea that specific rearing events may be quite difficult to distinguish from movements of the head based solely on the information of the accelerometer signals.

To conclude, the data augmentation procedure on the rearing epochs revealed to be not sufficiently powerful to allow a good classification also of rearing examples. The poor performance on the rearing class could also be due to the fact that the features extracted are not sufficiently descriptive of rearing events. Another source of error could be due to the confounding factor introduced by the mice identity. As said at the beginning of the section, we created the training and test sets by splitting the data randomly. It may be that the algorithms were not able to generalize well across subjects, resulting in a poor performance. Finally, another direction for improving the performance could be to exploit receiver operating characteristic (ROC) curves in order to find out more appropriate decision thresholds for the classification of the different behaviors.

CHAPTER 4

Unsupervised sleep partition

This chapter is devoted to the study of the electrophysiological signals recorded in mice during sleep. The corresponding analysis aims at finding and characterizing different qualitative stages within sleep. In chapter 3 we discussed a methodology to classify different behavioral states in mice based on the accelerometer signals. There, we used the word sleep to designate a single state of inactivity, to be distinguished from the active states, such as grooming and rearing. It turns out that this inactivity state is quite rich and diverse and is composed by at least two separate stages.

In this chapter, we decided to use an unsupervised approach for the classification of NREM and REM sleep and later for the investigation of a more detailed structure within NREM sleep. An unsupervised approach has the advantage of not requiring labels in the learning process. Moreover, by working directly with sample points, the classification might result more unbiased, even if overfit can happen also in the unsupervised setting.

In all the analyses we took a very similar approach to the one described in the previous chapter, namely, by splitting signals into epochs and extracting meaningful features from each epoch. This time, however, we took a per-animal approach. For each animal, we built the dataset, trained the algorithms and classified the stages independently.

4.1 Experiments and Datasets

The datasets used for this task came from $N = 14$ different $C57BL/6$ mice. They were males with an age at the time of the recording of about 70 weeks. The animals were

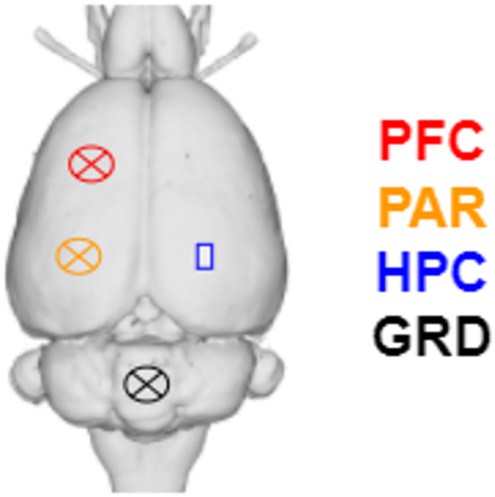

Figure 4.1: **Electrodes placement during the experiments**. Top view of a mouse brain with the locations of the electrodes. We used signals from the two left EEGs (PFC, prefrontal cortex, and PAR, parietal cortex). HPC: intracranial hippocampus (not used), GRD: ground reference.

genetically modified, 7 with an KI of human $ApoE3$ and 7 with an KI of human $ApoE4$. A total of $n = 14$ sessions were recorded. These were continuous recordings for 21 hours during day and night in the animal's home cage, where it was freely moving. Recordings started around 5 p.m. of one day and roughly ended at 2 p.m. the next day. A wireless Neurologger device was used to continuously record both cerebral and accelerometer signals. Cerebral signals consisted of two EEG channels over the frontal and parietal left lobe and two intracranial channels in the hippocampus that were not used in the analysis. The reference electrode was placed over the cerebellum, Figure 4.1.

Data coming from a sleep study in humans were also used for comparison with mice results. These data came from a sleep study performed at the Sleep Psychophysiology Lab at the University of Padova, with the purpose of comparing the memory consolidation role of sleep in insomniacs and in healthy people [85]. We have personally requested the data to the author of the study and only the data of the healthy subjects were used in the analysis. There were a total of $N = 13$ subjects, 7 of which were males, with a mean age of 24.3 years. The recording time for each subject lasted always 8 hours, from midnight to 8 a.m. of the following morning, but the total amount of time spent sleeping varied from subject to subject. The data provided for each subject consisted of just a list of labels, each

corresponding to $30s$-epoch scored by a human operator based on the AASM criteria [37]. The complete list of labels follows the time course of the recordings during the entire night.

Finally, mice data were manually scored by the CRO company Pshycogenics. Data was provided to Psychogenics as EDF file format. EEG data were read into Neuroscore (Data Sciences International) software for visualization and sleep-scoring. Using NeuroScore, artifacts were removed from the data and sleep stages assigned manually for every 10-second epoch using EEG, EMG, and/or locomotor activity (e.g. accelerometer data) by conventional methods as previously described (Ivarsson et al., 2005; Leiser et al., 2014, 2015; Parmentier-Batteur et al., 2012) as: active wake (less regular, low amplitude EEG with high EMG/LMA/Accelerometer); quiet wake (less regular, low amplitude EEG, with low or no EMG/LMA/Accelerometer); NREM sleep, consisting of high-amplitude waves with predominant delta ($1-4$ Hz), low or no EMG/LMA/Accelerometer; paradoxical (PS) or REM sleep, exhibiting stable, low-amplitude waves dominated by theta ($4-8$ Hz) with near absent EMG/LMA/Accelerometer. Intervals were then further splitted in 4-seconds epochs.

4.2 Unsupervised classification of REM vs NREM sleep

It is well known that REM sleep in mice is characterized by a strong theta $(\sim 6-9Hz)$ oscillatory activity in the hippocampus [86]. This oscillatory activity can be picked up by an EEG, in particular in the parietal lobe.

From Figure 4.2 it is possible to notice that the main characteristics of Wakefulness are the noisy EEG trace and the high amplitude accelerometer signal. NREM sleep, instead, is characterized by low-frequency and high-amplitude EEG traces, while the accelerometer is quite flat. The EEG signal of the REM sleep resembles the one of Wakefulness, but it is much less noisy and more regular, as it is driven by a strong theta activity. The accelerometer signal is more flat than in NREM sleep, because muscles are paralyzed during REM sleep.

We can confirm these observations also looking at the plots in Figure 4.3. They display the power spectral density (PSD) of the three signals. The PSD has units of mV^2/Hz and it is a density because an integration over the frequencies is needed to get the total power content. There are several features worth highlighting in every plot.
Wakefulness is the state that contains less overall EEG power of the three. Indeed, the EEG signal during wakefulness is often said to be "asynchronous", meaning that there are many frequency components in the PSD, none of which is very dominant. The NREM sleep

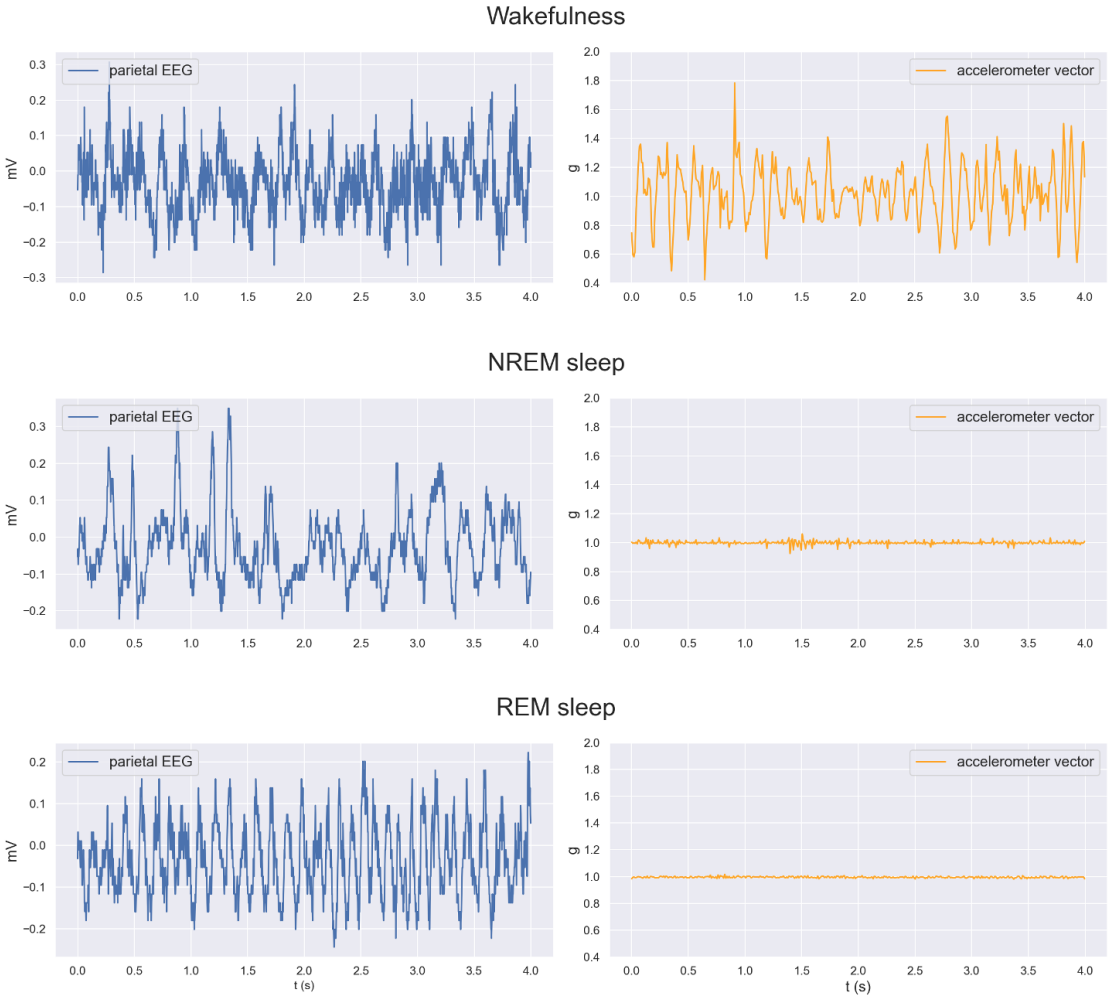

Figure 4.2: **EEG and accelerometer signals during Wakefulness, NREM and REM sleep**. The left plots show the parietal EEG signals, with the presence of slow waves during NREM and theta waves during REM. The plots on the right the length of the accelerometer vector constructed from the the three axes.

stage, on the contrary, possesses the highest overall EEG power and its peak frequency is lower than in Wakefulness, around $4Hz$. During the REM stage, there is a "shift" in the power content, with a higher peak frequency. Overall, the wakefulness PSD is somewhat in the middle between NREM and REM sleep, being more similar to the latter. This is the reason why REM sleep is also called "paradoxical" sleep, as the frequency content of the EEG signal during REM is quite similar to the one of Wakefulness.

For the task of separating sleep into NREM and REM stages, we started from data already filtered from activity segments. We took the same approach described in chapter 3,

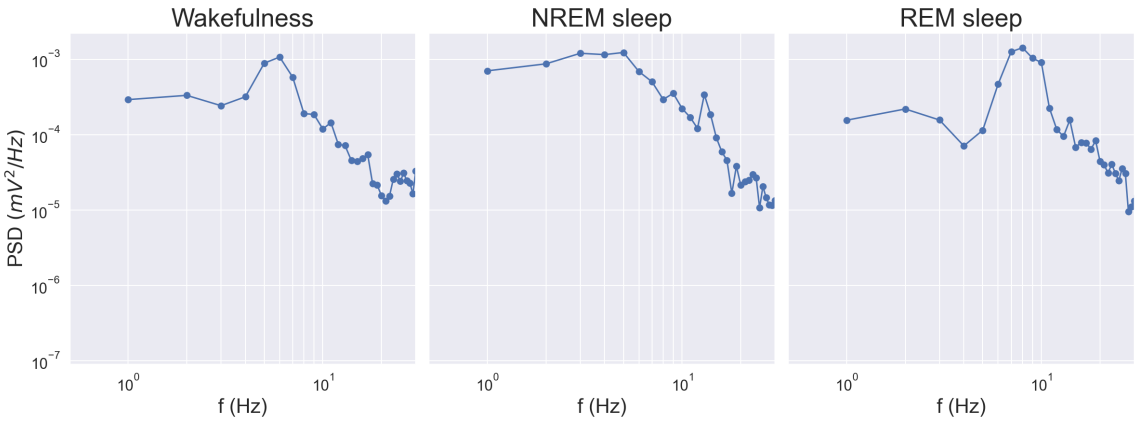

Figure 4.3: **Power spectral densities of the EEGs in Figure 4.2**. REM sleep shows a clear power peak around 8 Hz, similar to the PSD of Wakefulness. PSDs have been built using Welch method.

namely the splitting of the signals into epochs and the extraction of features from every epoch. In this case, however, since we wanted to keep the individual variability of every animal separate, we built a dataset and trained the algorithm for each session separately.

4.2.1 Description of features used

Given the importance of the theta oscillations during REM sleep, it was natural to adopt a measure of the content of these frequencies in the signal. To quantify this, we used the power between 5 and $10Hz$, based on what we can observe from Figure 4.3. Since an absolute value of the power has no physical meaning and can vary based on experimental conditions, we sought for a suitable normalization factor. A natural choice was the power contained in the lower frequencies, as we observe a shift in the peak frequency of the PSDs in Figure 4.3 from $\sim 4Hz$ during NREM to $\sim 8Hz$ during REM sleep. We took the ratio of θ/δ power as a feature, with the delta defined between 1 and $4Hz$. We used this feature for two reasons: first, it gives a way of normalizing the raw value of θ power; second, it enhances the detection of the shift towards higher frequencies observed during REM sleep, because there is more power in higher frequencies and less in lower ones, so the ratio takes both of these effects into account.

Based on [87], we also decided to use the ratio between "high" frequencies power, defined to be the power contained between 15 and 30 Hz and the "low" frequencies power, defined to be the power contained between 1 and 15 Hz.

4.2.2 Preprocessing of the features

Once these two features were computed for every epoch of the sleep segments, we pooled them together and created a train dataset for each session. However, a major difficulty in classifying NREM and REM sleep is represented by the different proportions of the two. In mice, REM sleep represents about 10% of the total sleep time. The imbalance in the classes' proportions poses a problem for every ML algorithm, because there may not be enough samples to properly train the algorithm on the minority class.
In order to alleviate this problem we took two measures:

1. Since the first episodes of REM sleep occur some hours after the beginning of the recordings, we just retained the sleep segments occurring after 10000 seconds from the starting point of the recording.
2. Since REM episodes usually occur during long stretches of sleep, we just retained the sleep segments lasting at least two minutes.

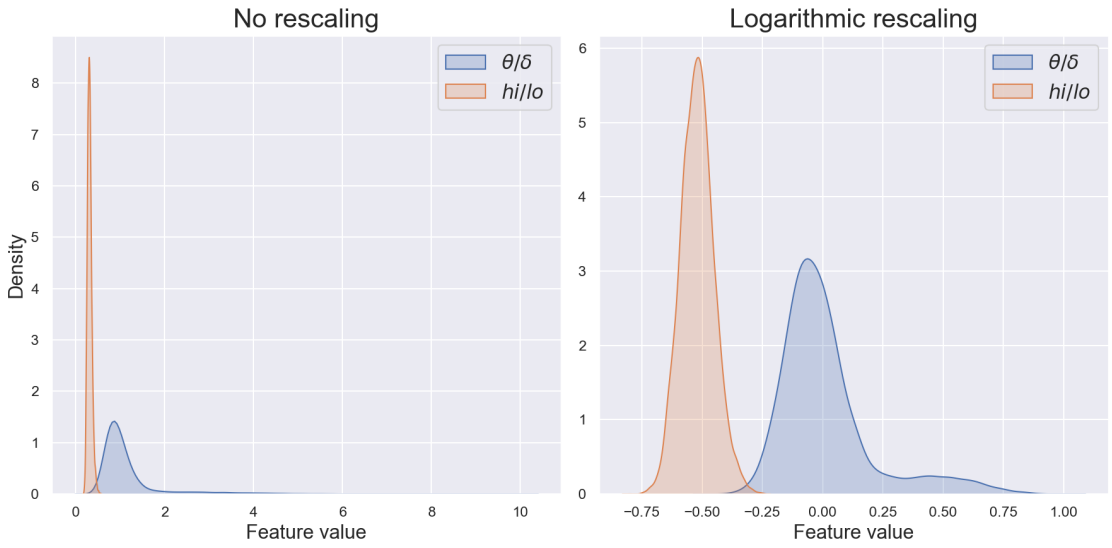

Figure 4.4: **Distributions of the two features before and after scaling**. KDE plots of an example dataset before (left) and after (right) log scaling. In the right plot we have called the features with the same name, but in fact they are $\log(\theta/\delta)$ and $\log(hi/lo)$, relative to the ones on the right.

The left plot of Figure 4.4 shows how the two features are distributed for an example dataset. For both features, the empirical probability distributions are very concentrated towards the lower values and the θ/δ feature possesses a very long tail with few samples

having very high values. These kinds of skewed distributions are often encountered when dealing with powers and ratios of powers. We rescaled the distributions by applying a logarithmic transformation to the two features, which results in the plot on the right of Figure 4.4. The effect of the logarithmic rescaling is to "expand" the distribution on the low values and "squeeze" it on the high ones. Indeed, Figure 4.4 right shows that the long tail of the θ/δ distribution is much more compact, the lower values are more spreaded and the two distributions have similar scales.
Moreover, the tail of the θ/δ distribution seems to show a separate cluster which is less dense than the main one and contains epochs with a high value of θ/δ, signs that it could represent REM epochs.

Another transformation that we applied to the datasets was a Gaussian smoothing, described in subsection 2.3.4, directly on the features' time series. One has to pay attention to smooth just points coming from consecutive epochs and not from different segments, in order to avoid artifacts. Figure 4.5 shows what are the effects of using different values for the standard deviation of the Gaussian kernel, i.e. in considering more points for the weighted average. The scatter plots show that the points get more and more clumped as σ increases and the distributions become more distorted. On the other hand, an intermediate value of σ could decrease the spreading of the points and make the clusters more identifiable. Indeed, we sought to find a value of sigma that allowed a better classification performance, while still working with a dataset not much different from the original one.

4.2.3 Performance of the clustering algorithms

Contrary to what we did in chapter 3, for this task we exploited unsupervised models that directly fit the datasets without any labels. This, in turn, reduced the task of classifying the two main stages of sleep into the task of finding two clusters, corresponding to NREM and REM sleep epochs.
We sought to find an appropriate clustering algorithm able to deal with the very different densities and shapes of the two clusters and the lower number of REM sleep epochs. For this reason, of the many possible clustering algorithms, the ones based on density estimation, such as DBSCAN, were not tested, since they work well with clusters having roughly similar densities.

We decided to exploit clustering algorithms that model directly the underlying distribution of the dataset, namely, K-means and the Gaussian mixture model (GMM), which we already discussed in subsection 2.2.4 and subsection 2.2.5. In those sections we discussed

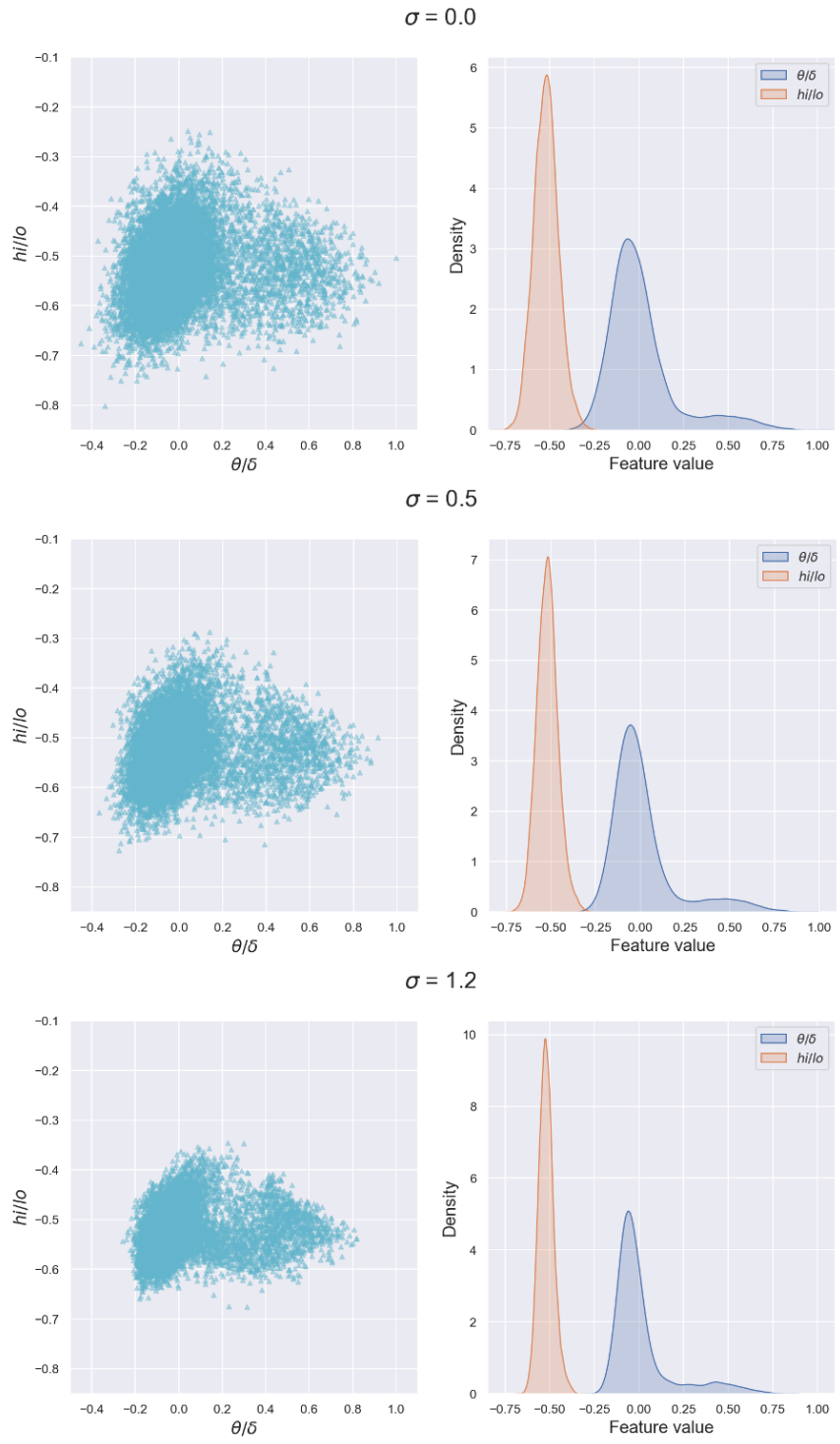

Figure 4.5: **Effect of the Gaussian smoothing**. Example dataset after the application of Gaussian smoothing with different values of the smoothing factor σ.

that K-means can be seen as a less powerful version of the GMM, because each cluster is modelled as having the same diagonal covariance matrix, while in the GMM each Gaussian component can have a full covariance matrix. Figure 4.6 shows this difference on an example

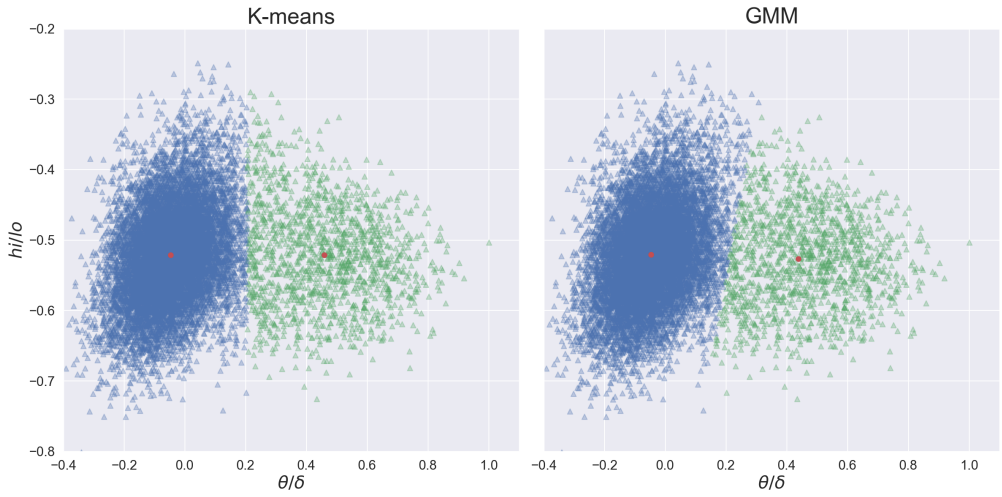

Figure 4.6: **Results of clustering with K-means and the GMM**. The algorithms have been applied to the example dataset. Red dots are the centers of the two clusters and the different colors represent the two clusters. The main difference lies in the fact that the GMM can have non-linear decision boundaries.

dataset, where the different colors represent the resulting clusters after the application of the K-means and GMM algorithms. The difference is barely visible and minimally affects just a few points near the decision boundary, but it can be useful to compare the performance of the two, to see if the higher representational power of the GMM is necessary. Figure 4.6 also shows that the centers of the two clusters are mainly separated in the horizontal axis, i.e. by the θ/δ feature, while the hi/lo feature does not seem to be very different between the two clusters.

To quantify the performance of the clustering algorithms, we compared the output of the clustering with manually scored labels provided by the company Psychogenics, as described in section 4.1. For each session, an expert scorer labelled every 4-second epoch as Active, Sleep, Paradoxical and Quiet Wakefulness. We actually did not use the latter, because it is defined as an intermediate state between sleep and wakefulness and was not directly comparable with any of our criteria. Therefore, to compute the metrics, we just disregarded all the epochs labelled as Quiet Wakefulness. For what concerns the other three labels, Activity corresponds to Wakefulness, Sleep to NREM sleep and Paradoxical to REM sleep.

For this task, we were interested in the correct identification of REM sleep. Therefore,

to compare our results to the manually scored labels, we set the REM epochs as the positive class and everything else as the negative one. Given the very high class imbalance resulting from this separation, we had to use metrics that are not affected by the imbalance.
We decided to compute four different metrics: Balanced Accuracy (BA), Precision (P), Recall (R) and F1-score. For the REM detection we wanted a high value for both the precision and the recall, to ensure that each REM interval gets correctly detected and that no additional epochs are incorrectly labelled. A right balance between P and R is additionally assured by a high F1-score.

Finally, we compared the labels also with another method used for detecting REM sleep, based on a classical thresholding method. This method consisted in taking the sleep segments, dividing the parietal EEG signal in 5-seconds epochs, calculating the power in the bands $6 - 8Hz(\theta)$ and $1 - 4Hz(\delta)$ and taking their ratio, similarly to what we did for computing the feature θ/δ. Then, a fixed-threshold decision is applied: if the ratio is above 4.5, the epoch is classified as REM sleep, otherwise as NREM sleep. Furthermore, a post-processing is applied after the detection: if two epochs scored as REM sleep are separated by less than 30 seconds, they are merged together. The advantage of this method is that it is very simple to implement and the resulting classification can be easily interpreted on the base of the power content.

The comparison with the labels was done for three different algorithms: the fixed-threshold decision, that we denote with "WF" (workflow), the K-means and the GMM algorithms. The last two algorithms depend on the parameter σ that controls the width of the Gaussian kernel used in the Gaussian smoothing applied to the epochs' features. Thus, we compared the performance of these last two algorithms when using different values of σ. We ran a grid of values of σ for each session and computed the values of the metrics, in order to evaluate some statistics over the different sessions.

Figure 4.7 summarizes using box plots the metrics obtained on the different sessions separately and when varying σ. The ML models reach high recalls, which increase with increasing values of σ. On the other hand, the WF method has very low recall. The precision, instead, reaches higher values using the WF. For the K-means algorithm, the precision increases with σ, while it reaches an optimum at $\sigma = 0.8$ in the case of the GMM. The majority of the plots in Figure 4.7 show the presence of a strong outlier. This outlier always corresponds to the same session, which shows a weaker θ oscillation during REM sleep, thus making this stage much harder to identify. Given the presence of this outlier, we used median values instead of mean ones to evaluate the metrics for the whole dataset. The

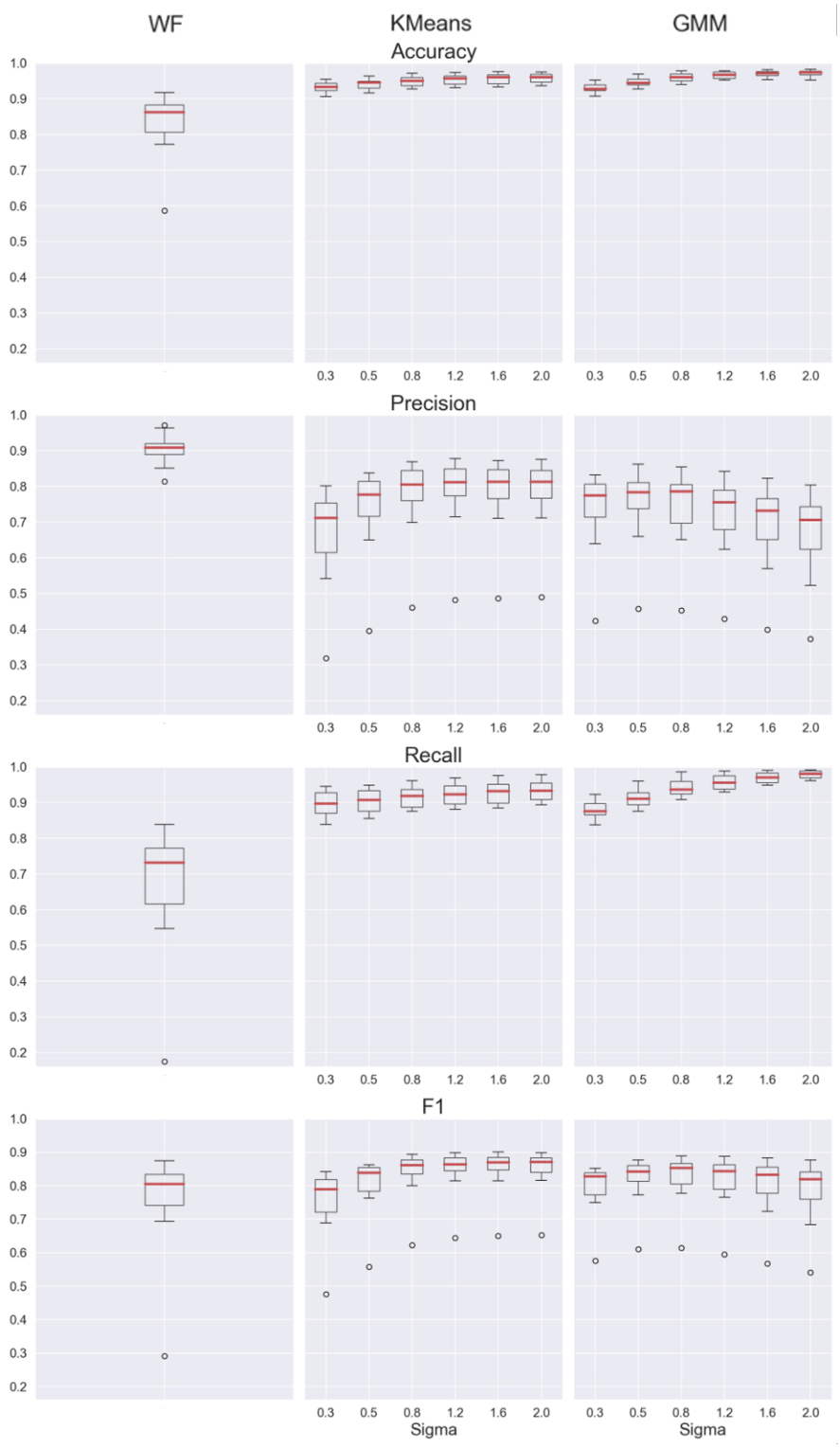

Figure 4.7: **Summary of the metrics for n=14 sessions**. For K-means and GMM the performance varies with the Gaussian smoothing parameter σ.

values of the F1-score, which we sought to maximize, are quite similar between the three algorithms. The best overall F1 score is 0.872 and is achieved by the K-means algorithm with $\sigma = 2$, while the score of the WF is 0.801 and the best one of the GMM is 0.853, achieved with $\sigma = 0.8$.

4.2.4 Algorithm refinement

Next, we sought different strategies to improve these results. From the results in Figure 4.7, it is clear that the major weakness of the ML algorithms is a low value of the Precision, meaning that there is a consistent number of false positives. A natural direction to improve the performance was to try to improve the precision of the algorithms, with the goal of improving the F1-score.

In particular, we adopted two strategies to improve the results.
The first one was applicable just to the GMM algorithm, given the probabilistic nature of the model. This strategy was based on the observation that, when performing clustering using the GMM, we are implicitly doing it in two steps:

1. First, we fit the model to the data. This returns the information about the underlying Gaussian components, namely, their means, covariance matrices and prior probabilities. At this point, we do not already have a partition of the dataset, but just a probability model fitted to the experimental data.
2. Then, in a second step, the way of producing clusters is to draw decision boundaries in the feature space based on the fitted probability distributions. Implicitly, up to this point the decision boundary was defined as the curve where the posterior probability of a point belonging to the REM cluster is equal to 0.5.

Figure 4.8 depicts the situation. When epochs are assigned to a cluster, the assignation is based on the posterior probability $p(\text{REM}|\boldsymbol{x})$, where $\boldsymbol{x}$ is the vector of features. On the sides, the class-conditional probability distributions of the two components for the two features are represented. Taking as example the θ/δ feature, the solid black line represents the decision boundary where $P(\text{REM}|x_1) = 0.5$, i.e where the class-conditional distributions intersect. But, in general, there is no a-priori reason to use this value instead of another one and the value adopted depends on the particular task. For example, to decrease the number of false positives, we may adopt a higher probability value for the decision boundary, as the one represented by the dashed line in Figure 4.8. In Figure 4.9 we have computed the posterior probabilities of the two classes for the θ/δ features, together with the two

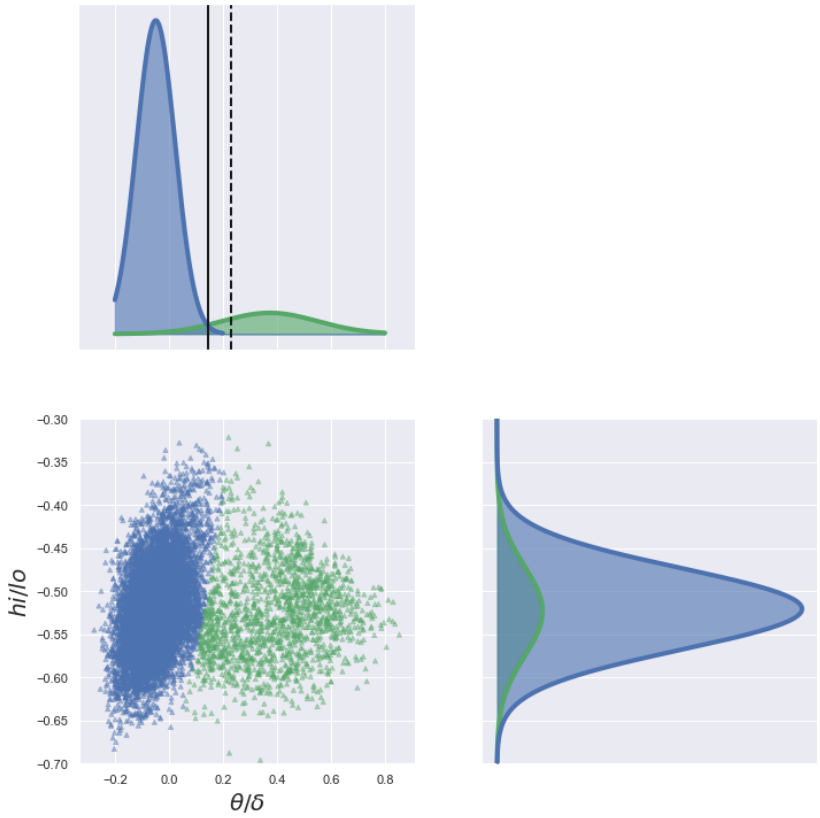

Figure 4.8: **Clustering of epochs starting from distributions**. The scatter plot shows the resulting clustering of the dataset. On the sides are shown "unnormalized" class-conditional probabilities for the two features, with highlighted the decision boundary for the first one.

different decision boundaries.

The solid black line corresponds to the point where the two posterior probabilities are equal and have a value of 0.5. In the region around this point $p(\text{NREM}|\theta/\delta)$ decreases, while $p(\text{REM}|\theta/\delta)$ increases with the same trend. The dashed line, instead, represents a decision boundary that assigns an epoch to the REM cluster only if the posterior probability for REM is very high, with the effect of minimizing the number of false positives.

In order to find the best value of the "probability threshold" for classifying REM sleep, we proceeded as follows. First, we fixed the value of σ to 0.8, which yielded the highest F1-score for the GMM. Then, we constructed a grid of values for the threshold and successively classified the epochs of each session based on whether or not their REM posterior probability was above or below the threshold value. This procedure mimics exactly how a ROC curve is build, from which optimal thresholds can be derived. Finally, for each threshold we computed the four metrics.

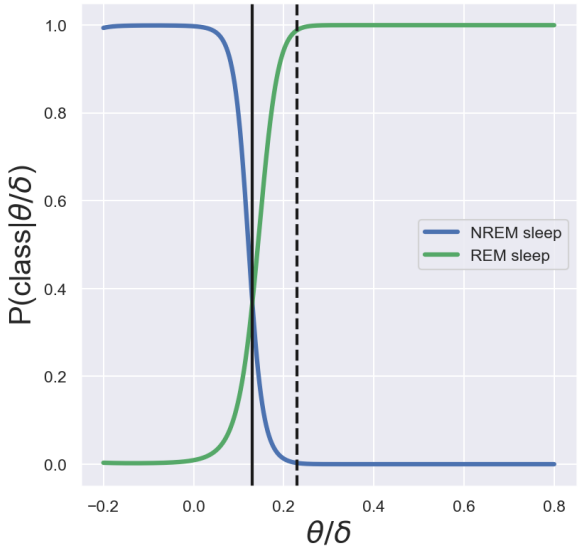

Figure 4.9: **Posterior probabilities for the two classes**. Highlighted with the solid and dashed black lines are the same decision boundaries of Figure 4.8.

Additionally, we also employed a second strategy, i.e. to post-process and clean the created REM intervals. We decided to delete every REM sleep interval lasting less than 10 seconds and merging distinct REM intervals closer than 10 seconds. We also applied this strategy to the intervals classified by the K-means algorithm, but it yielded no change in the metrics.

Figure 4.10 shows how the metrics change with different values of the threshold. The precision monotonically increases with increasing values of threshold, as one could expect, given that higher thresholds create less false positives. Correspondingly, the recall score decreases. Overall, the F1 score increases and seems more or less constant when varying the threshold value. We decided to use a value for the threshold of 0.95, for different reasons. As well as giving the second-best median F1-score, it also possesses low variance in the F1-score values of the different sessions, maintains the median recall value above 0.9 and keeps the gap between precision and recall quite small, as we sought to have high values of both metrics.

Thus, as the overall best configuration, we chose to use $\sigma = 0.8$ and the decision threshold$= 0.95$, achieving an overall median F1-score of 0.881, thus improving by approximately 1% the best F1-score achieved with the K-means.
Moreover, the result of the GMM algorithm has at least two advantages with respect to the K-means algorithm. The first one is that the "gap" between precision and recall is smaller:

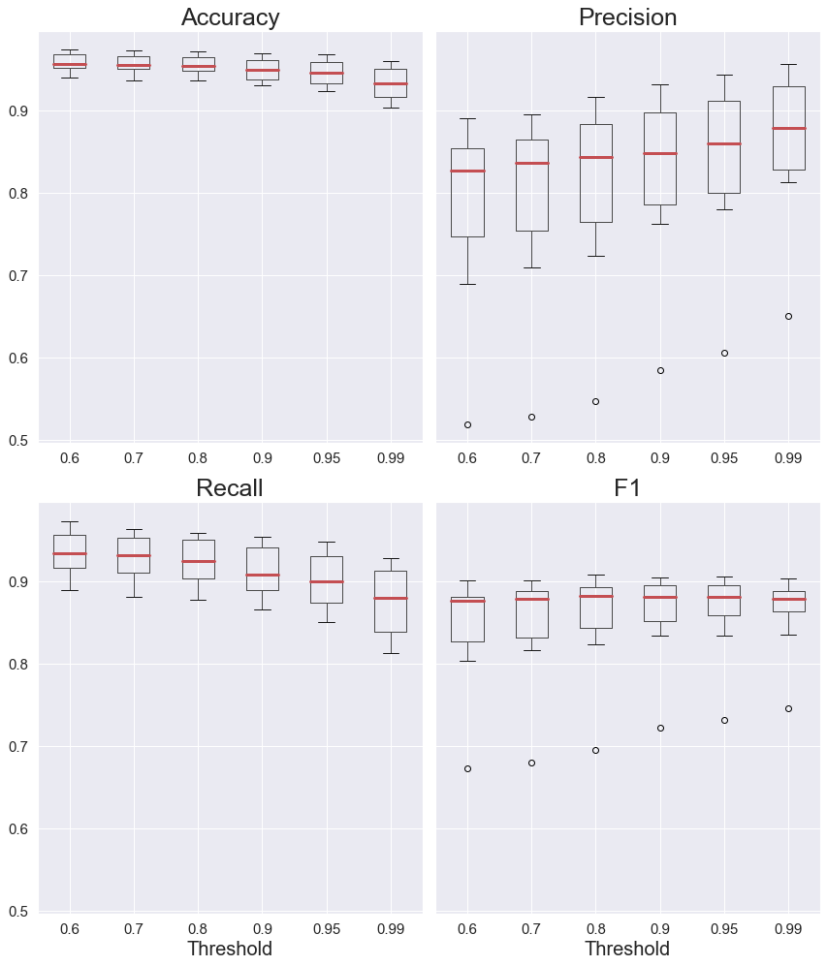

Figure 4.10: **Summary of the metrics for n=14 session with the GMM**. The metrics vary as a function of the "probability threshold" used for classifying REM sleep.

K-means reaches a precision of 0.813 and a recall of 0.933, while the GMM a precision of 0.861 and a recall of 0.901. For a decrease of about 3% in the recall score, an increase of about 5% in the precision is achieved using the GMM, making the latter algorithm more advantageous and showing the benefit in using the GMM instead of the K-means algorithm. The second advantage of the GMM is that also the outlier session gets better classified, as its F1 score rises from around 0.65 with K-means to 0.73 with the GMM.

4.2.5 Final results

To conclude, in Table 4.1 we report the average values of the metrics and the overall agreement, in the form of balanced accuracy, between the manual scoring and our predictions

using the GMM, including also the Wakefulness epochs that were already scored using the method described in chapter 3. Considering the epochs from all the three states, the overall median balanced accuracy is 0.935, which is quite high and for example higher than the ones reported in [87].

Balanced Accuracy	Precision	Recall	F1-score
0.931 ± 0.005	0.864 ± 0.016	0.900 ± 0.010	0.880 ± 0.006

Table 4.1: Summary performance of the GMM on n=13 sessions, excluding the outlier. The balanced accuracy was computed considering also Wake epochs. The values reported are average values, with errors representing the SEM.

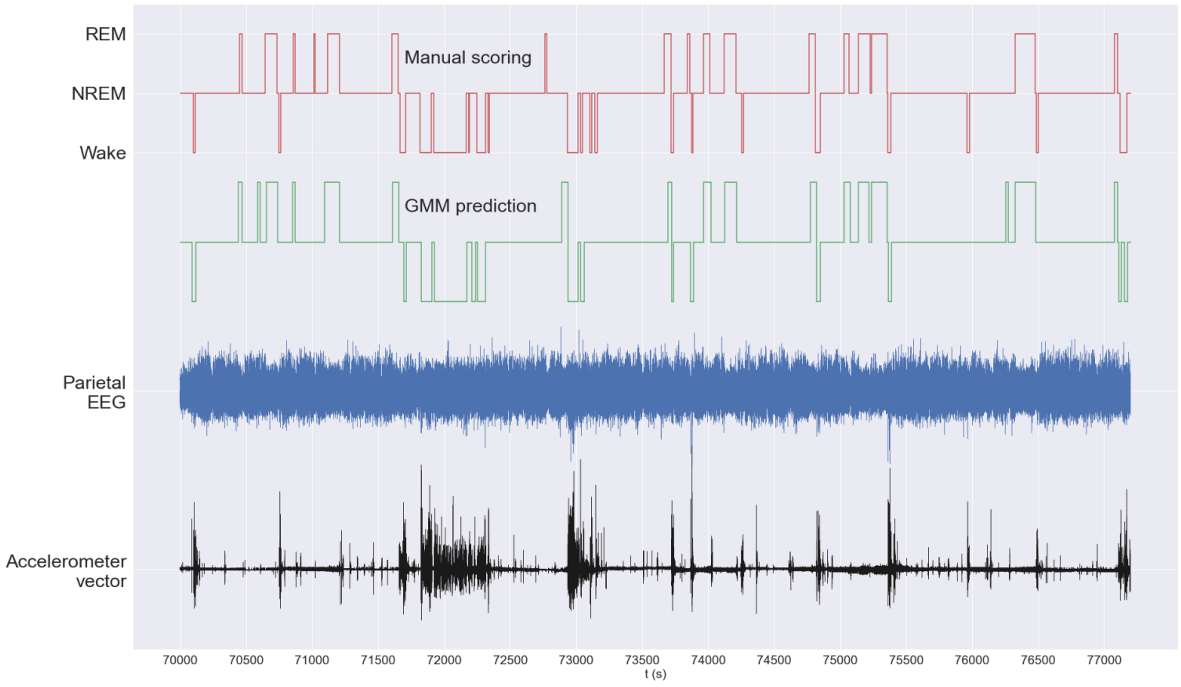

Figure 4.11: **Comparison between manual scoring and GMM prediction**. The two hypnograms in the upper part of the figure visually display the similarity between human scoring and algorithm prediction. Each horizontal "level" corresponds to a certain stage, as described on the left. Under the hypnograms there are the parietal EEG and accelerometer vector signals.

In Figure 4.11 we also plot the hypnogram of a portion of a session lasting two hours. We already presented the hypnogram as a tool for describing stages succession in subsection 1.4.1. The two hypnograms are made with the labels provided by the manual scoring

and the predictions made by the combination of the GMM and the algorithm described in chapter 3 for detecting activity and sleep. Looking at the succession of the stages in the upper part of Figure 4.11, we notice that the predictions are very good and the temporal dynamics resembles the one of the manual scoring. The only major mistakes happen concurrently with brief awakenings during sleep, that are simply disregarded in the predictions, or concurrently with short REM episodes, that are either not present in the prediction or are wrongly predicted. This result is in line with the fact that we retained just REM intervals lasting at least 10 seconds.

4.3 Unsupervised classification of NREM sleep

In the previous section, we obtained a procedure for identifying REM sleep. We now turn our attention to NREM sleep. We continued our "hierarchical" approach, seeking a way to further investigate and partition NREM sleep based on electrophysiological signals. In our study, we were driven by the hypothesis that also mice possess three NREM sleep stages, as observed in humans.

Already by a visual inspection of Figure 4.12, it is possible to see differences in frontal EEG signals during NREM sleep intervals. In Figure 4.12, the four plots on the left side show different EEG and accelerometer signals taken from a single session. The most striking feature is that two of the plots show EEG signals with lower amplitude, relatively to the other ones. In segments 2 and 4, moreover, the average frequency of the signals seems higher. The corresponding accelerometer vector signals show little signs of activity, the only relevant lasting less than one second in segment 3. When looking at the signals, low-amplitude frontal EEG is quite often accompanied by brief activity in the accelerometer.
In human sleep, oscillations with higher amplitude are associated with deeper stages of sleep, i.e. N2 and N3, while low-amplitude, mixed-frequency signals are associated with the N1 stage, which represents a transient state. Thus, already by looking at the EEG signals qualitatively, we realized that NREM sleep can really display different kinds of stages with very different and defined characteristics. These differences are found extensively during a recording, indicating that these different types of signals are representative of distinct biological stages.

The plots on the right of Figure 4.12, instead, show the PSDs of the different segments. Segments 1 and 3 share several characteristics. First, their PSDs do not exhibit a sharp peak and the maximum value of the spectrum occurs around $2-3Hz$. Second, the magnitude of

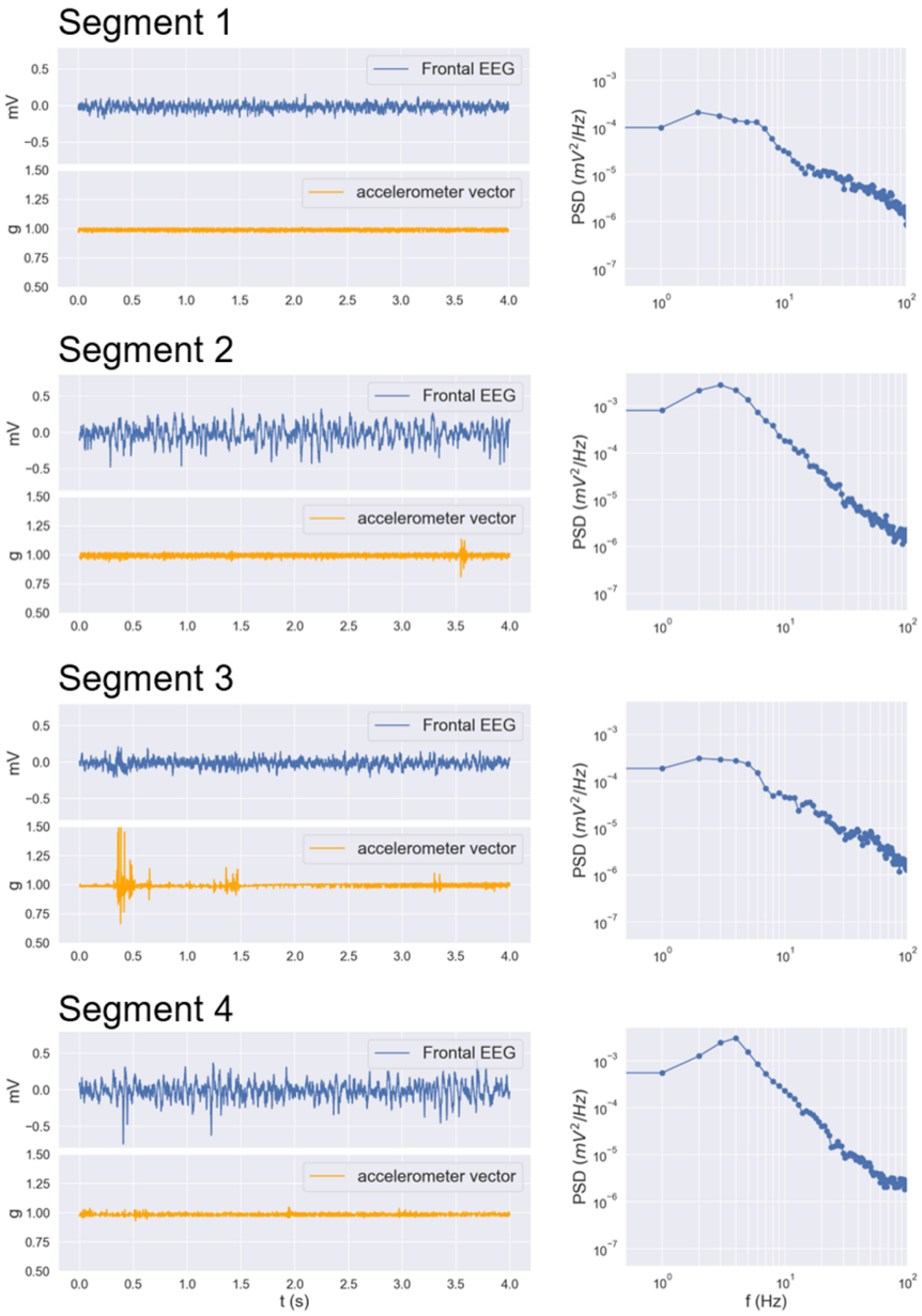

Figure 4.12: **Examples of qualitatively different signals during NREM sleep**. **Left**: four examples of frontal EEG and accelerometer signals lasting 4 seconds found during NREM sleep, in the same session. **Right**: corresponding PSDs of the EEG signals, with logarithmic scales on both axes.

the PSDs is quite low and the overall power content is an order of magnitude lower than in segments 2 and 4. On the contrary, these ones possess a dominant peak around $3-4Hz$. Also importantly, although the peak in segments 1 and 3 is an order of magnitude lower, the power content in the higher frequencies is comparable to that of segments 2 and 4, meaning that a higher proportion of power in segments 1 and 3 is occupied by higher frequencies.

We tackled the problem with the same approach of the previous section, namely, we used the EEG signal during NREM sleep, divided it into epochs lasting 4 seconds and extracted meaningful features from them, which were then used to model the distributions of the datasets through a GMM, with subsequent clustering of the epochs. Compared to the previous case, we used EEG over the frontal lobe and a set of different features. We decided to use the frontal EEG for this task because it was the one that showed the bigger and clearer changes within NREM sleep and because the frontal EEG was used as well in [47].

4.3.1 Description of features used

Given these spectral characteristics of the signals, we came up with five different features, suited for the task of characterizing and distinguishing different phases within NREM sleep.

The first three of these features were the power contained in three different frequency bands, with edges defined as $1-3Hz$, $3-6Hz$ and $6-18Hz$ respectively. In the following, we refer to these bands as δ,θ and α respectively. These names were chosen for convenience, reflecting the usual order adopted from low frequency bands to higher ones, but they do not generally reflect these bands definitions. The frequency range spanned by the bands is increasingly broader, to reflect the fact that lower frequencies have a higher power content and nearby lower frequencies can have very different properties. Moreover, we tried to capture the differences in the PSDs, noting that after $6Hz$ there is a drop in the power content and that segments 1 and 3 have a very low peak frequency.
Finally, these frequency edges were designed to mimic the criteria based on the EEG signals used in standardized human sleep scoring [37]:

1. N1 sleep stage is scored when "low amplitude, mixed frequency activity" is present. This term refers to the power content of a band with edges $4-7Hz$. Our θ band was designed to reflect this criterium.
2. N2 sleep stage is scored when sleep spindles occur. These events are defined to have frequencies ranging from 11 to 16 Hz and our α band was designed to incorporate

these events.

3. N3 sleep stage is characterized by slow wave activity, which is oscillations with frequencies around $0.5 - 2Hz$. We designed our δ band to reflect these very slow oscillations.

The fourth feature we adopted was the amplitude. The amplitude of the EEG signal in a certain epoch is defined as the difference between its highest peak and its lowest trough, as depicted in Figure 4.13.

Also in this case, this feature was used for the analogy with human sleep scoring, where N3

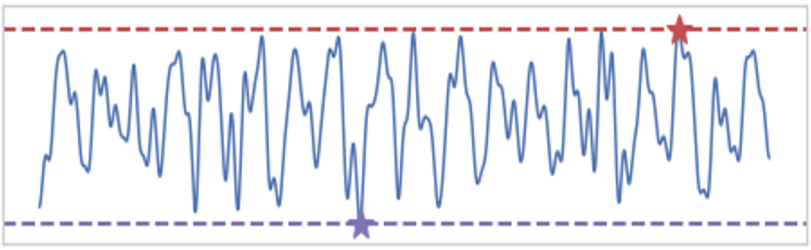

Figure 4.13: **Amplitude feature**. This feature is calculated by considering the "highest" and "lowest" points in an epochs.

is scored when waves have a peak-to-peak amplitude $> 75\mu V$, measured over the frontal regions. Apart from the exact magnitude of the amplitude, this rule clearly states that deeper stages of sleep should possess a high EEG signal amplitude.

Lastly, we also adopted the spectral entropy (SE) of the frontal EEG signal, which is a non-linear spectral measure. The SE stems from the concept of Shannon Entropy applied to the PSD of a signal. The Shannon Entropy of a mass probability distribution P of a discrete random variable X is defined as the expected "amount of information" [88]:

$$H(X) = -E[log_2 P(X)] = -\sum_i P(x_i) \cdot log_2[P(x_i)] \tag{4.1}$$

Where the x_i are the possible outcomes of the random variable X. By rewriting the quantity $-log_2 P(x_i)$ as $log_2[1/P(x_i)]$, we can see why this quantity is called the "information" content of a certain event x_i. If the event x_i is very rare, its probability is very low and the quantity $1/P(x_i)$, in turn, is very high. Therefore, also its information $I(x_i)$ will be high. For an almost certain event, instead, the ratio $1/P(x_i)$ is closer to one and its information closer to zero, because no information is gained in knowing that a certain event has happened.

In general, for a discrete random variable X with N possible outcomes, the probability

distribution that maximizes the Entropy $H(X)$ is the uniform one, with each outcome having a probability $1/N$.

The SE generalizes the concept of Shannon Entropy to the PSD of a signal [89]. To convert a PSD, like the ones in Figure 4.12, into a probability mass function, one first normalizes the PSD:

$$P(f) = \frac{PSD(f)}{\sum_{k=1}^{f_{max}} PSD(k)} \tag{4.2}$$

In this way, the different PSDs become comparable in scale, as can be observed by plotting the corresponding probability mass functions of two segments in Figure 4.14, top.

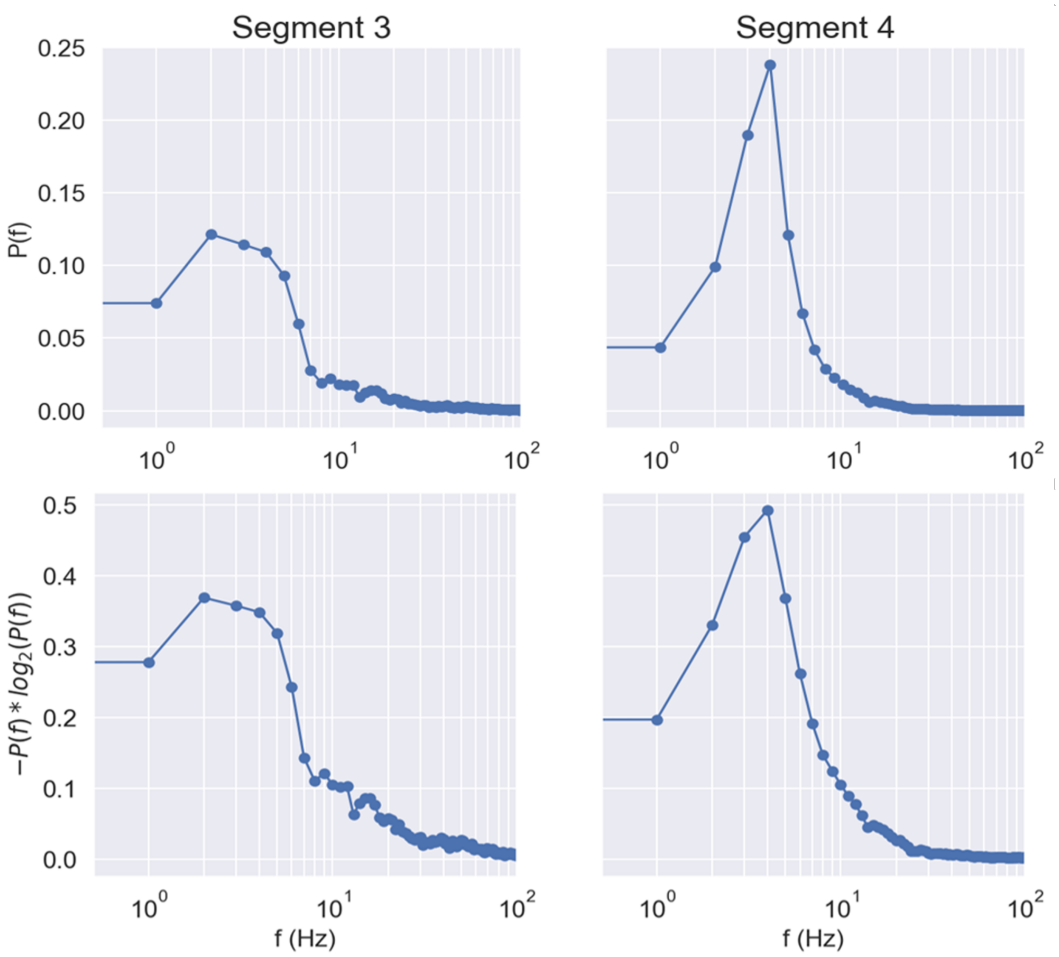

Figure 4.14: **Representation of the spectral entropy**. **Top**: "probability distribution" resulting from the PSDs of two different segments in Figure 4.12. **Bottom**: function used inside the summation of the SE.

With this representation it is now possible to better appreciate the difference between the

distribution on the left and the one on the right. The first looks "flatter" and more qualitatively similar to a uniform distribution, while the second exhibits a clear peak, more similar to a Gaussian or exponential distribution.
Then, the SE can be calculated from $P(f)$:

$$SE = -\sum_{f} \frac{P(f) \cdot log_2[P(f)]}{log_2[N]} \tag{4.3}$$

The factor $log_2[N]$, where N is the number of frequencies contained in $P(f)$, is the value of the SE assuming that $P(f)$ is uniformly distributed and is used to normalize the SE with values between zero and 1.
By looking at the probability distributions of the segments in Figure 4.14, we expect the one on the left to possess higher SE, given that it resembles more closely a uniform distribution. This is confirmed by computing the actual values of SE for the segments. Segment 3 has a normalized value of SE of 0.65, while segment 4 of 0.47. Figure 4.14, bottom, shows the function used inside the summation of the SE for the two segments.
In essence, the SE gives a measure of uncertainty about the distribution of the power content of a signal. A signal described by many frequency components has a high SE, because it can take many different oscillatory modes and thus many "states". For this reason, the "flatter" profile of segment 3 in Figure 4.14 gives rise to a higher SE, because it describes a system that can take up many different spectral components.

4.3.2 Preprocessing of the features

Once we extracted the five features, we sought an appropriate way of normalizing them in order to bring them on the same scale. Because of the shape of the distributions, we decided to adopt a standard scaling, described in section 2.3. An example of the distributions of the features of a session is plotted in Figure 4.15. Importantly, all the univariate distributions are bimodal and approximately resemble a Gaussian distribution.

4.3.3 Silhouette scores

To validate the outputs of the algorithm and to evaluate the clustering performance, we used a metric called Silhouette score [90]. The Silhouette score is computed per each

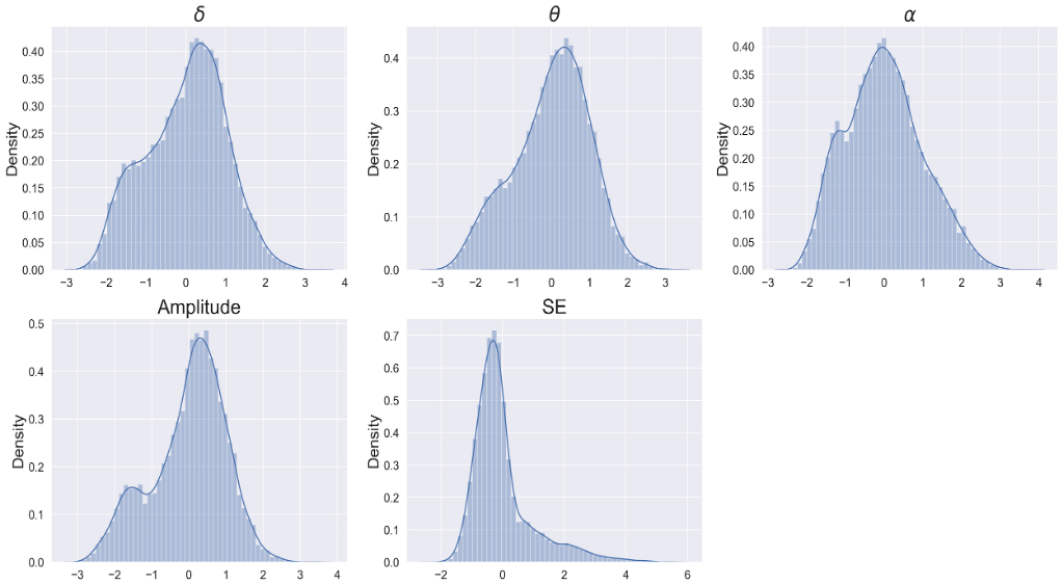

Figure 4.15: **Distributions of the features used in NREM sleep partition**. The five features are scaled and densities are normalized to give an integral of 1.

sample in the following way:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)} \tag{4.4}$$

The subscript i indicates that the Silhouette score is computed for the sample i. Let us denote with C_i the cluster to which sample i belongs. Then, a_i is the mean intra-cluster distance, that is, the mean distance between the sample i and all the other samples in C_i:

$$a_i = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \tag{4.5}$$

b_i, instead, is the mean nearest-cluster distance, i.e. the average distance between the sample i and all the samples in another cluster, different from C_i, that minimizes the distance from i:

$$b_i = \min_{k \neq i} \frac{1}{|C_i|} \sum_{j \in C_k} d(i, j) \tag{4.6}$$

Depending on whether a_i or b_i is greater, there are three possible cases:

$$s_i = \begin{cases} 1 - a_i/b_i & \text{if } a_i < b_i \\ 0 & \text{if } a(i) = b(i) \\ b_i/a_i - 1 & \text{if } a_i > b_i \end{cases} \tag{4.7}$$

From Equation 4.7 it is clear that s_i can only take values between -1 and 1.
The Silhouette score measures how well the sample i has been assigned to its cluster. For an extremely well assigned point, $a_i \ll b_i$ and $s_i \simeq 1$. On the other hand, when sample i is actually much closer to the samples of another cluster rather than the samples of its own cluster, $a_i \gg b_i$ and s_i is close to -1 and it is natural to think that the sample i has been misclassified. A value of $s_i \approx 0$ is intermediate and corresponds to the situation in which sample i can't be easily assigned to any cluster.
There are a various advantages in using Silhouette scores:

1. They do not depend on the particular clustering algorithm used, but are based on the samples in the dataset and on the labels assigned by the clustering method.
2. They allow to evaluate which is the "natural" number of clusters present in the dataset.
3. They are calculated for each sample, thus statistics can be computed for the entire dataset, like the average value.

About the last point, a common procedure is to build the so-called Silhouette plot [90], in which the Silhouette scores of all the samples in the dataset are represented together, by dividing the samples into the different clusters. The Silhouette plot is used to assess the validity of a clustering result, by quantifying the "width" of each cluster and studying how the scores distribute. Figure 4.16 shows an example of a Silhouette plot, computed on the clustering results of an example dataset.
In Figure 4.16 left, there are three clusters differentiated by the color and the size of the region they occupy in the vertical axis denotes their size. The samples in each cluster are ranked according to their Silhouette score, from highest to lowest, and are represented by a continuous line that "fills" the region corresponding to each cluster. Therefore, the continuous areas have to be thought of as composed by many small horizontal bars, reaching out for a quantity corresponding to the Silhouette score of each sample. The black dashed line represents the average Silhouette score for the entire dataset and it is useful to compare how the Silhouette scores in each cluster are distributed relative to the average value. A typical example of a good cluster would be the light blue one, because it mostly contains samples with a Silhouette score well above average.

Figure 4.16 left specifically represents the Silhouette plot that resulted from clustering the dataset in Figure 4.15 with a 3-components GMM. We initially chose this number of components to imitate the three classes in human NREM sleep and to verify the hypothesis that also mice possess three stages of NREM sleep. In the case of Figure 4.16 left, the average Silhouette scores for each cluster and for the whole dataset are 0.467, 0.195, 0.220

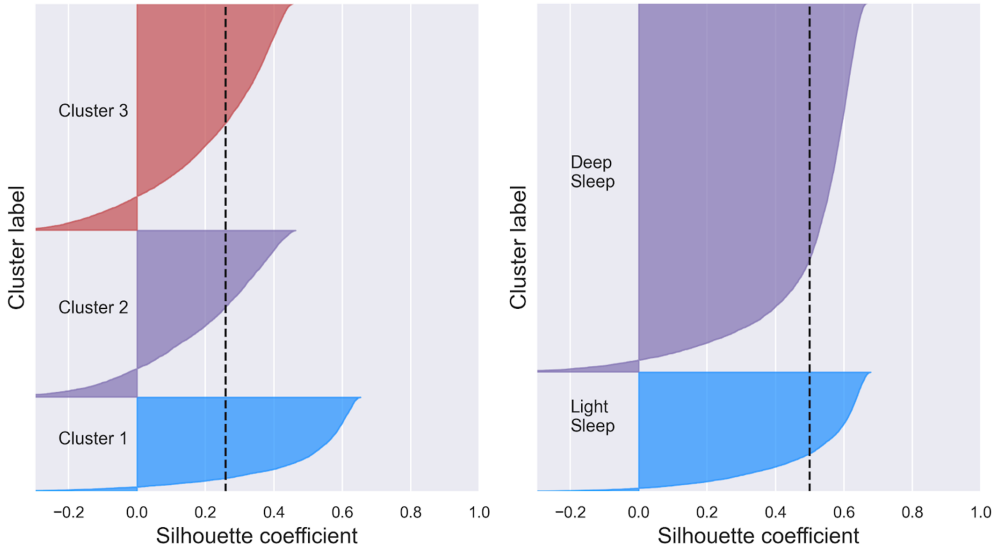

Figure 4.16: **Silhouette plots**. The plots were obtained on the same example dataset. They highlight the scores of the samples in each cluster and the dataset average. **Left**: scores using three clusters. **Right**: scores using 2 clusters.

and 0.260, respectively. These numbers confirm that the average Silhouette score for cluster 1 is much higher than the ones of clusters 2 and 3. The plot and the low average values indicate that the partition into clusters 2 and 3 is somewhat artificious, with many samples misclassified between the two clusters.

Given these results from the Silhouette analysis and the fact that similar results are found throughout all the different sessions (Figure 4.17), we decided to continue by using just two clusters.

The corresponding Silhouette plot of the example dataset when using a 2-components GMM is plotted in Figure 4.16 right.

The clusters are much more defined and both contain the majority of samples with an above-average Silhouette score. Also the average scores per cluster and of the dataset are much higher, being 0.506, 0.498 and 0.500. Importantly, the average scores of the two clusters are almost identical, sign that all the samples are generally well classified.

The reason for the names of the clusters in Figure 4.16 right comes from the fact that we observed that, throughout the different sessions, one cluster was always corresponding to low values of power and amplitude and high values of SE, while the contrary was true for the other cluster. The situation is shown in Figure 4.18. Given the description of the signals in Figure 4.12 and their PSDs, it seemed natural to assign the name "Deep sleep" to the cluster identified by high power and amplitude, in analogy with human sleep stages,

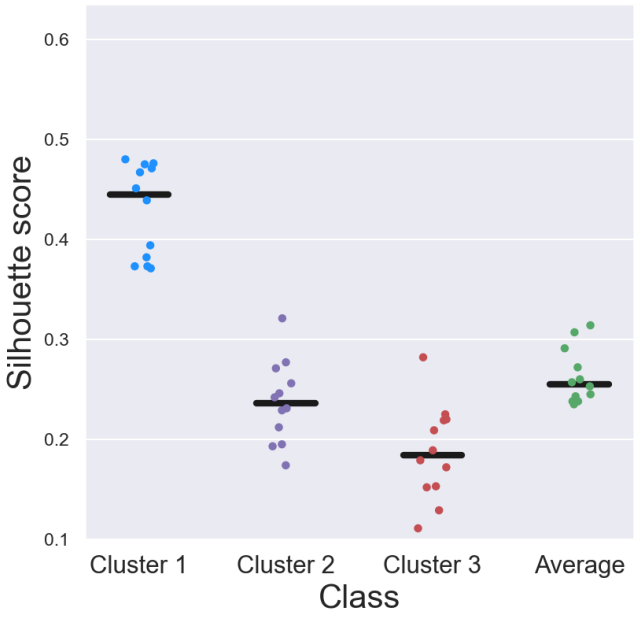

Figure 4.17: **Silhouette scores for n=13 sessions using 3 clusters**. Clusters 2 and 3 consistently show lower Silhouette scores.

where deeper stages, like N2 and N3, are characterized by increased amplitude and power of the EEG signals. In contrast, the other cluster is identified as "Light sleep", to indicate a more transient state, characterized by low power waves and mixed frequency activity and an increased SE.

4.3.4 Optimization of decision threshold

For the task of separating REM from NREM sleep, we took the approach of adapting the decision boundary based on which posterior probability maximized the F1-score calculated with the labels provided by the manual scoring and we used the same probability threshold for every session. For this task, instead, we took the approach of choosing the threshold value of the posterior probability of classifying "Light sleep", $p(\text{Light Sleep}|\boldsymbol{x})$, as the one that maximizes the average Silhouette score of the dataset. Importantly, this procedure was repeated independently for each dataset and was not used as an average value across the datasets, as we did for the REM sleep task. Again, we constructed a grid of values for the probability threshold and computed the resulting average Silhouette scores for each threshold.

An example output of this procedure is shown in Figure 4.19.

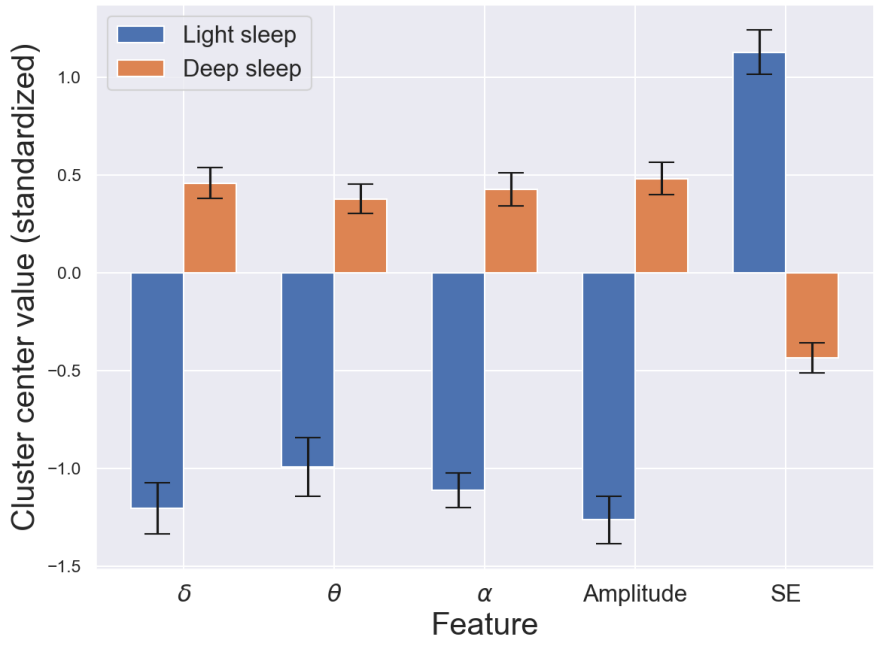

Figure 4.18: **Average centers for n=13 sessions**. Each session was fitted with a 2-component GMM, yielding the centers of the Gaussians distributions. The bars represent the (standardized) value of the center projected on each feature, colored by cluster. Black error bars are SEM.

Figure 4.19 left shows the values of the Silhouette scores changing as a function of the threshold. There are a few comments that can be made about this plot:

1. For the majority of threshold values, up to $0.90 - 0.95$, the trend of the three curves is approximately linear and a non-linear behavior appears just for very high values of threshold.
2. The average Silhouette score follows more closely the trend of the Deep Sleep cluster score, because it contains the higher proportion of samples, usually around $70 - 80\%$ of the total.
3. The average Silhouette score is a balance between the high increase of the score of the Light Sleep cluster when increasing the threshold and the corresponding slow decrease of the score of the Deep Sleep cluster. The difference between the magnitude of the two derivatives is balanced by the different proportions of samples in the two clusters.

Figure 4.19 right, instead, shows how the proportion of samples in the Light Sleep cluster decreases as the threshold increases. Also in this case the trend of the curve is approximately linear for the majority of threshold values and in this region the proportion

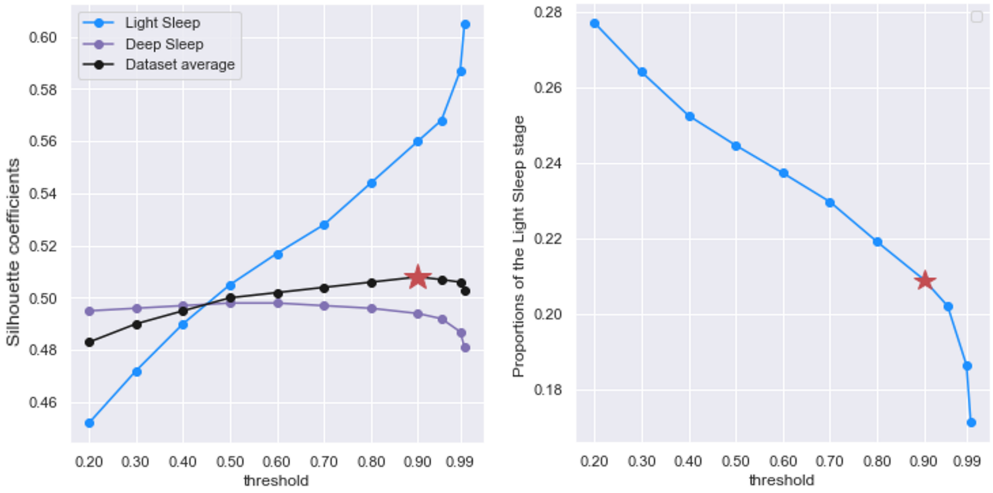

Figure 4.19: **Optimization procedure for the decision threshold**. **Left**: per-cluster and dataset-average Silhouette scores as a function of the probability threshold used to classify the Light Sleep cluster. The red star indicates the point of maximum in the average Silhouette score, in this case it is reached for a threshold of 0.90. The threshold values sampled were $[0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95 0.99, 0.999]$. **Right**: proportion of samples in the light sleep cluster as a function of the probability threshold.

decreases by approximately 5%. The behaviors shown in Figure 4.19 were consistently found throughout the different datasets, sign that the clustering procedure is robust.

4.3.5 Performance of the clustering algorithm

A GMM model with two components was fitted to $n = 13$ datasets, excluding the outlier session in the REM sleep detection. For each session, the threshold was optimized based on the method described above. Figure 4.20 shows the average and per-class Silhouette scores obtained for all the datasets.

All the scores computed were above 0.425, which is quite high and promising. Another very positive fact is that the average scores are high and they span a very compact range, indicating that the algorithm reaches approximately the same performance on each different dataset. The highest absolute scores are achieved by the Light Sleep cluster, but they also possess a higher spread. The higher variance is probably attributable to the fact that different datasets can contain very different proportions of Light Sleep and usually lower proportions have higher scores. On the other hand, the Deep Sleep class' scores are generally lower, but more compact, sign that a cluster with more samples is less sensitive to changes in the proportions.

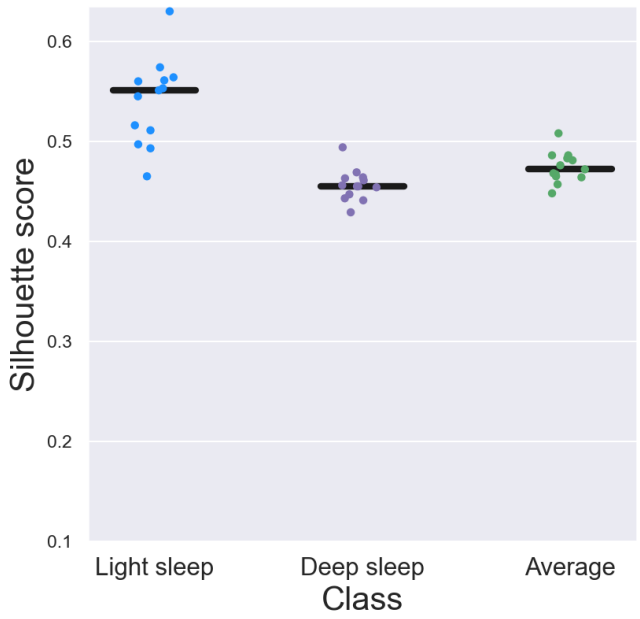

Figure 4.20: **Summary Silhouette scores of n=13 sessions**. Per-class and average Silhouette scores shown as strip plots. Each point represents a different session and the black lines the mean value.

4.3.6 Post-processing and creation of intervals

We then proceeded to post-process the created intervals, in order to assure more continuity and consistency of the different stages. In particular, we decided to take inspiration from the rule present in the AASM manual that states to continue to score an epoch as N2 sleep stage, even if its characteristics are more similar to N1 sleep stage [37]. This means that, if some epochs have been already identified as N2 sleep stage, the successive epochs are still scored as such, even in the presence of mixed-frequency and low-amplitude signals, except if there is an arousal. We converted this indication in the following rules to be applied to the raw epochs classified by the algorithm:

1. We merged Light Sleep intervals separated by just one Deep Sleep epoch.
2. We merged Deep Sleep intervals that are separated by at most 3 epochs (12 seconds).
3. We retained just Deep Sleep intervals lasting at least 4 epochs (16 seconds)

4.3.7 Comparison with human NREM sleep

After we scored and post-processed the epochs of each dataset, we proceeded to extract parameters and quantities related to the sleep stages found. To check the consistency and

conservation of these parameters, we compared the analysis of this experiment with the human sleep data described in section 4.1. Since the labels used in humans include the distinction in stages N1,2 and 3, we merged stages N2 and N3 into a single stage and we called it "Deep Sleep", in analogy with the naming used in the mice case; correspondingly, we called the N1 stage "Light Sleep".
The subsequent analyses closely follow the ones presented in Lacroix et al [47]. For all the analyses we used data from $N = 13$ mice ($n = 13$ sessions) and $N = 13$ human subjects ($n = 13$ recordings).

The first sleep parameter that we computed were the proportions of the different stages during the recordings. For each subject and for each mouse dataset, we counted the number of epochs scored as Wakefulness, Light Sleep, Deep Sleep and REM sleep, we divided them by the total number of epochs and multiplied the resulting numbers by 100, in order to obtain percentages.

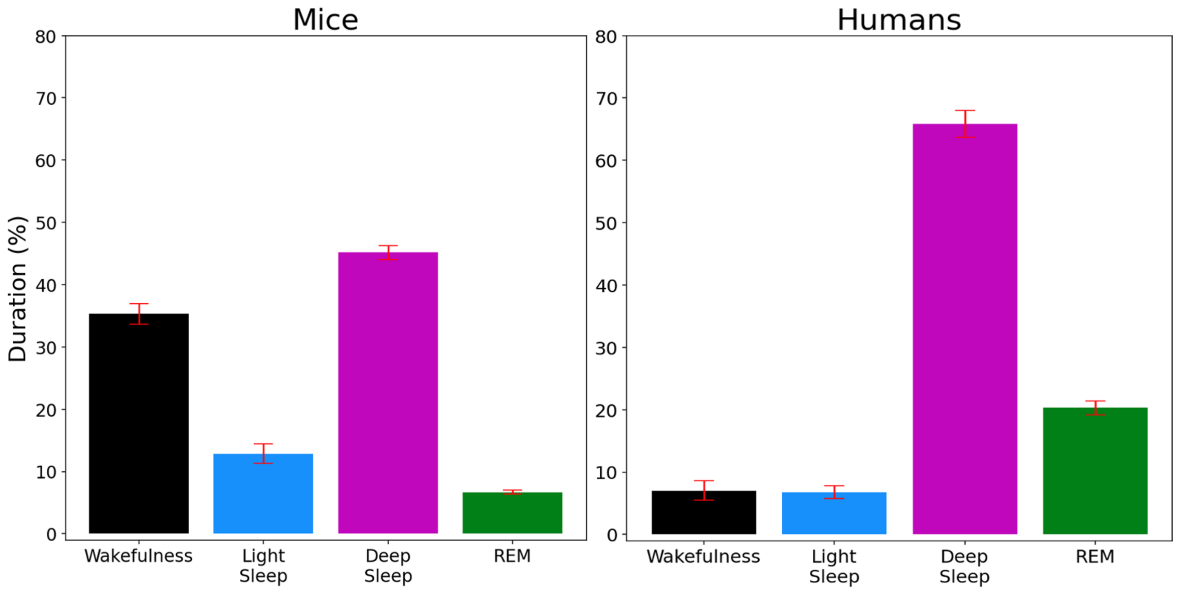

Figure 4.21: **Proportions of time spent in the different states in the two species**. Experiments lasted approximately 21 hours for mice and 8 for humans. The Red error bars are SEM.

The results in Figure 4.21 show that the proportion of Wakefulness changes drastically between the two species, but this fact is due to the difference in the two experiments. Humans recordings last 8 hours during the night, mice's ones last approximately 21 hours and cover both the dark and the light phases. In this case, it is natural that mice's datasets contained

a bigger proportion of wakefulness, as would happen if humans were recorded during the day.

To have a more meaningful comparison, we restricted this analysis to just the total sleep time (TST), i.e. the time spent in the different stages of sleep, disregarding the Wakefulness epochs.

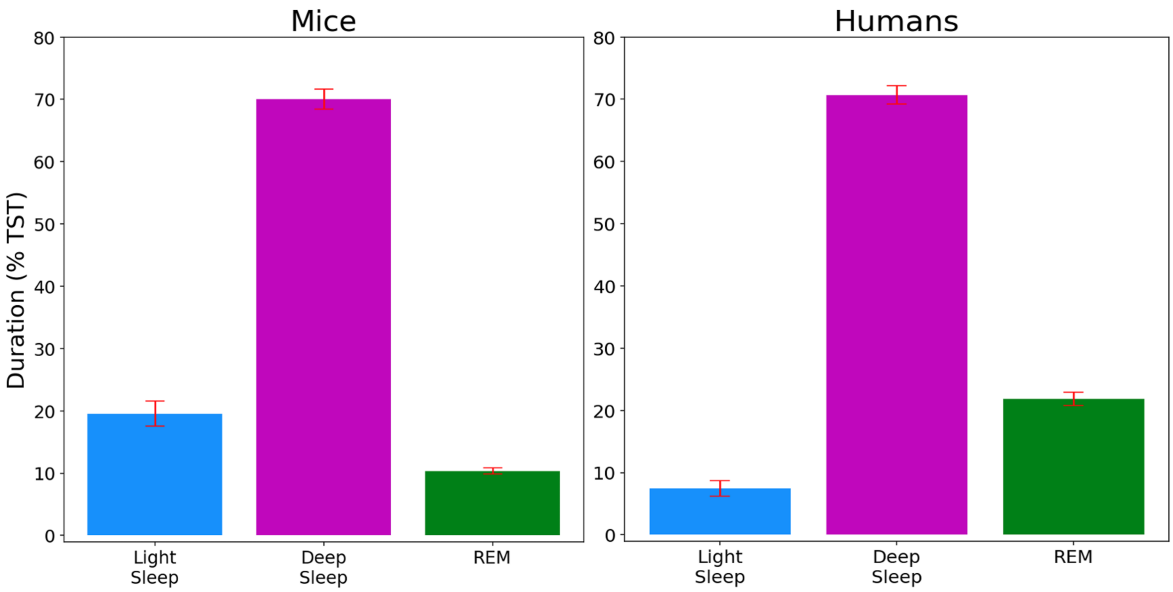

Figure 4.22: **Proportions of sleep time spent in the different stages in the two species**. The Red error bars are SEM.

Figure 4.22 shows that, considering just the TST, the proportion of Deep Sleep is almost identical in both species, accounting for a little more than 70% of the TST. What differentiate the two species are the proportions of Light and REM sleep. In mice, the average amount of REM sleep is less than half of the equivalent in humans, with the two proportions being 10.4% and 21.8%, respectively. This difference in the REM sleep proportion is compensated by an increased amount of Light Sleep in mice, which accounts for 19.6% of the TST, as opposed to an average 7.5% in the human subjects.

The next sleep parameter we computed was the average episode duration for the different states. What can be highlighted in Figure 4.23 is that for both species the average duration of Light Sleep intervals is shorter than the duration of Deep Sleep ones. Also the ratio between the average durations of the two stages is comparable between the two species.

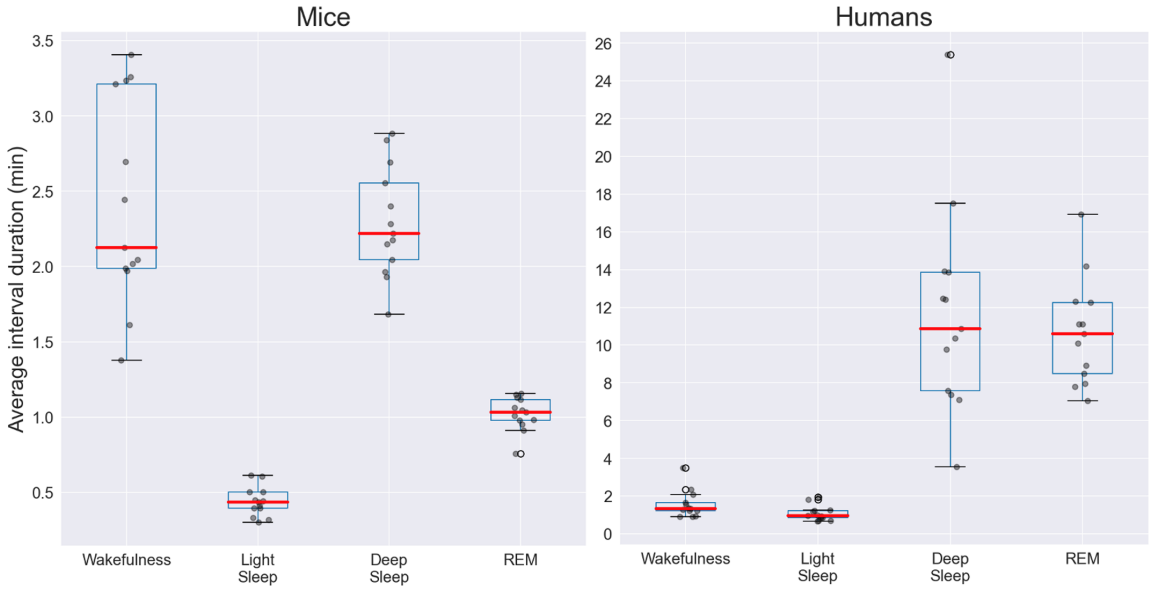

Figure 4.23: **Average interval duration for the different states in the two species**. Each dot represents a different mouse/subject and box plots are superimposed. Notice that the temporal scale of the intervals in the two species is very different.

Next, we investigated how the proportions of the different states changed during the recording. To approach the analysis, we divided the mice's recordings in blocks of 10 minutes and computed the proportion of each stage in the block.

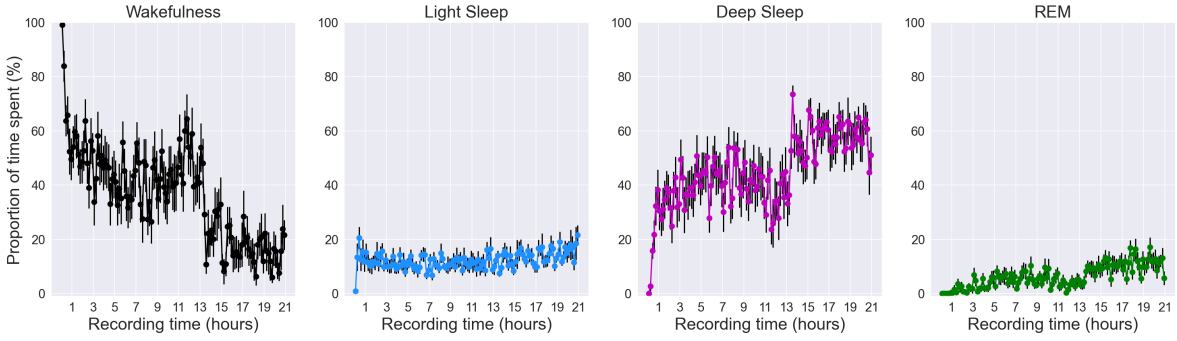

Figure 4.24: **Temporal dynamics of the different stages during mice experiments**. The normalization in the plot is done such that, fixing a certain timepoint, the sum of the duration across the 4 different stages is 100. Error bars are SEM.

Figure 4.24 shows that there is a sudden drop in the amount of wakefulness after approximately 13 hours from the beginning of the recording, with a corresponding increase in the amount of Deep Sleep. The cause of this behavior lies in the fact that lights were turned

on 13 hours after the start of the recording. Since mice are nocturnal animals, they tend to sleep more when lights are turned on. For this reason, and to have a better comparison between mice and humans experiments, we decided to restrict this analysis to the last 8 hours of recording in mice, from the 13^{th} to the 21^{st} hour of recording. This roughly corresponds to 6 a.m. and 2 p.m. of clock time.

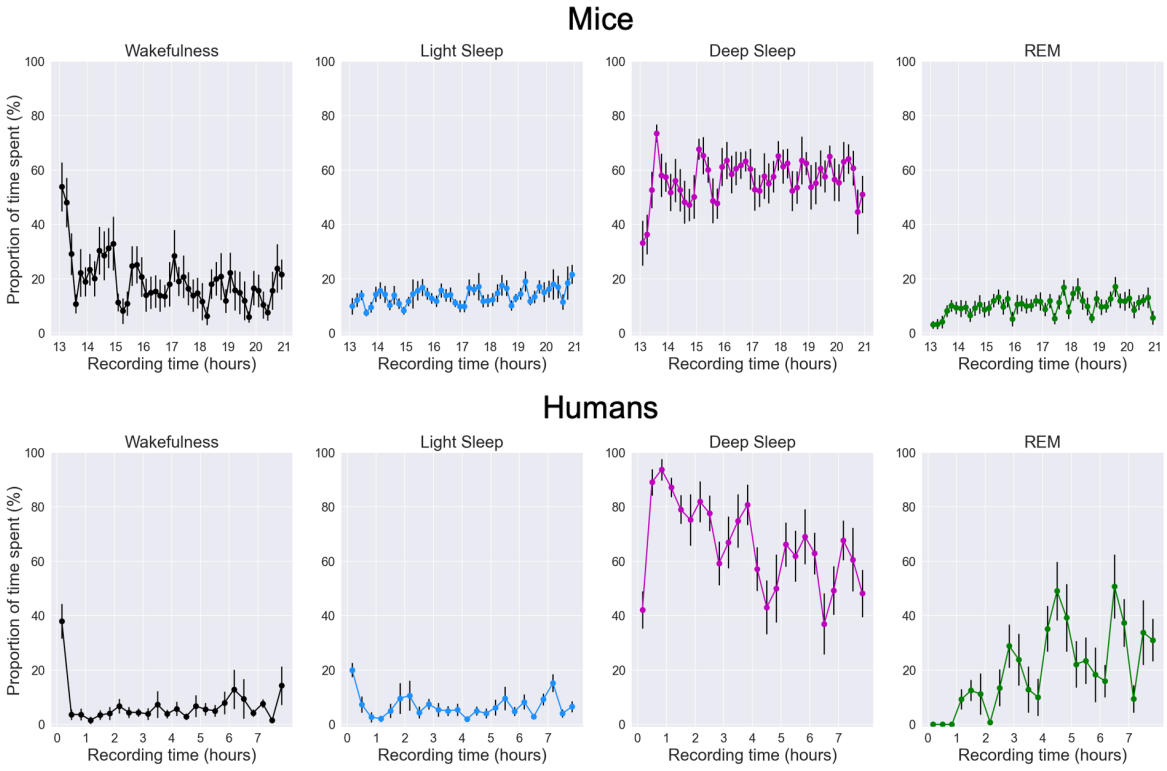

Figure 4.25: **Comparison between the temporal dynamics in the two species**. To account for the different temporal scales, the proportions in mice were calculated using 10-minutes blocks, while 20-minutes blocks were used in humans.

Figure 4.25 compares the time course of the four different stages in the two different species and we can notice some common characteristics. First, the amount of REM and Light sleep has a similar and increasing trend in both cases. Secondly, also Deep Sleep follows a similar trend in both cases, exhibiting a maximum at the beginning and progressively decreasing or remaining constant at later recording times. This behavior is an indication of the homeostatic regulation of sleep.

Figure 4.25 also shows that the different stages undergo a cycling structure. This is particularly evident from the time course of REM sleep in humans. This cycling structure

reflects the ultradian regulation of sleep. The ultradian cycle has different durations in different mammal species and lasts around 90 minutes in humans and 10 minutes in mice [39]. To check if this statement was true for our recordings, we defined a sleep cycle as the period between the end of a REM interval and the end of the next one. Cycles in mice were computed only in the light phase. By definition, every cycle contained only one REM sleep interval, but in general could include several Wakefulness, Light and Deep sleep intervals. Figure 4.26 shows the empirical distributions of the duration of one cycle in mice and in humans. Apart from cycles with a duration below 30 minutes in humans, which are non

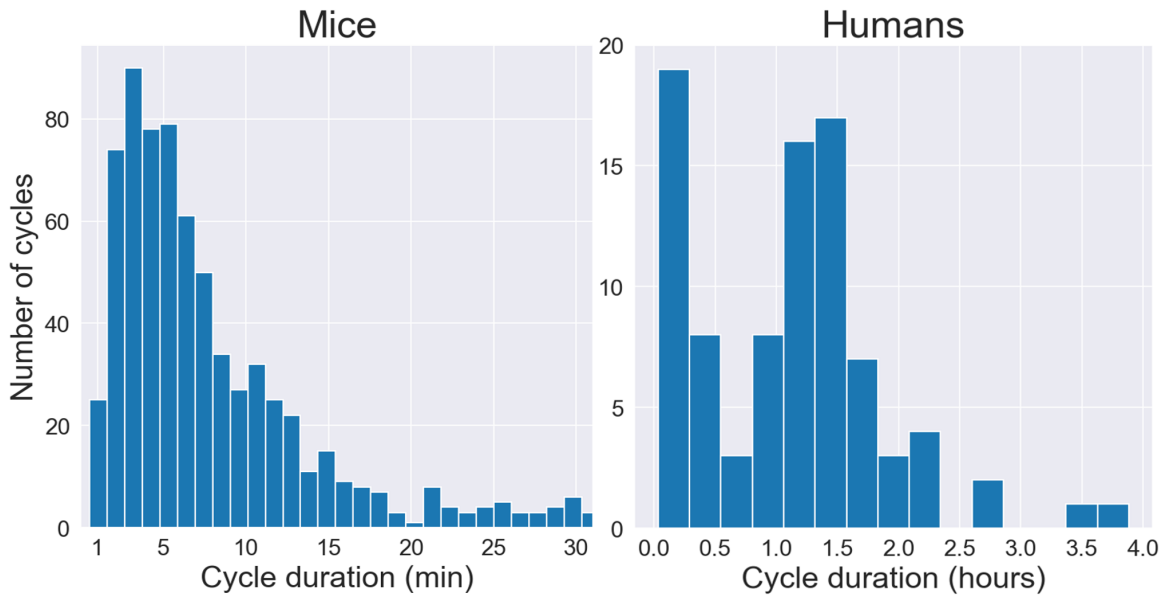

Figure 4.26: **Sleep cycle duration in the two species**. In the histograms all cycles from different mice/subjects are pooled together. Notice the different units on the x-axis. The mean and median values for mice are 9.26 and 6.07 minutes, while for humans are 1.45 and 1.19 hours.

physiological, cycles in humans' recordings have an average and a mode duration of about 1.5 hours, or 90 minutes, in accordance with our previous claim and the literature. On the other hand, the duration of a sleep cycle in mice has a mode at around $3-5$ minutes and an average duration of 9.3 minutes, again in accordance with the literature.

We further investigated the content of sleep cycles by computing the probability that a certain stage appears at a certain moment during the cycle. In order to do this, we defined a normalized time, going from 0 to 1, where 0 represents the beginning of a cycle and 1 its end, regardless of the duration of the cycle. In this way we could compare cycles with different duration because their normalized time coincides. To compute the probability of

occurrence of the different stages, we divided the normalized time into bins of 0.01 for mice and 0.05 for humans, in order to account for the different duration of the epochs. Then, for each cycle, we counted the overall number of occurrences of the different stages in every bin. Probabilities are then obtained by dividing by the total number of occurrences in a bin, such that the occurrence probabilities for different stages in every bin sum up to 1.

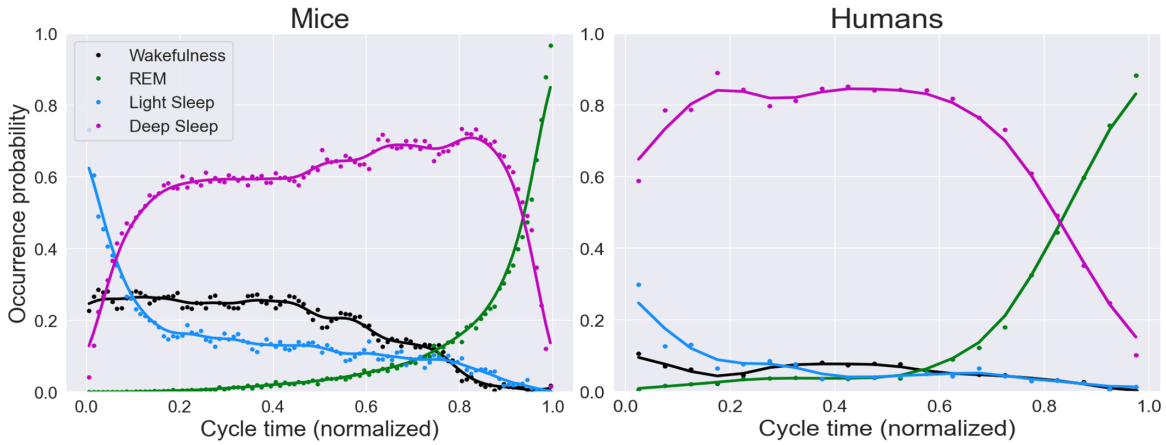

Figure 4.27: **Probability of occurrence of the different stages during a cycle**. Each dot represents the probability in each bin, while the solid line is obtained through a Gaussian kernel interpolation. To account for the different number of cycles used in mice (726) and humans (64), we used 100 bins for the former and 20 for the latter.

Figure 4.27 highlights that the occurrence probability of the different stages during a cycle in humans and mice displays many similarities and some differences.

Sleep cycles in both humans and mice present REM occurrence probability that is almost zero for the entire cycle and then increases very fast towards the end, arriving at 1. This is natural, if one thinks that, by definition, cycles must end with a REM interval.

Deep Sleep has the highest probability of occurrence for the central and major part of the cycle, but the behavior at the beginning of the cycle is very different in the two species, with the probability in humans being already very high.

In mice, Light Sleep has the highest occurrence probability at the beginning of the cycle, which then decreases as sleep progresses. At a certain point, Deep Sleep probability exceeds the Light Sleep one and stays approximately constant. Also Wakefulness probability remains quite high throughout the cycle. In humans, on the other hand, Light Sleep and Wakefulness probabilities are from the beginning very low and remain as such throughout the cycle. These differences may stem primarily from the very different occurrence of Wakefulness episodes in the two species and from the lower presence of Light Sleep in humans.

Moreover, the total number of cycles considered in humans is quite low, 64 vs 726 used in mice, fact that decreases the statistical significance of the analysis in humans.

Finally, transition matrices representing the number of transitions between the different stages were computed. For computing the transition matrices, intervals of the different stages were created and a transition was counted only between successive intervals. Therefore, by construction, the matrices have all zeros on the diagonal. One transition matrix was computed for each mouse and for each subject. We then normalized them by dividing the entries by the total number of counts in each matrix such that the sum of all the matrix elements sums up to 1. The normalization procedure was done in order to compare matrices from different recordings and between the species. Then, we computed the average values and we imposed a "transition threshold", such that only transitions accounting for more than this threshold were kept. The threshold was chosen as the value of the normalized number of transitions from Wakefulness to REM sleep. The rationale behind this decision is based on the evidence that humans [35], as well as other mammals, normally enter REM sleep from NREM sleep and not from Wakefulness. Thus, we considered those transitions as not meaningful and we disregarded as well all the transitions below this threshold.

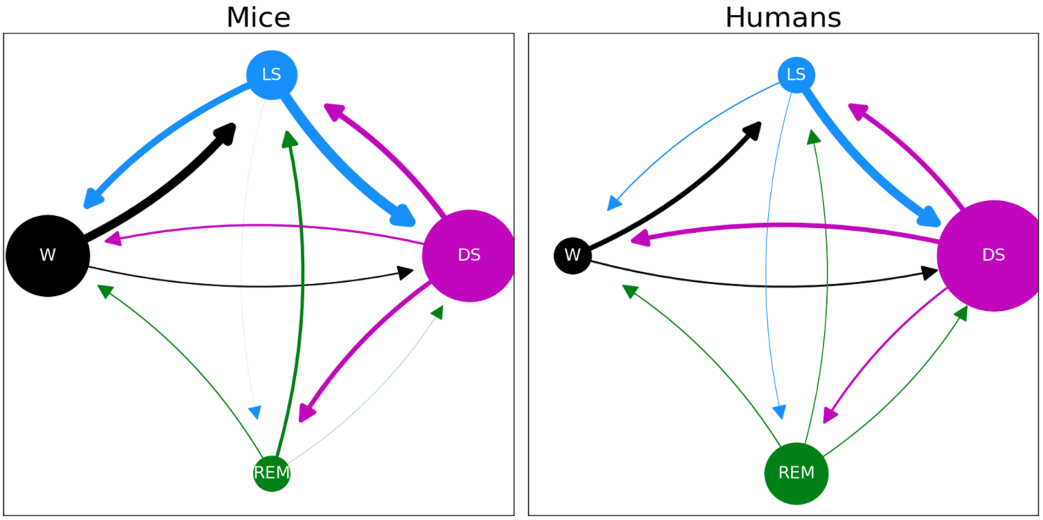

Figure 4.28: **Transition matrices normalized by total count**. The size of the circles represents the average proportions of the stages. The width of the arrows represents the normalized number of transitions from one stage to the other.

The two plots in Figure 4.28 are very similar. In particular, they share exactly the same qualitative transitions, although a quantitative analysis shows some differences. For example,

the amount of transitions from Wakefulness to Light Sleep and vice versa is much higher in mice, although this can be expected, given that mice have a polyphasic sleep. On the other hand, the amount of transitions from and to Deep Sleep shows little differences.
The transition matrices in Figure 4.28 have the limitation that they inflate the importance of transitions in stages that occupy a high proportion of recording time, as is the case of Wakefulness in mice.
In order to take into account the different proportions of the stages, we also constructed transition matrices normalized by using the total number of transitions in every row, that is, from a given stage. In this way, entries of the matrix can be interpreted as transition probabilities $P(s_t|s_{t-1})$, where t represents the current time and $t-1$ the previous one, with the property $\sum_{s_t} P(s_t|s_{t-1}) = 1$.

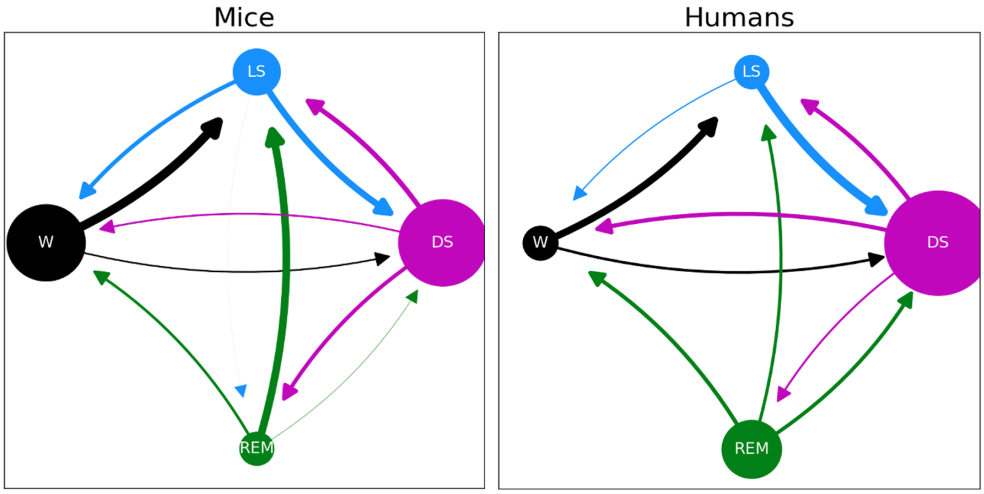

Figure 4.29: **Transition matrices normalized by row count**. The size of the circles and the width of the arrows have the same meaning as in Figure 4.28.

The two matrices in Figure 4.29 are even more similar than those in Figure 4.28. The only noticeable differences are represented by a lower transition probability from REM to Deep sleep in mice, with a corresponding probability increase from REM to Light sleep, and a higher transition probability from Light sleep to Wakefulness in mice. This last characteristic may indicate that the awakening threshold is lower in mice than in humans and more frequent and brief awakenings are observed in mice.

CHAPTER 5

Conclusions

In the present thesis we have sought to find and develop suitable data analysis techniques to tackle common problems encountered in computational neuroscience research. In particular our aim was to automatically classify awake behaviors and sleep phases in rodents. For this purpose, we have used Machine Learning and signal processing techniques described in chapter 2 and have analyzed long recordings of accelerometer and, mostly, EEG time series. The use of accelerometer signals for the classification of animal behavior is an emerging field.
This thesis thus describes supervised methods for the analysis of rodent behaviors in chapter 3, as well as unsupervised approaches to the classification of rodent sleep, including a further subdivision of the NREM stage into "light" and "deep" sleep in chapter 4.

The need of automatic techniques to classify different behaviors and sleep stages during neuroscience experiments is mainly due to three interrelated reasons:

1. These classification tasks are often very time consuming and sometimes rely on the personal judgment of the human scorer.
2. Although well trained, humans are always prone to errors and this manifests to a higher extent when scoring very long recordings.
3. Reasons 1 and 2 particularly create a problem as neuroscientists collect larger datasets.

Concerning the behavioral classification tasks chapter 3, we have found that it is possible to automatically classify the Wake/Sleep state of mice with very high accuracy by means of an accelerometer. We have also found that valuable insights can be gained for other more complex and stereotypical behaviors (e.g. grooming, rearing). Although the classification

metrics for these more complex behaviors were not as good as the ones of sleep and activity, promising results were achieved.

We have taken an approach similar to the ones previously described in [91] and [80] and we have split the signals into epochs and from each one we have extracted different relevant features and tested the performance with different classifiers, such as Logistic Regression, Random Forest and Neural Networks. Different aspects are worth to emphasize, also with respect to previous studies.

From one side, the data augmentation technique we have adopted helped to improve the performance of the classifiers. In comparison with the original paper [84], we have proposed a different approach that relies on preserving the empirical distributions of the different features as much as possible. For comparison, we have also trained a Random Forest model on non-augmented data. This baseline model did not classify correctly any of the rearing epochs in the test set and achieved just 0.6 F1-score on the grooming class. This indicates that, without the data augmentation procedure, the classifiers are not able to properly learn to classify the minority classes. With respect to this baseline model, both the RF and the NN models perform much better, both on the rearing and the grooming classes.

Sunderam et al. [44] used a MEMS accelerometer to classify different behaviors in rats, including feeding and grooming, where they achieved 0.56 F1-score. Compared to their results, we have achieved a much higher classification performance on the grooming class, with a 0.77 F1-score (Table 3.11).

In another study, Venkatraman et al. [92] classified different behaviors in mice, including grooming and rearing, exploiting a Neural Network. Our results seem worse than theirs, especially concerning rearing, but they used the True and False positive rates as metrics (TPR and FPR respectively). While the TPR is another way of naming the recall, the FPR is not directly comparable to the Precision as it takes into account the number of true negatives classified. We think that this metric is not ideal to measure the performance for classes with a low number of samples, because the number of true negatives is in general much higher than the number of false positives. Nevertheless, their reported TPR on rearing is much higher than ours and the FPR is comparable, while the TPR on grooming is slightly higher than ours, but our FPR is 5-fold lower, indicating that our overall classification of grooming can be better.

Therefore, compared to our results, the reported performance in similar studies using accelerometers in rodents is higher in some cases (Venkatraman et al. regarding rearing) and lower in others (Venkatraman et al. and Sunderam et al. regarding grooming).

For what concerns the successive investigation of sleep in chapter 4, we have obtained very positive results for the separation of sleep into NREM and REM stages and novel insights into a further partition of NREM sleep in a "light sleep" stage and a "deep sleep" stage.

We took an unsupervised approach for the classification of sleep into NREM and REM stages. There were three rationales behind this choice:

1. We wanted to find patterns in the data that could characterize different brain states during sleep, regardless of the naming conventions or the definitions, and cluster them. Then, in a second step, associate them according to the current knowledge.
2. The classification of epochs is very time consuming and a certain level of experience and practice is needed by the human scorer.
3. Usually in clinical and preclinical research, data shows disrupted sleep patterns and/or alterations of the characteristics of the different stages, for example power shifts in the different power bands. In this regard, we aimed at finding analyses that would be robust even in these cases.

We have obtained very positive results for the identification of REM sleep in section 4.2. Apart from the results in Table 4.1, we have also computed the so-called specificity for the REM class, which is equal to 0.990 ± 0.001 for the entire dataset (except the outlier session). Compared to other unsupervised approaches in mice, such as the ones presented by Yaghouby et al. [87] and Sunagawa et al. [40], we have reached higher sensitivity (recall) and specificity for the REM class, with both metrics being at least $2 - 3\%$ higher in our approach.

Also compared to supervised approaches our model performs very well. Rytkönen et al. [41] used naive bayes classifiers trained on a subsample of data for each session and reached average sensitivity and specificity of 91.6% and 95.3%. While the sensitivity is approx. 2% higher than ours, the specificity is 4% lower, suggesting that the number of false positive REM samples was much higher in their study. Instead, we were able to lower the false positive REM epochs by optimizing the decision boundary of the classification and, correspondingly, increasing the precision while maintaining the recall quite high. In another recent study, Barger et al. [42] exploited a deep convolutional neural network (CNN) to score directly images of the spectrogram into Wake, NREM and REM. From one side, they obtained REM metrics approx. 10% higher than ours, highlighting the potential and impact

of deep learning in sleep scoring applications. On the other side, besides requiring labelled training data, their algorithm requires that a subsample of epochs of each recording is manually scored before the algorithm can be applied. This can be a bottleneck if the number of datasets to be analyzed is very large.
All in all, we can say that our Gaussian Mixture model with the decision boundary optimization step reaches better results than other unsupervised approaches and also of other supervised approaches using simple classifiers.
It should in any case be noted that a total agreement with manual scorers is very difficult to achieve, but it is also not desirable, since it is a sign of probable overfitting: an algorithm that achieves total accuracy may have learned to just "imitate" the manual scoring, without actually being able to generalize to other situations. In many reported studies, the total agreement between different scorers on the same dataset is on the order of $90 - 95\%$ [93] [41], or even lower than 90% [94]. Therefore also our algorithm reaches comparable performances when compared to a human scorer.

In section 4.3 we have sought an approach to better partition the NREM stage previously identified. Figure 4.12 shows that, already by visual inspection, EEG signals during NREM sleep present strong differences in amplitude and spectral composition. We have investigated whether we could find consistent differences in the EEG signals during NREM sleep in mice. For this purpose we have used the power content in three different frequency bands ($1 - 3$, $3 - 6$ and $6 - 18$ Hz), the amplitude and the spectral entropy of the frontal EEG signal as features for a Gaussian mixture model, which we have then used as a clustering algorithm to identify consistent patterns in the epochs. In particular, we have used the Silhouette score as a metric to identify cluster-specific properties and the overall performance of the clustering. We have modelled the distributions with two components instead of three because the Silhouette scores of two of the clusters were consistently lower than the third one and showed no significant evidence of representing two distinct clusters, at least with the features we have chosen (Figure 4.17). We have finally introduced an optimization for the decision boundary also for this analysis, with the objective of maximizing the overall Silhouette score of each dataset. We consistently found a cluster containing epochs with reduced power in the three frequency bands, reduced amplitude and increased spectral entropy, relative to the other one (Figure 4.18). This suggested us to call the former cluster "light sleep", in comparison with human N1 stage, and the latter cluster "deep sleep", in comparison with human N2 and N3 stages.
The comparison of extracted sleep-related parameters between mice recordings and humans

recordings showed many similarities, pointing towards the evidence that the newly found stages in mice may actually be comparable to human ones. In particular, the fraction of the sleep time spent in deep sleep by the two species is nearly identical, the temporal dynamics of the time spent in the different stages during the night shows many similarities and even the detailed structure of sleep cycles shows many conserved features, although a bigger sample of human sleep cycles would be needed for an appropriate comparison. Finally, the dynamical structure of sleep seems conserved also for what concerns the transitions from one stage to the others. Transitions are qualitatively the same for the two species and normalized transition probabilities show quantitative similarities. Our analyses show also consistency with the ones reported by Lacroix et al. [47], who, however, identified three NREM stages and compared those directly with humans' N1, N2 and N3.

Given that there is no general consensus on if and how to classify different stages within NREM sleep, our contribution has been to show that there are relevant biological differences within NREM sleep in mice, which can be distinguished exploiting Machine Learning models and an accurate selection of sleep-relevant features. We think that a more detailed partition of NREM sleep in mice should be considered in every study, especially in the clinical ones, to improve translation and back-translation efficiency between the two species.
It remains to be investigated whether, by using additional or different features, the Deep sleep cluster we identified can be further separated into two stages and if these two stages can be compared to N2 and N3 human stages.

Bibliography

[1] T. Roth, "Insomnia: definition, prevalence, etiology, and consequences," *J Clin Sleep Med*, vol. 3, no. 5 Suppl, pp. 7–10, Aug 2007.

[2] "2018 alzheimer's disease facts and figures," *Alzheimer's & Dementia*, vol. 14, no. 3, pp. 367–429, 2018. [Online]. Available: https://alz-journals.onlinelibrary.wiley.com/doi/abs/10.1016/j.jalz.2018.02.001

[3] P. D. Meek, E. McKeithan, and G. T. Schumock, "Economic considerations in alzheimer's disease," *Pharmacotherapy: The Journal of Human Pharmacology and Drug Therapy*, vol. 18, 1998.

[4] S. E. Levy, D. S. Mandell, and R. T. Schultz, "Autism," *The Lancet*, vol. 374, no. 9701, pp. 1627–1638, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140673609613763

[5] K. Lyall, L. Croen, J. Daniels, M. D. Fallin, C. Ladd-Acosta, B. K. Lee, B. Y. Park, N. W. Snyder, D. Schendel, H. Volk, G. C. Windham, and C. Newschaffer, "The Changing Epidemiology of Autism Spectrum Disorders," *Annu Rev Public Health*, vol. 38, pp. 81–102, Mar 2017.

[6] M. R. Irwin and M. V. Vitiello, "Implications of sleep disturbance and inflammation for Alzheimer's disease dementia," *Lancet Neurol*, vol. 18, no. 3, pp. 296–306, 03 2019.

[7] "Action potential," https://en.wikipedia.org/wiki/Action_potential.

[8] A. L. HODGKIN and A. F. HUXLEY, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J Physiol*, vol. 117, no. 4, pp. 500–544, Aug 1952.

[9] G. Buzsáki, C. A. Anastassiou, and C. Koch, "The origin of extracellular fields and currents–EEG, ECoG, LFP and spikes," *Nat Rev Neurosci*, vol. 13, no. 6, pp. 407–420, May 2012.

[10] http://mugup.in/qna/Class11BiologyNeuralControlandCoordination/9623.

[11] M. Mascaro and D. J. Amit, "Effective neural response function for collective population states," *Network: Computation in Neural Systems*, vol. 10, no. 4, pp. 351–373, 1999, pMID: 10695764. [Online]. Available: https://doi.org/10.1088/0954-898X_10_4_305

[12] T. D. Sanger, "Neural population codes," *Current Opinion in Neurobiology*, vol. 13, no. 2, pp. 238–249, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959438803000345

[13] J. K. Chapin, "Using multi-neuron population recordings for neural prosthetics," *Nature neuroscience*, vol. 7, no. 5, pp. 452–455, 2004.

[14] G. Buzsáki, *Rhythms of the brain*. Oxford: Oxford University Press, 2006, oCLC: 984823562.

[15] P. L. Nunez, R. Srinivasan *et al.*, *Electric fields of the brain: the neurophysics of EEG*. Oxford University Press, USA, 2006.

[16] Y. Kajikawa and C. E. Schroeder, "How local is the local field potential?" *Neuron*, vol. 72, no. 5, pp. 847–858, Dec 2011.

[17] http://www.scholarpedia.org/article/Volume_conduction.

[18] http://www.scholarpedia.org/article/Source_localization.

[19] https://www.biofeedback-tech.com/articles/2017/9/19/quantitative-eeg-and-databases-meg5l.

[20] M. X. Cohen, *Analyzing neural time series data: theory and practice*. MIT press, 2014.

[21] "International Federation of Societies for Electroencephalography and Clinical Neurophysiology," *Electroencephalogr Clin Neurophysiol*, vol. 37, no. 5, p. 521, Nov 1974.

[22] G. Buzsáki and B. O. Watson, "Brain rhythms and neural syntax: implications for efficient coding of cognitive content and neuropsychiatric disease," *Dialogues Clin Neurosci*, vol. 14, no. 4, pp. 345–367, Dec 2012.

[23] I. V. Zhdanova, S. Y. Wang, O. U. Leclair, and N. P. Danilova, "Melatonin promotes sleep-like state in zebrafish," *Brain research*, vol. 903, no. 1-2, pp. 263–268, 2001.

[24] J. C. Hendricks, S. M. Finn, K. A. Panckeri, J. Chavkin, J. A. Williams, A. Sehgal, and A. I. Pack, "Rest in drosophila is a sleep-like state," *Neuron*, vol. 25, no. 1, pp. 129–138, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0896627300808776

[25] J. M. Siegel, "Chapter 10 - sleep in animals: A state of adaptive inactivity," in *Principles and Practice of Sleep Medicine (Sixth Edition)*, sixth edition ed., M. Kryger, T. Roth, and W. C. Dement, Eds. Elsevier, 2017, pp. 103–114.e4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323242882000106

[26] H. Moldofsky, F. Lue, J. Davidson, and R. Gorczynski, "Effects of sleep deprivation on human immune functions," *The FASEB Journal*, vol. 3, no. 8, pp. 1972–1977, 1989.

[27] M. H. Bonnet, "Effect of Sleep Disruption on Sleep, Performance, and Mood," *Sleep*, vol. 8, no. 1, pp. 11–19, 03 1985. [Online]. Available: https://doi.org/10.1093/sleep/8.1.11

[28] S. P. Drummond and G. G. Brown, "The effects of total sleep deprivation on cerebral responses to cognitive performance," *Neuropsychopharmacology*, vol. 25, no. 1, pp. S68–S73, 2001.

[29] H. Piéron, *Le problème physiologique du sommeil*. Masson, 1912. [Online]. Available: https://books.google.ch/books?id=PCoXAAAAYAAJ

[30] W. F. Flanigan, C. P. Knight, K. M. Hartse, and A. Rechtschaffen, "Sleep and wakefulness in chelonian reptiles. I. The box turtle, Terrapene carolina," *Arch Ital Biol*, vol. 112, no. 3, pp. 227–252, Jul 1974.

[31] I. Tobler, "Evolution of the sleep process: A phylogenetic approach," *Exp. Brain Res. Suppl*, vol. 8, pp. 207–226, 1984.

[32] R. Conduit and S. R. Robinson, "The neurobiology of sleep: neural circuitry and mechanisms," *Sleep Medicine*, 2017.

[33] T. Deboer, "Behavioral and electrophysiological correlates of sleep and sleep homeostasis," *Curr Top Behav Neurosci*, vol. 25, pp. 1–24, 2015.

[34] P. Achermann and A. A. Borbély, "Mathematical models of sleep regulation," *Front Biosci*, vol. 8, no. 6, p. 1064, 2003.

[35] H. Zepelin, J. M. Siegel, and I. Tobler, "Chapter 8 - mammalian sleep," in *Principles and Practice of Sleep Medicine (Fourth Edition)*, fourth edition ed., M. Kryger, T. Roth, and W. C. Dement, Eds. Elsevier, 2005, pp. 91–100.

[36] T. Deboer, "Sleep homeostasis and the circadian clock: Do the circadian pacemaker and the sleep homeostat influence each other's functioning?" *Neurobiology of Sleep and Circadian Rhythms*, vol. 5, pp. 68–77, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2451994417300068

[37] C. Iber, S. Ancoli-Israel, A. J. Chesson, and S. Quan, "The aasm manual for the scoring of sleep and associated events: rules, terminology and technical specifications," *1st ed. Westchester, IL: American Academy of Sleep Medicine*, 2007.

[38] K. Susmáková, "Human sleep and sleep eeg," 2005.

[39] F. Weber and Y. Dan, "Circuit-based interrogation of sleep control," *Nature*, vol. 538, no. 7623, pp. 51–59, 2016.

[40] G. A. Sunagawa, H. Séi, S. Shimba, Y. Urade, and H. R. Ueda, "Faster: an unsupervised fully automated sleep staging method for mice," *Genes to cells : devoted to molecular & cellular mechanisms*, vol. 18, no. 6, pp. 502–518, Jun 2013, 23621645[pmid]. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/23621645

[41] K.-M. Rytkönen, J. Zitting, and T. Porkka-Heiskanen, "Automated sleep scoring in rats and mice using the naive Bayes classifier," *Journal of Neuroscience Methods*, vol. 202, no. 1, pp. 60–64, 2011.

[42] Z. Barger, C. G. Frye, D. Liu, Y. Dan, and K. E. Bouchard, "Robust, automated sleep scoring by a compact neural network with distributional shift correction," *PLOS ONE*, vol. 14, no. 12, p. e0224642, 2019.

[43] G. Z. d. S. Lima, S. R. Lopes, T. L. Prado, B. Lobao-Soares, G. C. d. Nascimento, J. Fontenele-Araujo, and G. Corso, "Predictability of arousal in mouse slow wave sleep by accelerometer data," *PLOS ONE*, 2017.

[44] S. Sunderam, N. Chernyy, N. Peixoto, J. P. Mason, S. L. Weinstein, S. J. Schiff, and B. J. Gluckman, "Improved sleep-wake and behavior discrimination using MEMS accelerometers," *J Neurosci Methods*, vol. 163, no. 2, pp. 373–383, Jul 2007.

[45] B. M. Bergmann, J. B. Winter, R. S. Rosenberg, and A. Rechtschaffen, "NREM sleep with low-voltage EEG in the rat," *Sleep*, vol. 10, no. 1, pp. 1–11, Feb 1987.

[46] E. L. van Luijtelaar and A. M. Coenen, "An EEG averaging technique for automated sleep-wake stage identification in the rat," *Physiol Behav*, vol. 33, no. 5, pp. 837–841, Nov 1984.

[47] M. M. Lacroix, G. d. Lavilléon, J. Lefort, K. E. Kanbi, S. Bagur, S. Laventure, Y. Dauvilliers, C. Peyron, and K. Benchenane, "Improved sleep scoring in mice reveals human-like stages." *bioRxiv*, p. 489005, 2018.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[49] T. M. Mitchell *et al.*, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

[50] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

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

[52] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[53] https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9.

[54] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2002.

[55] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[56] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*. PMLR, 2013, pp. 1139–1147.

[57] https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826.

[58] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[59] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[60] https://en.wikipedia.org/wiki/Decision_tree_learning.

[61] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists," *Physics Reports*, vol. 810, p. 1–124, May 2019. [Online]. Available: http://dx.doi.org/10.1016/j.physrep.2019.03.001

[62] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[63] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[64] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[65] https://en.wikipedia.org/wiki/Artificial_neural_network.

[66] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[68] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[69] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[70] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[71] L. Cohen, *Time-frequency analysis*. Prentice Hall PTR Englewood Cliffs, NJ, 1995, vol. 778.

[72] A. L. Vyssotski, A. N. Serkov, P. M. Itskov, G. Dell'Omo, A. V. Latanov, D. P. Wolfer, and H. P. Lipp, "Miniature neurologgers for flying pigeons: multichannel EEG and action and field potentials in combination with GPS recording," *J Neurophysiol*, vol. 95, no. 2, pp. 1263–1273, Feb 2006.

[73] A. J. Spink, R. A. Tegelenbosch, M. O. Buma, and L. P. Noldus, "The EthoVision video tracking system–a tool for behavioral phenotyping of transgenic mice," *Physiol Behav*, vol. 73, no. 5, pp. 731–744, Aug 2001.

[74] O. Friard and M. Gamba, "Boris: a free, versatile open-source event-logging software for video/audio coding and live observations," *Methods in ecology and evolution*, vol. 7, no. 11, pp. 1325–1330, 2016.

[75] F. Chollet and H. Liu, *A (not so) short introduction to MEMS*. memscyclopedia.org, 08 2018.

[76] M. Dadafshar, "Accelerometer and gyroscopes sensors: operation, sensing, and applications," *Maxim Integrated [online]*, 2014.

[77] S. E. Lyshevski, *MEMS and NEMS: systems, devices, and structures*. CRC press, 2018.

[78] S. Beeby, G. Ensel, N. M. White, and M. Kraft, *MEMS mechanical sensors*. Artech House, 2004.

[79] https://www.analog.com/en/app-notes/an-1057.html.

[80] W. Rast, S. E. Kimmig, L. Giese, and A. Berger, "Machine learning goes wild: Using data from captive individuals to infer wildlife behaviours," *PLoS One*, vol. 15, no. 5, p. e0227317, 2020.

[81] L. Lush, S. Ellwood, A. Markham, A. Ward, and P. Wheeler, "Use of tri-axial accelerometers to assess terrestrial mammal behaviour in the wild," *Journal of Zoology*, vol. 298, no. 4, pp. 257–265, 2016.

[82] A.-H. Hokkanen, L. Hänninen, J. Tiusanen, and M. Pastell, "Predicting sleep and lying time of calves with a support vector machine classifier using accelerometer data," *Applied Animal Behaviour Science*, vol. 134, no. 1-2, pp. 10–15, 2011.

[83] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[84] F. Wang, S. Zhong, J. Peng, J. Jiang, and Y. Liu, "Data augmentation for eeg-based emotion recognition with deep convolutional neural networks," in *MultiMedia Modeling - 24th International Conference, MMM 2018, Proceedings*. Springer-Verlag, Jan. 2018, pp. 82–93.

[85] N. Cellini, M. de Zambotti, N. Covassin, M. Sarlo, and L. Stegagno, "Impaired off-line motor skills consolidation in young primary insomniacs," *Neurobiology of learning and memory*, vol. 114, pp. 141–147, 2014.

[86] D. K. Welsh, G. S. Richardson, and W. C. Dement, "A circadian rhythm of hippocampal theta activity in the mouse," *Physiology & behavior*, vol. 35, no. 4, pp. 533–538, 1985.

[87] F. Yaghouby and S. Sunderam, "Segway: A simple framework for unsupervised sleep segmentation in experimental eeg recordings," *MethodsX*, vol. 3, pp. 144–155, 2016.

[88] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[89] H. Helakari, J. Kananen, N. Huotari, L. Raitamaa, T. Tuovinen, V. Borchardt, A. Rasila, V. Raatikainen, T. Starck, T. Hautaniemi, T. Myllylä, O. Tervonen, S. Rytky, T. Keinänen, V. Korhonen, V. Kiviniemi, and H. Ansakorpi, "Spectral entropy indicates electrophysiological and hemodynamic changes in drug-resistant epilepsy – a multimodal mreg study," *NeuroImage: Clinical*, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213158219301135

[90] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[91] R. Nathan, O. Spiegel, S. Fortmann-Roe, R. Harel, M. Wikelski, and W. M. Getz, "Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffon vultures," *Journal of Experimental Biology*, 2012.

[92] S. Venkatraman, X. Jin, R. M. Costa, and J. M. Carmena, "Investigating neural correlates of behavior in freely behaving rodents using inertial sensors," *J Neurophysiol*, vol. 104, no. 1, pp. 569–575, Jul 2010.

[93] D. Miladinovic, C. Muheim, S. Bauer, A. Spinnler, D. Noain, M. Bandarabadi, B. Gallusser, G. Krummenacher, C. Baumann, A. Adamantidis, S. A. Brown, and J. M. Buhmann, "SPINDLE: End-to-end learning from EEG/EMG to extrapolate animal sleep scoring across experimental settings, labs and species," *PLoS Comput Biol*, vol. 15, no. 4, 04 2019.

[94] S. Bagur, M. M. Lacroix, G. de Lavilléon, J. M. Lefort, H. Geoffroy, and K. Benchenane, "Harnessing olfactory bulb oscillations to perform fully brain-based sleep-scoring and real-time monitoring of anaesthesia depth," *PLoS Biol*, vol. 16, no. 11, 11 2018.