Unraveling the Brain: a Quantitative Study of EEG Classification Techniques

Lênon Guimarães Silva Alípio

Dissertação apresentada ao
Instituto de Matemática e Estatística
da Universidade de São Paulo
para obtenção do título de
Mestre em Ciências

Programa: Mestrado em Estatística

Orientador: Prof. Dr. Florencia G. Leonardi

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES

São Paulo, Novembro de 2020

Unraveling the Brain: a Quantitative Study of EEG Classification Techniques

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 09/04/2021. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof[a]. Dr[a]. Florencia Graciela Leonardi - IME-USP
- Prof. Dr. Daniel Yasumasa Takahashi - UFRN
- Prof. Dr. Daniel Fraiman Borrazas - UDESA

Resumo

Guimarães, L. S. A. **Desvendando o Cérebro: um Estudo Quantitativo de Técnicas para Classificação de EEG**. 2020. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2020.

O problema de classificação de EEG, onde procura-se identificar estados mentais através da análise de sinais eletroencefalográficos, vem ganhando crescente atenção da comunidade científica com os recentes avanços nas tecnologias de EEG e nas técnicas de Big Data e Machine Learning. No entanto, muitas das pesquisas atuais sendo realizadas sobre esse assunto apresentam falhas metodológicas significativas, como a não otimização de hiperparâmetros de modelos, vazamento de informação entre bancos de dados de treino e teste, escolhas equivocadas de referências de comparação, entre outros, o que torna duvidosos muitos dos resultados obtidos. Por esse motivo, não é claro quais são os melhores métodos para o problema da classificação de EEG atualmente, nem como eles se comparam entre si. Nesta dissertação, abordamos esse problema fazendo, primeiramente, um levantamento de métodos propostos na literatura científica que alegam resultados equivalentes ao estado da arte, e que tenham aderido a diretrizes estatísticas e da ciência de dados que possam sustentar tal afirmação. Em seguida, realizamos uma comparação quantitativa desses métodos em 4 bancos de dados de EEG diferentes. Dos 11 métodos estudados, mostramos que aqueles baseados no uso de Transformadas de Fourier, Transformadas Wavelet, e Parâmetros de Hjorth são os que apresentam melhor desempenho geral, e podem ser usados como uma forte referência contra a qual se comparar quaisquer novos métodos e análises propostos futuramente no campo de classificação de EEG.

Palavras-chave: Classificação de EEG, Aprendizado de Máquina, Redes Neurais, Wavelets, Transformada de Fourier.

Abstract

Guimarães, L. S. A. **Unraveling the Brain: a Quantitative Study of EEG Classification Techniques**. 2020. 120 f. Dissertation (Masters) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2010.

The problem of EEG Classification, where one tries to identify neural conditions through electroencephalographic signal analysis, has been gathering increasing attention from the scientific community with the recent advances in EEG technology and Big Data/Machine Learning techniques. However, much of the current research on this topic presents significant methodological flaws, such as non-optimization of models' hyperparameters, data leakage between train and test datasets, and poor choice of comparison baselines, among others, which render many of the obtained results dubious. Thus, it is not clear what are the state-of-the-art methods for the EEG Classification problem today, nor how they compare to one another. In this dissertation, we tackle this problem by, first, surveying methods proposed in the scientific literature which claim to achieve state-of-the-art performance while still adhering to data science and statistical guidelines that can sustain such a claim. Then, we make a quantitative comparison of these methods on four different EEG datasets. Of the 11 methods studied, we show that those based on Fourier Transforms, Wavelet Transforms, and Hjorth Parameters are the ones with the best overall performance, and can that they can be used as a strong baseline against which any new methods and analyses hereafter proposed in the EEG Classification field should be compared.

Keywords: EEG Classification, Machine Learning, Neural Networks, Wavelets, Fourier Transform.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introduction

Using signals from the brain to make predictions about behavior, perception, and cognitive state is becoming increasingly important not only within Neuroscience, but across a wide array of different fields, such as Psychology (Qazi *et al.*, 2019), Marketing (Yadava *et al.*, 2017), Gaming (Jiang *et al.*, 2019), and Robotics (Jia *et al.*, 2012), among many others.

Of all available methods to extract and analyze such signals, Electroencephalography (EEG) is one of the most widely used due to its high temporal resolution, non-invasiveness, and relatively low financial cost, especially when compared to other techniques such as fMRI and MEG (Craik *et al.*, 2019).

In the context of neural decoding - that is, of trying to identify events, stimulus, and intents through analysis of neuronal activity patterns - the use of EEG is already very well established, and recent advances in Machine Learning and Big Data technologies, as well as increasing availability of EEG datasets, have resulted in even bigger interest in this subject from the scientific community, with a plethora of new papers and studies being released every month. Works by Craik *et al.* (2019), Lotte *et al.* (2018), Roy *et al.* (2019), Zhang *et al.* (2019), and Gu *et al.* (2020), to name a few, showcase how vast are the recent contributions made to this field by reviewing some of the most important works published in the past few years.

A closer look at these works, however, reveal that many of them present significant methodological flaws, especially when it comes to studies regarding the problem of EEG Classification, where one uses Statistical and Machine Learning techniques on EEG signals to classify certain aspects of the person it belongs too, such as the presence or absence of a mental illness, what emotions they are feeling, the direction they are moving a limb, etc. These works, which often claim that their novel approaches achieve state-of-the-art performance on EEG Classification tasks, do not adhere to some important methodological guidelines in Data Science, which render many of their results dubious, if not downright wrong.

Some of the guidelines many of these studies fail to consider, as well as papers where this happens, are:

1. **Guarantee no data leakage between test and training datasets.**

   To accurately evaluate the performance of any Machine Learning model, the data used to train the model (the *training dataset*) must not contain any information about the data used to test it (the *test dataset*). When information is shared between the datasets - the so-called *data leakage* - the evaluation process is heavily compromised, since the model, having already seen the testing data during its training, will achieve much better performance at the test dataset than it would at truly unseen data in a real-world scenario.

   For example, recent works by Spampinato *et al.* (2017) and Kavasidis *et al.* (2017) claimed to have developed groundbreaking deep learning models that could predict and reconstruct what image a person was seeing by analyzing their EEG signals with an accuracy of over 82%, much higher than that achieved by any other method at a task like this. However, later analysis made by Li *et al.* (2018) has shown that their results could only be obtained due to

a subtle data leak caused by a flaw in the design of their experiments, and that when their models were applied to a test dataset with no leakage, it performed no better than a random guess.

2. **Hyperparameter optimization must be done when comparing models of different complexities.**

   The performance of a classification model is highly related to how well-tuned its hyperparameters are. The more complex a models is - that is, the more hyperparameters it has - the more care must be given to the tuning process, since finding the optimal set of hyperparameters for a model that has many of them - like a Random Forest, which can easily have more than ten, depending on the implementation - is much harder than finding it for one with only a few - like a Logistic Regression, which usually has only one, the regularization strength. To properly compare the performance of models of different complexities, it is fundamental that their hyperparameters are adequately optimized. Otherwise, the simpler models might achieve better results solely based on the mistuning of the more complex ones.

   For example, in Dadi *et al.* (2019) benchmark of numerous different methods to classify fMRI signals, Logistic Regression was determined as the best one when it came to classification accuracy. However, that study did not perform any hyperparameter optimization of the models it was evaluating, so the Logistic Regression's comparatively better performance could be because it had much fewer hyperparameters than the others against it was compared. As a matter of fact, when adapting the available benchmark code[1] to include the hyperparameter optimization of the Random Forest, one of many models the Logistic Regression was compared against, the accuracy obtained with the Random Forest was similar, and sometimes even greater, than that of the Logistic Regression, which puts into question some of the results of the benchmark.

3. **Evaluate results over multiple train/test splits.**

   A model's performance on a test dataset is expected to be an unbiased estimate of its actual performance in a real-world scenario. However, when this dataset is small - usually the case with EEG studies - the obtained performance can be biased, since this set might be comprised mostly of the easier to predict cases. To avoid this potential bias, the model should be evaluated on different test datasets, that can be obtained by choosing different split points in the available data when dividing it into train and test sets. This way, we can have a confidence interval of the model's performance instead of a punctual estimation.

   Analyses published by Omerhodzic *et al.* (2013), Subasi e Ercelebi (2005), and Jolly *et al.* (2019), to name a few, are some examples of works that provide only punctual estimations of performance, not making any cross-validation of their results.

4. **Code developed for the study should be released online.**

   Though more of a general recommendation than a strict guideline, publishing online the code developed for a paper is highly desirable. It allows for easy double-checking of the presented results by third parties, and when a new method is being proposed, it is much simpler for others to implement it into their own works. Depending on the complexity of the proposed methods, releasing a baseline code for them is pretty much mandatory, since implementing new, highly complex models from scratch is a challenge that might turn away many researchers - especially those without a Data Science or Statistics background - from experimenting with it.

   For instance, a recent study by Qazi *et al.* (2019) proposed a new Deep Learning architecture to detect emotions from EEG signals, which he claimed to beat other state-of-the-art systems by a large margin. However, no code for this new method was released, which makes validating

[1] https://github.com/neurospin/pypreprocess

his claims, as well as applying the method on other scenarios, very difficult. As another example, Tirupattur *et al.* (2018) recently presented a method to recreate images one was thinking by analyzing their EEG signals with a GAN-based Deep Learning model. After some independent analysis of their published results, Li *et al.* (2018) put into question the true efficacy of the new method by pointing out some inconsistencies in their study, but since no code whatsoever was released - that is, neither the code for the method itself, nor the code that generated the results published - its impossible to precisely point out where the mistakes (if any) are, and work towards a fix.

5. **Results should be compared against strong baselines from literature.**

   Whenever a new method is being proposed, it is important that the results it yields be compared against results from previous state-of-the-art techniques so that a fair analysis of the new method's strengths and weaknesses can be made. This is particularly true in the field of EEG Classification, where many different approaches exist to the neural decoding task, and the real value of any newly proposed method can only be properly assessed when placed alongside the best-known models from each of these approaches.

   Alturki *et al.* (2019), Rodriguez *et al.* (2018), and Vézard *et al.* (2015), to name a few, have all proposed new methods for EEG Classification while not providing any comparison with other state-of-the-art neural decoding models, or even with basic baselines from literature.

In summary, while there is great interest from the scientific community on the EEG Classification problem, with many papers and studies on this field being published every month, the methodological flaws commonly found in these works make it hard to have a clear picture of what are the most promising approaches to the neural decoding task, and what are the baseline models to beat.

In this dissertation, we tackle this problem by proposing a quantitative study of current state-of-the-art methods for EEG Classification. We will analyze some of the best performing models from the literature by applying them to a number of different EEG Datasets, and, judging the results through the necessary statistical guidelines, present what is today the overall best one, which can be used as a strong baseline against which any new methods and analysis hereafter proposed in the EEG Classification field should be compared.

1.1 Previous Surveys

There have been some recent works that attempted to summarize the current state of EEG Classification studies. Most notably:

- Roy *et al.* (2019) made a systematic review of neural decoding Deep Learning models, surveying a total of 156 papers. This review was of a qualitative nature - that is, it didn't apply the proposed models on any new datasets or validated their results, focusing more on sharing their overall performance and discussing trends and implementation details. It also came across many of the methodological flaws mentioned before: only 7% of the surveyed papers released the code and data used for their studies, making the vast majority of the models and analysis proposed hard or impossible to reproduce, and less than 20% of the papers made any statistical comparison between their results and other baselines from literature.

- Lotte *et al.* (2018) also made a qualitative review of EEG Classification methods, focusing not only on Deep Learning approaches, but on a more vast array of models, which included LDA, SVM, and Random Forest, among others. This review only surveyed papers regarding Brain-Computer Interface (BCI) tasks, a subset of EEG Classification problems. Curiously, it pointed out that, in general, Deep Learning methods were not as effective as other more traditional models, while Roy *et al.* (2019) claimed the opposite, a contradiction that arose mainly due to the lack of a quantitative aspect to these studies.

- Sun e Zhou (2014) made a theoretical review of some of the most important methods for EEG Classification in BCI tasks. This review, however, focus purely on the theory behind the techniques employed, not presenting any evaluation of the methods over real datasets or performance comparison between them.

- Siuly *et al.* (2016), when proposing new methods of EEG Classification for BCI and Epilepsy Detection tasks, made a quantitative evaluation of their proposed models against other state-of-the-art methods from literature, following many of the necessary statistical guidelines mentioned before. No code was released for this evaluation, the datasets used for the study belonged only to two categories (BCI and Epilepsy Detection), and no Deep Learning method was considered in the comparison, but, other than this, it's one of the most complete reviews in the literature to date.

The study we propose in this dissertation serves as a much-needed complement to the aforementioned reviews, offering a quantitative analysis of verified state-of-the-art methods over different EEG datasets, which will help define what are the current best models in the EEG Classification field, and what approaches should researches in the area be paying attention to.

1.2 Study Outline

In total, we analyze 11 different methods of EEG Classification over four different EEG datasets.

Since it would be unfeasible to evaluate quantitatively all methods recently proposed in the literature, we focus our survey on models that met the following criteria:

- State-of-the-art performance on EEG Classification tasks, proven with quantitative, reproducible results and backed by peer-reviewed publications.

- Generalization capability - that is, the model should not be task-specific, but able to be used in any EEG Classification problem with minimum adaptation.

- Finite, reachable training times for any typical EEG dataset - this excludes from our survey models that might achieve state-of-the-art performance, but can only be trained on small, non-EEG datasets due to the models' high time complexity, such as HIVE-COTE (Lines *et al.*, 2016).

With this in mind, we narrow our analysis to 11 models, which we believe create an accurate picture of the overall trends and best methods in the field of EEG Classification today. These 11 methods can be grouped into three categories, as follows:

- **Univariate Featurization Methods**

  These methods rely on creating independent sets of features from each of the EEG Channels used in the experiment and concatenating them into a single vector, which is then fed to a traditional Machine Learning classifier.

  Of the myriad of possible features that can be created from each of these channels - channels which, in practice, can be seen as a set of univariate time series - literature shows that those made from Fourier Transforms (Shaker, 2006), Wavelet Transforms (Kumar *et al.*, 2017), and Hjorth Parameters (Kesić e Spasić, 2016) are among the most effective in discriminating different classes in EEG Classification problems, and will be the ones we will investigate in this study.

  Regarding the Machine Learning Classifier, we will analyze four of the most effective and commonly used models, not only in Neuroscience, but in Data Science as a whole: Logistic Regression with L2 Regularization, Support Vector Machine, Random Forest, and Gradient Boosted Trees (Hastie *et al.*, 2009).

So, naming **FS** the feature set composed of all features created by Fourier Transforms, Wavelet Transforms, and Hjorth Parameters of each EEG channel, we have a total of four Univariate Featurization Methods to analyze:

1. **FS + Logistic Regression with L2 Regularization**
2. **FS + Support Vector Machine**
3. **FS + Random Forest**
4. **FS + Gradient Boosted Trees**

- **Multivariate Transformation Methods**

  These methods frame the EEG data not as a set of univariate time series, but as a single multivariate entity, which they transform into a discriminative shape that can be fed to a traditional Machine Learning Classifier.

  We shall analyze a total of five methods in this category. The first of them is the Dynamic Time Warping (**DTW**) plus 1-Nearest-Neighbor (**1NN**) approach. Here, we transform the multivariate time series into a similarity matrix with DTW, and use 1NN to classify the observations according to what class they are most similar to. This a standard, tried-and-true method for time series classification (Senin, 2008), and has shown some promising results for EEG related tasks (Martin, 2015; Yamauchi *et al.*, 2015).

  The other four methods are related to the WEASEL+MUSE (**WM**) transform, a state-of-the-art method that uses symbolic representations to extract meaningful features from multivariate time series (Schäfer e Leser, 2017), features which we will use as inputs to the same four classifiers analyzed in the Univariate Featurization section.

  So, in short, the five Multivariate Transformation methods we will study are:

5. **DTW + 1NN**
6. **WM + Logistic Regression with L2 Regularization**
7. **WM + Support Vector Machine**
8. **WM + Random Forest**
9. **WM + Gradient Boosted Trees**

- **Deep Learning Methods**

  Artificial Neural Networks with deep layers have recently gained significant prominence in scientific research by achieving some unprecedented results in a vast array of fields, like Image Analysis, Natural Language Processing, and Time Series Forecasting, among many others. In Neuroscience, the study of Deep Learning methods is ever-increasing and seen with great interest by the scientific community, as mentioned before.

  In this survey, we will analyze two different Deep Learning architectures. The first is the Convolutional Neural Network (**CNN**) architecture, which is the most prevalent one in EEG Classification papers today (Craik *et al.*, 2019). The second one is the InceptionTime architecture, which is based around the Inception neural modules recently proposed by Google, and that outperforms many of the state-of-the-art models in a great number of different time series classification tasks (Fawaz *et al.*, 2019):

10. **CNN**
11. **InceptionTime**

These 11 methods are evaluated over the following four different EEG datasets:

1. **DEAP**: A multimodal dataset for the analysis of human affective states. In this dataset, EEG signals of 32 participants were recorded as each watched 40 one-minute long excerpts of music videos. The task is to predict whether they liked the video or not (Koelstra *et al.*, 2011).

2. **EEG Alcoholism**: This dataset arises from a large study to examine EEG correlates of genetic predisposition to alcoholism. It contains measurements from 64 electrodes over 122 subjects - controls and alcoholics - where each subject completed 120 trials being exposed to different pictures from objects of the Snodgrass and Vanderwart picture set (Snodgrass e Vanderwart, 1980). The task is to predict whether an individual rated the video as having high or low valence (that is, if the video evoked positive or negative sentiments).

3. **FingerMovements**: This dataset was recorded from a normal subject during a no-feedback session. The subject sat in a normal chair, relaxed arms resting on the table, fingers in the standard typing position at the computer keyboard. The task was to press with the index and little fingers the corresponding keys in a self-chosen order and time, i.e. using self-paced key typing. The objective here is to predict whether he will press a key with his left or his right hand (Blankertz *et al.*, 2002).

4. **SelfRegulationSCP1**: Here, the data was taken from a healthy subject that was asked to move a cursor up and down on a computer screen, while his EEG signals were taken. During the recording, the subject received visual feedback of his slow cortical potentials (Cz-Mastoids). Cortical positivity leads to a downward movement of the cursor on the screen. Cortical negativity leads to an upward movement of the cursor. The task is to predict whether the individual wants to move the cursor up or down (Birbaumer *et al.*, 1999).

1.3 Contributions

The three main contributions of this study are:

- Making a quantitative analysis of some of the most prominent methods for EEG Classification today, a type of analysis which is severely lacking in the literature, and that can help identifying the best current models, elucidating research trends, and guiding future studies in the field.

- Defining a strong baseline method against which new EEG Classification models should be compared, a baseline that can both assist researchers during the performance evaluation of their new proposals, and help standardizing comparison methodologies in the field.

- Providing a comprehensive, technical review of the methods analyzed and of the statistical and data science guidelines followed, which can be used as a reference by researchers in the field.

1.4 Study Organization

Chapter 2 gives a theoretical overview of the methods analyzed in this study and the statistical and data science guidelines followed to gauge their performance. Chapter 3 gives a more detailed explanation of the datasets and the methods analyzed, the procedure employed to evaluate them, and the results obtained from our analysis. The final conclusions and suggestions for future work are given in Chapter 4.

Capítulo 2

Literature Review

This chapter gives an overview of all the theories and concepts that support our quantitative analysis of EEG Classification methods. More specifically:

- In Section 2.1 we explain in more detail what are the Electroencephalogram signals, how they are measured, the five major brain wave types, and give a more technical definition of the EEG Classification problem.
- In Section 2.2 we present the Statistical and Data Science guidelines necessary to ensure the validity of our analysis: the K-fold Cross Validation, Grid Search for hyperparameter optimization, and the use of Critical Difference Diagrams for performance metrics evaluation.
- In Section 2.3 we review some Feature Engineering methods used to extract features from the EEG signals: Fourier Transforms, Wavelet Transforms, Hjorth Parameters, WEASEL+MUSE, and Dynamic Time Warping.
- Finally, in Section 2.4 we give the formal definition behind the most used models to classify the EEG signals: k-Nearest Neighbours, Logistic Regression, Support Vector Machines, Random Forest, Gradient Boosted Trees, and Neural Networks.

2.1 Electroencephalography (EEG)

Electroencephalography (EEG) is a non-invasive brain-imaging method that records the brain's electrical activity at the surface of the scalp. The strength of EEG lies in its high temporal resolution, its relatively low cost - especially when compared to other brain-imaging methods, such as fMRI and MEG - and in the information-dense signals it produces, which can be used in a wide variety of tasks related to the state of the brain, such as Emotion Recognition, BCI, and Diagnostics, among many others (Sanei e Chambers, 2013).

EEG equipment usually comes in the form of a cap or headset that has several electrodes or sensors which are designed to fix to the surface of the head, as illustrated in Figure 2.1. The number of electrodes used in typical recordings varies with the kind of experiment being made and the budget, ranging from less than 10 to possibly more than 100 (Abhang *et al.*, 2016).

However, regardless of the number of electrodes used, their placement over the head usually follow a standard, international protocol named 10–20 System, illustrated in Figure 2.2. As explained by Hu e Zhang (2019):

> This system contains standardized locations of 75 electrodes on the scalp. The benchmarks are first defined in the system (i.e., the nasion, inion, and two preauricular points). Locations of other electrodes can be determined according to those datum points. The electrodes are placed at 10% and 20% points along lines of latitude and longitude, and this is the reason why the system is called the 10–20 system. The name of each electrode contains two parts,

that is, one or two letters and a number. The letter represents the general brain region of the electrode (Fp, frontal pole; F, frontal; C, central; P, parietal; O, occipital; and T, temporal). The number stands for the distance from the midline; the larger the number is, the greater distance it is from the midline (electrodes on the midline are labeled with a "z" for zero). Odd numbers are used at the left hemisphere, and even numbers are used in the right hemisphere (left and right side is from the point of view of a subject)

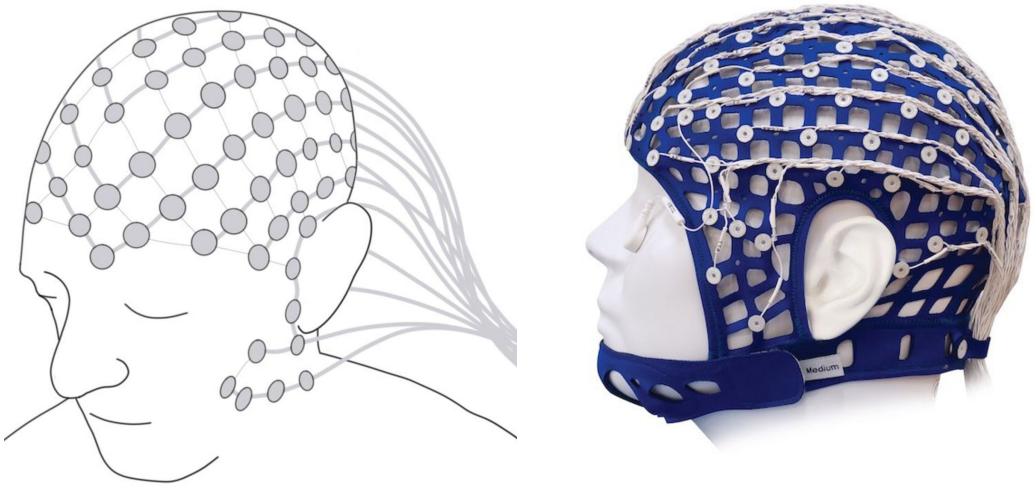

Figura 2.1: *Typical EEG cap*

Normally, the EEG signals are categorized into five main Brain Waves categories according to their frequency, namely the Delta (δ, < 4 Hz), Theta (θ, 4–8 Hz), Alpha (α, 8–12 Hz), Beta (β, 12–35 Hz), and Gamma (γ, > 35 Hz) waves, as shown in Figure 2.3. These waves intensities vary depending upon what activities a person is performing, and can provide important information regarding one's health and state of mind. More specifically, as described by Abhang *et al.* (2016):

- **Gamma Waves ($>$ 35 Hz)**

  A gamma wave is considered to be the fastest brain activity. It is responsible for cognitive functioning, learning, memory, and information processing. Prominence of this wave leads to anxiety, high arousal, and stress; while its suppression can lead to Attention Deficit Hyperactivity Disorder (ADHD), depression, and learning disabilities. In optimal conditions gamma waves help with attention, focus, binding of senses (smell, sight, and hearing), consciousness, mental processing, and perception.

- **Beta Waves (12-35 Hz)**

  Beta waves are high-frequency, low-amplitude brain waves that are commonly observed in an awaken state. They are involved in conscious thought and logical thinking, and tend to have a stimulating effect. Having the right amount of beta waves allows us to focus. Prominence of this wave causes anxiety, high arousal, an inability to relax, and stress, whereas its suppression can lead to ADHD, daydreaming, depression, and poor cognition. In optimal conditions beta waves help with conscious focus, memory, and problem solving.

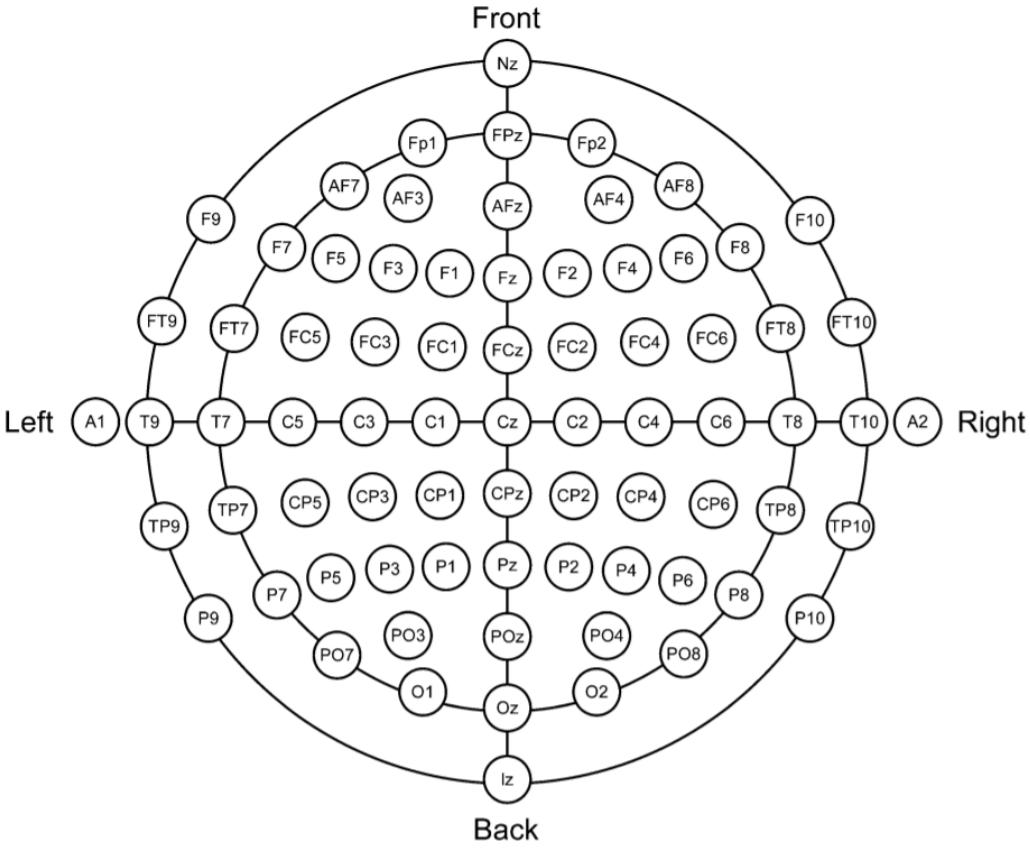

Figura 2.2: *International 10–20 System for electrode placement (Hu e Zhang, 2019)*

- **Alpha Waves (8-12 Hz)**

  Alpha waves have a frequency range between beta and theta. They help us calm down when necessary and promote feelings of deep relaxation. Alpha waves are found prominently in daydreaming, inability to focus, and being very relaxed. If they are suppressed it can cause anxiety, high stress, and insomnia. When they are optimal it leads to a relaxed state.

- **Theta Waves (4-8 Hz)**

  This particular frequency range is involved in daydreaming and sleep. ADHD, depression, hyperactivity, impulsivity, and inattentiveness are observed when theta waves are prominent; if they are suppressed, anxiety, poor emotional awareness, and stress can be seen. In an optimal state, theta helps in creativity, emotional connection, intuition, and relaxation. Theta waves have benefits of helping to improve our intuition and creativity, and making us feel more natural. Theta is also involved in restorative sleep.

- **Delta Waves ($<$ 4 Hz)**

  Delta waves are the slowest recorded brain waves in human beings. They are found most often in infants and young children, and are associated with the deepest levels of relaxation and restorative, healing sleep. Delta is prominently seen in brain injuries, learning problems, inability to think, and severe ADHD. If this wave is suppressed, it leads to an inability to rejuvenate the body and revitalize the brain, and poor

sleep. Adequate production of delta waves helps us feel completely rejuvenated and promotes the immune system, natural healing, and restorative/deep sleep.

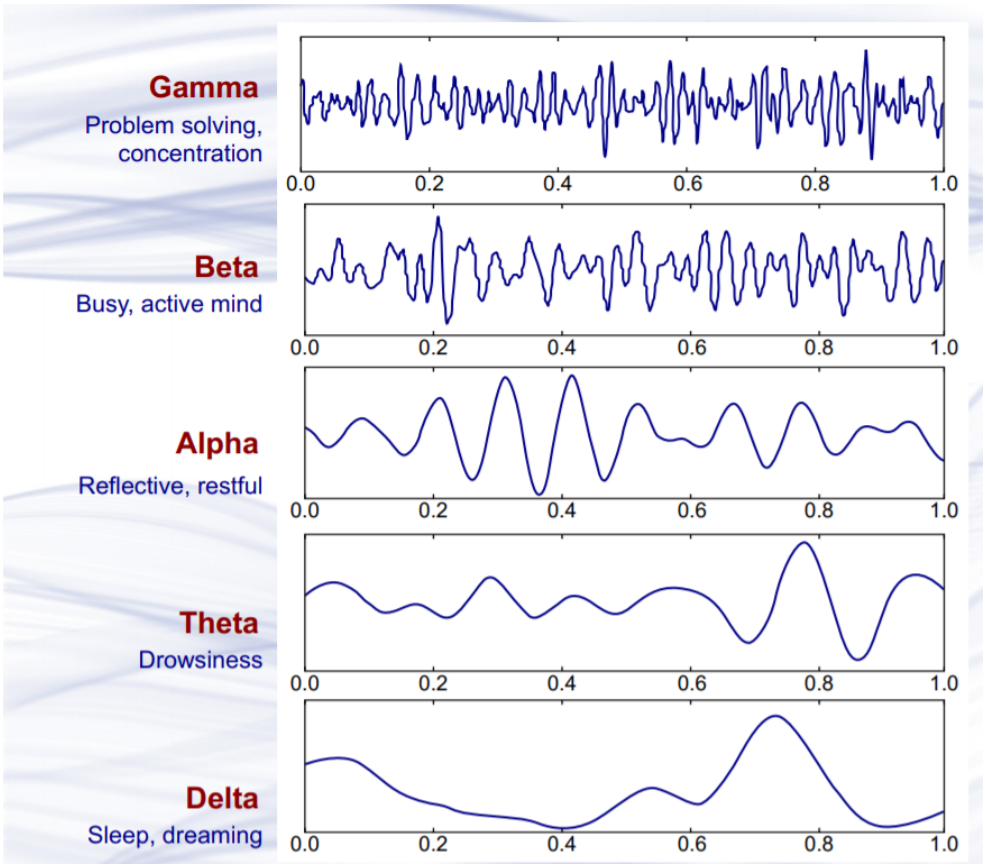

Figura 2.3: *Samples of the five main Brain Waves categories (Abhang et al., 2016)*

2.1.1 EEG Classification

The problem of EEG Classification can be formally defined as follows:

- Let $\mathbf{MT^j}$ be the multivariate timeseries formed by all $\mathbf{X_i}$ signals obtained from a given EEG trial $\mathbf{j}$, where $\mathbf{i}$ is the number of channels used in the experiment. Let $\mathbf{K}$ be the total number of classes the trial can be classified into, and $\mathbf{MT^j_{feat}}$ be a vector of features created from $\mathbf{MT^j}$. The EEG Classification problem consists in finding a model $\mathbf{f}$ that takes these features as input and outputs the class of the trial, that is, $\mathbf{f}\left(\mathbf{MT^j_{feat}}\right) = \mathbf{k^j} \in \{\mathbf{1}, \ldots, \mathbf{K}\}$.

Our analysis will consider only the binary classification case, where $\mathbf{K} = \mathbf{2}$. Also, depending on the model being used, the featurization step is not necessary, since it can take the raw multivariate time-series as an input, which is the case with Neural Networks, for example.

Figure 2.4 illustrates a typical EEG Classification pipeline. The Featurization and Modelling steps will be discussed in detail in Sections 2.3 and 2.4, respectively. Regarding the Preprocessing step, we summarize here the most commonly used techniques - most of them applied in the datasets used in our study (see Section 3.1) - and refer the reader to Sanei e Chambers (2013) for a more in-depth review.

1. **Filtering**

   It is common that the first step in the preprocessing pipeline is filtering, which consists of removing some of the noise of the EEG recording by passing it through some frequency filters

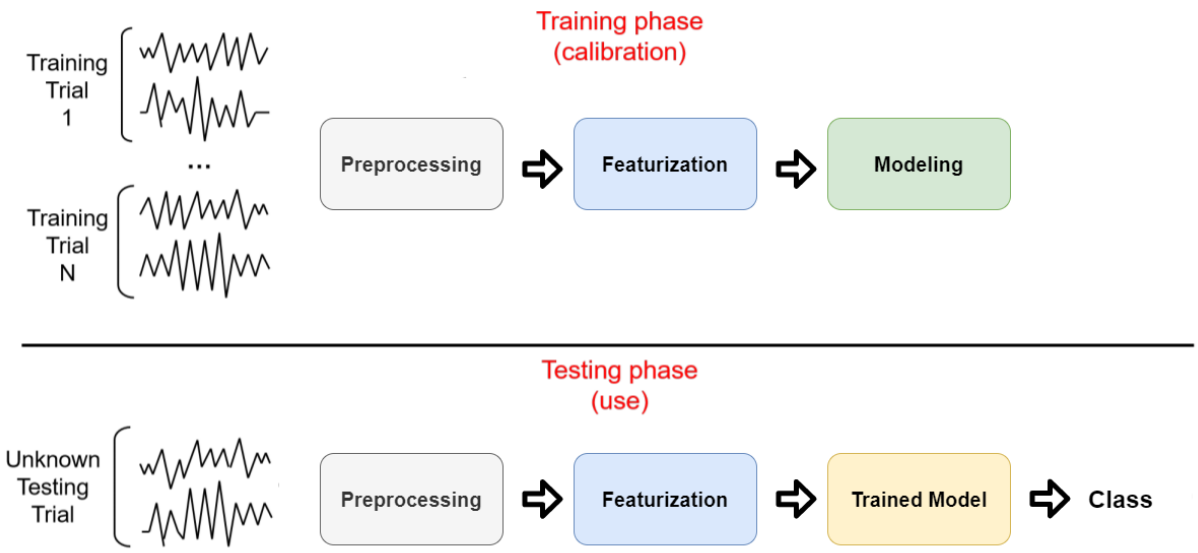

Figura 2.4: *Typical EEG Classification process (adapted from Lotte et al. (2018))*

- usually the lowpass filter, highpass filter, bandpass filter, and/or bandstop filter, illustrated in Figure 2.5.

The filters are chosen according to which frequencies the researcher wants to keep or remove. Some of the most usual filterings are as follows:

- Since neuronal information is fully contained in the EEG signals below 100 Hz, a lowpass filter can be used to remove frequencies above this threshold.
- Highpass filters, usually with a limit of 0.1 Hz, can be applied to the EEG signals to remove noise due to low-frequency drifts.
- Noises due to power lines influence (usually in the 50-60 Hz range, depending on the country) can be removed with a bandstop filter.
- If the researcher is interested in analyzing only a specific type of brain wave, bandpass filters can be used to keep only the signals in the desired frequency range.

2. **Bad Channels**

   During the Electroencephalography trials, some channels might produce inaccurate signals due to equipment malfunction, electrode misplacement, saturation, etc.

   If a channel presents problems in most trials, the most usual correction is to exclude its signals from the analysis. It is important that the channel then be removed from all trials, even those where it did not fail, so that the data matrices obtained from these trials have the same dimension in all cases.

   If, however, the channel only present problems on a few trials, one could choose to 'repair' its signals instead of removing them, a solution that is especially interesting when the number of electrodes being used is small, and the complete removal of a channel would incur in an unacceptable loss of information. The most common repair technique is interpolation by spherical splines, which consists of replacing the flawed signal with a weighted sum of the other non-flawed channels' signals. This technique is explained in detail by Perrin *et al.* (1989).

3. **Reference electrode**

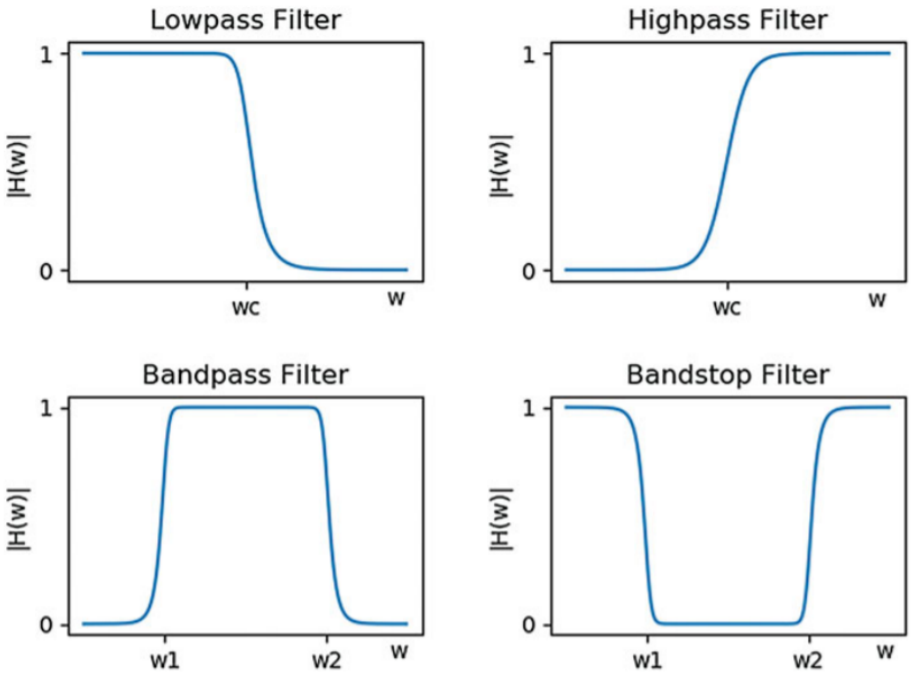

Figura 2.5: *Four types of filters commonly used in EEG preprocessing (Abhang et al., 2016)*

In practice, what EEG electrodes measure are differences in electrical potentials between two points - that is, the EEG signal of any channel is the difference in electrical potential between the region that channel is attached to and some other point. In usual applications, this other point can be a ground electrode, but in EEG this induces noise into the analysis, since the non-cerebral electrical oscillations of the ground, not present at the scalp electrodes, will be transferred to the EEG signals if the ground is used as a reference.

To eliminate this ground-related noise, one of the EEG channels itself, usually called the Online Reference, can be used as the reference for the other channels. This has the effect of canceling noise stemming from the ground ([channel – ground] – [online reference – ground] = channel – online reference).

During the recording of the trials, the choice of this Online Reference is not that important, since the data can be re-referenced at any time to a new reference - without loss of information - by simply subtracting the signal of the new reference from each EEG channel. The choice of reference will be important, however, during the analysis of the data, since the amplitude of the EEG signals will tend to be higher the further away their channels are from the reference. Therefore, some good practices when deciding upon a reference are:

- The reference should ideally be symmetric in relation to the other channels, otherwise, the voltage distribution on the head will likely be biased towards one hemisphere.
- The reference should ideally be subjected to the same noise as that on other scalp electrodes, since any noise specific to the reference will be replicated to the other channels.
- The reference should ideally not be located too close to the region of interest of the researcher. That's because, since the amplitude of channels closer to the reference tends to be lower, placing it close to the region of interest might lessen the effects captured in this area.
- If the experiments are being done with a large number of electrodes (something close to a hundred), the electrode montage will cover nearly the whole head, and it's suggested that the reference then be not a single electrode, but the average of all channels.

4. **Artefact removal**

   Artifacts are signals with no cerebral origin which are embedded as noise into the EEG recordings, and can be subdivided into two categories: physiological, which originate from sources in the body, such as eye blinks, eye movements, head movements and heartbeats, and non-physiological, which originate from outside-world interference, such as power lines' electrical fields, equipment malfunction, and poor placement of the electrodes over the scalp.

   These artifacts can be mitigated by taking some precautions during the experiments, such as instructing subjects to suppress eye movement, and shielding the room from external electric interference, but their presence is inevitable. Many techniques have been developed to remove these artifacts in the preprocessing step, with Independent Component Analysis (ICA) being one of the most effective and widely used.

   In general terms, this technique consists in decomposing the EEG signals into a series of statistically independent components, classifying them as artefactual or neural related, removing the artefactual ones, and transforming the remaining components back to their original EEG signals format. This has the effect of cleaning the EEG data of many of its artifacts - especially those related to eye movement - while retaining much of the neuronal information present in the raw signals. Stone (2002) and Tharwat (2018) can be consulted for a more in-depth explanation.

2.2 Methodological Guidelines

In the EEG Classification problem, many different models can be developed for the classification task, each with its own methodologies and technicalities. To find the most accurate of them, some guidelines from Statistics and Machine Learning literature should be followed to ensure that the model selection process is fair and precise. In this Section, we review some of the most important of these guidelines.

2.2.1 k-fold Cross Validation

A classifier model is obtained by training it over a *train dataset*, where it learns a mapping between the EEG signals (or the feature vectors derived from it) and their corresponding class labels. This learning process is reflected in the parameters of the model, such as the coefficients of a Logistic Regressor, or the split points of a Decision Tree (Hastie *et al.*, 2009).

Once the model is trained, its performance can be evaluated on a *test dataset*, composed of samples not present in the train dataset. This evaluation consists in using the trained model to classify the unseen samples of the test dataset, and quantifying the results with metrics such as Accuracy, F1 Score, and AUC ROC, which can be seen as estimates of how well the model will generalize to new data.

However, the problem with this approach is that the fewer samples we have in our test dataset, the less reliable the metrics calculated over it are as estimates of the model's generalization performance. For instance, let's imagine we want to evaluate a newly proposed EEG classification model over a dataset with 100 trials. One way to do this evaluation is to train the model over the first 80 trials and test it over the last 20, obtaining an accuracy $\mathbf{Acc_a}$. Another way would be to train it over the last 80 trials, and evaluate it over the first 20, obtaining an accuracy $\mathbf{Acc_b}$. Since, in both cases, the test datasets are so small, $\mathbf{Acc_a}$ will likely be significantly different from $\mathbf{Acc_b}$, and it is not possible to know which one better reflects the model's true predictive performance.

To overcome this problem, the **k-fold Cross-Validation** procedure has been developed, where the data is split into k roughly equal-sized bins, and each k-th bin is used as a test dataset while the model is trained over the other $k-1$ bins (see Figure 2.6). This ensures all data available be used both for training and testing the model, and that our evaluation of its performance rests not on a punctual estimation, which is heavily dependent on the train/test split point, but on k different

values calculated over the whole dataset, and whose mean is a better estimate of the model's true generalization capabilities (Hastie *et al.*, 2009).

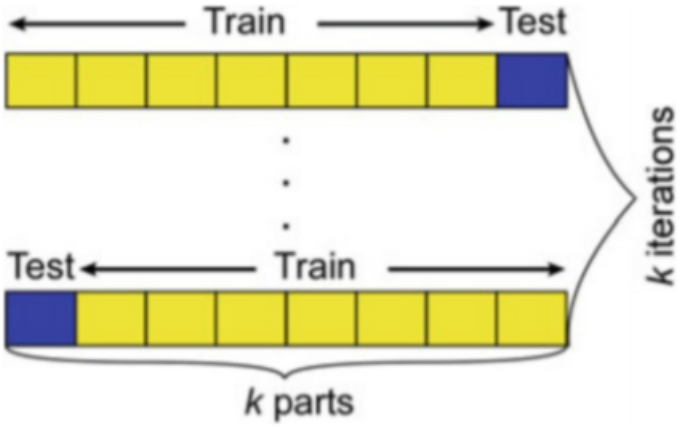

Figura 2.6: *k-fold Cross Validation (Hu e Zhang, 2019)*

2.2.2 Hyperparameters optimization

We call hyperparameters the parameters of a model that are not learned from the data during the training phase, but that must be set manually beforehand, like the regularization strength of a Lasso Regressor, the maximum depth of a Decision Tree, or the number of layers of a Neural Network, for example. Tunning these hyperparameters is an essential step in the modeling process, since the model's performance is heavily dependent upon how close its hyperparameters are to their optimal values (Bishop, 2006).

There are many techniques to make this hyperparameter tuning, with by far the most used being **Grid Search**, which is a model-agnostic, exhaustive search method that consists in defining a set of hyperparameters values one wants to investigate, training models with all possible combinations on this set, evaluating these models on a validation dataset, and choosing the combination that achieves the best performance over it (Hastie *et al.*, 2009).

An important aspect of this procedure is defining what the validation dataset is going to be. There are basically two options to do this, as follows:

- **Holdout**

  If we are on a scenario where data is abundant, the easier way to do the hyperparameters optimization is by simply dividing the available data into three datasets: a training one, which we will use to fit the model; a test one, which we will use to evaluate it; and a validation one, sometimes called the *holdout set*, used to check what the optimal combination of hyperparameters is - see Figure 2.7.

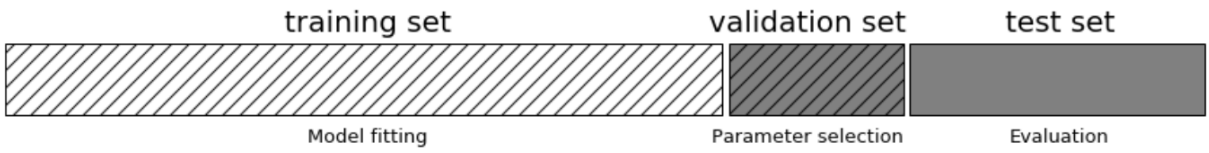

Figura 2.7: *Grid Search with holdout validation set (Mueller, 2020)*

- **Cross-Validation**

  If, however, we are dealing with a scenario where data is scarce - usually the case with EEG studies - the cross validation procedure described in Section 2.2.1 can be used to assist in

the Grid Search. First we divide the data into k folds, using each k-th fold as a test dataset and training the model over the other $k-1$ folds, just as before. The difference is that, when training the model, the data on the $k-1$ folds is then divided again into j folds (where usually $j < k$), and each j-th fold is used as a validation set, with the remaining folds being used to fit the model. This ensures all data available is used efficiently to both evaluate the model and find the best combination of its hyperparameters. A single iteration of this procedure is illustrated in Figure 2.8.

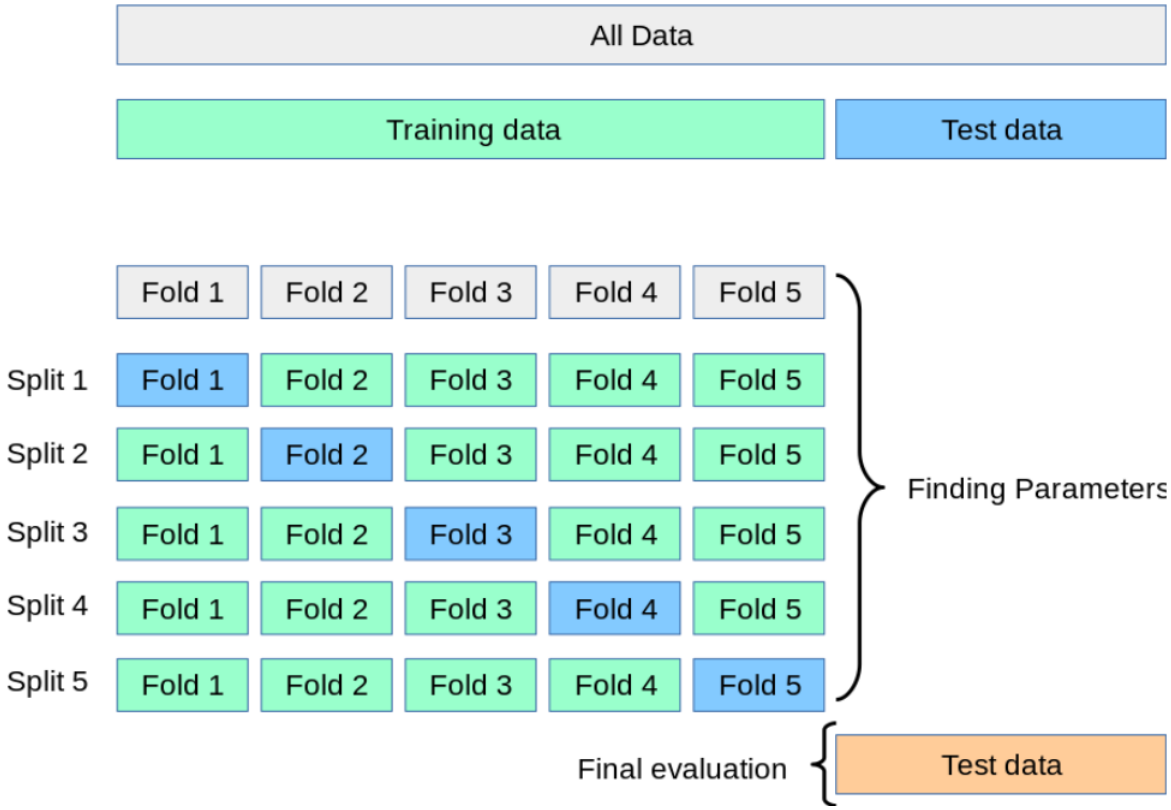

Figura 2.8: *One iteration of Grid Search with cross validation (Mueller, 2020)*

2.2.3 Critical Difference Diagram

When the goal of a study is to compare the performance of different models over multiple datasets, it's fundamental that theoretically sound statistical tests be made to ensure the validity of any conclusions made about which models are better than others. To this end, Critical Difference Diagrams are one of the most effective methods to accurately rank models in these multi-dataset scenarios (Demšar, 2006).

Basically, this method has three steps: first, it employs a Friedman Test to verify if, when ranking the models' performances over the datasets, there is any significant difference in their rankings or not; then, if there are, it employs pairwise Nemenyi tests to verify precisely where these differences lie; and, finally, it draws a diagram to visually show these differences and rank the models.

More specifically, suppose we are evaluating $\mathbf{k}$ models over $\mathbf{N}$ datasets. Let $\mathbf{r_j^i}$ be the rank of the j-th of $\mathbf{k}$ algorithms on the i-th of $\mathbf{N}$ data sets. The Friedman test compares the average ranks of the algorithms, $\mathbf{R_j} = \frac{1}{\mathbf{N}} \sum_{\mathbf{i}} \mathbf{r_i^j}$, through the Friedman statistic:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{2.1}$$

In Friedman test, the null hypothesis is that the performance of all models is the same, so that

all $\mathbf{R_j}$ are equal. Under this hypothesis, the Friedman statistic has a $\chi^2_{\mathbf{F}}$ distribution with $k-1$ degrees of freedom.

If the null hypothesis is rejected, we can proceed to the Nemenyi tests, where we compare all classifiers to each other. In this test, the performance of two classifiers is significantly different if their corresponding average ranks differ by at least the Critical Difference statistic:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{2.2}$$

where the variable $\mathbf{q}_\alpha$ is based on the Studentized Range statistic, according to what significance level α is being used. In other words, we calculate **CD** for given **k**, **N** and α, and, for all $\mathbf{i}, \mathbf{j} \in \{\mathbf{1}, \ldots, \mathbf{k}\}$, we say that model **i** is better than model **j** if $\mathbf{R_i} - \mathbf{R_j} > \mathbf{CD}$. Otherwise, we have that, given the available data, there is no significant difference in the models' performances.

Finally, once all Nemenyi tests have been done, the results can be visually represented with a Critical Difference Diagram, illustrated in Figure 2.9. In this diagram, we connect the groups of models that are not significantly different - that is, those where $\mathbf{R_i} - \mathbf{R_j} < \mathbf{CD}$ - with a thick line. In the Figure, for example, we have that, by average rankings, we would rank the models' performances as $3 > 5 > 4 > 2 > 1$. However, the Nemenyi tests show us that the difference in performance in models 3 and 5 is not statistically significant, the same happening with models 4, 2 and 1.

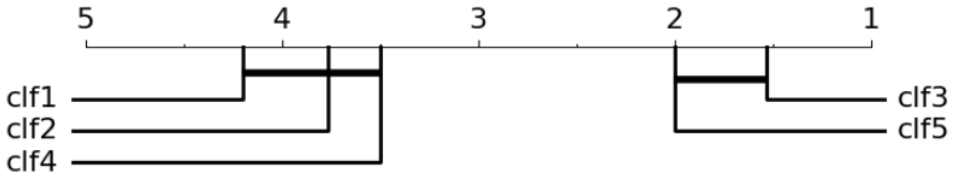

Figura 2.9: *Critical Difference Diagram example (Ismail Fawaz et al., 2019)*

2.3 Featurization

As discussed in Section 2.1.1, EEG signals can be seen as multivariate time series, a form of unstructured data that isn't accepted as input by the majority of traditional Machine Learning classifiers. To solve this, these signals can be transformed into structured feature vectors that summarize the information in the raw data in a form that can be fed to a traditional classifier. In this Section, we review some of the most important featurization methods for raw EEG data.

2.3.1 Fourier Transform

The Fourier Transform is a technique that allows us to extract frequency band information from a signal, which, in an EEG context, means that we can use it to generate features specifically for each of the brain waves described in Section 2.1.

To understand how the Transform work, we must first understand the Fourier Series. Let $f(t)$ be a periodic signal with period T and angular frequency $\omega = 2\pi/T$. If the integral of this signal over one cycle is finite (the so-called weak Dirichlet condition), then it is possible to express this signal as a Fourier Series $P(t)$, which is a sum of sines and cosines, as follows:

$$\begin{aligned} P(t) &= \frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t) + \ldots + b_1\sin(\omega t) + b_2\sin(2\omega t) + \ldots \\ &= \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left[a_n\cos(n\omega t) + b_n\sin(n\omega t)\right], \end{aligned} \tag{2.3}$$

where the coefficients a_n and b_n are the amplitudes of the sine and cosine components, respectively. To obtain them, we note that, for $P(t)$ to be as close as possible to $f(t)$, it must minimize the following error function E:

$$E^2 = \int_t^{t+T} [P(t) - f(t)]^2 dt, \tag{2.4}$$

where we can evaluate the integral over only one cycle of the signal because of its periodic behavior. Then, the values of a_n and b_n must satisfy:

$$\frac{\partial E^2}{\partial a_n} = 0 \quad \text{and} \quad \frac{\partial E^2}{\partial b_n} = 0 \tag{2.5}$$

With some algebraic manipulations, it's easy to show that the coefficients will then be given by (Van Drongelen, 2018)

$$a_n = \frac{2}{T} \int_T f(t) \cos(n\omega t) dt \tag{2.6}$$

$$b_n = \frac{2}{T} \int_T f(t) \sin(n\omega t) dt \tag{2.7}$$

In other words, the signal $f(t)$ will be approximated by a sum of sines and cosines of frequencies $n\omega$ and amplitudes a_n and b_n, respectively. Figure 2.10 illustrates this for a square wave signal, approximated by the sum of five sine waves with different amplitudes and frequencies.

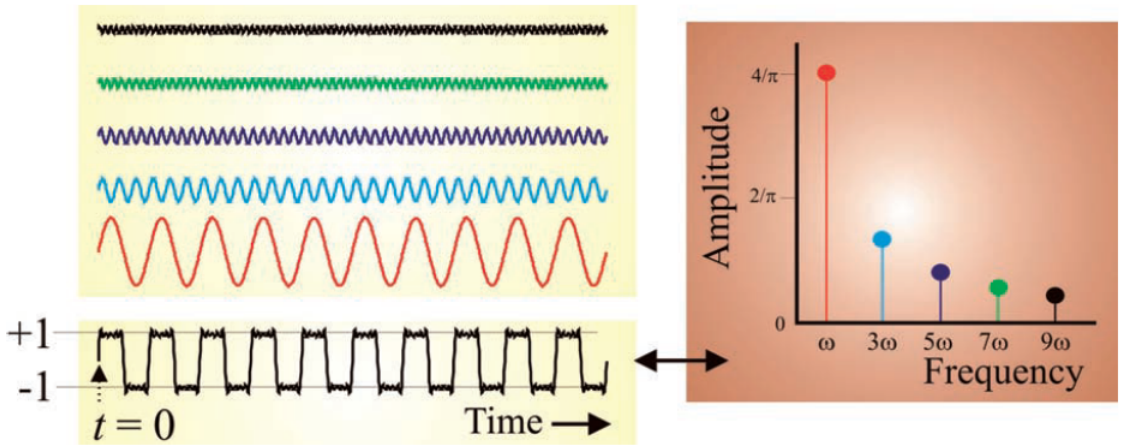

Figura 2.10: *Square wave signal approximated by a Fourier Series (Van Drongelen, 2018)*

Using Euler's Identity $e^{jx} = \cos(x) + j\sin(x)$, the Fourier Series can be more compactly written in the following complex notation:

$$P(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega t} \tag{2.8}$$

$$c_n = \frac{1}{T} \int_T f(t) e^{-jn\omega t} dt \tag{2.9}$$

Note that both the equations with real and with complex notations are equivalent - that is, substituting Euler's Identity into equations 2.8 and 2.9, we get exactly equations 2.3, 2.6 and 2.7.

So, in short, periodic signals can be modelled by Fourier Series as a sum of sine and cosine terms over discrete frequency values ($n\omega$). To extend this to aperiodic signals, we can simply make $T \to \infty$. This has the result of making $\omega \to d\omega$, such that the summation in Equation 2.8 can be substituted by an integration. With some algebraic manipulation (the reader is referred to Van Drongelen (2018)

for the full derivation) the Fourier Series becomes:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \tag{2.10}$$

This is called the Fourier Transform, which can be understood as the limiting case of the complex Fourier series when the period of the analyzed signal grows to infinity. More specifically, just as the Fourier Series allows us to extract the amplitudes of the (discrete) frequencies that composes a periodic signal, the Fourier Transform allows us to extract the power of the (continuous) frequencies that composes an aperiodic one, as illustrated in Figure 2.11, in what's called the *Power Spectrum* of a signal.

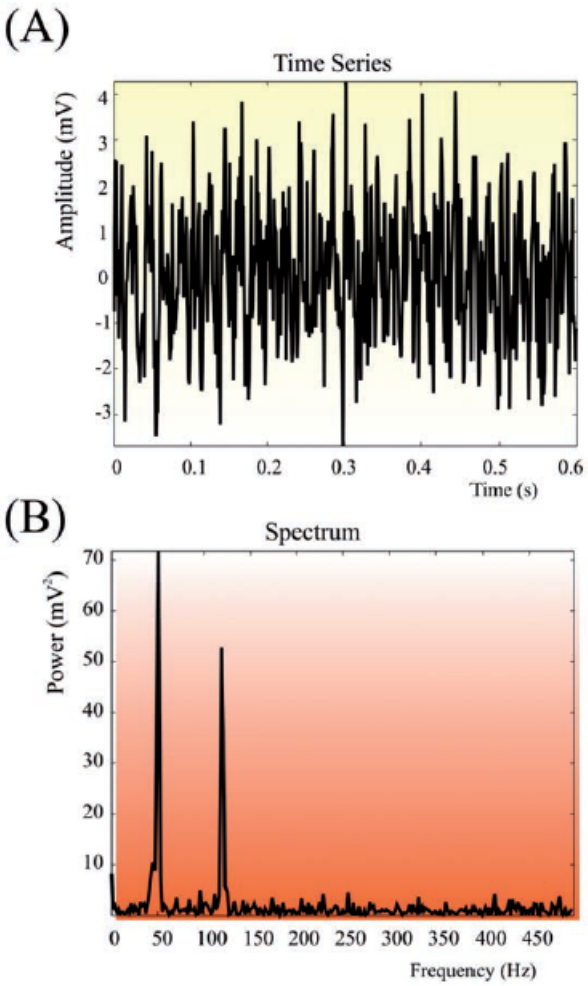

Figura 2.11: *Aperiodic time series, and power of its constituting frequencies. Note how the two sinusoidal signals at 50 and 120 Hz, that are buried in noise (A), become clearly visible in the frequency domain (B) (Van Drongelen, 2018)*

In an EEG Classification context, the Fourier Transform can be used to determine, for each EEG channel, which brain waves are the most prevalent. More specifically, in this study we will create 10 features per channel using Fourier Transforms, the first 5 being the **Power Spectral Intensity** of the brain waves, which are the total power contained in the frequency band associated with each wave, and the last 5 being the **Relative Intensity Ratio** of the brain waves, which are the Power Spectral Intensity of each wave divided by the total power of the signal (Bao *et al.*, 2011).

2.3.2 Wavelet Transform

Wavelets are mathematical functions that cut up data into different frequency components, similar to the Fourier Transform, but with the capacity of studying each component with a resolution matched to its scale - that is, it can capture both gross features of the signal by sliding a large window over it, and also fine details by sliding a small one (Graps, 2020).

To understand how wavelets work and why they are an important complement to the Fourier Transform, we first highlight one of the biggest limitations of the Fourier Transform, which is the fact that it was developed to be used with stationary signals, that is, those where the frequency content of the signal does not change with time. This means that the transformation of a signal with time-varying frequencies will result in a power spectrum that does not capture this time dependency, an information that is extremely important in an EEG context, where the latency between a stimulus and the brain's response to it might be very useful for a classification model (Polikar, 1996).

To illustrate this, consider the signal given by Equation 2.11 and plotted in Figure 2.12:

$$x(t) = \cos(2\pi * 10t) + \cos(2\pi * 25t) + \cos(2\pi * 50t) + \cos(2\pi * 100t) \tag{2.11}$$

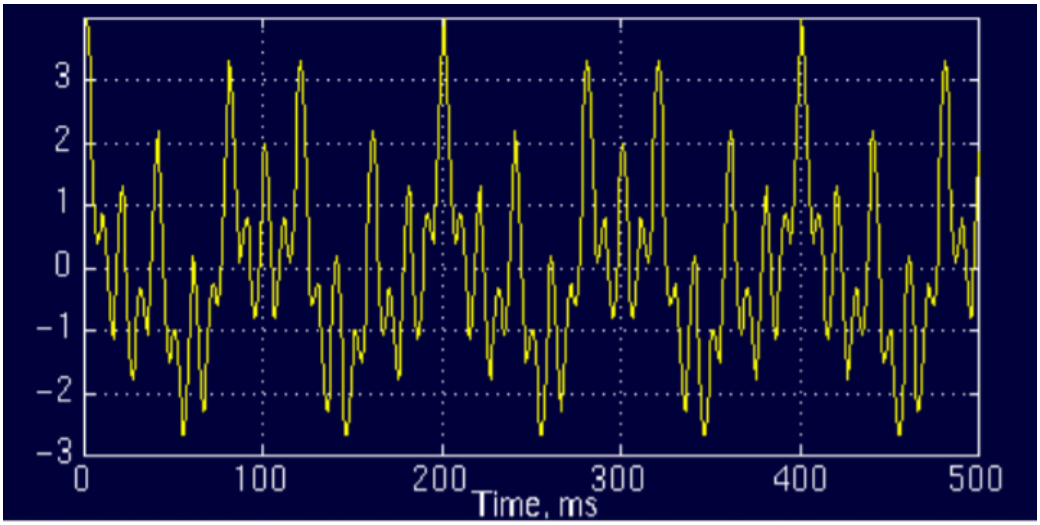

Figura 2.12: *Stationary signal (Polikar, 1996)*

This is a stationary signal, since its constituting frequencies of 10, 25, 50 and 100 Hz exist at all time points. The Fourier Transform of this signal is given in Figure 2.13. Note the clear delimitation of the four frequency components of the signal.

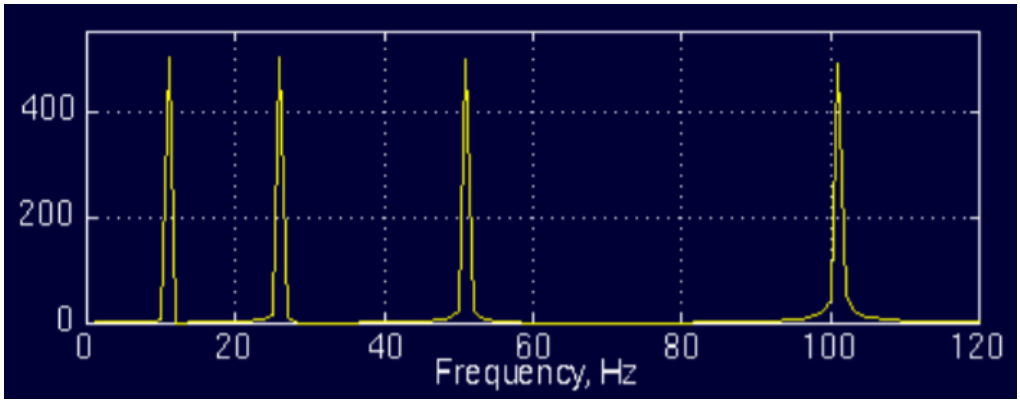

Figura 2.13: *Stationary signal frequency spectrum (Polikar, 1996)*

Now, consider a non-stationary signal with the same four frequency components, but occurring

at four different time intervals: The interval 0 to 300 ms has a 100 Hz sinusoid; the interval 300 to 600 ms has a 50 Hz sinusoid; the interval 600 to 800 ms has a 25 Hz sinusoid; and finally the interval 800 to 1000 ms has a 10 Hz sinusoid. This signal is illustrated in Figure 2.14.

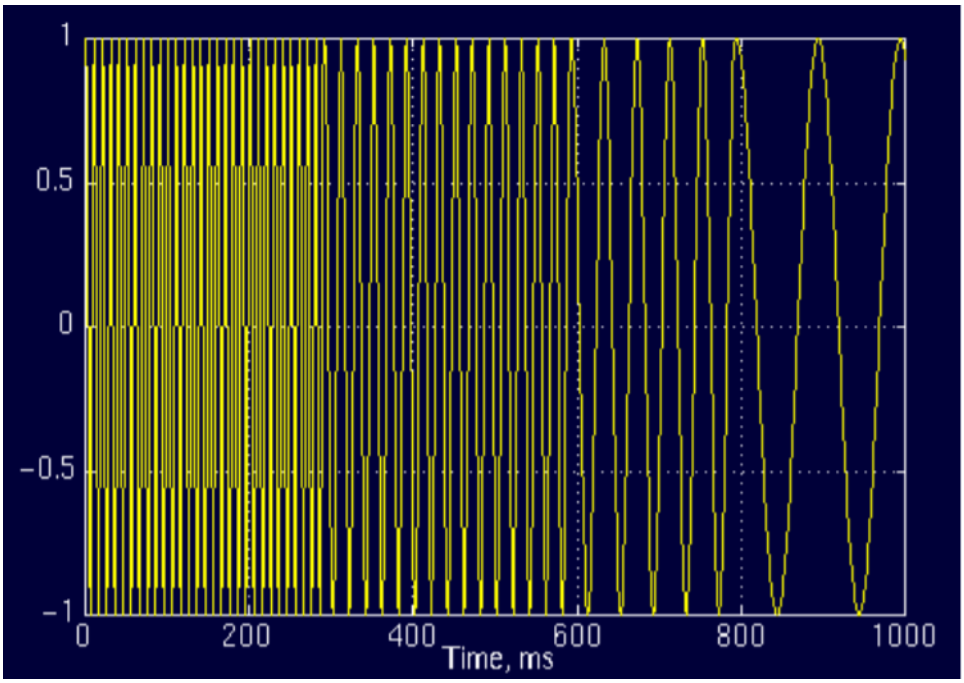

Figura 2.14: *Non-stationary signal (Polikar, 1996)*

Its Fourier Transform is presented in Figure 2.15. Note that, despite little ripples (caused mainly by changes from one frequency component to another) and slightly higher amplitudes for the higher frequency components (caused mainly by the fact they last a little longer than the lower frequencies), the power spectrum is very similar to that of the stationary signal. In other words, the Fourier Transform can tell us what frequency components exist, but not when they manifest in the signal (Polikar, 1996).

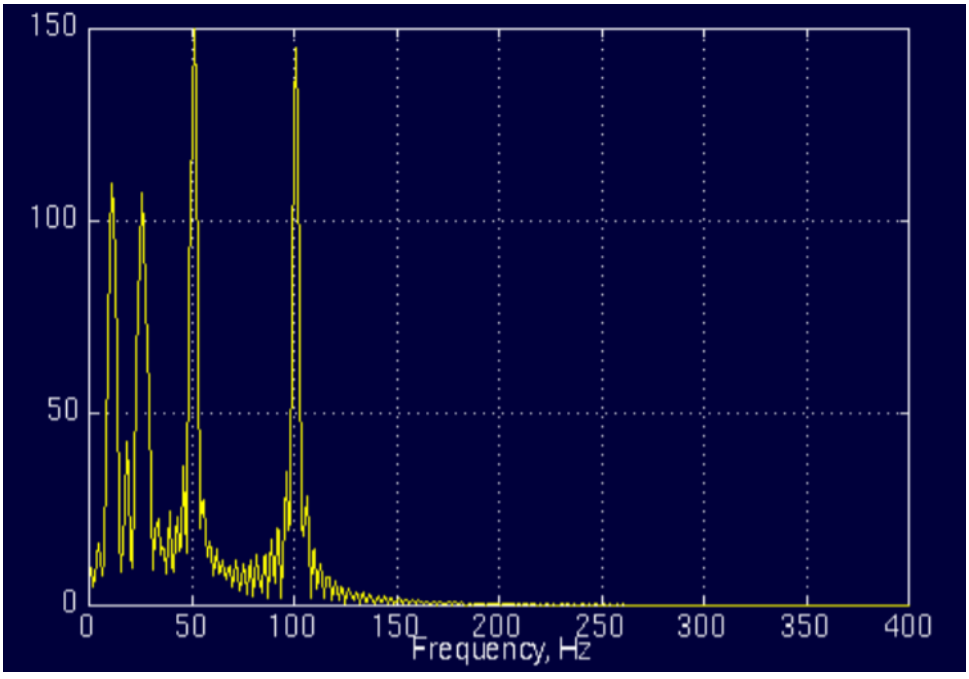

Figura 2.15: *Non-stationary signal frequency spectrum (Polikar, 1996)*

Some adaptations can be done to the traditional Fourier Transform in order for it to represent this time-frequency information, such as the Short Time Fourier Transform (Mertins e Mertins, 1999). The Wavelet Transform, on the other hand, has this capability naturally built into it, and is widely used today on problems where time-frequency representation of non-stationary signals is important.

More specifically, the Continuous Wavelet Transform (CWT) is defined as:

$$CWT_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^* \left(\frac{t-\tau}{s} \right) dt, \tag{2.12}$$

where $x(t)$, the original signal, is transformed according to a function of two parameters, τ and s, the so-called translation and scale parameters, respectively. $\psi(t)$ is the Mother Wavelet, a transforming function that defines the CWT.

First, let's understand the scale parameter. Just as the scale in maps, high scales correspond to a non-detailed global view of the signal, and low scales correspond to a more detailed view. In mathematical terms, scaling either dilates or compress a function - in this case, the Mother Wavelet $\psi^* \left(\frac{t-\tau}{s} \right)$. Since s is in the denominator, raising it means 'stretching out' the signal, giving it smaller frequency, and lowering it means compressing it, giving it a higher frequency. This is illustrated in Figure 2.16 for a cosine wave - the bigger the scale parameter, the lower is the signal's frequency.

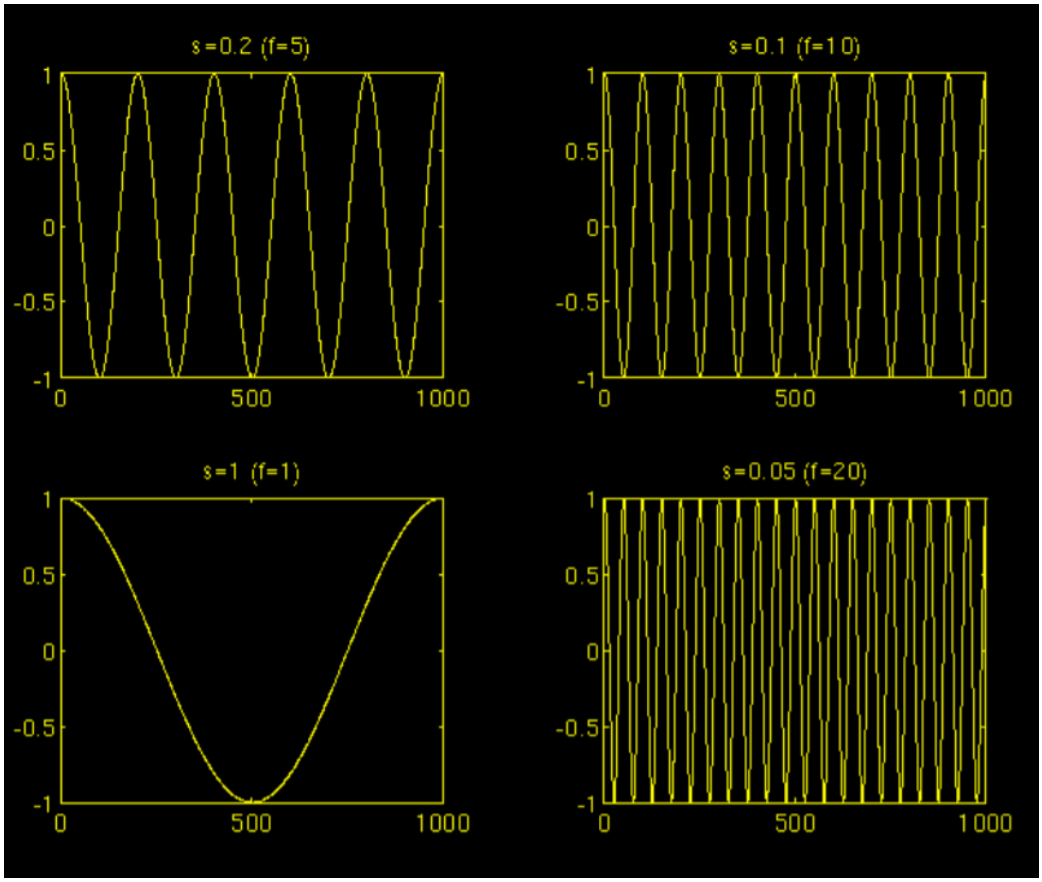

Figura 2.16: *Influence of scale parameter on a cosine signal (Polikar, 1996)*

The translation parameter refers to the position of the Mother Wavalet as it slides over the signal. As τ is increased, the Wavelet moves to the right, so that the integral in the CWT can be evaluated over different parts of the signal.

And finally, the Mother Wavelet itself is a finite, oscillatory function that will be convoluted with the signal to transform it. There are many different possible wavelet functions, with one of the most used, specially in EEG studies, being the **Daubechies 4** (db4), illustrated in Figure 2.17

The computation of a CWT works as follows: let $x(t)$ be the signal to be analyzed. Once a

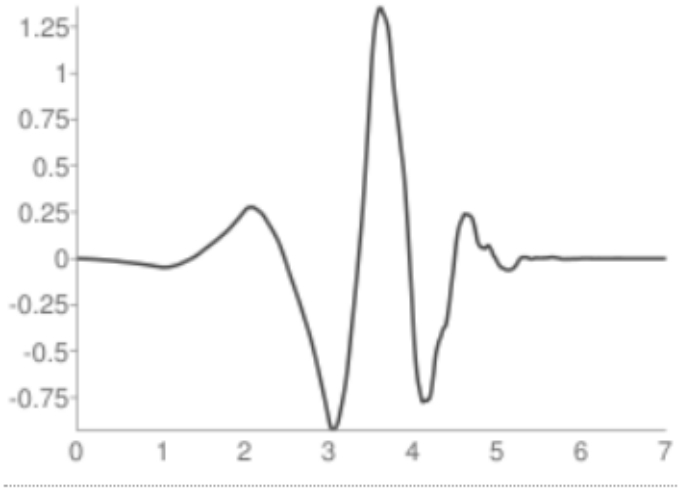

Figura 2.17: *Daubechies 4 (db4) Wavelet*

Mother Wavelet is chosen, the computation starts with $s = 1$ and will continue for increasing values of s - that is, the analysis will start from high frequencies and proceed towards low frequencies. This first value of s will correspond to the most compressed wavelet. As the value of s is increased, the wavelet will dilate.

The wavelet is placed at the beginning of the signal at the point which corresponds to $time = 0$. The wavelet function at scale 1 is multiplied by the signal and then integrated over all times. The result of the integration is then multiplied by the constant number $1/\sqrt{s}$. This multiplication is for energy normalization purposes so that the transformed signal will have the same energy at every scale. The final result is the value of the transformation, i.e., the value of the continuous wavelet transform at time zero and scale $s = 1$. In other words, it is the value that corresponds to the point $\tau = 0$ and $s = 1$ in the time-scale plane.

The wavelet at scale $s = 1$ is then shifted towards the right by τ amount to the location $t = \tau$, and the CWT is computed to get the transform value at $t = \tau$ and $s = 1$ in the time-scale plane. This procedure is repeated until the wavelet reaches the end of the signal. One row of points on the time-scale plane for the scale $s = 1$ is then completed. After this, s is increased by a small value, and the above procedure is repeated. Once this process is done for all desired values of s, the CWT of the signal has been calculated.

Figures 2.18 and 2.19 illustrates this computation. In 2.18 the scale value is 1, corresponding to the lowest scale, or highest frequency, where the Mother Wavelet, in blue, is at its most compressed state. The Wavelet then slides over the signal as different values of τ are used, and at each point it is multiplied by the signal and integrated over time. The result of this integration will be very high at points where this frequency component exists in the signal, and very low everywhere else.

After this process is done for $s = 1$, it is then repeated for all other desired values of s. In Figure 2.19, it's illustrated for $s = 5$. Notice how the wavelet is now more 'stretched out', meaning it represents a lower frequency component. The result of its multiplication with the signal and integration over time will now be higher at time points where this new frequency is present. And so on for all s values to be evaluated.

As a more practical example, let $x(t)$ be the non-stationary signal constituted by 4 different frequency components plotted in Figure 2.20. The CWT of this signal, calculated with the procedure just described, is presented in Figure 2.21.

Note how, for lower translation values - that is, for the portion of the signal closer to $time = 0$ - the CWT indicates only the existence of high frequency (low scale) components, while for bigger translations values, it shows the presence of lower frequency (high scale) components, as was expected.

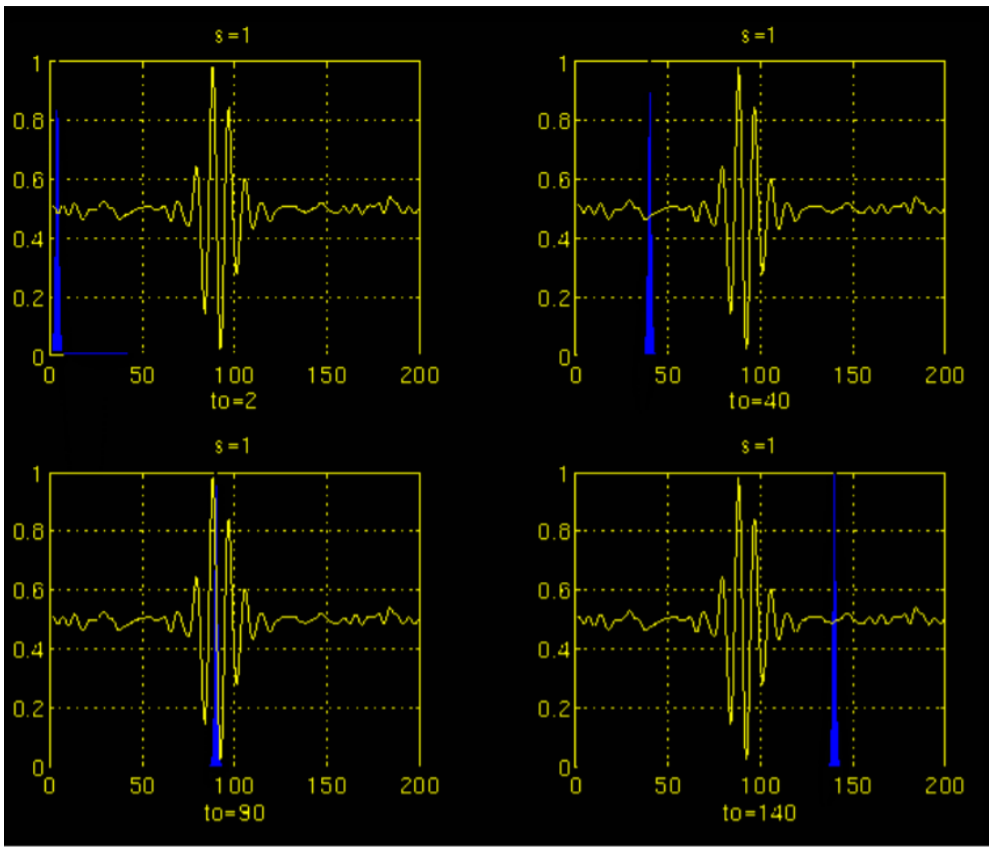

Figura 2.18: *Example of CWT computation for* $s = 1$ *(Polikar, 1996)*

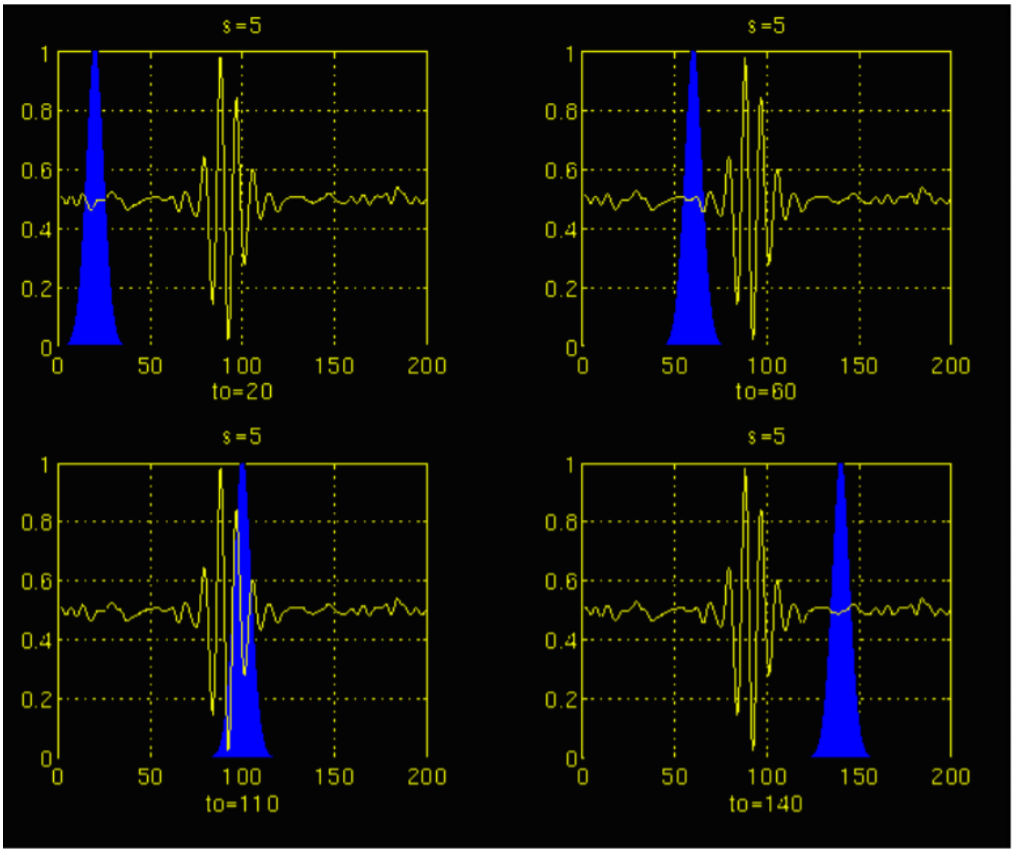

Figura 2.19: *Example of CWT computation for* $s = 5$ *(Polikar, 1996)*

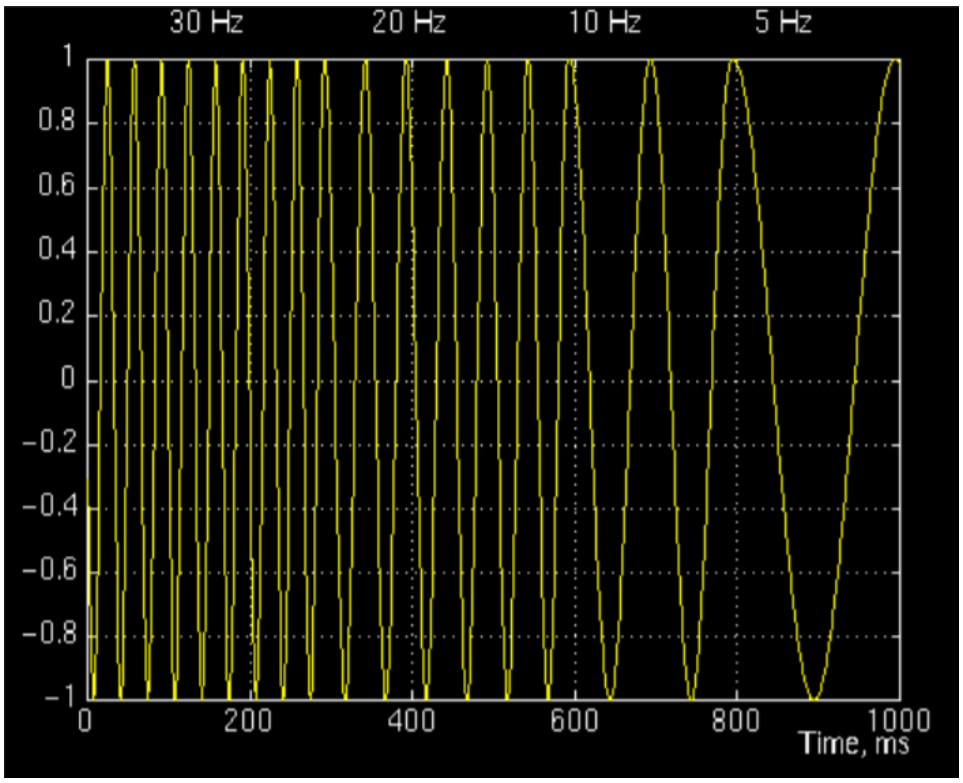

Figura 2.20: *Example of non-stationary signal for CWT computation (Polikar, 1996)*

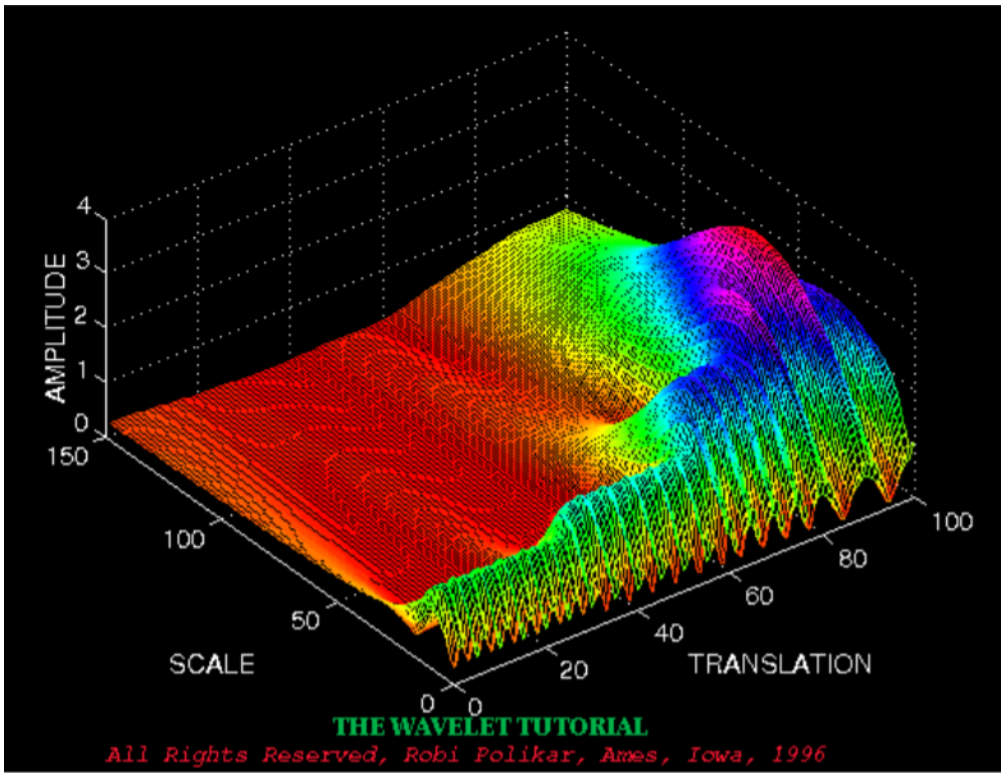

Figura 2.21: *CWT for the signal in Figure 2.20 (Polikar, 1996)*

In an EEG Classification context, what's more commonly used is the Discrete Wavelet Transform (DWT), an adaptation of the CWT where only a finite, discrete quantity of scale and translations parameters are evaluated. More specifically, there is a computationally efficient method of calculating this discrete transformation called the Fast Wavelet Transform, which involves the use of filter

banks and sequential subsampling of the signal, that gives us all the amplitudes for different frequencies (called the Wavelet Coefficients) concatenated into a single vector. The reader is referred to Addison (2017) for a more in-depth explanation.

So, in short, for every EEG channel we will calculate the Discrete Wavelet Transform, which will give us a vector of Wavelet Coefficients containing information about the frequency components of the signal. From each of these vectors we will calculate 11 features: the 5th, 25th, 75th and 95th percentile, the median, the mean, the minimum, the maximum, the standard deviation, the variance, and the root mean square of the Wavelet Coefficients.

2.3.3 Hjorth Parameters

Introduced by Bo Hjorth in 1970, the Hjorth Parameters are three parameters calculated over the time domain of an EEG signal that have been shown to aid in increasing the accuracy of classification models in tasks related to sleep analysis (Hamida *et al.*, 2015), brain-computer interfacing (Vidaurre *et al.*, 2009), and seizure detection (Cecchin *et al.*, 2010).

Letting $y(t)$ be an EEG signal, the three parameters are (Hjorth, 1970):

- **Activity**, representing the mean power of the signal:

$$\text{Activity} = \text{var}(y(t)) \tag{2.13}$$

- **Mobility**, representing the standard deviation of the signal's slope with reference to the standard deviation of its amplitude:

$$\text{Mobility} = \sqrt{\frac{\text{var}\left(\frac{dy(t)}{dt}\right)}{\text{var}(y(t))}} \tag{2.14}$$

- **Complexity**, representing the change in the signal's frequency. More precisely, the parameter compares the signal's similarity to a pure sine wave, where the value converges to 1 the more similar it gets:

$$\text{Complexity} = \frac{\text{Mobility}\left(\frac{dy(t)}{dt}\right)}{\text{Mobility}\,(y(t))} \tag{2.15}$$

Therefore, for each EEG channel, we will compute these 3 parameters.

2.3.4 Dynamic Time Warping

Dynamic Time Warping (DTW) is a well known and widely used algorithm to obtain a time-series similarity measure which minimizes the effects of shifting and distortion in time by allowing non-linear transformation of time series in order to detect similar shapes with different phases, as illustrated in Figure 2.22. Originally developed to deal with automatic speech recognition, it is now used in many different fields where it is important to cope with time deformations and different speeds associated with time-dependent data (Senin, 2008).

The objective of DTW is to compare two time series $X = (x_1, x_2, \ldots, x_N)$ of length $N \in \mathbb{N}$ and $Y = (y_1, y_2, \ldots, y_M)$ of length $M \in \mathbb{N}$. To compare the two sequences, one needs a local cost measure, sometimes also referred to as local distance measure, which is defined to be a function $c(x, y)$ that is small (low cost) if x and y are similar to each other, and is large (high cost) otherwise. Typical cost measures are the Euclidian distance and the Manhattan distance (absolute value of the difference between the time points) (Müller, 2007).

Evaluating the local cost measure for each pair of elements of the sequences X and Y, we obtain the cost matrix $C \in \mathbb{R}^{N \times M}$ defined by $C(n, m) = c\,(x_n, y_m)$, as illustrated in Figure 2.23. The goal

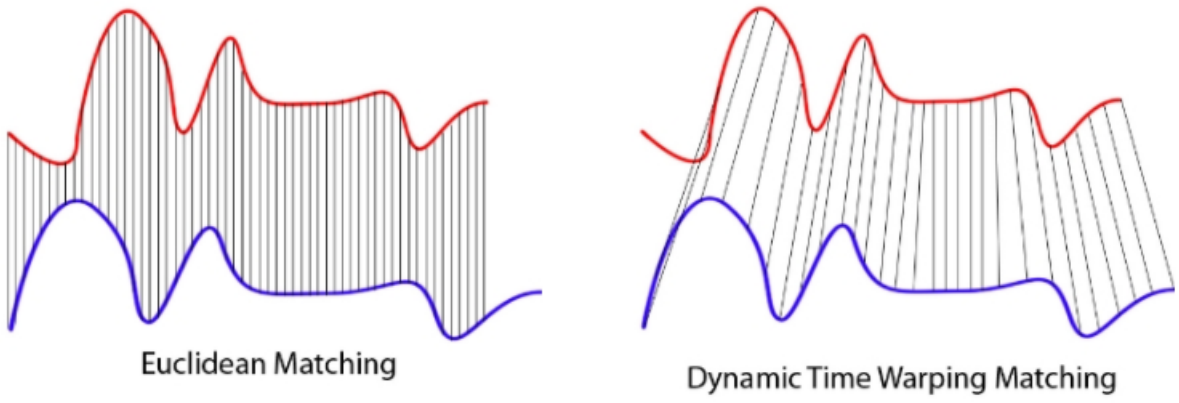

Figura 2.22: *Euclidean matching, with one-to-one matching of the time points, and DTW matching, which asynchronously adjusts parts of the curves to match up similar shapes (Tolpygo, 2016)*

of the DTW algorithm is to then find an alignment between X and Y having minimal overall cost. This alignment can be formally defined as a sequence $p = (p_1, \ldots, p_L)$, called a *warping path*, with $p_\ell = (n_\ell, m_\ell) \in [1:N] \times [1:M]$ for $\ell \in [1:L]$ satisfying the following three conditions:

1. **Boundary condition**: $p_1 = (1,1)$ and $p_L = (N, M)$.
2. **Monotonicity condition**: $n_1 \leq n_2 \leq \ldots \leq n_L$ and $m_1 \leq m_2 \leq \ldots \leq m_L$
3. **Step size condition**: $p_{\ell+1} - p_\ell \in \{(1,0), (0,1), (1,1)\}$ for $\ell \in [1:L-1]$

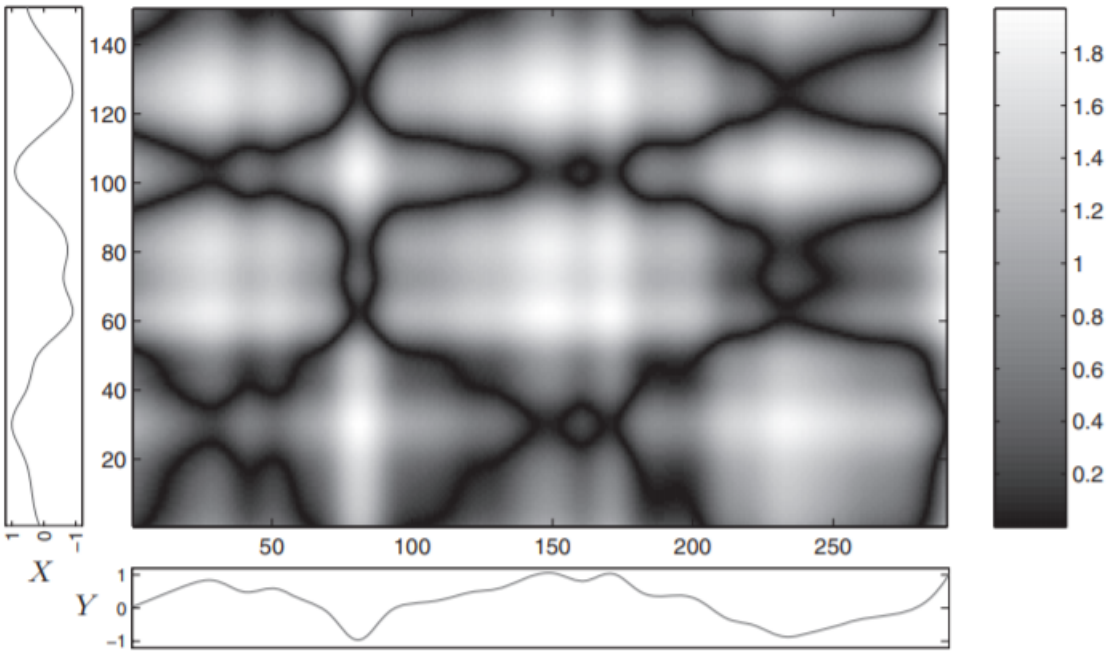

Figura 2.23: *Cost matrix of two time-series X (vertical axis) and Y (horizontal axis) using the Manhattan distance as local cost measure (Müller, 2007)*

In other words, as explained in Müller (2007):

> A warping path $p = (p_1, \ldots, p_L)$ defines an alignment between two sequences $X = (x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$ by assigning the element x_{n_ℓ} of X to the element y_{m_ℓ} of Y. The boundary condition enforces that the first elements of X and Y as well as the last elements of X and Y are aligned to each other. In other words, the alignment

refers to the entire sequences X and Y. The monotonicity condition reflects the requirement of faithful timing: if an element in X precedes a second one this should also hold for the corresponding elements in Y, and vice versa. Finally, the step size condition expresses a kind of continuity condition: no element in X and Y can be omitted and there are no replications in the alignment (in the sense that all index pairs contained in a warping path p are pairwise distinct)

Figure 2.24 illustrates the three conditions.

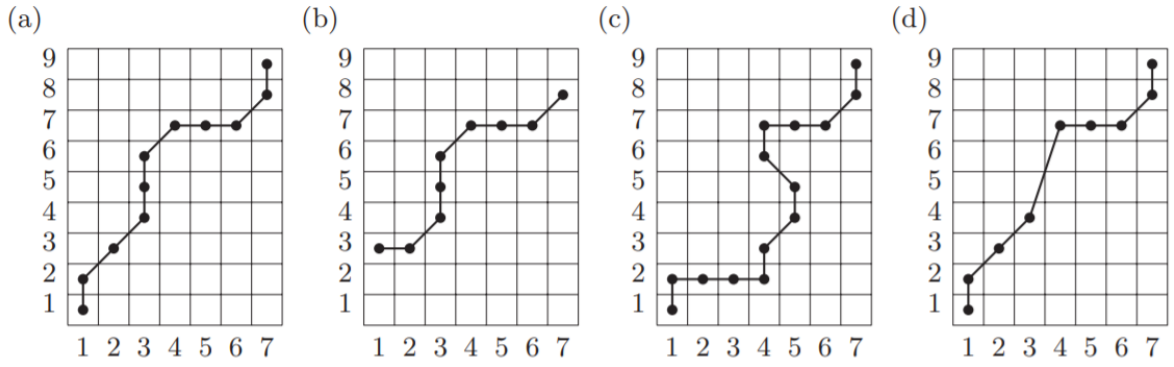

Figura 2.24: *Illustration of warping paths for some sequence X of length $N = 9$ and some sequence Y of length $M = 7$.* ***(a)*** *Admissible warping path satisfying the three DTW conditions.* ***(b)*** *Boundary condition is violated.* ***(c)*** *Monotonicity condition is violated.* ***(d)*** *Step size condition is violated (Müller, 2007)*

The total cost $c_p(X, Y)$ of a warping path p between X and Y with respect to the local cost measure c is defined as:

$$c_p(X, Y) = \sum_{\ell=1}^{L} c\left(x_{n_\ell}, y_{m_\ell}\right) \tag{2.16}$$

Furthermore, an optimal warping path between X and Y is a warping path p^* having minimal total cost among all possible warping paths. The *DTW* distance $\mathrm{DTW}(X, Y)$ between X and Y is then defined as the total cost of p^*:

$$\begin{aligned} \mathrm{DTW}(X, Y) &= c_{p^*}(X, Y) \\ &= \min\left\{c_p(X, Y) \mid p \text{ is an } (N, M) \text{ -warping path }\right\} \end{aligned} \tag{2.17}$$

So, in short, the DTW between two sequences can be seen as a metric which will be lower the more similar the sequences are. In an EEG Classification context, note that, unlike the methods described in Sections 2.3.1, 2.3.2 and 2.3.3, the DTW does not return a feature vector for each EEG channel. Instead, it can be used to define, for each channel of an unlabelled trial, what is the labeled trial with a most similar corresponding channel, such that the unlabelled trial can be classified with the same label of the trial with the most channels similar to it.

More specifically, consider a binary classification problem where an EEG trial can be classified as either belonging to group A or B. Let there be k labeled trials, each with n channels. To classify an unlabelled trial, we calculate the DTW of each of its channels in relation to the corresponding channel in all of the k trials, such that we end up with a $n \times k$ matrix where the element x_{ij} correspond to the DTW of the i-th channel to the corresponding channel of the j-th labeled trial. Then, using a Nearest Neighbour approach (see Section 2.4.1), each channel can be individually classified as belonging to class A or B according to what labeled channel it is most similar to (that is, the channel it has the lowest DTW with). Finally, the unlabelled trial can then be classified in the group most of its channels belong to.

2.3.5 WEASEL+MUSE

The WEASEL+MUSE classifier, which stands for *Word ExtrAction for time SEries cLassification plus Multivariate Unsupervised Symbols and dErivatives*, is a method proposed in 2017 which is able to extract a vector of highly discriminative features out of multivariate time series (MTS) that can be used in classification tasks. It has shown to achieve state-of-the-art performance on a variety of MTS classification problems, while still being computationally easy to calculate (Schäfer e Leser, 2017).

To put it simply, WEASEL+MUSE is to multivariate time series what the Bag-of-Words (BoW) model is to text data, so a brief review of BoW is a good starting point to understand the WEASEL+MUSE. Basically, the BoW represents every document from a corpus as a vector whose length is equal to the vocabulary of the corpus, and the values are the counts of each word in this vocabulary. To illustrate this, consider a corpus formed by the following three documents:

1. The elephant sneezed at the sight of potatoes.
2. Bats can see via echolocation. See the bat sight sneeze!
3. Wondering, she opened the door to the studio.

The vocabulary of this corpus is composed by all (tokenized) words that appear at least once in at least one of the documents. For this particular example, the vocabulary would be the vector [at, bat, can, door, echolocation, elephant, of, open, potato, see, she, sight, sneeze, studio, the to, via, wonder] - notice how all documents can be reconstructed just with the words in the vocabulary. In the tradional BoW model, every document is then represented as a vector with the word counts of the vocabulary, as illustrated in Figure 2.25. A more in-depth explanation of BoW can be found on Bengfort *et al.* (2018).

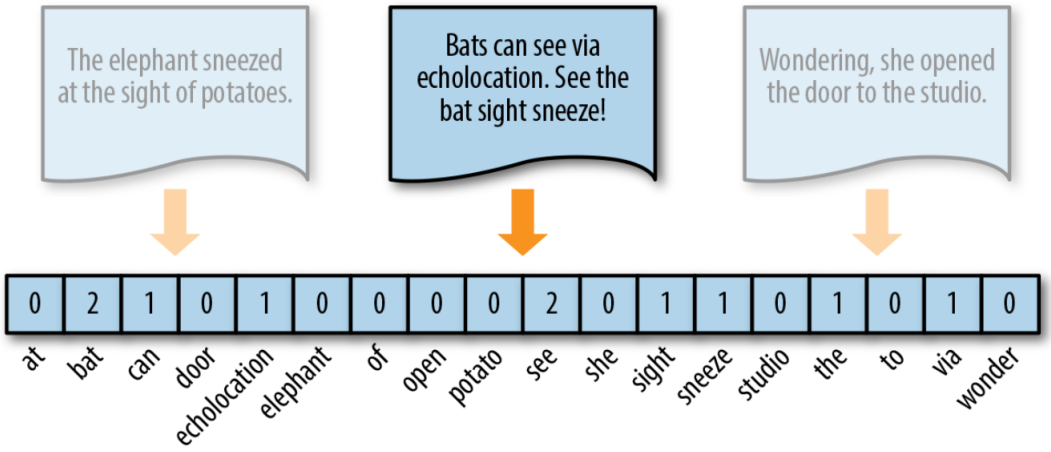

Figura 2.25: *Bag-of-Words modelling of a three document corpus (Bengfort et al., 2018)*

In a MTS context, the corpus is the dataset of all multivariate time series, the documents are the different MTS, and the words are segments obtained by sliding a window over each individual dimension of the MTS. WEASEL+MUSE uses two main techniques to accomplish this: Symbolic Fourier Approximation (SFA), to convert the time series to words, and Bag-of-Patterns (BoP), to extract subsequences of the data and transform them into a vector of numeric values.

First, The Symbolic Fourier Approximation is used to create an alphabet, the equivalent of a vocabulary in a text data context. It works as follows (Schäfer e Högqvist, 2012):

1. All individual time series which form the MTS are decomposed into a sum of sine waves by using a Fourier Transform (see Section 2.3.1). The first few sine waves correspond to slowly changing Sections and represent the coarse distribution, while later waves represent rapid

changes like gaps or noise. Thus the use of only the first few sine waves produces a good approximation of a time series, so that we can consider only the first w Fourier Coefficients in our decomposition.

2. After obtaining the coefficients for all time series, these values are binned into c discrete bins for each of the w coefficients, where the first bin is named A, the second B, and so on. This way, all the time series will be transformed into a word of length w made out of c possible different letters, where each letter will be determined according to what bin the Fourier Coefficients of the time series fall into.

Figures 2.26 and 2.27 illustrate this process. Once it is done, we have an alphabet - that is, a binning of values over all Fourier Coefficients that translate them into letters - that can be used to transform any time series into a word.

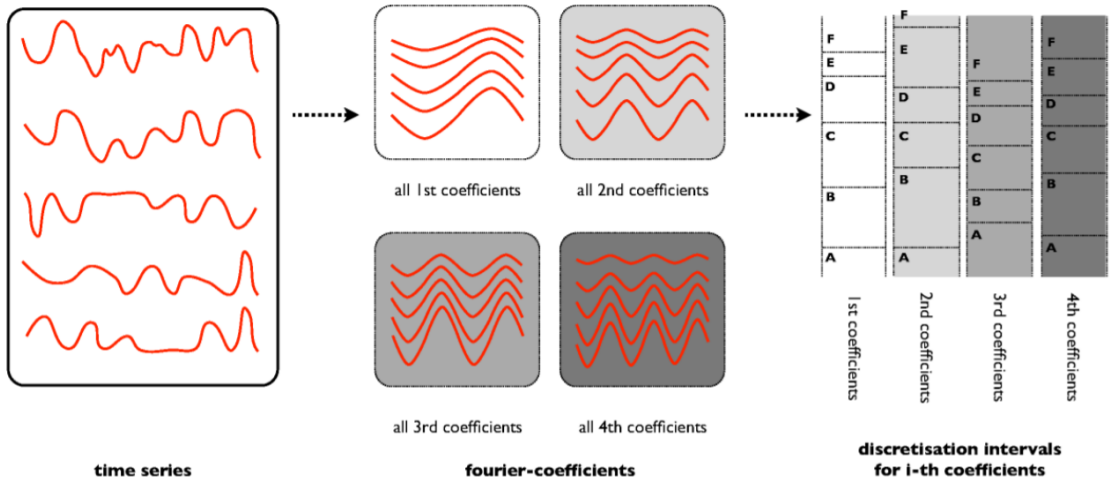

Figura 2.26: *Fourier Transformation and binning of the coefficients for the contruction of a SFA alphabet. In this example, we have $w = 4$ (only the first 4 Fourier coefficients are used in the time series transformation) and $c = 6$ (the coefficient values are binned into 6 distinct bins), therefore the time series are translated into words of length 4, composed by 6 possible different letters (Schäfer e Högqvist, 2012)*

With the ability to translate time series into words, the Bag-of-Patterns method is used to translate not whole time series, but individual subsequences of it into words, and them create a histogram of these words which will represent the MTS. It works as follows (Lin *et al.*, 2012):

1. First, the MTS is split into its dimensions, where each dimension can be seen as an univariate time series. From each time series, subsequences are extracted via a sliding window, as illustrated in Figure 2.28. Specifically in WEASEL+MUSE, these subsequences are also extracted from the derivative of the original time series, and many different windows sizes are used, in order to increase the final accuracy of the model.

2. Each of these subsequences are then transformed into words using the SFA transform. Specifically in WEASEL+MUSE, the number of the dimension of the time series and the size of the window used to generate the subsequence are added as a preffix to the words, which allow them to more adequately represent the structure of the MTS. For instance, '1-12-aaa' refers to the 'aaa' word generated on a subsquence of the first dimension of a MTS with a sliding window of size 12, while '5-8-aaa' represents a 'aaa' word generated on a subsequence of the fifth dimension of a MTS with a sliding window of size 8.

3. Finally, once all subsequences of all dimensions of the MTS are translated into words, a histogram is created with the words, and a vector with their counts, very similar to that of the Bag-of-Words model in text data, is used to represent the MTS.

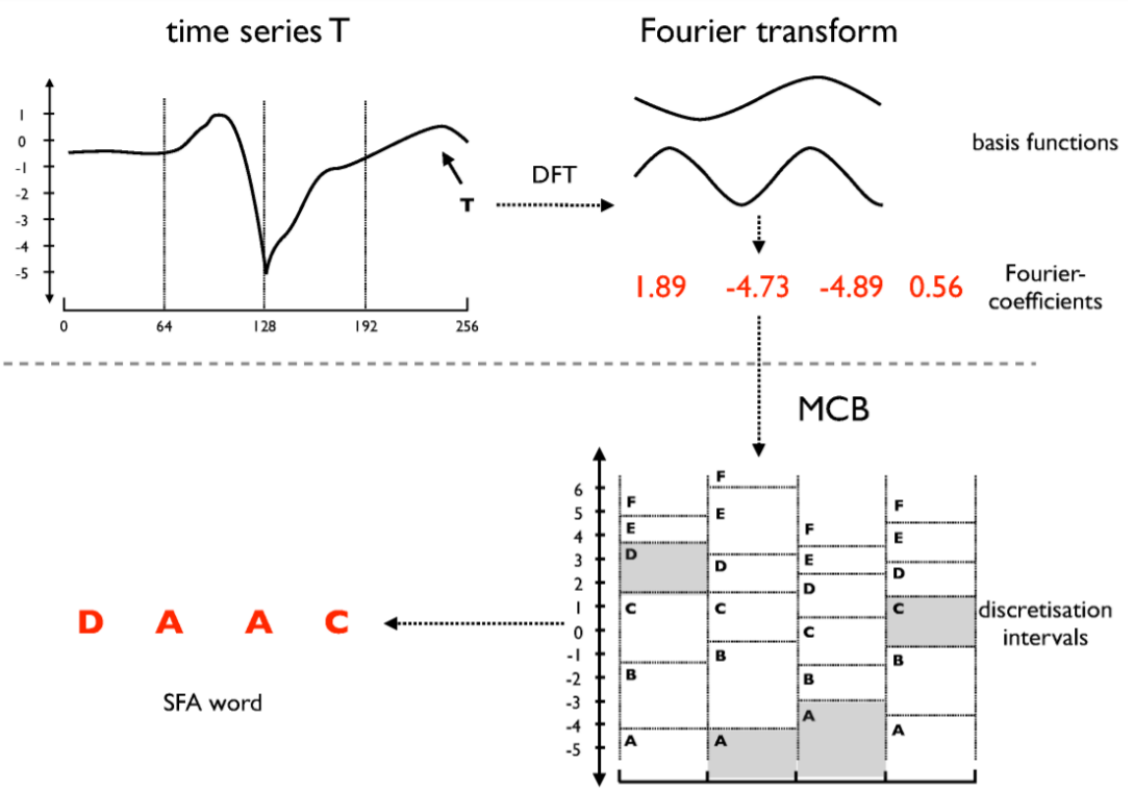

Figura 2.27: *SFA Transformation of a new time series once the alphabet (binning intervals for each Fourier coefficient) has been created (Schäfer e Högqvist, 2012)*

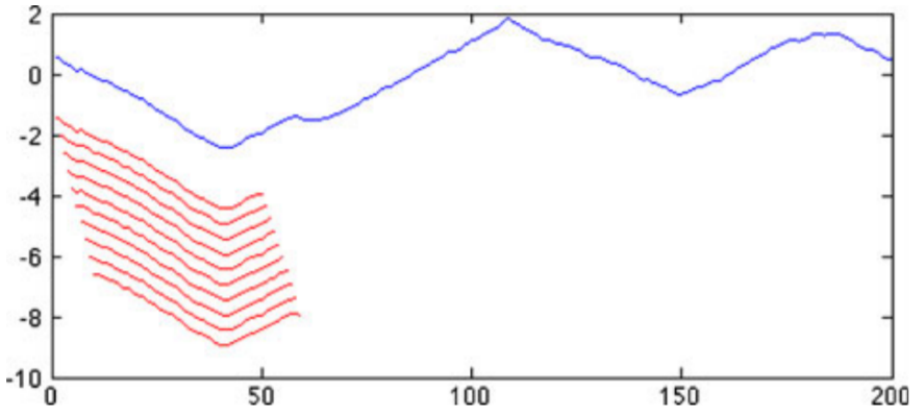

Figura 2.28: *First Subsequences extracted from a time series via sliding window (Lin et al., 2012)*

So, in short, the WEASEL+MUSE transforms a MTS into a vector of word counts, where each word represents a type of subsequence. This vector is highly discriminative, and can be used as input to train any typical machine learning classifier. Figure 2.29 illustrates this transformation process.

Finally, two important points to keep in mind regarding the actual implementation of WEASEL+MUSE:

- The length of the words w, the size of the alphabet c, the sizes of the windows used to extract the subsequences, and their step size are all hyperparameters whose optimal values can be found by using the techniques described in 2.2.2.
- A feature selection by means of logistic regression and chi-squared tests is done as a final step of the WEASEL+MUSE to reduce the redundancy and sparsity of the vector outputted by the model.

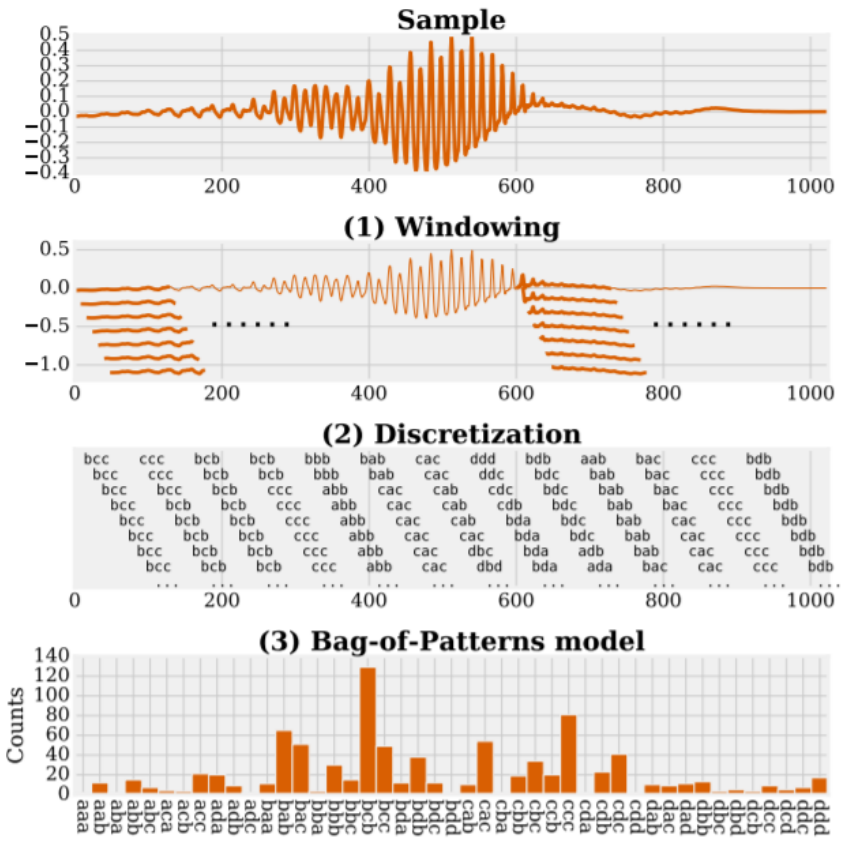

Figura 2.29: *Illustration of the WEASEL+MUSE transformation of a one-dimensional signal (Schäfer e Leser, 2017)*

2.4 Modelling

In the EEG Classification problem, the modelling step consists in training a model which takes EEG signals (or feature vectors derived from it) as input, and outputs the class of the corresponding trial. There are a myriad of possible models that can be used to this end, and in this Section we will review some of the most important of them.

2.4.1 k-Nearest Neighbours

k-Nearest Neighbour (k-NN) is one of the most basic and prominent algorithms in the field of Machine Learning and Data Mining. Despite being first proposed in the 60's, it is still a heavily used and researched method, due to its simplicity and effectiveness on a vast array of problems (Asim e Zakria, 2020).

Basically, k-NN works by assigning to an observation the most prevalent class of its $\mathbf{k}$ nearest neighbours, as illustrated in Figure 2.30. In other words, if $\{\mathbf{y_1}, \ldots, \mathbf{y_k}\}$ are the classes of the $\mathbf{k}$ nearest neighbours of observation $\mathbf{x}$, we have that the predicted class of $\mathbf{x}$ when using k-NN is simply $\mathbf{mode}(\{\mathbf{y_1}, \ldots, \mathbf{y_k}\})$.

When using k-NN for classification, there are some important points we should pay attention to:

- To establish how close observations are to each other - and thus find their nearest neighbors - a distance metric must be used. Normally this metric is the Euclidean Distance, but others, such as the Malahanobis Distance or DTW, can also be used, with the best metric depending on the nature of the problem.

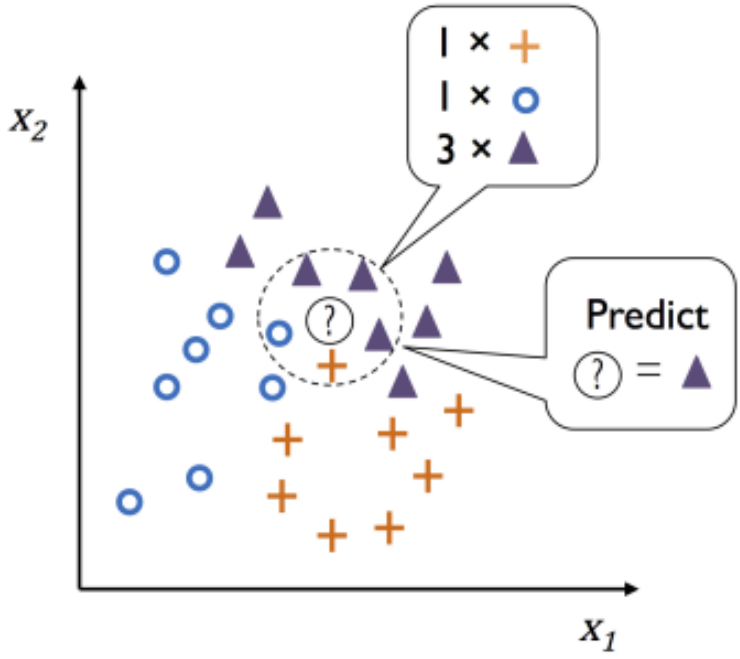

Figura 2.30: *Illustration of k-NN for a 3-class problem with k=5 (Raschka, 2018b).*

- The number of neighbors **k** is a hyperparameter whose optimal value can be found by using the techniques described in Section 2.2.2.
- The k-NN algorithm is particularly susceptible to the Curse of Dimensionality, which asserts that with increasing number of dimensions (that is, of explanatory variables being used), the larger is the volume needed to capture a fixed number of neighbors, in such a way that, as the number of dimensions grows, so does the volume, and the nearest neighbors to any given point become less and less similar to it as their distance increases (Raschka, 2018b).

  In this scenario, Dimensionality Reduction techniques can be used to help mitigate the problem, such as LASSO and PCA (Bishop, 2006).

2.4.2 Logistic Regression

The Logistic Regression is a commonly used model to predict the probability of any observation belonging to a certain class. In the binary case - that is, $\mathbf{y_i = 0}$ or $\mathbf{y_i = 1}$ - we model the probability that $\mathbf{y_i = 1}$ with the sigmoid function:

$$\mathbb{P}(y_i = 1|x_i) = \frac{\exp\left(\beta^T x_i\right)}{1 + \exp\left(\beta^T x_i\right)} \tag{2.18}$$

Where $\mathbf{x_i}$ is the feature vector of the observation we are trying to predict, and β are the parameters of the regressor, which will be estimated during training. The most common way to estimate β is by using the Maximum Likelihood Estimates (**MLE**), as follows:

1. First, by letting $\pi_i = \mathbb{P}\left(y_i = 1 \mid x_i\right)$, we can rearrange Equation 2.18 to show that the log odds of the positive class is a linear combination of the predictor variables $\mathbf{x_i}$:

$$\log\left(\frac{\pi_i}{1 - \pi_i}\right) = \beta^T x_i \tag{2.19}$$

2. Next, we define our likelihood function, $L(\beta)$. Given a dataset of samples $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}, i = 1, \ldots, n$, and taking all y_i as i.i.d Bernoulli random variables with p.d.f $f_i\left(y_i\right) =$

$\pi_i^{y_i} (1 - \pi_i)^{1-y_i}$, the likelihood is:

$$L(\beta) = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i} \tag{2.20}$$

Therefore, the log-likelihood $l(\beta)$ is:

$$\begin{aligned} l(\beta) &= \log_e \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i} \\ &= \sum_{i=1}^{n} [y_i \log_e \pi_i + (1 - y_i) \log_e (1 - \pi_i)] \\ &= \sum_{i=1}^{n} \left[y_i \log_e \left(\frac{\pi_i}{1 - \pi_i} \right) \right] + \sum_{i=1}^{n} \log_e (1 - \pi_i) \end{aligned} \tag{2.21}$$

Finally, substituting Equation 2.19 into 2.21, we have:

$$l(\beta) = \sum_{i=1}^{n} y_i \left(\beta^T x_i \right) - \sum_{i=1}^{n} \log_e \left[1 + \exp \left(\beta^T x_i \right) \right] \tag{2.22}$$

3. The MLE of β are the values of β which maximize $l(\beta)$. No closed-form solution exists for this problem, so numeric optimization procedures, such as the Newton–Raphson or Iteratively Reweighted Least Squares, must be used (Kutner *et al.*, 2005).

One important improvement that can be made to the Maximum Likelihood Estimates comes with the addition of a regularization term to the likelihood function, which helps reducing the variance of the estimates and colinearity effects (Hastie *et al.*, 2009).

More specifically, the L1 Regularized Logistic Regression adds the Lasso penalty to the log-likelihood:

$$l(\beta) = \sum_{i=1}^{n} y_i \left(\beta^T x_i \right) - \sum_{i=1}^{n} \log_e \left[1 + \exp \left(\beta^T x_i \right) \right] - \lambda \sum_{j=1}^{p} |\beta_j| \tag{2.23}$$

Where λ is the regularization strength, whose optimal value can be found by using the techniques described in Section 2.2.2. The MLE over this penalized likelihood can be calculated using the same numeric optimization procedures mentioned before.

2.4.3 Support Vector Machines

A Support Vector Machine (**SVM**) is a powerful and versatile Machine Learning model, capable of performing linear and nonlinear classification, regression, and even outlier detection, by constructing hyperplanes in transformed versions of the feature space that optimally separate the training instances (Géron, 2019).

To understand how a SVM works, it's easier to start with the simplest possible case, that of linearly separable binary classification, where a single hyperplane can perfectly separate the two classes, as illustrated in Figure 2.31.

Let our training dataset have L training points, where each input x_i has D attributes and is in one of two classes y_i = -1 or +1:

$$\{\mathbf{x}_i, y_i\} \quad \text{where} \quad i = 1 \ldots L, y_i \in \{-1, 1\}, \mathbf{x} \in \Re^D \tag{2.24}$$

The hyperplane which separates the data can be described by $w \cdot x + b = 0$, where w is a vector normal to the hyperplane and $\frac{b}{\|w\|}$ is the distance from the hyperplane to the origin.

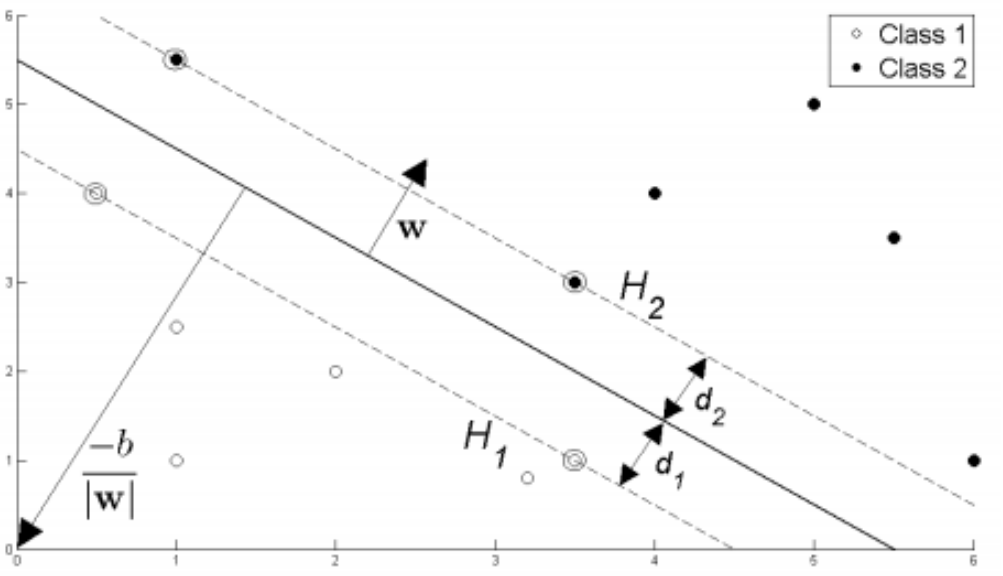

Figura 2.31: *Illustration of a SVM for a linearly separable binary case in two dimensions (Fletcher, 2008)*

Training an SVM consists in finding the hyperplane that optimally separate the two classes, which will be the one that is as far as possible from the closest members of both classes (shown inside circles in Figure 2.31). In mathematical terms, this means finding w and b such that:

$$\begin{aligned} \mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad & \text{for } y_i = +1 \\ \mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad & \text{for } y_i = -1 \end{aligned} \tag{2.25}$$

Or, more concisely:

$$y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) - 1 \geq 0 \; \forall_i \tag{2.26}$$

Not only that, but w and b must also satisfy the condition that d_1 and d_2 should be equal and as high as possible. With some linear algebra, it's easy to show that this is obtained for values of w that minimizes $\frac{1}{2}\|w\|^2$ (Fletcher, 2008). Therefore, our SVM model consists in the hyperplane obtained through the following optimization procedure:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{such that} \quad y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) - 1 \geq 0 \; \forall_i \tag{2.27}$$

This is an optimization problem with a convex quadratic objective and only linear constraints, which can be easily solved by any Quadratic Programming solver using Lagrange Multipliers. The reader is referred to Nocedal e Wright (2006) for an in-depth explanation of this optimization procedure.

Now, let's consider the non-linearly separable binary classification case, as illustrated in Figure 2.32. Here, no hyperplane can fully separate the data, and some instances of data will necessarily be misclassified.

To extend the SVM model to this new scenario, the constraints described in 2.25 can be altered to include a positive slack variable $\xi_i, i = 1, \ldots L$, which allows the classification of data points on the incorrect side of the hyperplane:

$$\begin{aligned} \mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \xi_i \quad & \text{for } y_i = +1 \\ \mathbf{x}_i \cdot \mathbf{w} + b \leq -1 + \xi_i \quad & \text{for } y_i = -1 \\ \xi_i \geq 0 \;\; \forall_i & \end{aligned} \tag{2.28}$$

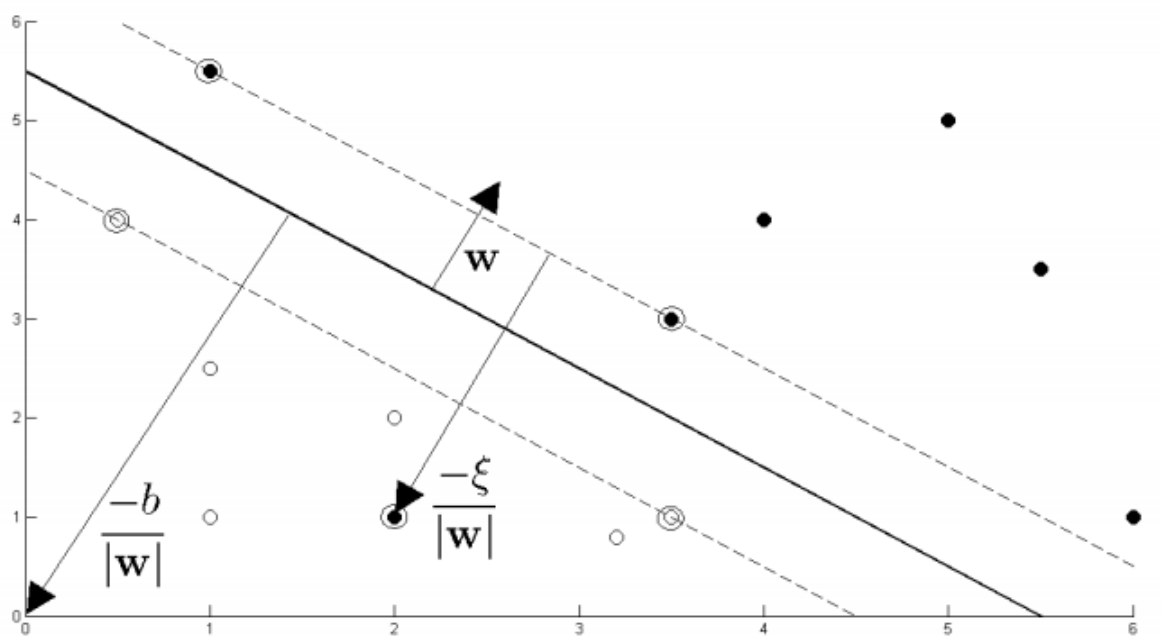

Figura 2.32: *Illustration of a SVM for a non-linearly separable binary case in two dimensions (Fletcher, 2008)*

Or, more concisely:

$$y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) - 1 + \xi_i \geq 0 \quad \text{where} \quad \xi_i \geq 0 \forall_i \tag{2.29}$$

With this new formulation, misclassified instances will have a penalty ξ that increases with the distance from the hyperplane. Since we are interested in reducing the number of misclassifications, our optimization problem can now be expressed as:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L}\xi_i \quad \text{s.t.} \quad y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) - 1 + \xi_i \geq 0 \quad \forall_i, \tag{2.30}$$

where the parameter C controls the trade-off between the slack variable penalty and the size of the classification margin, and whose optimal value can be found by using the techniques described in Section 2.2.2.

Again, this is an optimization problem with a convex quadratic objective and only linear constraints, which can be solved with the same techniques mentioned before. Once solved - that is, once the optimal values of w and b have been found - any new instance x' can be classified by simply evaluating $y' = \text{sign}\left(\mathbf{w} \cdot \mathbf{x}' + b\right)$.

Finally, in this non-linear scenario, there is another modification we can make to our optimization problem which can vastly improve the accuracy of the SVM model, the so called *Kernel Trick*. This modification allows us to take data which is heavily non-linear in the given dimensions of the problem and map it to a higher dimensional space where it is linearly separable, as illustrated in Figure 2.33.

This trick is based on the fact that, when solving the optimization problem in Equation 2.30, only the inner products of the feature vectors, $\mathbf{x}_i \cdot \mathbf{x}_j$, has to be calculated. This means that if the functions can be recast into a higher dimensionality space by some potentially non-linear feature mapping function $\mathbf{x} \longmapsto \phi(\mathbf{x})$, only inner products of the mapped inputs in the feature space need to be determined, without us needing to explicitly calculate ϕ (Fletcher, 2008). In other words, if the problem we are modeling is not linearly separable in the space of the given features, the *Kernel Trick* can map them to a higher dimension where they are potentially linearly separable, in a computationally efficient way. Details of how to implement this trick during optimization can be

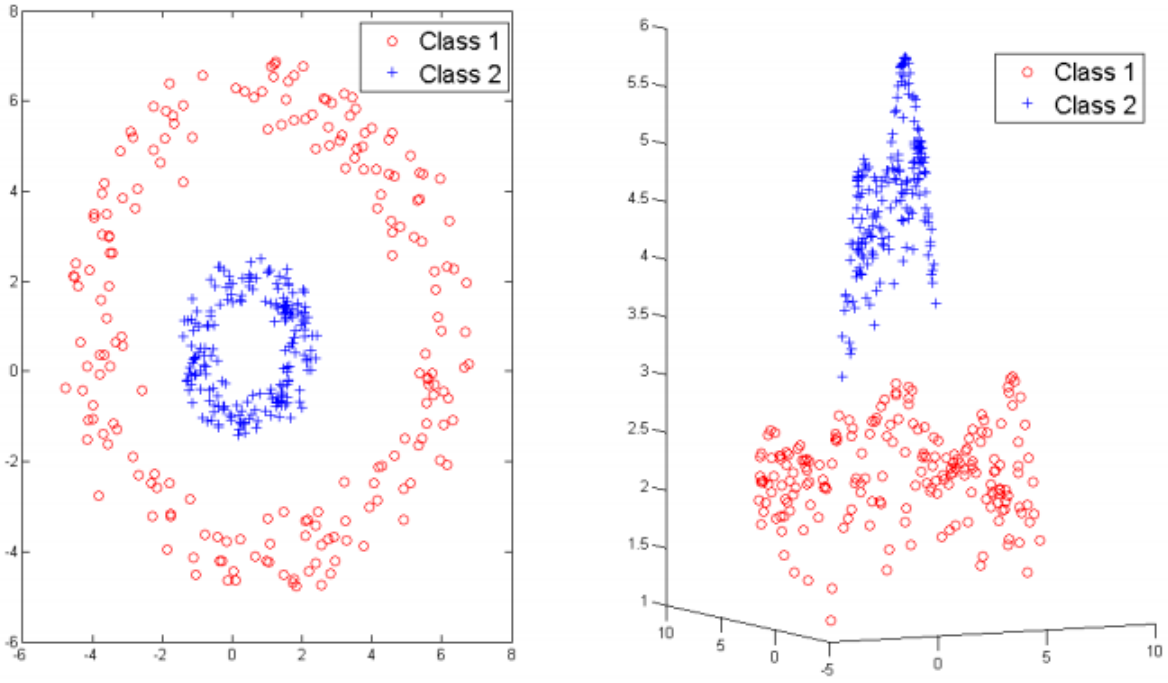

Figura 2.33: *Re-mapping of non-linearly separable data into higher dimensions using a Radial Basis Kernel (Fletcher, 2008)*

found on Hastie *et al.* (2009).

2.4.4 Random Forest

Initially proposed by L. Breiman in 2001, the Random Forest algorithm has been extremely successful in the machine learning community as a general-purpose classification and regression method. By combining multiple randomized decision trees and aggregating their predictions, this model is well suited for a vast array of problems, even those where the number of variables is much larger than the number of observations, a commonly seen scenario in EEG Classification problems (Biau e Scornet, 2016).

In order to understand how a Random Forest work, let's first consider a single Decision Tree, which can be seen as its building block. Basically, a Decision Tree is a non-parametric model that divides the $\mathbb{R}^d$ data space into K disjoint feature spaces $\{\mathcal{R}_1, \ldots, \mathcal{R}_K\}$, where each $\mathcal{R}_j \subset \mathbb{R}^d$, such that on each subspace $\mathcal{R}_j$ the same prediction is made for all $x \in \mathcal{R}_j$, as illustrated in Figure 2.34.

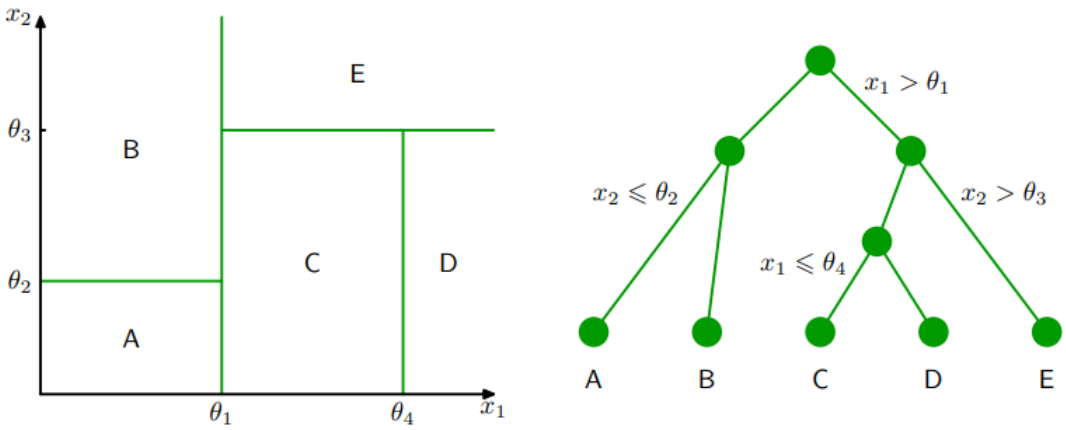

Figura 2.34: *Illustration of a Decision Tree in a two dimensional feature space (Schaar, 2017)*

Training a tree consists in finding, for each node of the tree, which feature and what cutoff value will split the data space in a way that will minimize the error of the prediction in the resulting subspaces. There are many different kinds of tree and training procedures to find this, with **CART** (*Classification And Regression Tree*) being one of the most commonly used today, specially in the context of Random Forests (Choi, 2017).

In CART, every node splits the data space on the feature that will result in the largest Information Gain IG, which is defined as follows:

$$IG\left(D_p, f\right) = I\left(D_p\right) - \sum_{j=1}^{m} \frac{N_i}{N_p} I\left(D_j\right), \tag{2.31}$$

where f is the feature to perform the split, D_p is the data on the parent node, D_j is the data on the j-th child node, I is the impurity measure, N_p is the total number of samples at the parent node, and N_j is the number of samples in the j-th child node. In CART, the impurity measure used is the Gini Impurity, defined as:

$$I_{Gini} = 1 - \sum_{i=1}^{j} p_i^2, \tag{2.32}$$

where j is the number of classes present in the node and p_i is the distribution of the i-th class in the node.

So in every node of the tree, the split will be done on the feature that results in nodes with the lowest impurity. The depth of the tree - that is, how many nodes in total will be created - is an hyperparameter whose optimal value can be found by using the techniques described in Section 2.2.2.

CART has some very attractive characteristics as a predictive model, such as naturally handling continuous, categorical, and missing data, robustness to outliers, and being easily interpretable. However, they are also very prone to overfit the training data, and generally have much higher variance than other methods.

The Random Forest method attempts to overcome these problems by making predictions based not on the outcome of one tree, but of many. To do this, it first creates bootstrap samples of the dataset, as illustrated in Figure 2.35. On each of these bootstraps it then trains a CART, with the additional condition that, at each node of each tree, only a subset of the total available features are used to evaluate the optimal split, which results in heavily uncorrelated trees, since they will be trained not only on different bootstrap samples, but also on different features.

The prediction of the Random Forest will then be the prediction of the majority of the trees - or the mean, in a regression context. And to understand why this prediction is better than that of a single tree, first we recall that, by the Bias-Variance Decomposition, the error of a model can be seen as the sum of a bias term and a variance term. Decision trees, if sufficiently deep, have very small biases, but huge variances. Now, let's take a Random Forest composed of n different trees, with X_i being the prediction of the i-th tree. If we consider all trees to have the same variance σ^2, and take the mean of their predictions as the prediction of the Random Forest, we have that its variance can be expressed as:

$$\begin{aligned}
\operatorname{Var}(\bar{X}) &= \operatorname{Var}\left(\frac{1}{n}\sum_i X_i\right) \\
&= \frac{1}{n^2}\sum_{i,j} \operatorname{Cov}\left(X_i, X_j\right) \\
&= \frac{n\sigma^2}{n^2} + \frac{n(n-1)\rho\sigma^2}{n^2} \\
&= \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2,
\end{aligned} \tag{2.33}$$

Figura 2.35: *Illustration of bootstrap sampling (Raschka, 2018a)*

where ρ is the correlation between the trees. Therefore, we have that, the bigger the number of trees n and the smaller the correlation ρ, the lower the variance of the prediction, which explains the advantage of using a Random Forest over a single Decision tree.

2.4.5 Gradient Boosted Trees

Gradient Boosted Trees is a method based on the idea that by combining weak classifiers - that is, classifiers which have, by themselves, high error rates, such as low depth Decision Trees - it's possible to create an ensemble that greatly outperforms any of its individual components (Hastie *et al.*, 2009).

The process of boosting is that of sequentially applying weak classification algorithms to repeatedly modified versions of the training data, thereby producing a sequence of weak classifiers $G_m(x), m = 1, 2, \ldots, M$. The final prediction of the boosted model is the ensemble of all weak classifiers through a weighted majority vote:

$$G(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right) \tag{2.34}$$

Where $\alpha_1, \ldots, \alpha_M$ are the weights computed by the boosting algorithm to give higher influence to more accurate classifiers in the sequence. This process is illustrated in Figure 2.36.

For boosted trees, the G_m classifiers are decision trees such as CART, described in Section 2.4.4. To train the full ensemble, we start by fitting the first tree on the training dataset. The second tree will then be fit in an adapted version of this dataset, one where the observations the first tree misclassified will be given a higher weight than those it classified correctly, in a way that will lead the second tree to perform better on the cases the first one struggled with. This goes on for all the M trees of the ensemble, and, by the end of the training process, those trees who achieved an overall better performance will have a greater weight α_m in the voting phase.

The training process to find the trees G_m and the weights α_m relies on an iterative optimization procedure based on the Gradient Descent technique, which is where the name Gradient Boosted Trees comes from. For an in-depth explanation of this training process, the reader is referred to Ke *et al.* (2017) and Chen e Guestrin (2016).

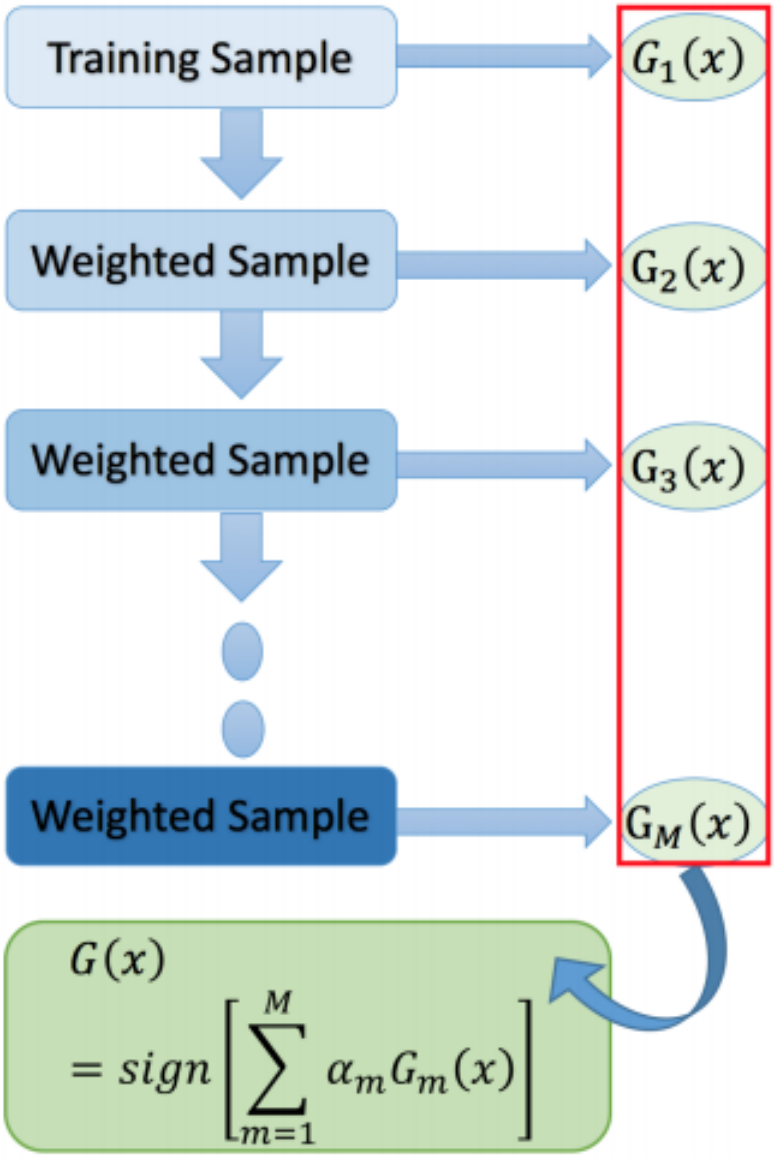

Figura 2.36: *Illustration of boosting procedure (Paul, 2016)*

2.4.6 Deep Learning

In recent years, Deep Learning models have gathered immense interest from the machine learning and artificial intelligence community by achieving unparalleled performance on a vast array of tasks on many different domains, such as Natural Language Processing, Image Recognition, Time Series Prediction, and many others (Emmert-Streib *et al.*, 2020).

Generally, Deep Learning describes a family of learning algorithms that have in common the usage of multi-layered Neural Networks to model data, with different neural architectures being used for different kinds of problems. So to understand Deep Learning, we first have to understand Neural Networks, starting with its most basic component, the Neuron, illustrated in Figure 2.37.

Initially inspired by the biological neurons of the brain, the Neuron, in a Neural Network context, is an algorithm for both regression and classification problems, which takes any feature vector $[x_1, \ldots, x_n]$ and outputs a prediction $\hat{y} = f\left(\sum w_i x_i + w_0\right)$, where f is an activation function that varies according to the type of problem at hand - in a binary classification case, for example, it can be a Sigmoid function, such that the output will be a probability (Géron, 2019).

Though simple to understand and easily applicable in many different scenarios, a single Neuron has very limited modeling power, being unable to capture more complex relations between the

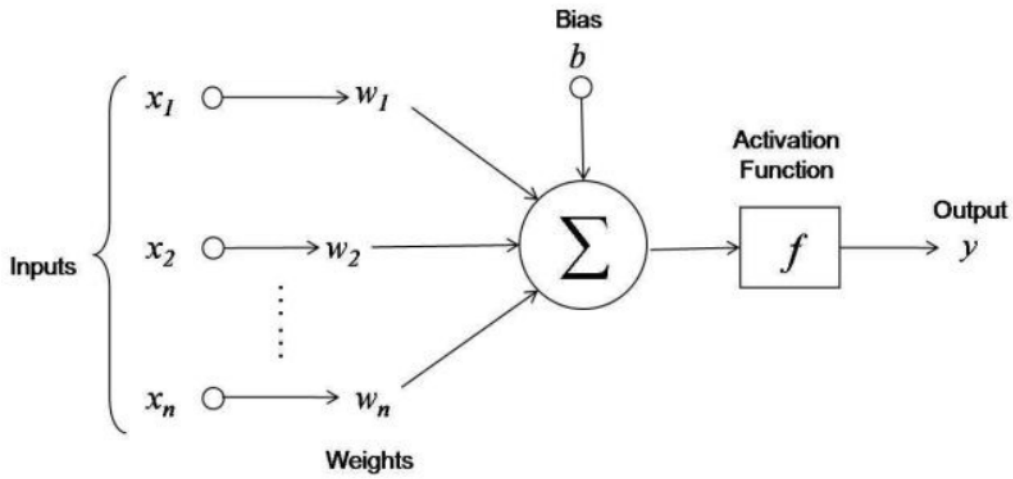

Figura 2.37: *Operations done by a Neuron*

features and the outputs. To overcome these limitations, Neural Networks, which consist of layers of interconnected neurons, were created. The idea behind them is similar to that of *boosting*, described in Section 2.4.5, where multiple simpler models are combined into a more complex one, with much higher predictive power.

The most common of Neural Networks is the Multilayer Perceptron (MLP), illustrated in Figure 2.38, where the neurons are arranged into a set of layers such that every unit in one layer is connected to every unit in the next layer. The first layer is the input layer, and its units take the values of the input features. The last layer is the output layer, and it has one unit for each value the network outputs (i.e. a single unit in the case of regression or binary classification, or K units in the case of K-class classification). Its activation function will vary according to the problem at hand - it can be a Sigmoid for a binary classification or a Linear function for a regression problem, for example. And finally, all the layers in between these two are known as hidden layers, where all neurons have the same activation function, generally a ReLu (Groose, 2020).

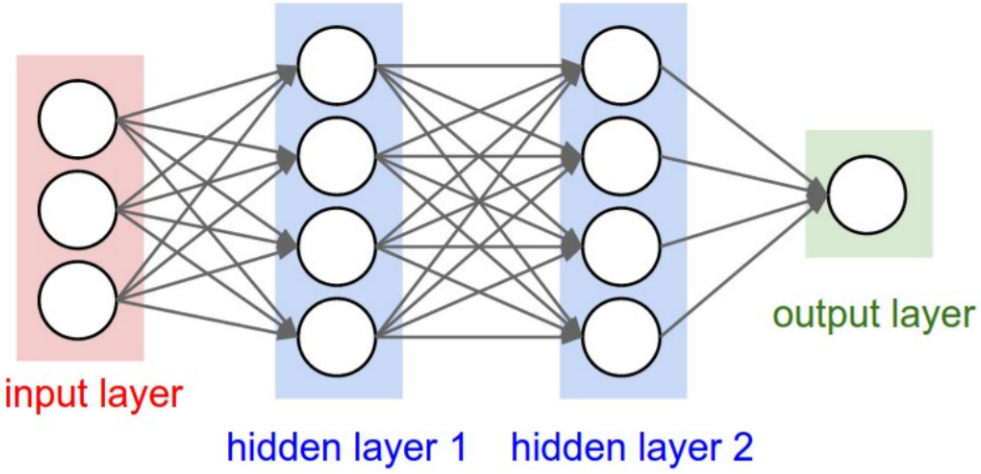

Figura 2.38: *A Multilayer Perceptron with 3 input units, 4 hidden units in the first and second hidden layer, and 1 output unit (Zemel, 2016)*

Let $\boldsymbol{\theta} = \left(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(L-1)}, \boldsymbol{\theta}^{(L)}\right)$ be the set of weights w of the MLP's neurons, where $\boldsymbol{\theta}^{(L)}$ are the weights between layers $L-1$ and layer L. Training the network consists in finding the

weights $\boldsymbol{\theta}$ that minimizes some loss function averaged over a training set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{n}$

$$\text{minimize}\,\Theta \quad \frac{1}{n}\sum_{t=1}^{n} \ell\left(f_\Theta\left(\mathbf{x}_t\right), y_t\right) \tag{2.35}$$

Where $f_\Theta\left(\mathbf{x}_t\right)$ is the output of the MLP, and the loss function l varies according to the problem, with MSE and Binary Cross-Entropy being the most usual choices for regression and binary classification problems, respectively. This is a hard optimization problem, highly non-convex and with many local minima, whose solution today is found through the Backpropagation and Gradient Descent techniques. The reader is referred to Nielsen (2015) for an in-depth explanation of this optimization procedure.

2.4.6.1 Convolutional Neural Networks

Convolutional Neural Network (CNN), which initially emerged from the study of the brain's visual cortex, is a network architecture that has achieved state-of-the-art performance on a vast array of problems, outperforming both MLP and other more traditional Machine Learning methods in many tasks related to Image Recognition, Natural Language Processing and Time Series Prediction (Emmert-Streib *et al.*, 2020).

Like MLPs, CNNs also have neurons as its most basic unit, but instead of fully connected layers, it uses convolutional layers, where each neuron is only connected to a patch of neurons in the next layer. To understand how CNNs can be used in a EEG Classification problem, that is, in a multivariate time series classification context, where the convolutions will be one dimensional - **Conv1d** - its easier to first understand how it is used on an image classification problem, where the convolutions are two dimensional - **Conv2d**.

Figure 2.39 illustrates a typical Conv2d. It has three different types of layers: Convolutional, Pooling, and Fully-connected, which is the same as the MLP.

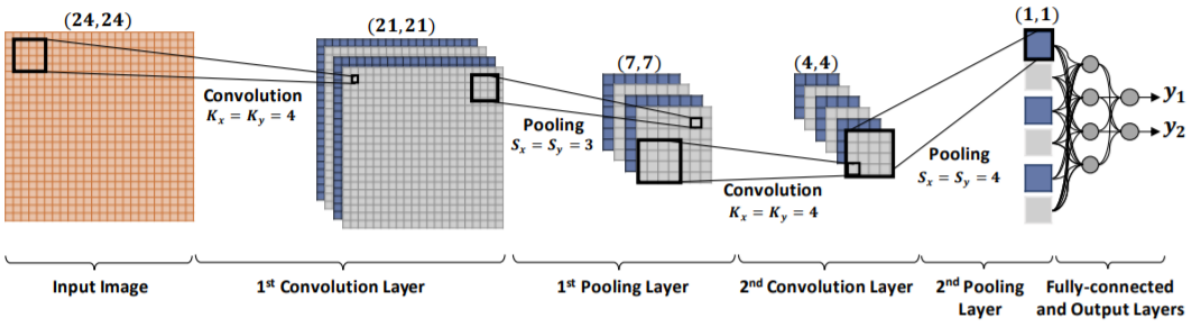

Figura 2.39: *Typical two dimensional CNN for image classification (Kiranyaz et al., 2019)*

The Convolutional layers are 2D arrangements of neurons, where the inputs to each neuron are just a subset of the outputs of the previous layer, as illustrated in Figure 2.40. More specifically, a neuron located in row i, column j of a given layer is connected to the outputs of the neurons in the previous layer located in rows i to $i + f_h - 1$, columns j to $j + f_w - 1$, where f_h and f_w are the height and width of the convolutional kernel. In order for a layer to have the same height and width as the previous layer, it is common to add zeros around the inputs, as shown in the Figure - this is called zero padding. And finally, note that a convolutional layer doesn't have to be a 1-to-1 mapping of the input image to a transformed, kernelized space - a single convolutional layer can have multiple kernels, mapping the input to multiple different spaces. We call the number of kernels a convolutional layer has the *depth* of the layer. In Figure 2.39, for instance, the first convolutional layer has a depth of 4, and the second one a depth of 6 (Géron, 2019).

The Pooling layer, similarly to the Convolutional one, connects each neuron to a limited subset of the outputs of the previous layer, but here there are no weights between the layers. Its goal is simply to subsample (i.e., shrink) the input image in order to reduce the computational load

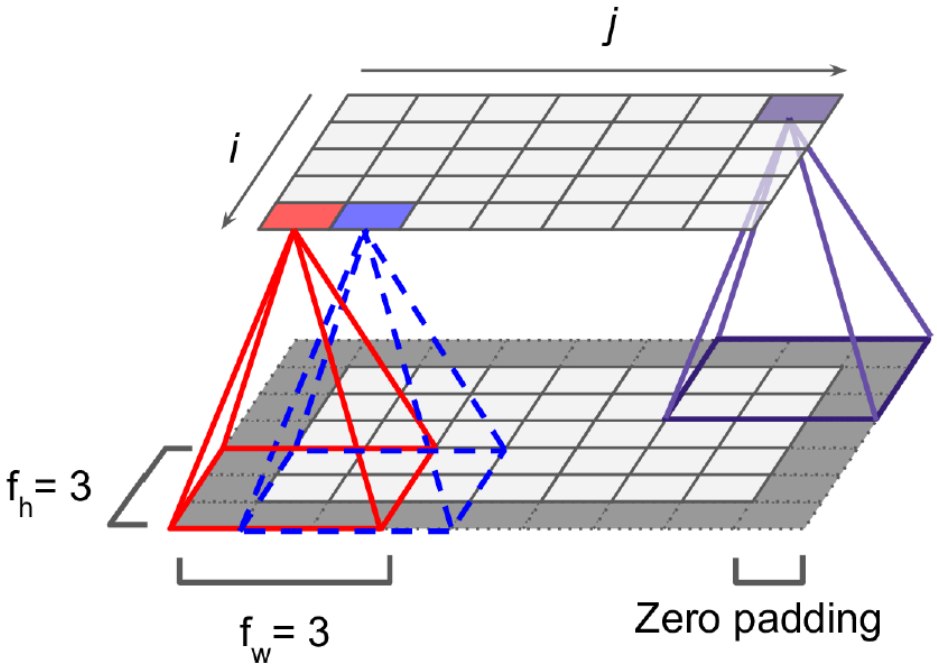

Figura 2.40: *Convolutional layer of a Conv2d (Géron, 2019)*

and the number of parameters of the model, thereby reducing the risk of overfitting. All it does is aggregate the inputs using an aggregation function such as the max or mean. Figure 2.41 shows a max pooling layer, which is the most common type of pooling layer, with a 2x2 pooling kernel - notice how the output image is half the size of the input.

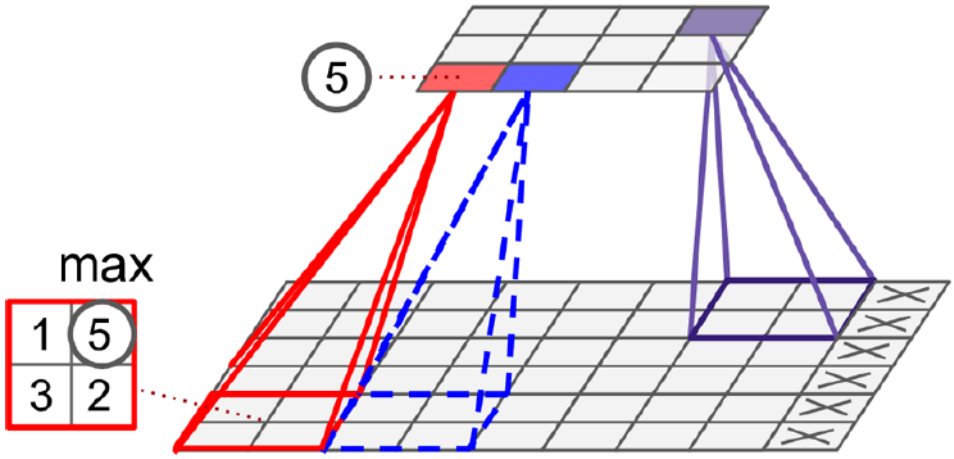

Figura 2.41: *Pooling layer of a Conv2d (Géron, 2019)*

And finally, the Fully-connected layer is the MLP described in Section 2.4.6. It flattens the output of the previous Convolutional or Pooling layer, passes it through the fully connected neurons of the hidden layers, and then through the output neurons, giving the final prediction of the CNN.

So how to use CNNs in the EEG Classification problem? Note that the only difference between a convolution over an image and over a time series is the number of dimensions: on an image, the convolution happens in two dimensions, *height* and *width*, while on a time series, even a multivariate one, the convolution happens only in one, *time*. What this means is that by simply limiting the number of dimensions in the Convolution and Pooling layers to one, we can use a CNN in a time series classification task, the so called Conv1d, as illustrated in Figure 2.42.

The same concepts discussed in the Conv2d case apply to the Conv1d: the time series will

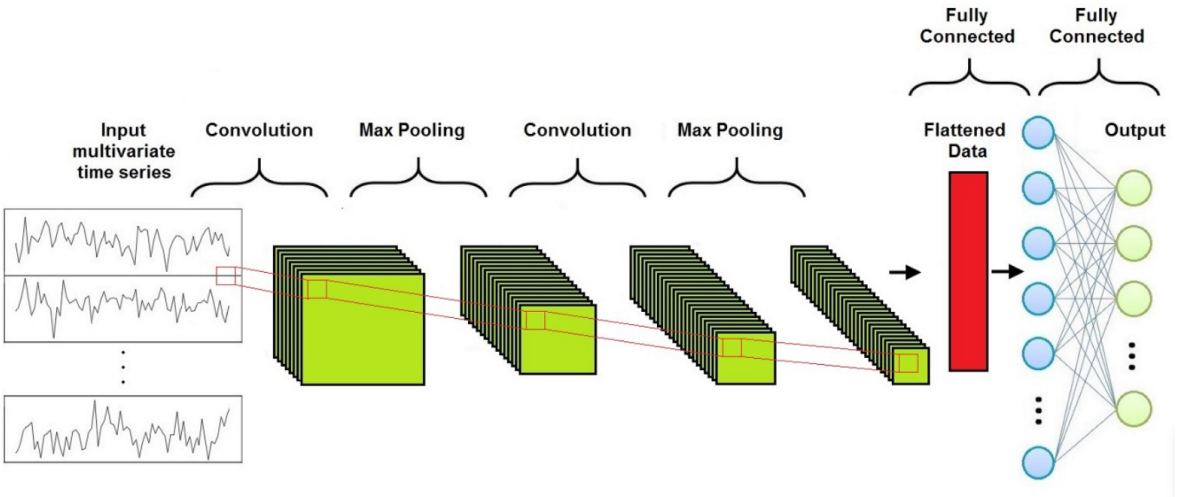

Figura 2.42: *Typical Conv1d for time series classification problems (Del Pra, 2020)*

pass through Convolutional layers, where a convolutional kernel will slide over the time dimension connecting segments of the series to neurons in the next layer; though Pooling layers, where a downsample of the time series is performed; and finally through an MLP, where the resulting features are flattened into a vector that is fed to the neurons in the hidden layers, and then to the output neurons, which will give the final prediction of the network.

2.4.6.2 InceptionTime

Up until 2014, the CNN architecture described in Section 2.4.6.1 was the most used in image recognition tasks due to its unparalleled, state-of-the-art performance. Back them, improvements over this architecture consisted mainly in adding more and more layers to the network, which worked well till the point these larger networks' tendency to overfit, as well as their diminishing returns on performance against computational load required to train them, started to get too severe.

To solve these problems, researchers at Google proposed the **Inception** network in 2014, which was constantly upgraded in the next few years, to the point where it stands today as one of the go-to architectures to solve image related tasks, achieving performances greater than that of a standard CNN while still being comparatively easier to train (Szegedy *et al.*, 2016).

In 2019, Fawaz *et al.* (2019) proposed **InceptionTime**, a neural network designed specifically to time series classification tasks that incorporated the improvements of the Inception architecture. To date, this network has been shown to be the best deep learning approach to the classification of time series, univariate or multivariate, due to its state-of-the-art performance and reachable training times.

To understand how InceptionTime works, first we need to understand a traditional Inception network, as illustrated in Figure 2.43. There are two major differences in relation to a standard CNN. First, the Convolutional and Pooling layers are substituted by Inception Modules. Second, there is the inclusion of residual connections.

One of the problems of standard CNNs is that the choice of where to put Convolutional and where to put Pooling layers, as well as the size of the kernels, are hyperparameters that have to be manually set, which can result either in the wrong values being chosen, or in very deep networks that have many different combinations of kernels/layers in an attempt to approach this optimal configuration. Inception Modules, illustrated in Figure 2.44, overcomes this problem by combining both Convolutional and Pooling layers in a single structure, in such a way that the optimal configuration will arise by itself during the training of the model. More specifically, each Inception Module is composed of four layers, as follows (Del Pra, 2020):

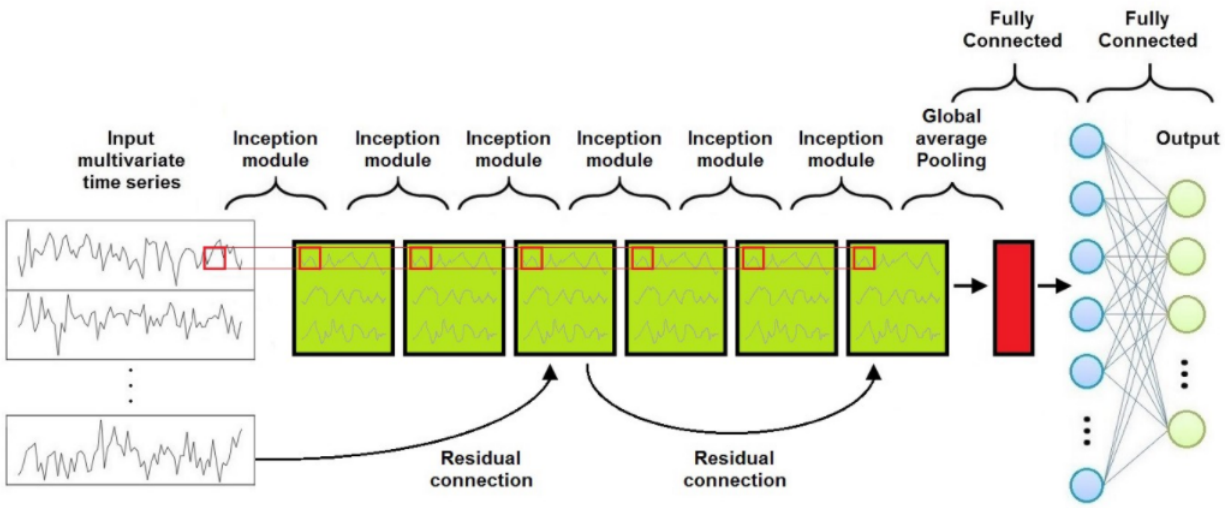

Figura 2.43: *Inception Network for time series classification (Del Pra, 2020)*

- The first layer is a bottleneck layer, that reduces the dimensionality of the inputs. This reduces also the computational cost and the number of parameters, speeding up training and improving generalization.
- The second major component of the Inception Module is a set of parallel Convolutional Layers of different size acting on the same input feature map. For example in the Figure there are three different convolutions with filter size 10, 20, and 40.
- The third layer is a MaxPooling, that introduces the ability of having a model that is invariant to small perturbations.
- The fourth and last layer is a Depth Concatenation Layer, where the output of each independent parallel convolution and of the MaxPooling is concatenated to form the output multivariate time series of the current Inception Module.

By stacking multiple Inception Modules, the network is able to extract latent hierarchical features of multiple resolutions thanks to the use of convolutions with different sizes, with the kernels that provide the most information to the task at hand receiving bigger weights automatically during the training of the model, eliminating the necessity of hand picking kernel sizes and layers' ordering.

The residual connections, on the other hand, are simple modifications that allow the output of a layer to not only serve as the input for the next layer, but also to feed into the input of subsequent ones. This trick makes it easier, from an optimization point of view, to train deeper networks. The reader is referred to He *et al.* (2016) for an in-depth explanation of this procedure.

Finally, the idea behind InceptionTime is that a single Inception network usually exhibits high standard deviation in its accuracy, such that it might perform very well on one kind problem, but very poorly on another. To solve this, InceptionTime is actually an ensemble of 5 different Inception networks, each with their own weights, trained in parallel over the given dataset. Since the optimization process to find the optimal neuronal weights is stochastic, each network will have different weights, even if being trained on the same data. The output of the InceptionTime is then an average of the response of each individual network, resulting in a prediction with lower variance and higher predictive power (Fawaz *et al.*, 2019).

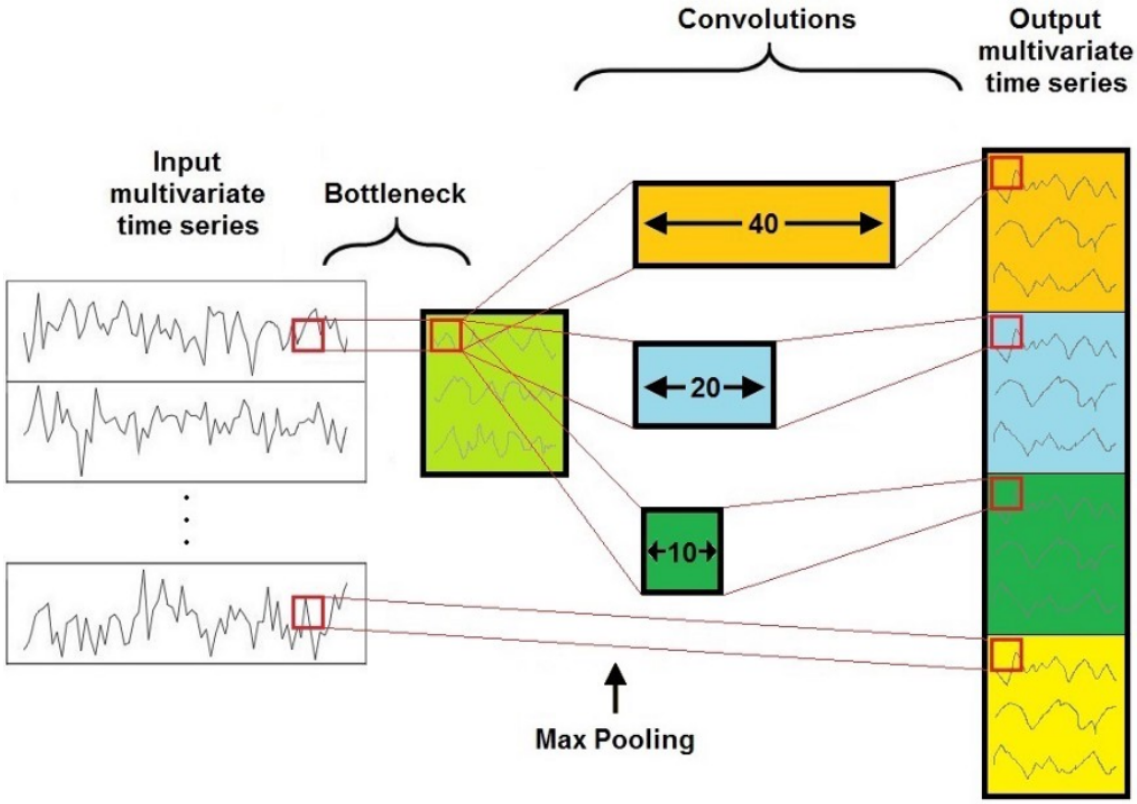

Figura 2.44: *Inception module (Del Pra, 2020)*

Capítulo 3

Methodology and Results

To quantitatively assess the state of current EEG Classification methods and identify a strong, task-agnostic baseline, our study focuses on 11 different methods, evaluated over four different EEG datasets.

As discussed in Section 1.2, these methods were gathered from surveying the literature for models that achieved state-of-the-art performance on EEG Classification problems, and that were backed by peer-reviewed publications and reproducible results. We focused our research on task-agnostic models - that is, models that consistently give good results across many different EEG Classification problems with minimum adaptation. With these criteria, we narrowed our analysis to 11 different methods that we believe create an accurate picture of the most successful techniques used in the field today, as supported by the works of Roy *et al.* (2019), Lotte *et al.* (2018), Hastie *et al.* (2009), Houssein *et al.* (2017), Sun e Zhou (2014), among many others.

These 11 methods were evaluated over four EEG datasets, related to problems of Emotion Recognition, Disease Detection, and Brain-Computer Interfacing (Motor-imagery and ERP-based). All these datasets are publicly available, and the code developed for this study can be found online[1].

In Section 3.1 we describe the datasets, in 3.2 we define the 11 methods in more detail, and in 3.3 we present the methodology behind our analysis.

3.1 Datasets

The 4 datasets used in this study are as follow:

- **DEAP**

  A multimodal dataset for the analysis of human affective states. In this dataset, EEG signals of 32 participants were recorded as each watched 40 one-minute long excerpts of music videos. Participants rated each video in terms of the levels of arousal, valence, like/dislike, dominance and familiarity.

  The participants were 50% male, 50% female, aged between 19 and 37 (mean age 26.9) The experiments started with a 2 minute baseline recording, during which a fixation cross was displayed to the participants (who were asked to relax during this period). Then the 40 videos were presented in 40 trials, each consisting of the following steps:

  1. A 2 second screen displaying the current trial number to inform the participants of their progress.
  2. A 5 second baseline recording (fixation cross).
  3. The 1 minute display of the music video.
  4. Self-assessment for arousal, valence, liking and dominance, with a score ranging from 1 to 9.

[1] https://github.com/lenongsa/eeg_survey

The EEG signals were recorded with a Biosemi ActiveTwo system at a sampling rate of 512 Hz, using 32 active AgCl electrodes (placed according to the international 10-20 system). In our study, we used a pre-processed version of this data, downsampled to 128 Hz, bandpass frequency filtered from 4 to 45 Hz, and with EOG artifacts removed (Koelstra *et al.*, 2011).

The task we will attempt is to predict whether an individual rated a video with high (score 5-9) or low (score 1-4) valence - that is, if the video evoked positive or negative sentiments on him.

- **EEG Alcoholism**

  This data arises from a large study to examine EEG correlates of genetic predisposition to alcoholism. It consists of two groups of subjects: alcoholic and control. Each subject was exposed for 1 second to either a single stimulus (S1) or to two stimuli (S1 and S2) which were pictures of objects chosen from the 1980 Snodgrass and Vanderwart picture set (Snodgrass e Vanderwart, 1980). When two stimuli were shown, they were presented in either a matched condition where S1 was identical to S2 or in a non-matched condition where S1 differed from S2. The idea behind this experiment is that the neural activity of the subject, when exposed to these stimuli, correlates to he being or not an alcoholic, as illustrated in Figure 3.1.

  This dataset contains 120 trials for 121 subjects, recorded with 64 electrodes at 256 Hz. The data collection process is described in detail in Zhang *et al.* (1995). In our study, we use 10 trials per subject, and our task is to predict whether an individual is an alcoholic or not based on his EEG signals when exposed to a S1 stimulus.

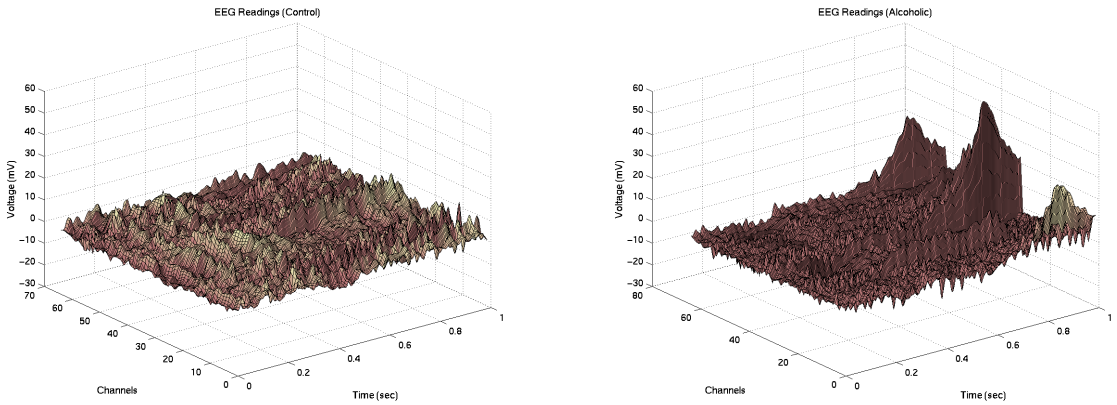

Figura 3.1: *EEG signals of control and alcoholic subjects when exposed to S1 stimulus*

- **FingerMovements**

  This dataset was recorded from a normal subject during a no-feedback session. The subject sat in a normal chair, relaxed arms resting on the table, fingers in the standard typing position at the computer keyboard. The task was to press with the index and little fingers the corresponding keys in a self-chosen order and time, i.e. using self-paced key typing. The experiment consisted of 3 sessions of 6 minutes each. All sessions were conducted on the same day with some minutes break in between. Typing was done at an average speed of 1 key per second.

  There is a total of 416 cases. Each case is a recording of 28 EEG channels of 500 ms length each ending 130 ms before a key-press. Signals were recorded at 1000 Hz with a band-pass filter between 0.05 and 200 Hz, and posteriorly downsampled to 100 Hz, so that each channel consists of 50 observations (Blankertz *et al.*, 2002).

  The task here is to predict whether the subject will move his right hand or his left one.

- **SelfRegulationSCP1**

In this dataset, a healthy subject was asked to move a cursor up and down on a computer screen, while his cortical potentials were taken. During the recording, the subject received visual feedback of his slow cortical potentials (Cz-Mastoids). Cortical positivity leads to a downward movement of the cursor on the screen. Cortical negativity leads to an upward movement of the cursor.

There is a total of 561 trials, each lasting 6 seconds. During every trial, the task was visually presented by a highlighted goal at either the top or bottom of the screen to indicate negativity or positivity from second 0.5 until the end of the trial. The visual feedback was presented from second 2 to second 5.5. Only this 3.5 second interval of every trial is provided.

The recordings were made with 6 EEG channels at 256 Hz, which, at a recording length of 3.5 seconds, results in 896 samples per channel for every trial.

The task here is to predict whether the subject is increasing or decreasing his slow cortical potentials - or, in other words, if the subject wanted to move the cursor up or down.

In short, these datasets relate to different kinds of EEG Classification problems, and reflect the varying degrees of data complexity and variability typical in the field today, as summarized in Table 3.1.

Dataset	Trials	Channels	Time points	Positive Class Ratio
DEAP	1260	33	8064	0.5
EEG Alcoholism	1210	64	256	0.5
FingerMovements	416	28	50	0.5
SelfRegulationSCP1	561	6	896	0.5

Tabela 3.1: *Datasets Summary*

3.2 Methods

The 11 methods analyzed in this study can be divided into three categories, as follow:

- **Univariate Featurization Methods**

  These methods rely on creating independent sets of features from each of the EEG Channels used in the experiment and concatenating them into a single vector, which is then fed to a traditional Machine Learning classifier.

  More specifically, let $\boldsymbol{X_i^j}$ be the EEG signal obtained from the i-th channel of trial j, and $\boldsymbol{FT_i^j}$ be the features obtained from Fourier Transforms (see Section 2.3.1), $\boldsymbol{WT_i^j}$ the features obtained from Wavelet Transforms (see Section 2.3.2), and $\boldsymbol{FD_i^j}$ the features obtained from Hjorth Parameters (see Section 2.3.3) when applied to this $\boldsymbol{X_i^j}$ signal. With n being the total number of EEG channels, we can concatenate this features into a single vector as follows:

$$FT^j = [FT_1^j, FT_2^j, ..., FT_n^j]$$
$$WT^j = [WT_1^j, WT_2^j, ..., WT_n^j]$$
$$FD^j = [FD_1^j, FD_2^j, ..., FD_n^j]$$

$$\mathbf{FS^j} = [\mathbf{FT^j}, \mathbf{WT^j}, \mathbf{FD^j}]$$

Where $\mathbf{FS^j}$ (named for Feature Set) is a single vector of features for each j-th trial. Finally, letting $\mathbf{f}$ be a trained classification model, all trials can be classified with $\mathbf{f}(\mathbf{FS^j})$, as illustrated in Figure 3.2.

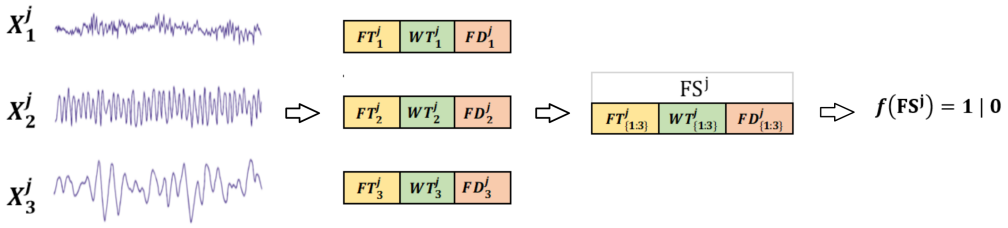

Figura 3.2: *Univariate Featurization Methods*

In total, we will study four different classifier models for this approach: Logistic Regression with L2 Regularization, Support Vector Machine, Random Forest, and Gradient Boosted Trees. This results in four Univariate Featurization Methods to be analyzed:

1. **FS + Logistic Regression with L2 Regularization**
2. **FS + Support Vector Machine**
3. **FS + Random Forest**
4. **FS + Gradient Boosted Trees**

- **Multivariate Transformation Methods**

  These methods frame the EEG data not as a set of univariate timeseries, but as a single multivariate entity, which they transform into a discriminative shape that can be fed to a traditional Machine Learning Classifier.

  We will analyze two transformation methods: Dynamic Time Warping, a distance-based method (**DTW**, see Section 2.3.4) and WEASEL+MUSE, a feature-based one (**WM**, see Section 2.3.5).

  First, for **DTW**, if k is the total number of labelled trials in the dataset, we create a $n \times k$ similarity matrix for each unlabelled trial where the element x_{ij} is the Dynamic Time Warping distance of the i-th channel of the unlabelled trial to the corresponding channel of the j-th labelled trial. Then, the classification of all channels of the unlabelled trial can be done using a 1-Nearest Neighbour model (**1-NN**, see Section 2.4.1), by giving it the same class as the labelled channel it is closest to. Finally, the unlabelled trial can be given the same label as that of the majority of its channels. This procedure is the same as that detailed in Section 2.3.4.

  Next, for **WM**, we create a $\mathbf{FS^j}$ feature vector for every j-th trial in the dataset, similar to that of the Univariate Featurization Methods. The difference is that here this vector isn't made by concatenating features from each channel, but is created at once with data from all the channels with the WEASEL+MUSE approach - this procedure is detailed in 2.3.5. Finally, letting $\mathbf{f}$ be a trained classification model, all trials can be classified with $\mathbf{f}(\mathbf{FS^j})$.

  Since we are going to study the same classifier models as in the Univariate Featurization Methods, we have a total of five different Multivariate Transformation Methods to analyze:

5. **DTW + 1NN**
6. **WM + Logistic Regression with L2 Regularization**
7. **WM + Support Vector Machine**
8. **WM + Random Forest**
9. **WM + Gradient Boosted Trees**

- **Deep Learning Methods**

  In this survey, we will analyze two different Deep Learning architectures. The first is the Convolutional Neural Network architecture (**CNN**, see Section 2.4.6.1), which is the most prevalent one in EEG Classification papers today (Craik *et al.*, 2019). The second one is the **InceptionTime** architecture (see Section 2.4.6.2), that is based around the Inception neural modules recently proposed by Google, and that outperforms many of the state-of-the-art models in a great number of different time series classification tasks (Fawaz *et al.*, 2019).

  Both architectures take the raw EEG signals as input - no feature engineering or data transformation are necessary, since these are made internally by the network. Just as traditional Machine Learning Classifiers, these Deep Learning models, once trained, will take the EEG signals of any unlabelled trial and classify it accordingly:

  10. **CNN**
  11. **InceptionTime**

3.3 Quantitative Analysis

To evaluate the performance of these 11 methods over the 4 datasets, we employ the following procedure:

1. For each dataset, create 10 stratified train/test folds (see Section 2.2.1) with 10% of the data in the test set, and 90% in the train set.
2. For each of these splits, train the 11 methods over the train set and evaluate them over the test set. The performance metric we will use is Accuracy, chosen for its ease of understanding and due to the fact that our positive and negative classes are fairly balanced across all datasets.
3. During the training phase, use 3-fold grid search cross validation to optimize the models' hyperparameters (see Section 2.2.2).
4. Once we have calculated the accuracy 10 times for each method, for each dataset, create a Critical Difference Diagram to rank them according to their performance (see Section 2.2.3).

With this, we have a statistically safe methodology to determine, among the methods analyzed, which ones have a significant higher performance over the datasets used, and could then be utilized as a strong baseline in EEG Classification problems.

3.4 Results

A cloud machine with 82 CPUs, 200 GB RAM and a NVIDIA Tesla P100 GPU with 16 GB memory was used to run the procedure described in Section 3.3, with the Python code developed for the calculations available at https://github.com/lenongsa/eeg_survey. The results obtained are shown in Tables 3.2 and 3.3, with their Critical Difference Diagram presented in Figure 3.3.

From these results, we can gather that:

1. The **Univariate Featurization Methods** outperform all other methods on all datasets. However, the Critical Difference Diagram shows there is no statistically significant difference in performance based on which classifier model is chosen. In other words, this tell us that feature vectors made out of Fourier Transforms, Wavelet Transforms, and Hjorth Parameters on each individual EEG channel are more effective in discriminating classes in an EEG Classification Problem than features created out of multivariate transformation methods like WEASEL+MUSE and DTW, or features automatically created out of Deep Learning methods.

Method	Dataset			
	EEG Alcoholism	**SelfRegulationSCP1**	**DEAP**	**FingerMovements**
FS + LR	0.90	0.85	0.65	0.55
FS + SVM	0.87	**0.86**	0.67	**0.59**
FS + RF	**0.91**	0.83	**0.70**	0.58
FS + GBT	0.89	0.85	0.69	0.55
InceptionTime	0.84	0.84	0.68	0.56
CNN	0.73	0.84	0.61	0.52
WM + LR	0.73	0.82	0.62	0.53
WM + SVM	0.70	0.82	0.65	0.52
WM + RF	0.70	0.83	0.64	0.54
WM + GBT	0.68	0.80	0.61	0.52
DTW + 1NN	0.68	0.83	0.58	0.51

Tabela 3.2: *Average 10-fold accuracy of the methods on the studied datasets*

Method	Dataset			
	EEG Alcoholism	**SelfRegulationSCP1**	**DEAP**	**FingerMovements**
FS + LR	0.05	0.04	0.05	0.06
FS + SVM	0.06	0.03	0.03	0.07
FS + RF	0.03	0.04	0.04	0.07
FS + GBT	0.05	0.05	0.02	0.05
InceptionTime	0.03	0.04	0.04	0.06
CNN	0.06	0.04	0.04	0.07
WM + LR	0.02	0.06	0.05	0.07
WM + SVM	0.06	0.06	0.03	0.08
WM + RF	0.03	0.04	0.03	0.06
WM + GBT	0.04	0.07	0.06	0.07
DTW + 1NN	0.05	0.06	0.07	0.07

Tabela 3.3: *Standard deviation of the accuracy of the methods over 10 folds on the studied datasets*

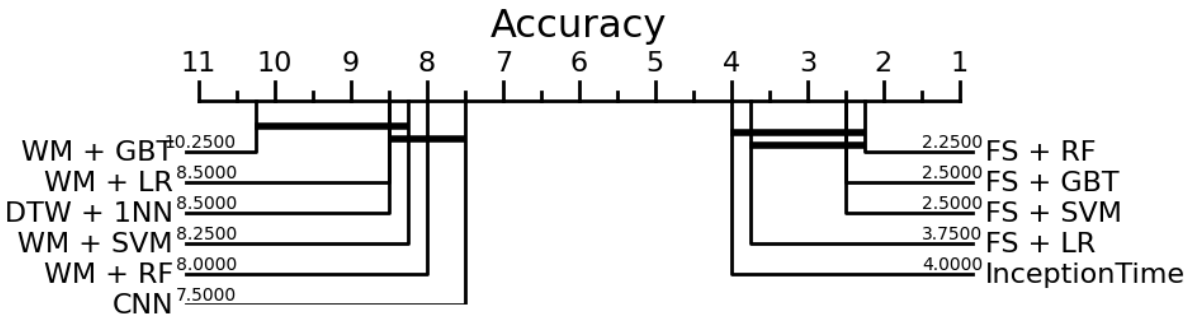

Figura 3.3: *Critical Difference Diagram for the 10 methods evaluated*

But once this univariate feature vector is created, the choice of which classifier model to use with them is not as important, since our experiments show no significant difference between them in regards to classification accuracy.

2. One of the Deep Learning methods, **InceptionTime**, has shown promising results, ranking on average at 4th place in classification accuracy. Statistically, the Critical Difference Diagram shows that, despite still being worse than the set of univariate featurization methods studied, its performance is not significantly different than that of any singular method of the set. This is in agreement with the results of Lotte *et al.* (2018), which claimed that Deep Learning

methods are still not as accurate as more traditional classifiers in EEG Classification Problems, though this gap in performance has been steadily diminishing.

3. **Multivariate Transformation Methods**, though much simpler to implement, have shown significantly worse performance than all other methods. **1D-CNN**, the simplest Deep Learning architecture for time series classification, has also shown worse performance than the alternatives.

4. Also noteworthy to mention is that, when analyzing the performance of the methods over each individual - that is, not their average accuracy on all the folds, but how they classified or misclassified each individual point in the datasets - we see that, in general, the methods that perform better correctly classify the same individuals the methods that performed worse did, while also being able to correctly classify other individuals the worse methods couldn't. In other words, what we have is not a scenario where each method has a specific group of individuals it performs better upon, but one where all methods can correctly classify the easier cases on the datasets, but only the better ones can do so on the harder individuals.

   This idea is further reinforced by the fact that initial tests with a model which is an ensemble of the 11 methods studied did not perform any better than the **Univariate Featurization Methods** methods, which suggests that the worst performing models aren't capturing any characteristic of the individuals the better ones aren't already.

How to explain the superior performance of the more traditional Univariate Featurization models? The explanation most likely has to do with two factors:

- **Frequency-based features are more discriminant than time-based ones.**

  One of the major differences between the Univariate Featurization Methods and the alternatives is that they explicitly convert the EEG data from the time domain to the frequency domain during the feature engineering step, while the Multivariate Transformation and Deep Learning methods act only upon time domain data.

  As discussed in Iscan *et al.* (2011), Harpale e Bairagi (2016) and Herrmann *et al.* (2014), among many others, the frequency components that make up an EEG signal hold valuable information when it comes to classifying them, information that is not readily available on the time domain representation of the signal. Therefore, classification methods that use features created out of the frequency domain might have a performance edge over models trained solely on time domain variables.

- **Low complexity models fit the small datasets better.**

  In a machine learning context, model complexity refers to the number of parameters a model adjusts during its training phase. As discussed in Chollet (2017) and Brigato e Iocchi (2020), it's well known that, since Deep Learning models are commonly of very high complexity, with often thousand of adjustable parameters in the form of weights between different neurons, a lot of data is required to train them without overfitting.

  Since the datasets we are using here are very small - as is usual with EEG studies - this might help explain the lower performances of InceptionTime when compared to the much simpler (in terms of Model Complexity) Univariate Featurization methods.

Capítulo 4

Conclusions

The EEG Classification problem, where one tries to identify events, stimuli, and intents through analysis of EEG signals, has been gathering increasing attention from the scientific community with the recent advances in EEG data availability, and Machine Learning and Big Data technology. However, a vast number of papers recently published in this field fail to adhere to critical statistical guidelines that would ensure the validity of their results.

In this dissertation, we contribute to the field by, first, providing a review of some of the most prominent methods for EEG Classification and the statistical and data science guidelines necessary for their correct implementation, and, second, by making a quantitative analysis of these methods over four different EEG datasets, in order to find a strong, task-agnostic baseline against which new models for EEG Classification should be compared.

In Chapter 2 we give a review of the phases of an EEG Classification problem, starting with a description of Electroencephalography itself (2.1), then discussing the methodological guidelines that should be followed when attempting to solve the classification problem: Cross Validation for better estimation of the model's generalization capabilities (2.2.1), Hyperparameters Optimization for achieving the model's best performance (2.2.2), and Critical Difference Diagrams for results comparison over multiple datasets (2.2.3). Then, we presented some of the most effective featurization methods to transform the EEG signals into discriminative feature vectors: Fourier Transformations (2.3.1), Wavelet Transformations (2.3.2), Hjorth Parameters (2.3.3), Dynamic Time Warping (2.3.4), and the newly proposed WEALSEL+MUSE (2.3.5). And finally, we reviewed some of the most used classification models in the literature today: K-Nearest Neighbours (2.4.1), Logistic Regression (2.4.2), Support Vector Machines (2.4.3), Random Forest (2.4.4), Gradient Boosted Trees (2.4.5), Convolutional Neural Networks (2.4.6.1), and InceptionTime (2.4.6.2).

In Chapter 3 we present our methodology to determine which of these featurization methods and classification models achieve the best performance on the EEG Classification problem and can be used as a strong, task-agnostic baseline. The results of our experiments show us that the Univariate Featurization Methods - that is, those which create a feature vector by concatenating Fourier Transforms, Wavelet Transforms, and Hjorth Parameters of each individual channel of an EEG signal - are the best among the featurization methods, and that there is no statically significant difference between the performance of Logistic Regression, Support Vector Machines, Random Forest, and Gradient Boosted Trees as classification models. Also, it can be seen that the newly proposed Deep Learning architecture InceptionTime, despite performing worse than the set of all univariate featurization methods studied, had similar performance to any singular member of the set, a result which is in support of the great potential of Deep Learning methods in EEG Classification problems, as pointed out by Roy *et al.* (2019) and Lotte *et al.* (2018), among many others.

We hope this study can help researchers in the field by providing both a review of the most important statistical aspects related to the EEG Classification problem and a strong baseline method against which any new models can be compared. As for future work, new datasets can be explored in the quantitative part of the study to ensure that our results hold for an even greater array

of problems, new models can be added to the methodology to check if a new baseline emerges, and a deeper analysis of preprocessing techniques can be made to verify their impact on the final performance of the classifiers.

Referências Bibliográficas

Abhang *et al.***(2016)** Priyanka A Abhang, Bharti W Gawali e Suresh C Mehrotra. *Introduction to EEG-and speech-based emotion recognition*. Academic Press. Citado na pág. vii, 7, 8, 10, 12

Addison(2017) Paul S Addison. *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC press. Citado na pág. 25

Alturki *et al.***(2019)** Fahd A Alturki, Khalil AlSharabi, Majid Aljalal e Akram M Abdurraqeeb. A dwt-band power-svm based architecture for neurological brain disorders diagnosis using eeg signals. Em *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, páginas 1–4. IEEE. Citado na pág. 3

Asim e Zakria(2020) Muhammad Asim e Muaaz Zakria. Advanced knn: A mature machine learning series. *arXiv preprint arXiv:2003.00415*. Citado na pág. 31

Bao *et al.***(2011)** Forrest Sheng Bao, Xin Liu e Christina Zhang. Pyeeg: an open source python module for eeg/meg feature extraction. *Computational intelligence and neuroscience*, 2011. Citado na pág. 18

Bengfort *et al.***(2018)** Benjamin Bengfort, Rebecca Bilbro e Tony Ojeda. *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning*. "O'Reilly Media, Inc.". Citado na pág. vii, 28

Biau e Scornet(2016) Gérard Biau e Erwan Scornet. A random forest guided tour. *Test*, 25(2): 197–227. Citado na pág. 36

Birbaumer *et al.***(1999)** Niels Birbaumer, Nimr Ghanayim, Thilo Hinterberger, Iver Iversen, Boris Kotchoubey, Andrea Kübler, Juri Perelmouter, Edward Taub e Herta Flor. A spelling device for the paralysed. *Nature*, 398(6725):297–298. Citado na pág. 6

Bishop(2006) Christopher M Bishop. *Pattern recognition and machine learning*. springer. Citado na pág. 14, 32

Blankertz *et al.***(2002)** Benjamin Blankertz, Gabriel Curio e Klaus-Robert Müller. Classifying single trial eeg: Towards brain computer interfacing. Em *Advances in neural information processing systems*, páginas 157–164. Citado na pág. 6, 48

Brigato e Iocchi(2020) Lorenzo Brigato e Luca Iocchi. A close look at deep learning with small data. *arXiv preprint arXiv:2003.12843*. Citado na pág. 53

Cecchin *et al.***(2010)** Thierry Cecchin, Radu Ranta, Laurent Koessler, Olivier Caspary, Hervé Vespignani e Louis Maillard. Seizure lateralization in scalp eeg using hjorth parameters. *Clinical Neurophysiology*, 121(3):290–300. Citado na pág. 25

Chen e Guestrin(2016) Tianqi Chen e Carlos Guestrin. Xgboost: A scalable tree boosting system. Em *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, páginas 785–794. Citado na pág. 38

Choi(2017) Hyeong Choi. Classification and regression tree (cart), 2017. URL https://www.math.snu.ac.kr/~hichoi/machinelearning/lecturenotes/CART.pdf. Citado na pág. 37

Chollet(2017) Francois Chollet. The limitations of deep learning. *Deep Learning With Python*. Citado na pág. 53

Craik *et al.***(2019)** Alexander Craik, Yongtian He e Jose L Contreras-Vidal. Deep learning for electroencephalogram (eeg) classification tasks: a review. *Journal of neural engineering*, 16(3): 031001. Citado na pág. 1, 5, 51

Dadi *et al.***(2019)** Kamalaker Dadi, Mehdi Rahim, Alexandre Abraham, Darya Chyzhyk, Michael Milham, Bertrand Thirion, Gaël Varoquaux, Alzheimer's Disease Neuroimaging Initiative *et al.* Benchmarking functional connectome-based predictive models for resting-state fmri. *NeuroImage*, 192:115–134. Citado na pág. 2

Del Pra(2020) Marco Del Pra. Time series classification with deep learning, 2020. URL https://www.experfy.com/blog/time-series-classification-with-deep-learning/. Citado na pág. viii, 43, 44, 45

Demšar(2006) Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30. Citado na pág. 15

Emmert-Streib *et al.***(2020)** F Emmert-Streib, Z Yang, H Feng, S Tripathi e M Dehmer. An introductory review of deep learning for prediction models with big data. front. *Artif. Intell*, 3 (4). Citado na pág. 39, 41

Fawaz *et al.***(2019)** Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller e François Petitjean. Inceptiontime: Finding alexnet for time series classification. *arXiv preprint arXiv:1909.04939*. Citado na pág. 5, 43, 44, 51

Fletcher(2008) Tristan Fletcher. Support vector machines explained, 2008. URL https://cling.csd.uwo.ca/cs860/papers/SVM_Explained.pdf. Citado na pág. viii, 34, 35, 36

Géron(2019) Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media. Citado na pág. viii, 33, 39, 41, 42

Graps(2020) Amara Graps. An introduction to wavelets, 2020. URL https://www.eecis.udel.edu/~amer/CISC651/IEEEwavelet.pdf. Citado na pág. 19

Groose(2020) Roger Groose. Csc321 neural networks and machine learning, 2020. URL https://www.cs.toronto.edu/~lczhang/321/notes/note. Citado na pág. 40

Gu *et al.***(2020)** Xiaotong Gu, Zehong Cao, Alireza Jolfaei, Peng Xu, Dongrui Wu, Tzyy-Ping Jung e Chin-Teng Lin. Eeg-based brain-computer interfaces (bcis): A survey of recent studies on signal sensing technologies and computational intelligence approaches and their applications. *arXiv preprint arXiv:2001.11337*. Citado na pág. 1

Hamida *et al.***(2015)** Sana Tmar-Ben Hamida, Beena Ahmed e Thomas Penzel. A novel insomnia identification method based on hjorth parameters. Em *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, páginas 548–552. IEEE. Citado na pág. 25

Harpale e Bairagi(2016) Varsha K Harpale e Vinayak K Bairagi. Time and frequency domain analysis of eeg signals for seizure detection: A review. Em *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)*, páginas 1–6. IEEE. Citado na pág. 53

Hastie *et al.***(2009)** Trevor Hastie, Robert Tibshirani e Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media. Citado na pág. 4, 13, 14, 33, 36, 38, 47

He *et al.***(2016)** Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun. Deep residual learning for image recognition. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 770–778. Citado na pág. 44

Herrmann *et al.***(2014)** Christoph S Herrmann, Stefan Rach, Johannes Vosskuhl e Daniel Strüber. Time–frequency analysis of event-related potentials: a brief tutorial. *Brain topography*, 27(4):438–450. Citado na pág. 53

Hjorth(1970) Bo Hjorth. Eeg analysis based on time domain properties. *Electroencephalography and clinical neurophysiology*, 29(3):306–310. Citado na pág. 25

Houssein *et al.***(2017)** Essam H Houssein, Aboul Ella Hassanien e Alaa AK Ismaeel. Eeg signals classification for epileptic detection: a review. Em *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing*, páginas 1–9. Citado na pág. 47

Hu e Zhang(2019) Li Hu e Zhiguo Zhang. *EEG Signal Processing and Feature Extraction.* Springer. Citado na pág. vii, 7, 9, 14

Iscan *et al.***(2011)** Zafer Iscan, Zümray Dokur e Tamer Demiralp. Classification of electroencephalogram signals with combined time and frequency features. *Expert Systems with Applications*, 38 (8):10499–10505. Citado na pág. 53

Ismail Fawaz *et al.***(2019)** Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar e Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963. Citado na pág. vii, 16

Jia *et al.***(2012)** Wenchuan Jia, Dandan Huang, Xin Luo, Huayan Pu, Xuedong Chen e Ou Bai. Electroencephalography (eeg)-based instinctive brain-control of a quadruped locomotion robot. Em *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, páginas 1777–1781. IEEE. Citado na pág. 1

Jiang *et al.***(2019)** Linxing Jiang, Andrea Stocco, Darby M Losey, Justin A Abernethy, Chantel S Prat e Rajesh PN Rao. Brainnet: a multi-person brain-to-brain interface for direct collaboration between brains. *Scientific reports*, 9(1):1–11. Citado na pág. 1

Jolly *et al.***(2019)** Baani Leen Kaur Jolly, Palash Aggrawal, Surabhi S Nath, Viresh Gupta, Manraj Singh Grover e Rajiv Ratn Shah. Universal eeg encoder for learning diverse intelligent tasks. Em *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, páginas 213–218. IEEE. Citado na pág. 2

Kavasidis *et al.***(2017)** Isaak Kavasidis, Simone Palazzo, Concetto Spampinato, Daniela Giordano e Mubarak Shah. Brain2image: Converting brain signals into images. Em *Proceedings of the 25th ACM international conference on Multimedia*, páginas 1809–1817. Citado na pág. 1

Ke *et al.***(2017)** Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye e Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. Em *Advances in neural information processing systems*, páginas 3146–3154. Citado na pág. 38

Kesić e Spasić(2016) Srdjan Kesić e Sladjana Z Spasić. Application of higuchi's fractal dimension from basic to clinical neurophysiology: a review. *Computer methods and programs in biomedicine*, 133:55–70. Citado na pág. 4

Kiranyaz *et al.***(2019)** Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj e Daniel J Inman. 1d convolutional neural networks and applications: A survey. *arXiv preprint arXiv:1905.03554*. Citado na pág. viii, 41

Koelstra *et al.***(2011)** Sander Koelstra, Christian Muhl, Mohammad Soleymani, Jong-Seok Lee, Ashkan Yazdani, Touradj Ebrahimi, Thierry Pun, Anton Nijholt e Ioannis Patras. Deap: A database for emotion analysis; using physiological signals. *IEEE transactions on affective computing*, 3(1):18–31. Citado na pág. 6, 48

Kumar *et al.***(2017)** Nitendra Kumar, Khursheed Alam e Abul Hasan Siddiqi. Wavelet transform for classification of eeg signal using svm and ann. *Biomedical and Pharmacology Journal*, 10(4): 2061–2069. Citado na pág. 4

Kutner *et al.***(2005)** Michael H Kutner, Christopher J Nachtsheim, John Neter, William Li *et al. Applied linear statistical models*, volume 5. McGraw-Hill Irwin New York. Citado na pág. 33

Li *et al.***(2018)** Ren Li, Jared S Johansen, Hamad Ahmed, Thomas V Ilyevsky, Ronnie B Wilbur, Hari M Bharadwaj e Jeffrey Mark Siskind. Training on the test set? an analysis of spampinato et al.[31]. *arXiv preprint arXiv:1812.07697*. Citado na pág. 1, 3

Lin *et al.***(2012)** Jessica Lin, Rohan Khade e Yuan Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315. Citado na pág. viii, 29, 30

Lines *et al.***(2016)** Jason Lines, Sarah Taylor e Anthony Bagnall. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. Em *2016 IEEE 16th international conference on data mining (ICDM)*, páginas 1041–1046. IEEE. Citado na pág. 4

Lotte *et al.***(2018)** Fabien Lotte, Laurent Bougrain, Andrzej Cichocki, Maureen Clerc, Marco Congedo, Alain Rakotomamonjy e Florian Yger. A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update. *Journal of neural engineering*, 15(3):031005. Citado na pág. vii, 1, 3, 11, 47, 52, 55

Martin(2015) Dinov Martin. Using dynamic time warping for quantifying effects of sinusoidal oscillation deviations during eeg time series prediction and for finding interesting eeg and fmri changes. *BMC Neuroscience*, 16(S1):P63. Citado na pág. 5

Mertins e Mertins(1999) Alfred Mertins e Dr Alfred Mertins. *Signal analysis: wavelets, filter banks, time-frequency transforms and applications*. John Wiley & Sons, Inc. Citado na pág. 21

Mueller(2020) Andreas Mueller. Cross validation and grid search, 2020. URL https://amueller.github.io/ml-training-intro/slides/03-cross-validation-grid-search.html. Citado na pág. vii, 14, 15

Müller(2007) Meinard Müller. *Information Retrieval for Music and Motion*. Springer Verlag. ISBN 3540740473. Citado na pág. vii, 25, 26, 27

Nielsen(2015) Michael A Nielsen. *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA. Citado na pág. 41

Nocedal e Wright(2006) Jorge Nocedal e Stephen Wright. *Numerical optimization*. Springer Science & Business Media. Citado na pág. 34

Omerhodzic *et al.***(2013)** Ibrahim Omerhodzic, Samir Avdakovic, Amir Nuhanovic e Kemal Dizdarevic. Energy distribution of eeg signals: Eeg signal wavelet-neural network classifier. *arXiv preprint arXiv:1307.7897*. Citado na pág. 2

Paul(2016) Subrata Paul. Boosted regression and classification trees, 2016. URL http://math.ucdenver.edu/~spaul/empty/hostedfiles/Resources/Boosted_Trees.pdf. Citado na pág. viii, 39

Perrin *et al.***(1989)** François Perrin, J Pernier, O Bertrand e JF Echallier. Spherical splines for scalp potential and current density mapping. *Electroencephalography and clinical neurophysiology*, 72(2):184–187. Citado na pág. 11

Polikar(1996) Robi Polikar. The wavelet tutorial, 1996. URL http://web.iitd.ac.in/~sumeet/WaveletTutorial.pdf. Citado na pág. vii, 19, 20, 21, 23, 24

Qazi *et al.***(2019)** Emad-ul-Haq Qazi, Muhammad Hussain, Hatim AboAlsamh e Ihsan Ullah. Automatic emotion recognition (aer) system based on two-level ensemble of lightweight deep cnn models. *arXiv preprint arXiv:1904.13234*. Citado na pág. 1, 2

Raschka(2018a) Sebastian Raschka. Stat 479: Machine learning lecture notes, 2018a. URL https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/07_ensembles_notes.pdf. Citado na pág. viii, 38

Raschka(2018b) Sebastian Raschka. Machine learning lecture notes, 2018b. URL https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf. Citado na pág. viii, 32

Rodriguez *et al.***(2018)** Paula Ivone Rodriguez, Jose Mejia, Boris Mederos, Nayeli Edith Moreno e Victor Manuel Mendoza. Acquisition, analysis and classification of eeg signals for control design. Citado na pág. 3

Roy *et al.***(2019)** Yannick Roy, Hubert Banville, Isabela Albuquerque, Alexandre Gramfort, Tiago H Falk e Jocelyn Faubert. Deep learning-based electroencephalography analysis: a systematic review. *Journal of neural engineering*, 16(5):051001. Citado na pág. 1, 3, 47, 55

Sanei e Chambers(2013) Saeid Sanei e Jonathon A Chambers. *EEG signal processing*. John Wiley & Sons. Citado na pág. 7, 10

Schaar(2017) Mihaela Schaar. Classification and regression trees, 2017. URL http://www.stats.ox.ac.uk/~flaxman/HT17_lecture13.pdf. Citado na pág. viii, 36

Schäfer e Högqvist(2012) Patrick Schäfer e Mikael Högqvist. Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. Em *Proceedings of the 15th international conference on extending database technology*, páginas 516–527. Citado na pág. viii, 28, 29, 30

Schäfer e Leser(2017) Patrick Schäfer e Ulf Leser. Multivariate time series classification with weasel+ muse. *arXiv preprint arXiv:1711.11343*. Citado na pág. viii, 5, 28, 31

Senin(2008) Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23):40. Citado na pág. 5, 25

Shaker(2006) Maan M Shaker. Eeg waves classifier using wavelet transform and fourier transform. *brain*, 2(3). Citado na pág. 4

Siuly *et al.***(2016)** Siuly Siuly, Yan Li e Yanchun Zhang. Eeg signal analysis and classification. *IEEE Trans Neural Syst Rehabilit Eng*, 11:141–4. Citado na pág. 4

Snodgrass e Vanderwart(1980) Joan G Snodgrass e Mary Vanderwart. A standardized set of 260 pictures: norms for name agreement, image agreement, familiarity, and visual complexity. *Journal of experimental psychology: Human learning and memory*, 6(2):174. Citado na pág. 6, 48

Spampinato *et al.***(2017)** Concetto Spampinato, Simone Palazzo, Isaak Kavasidis, Daniela Giordano, Nasim Souly e Mubarak Shah. Deep learning human mind for automated visual classification. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 6809–6817. Citado na pág. 1

Stone(2002) James V Stone. Independent component analysis: an introduction. *Trends in cognitive sciences*, 6(2):59–64. Citado na pág. 13

Subasi e Ercelebi(2005) Abdulhamit Subasi e Ergun Ercelebi. Classification of eeg signals using neural network and logistic regression. *Computer methods and programs in biomedicine*, 78(2): 87–99. Citado na pág. 2

Sun e Zhou(2014) Shiliang Sun e Jin Zhou. A review of adaptive feature extraction and classification methods for eeg-based brain-computer interfaces. Em *2014 International Joint Conference on Neural Networks (IJCNN)*, páginas 1746–1753. IEEE. Citado na pág. 4, 47

Szegedy *et al.***(2016)** Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke e Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*. Citado na pág. 43

Tharwat(2018) Alaa Tharwat. Independent component analysis: An introduction. *Applied Computing and Informatics*. Citado na pág. 13

Tirupattur *et al.***(2018)** Praveen Tirupattur, Yogesh Singh Rawat, Concetto Spampinato e Mubarak Shah. Thoughtviz: visualizing human thoughts using generative adversarial network. Em *Proceedings of the 26th ACM international conference on Multimedia*, páginas 950–958. Citado na pág. 3

Tolpygo(2016) Alexander Tolpygo. Dynamic time warping: Time series analysis ii, 2016. URL https://sflscientific.com/data-science-blog/2016/6/3/dynamic-time-warping-time-series-analysis-ii. Citado na pág. vii, 26

Van Drongelen(2018) Wim Van Drongelen. *Signal processing for neuroscientists*. Academic press. Citado na pág. vii, 17, 18

Vézard *et al.***(2015)** Laurent Vézard, Pierrick Legrand, Marie Chavent, Frédérique Faïta-Aïnseba e Leonardo Trujillo. Eeg classification for the detection of mental states. *Applied Soft Computing*, 32:113–131. Citado na pág. 3

Vidaurre *et al.***(2009)** Carmen Vidaurre, Nicole Krämer, Benjamin Blankertz e Alois Schlögl. Time domain parameters as a feature for eeg-based brain–computer interfaces. *Neural Networks*, 22(9):1313–1319. Citado na pág. 25

Yadava *et al.***(2017)** Mahendra Yadava, Pradeep Kumar, Rajkumar Saini, Partha Pratim Roy e Debi Prosad Dogra. Analysis of eeg signals and its application to neuromarketing. *Multimedia Tools and Applications*, 76(18):19087–19111. Citado na pág. 1

Yamauchi *et al.***(2015)** Takashi Yamauchi, Kunchen Xiao, Casady Bowman e Abudullah Mueen. Dynamic time warping: A single dry electrode eeg study in a self-paced learning task. Em *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, páginas 56–62. IEEE. Citado na pág. 5

Zemel(2016) Richard Zemel. Csc 411: Neural networks i, 2016. URL https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/10_nn1.pdf. Citado na pág. viii, 40

Zhang *et al.***(2019)** Xiang Zhang, Lina Yao, Xianzhi Wang, Jessica Monaghan, David Mcalpine e Yu Zhang. A survey on deep learning based brain computer interface: Recent advances and new frontiers. *arXiv preprint arXiv:1905.04149*. Citado na pág. 1

Zhang *et al.***(1995)** Xiao Lei Zhang, Henri Begleiter, Bernice Porjesz, Wenyu Wang e Ann Litke. Event related potentials during object recognition tasks. *Brain research bulletin*, 38(6):531–538. Citado na pág. 48