1 2  9 0
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Júlio Cordeiro Medeiros

Software code complexity assessment using EEG features

Towards a new software engineering paradigm using biofeedback

Thesis submitted to the
University of Coimbra for the degree of
Master in Biomedical Engineering

Supervisors:
Prof. Dr. César Alexandre Domingues Teixeira (DEI-FCTUC & CISUC)
Prof. Dr. Paulo Fernando Pereira de Carvalho (DEI-FCTUC & CISUC)

Coimbra, 2019

This work was developped under project BASE, Biofeedback Augmented Software Engineering (POCI - 01-0145 - FEDER- 031581), in collaboration with:

CISUC - Centre for Informatics and Systems of University of Coimbra

"*Shall I refuse my dinner because I do not fully understand the process of digestion?*"

OLIVER HEAVISIDE

Agradecimentos

Em primeiro lugar, gostaria de expressar a minha enorme gratidão ao Professor Doutor César Teixeira e Professor Doutor Paulo de Carvalho por toda a excelente orientação ao longo de este ano letivo. Um muito obrigado pela disponibilidade, motivação, discussão, ajuda e aprendizagem que proporcionaram e transmitiram ao longo da realização deste trabalho.

Quero também agradecer ao projecto BASE, Biofeedback Augmented Software Engineering (POCI - 01-0145 - FEDER- 031581) e à FCT, pelo apoio financeiro prestado no desenvolvimento deste trabalho, bem como aos elementos do grupo do projecto pelo apoio ao longo do estudo.

Gostaria de também agradecer a todos os meus colegas de laboratório, pelo ambiente saudável, educativo e motivante ao longo do ano letivo. Quero deixar um especial obrigado à Adriana Leal por toda a preocupação, opiniões e críticas, e paciência que teve.

Aos meus colegas e amigos de Coimbra, um grande obrigado por estes 5 anos de companheirismo, por todos os momentos partilhados, desde os mais educativos até aos mais boémios. De certo muitos momentos destes ficarão marcados para a vida, bem com as amizades feitas aqui, em Coimbra, que permanecerão. Quero agradecer especialmente ao meu amigo Fábio pelo companheirismo desde o primeiro ano, tanto nos momentos académicos e de convívio como também nos momentos sérios, obrigado pelos conselhos e pela motivação, de certo uma grande amizade que ficará. Quero também agradecer aos meus amigos dos Açores, que apesar de longe, estiveram sempre presentes, quer para espairecer como também para motivar.

Gostaria também de deixar aqui o maior agradecimento de todos, à minha família, especialmente aos meus Pais e aos meus Irmãos, por todos os sacrifícios, preocupações, apoios e motivações em todos os momentos que mais precisei ao longo do meu percurso académico, começando no primeiro dia que pisei o solo de Coimbra. Um Eterno Muito Obrigado. Quero agradecer ainda à minha Madrinha, por toda a preocupação e apoio, e por ter fomentado este bichinho pelas ciências desde míudo. Sem dúvida que sem vocês não estaria chegado aqui.

Por último, mas de grande importância, quero agradecer à minha namorada, Francisca, por todos os momentos vividos e por acompanhar-me ao longo destes anos de faculdade, sempre presente com uma atitude incrivelmente positiva tanto nos bons como também nos maus momentos. Devo muito à sua pessoa. Muito obrigado por tudo, e especialmente pela paciência neste último ano atarefado.

Um Muito Obrigado a Todos!

Resumo

O presente estudo foi desenvolvido no âmbito do projeto BASE, Biofeedback Augmented Software Engineering (POCI - 01-0145 - FEDER-031581) que se encontra a decorrer, e que visa a monitorização das funções cognitivas do cérebro durante o desenvolvimento de código de software, de modo a detectar possíveis bugs que possam ocorrer devido a mudanças no estado emocional do programador. Mais especificamente, o objetivo deste estudo inicial é investigar a atividade cerebral durante a compreensão do código por meio da análise de sinais de Eletroencefalograma (EEG) adquiridos de vários voluntários e tentar estabelecer possíveis biomarcadores de EEG sensíveis a diferentes níveis de carga mental. O uso de EEG para esse fim é relevante, uma vez que os estudos existentes em detecção de bugs foram principalmente focados em técnicas de neuroimagem com base em *Functional Magnetic Ressonance Imaging* (fMRI), as quais apresentam desvantagens, como por exemplo, o desconforto para o sujeito e a impossibilidade de serem adquiridas em condições normais de programação.

O presente estudo foi conduzido usando informação de 64 canais de EEG adquiridos de 30 participantes durante tarefas de compreensão de código. Os sujeitos foram submetidos a três ensaios diferentes, correspondentes a três tarefas de compreensão do código de diferente dificuldade. Os três níveis diferentes de complexidade de código considerados foram de acordo com cinco métricas de complexidade de software amplamente utilizadas na area de engenharia de software.

Através de uma sequência de métodos de pré-processamento, artefatos foram identificados e removidos dos sinais de EEG, para posteriormente proceder-se à extração

de caraterísticas lineares e não lineares. Assim, foi possível investigar a possibilidade de distinguir diferentes níveis de complexidade com base nas características do EEG. Para isso, foram utilizados diferentes tipos de modelos de classificação, binário ou multiclasse, através da combinação de quatro métodos de seleção/redução de caracteristicas com quatro classificadores diferentes.

Considerando um modelo multiclasse utilizando *Principal Component Analysis* (PCA) e o classificador *Support Vector Machine* (SVM) com kernel linear, obteve-se um F-Measure de 93.60% para a complexidade do código fácil, 50.60% para a complexidade do código intermédio, 47.09% para a complexidade do código avançada e 94.42 % para a tarefa de controlo (leitura de texto). Estes resultados revelam uma evidência de saturação do esforço mental com o aumento da complexidade do código e também sugerem que as métricas de complexidade de código usadas actualmente não captam a carga cognitiva, e por isso podem não ser a melhor abordagem para avaliar o risco de bugs nos códigos.

A partir da análise realizada neste estudo, também se verificou que as características relacionadas com as atividades Teta, Alfa e Beta foram as mais comuns entre as características selecionadas com maior poder de discriminação da complexidade das tarefas. No que respeita aos canais que contribuíram com mais informação, eles estavam localizados predominantemente na região frontal (principalmente em Fz, F2 e FCz), centro-parietal (principalmente em CPZ e CP2) e parietal (principalmente em Pz). Estes resultados estão de acordo com as características e regiões (lobos frontal e parietal) relatadas em estudos relacionados, como sendo as mais relevantes para medir a carga cognitiva nas áreas investigação de compreensão de código e da carga de trabalho mental.

Por fim, como estudo preliminar, utilizando duas das características mais discriminantes, foi explorada a possibilidade de realizar-se uma análise espaço-temporal, a fim de identificar áreas que exibiram um esforço mental elevado durante a tarefa de código que está sendo realizado. Estas áreas foram comparadas com (i) os aglomerados dos dados do *Eye tracking*; (ii) as regiões críticas apontadas por profissionais especializados; e (iii) outros dois biosinais (Variabilidade da Frequência Cardíaca e

Pupilografia).

Palavras-Chave: Processamento de Bio-sinais, Electroencefalograma, Biofeedback, Engenharia de Software

Abstract

This study was developed on behalf of the on-going project BASE, Biofeedback Augmented Software Engineering (POCI - 01-0145 - FEDER- 031581), which aims at monitoring cognitive functions of the brain during code development to detect possible bugs that might occur due to shifts in the emotional condition of the subject. Specifically, the goal of this initial study is to investigate the brain activity during code comprehension through the analysis of Electroencephalogram (EEG) signals acquired from multiple volunteers and try to establish possible EEG biomarkers sensitive to different levels of mental workload. The use of EEG for this purpose is relevant, since the existing studies in bug detection were mostly focused on neuroimaging techniques based on fMRI, which have disadvantages, such as subject discomfort and impossibility to be acquired in normal programming condition.

The study reported herein was conducted using information from 64 channels of EEG recorded from 30 subjects during code comprehension tasks. The subjects were submitted to three different trials corresponding to three different difficulty code comprehension tasks. The three different code complexity levels considered were according to five software complexity metrics widely used.

Through a sequence of preprocessing methods, artifacts were identified and removed from EEG signals for further extraction of linear and nonlinear features. Then, the possibility to distinguish different levels of complexity based on EEG features was investigated. For this purpose, different types of classification models were considered, either binary or multiclass, by combining four different feature selection/reduction methods with four different classifiers.

Considering a multiclass model using Principal Component Analysis (PCA) and a Support Vector Machine (SVM) classifier with linear kernel, it was obtained a F-Measure of 93.60% for code complexity easy, 50.60% for code complexity intermediate, 47.09% for code complexity advanced and 94.42% for the control (reading) task. These results reveals an evidence of mental effort saturation as code complexity increases and also suggest that current code complexity metrics do not capture cognitive load and might not be the best approach to assess bug risk.

From the analysis, it was also found that the features related with the Theta, Alpha and Beta activity were the most common among the selected features with highest discriminative power. Concerning the channels which contributed with more information, they were located predominantly in frontal (mainly in Fz, F2 and FCz), central-parietal (mainly in CPz and CP2) and parietal (mainly in Pz) regions. These findings are in agreement with the features and regions (Frontal and Parietal Lobes) more relevant to cognitive load in the areas of mental workload and code comprehension, reported in related studies.

Finally, as a preliminary study, using two of the most discriminant features, it was explored the possibility of a space-temporal analysis in order to spot areas that exhibited a higher mental effort during the code task being performed. These areas were compared to the (i) clusters of Eye tracking data; (ii) critical regions pointed by expert professionals; and (iii) other two biosignals (Heart Rate Variability and Pupillography).

Keywords: Bio-signal Processing, Electroencephalogram, Biofeedback, Software Engineering

Contents

List of Figures

List of Tables

List of Abbreviations

ANS Autonomic Nervous System. 2, 8, 9

AR Autoregressive. 41, 42

BSS Blind Source Separation. 24, 29

CEV cumulative explained variance. 49

CIC Comments and Identifiers Consistency. 5

CLT Cognitive Load Theory. 6

CNS Central Nervous System. 7

CR Comments Readability. 5

ECG Electrocardiogram. 57, 58

EDA Electrodermal Activity. 15, 21

EEG Electroencephalogram. 2, 3, 4, 9, 10, 11, 13, 14, 15, 16, 17, 21, 22, 23, 24, 26, 27, 28, 29, 33, 34, 35, 37, 38, 39, 40, 41, 55, 57, 58, 75, 77

EMG Electromyogram. 32

FIR Finite Impulse Response. 24

FLDA Fisher Linear Discriminant Analysis. 49, 50, 66, 67, 77, 80, 82

fMRI Functional Magnetic Resonance Imaging. 13

fNRIS Functional Near-Infrared Spectroscopy. 13

HRV Heart Rate Variability. 18, 57, 58

IAF Individual Alpha Frequency. 16

ICA Independent Component Analysis. 24, 25, 29, 30, 31

ICG Impedance Cardiography. 21

IIR Infinite Impulse Response. 24

k-NN k-Nearest Neighbors. 51, 66, 67, 69, 77, 80

LTM Long-Term Memory. 6

MSE mean square error. 41

NMI Normalized Mutual Information. 48

PACF Partial Autocorrelation function. 41

PCA Principal Component Analysis. 48, 49, 60, 62, 64, 67, 69, 71, 73, 78, 80, 84

PNS Peripheral Nervous System. 7

PPG Photoplethysmography. 21

PSD Power Spectral Density. 14, 15, 16, 35, 36

SNR signal-to-noise ratio. 25

SNS Somatic Nervous System. 8

SVM Support Vector Machines. 14, 17, 50, 61, 66, 67, 69, 77, 79, 80, 82, 84

TSE total squared error. 41

WM Working Memory. 6, 7

1

Introduction

This project was developed within the scope of the curricular unit Project of the Integrated Master in Biomedical Engineering of the Faculty of Sciences and Technology of the University of Coimbra, in the academic year 2018-2019. It is part of the on-going project BASE, Biofeedback Augmented Software Engineering (POCI - 01-0145 - FEDER- 031581), funded by FCT.

1.1 Context

Nowadays with the continuous evolution of technology, software programming is the groundwork in a wide range of areas, going from health applications to autonomous driving challenges. Software programmers face an enormous pressure to develop the programs that integrate all the requirements for each field. A great effort is demanded to detect/fix bugs, change code and understand different types of logical thinking and different languages, that, in the end, enables the acquisition of different mathematical or symbolic manipulation expertise. Every year there is a great loss in software industry, hitting the $1.7 trillion in 2017, due to software bugs or security failures and also because of the amount of time it takes for a programmer to understand codes and detect those bugs [1]. In this industry, on average, for a completed code, there are about 15-50 errors per 1000 lines of code [2]. This is where project BASE comes in, with the goal of developing an approach using biofeedback that will allow bug warning and programmers to review certain areas of the code that need a more thorough inspection.

1.2 Motivation

Studies reveal that about 70% of a programmer's time is spent on code comprehension [3]. In addition, the quality of code is closely linked to the complexity of code and to the programmers ' capacity to comprehend and master such complexity. For this reason, it is crucial to understand the relationship between the neural mechanisms engaged in code comprehension and the metrics used to evaluate code complexity.

1.3 Objectives

The work developed for this thesis is an initial study integrated in the project BASE that aims at understand the brain mechanisms involved in different code comprehension tasks with different complexities based on neurophysiological responses observed in the Electroencephalogram (EEG) information. This inspection is performed by using a conventional pipeline, which involves: pre-processing, feature extraction & selection and classification.

Another goal of this thesis is also to inspect the relationship between possible EEG biomarkers and Autonomic Nervous System (ANS) responses in order to investigate the possibility of replace EEG by more comfortable measurement methods that allow monitoring of such responses in daily life conditions.

1.4 General outline of the thesis

The structure of this thesis proceeds as follows: **Chapter 2** describes briefly the basic concepts necessary to understand the thesis background, such as the Software Metrics usually considered, the Cognitive Load, the Human Nervous System and the Electroencephalography; **Chapter 3** focus on the state of the art of code comprehension assessment in the field of software engineering and on the use of the EEG for this purpose; **Chapter 4** describes the data acquisition protocol, obtained in the scope of a previous work developed for the project, and that will be used in this

thesis; **Chapter 5** details the methods used for preprocessing of EEG data, feature engineering, classification, statistical analysis and the types of analysis performed; **Chapter 6** presents the main results and respective discussion from different perspectives; and finally, **Chapter 7** highlights the main findings and also points out the limitations and future work.

1.5 Contributions

The main contributions from this thesis are as follows:

- Evidence of how EEG can be a powerful technique for biofeedback in the context of Software Reliability;
- Comparison between models, using different feature selection methods and classifiers, in the prediction of code complexities;
- The best features and more relevant regions, with the purpose of validation of existent and new EEG biomarkers for mental workload related with Software Engineering;
- Evidence of saturation of the mental effort of the participants with the complexity level;
- Verification that the complexity metrics used worldwide in Software Engineering are not the best approach for measuring code complexity.

The following publications were also produced in the context of this thesis:

- Abstract and poster presentation at the IEEE 6^{th} Portuguese Meeting in Bioengeneering (6^{th} ENBENG) organized by the IEEE EMBS Portugal Chapter, held on 22-23 February in Lisbon. The results of a preliminary study regarding the assessment of EEG's potential to distinguish software codes with different complexities were presented;
- Four page, 2 columns, article submitted and accepted at the 41^{st} International Engineering in Medicine and Biology Conference (41^{st} EMBC) held in Berlin,

Germany from 23 to 27 of July, 2019. The article is a comprehensive study on the assessment of software code complexity using several linear EEG features.

2

Basic Concepts

2.1 Software Engineering Metrics

In Software Engineering, there are several types of metrics used worldwide for many purposes, typically in the context of software testing, identification of software parts that may need to be rewritten, improvement of code quality and reduction of maintenance costs, among others [4]. These metrics can be relative to the code, documentation, or developer subject [5].

Concerning code-related metrics, as the name suggests, it focuses on the algorithm code, and there are several metrics for this target, such as the well known McCabe Cyclomatic Complexity [6], average Number of Nested Block Depth, Number of Parameters and Lines of codes, among others [7].

The documentation related metrics are used to measure the quality of the documentations and include the Comments Readability (CR) and the Comments and Identifiers Consistency (CIC) [8].

At last, the developer-related metrics are used to quantify the programmer's experience, such as the time (in years) spent programming in general or/and in a specific programming language [5].

A brief description of the metrics used for code complexity and comprehensibility assessment [9] that are mentioned throughout this thesis can be found in Table 2.1.

Table 2.1: Brief description of the code complexity software metrics used in the present work.

Code Complexity Metrics	**Description**
Lines of codes	Measures the number of lines in the code
Number of Parameters	Measures the number of parameters in the code
Weighted Method Count	Measures the sum of the complexity of the code methods
McCabe Cyclomatic Complexity	Measures the number of paths that are linearly independent in the code
Nested Block Depth	Measures the maximum nested block depth in the code, i.e., the depth level of the block (e.g. condition) that is deeper in the code

2.2 Cognitive Load

In information processing there are two fundamental keys: Working Memory (WM) and Long-Term Memory (LTM). The WM has limited capacity and a person is purely cognizant of the information that is held in the WM [10]. It is based on this ground that the Cognitive Load Theory (CLT) [11] is formed. The LTM mainly differs from the WM in duration and in capacity. While WM has limited capacity, LTM does not have a capacity limit, it has an ample knowledge storage domain. LTM organizes the information, hence helping to reduce the load of WM to process the information [12].

The CLT, developed and suggested in 1988 by Sweller [11], comprises the existing correlations between the WM and the LTM, how organization and information grouping affect the WM capacity and disclose the neural activity behind certain tasks. This theory defines the cognitive load in three types: Intrinsic, Extraneous and Germane cognitive load. Intrinsic cognitive load is independent from the task representation, being only affected by the demand of the inherent complexity of the task on the WM of the user. Extraneous cognitive load, on the contrary is influenced by the representation of the task to the user, i.e., information presented in a

more complex manner requires more WM capacity. The latter, Germane cognitive load, is related to the effort required from a task to process and construct new ways of organizing ideas, i.e., finding a novel path to complete a task increases the Germane cognitive load. This cognitive load is associated with schemas construction – organized pattern of information in the brain's neural network [13]. The junction of these three types of cognitive load forms the total cognitive load demanded for a certain task. From the point of view of Software Engineering, the total cognitive load is sometimes also referred as mental effort or cognitive stress [14–16], since most of the studies do not want to assess the individual type of cognitive load but the overall effort on the task being performed.

2.3 Human Nervous System

The nervous system controls everything in the human body. From physiologic needs to muscular responses and reflexes, all systems depend on the nervous system. The human body works with constant series of stimuli inputs, their integration in the nervous system and output responses to each specific stimuli received.

The electrical signal produced in the human body is a result of constant changes on the cell's membrane permeability to specific ions, Na^+ and K^+. The cell membrane has a resting potential of -70mV, but when a stimuli is generated, it triggers the depolarization of the membrane, initiating the action potential. This action potential will be carried through a neuron chain to the brain, by triggering an electrical chemical gradient generated by a series of depolarizations, repolarizations and hyperpolarizations. When the signal reaches the brain, the brain elaborates an adequate response and sends the information again via neurons, that communicate between them, transmitting the electrical/chemical signal through synapses, until it generates the output [17].

The nervous system is divided in two main parts, the Central Nervous System (CNS) and the Peripheral Nervous System (PNS). The first one involves the brain and the spinal cord, that function as the main control centre of the Nervous System, where

all the stimuli come to be interpreted, and all the specific responses are formed according to inputs. The Peripheral System is the one that include all the nerves that branch from the brain/spinal cord to every part of the body. This system is responsible for transporting the signals, either to or from the brain, ensuring, for example, that muscles contract or that the secretion of certain chemicals increase or decrease [17, 18].

The Central Nervous System, more precisely the cortex of the brain, can be divided in four main lobes: Frontal, Temporal, Parietal and Occipital. Each one of those four lobes, represented in Fig. 2.1, is responsible or associated to different functions [19]. With regard to the Frontal lobe, it is associated with motor actions, working memory and conscious thoughts. As for the Temporal lobe, it is related to auditory and olfactory functions. The Parietal lobe is linked to the integration of the sensory information from several senses. Finally, the Occipital lobe is associated to visual functions [19, 20].

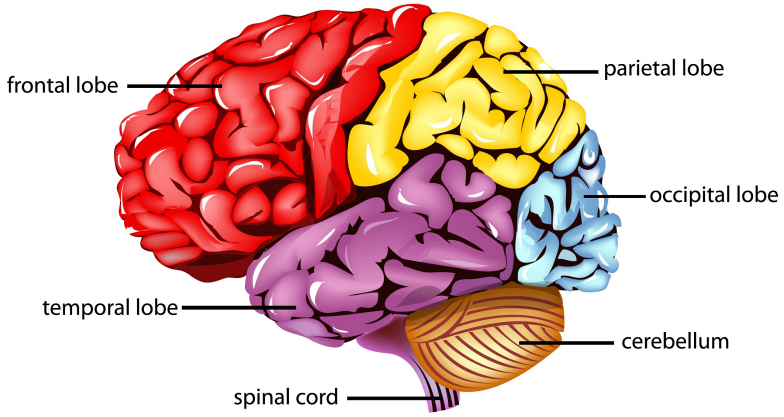


Figure 2.1: Division of the cortex into four lobes: frontal, temporal, parietal and occipital. Adapted from Alotaiby et al. [21].

Concerning the Peripheral System, it can be divided in two divisions, the Sensory Division, involving afferent pathways, the one that acquires sensory stimuli, and the Motor Division that includes the efferent pathways, responsible to send the response to muscles and glands. The Motor division is then, composed by the Somatic Nervous System (SNS) and the Autonomic Nervous System (ANS). The SNS commands voluntary contractions, i.e., the one responsible for big muscle movement. The ANS, therefore, is related to all the involuntary functions, such as heart beating, lungs

breathing, stomach digestion or temperature changing. Finally, this system, that controls the involuntary movements, is divided in Parasympathetic and Sympathetic Systems [18]. These two branches that integrate the ANS, have opposite functions: the Sympathetic System alarms the body for certain situations, while the Parasympathetic System is responsible for the relaxation. When combined, both systems responses deliver sensations such as stress, relaxation, fear or panic, for example [17, 22].

The responses of the ANS, such as, increasing heart rate, changes of breathing frequency, blood pressure, secretions, temperature changing are correlated with several feelings, e.g., stress, panic, among others. Then, studying EEG signals to predict stages of stress, attention and mental effort, should be possible, since all these responses of the body are signals generated in the brain. In order to uncover the neural mechanisms behind code comprehension and bug detection, an analysis of the signals that originate the responses to situations of increased mental effort, required for a programmer, through scalp EEG recordings, would , theoretically, help to identify some neuronal mechanism biomarkers involved in these type of tasks.

Code comprehension involves cognitive processes, which requires different capabilities of the brain, such as abstraction level, memory, information processing, logical thinking, among others [14, 23]. Therefore, the effort spent during these processes, that can be perceived as cognitive stress, evidences the importance of evaluating measurements that can capture the cognitive load of the programmers during different types of tasks.

2.4 Electroencephalography

Human EEG recording started in 1924 by the German psychiatrist Hans Berger [24] and from there on EEG became an important tool in the field of neurology and clinical neurophysiology [25]. EEG can be a powerful tool in this study, since it records electrical activity of the brain, which is one of the most relevant physiological areas analysed when assessing cognitive workload [26].

The brain electrical activity can be captured using electrodes on the surface of the scalp (non-invasive) or using intracranial (invasive) electrodes. These electrodes will measure the voltage fluctuations that occur when there are local ionic current flows produced by the sum of activations of thousands of neurons [25].

Regarding the type of EEG used in this work, the surface-EEG, it is important to notice that this is the most frequently used in several studies and applications, since it offers advantages such as its non-invasive nature, more general perspective of the brain than the localized invasive measurement approach, and, with advances in technology, it is turning into a low cost and portable solution to understand the neural mechanisms [27, 28]. However, one of the drawbacks that can be pointed out is the fact that the neural signals are filtered by the bone and skin, resulting in a reduced bandwidth. Other negative aspect is the contamination by artifacts (electrical & muscular) that makes the analysis more difficult. [29].

Using surface EEG, there are several standard configurations for the electrodes location, e.g, the well known International 10-10 or 10-20 System configurations. The difference between the configurations is due to the distance between adjacent electrodes. This distance corresponds to a percentage of the total distance from front to back or left to right side of the head. For better visualization, Fig. 2.2 depicts an example of the International 10-20 System.

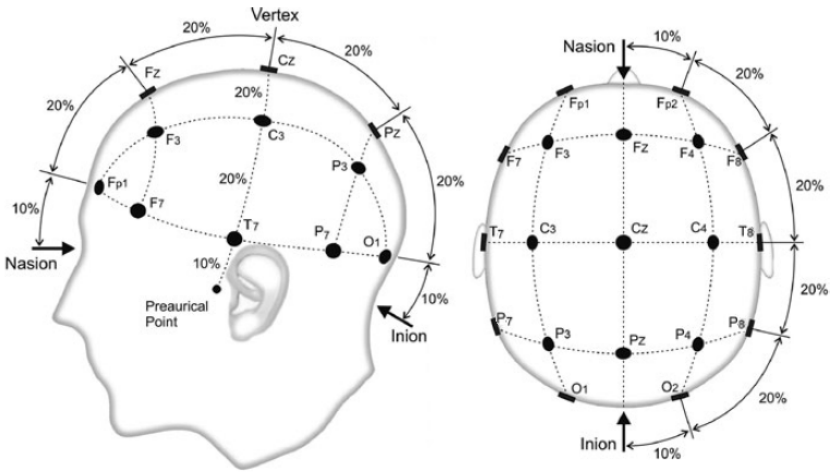


Figure 2.2: Configuration of the electrodes placements using the International 10-20 System (adapted from Malmivuo et al. [30]).

As for the electrode labels, each electrode is named based on its location on the subject's scalp: the first letter is defined with respect to the region of the brain that it belongs (Pre-Frontal - "Fp", Frontal - "F", Temporal - "T", Central - "C", Parietal - "P" and Occipital - "O") or between regions (Pre-Frontal and Frontal - "AF", Frontal and Central - "FC", Frontal and Temporal - "FT", Central and Parietal - "CP" and "PO" if between Parietal and Occipital). Furthermore, if the electrode is located in the midline sagittal plane of the brain it will have an additional letter "z". If a given electrode is located on the left hemisphere it will have a odd number associated, while if it is located on right hemisphere, an even number is considered.

Finally, the signal recorded from the EEG is always a relative value, since it is the difference of the electric potential between the electrode where it is being measured and a reference electrode, typically spotted between Cz and Pz. The signals are typically measured in microvolts (μV), presenting an amplitude about 10-100μV and being commonly divided into 5 main frequency bands (Delta, Theta, Alpha, Beta and Gamma) [31], presented in the Table 2.2.

Table 2.2: EEG frequency brain bands and associated brain state

Name	**Frequency Range**	**Associated state of brain**
Delta	<4 Hz	Unconscious / Deep sleep
Theta	4 - 8 Hz	Conscious / Imagination / Memory
Alpha	8 - 13 Hz	Conscious / Relaxed mental activity
Beta	13 - 30 Hz	Conscious / Emotional /Focused
Gamma	>30 Hz	Conscious / High mental activity

3

State of the Art

The interest in the field of bug detection and warning during programming/coding and the underlying neurophysiological processes has recently emerged in the scientific community. In fact, there are a few studies in this area using Functional Magnetic Resonance Imaging (fMRI) [23, 32]. In these two imaging studies, the authors analysed the Brodmann areas [33] that were more activated during code comprehension and bug detection, and found that frontal and parietal lobes appeared to be the most activated regions during those tasks. Such achievements prompt further investigation of other imaging tools and also of biosignals such as Electroencephalogram (EEG), which, given the current technological advances and the arrival of low-cost off-the-shell acquisition devices, are becoming more user-friendly [28, 34].

In recent years, the measurement of the mental workload during programming has attracted considerable interest in different areas, ranging from the educational to the software engineering sectors. Such an endeavour aims at increasing programmer productivity and developing quality software programs. Predominantly, proposed studies in these areas are based on the use of fMRI, [35], Functional Near-Infrared Spectroscopy (fNRIS) [36] and EEG [15, 37–41]. The latter, as already mentioned above, will be the major focus in this study and, therefore, the main related topic of this state of the art.

Regarding software engineering, in 2017, a study by Scalabrino et al. [5] with 46 participants, focused on the analysis of correlations between code understanding of the programmers and the different complexity metrics used in software engineering. To this purpose, they considered and explored 121 existing and new metrics

(developed by the authors), concerning code, documentation and developer related metrics. The authors showed, through the report's evaluation of the participants, that none of the existing metrics are able to measure the code understandability. In a recent study [42], in 2019, published by the same authors, they increased the number of participants from 46 to 63, yet the conclusions remained the same.

Given the impact of the programmer's emotions during the software tasks and vice versa [14, 43, 44], it is interesting to investigate which features are most commonly being used in emotion recognition through the analysis of EEG. In 2017, a survey was carried out by Alarcão et al. [20] on the recognition of emotions using EEG. In this study they reviewed articles from 2009 to 2016 and ended up with a subset of 99 quality articles selected according to Brouwer's recommendations [45], who investigated the most common pitfalls and how to avoid them during analysis of cognitive and affective states. In this survey, as expected, the authors found that from the selected papers, the most used features (in 89.4% of the studies) were related to the five brain waves frequency bands (Delta, Theta, Alpha, Beta and Gamma), typically captured by the Power Spectral Density (PSD). Regarding the classifiers used in the reviewed studies, the authors found that the most used one (in 59% of the studies) was the Support Vector Machines (SVM). Another important aspect noted by authors, was that the most dominant emotion-related activated brain regions, reported in most articles, were the frontal and parietal regions. Table 3.1 summarizes the findings reported by Alarcão et al [20] regarding feature extraction and classification.

Table 3.1: Most common features and classifiers used in Emotion Recognition studies using EEG signals, from the most frequent (1) to the less frequent (6), according to the review work done by Alcarão et al. [20].

Most Frequent Features	Absolute/Relative Power of brain wave bands (1)	High Order Crossings (4)
	Statistical measures (2)	Fractal dimension (5)
	Entropy measures (3)	Assymetry Index (6)
Most Frequent Classifiers	Support Vector Machines (1)	Quadractic Discriminant Analysis (4)
	k-Nearest Neighbors (2)	Naive Bayes (5)
	Linear Discriminant Analysis (3)	Multi-Layer Percepton Back Propagation (6)

Several research studies reporting analysis of task engagement and mental workload through the acquisition of EEG have been published in the last twenty years and in an extensive range of fields [46–48]. In many of them, strong correlations were found between the states of the subjects and the brainwave frequency bands. The findings pointed mainly to the strength of Theta and Alpha waves to represent these mental states [47–50].

In 2014, Fritz et al. [37] conducted an exploratory study using Eye-tracker, EEG and Electrodermal Activity (EDA) sensors in 15 properly selected software programmers, to evaluate the difficulty felt during eight different tasks of code comprehension in C#, as well as the psycho-physiological features associated with it. Regarding the EEG, the authors used the one channel off-the-shelf NeuroSky MindBand EEG sensor that can generate two signals, "Attention" and "Meditation" [51]. Mean, standard deviation, maximum and minimum values were computed from the two signals, which were considered to capture certain states of consciousness. From the EEG signal, the PSD of the frequency bands (Delta, Theta, Alpha, Beta, Gamma) was computed as well as the ratios between each of the frequency bands. The authors also extracted the rate of eye blinks per minute from the EEG, which according to Brookings et al. [52], is expected to decrease for more difficult tasks. Afterwards, using Naive Bayes classifier, the authors performed three analysis of the prediction of difficulty (easy or difficult): by participant, by task and by participant-task. For the situation of prediction by participant-task, the authors achieved a F-measure of only 56.73% using EEG information, which increased up to 67.71% by fusing the information from the three sensors. Although being the first automated approach for assessment of mental workload during code comprehension using psycho-physiological information, and yet promising results have been achieved in this study, further work regarding this topic can be explored using only EEG, and also including an higher number of participants.

In 2014, Igor Crk and Timotthy Kluthe [38] performed a study reporting a binary qualification of a programmer's expertise through code comprehension. They recorded EEG signals from 14 electrodes in 34 participants during programming

tasks in Java. Through the analysis of Theta and Alpha waves and the reports filled by the subjects for the different tasks, the authors found evidence that allows to distinguish subjects from different levels of expertise with an accuracy ranging from 53% to 63 %, depending on the task. They also noted the power of the Alpha band in distinguishing between states of rest and mental effort through differences in the average of the PSD of each one. The same authors published a similar work in 2016, also regarding code comprehension and expertise level, using EEG, but this time focusing only on the Alpha wave band. More specifically, they analysed the Individual Alpha Frequency (IAF) [53], which takes into account the subjects' ages and brain maturation, for a better analysis of inter-subjects. The authors found correlations between the IAF (increase) and the correct answers given by the participants to the reports related to each one of the codes.

In 2016, Lee et. al [39] recorded EEG signals from 18 subjects while they were performing code comprehension tasks, in Java, to analyse neurophysiological processes occurring during tasks and the possibility to distinguish between expert programmers from beginners. From the signals recorded using 13 EEG electrodes, the authors explored the relative power spectrum of the five brain frequency bands per electrode and per region of the brain. The results showed that high frequencies are dominant features, namely Beta and Gamma waves, and the most significant channels were from frontal and parietal regions. In addition, the authors also found that in expert programmers there is more activation in F3 and P8 channels than in the novice programmers. Nevertheless, the authors are aware of the limitations of such results, since there are few studies in this area and there is the need for more studies for comparison of results and conclusions. Lastly, they also point out the problem of generalization power, since only 18 subjects participated in the study.

In 2017, the authors Yeh et al. [40] conducted a study focusing on brain activity during 12 C/C++ code comprehension tasks with only two difficulty levels. EEG signals were recorded using electrodes placed in frontal region from eight subjects. Subsequently, the authors calculated the power of the Theta and Alpha bands of the signals, for statistical analysis. Using these two frequency bands, the authors

obtained statistical significant differences in all eight electrodes used, at 0.05 significance level. However, within the same type of difficulty, there were no statistically significant differences. Moreover, given the low number of participants, it becomes difficult to generalize these results.

Also in the same year of 2017, Lee et al. [41] conducted a study with 38 participants, 20 novices and 18 expert programmers, with the aim of predicting their expertise level (novice or expert) or the difficulty (easy/difficult) of the comprehension tasks they performed, through the use of EEG and eye tracking data. Using the SVM classifier and a 10-fold Cross Validation, the authors obtained better results using the data from both sensors than using it separately. Thus, the best result obtained for the prediction study of the difficulty of the task was a F-Measure of 66.6%, while for the study of prediction of the participants' level of expertise, a F-Measure of 97% was achieved. Although striking results have been obtained from the analysis of a considerable number of participants, the authors did not specify what features were extracted from the recorded EEG, keeping unknown the EEG features that led to these surprising results, more precisely the results on the prediction of the level of expertise.

Finally, the most recent study on this topic was conducted by Kosti et al. [15] in 2018 and had the purpose of investigating the brain activity during two different programming tasks: comprehension and inspection of syntax errors in C code. Using only 14 electrodes, they recorded EEG signal from 10 participants, and subsequently, the authors performed two types of analysis, one using the spectral power of each electrode alone, and another inspecting functional connectivity. The authors found that Theta, Beta and Gamma waves during comprehension tasks correlates with cognitive effort, with stronger correlations observed for higher frequency waves. Another surprising finding was the detection of much higher activation of Theta and Beta waves in comprehension tasks than in tasks of inspection of syntax errors. Authors argued that this was due to the fact of inspection tasks being considered easier tasks that do not require as much effort in the imagination of the program output as it is demanded for comprehension tasks. Despite these encouraging results, it is

necessary to increase the number of samples for a stronger generalization, which is also pointed out by the authors in the limitations of the study.

With the same aim of assessing code complexity, there is already being done studies exploring other biosignals for this purpose inserted on the project BASE. Recently in 2019, a study included on the master's dissertation of Gonçalo Duarte [54], who also performed the data acquisition followed in this study, showed that, by using Heart Rate Variability (HRV) features, it was only possible to distinguish two levels of code complexity, even though there were three complexity levels according to software complexity metrics. The Eye tracking and Pupillography data is also being explored, for the same purpose [55].

In sum, the Table 3.2 shows a brief summary of information regarding the feature types, brain locations and classifiers that may contribute to achieve the objectives of this work, according to the related works.

Table 3.2: Summary of significant features types, classifiers and brain locations reported by related studies.

Summary Remarks					
Features		**Locations**		**Classifiers**	
Brain frequency bands derived features (specifically from Theta, Alpha, Beta and Gamma)	Statistical features	Frontal Lobe	Parietal Lobe	Support Vector Machines	Naive Bayes

4

Data Acquisition Protocol

4.1 Dataset Description

As result of the work undertaken by Gonçalo Duarte [54] (inserted in the scope of the ongoing project BASE), 30 subjects were selected to participate in the study' acquisitions. This group of volunteers was selected after a series of interviews of students, professors and professional software developers, with experience in *Java* programming language. Specifically, from the 30 participants, 24 were male and 6 female, with ages ranging from 19 to 42, and average age of 24 years old. Also through the interview and based on years of experience in *Java* programming or in the number of lines of code programmed in *Java* in the last months or years, the participants were classified into three levels of proficiency: intermediate, advanced and expert. In sum, there were 13 intermediate, 12 advanced and 5 expert participants.

The data collection was authorized by all the participants involved and the written consent was approved by the Ethics Committee of the Faculty of Medicine of the University of Coimbra, in accordance with the Declaration of Helsinki.

4.2 Protocol Description

Subjects were submitted to three different trials of code comprehension tasks using three code snippets in JAVA, having each code snippet different complexity levels. Each trial consisted on a control task of text reading in natural language (60 seconds maximum) and a task of code comprehension of a certain complexity level (10

minutes maximum). Before and after each task, an empty screen with a cross in the middle was shown for 30 seconds to the subject, acting as a baseline interval for the next task.

After each trial, the subjects answered two questionnaires. With the purpose of guaranteeing that the subject was concentrated during the trials, in the first questionnaire the subject had to explain in general the algorithm of the finished trial. On the second one, the subject filled a survey based on NASA-TLX (Task-Load Index)[1] survey [56] with four questions, rating it from 1 to 6, in order to assess the subjective mental effort, task fulfilment, pressure over time and frustration of the subject while doing the code comprehension task.

This acquisition protocol is represented in Fig. 4.1, with an estimated experience time of less than two hours for each subject.

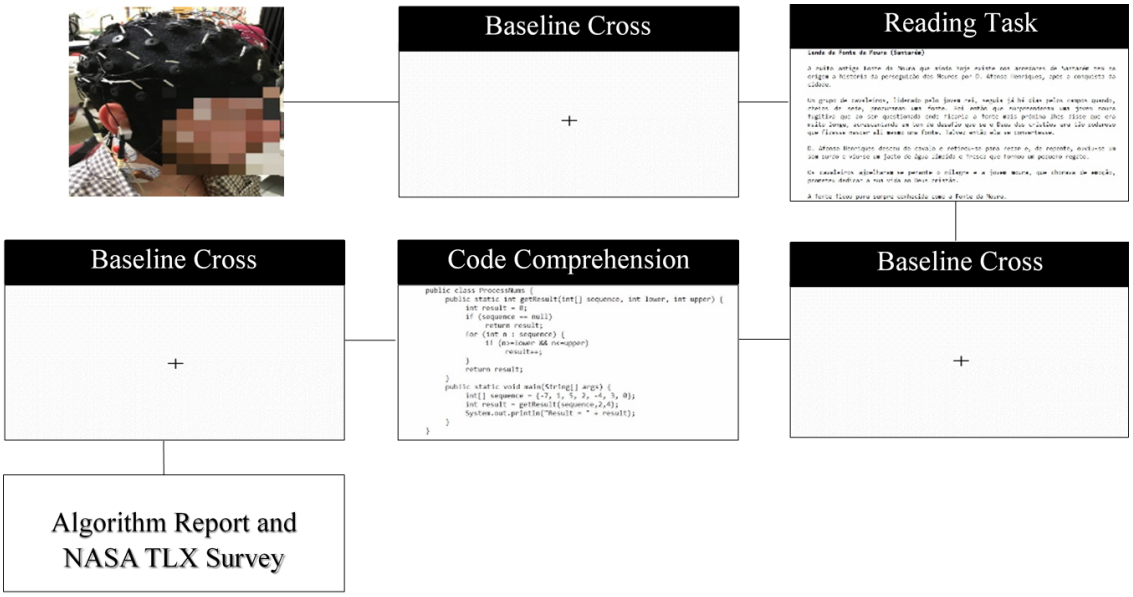


Figure 4.1: Diagram of one trial procedure with an empty screen with a fixed cross, a reading task as reference for analysis and a code comprehension task. The procedure was repeated for the three different trials (differing in the code complexity).

Regarding the order of the trials, it was performed according to an increasing degree of complexity of the code. In Fig. 4.2, it is possible to observe the three types of code complexity evaluation, as a percentage of the maximum value, for each one of the five complexity metrics used (Table 2.1 in the Background section 2.1). According

[1]NASA-TLX site: https://humansystems.arc.nasa.gov/groups/TLX/

to the metrics, differences between the three different codes complexity are visible, being the major difference noticed for the McCabe Cyclomatic Complexity metric, one of the most popular and widely used in software engineering [57]. The three different complexity codes can be seen in the Figures A.1, A.2 and A.3, respectively, in the experimental protocol appendix A.

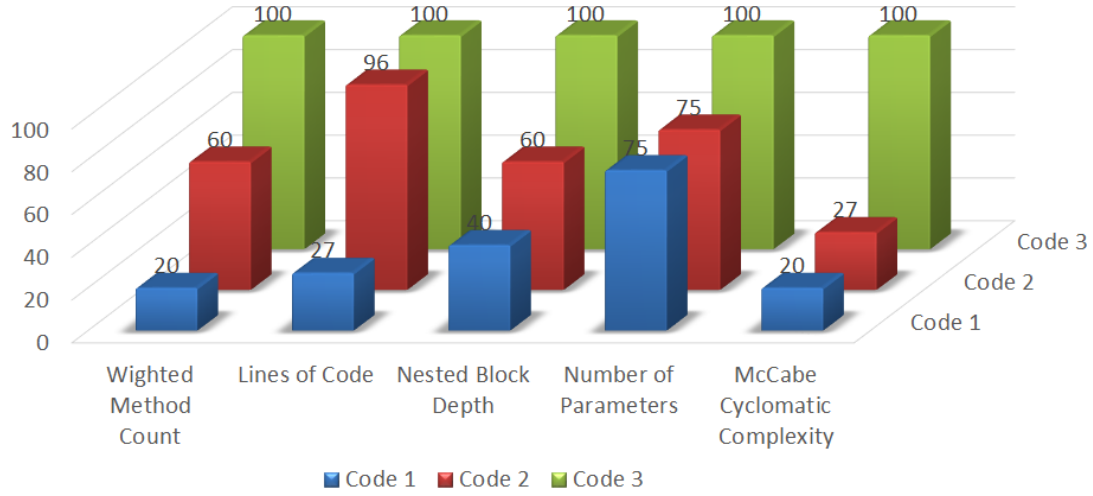


Figure 4.2: Complexity level (in % of the maximum) of each code tasks according to each one of the five software complexity metrics used.

4.3 EEG Acquisition Setup

For data acquisition, the experimental setup comprised a set of varied sensors, ranging from Electroencephalography (EEG), Electrocardiography (ECG), Impedance Cardiography (ICG), Photoplethysmography (PPG), Electrodermal Activity (EDA) to Eye tracking with Pupillography.

Concerning the main focus of this work, the EEG signals were acquired using the Neuroscan SynAmps 2 amplifier, from Compumedics, with a sampling frequency of 1000 Hz and 64 channels placed according to the International 10-10 system. Neuroscan, besides including the channels M1, M2, CB1 and CB2, also included four integrated bipolar leads for EMG, ECG, and the ocular channels, VEOG and HEOG.

In the acquisition setup designed to record the EEG data, the EEG quick-cap was

connected to the amplifier through the EEG HeadBox, which consequently was connected to the acquisition computer that controls the whole experience, communicating synchronously with all sensors and to the second computer, used to present the stimuli to the participant.

During the data acquisition of EEG from one participant, there were several electrodes that did not work, and besides they were located in frontal and parietal lobes, which, according to related studies, are relevant locations for the analysis. For that reason, the EEG data of this subject was not considered for the EEG analysis. Later, three more subjects were removed since Eye tracking data suggested that they did not focus during some of the trials. Thus, the initial dataset was reduced to 26 subjects for analysis.

5

Methods

This section includes four sub-sections describing the steps and methods taken from the preprocessing of the EEG data to the analysis and interpretation of postprocessed neural signals (see Figure 5.1). It also includes a sub-section of statistical analysis methods and a last sub-section describing the different studies carried.

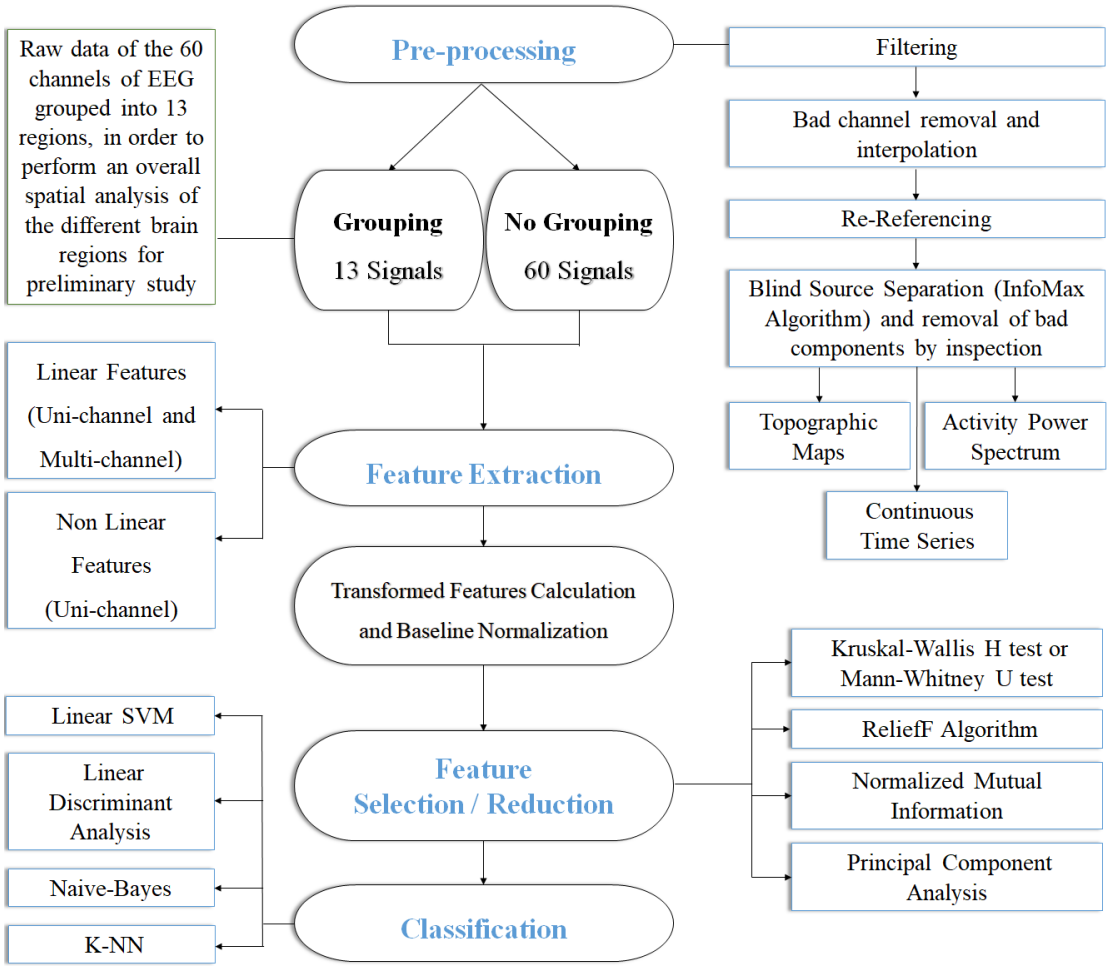


Figure 5.1: Overview of the pipeline with the steps and methods performed.

5.1 Preprocessing

The EEG preprocessing starts with the filtering of raw EEG data, which is then followed by inspection of the data for bad channels interpolation, re-referencing and finishes with a Blind Source Separation (BSS), applied for further artifacts removal. This was performed using the open source toolbox EEGLAB [58], one of the most widely used software for preprocessing/analysis of EEG data [59]. Before proceeding to data filtering step, it was added the EEG channels locations information to the data in the software, and it was also removed the eight channels (M1, M2, CB1, CB2, ECG, EMG, VEOG, HEOG).

5.1.1 Filtering

In this stage, a Finite Impulse Response (FIR) filter with Hamming sinc window was applied to the EEG recordings using the function *firfilt* from EEGLAB. The filter's group delay was left-shifted, taking into account that the group delay is an integer number of samples, not needing more computation, like the cases of Infinite Impulse Response (IIR) filters, to ensure zero-phase distortion [60].

A windowed sinc filter is considered an approximated ideal filter, since, as the name suggests, its impulse is based on sinc-function in the time domain, approximating the frequency response to a rectangular magnitude response [60]. The windowing to the impulse response is used in order to reduce the passband and stopband ripple, being the Hamming the chosen window for this study. The number of points M used in the filter kernel was assigned automatically by default of the function, according to the quotient between the normalized transition width of the Hamming window (ΔF=3.3) and the transition band width defined by the passband edges.

The order of steps involved, with the respective description, are described bellow:

- High-pass Filter: with a cut-off frequency at 1 Hz, was applied in order to remove DC component and slow frequency drifts (depicted in Figure 5.2). This cut-off value was considered since it was proven that, when using Independent Component Analysis (ICA) for blind source separation, this procedure

produces better results in terms of signal-to-noise ratio (SNR) and in better dipole-like brain sources ICA components [61].

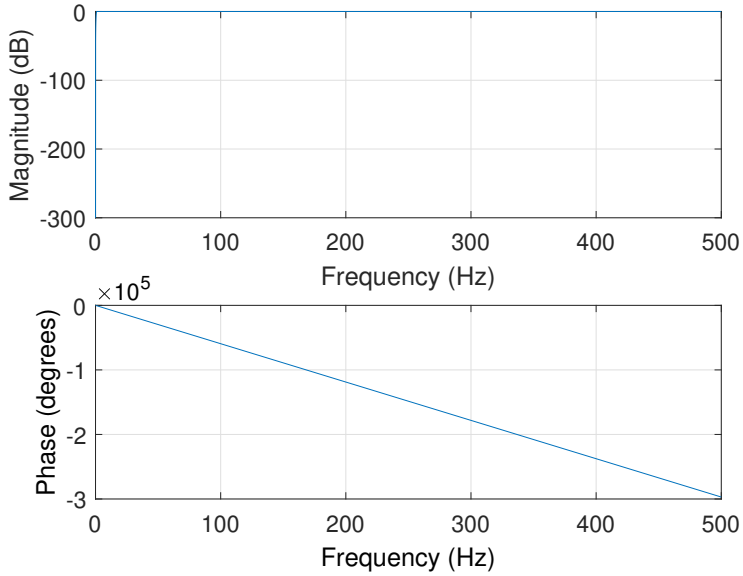


Figure 5.2: Frequency Response of the High-pass Filter with a cutt-off frequency of 1 Hz.

- Low-pass Filter: with a cut-off frequency of 90 Hz since it is considered the upper limit of the frequency band of interest for the analysis (see Figure 5.3).

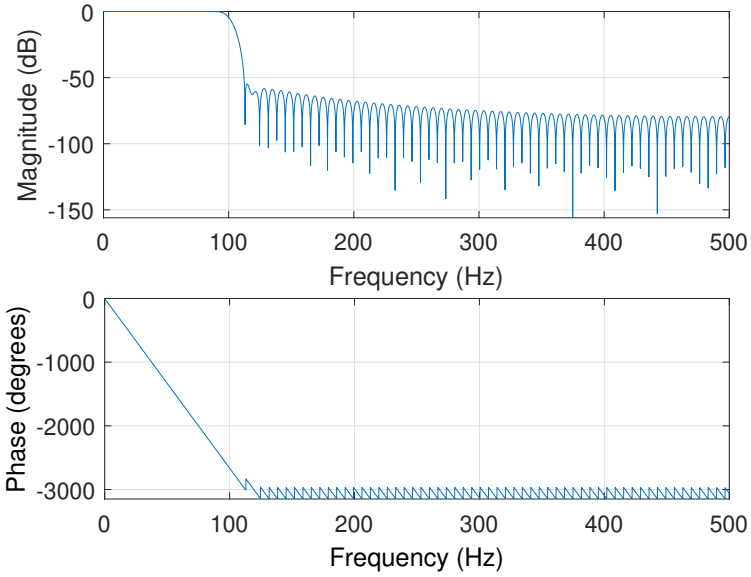


Figure 5.3: Frequency Response of the Low-pass Filter with a cut-off frequency of 90 Hz

- Notch Filter: applied in order to remove the powerline interference at 50 Hz (see Figure 5.4).

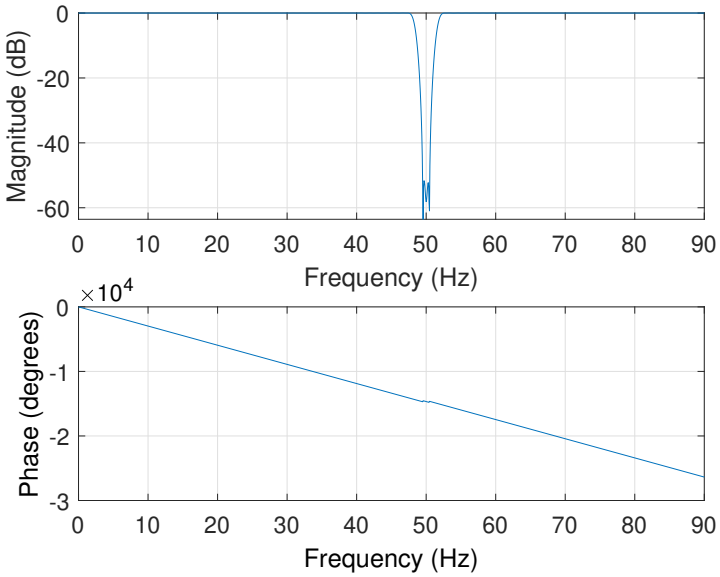


Figure 5.4: Frequency Response of the Notch Filter applied at 50 Hz.

An example of some of the results of the High-pass, Low-pass and Notch filters applied to signals obtained from the 60 EEG channels is depicted in Figure 5.5. It is also visible the presence of artifacts that need to be removed before proceeding for data analysis (eye blinks occurring around 56, 58 and 60 seconds).

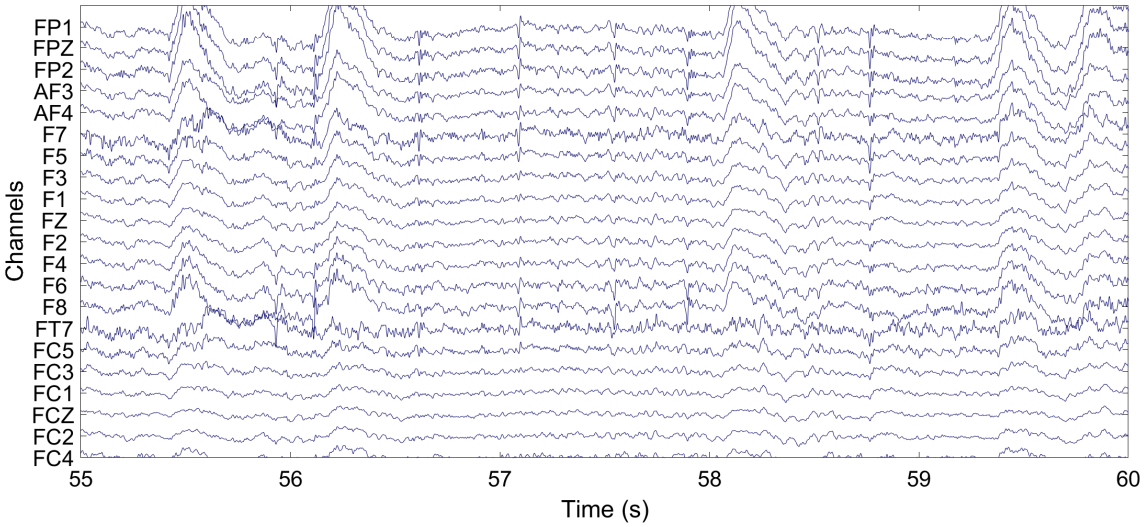


Figure 5.5: Overview of the filtered EEG data from 20 of the 60 channels. Scale: 35 microvolts(μV).

5.1.2 Channels Spatial Interpolation

Although the impedance and functionality of each electrode is checked before each acquisition, it is possible that electrode malfunctioning occurs until the end of the trial. It can be the result of the participant's movement that might lead to electrode detachment from the scalp, for a moment or until the end of the acquisition. When this event is not corrected, it might have a considerable impact on the remaining analysis. Therefore, it is necessary to perform visual inspection of the EEG data, in the time domain, and proceed to the removal and replacement of these flat or noisy channels, by interpolated signals using the remaining channels' information. Once this step is completed, the data can be re-referenced.

Regarding the interpolation step, it was performed with the *eeg_interp* function of EEGLAB, using the spherical spline interpolation algorithm from Perrin et al. [62].

In the Fig. 5.6, by maximizing the scale for better visualization of data, it is possible to observe an example of the presence of a bad channel, in this case the C3 channel, marked with a red rectangle, that will be removed and interpolated.

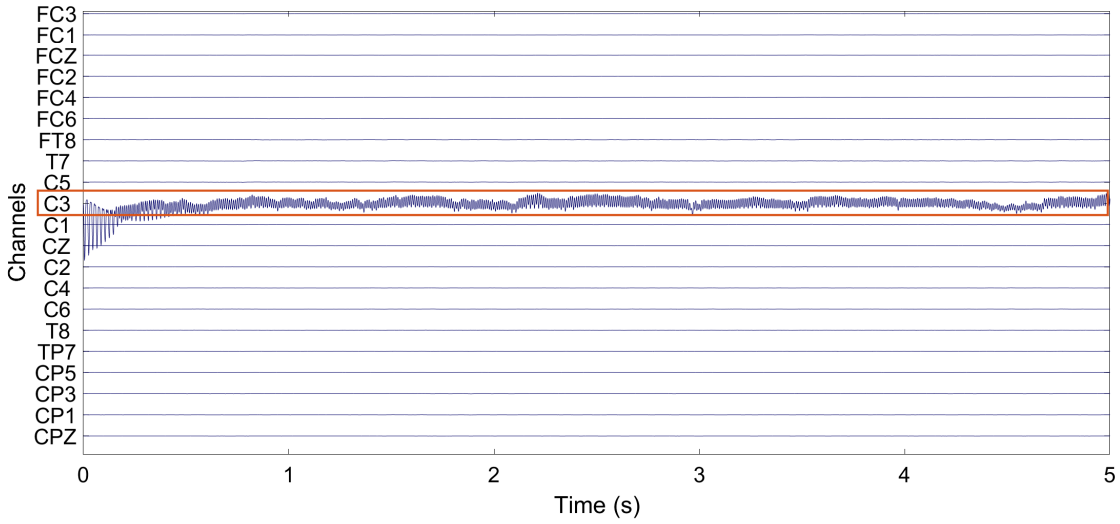


Figure 5.6: Visual inspection of EEG bad channels for removal and interpolation. In this case of this trial from a participant, the C3 channel was removed and interpolated. Scale: $5000\mu V$.

5.1.3 Re-referencing

There are several methods of doing the re-reference, instead of using the reference electrode chosen during the acquisition, and it can be considered any other electrode as the new common reference. However, the new reference should be carefully chosen, since by choosing an exact electrode any activity in this electrode will be reflected in all other electrodes [63]. Furthermore, if the selected electrode is capturing brain activity, re-referencing to it may lead to loss of information.

Some of the approaches chosen for re-reference can be left/right mastoid reference, averaged mastoids reference, nose reference and the average reference of all channels [64]. Regarding nose reference, it is rarely used in literature. Concerning mastoid reference, although its location is considered further from the brain, it still remains close enough to it, so that there is the hypothesis of the mastoid reference containing some neural activity [63]. For that reason, this might not be the best approach in the re-referencing. The same happens for averaged mastoids reference, despite that it can provide better results since there is less lateralization bias [63, 65].

In this study, for the re-reference, despite not having the best high density of electrodes, to achieve better results [63, 66], it was used the average reference, which is performed, as the name suggest, by doing the average of all 60 channels and the linear transformation of the data. This was performed using the *pop_reref* function of EEGLAB. The importance of this step is not only to eliminate some noise common to all channels, but also because of the fact that the reference electrode should not be around regions of interest with important brain activity for the analysis [63]. So, in this case, since the most activated regions during code comprehension are also being investigated, it is important to change the original reference, which is between Cz and Pz electrodes, for a proper spatial analysis.

5.1.4 Blind Source Separation

Despite the various preprocessing steps already taken, there are still many artifacts to remove from the EEG signals, such as ocular (eye blinks and eye movement),

muscle and cardiac artifacts. Therefore, Independent Component Analysis (ICA) is applied for Blind Source Separation (BSS) in order to accomplish artifact removal. BSS consists in the decomposition of a matrix of signals into the different recovered source estimation signals that produced the signals captured by the sensors [67]. In other words, it consists in going in the inverse direction of the scheme presented on Fig. 5.7 towards finding the unmixing matrix. In this case, the mixture of the sources is the EEG signal captured by the different EEG electrodes. This way, ICA works by searching for a linear transformation that maximizes the statistical independence between the output components [68].

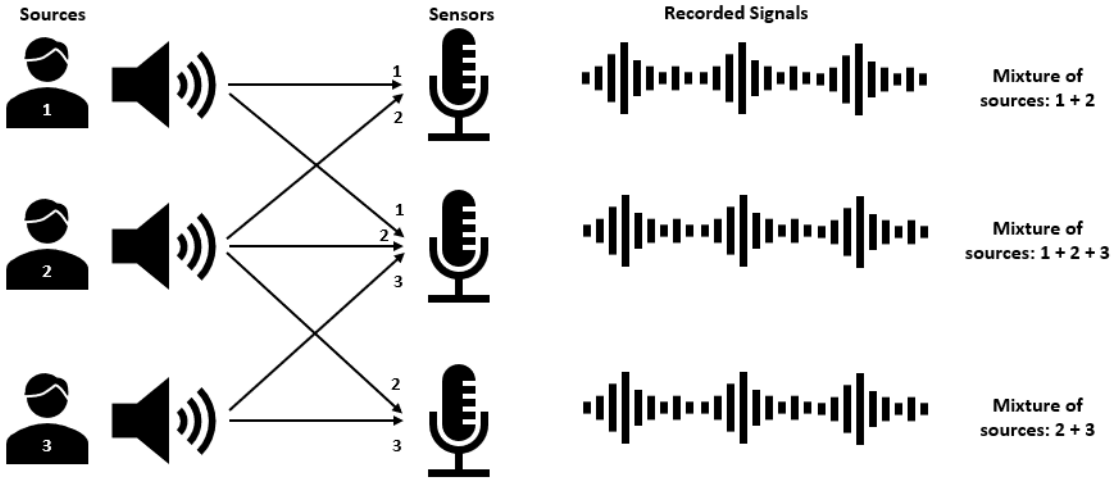


Figure 5.7: Scheme of the formation of the recorded signals and how the BSS works by going in the inverse direction (right to left) of the scheme, in order to compute the estimation of the original signal sources.

When preparing the data to run ICA, large muscular activity or other strange events (non stationary data) were rejected manually from the data, in order to improve the ICA decomposition quality [65, 69].

In a recent study carried by Dharmaprani et al [70], analysing the different ICA algorithms and their performance discriminating between EEG and artifacts components, the authors found that the best ICA approach was the FastICA [71] or Infomax [72] algorithm. From these two, an extended version of the Infomax algorithm was chosen to be used in the current study, since it was showed in another recent work, regarding the State of the Art about EEG artifact removal, that this algorithm had better performance in removing ocular and myogenic artifacts [73].

The Infomax algorithm consists in minimizing the Mutual Information between the components [73], maximizing the independence between them. Years later, the extended algorithm was introduced by Lee at el. [74], with the use of negentropy maximization projection, making it possible to separate mixed signals with different source distributions (sub- and super-Gaussian distributions).

Thus, by using the function *runica* from EEGLAB, it was possible to apply the Extended Infomax Algorithm. In this function there are two main conditions to stop ICA computation: when the differences in ICA weights are less than 1×10^{-6} between consecutive runs or when it reaches 512 interactions. The latter condition was changed to 2000 to ensure that the first condition was dominant, but at the same time guarantees that if it does not converge, it stops.

After computing the ICA components, the components associated with artifacts were selected, and subsequently removed, by inspection of the component:

- Topographic map;
- Activity power spectrum;
- Continuous time course.

In the following figures (Fig.5.8 - 5.12) it will be demonstrated and described examples of typical ICA components obtained.

A neural component can be recognized (i) by the topographic map of the dipole type of the ICA weights, (ii) by the decrease of the power spectrum magnitude with the increase of the frequency and (iii) by the typical peaks at certain frequencies on the power spectrum of the component (mainly around 10 Hz) [75]. An example of brain-related component can be observed in Fig. 5.8.

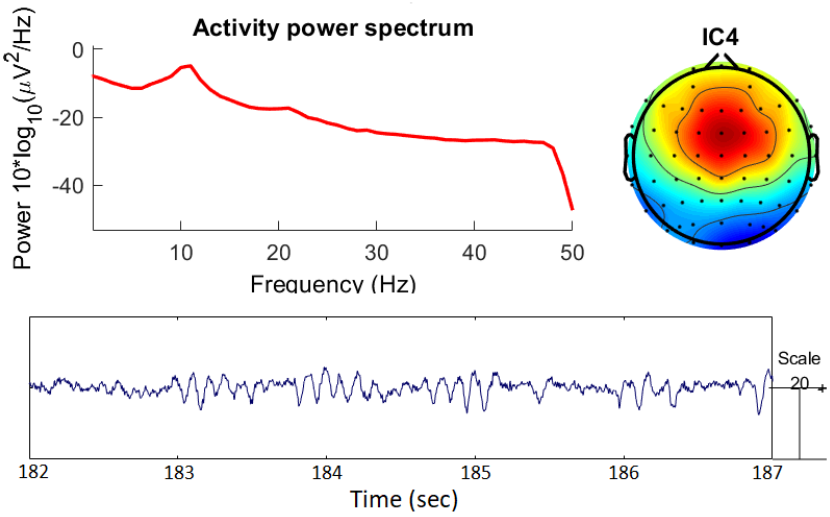


Figure 5.8: Example of the topographic map, activity power spectrum and continuous time course of a neural activity ICA component.

On the other hand, regarding artifact components, in Fig. 5.9, it can be easily recognized an eye blink artifact component by visualization of the component topographic map with maximum ICA component weights in the frontal region, near to the eyes [69]. Another easy way to detect this artifact is by noticing the large amplitude of eye blinking in the component's time course.

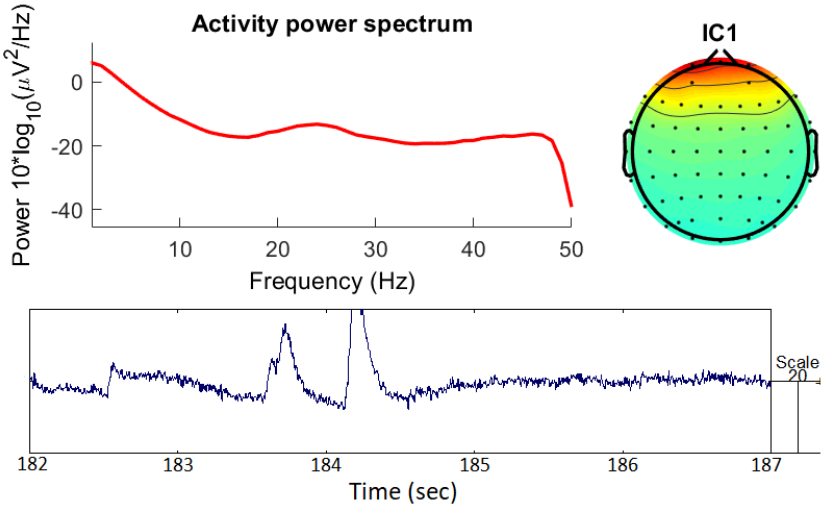


Figure 5.9: Example of the topograpich map, activity power spectrum and continuous time course of ICA component with eye blinking artifacts.

In the Fig. 5.10, it is possible to observe another example of an ICA component removed that presented ocular artifacts, i.e., the saccades and microsaccades (seen

in the sec 184-185 of component's time course), particularly, in the lateral frontal regions.

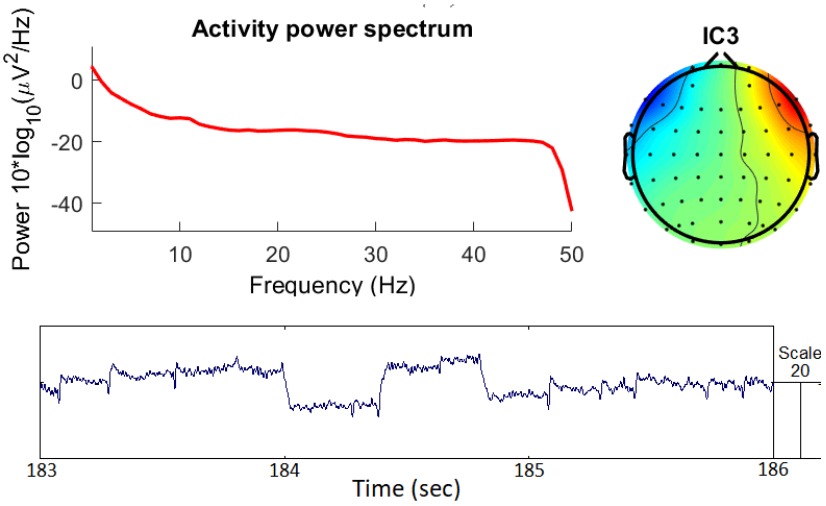


Figure 5.10: Example of the topograpich map, activity power spectrum and continuous time course of ICA component with saccades artifacts.

Another example of a removed component, containing cardiac artifact, is represented in the Fig. 5.11, where it can be easily recognized in the component's time course.

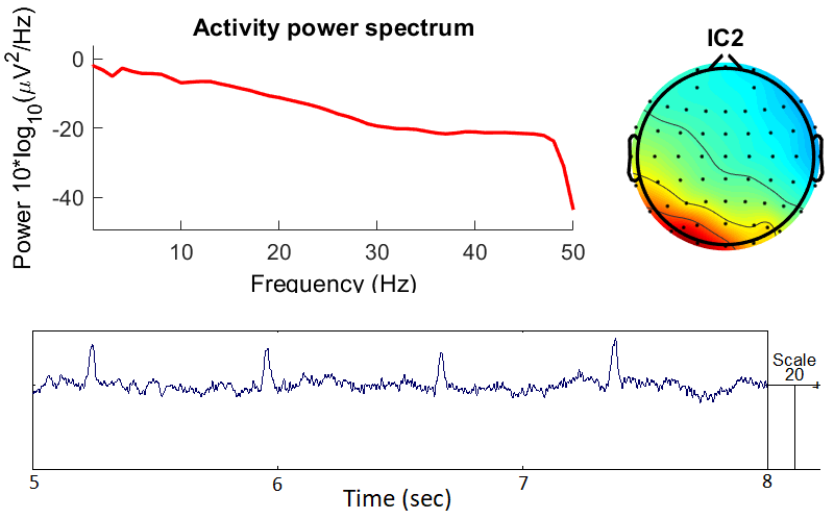


Figure 5.11: Example of the topograpich map, activity power spectrum and continuous time course of a cardiac artifact ICA component.

Figure 5.12 depicts an example of another common type of artifact, induced by involuntary muscle movement and typically detected in Electromyogram (EMG). The component contains muscle activity, visible by its local weight and the burst

in the activity power spectrum from 30 Hz to 40 Hz in comparison to the power spectrum in the lower frequencies [69].

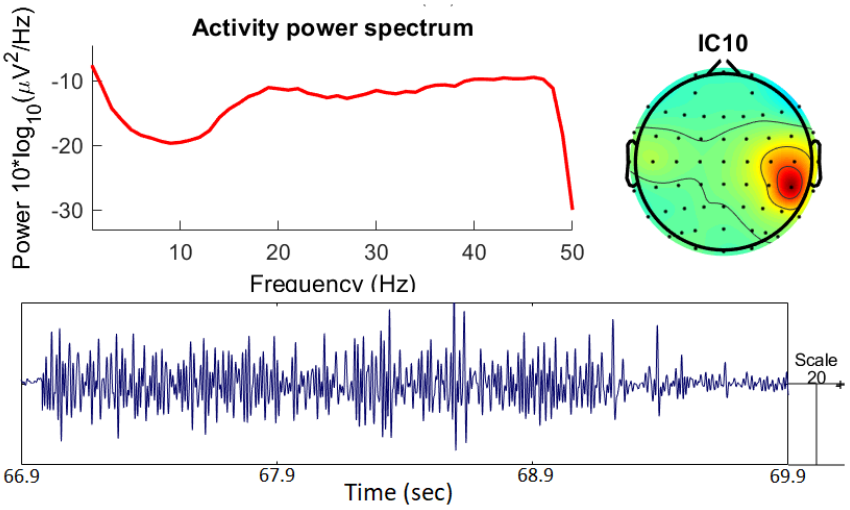


Figure 5.12: Example of the topograpich map, activity power spectrum and continuous time course of ICA component with muscle activity.

5.2 Feature Extraction

After preprocessing the EEG data, two types of analysis were performed (further detail in Section 5.6): one using only 13 signals (grouping of EEG signals per region by their average - considering the scalp division in 13 regions), and another one using all 60 EEG signals. This was done with the purpose of investigating the discriminative power of certain groups of features (linear and non-linear), and the computational cost that would imply if it were an analysis using all 60 channel signals.

For feature extraction, it was discussed and thought three possible different approaches: (i) "Hand Craft", by exploring and analyse only features that were truly known to be related with mental workload during code comprehension tasks; (ii) "Ad-hoc", by extracting several types of features and by performing afterwards feature selection based on the data; and (iii) "Learning" methods which originate own features representative of the data, e.g., the Autoencoders.

The features explored in this work were chosen based on the knowledge of the most

commonly reported features in emotion recognition and mental workload studies. That said, it consists basically on combining the first two interpretative approaches mentioned before (i) and (ii), allowing to explore already known important features but at the same time to investigate new possible discriminant features for distinguish different complexity levels of code comprehension tasks. The "Learning" approach was not considered, since it needs a large amount of data for a successful analysis.

5.2.1 Feature Description

This section will try to briefly address the features explored, which can be divided into two main groups: linear and non-linear features.

1. Linear features

   Linear features are computed using methods that extract amplitude and frequency information from a single EEG electrode (uni-channel) or from multiple electrodes (multi-channel). These features were divided into three groups: Uni-channel Time Domain, Uni-channel Frequency Domain and Multi-channel features.

   (a) Uni-channel Time Domain features

   - Statistical features

     This type of features, very commonly used in EEG analysis [20], characterize amplitude changes and distribution of the signal over time. The first, third and fourth central moment and the standard deviation [76], extracted from time domain EEG, capture information about the signal's amplitude distribution:

     – Mean (first central moment) of raw and normalized signal: measure of the central tendency

     – Skewness (third central moment): measure of the asymmetry

     – Kurtosis (fourth central moment): a measure of the tailedness

     – Standard deviation: measure of the dispersion

The second statistical moment, the variance, was not extracted here, since it is already included in the following Hjorth parameters.

- Hjorth Parameters

  In 1970, in order to develop a quantitative approach to describe the EEG signals, Hjorth [77] derived a set of three parameters, shown below, which are widely used nowadays [78] to describe the signal in terms of amplitude, time scale and complexity, respectively:

  – Activity - measures the variance of the signal's amplitude.

$$\text{Activity} = \sigma^2(S(t)) \tag{5.1}$$

  – Mobility - measures the variance of the slope in relation to the variance of the signal's amplitude.

$$\text{Mobility} = \frac{Activity(S'(t))}{Activity(S(t))} \tag{5.2}$$

  – Complexity - measures the deviation of the signal from the pure sine shape.

$$\text{Complexity} = \frac{Mobility(S'(t))}{Mobility(S(t))} \tag{5.3}$$

  being σ the standard deviation, S(t) the signal in the window analysed and S'(t) the first derivative of it.

(b) Uni-channel Frequency Domain features

These are the most popular type of features used in EEG studies. The Power Spectrum Density (PSD) was calculated by squaring the absolute value of the Fast Fourier Transform (*fft* function in MATLAB) of the signal. Then, from this PSD frequency series, several features were extracted aiming at analysing specific frequency bands. Among these features it is expected that Theta, Alpha, Beta and Gamma bands stand out as result of the increase of mental workload [15,39,40], either individually or when

combined.

- Power features - obtained by computing the area under the PSD curve.
  - Total Power - corresponding to the total area of the frequencies of interest.
  - Absolute and Relative Power of frequency bands - Delta(0-4 Hz), Theta (4-8 Hz), Alpha (8-13 Hz), Beta (13-30Hz) and Gamma (30-90 Hz). The latter since it has a wide range, was divided into three sub-bands: Low Gamma (30-50 Hz), Medium Gamma (50-70 Hz), High Gamma (70-90 Hz).
  - Power ratios between bands (combinations of two between all the seven bands) - by including these ratios, it might be possible to improve the analysis by reducing the variability of the PSD between different subjects, as Fritz et al. suggested on their work [37].
  - Task engagement indexes - first reported by Pope et al. (1995), ratios using Theta, Alpha and Beta power bands, are being widely used for two possible representative indexes of the participants' engagement during tasks [79–81]:

$$\text{Index 1} = \frac{\beta Power}{\theta Power + \alpha Power} \tag{5.4}$$

$$\text{Index 2} = \frac{\theta Power}{\beta Power + \alpha Power} \tag{5.5}$$

- Average frequency - as the name suggest, the estimation of the mean frequency of the PSD of the signal, in order to explore what are the most predominant frequencies (low or high) in the different tasks.
- Alpha peak frequency - the frequency corresponding to the maximum

peak in the Alpha band has been shown to be able to differentiate mental states [82], with some studies suggesting that is positively correlated with cognitive performance [83].

(c) Multi-channel features

- Differential Asymmetry and Rational Asymmetry - the difference and quotient of the power of the frequency bands between pairs of electrodes (left-right brain hemispheres) has been extensively explored to find relations between brain locations [78, 84].
- Cognitive load index "Brainbeat" - was showed by Holm et al. [85] to be a powerful feature in estimating cognitive load in tasks through the ratio between powers of two frequency bands from frontal and parietal brain location:

$$\text{"Brainbeat" Index} = \frac{\theta Power(Fz)}{\alpha Power(Pz)} \tag{5.6}$$

2. Non-linear features

Given the complexity of the EEG signals, linear features can be limited by providing only part of the information about neural activity, because there may also be processes resulting from non-linearities [86]. Thus, it is of great importance to add to the analysis some of the most commonly used non-linear features, allowing a broader and complete analysis of the brain functions, along with the linear features. However, an important aspect to be retained is that these features are associated with a higher computational cost than linear features.

In this group of features, which will be briefly described below, only features of the time domain were extracted from single EEG electrodes.

- Phase Space Reconstructed features

  One of the most common methods used in non-linear analysis consists in the reconstruction of a multidimensional dynamical system (also known

as Phase Space Reconstruction) by taking different time delays (τ) of the input time domain signal [86]. Then, from the dynamics of the underlying m-dimensional system reconstructed (being m the so-called embedding dimension necessary to describe the behaviour of the signal) it is possible to extract three non-linear features very commonly used:

- Correlation Dimension - by measuring of the phase space dimension occupied by the data points, it gives information about the complexity in the dynamics of the system [87], in the embedding dimension m and using an embedding delay τ.
- Largest Lyapunov Exponent - measures the exponential divergence of close trajectories in the phase space, providing information about the predictability of the dynamics of the system [88], in the embedding dimension m and using an embedding delay τ.
- Approximate Entropy - measure of randomness that gives information about the disorder of the dynamics of the system [89], in the embedding dimension m and using an embedding delay τ.

It should be noted that, for the computation of the above features, the selection of the embbeding dimenson m and the time delay τ was obtained using *phaseSpaceReconstruction* function from MATLAB.

Another feature, in spite of being more used in other biosignals [90, 91] and not so much explored in EEG signals, was also investigated. The feature was extracted taking into account that during the phase space reconstruction step, the embedding matrix had already been computed for the embedding dimension m and the embedding delay τ, required for the present feature:

- Simplicity Measure S - it is a measure that can also provide information about the complexity of the dynamic system, being more insensitive to the presence of additive noise in the system compared to the Entropy [90]. It is calculated through the computation of the

entropy H of the normalized eigenvalues from the correlation matrix of the embedding matrix. Then, the feature takes the final form as:

$$S = \frac{1}{2^H} \tag{5.7}$$

- Other features

  In addition to the previous features, other non-linear features that do not require the calculation of the phase space reconstruction were also extracted. These features are briefly described as follows:

  - Fractal Dimension (FD) - another frequently used feature to measure the complexity of the EEG signal. A fractal can be seen as an object with fragmented geometric shape, with the fractal dimension defining the degree of self-similarity that the object has in its different scales of observation. For its calculation, it was used the Higuchi algorithm [92], since it has been showed in several studies that this method gives a better estimation value of FD (closer to the theoretical one) [93, 94]. This method consists on calculating the mean length of the curve for each of the k-sample sets. Then by plotting, in log-log scale, the length curve against k, the slope will express the FD. The optimal number for the maximum parameter k was searched through plotting FD value against different $kmax$. It was found that a good value would be $kmax$=100, since after this value the FD becomes stable [95].

  - Hurst Exponent (H_e) - is also used in fractal analysis and it gives a measure of long-term memory of a fractal time series, giving information on the its predictability [96]. A value of H_e=0.5 corresponds to a random time series, while values from 0-0.5 indicate a time series with long-term changes between high and low values. On the other hand, values from 0.5-1 indicate a long-term trend.

  - Multifractal Spectrum ($D(h)$) - it is another common feature used

in fractal analysis, and it allows for the quantification of how the strength and local regularity of a signal varies over time [97]. The local regularity is measured by the strength of the signal's singular behaviour at time t or around a point, through the *Hölder exponent* h [98]. From this spectrum, 11 different parameters were calculated in order to describe its form (see example of spectrum in Fig. 5.13.)

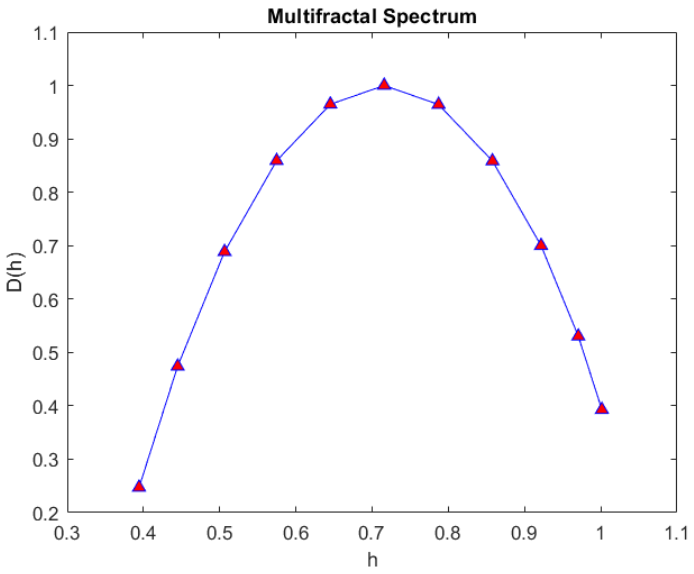


Figure 5.13: Example of multifractal spectrum obtained from a random 1-second window of EEG signal.

5.2.2 Selection of Extraction Window

Before proceeding to the extraction of features, the window size used to span the EEG signal over time and compute each value of a given feature was investigated. In EEG analysis, there is no rule of thumb to choose the window size for analysis. Different studies despite using the same sampling frequency, use different window size, varying from 1 second window to 20 second in different analysis [99–101].

In a preliminary study, a 5 second window with a step of 1 second (80% overlap) were used to extract EEG signal features. Subsequently, it was concluded that it would be interesting to investigate other window sizes, in order to find the best one for this study.

Thus, in order to choose the proper window size for the EEG analysis, it was thought

to be interesting to explore the dynamic (fast or slow) of the process, i.e., its stabilization time. As the input to the system that generates the EEG signals is totally unknown, Autoregressive (AR) models were considered useful for the task of choosing a window size given that they can predict an approximate output value based on the past outputs [102]. This way, using the least-squares method, and by minimizing the mean square error (MSE), it is possible to obtain the output coefficients of the AR model to attain the transfer function of the process [102].

The designing of an AR model requires the definition of the order of the model, P, which should be carefully investigated, since, as it is expected, the higher the number of coefficients of the model, the lower the error of the model becomes, but higher the complexity. One approach to find the optimal order is using the Levinson- Durbin Recursive algorithm and search for the minimum of total squared error (TSE) in function of the order P [102,103]. Other possible approach is by analysing the Partial Autocorrelation function (PACF) and search for the lag at which the PACF starts to reach zero, being that lag, the optimal order of the AR model [104].

Both of aforementioned approaches are exemplified in Figure 5.14. It can be observed that a good choice for the optimal order its around order 20, since after this value the curve for the Levinson-Durbin Recursive algorithm becomes flat and the PACF values tend to approximate zero.

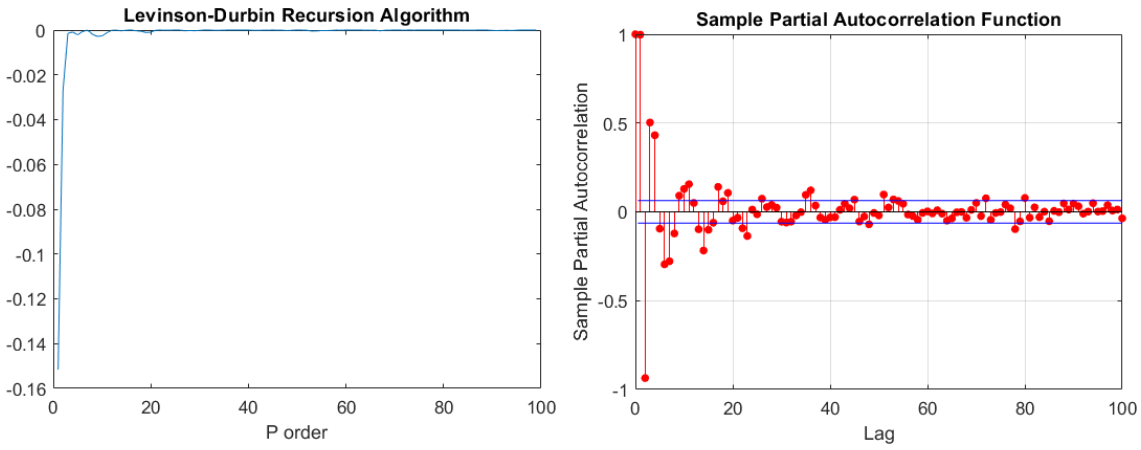


Figure 5.14: Search of the optimal order P for AR model. In the left side is represented the Levinson-Durbin Recursive algorithm approach, while on the right side it is represented the Partial Autocorrelation function approach.

After computing the output coefficients of the AR model for different window sizes and over the time of the signals of the tasks, it was possible to inspect the poles from the transfer function obtained for each situation. It was observed and also concluded that the dominant poles of the process did not vary much in the different cases. Fig. 5.15 depicts the step response for one example of a 5 second window, and it can be seen that it presents a fast process, with a settling time (2% criterion) of only 169 milliseconds, approximately.

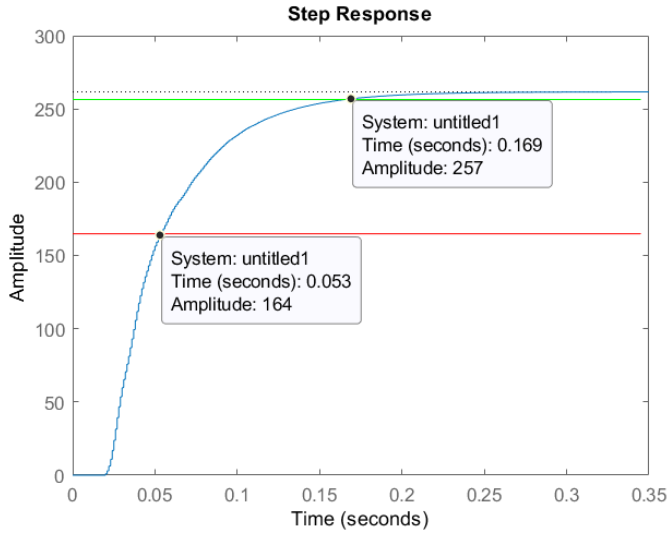


Figure 5.15: Step response of the process for an example of 5-second window of the signal, in order to analyse the dynamic of the system and its settling time.

Hereupon, any of the most commonly used windows mentioned before (1-20 seconds windows), being greater than the stabilization time (0.169 seconds), can be possible to be used for feature extraction.

Afterwards, the existence of statistical differences between the performances obtained for different window sizes was assessed using only the linear features. Windows of 1 second to 10 seconds duration and 80% overlap were inspected. The reason for using only linear features was due to the fact that non-linear features required high computational time to be extracted, making it difficult to use all these features for this analysis for each window size. For that reason only linear features were used in order to select the best window size (see appendix B to inspect computational time differences). Concerning the maximum window size analysed, no windows longer than 10 seconds were used in order to preserve a sufficient number

of samples to analyse on the events with reduced time (e.g., Baseline cross with 30 seconds).

Subsequently, it was investigated if there was significant statistical difference between the performances from the different multiclass models (Code 1 vs Code 2 vs Code 3 vs the Reading (as control) task) using different window sizes to extract the linear features. For this step, a Linear Discriminant Analysis Classifier (further details of this classifier will be presented in section 5.4) was chosen to perform a simple and fast classification (low computational cost).

Afterwards, it was considered two evaluation metrics for inspect the statistical differences: the Accuracy of the model and the F-Measure of the Reading (as control) task. By using Shapiro Wilk and Kolmogorov Smirnov tests, it was possible to conclude that both parameters values distribution did not follow a normal distribution. For that reason and considering that more than two independent groups of variables (10 window sizes) were under inspection, statistical differences were ascertained using non parametric Kruskal–Wallis H test (see Fig. 5.16).

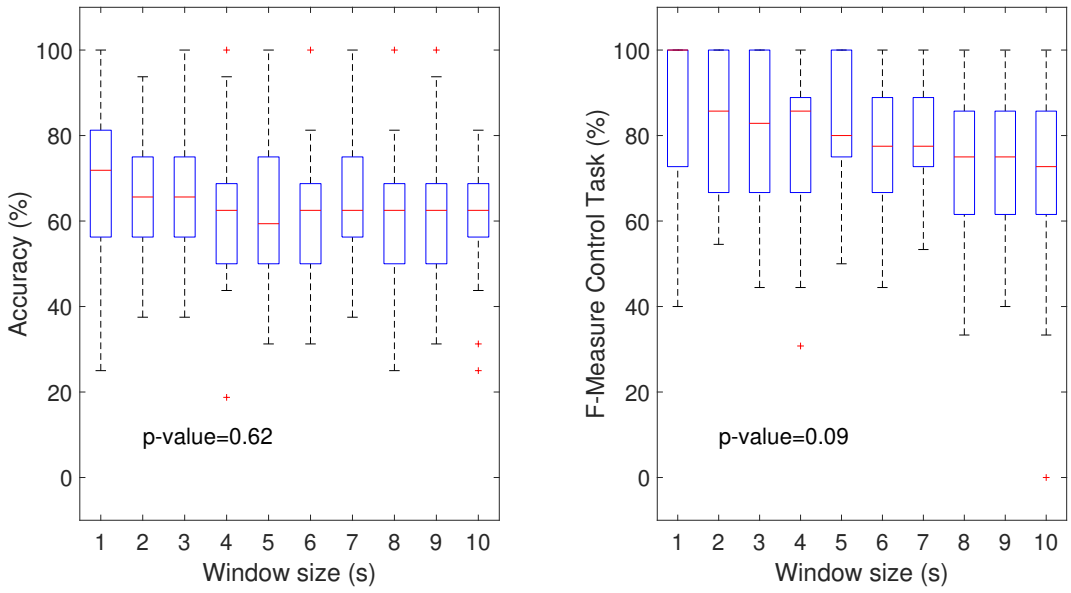


Figure 5.16: Analysis of the window size selection (in seconds) using the Fisher Linear Discriminant Analysis classifier. On the left side is presented the Accuracy in % of the global tasks classification (3 Codes + Reading Control Tasks) with p-value of 0.62. On the right is presented the F-Measure of the Text Control Task with p-value of 0.09.

As no statistically significant differences (p-values > 0.05) were found for the two performance measures, the smaller window size (1 second) was chosen for feature extraction, for a better temporal resolution.

It must be understood that for a future work, with computational resources, it would be interesting to make a global analysis (using both linear and non-linear features) or use adaptive window size, since the best windows found for linear features may not be the best for non-linear ones.

It is also important to note that in this procedure of feature extraction, the features were extracted only after 5 seconds after each event trigger, due to the transitions of stimulus and to the reaction of some participants to the trigger beep [54].

In the following Table 5.1 is represented a summary of the types of features described in this section, as well as the number of features extracted for each analysis.

Table 5.1: Summary of the types and number of the features extracted for the 13 Regions Analysis and for the whole Scalp Analysis (60 signals). For the latter, only linear features were extracted. Further details of the analyses in section 5.6.

Types		Features		
		Feature Name	Number of Components	
			13 Regions Signals	60 Scalp Signals
Linear	Uni-channel Time Domain	Statistical Features	65	300
		Hjorth Parameters	39	180
	Uni-channel Frequency Domain	PSD Features	520	2400
	Multi-channel	Differential Asymmetry	28	63
		Rational Asymmetry	28	63
		”Brainbeat” index	1	1
non-linear	Uni-channel Time Domain	Correlation Dimension	13	~~60~~
		Largest Lyapunov Exp.	13	~~60~~
		Approximate Entropy	13	~~60~~
		Simplicity Measure	13	~~60~~
		Fractal Dimension	13	~~60~~
		Hurst Exponen	13	~~60~~
		Multifractal Spectrum	143	~~660~~
Total			902	3007

5.2.3 Normalization

After the extraction of the aforementioned features for each one of the event tasks and for each trial of all the participants, a feature normalization of the principal events (Code and Reading) was performed. This was done in order to reduce the high inter-subject variability and even to reduce the intra-suject variability throughout the experiment [105].

There are a few approaches to perform the normalization, but in this study, taking into account that there were neutral load events (e.g., the event of the cross), it can be possible to normalize the features with respect to that baseline event. Thus, each cross-event was used for the normalization of the baseline of the next event (1^{st} Cross normalized the Reading task while the 2^{nd} Cross normalized the Code Comprehension task, as it is represented previously on Fig. 4.1, in the protocol description section). It was then verified if the feature values of the cross events had a normal distribution using both Shapiro-Wilk and Kolmogorov-Smirnov tests, in order to choose the mean or median of the event to perform normalization on the following task. Finally, for the same feature, to each sample of the Reading Text and Code Comprehension tasks, it was subtracted and then divided the mean or median value of the cross event, becoming a percentage change value relative to the cross baseline task [106].

5.2.4 Feature Transformation

In order to capture the state of the subject for each code complexity and respective control (reading text) tasks, while maintaining sufficient instances for classification, each task was divided into four segments and five parameters were computed from the normalized features for each segment of those tasks. The parameters calculated were: maximum, minimum, mean, standard deviation and median.

As already mentioned, each task was normalized with the cross baseline, reducing the subject inter and intra-variability. Therefore, concerning the situations of multiclass models, in order to differentiate Code 1, Code 2, Code 3 and global Control, all the

Control tasks from each trial were grouped and then the same procedure of the 5 parameters calculation was performed for each segment of this new group.

Thus, in the end there are two final datasets: one with 624 samples (26 subjects $\times$ 6 tasks $\times$ 4 segments) and another with 416 samples (26 subjects $\times$ 4 tasks $\times$ 4 segments). Both datasets have 4510 features (902 features $\times$ 5 transformed parameters) for the 13 Regions Analysis. For the case of the Whole Scalp Analysis, as it did not consider the non-linear features, the datasets have 15035 features (3007 features $\times$ 5 transformed parameters), instead of 20135 features (4027 features $\times$ 5 transformed parameters), as can it was already observed in Table 5.1.

5.3 Feature Selection

After feature extraction, normalization and transformation, the resulting dataset is ready to be submitted to feature selection and/or dimensionality reduction. These two steps are considered of utmost importance since by performing them before classification it might be possible to improve the learning efficiency of the classifier, their prediction performance, and prevent from overfitting (critical feature dimension) [107]. In this work, four different approaches were investigated for feature selection and dimensionality reduction, separately.

It should be noted that previously to this step, as a preprocessing step for feature selection and classification, feature scaling was performed to the data by using a z-score in order to standardize the values of the features to zero mean and unit variance values, improving the feature selection or/and classification methods, since the features are all in the same range of values.

With the aim of keeping interpretation regarding the features selected, three filter methods were used for feature selection.

5.3.1 Kruskal–Wallis H test or Mann-Whiney U-test

By applying Kolmogorov-Smirnov and Shapiro-Wilk tests it was possible to verify that most of the features of the dataset did not follow a normal distribution.

Based on that, and since the samples are independent, it was considered a non parametric independent test for feature selection like the Kruskal–Wallis H-test or Mann-Whiney U-test. On one hand, the latter is used to compare two independent groups, rejecting or accepting the null hypothesis that the samples originate from the same distribution [108] with significance level α of 0.05 and ranking features according to the U statistic. On the other hand, Kruskal–Wallis H test is an extended case of the previous test, differing only by being used when analysing more than two independent groups and by applying different test statistic H. This test is based on first sorting the values of each feature for all the classes and afterwards compute the H statistics for each feature, according to the Equation 5.8.

$$H = \frac{12}{n \times (n+1)} \times \sum_{i=1}^{c} n_i \times (R_i - \overline{R})^2 \tag{5.8}$$

being n the total number of samples of the data set in question, c the number of classes, n_i the number of samples of class i, R_i the average rank of observations from class i and $\overline{R}$ the average rank for all the observations from all classes. The rank is based on the sorting of the feature values.

In sum, the Mann-Whiney U-test was used for the binary classification models while the Kruskal–Wallis H-test was used for the multiclass classification models. The features were then selected based on their ranking according to the U and H statistics (more discriminant ones are related with higher H or U).

Afterwards, the subset of features selected were inspected for redundancy by computing the Pearson correlation coefficient [109] between features. Then, the features that had more than 0.90 absolute correlation value, i.e., that are strongly correlated, were removed, remaining only one feature of them for the final subset.

5.3.2 ReliefF Algorithm

This method is known to be a robust and noise tolerant method for feature selection, ranking the features by weights [110,111]. It is based on weighting the features through several iterations, checking how do the features behave by analysing the

difference between the features' values of a random instance in relation to the features' values of other k neighbours instances, being ones from the same class and others from different class of the random instance. In each iteration the weights of the features (W_j) are updated (for more detail of the algorithm see page 3 from [110]), and at the end, the features are selected based on the final ranking of the weights. Concerning the number of neighbours k to use on the algorithm, it was defined to be 10 since it was proved to give satisfactory results [112]. In the same way of the previous filter method, it was also used the Pearson correlation coefficient in order to remove redundant features of the selected subset of features from ReliefF Algorithm.

5.3.3 Normalized Mutual Information

In order to select relevant but at the same time non-redundant features, another filter method for feature selection was explored. This method, Normalized Mutual Information (NMI), consists in selecting the best subset of features based on mutual information MI, by measuring the relevancy of the features to the classes and the redundancy within the set of features S [113]. This particular method, by adding an entropy condition for normalization of the mutual information between features, was found to give a better performance since it improves the bias problem that occurs in previous features selection methods based only on the mutual information between the features and classes [114]. The score obtained for each feature is given by the measure G:

$$G = MI(C; f_i) - \frac{1}{S} \times \sum_{f_s \in S} NMI(f_s; f_i) \tag{5.9}$$

being $MI(C; f_i)$ the mutual information between class C and the feature f_i, S the number of the selected subset of features and $NMI(f_s; f_i)$ the normalized mutual information between the feature i and the subset of features S.

On the other hand, additionally to the feature selection techniques previously described, a well-known, widely used dimensionality reduction technique, the Principal

Component Analysis (PCA), was also tested in this work.

5.3.4 Principal Component Analysis

PCA consists on performing feature space reduction while preserving data variability as much as possible. This is done through an orthogonal transformation of the feature space into new directions (principal components) which are determined by computing the eigenvectors from the data covariance matrix. During this process, the variance is maximized and the correlation between the components is minimized [115]. The number of dimensions, i.e., number of principal components, selected was based on the percentage of the cumulative explained variance (CEV) that is decided to maintain [116]:

$$CEV_m(\%) = \frac{\sum_{j=1}^{m} \lambda_j}{\sum_{j=1}^{T} \lambda_j} \times 100 \tag{5.10}$$

being λ the magnitude of the eigenvalue of the component j (correspondent to the variance of the data in this direction), m the number of the first m principal components used for compute the CEV and T the total number of principal components that exists in the new space.

5.4 Classification

After feature selection or dimensionality reduction, four different classifiers were considered for the realization of the models that will try to differentiate the different tasks scenarios.

5.4.1 Fisher Linear Discriminant Analysis Classifier

Fisher Linear Discriminant Analysis (FLDA) classifier performs a linear transformation of the training data, projecting it in directions that (i) maximize the separability between the classes (numerator of eq. 5.11) and (ii) minimize the variability within the classes (denominator of eq. 5.11). This way it is possible to find the best separation plane between the different classes for the classification of new samples

[117]. Although FLDA is a binary classifier by default, it can be used in multiclass scenarios by considering One-vs-All strategy.

$$J(w) = \frac{w^T S_B w}{w^T S_w w} \tag{5.11}$$

being w the direction of projection to be determined, S_B the between-class scatter matrix and S_w the within-class scatter matrix.

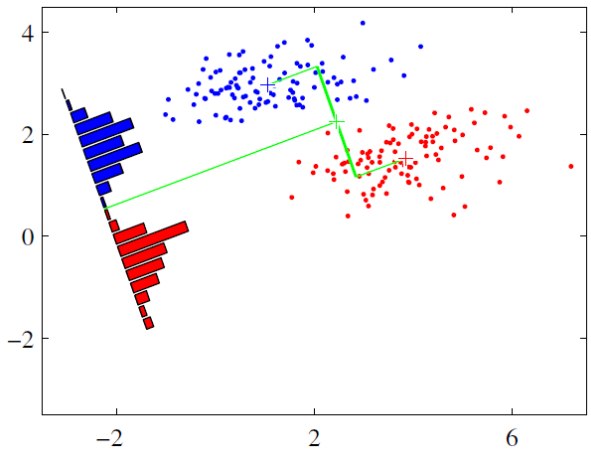


Figure 5.17: Illustration of how FLDA classifier performs by finding the best projection of the data for classification of new samples. In this case for a binary scenario in a 2D space, it is possible to see a projection that allows a clear separation between the two classes. Adapted from [118].

5.4.2 Support Vector Machine

A Support Vector Machine (SVM) with a linear kernel was used in this study in order to minimize the risk of over-fitting, which may be more likely if non-linear kernels (Gaussian or Polynomial) are used. This linear classifier discriminates new samples based on the optimal hyperplane that maximizes the margin between the classes (see Fig. 5.18) in the training data [119, 120]. SVM are natively binary classifiers, but can be used in multi-class problems by combining them using a One-Vs-One, or One-vs-All strategy. Furthermore, the linear kernel SVM has a free parameter, the cost (C), that should be tuned, and that is related to the degree of misclassification that one are able to assume. A higher C value leads to a smaller-margin hyperplane and therefore to a harder penalization of the wrong classification, while a smaller

C value leads to a larger-margin, allowing the model to be more permissive to misclassification.

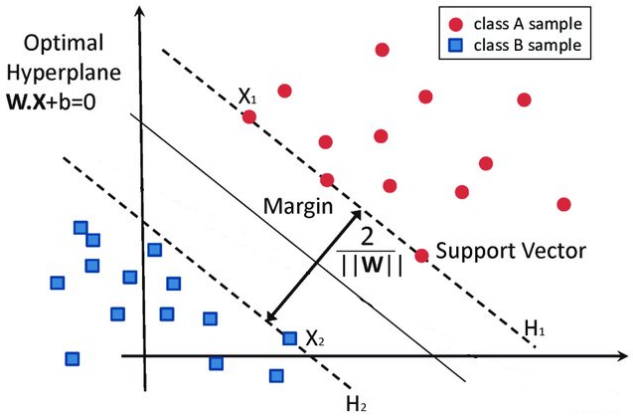


Figure 5.18: Illustration of Linear SVM for a binary classification. The optimal hyperplane $W.X + b = 0$ is the one that maximize the separation margin $\frac{2}{||w||}$ delimited by the supper vectors, as represented in the illustration. Adapted from [121].

5.4.3 k-Nearest Neighbors Algorithm

This classifier, k-Nearest Neighbors (k-NN), is a lazy learner as its training phase is just the storing of the labelled training data. Thus, in testing, a new sample is classified based on the class that predominate among the k nearest neighbors around the mentioned sample [122]. To choose the best odd number of neighbors k for the classifier, a grid search was performed in the validation step.

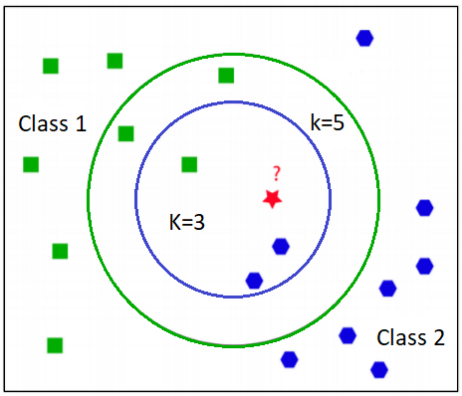


Figure 5.19: Illustration of how k-NN performs for classifying a new sample (red), using $k = 3$ or $k = 5$. If $k = 3$, the new sample will be assign to Class 2 (blue), while if $k = 5$ the sample will be assign to Class 1 (green). Adapted from [122].

5.4.4 Naive Bayes Classifier

Finally, unlike the previous classifiers, the fourth classifier considered in this study is a probability-based model. This classifier is based on the Bayes theorem (see equation 5.12).

$$P(y \mid \vec{x}) = \frac{P(y)P(\vec{x} \mid y)}{P(\vec{x})} \tag{5.12}$$

being y the class label (target), $\vec{x}$ the vector of features (predictor), $P(y)$ the Prior Probability of the class y obtained by the ratio of the number of samples from class y and the total number of samples, $P(\vec{x})$ the Evidence or Predictor Prior Probability, $P(\vec{x} \mid y)$ the Likehood, and $P(y \mid \vec{x})$ the Posterior Probability.

Considering the conditional independence of all features assumed by Naive Bayes, the Likehood in the previous equation can be decomposed, using the chain rule, in the individual independent features. Furthermore, taking into account that the denominator, $P(x_1)P(x_2)...P(x_n)$, is a constant value with equal value for all the samples of the dataset, it can be removed and introduced a proportionality (see equation 5.13).

$$P(y \mid x_1,...,x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{5.13}$$

Thus, Naive Bayes classifier, by taking into account the Bayes theorem and assuming conditional independence of all features, it classifies a new sample into the class with higher posterior probabilities [123], i.e., $\underset{y}{\text{argmax}} P(y) \prod_{i=1}^{n} P(x_i \mid y)$. There are no parameters needed for grid search, since it is based on the probabilities derived from training data.

5.4.5 Validation and Evaluation Metrics

Given that there was not a large amount of data, it was not possible to separate from the start two datasets, one for only test validation step and the remaining for

the final test classification. Thus, it was performed only a Leave-One-Subject-Out cross-validation procedure [124]. This method consists in training the classifiers with the samples of 25 subjects and testing them with the samples of one subject. Thus, the overall grid search results for each model as well as the validation results are obtained based on the statistics from 26 runs. The choice of this type of cross-validation makes it possible to create a model that it is similar to a daily case of classifying new samples from a new different subject with previous information about other subjects.

In order to evaluate the performance of the models (features selection/dimensionality reduction methods combined with different classifiers), it was considered five evaluation metrics:

- Accuracy - corresponds to the ratio between the number of correctly predicted samples, i.e., the True Positives (TP), of all classes n and the total number of samples, giving information of the overall performance of the predictor model.

$$ACC = \frac{\sum_{i=1}^{n} TP_i}{Total\ Samples} \tag{5.14}$$

- Recall (or Sensitivity) - given a class, it consists in the ratio between the number of correctly predicted samples of that class, i.e., the True Positives (TP), and the total number of samples labelled as to belong to that class, i.e., True Positives (TP) plus False Negative (FN).

$$RC = \frac{TP}{TP + FN} \tag{5.15}$$

- Precision - given a class, it consists in the ratio between the number of correctly predicted samples of that class, True Positives (TP), and the total number of samples that were labelled by the classifier as to belong to that class, i.e., True Positives (TP) plus False Positives (FP).

$$PR = \frac{TP}{TP + FP} \tag{5.16}$$

- Specificity - given a class, it consists in the ratio between the number of correctly predicted samples of the other classes, True Negative (TN), and the total number of samples that are labelled with those classes, i.e., True Negative (TN) plus False Positive (FP). It should be noted that for the cases of study involving multiclass models, One-against-All was considered for computation of this evaluation parameter for each class.

$$SP = \frac{TN}{TN + FP} \tag{5.17}$$

- F-measure (F_1-score) - is the harmonic average of the precision and recall parameters, calculated for each class, providing an overall evaluation per class of the False Negative (FN) and False Positives (FP).

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.18}$$

5.5 Statistical Analysis

In order to inspect what were the best combinations of feature selection/reduction and classifiers models, it was verified from the validation results if there was significant statistical differences between the different methods/classifiers combinations. Since the models to be compared are independent, it was only considered independent statistical tests.

First of all, for each statistical analysis, it was verified the normality of the performance values distribution of the models by using Shapiro-Wilk and Kolmogorov-Smirnov tests. Afterwards, if the performance values distribution of all models were normal, the statistical tests used were the parametric ones, otherwise, non parametric tests were considered. Lastly, the statistical test was chosen depending on the number of models being compared: two groups or more than two groups. In Fig. 5.20, it is illustrated a scheme of the steps made in order to choose the statistical test to be used.

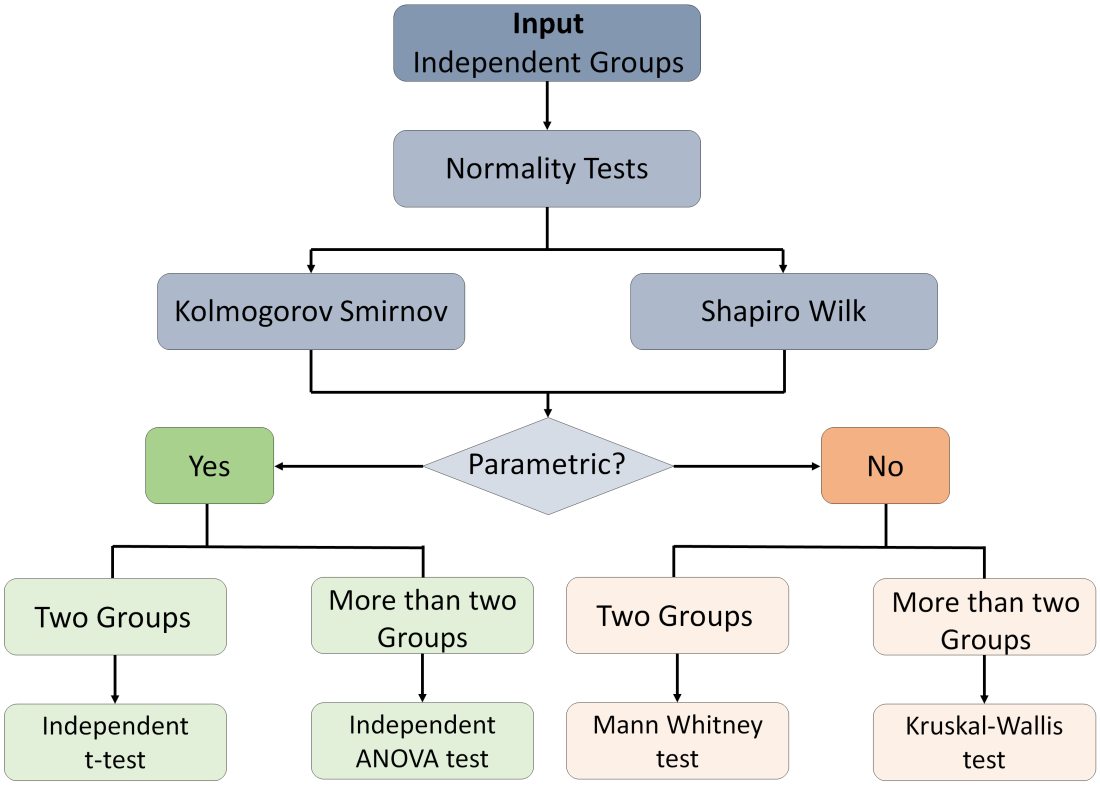


Figure 5.20: Pipeline of classification of the statistical test to be used for the different analysis.

It should be noted that either for the normality verification or the statistical analysis, it was considered a significant level α of 0.05.

5.6 Analysis Performed

5.6.1 Study 1: 13 Regions Analysis

For this study, an analysis was conducted considering the division of the brain into 13 regions of electrode clusters. After the preprocessing step of the 60 EEG signals, the average of the signals that belonged to the same region was computed, reducing the number of signals to analyse to 13. This way, despite loosing some information through the step of averaging, it is possible to considerably reduce the computational cost, and consequently allow for the investigation of the different types of features (linear and non-linear) and at the same time explore the different regions of the brain.

The 13 regions considered are depicted in Fig. 5.21.

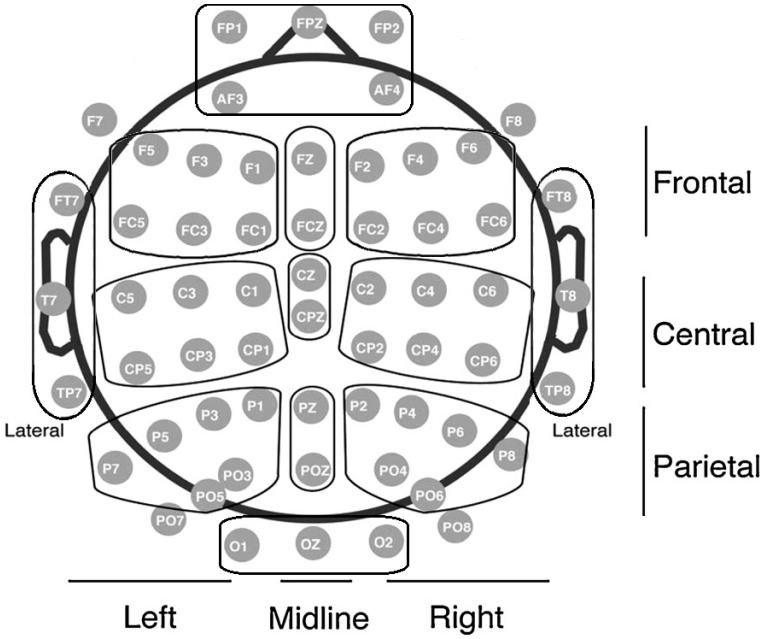


Figure 5.21: Division of the EEG electrodes into 13 regions. Adapted from Kielar et al. [125].

Finally, all combinations of the different feature selection methods with the different classifiers were trained and tested in order to create a model that could distinguish the three different code complexities and/or the resting (control) task.

In Fig. 5.22, it is possible to visualize a schematic representation of the chain of thought for the overall analysis performed.

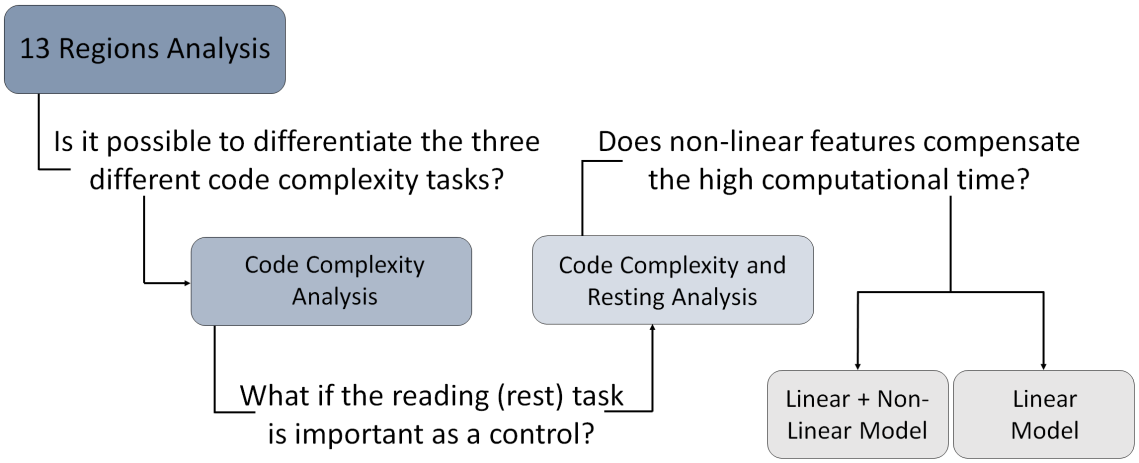


Figure 5.22: Scheme of the chain of thought during 13 Regions Analysis.

5.6.2 Study 2: Whole Scalp Analysis

In this study, the 60 EEG signals were submitted to feature extraction, selection or reduction and classification. By using all signals available, in addition to give a better spatial resolution, it can also be possible to verify if any information was lost by using the previous low complex dataset and/or new findings would be achieved. However, such comparison was only performed for the linear features due to results obtained in the previous analysis and to the associated lower computational cost (further details will be presented in section 6.1.3).

Similarly to the 13 Region Analysis, all combinations of the different feature selection methods with different classifiers were explored in order to create a model that could distinguish the three different code complexities and the resting (control) task.

In Fig. 5.23, it is possible to visualize a schematic representation of the chain of thought of the overall analysis performed, considering the previous analysis results.

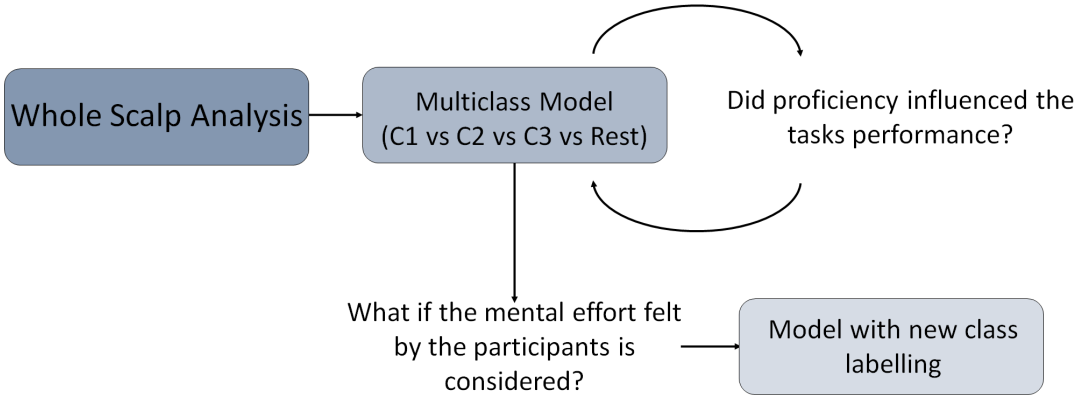


Figure 5.23: Scheme of the chain of thought during Whole Scalp Analysis.

5.6.3 Study 3: Space-Temporal Features Analysis

Finally, as a preliminary analysis, in order to make a more ambitious analysis by lines or blocks of code, the EEG signals together with other two biosignals (Heart Rate Variability (HRV), from Electrocardiogram (ECG), and Pupillography) and the Eye tracking information were inspected. A spatial-temporal analysis can then be performed this way, making it possible to visualize how the different signals

correlate and explore possible biomarkers associated with the more complex parts of the code.

Additionally, all the signals information is also synchronized with a figure of the task with lines or blocks of the code highlighted by a small group of four experienced software professionals. The group selected the regions they considered as potentially critical areas, i.e., regions that may demand more mental effort.

Concerning the other biosignals, it was used the HRV, extracted from the ECG, and the Pupillography. The most discriminant feature found on those modalities were related to the feature LH, which corresponds to the ratio between the power in low frequency range and the power in high frequency range [126]. Thus, the most discriminant features from EEG, obtained by feature selection methods, will be compared with those two features.

Regarding the Eye tracking data, it will allow the synchronization of spacial and temporal information along the biosignals, and therefore achieve a more thorough and complete analysis. The preprocessing and clustering of the Eye tracking data used, was previously performed by Couceiro R. et al. and the methodology can be seen in [126]. Concerning the clustering step of the data points, it was according to three features: y-coordinate along the code, time instants and reading velocity.

In order to evaluate the variation of the EEG features values over the time, each feature value was plotted at the instant of the corresponding window from which it was extracted. The instant was considered the middle instant of the window used. Furthermore, a moving average filter, with window length of 5 samples, was applied to the signal (feature values vector), in order to smooth the signal for better visualization.

6

Results

In this section it will be presented the main results obtained the for three different studies: 13 Regions Analysis; Whole Scalp Analysis; Space-Temporal Features Analysis.

6.1 Study 1: 13 Regions Analysis

In this study, different models were trained and validated either for multiclass or binary scenario, using both linear and non-linear features. The most discriminant features for each scenario were also investigated, in order to uncover if there are features that stand out depending on the tasks to be distinguished.

6.1.1 Code Complexity Analysis

In a first approach, the main focus of the work was to explore the ability of a multiclass model in distinguishing the three different code complexities, Code 1 vs Code 2 vs Code 3 (from the easiest to the most complex according to the software metrics).

For this general multiclass model, for each type of classifier, it was verified if there was significant statistical differences between the feature selection/reduction methods used. As already explained in the section 5.5 concerning Statistical analsyis, firstly, Shapiro-Wilk and Kolmogorov-Smirnov tests were used to check if the values of the accuracy of each model present a normal distribution. It was concluded that, for each classifier, at least one of the feature selection/reduction method, the

statistical tests rejected the null hypothesis of data following a normal distribution. Based on that, and since more than two independent groups of data were under analysis, the non-parametric test Kruskal-Wallis was used to assess if there was significant statistical differences. The boxplots for each of the classifiers and each correspondent p-value are represented in the following Fig. 6.1.

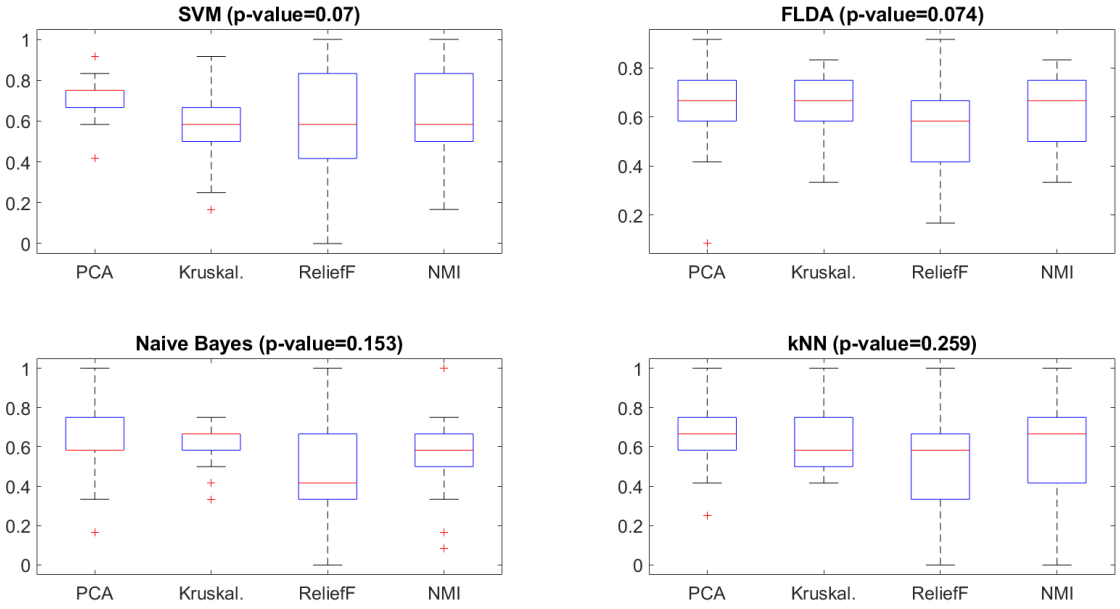


Figure 6.1: Boxplots and corresponding Kruskal-wallis p-value indicating the existence of statistical differences among the four feature selection/reduction methods accuracies. A p-value was obtained for each of the classifiers trained to distinguish the three code tasks.

By looking at the results, it can be observed that for each classifier models the p-values are above the significance level of 0.05, allowing to conclude that there are no significant differences between the methods for feature selection/reduction for these type of multi-class models.

As no significant statistical differences were found between the performances of the feature selection/reduction methods for all classifiers, the results obtained for PCA were randomly chosen to be depicted in Table 6.1, while the results for the remaining methods can be found in Tables C.1 (Kruskal-Wallis H test), C.2 (ReliefF Algorithm) and C.3 (Normalized Mutual Information), in Appendix C.1.1. Those tables present the average performance for different evaluation metrics, and the respective standard deviation values, obtained for the four classifiers designed to distinguish between the

three Codes. The parameters returned by grid search for each of these models are also presented.

Table 6.1: Performance of the four different classifiers after PCA feature reduction (with 70% variance preserved), for the multiclass classification scenario: Code 1 vs Code 2 vs Code 3.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^1)	C1	98.08 ± 6.79	97.69 ± 6.51	98.11 ± 5.44	97.62 ± 5.02	70.83 ± 12.74
	C2	59.61 ± 38.12	56.80 ± 29.79	76.82 ± 18.26	52.89 ± 27.74	
	C3	54.81 ± 34.65	61.32 ± 34.39	80.22 ± 19.63	51.67 ± 27.06	
FLDA	C1	94.23 ± 16.29	99.23 ± 3.92	95.67 ± 19.67	95.62 ± 12.39	68.27 ± 18.86
	C2	58.65 ± 41.79	52.99 ± 35.16	73.76 ± 25.04	50.46 ± 32.25	
	C3	51.92 ± 41.18	51.21 ± 38.99	77.35 ± 23.13	46.57 ± 33.43	
Naive B.	C1	88.46 ± 22.62	93.74 ± 21.18	88.24 ± 29.16	90.09 ± 20.64	61.86 ± 21.24
	C2	65.38 ± 36.10	47.74 ± 29.71	61.58 ± 25.79	52.95 ± 28.47	
	C3	31.73 ± 42.75	31.56 ± 40.89	79.67 ± 24.82	29.46 ± 38.29	
k-NN (k=5)	C1	89.42 ± 20.22	96.99 ± 8.71	93.27 ± 21.86	91.32 ± 14.99	68.27 ± 18.56
	C2	57.69 ± 39.86	51.83 ± 36.54	75.20 ± 22.28	51.09 ± 34.25	
	C3	57.69 ± 37.26	57.20 ± 33.29	76.80 ± 18.79	53.58 ± 30.68	

An overall performance of 70% of accuracy was achieved with the linear SVM classifier, using the One-Against-One multi-class strategy and a cost parameter (C) value of 2^1. Furthermore, although there is a clear distinction between the Code 1 and the other two Codes (F-Measure around 98%), the model is not able to distinguish the more complex Codes, i.e., the Code 2 and Code 3 (F-Measures only around 52%).

Moreover, concerning Table 6.1, although it seems that the SVM achieved the best performance, it was verified if there was significant statistical differences between the performances (accuracy) of the different classifiers models. As the performance values of some groups did not follow a normal distribution, a non parametric statistical test was considered. Therefore, since the groups are independent and more than

two groups to be compared, it was used the Kruskal-Wallis test. It was obtained a p-value of 0.247, a value superior to the significance level of 0.05, leading to the conclusion that there are no significant statistical differences between the classifiers for these type of models. It was also inspected the other feature selection methods if there were any classifier that stand out. However, it was also obtained no significant statistical differences between the classifiers performance in each feature selection method. The boxplots and the multiple comparison tests of the classifiers corresponding to PCA and remaining feature selection methods are presented in figures C.1 and C.2 in Appendix C.1.1.

Afterwards, taking into account the Kruskal–Wallis H test as feature selection method, it was inspected the most frequent features in order to investigate the contribution of the different features type and location. This was performed by analysing the first one hundred selected features in all folds of validation. Figure 6.2 presents a topographic map indicating the brain regions corresponding to the most frequent selected features. In Fig. 6.3 a radar plot is depicted containing information about the most frequent type of features.

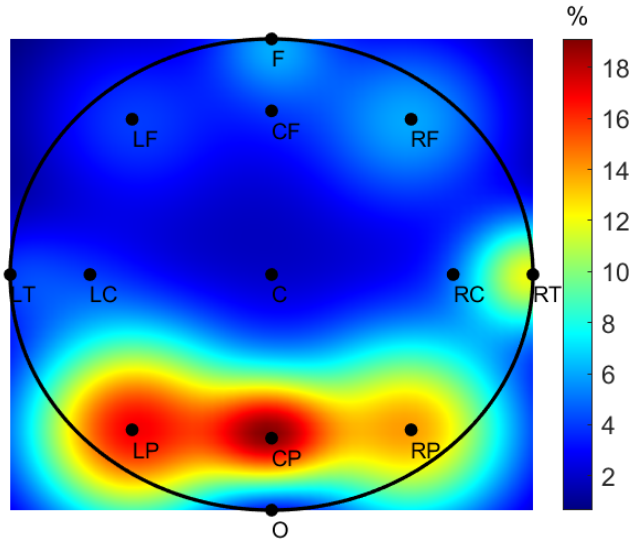


Figure 6.2: Topographic map representing the percentage of the features corresponding to each brain region after feature selection with Kruskal–Wallis H test, for the multiclass scenario: Code 1 vs Code 2 vs Code 3.

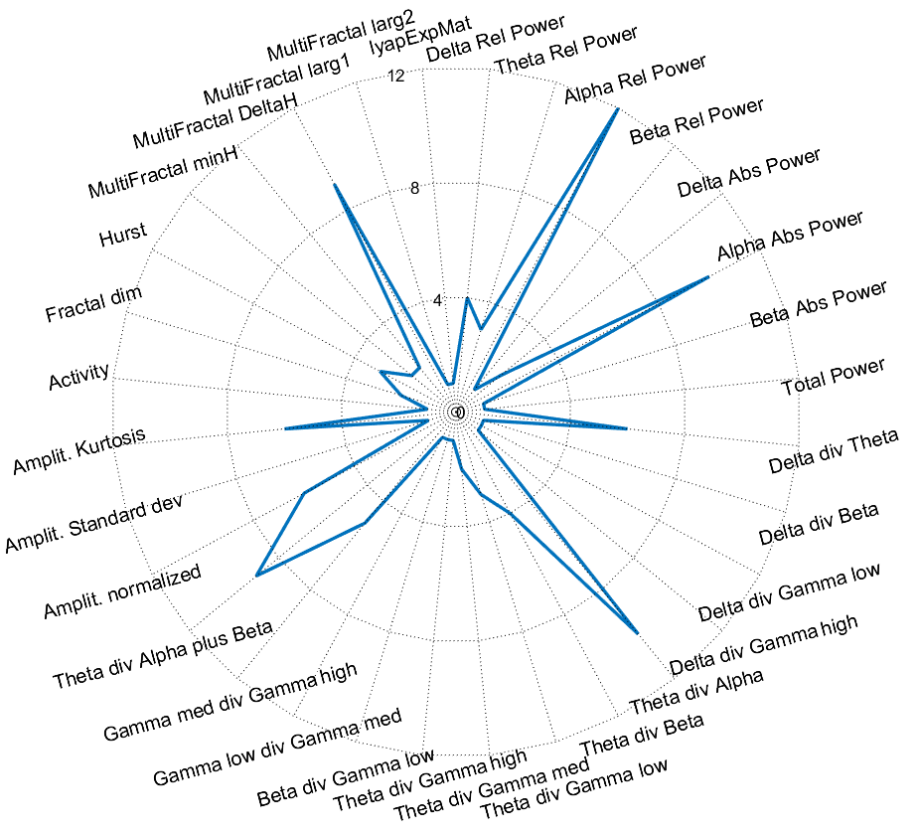


Figure 6.3: Radar plot depicting which type of features are more frequent (in %) in the dataset obtained after feature selection, for the multiclass scenario: Code 1 vs Code 2 vs Code 3.

From the above figures, it was found that in this model that the most frequent features selected belongs to the parietal regions. Such finding is in accordance with the results reported in previous studies regarding comprehension tasks and the mental workload, which pinpoint frontal and parietal regions as the most relevant ones [39]. Additionally, the most frequent type of features correspond to the Absolute and Relative power of Alpha band and the ratios between the Power of Theta and the Power of Beta and/or Alpha. These results are in line with the findings of recent studies in the area as already mentioned in Chapter 3, which emphasized the significant discriminative power of the Theta, Alpha and Beta bands in distinguishing tasks difficulty and assessing mental workload [15, 39, 40], suggesting positive correlation of Theta power and negative correlation of Alpha power with the increase of code complexity.

Afterwards, in order to inspect the interclass discrimination performance, specifically

between Code 2 and Code 3, binary classification models were trained and the corresponding training datasets results were analysed.

Table 6.2 presents the performance results of the binary classifications using PCA, for the different classifiers. The results for the remaining feature selection methods can be found in Tables C.4 (Mann-Whitney U-test), C.5 (ReliefF Algorithm) and C.6 (Normalized Mutual Information) in Appendix C.1.1.

Table 6.2: Performance of the four different classifiers after PCA feature reduction (with 70% variance preserved), for the binary classification scenario: Code task vs Code task.

Classifier	Binary Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM ($C=2^7$)	C1 vs C2	100.00 ± 0.00	96.67 ± 9.66	95.19 ± 14.18	98.03 ± 5.74	97.60 ± 7.09
	C1 vs C3	99.04 ± 4.90	98.35 ± 8.40	97.12 ± 14.71	98.40 ± 5.94	98.08 ± 7.66
	C2 vs C3	51.92 ± 41.79	46.52 ± 34.97	55.77 ± 43.77	44.32 ± 32.11	53.85 ± 19.93
FLDA	C1 vs C2	98.08 ± 6.79	98.72 ± 6.54	98.08 ± 9.81	98.13 ± 5.36	98.08 ± 5.80
	C1 vs C3	96.15 ± 11.60	100.00 ± 0.00	100.00 ± 0.00	97.62 ± 7.41	98.08 ± 5.80
	C2 vs C3	52.88 ± 40.82	47.61 ± 33.25	47.12 ± 43.78	44.05 ± 28.37	50.00 ± 18.37
Naive B.	C1 vs C2	92.31 ± 18.40	89.43 ± 18.39	81.73 ± 32.83	88.09 ± 15.80	87.02 ± 16.76
	C1 vs C3	92.31 ± 22.10	91.09 ± 21.52	92.31 ± 15.44	90.97 ± 20.85	92.31 ± 14.61
	C2 vs C3	67.31 ± 35.19	51.97 ± 26.55	26.92 ± 39.32	52.72 ± 21.28	47.12 ± 18.48
k-NN (k=3)	C1 vs C2	91.35 ± 19.93	92.05 ± 13.73	90.38 ± 17.43	90.13 ± 16.25	90.87 ± 13.49
	C1 vs C3	89.42 ± 21.42	95.26 ± 12.37	95.19 ± 12.29	90.89 ± 17.37	92.31 ± 14.18
	C2 vs C3	55.77 ± 35.57	48.25 ± 33.09	54.81 ± 36.07	50.71 ± 32.59	55.29 ± 24.02

By looking at the results, it is possible to observe the expected clear distinction between (i) Code 1 and Code 2, and (ii) Code 1 and Code 3 (accuracy around 98%), as already seen in previous results of the multiclass models. However, regarding the objective of this binary approach, the models' behaviour when distinguishing Code 2 from Code 3 (the most complex codes) resembles a random prediction.

At this point, in order to investigate the discriminant features, it was used the Mann-Whiney U-test as feature selection method, and the features selected were inspected (see Figure 6.4).

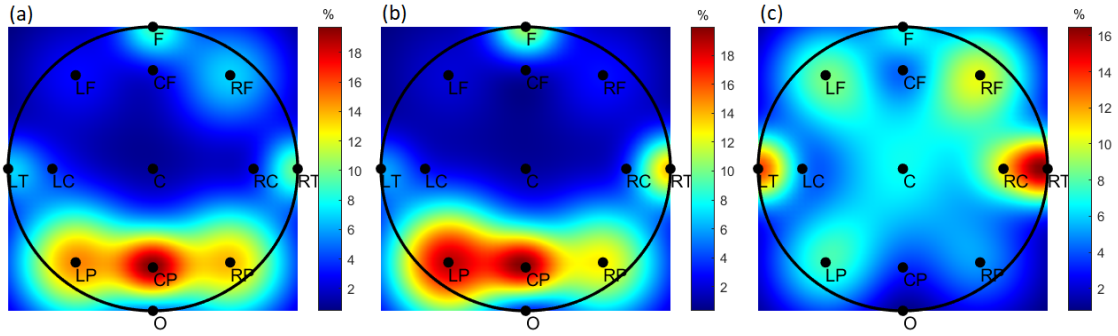


Figure 6.4: Topographic map representing the percentage of the features corresponding to each brain region after feature selection with Mann-Whitney U-test, for the binary classification scenario of Codes: (a) C1 vs C2; (b) C1 vs C3; (c) C2 vs C3.

It can be seen that for the binary classification between the easier Code (1) and the more complex Codes (2 and 3), the features selected in each brain location were similar, with more features being selected from the parietal lobe. On the other hand, when trying to distinguish between Code 2 and Code 3, the selected features did not correspond to the same regions mentioned previously but rather were found to spread across the brain, perhaps trying to overfit for this case that seems difficult to differentiate, as reflected in the low performance of the classification.

In Fig. 6.5, it is represented the boxplots of the highest rank feature for the different classes for each one binary classification problem.

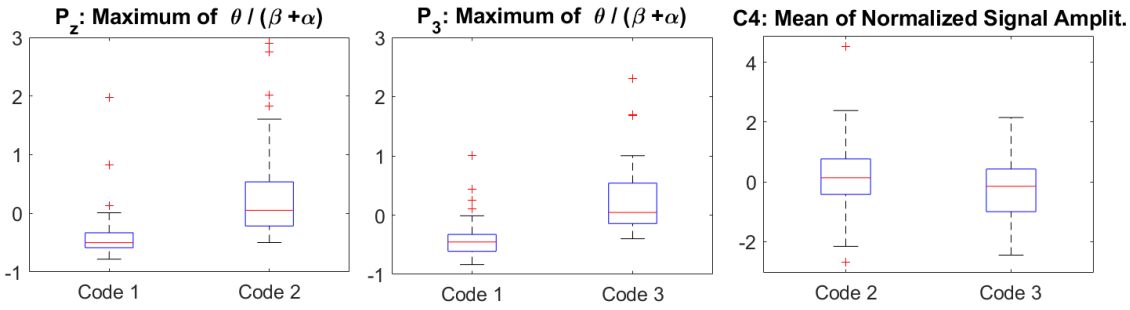


Figure 6.5: Boxplot of the highest rank feature for each binary situation.

Although the presence of some outliers, it can be observed a clear distinction between Code 1 and Codes 2 and 3, using the maximum of $\frac{\theta Power}{\beta Power + \alpha Power}$ during the tasks,

suggesting the synchronization of the Theta power and a desynchronization of Beta and Alpha powers with the increase of the code complexity [49]. However the highest rank feature for the case Code 2 vs Code 3, does not enable a good separation of the classes as expected.

6.1.2 Code Complexity and Resting Analysis

After concluding previous analysis, a more robust analysis was considered: training and validating a multiclass model to distinguish Code 1, Code 2, Code 3 and the Resting Control task, i.e., the reading task (see section 4.2). This way it is possible to ascertain if the models are using features that represent mainly mental workload states and ensure a lower possibility of overfitting.

A statistical analysis was conducted on the accuracy results obtained for the different feature selection/reduction methods obtained for each one of the four classifiers, SVM, FLDA, Naive bayes and k-NN, for this multiclass problem (see Figure 6.6).

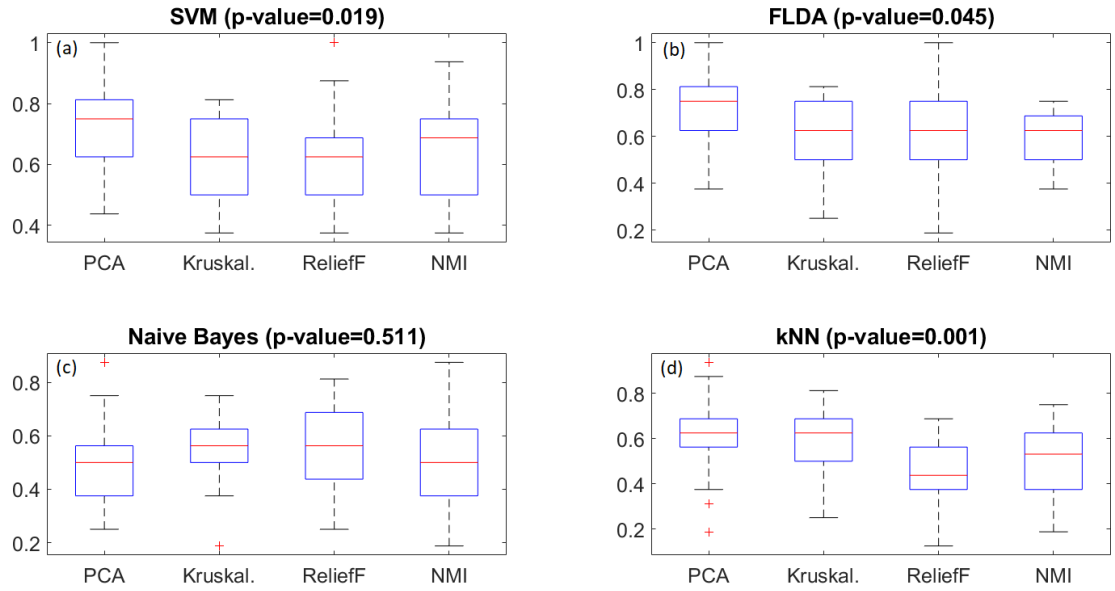


Figure 6.6: Boxplots and corresponding statistical test p-value indicating the existence of statistical differences among the four feature selection/reduction methods accuracies. A p-value was obtained for each of the classifiers trained to distinguish the three code tasks and the resting task. The statistical tests performed were: (a) and (b) Kruskal-Wallis test; (c) and (d) Analysis of Variance (ANOVA).

From this results, it can be observed that there are significant statistical differences

between the feature selection/reduction methods using the SVM, FLDA and k-NN, considering a significance level of 0.05. Additionally, as can be seen in Figure 6.7, where is represented a multiple comparison test of pairwise comparison accuracy results of the different models, PCA is the method that more often presents statistical significant differences for the SVM, FLDA and k-NN classifiers.

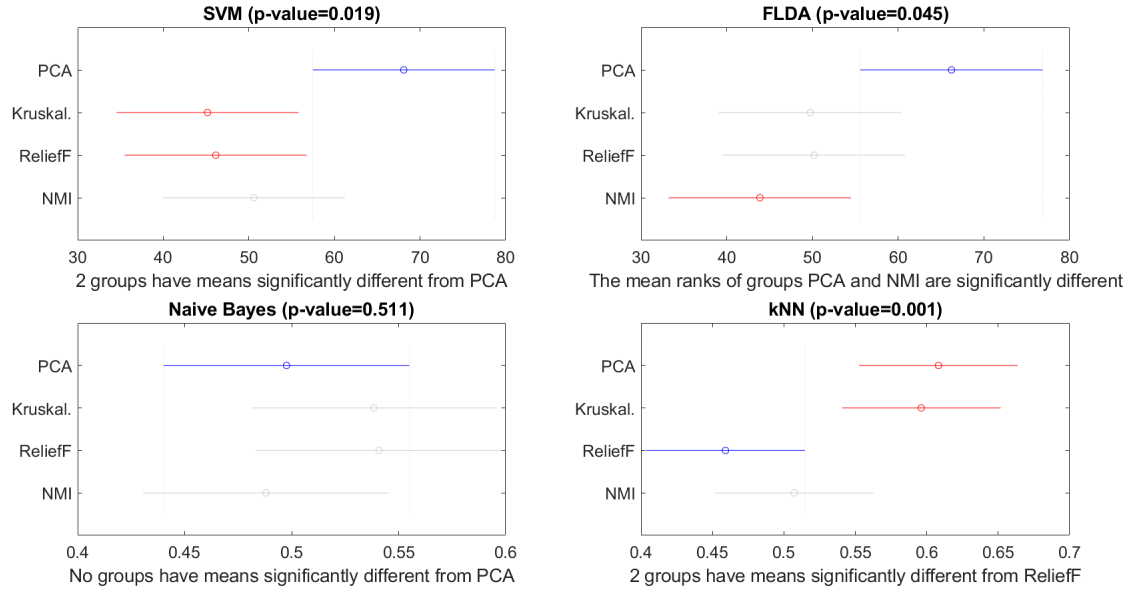


Figure 6.7: Multiple comparison test and respective p-value of the statistical differences between each method of feature selection/reduction for the different classifiers, as classification models of the three code and resting tasks.

Based on the above, the results obtained by applying PCA and subsequently the different classifiers are presented in Table 6.3. The results for the remaining feature selection methods can be found in Tables C.7 (Kruskal-Wallis H test), C.8 (ReliefF Algorithm) and C.9 (Normalized Mutual Information), in Appendix C.1.2.

Table 6.3: Performance obtained for the four classifiers after PCA feature reduction (with 70% variance preserved), for the multiclass classification scenario: Code 1 vs Code 2 vs Code 3 vs Resting Control.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-6})	C1	89.42 ± 22.55	94.62 ± 20.05	99.22 ± 2.76	91.27 ± 20.39	73.32 ± 13.41
	C2	55.77 ± 43.19	43.03 ± 32.74	82.77 ± 12.50	47.13 ± 35.15	
	C3	50.96 ± 35.69	52.78 ± 37.82	85.28 ± 15.62	48.10 ± 31.16	
	Control	97.12 ± 8.15	89.94 ± 14.30	92.42 ± 13.26	92.55 ± 9.47	
FLDA	C1	75.96 ± 37.07	88.46 ± 32.58	100.00 ± 0.00	79.45 ± 34.50	70.91 ± 15.51
	C2	54.81 ± 42.44	44.53 ± 32.87	82.20 ± 16.61	45.95 ± 32.70	
	C3	53.85 ± 41.65	51.69 ± 39.02	86.47 ± 15.86	49.06 ± 35.79	
	Control	99.03 ± 4.90	81.74 ± 21.99	83.67 ± 21.55	87.71 ± 14.41	
Naive B.	C1	50.00 ± 39.37	65.34 ± 43.51	95.38 ± 10.44	53.69 ± 38.00	49.76 ± 16.82
	C2	32.69 ± 35.90	35.94 ± 40.20	83.66 ± 18.81	31.01 ± 32.41	
	C3	25.96 ± 37.74	31.28 ± 41.12	85.12 ± 19.91	25.38 ± 34.36	
	Control	90.38 ± 18.81	47.22 ± 18.17	46.71 ± 29.80	59.95 ± 16.45	
k-NN (k=3)	C1	68.27 ± 35.04	84.87 ± 28.36	96.72 ± 5.56	72.18 ± 31.05	60.82 ± 16.26
	C2	41.35 ± 33.87	45.97 ± 36.01	81.48 ± 15.09	40.20 ± 29.99	
	C3	46.15 ± 37.88	47.34 ± 33.01	80.79 ± 12.44	43.04 ± 29.96	
	Control	87.50 ± 16.20	68.30 ± 23.45	72.33 ± 26.15	73.71 ± 16.67	

By looking at this table it is possible to observe a decrease in F-measure of Code 1 (from 98% to 91%) when compared to the results from the multiclass model considering only the three codes (see Table 6.1). Surprisingly, instead of the expected maximum separation, the F-Measure of the resting (Control) task was only around 93%. Regarding the performance of the higher complexity Codes, it remained the same as the previous multiclass models, being not possible to distinguish between these tasks.

The boxplots from the statistical differences between the classifiers' accuracy re-

sults for a given feature reduction/selection method are presented in Figure C.3, in Appendix C.1.2. Statistical significant differences between the models were found between SVM classifier against Naive Bayes and k-NN classifiers, more specifically on PCA, ReliefF Algorithm and Normalized Mutual Information methods (see multiple comparison tests in Figure C.4, in Appendix C.1.2).

Similar to previous analyses, it was also investigated what type and region the most frequent selected features belonged to. For this situation, this was also performed using Kruskal–Wallis H test feature selection method.

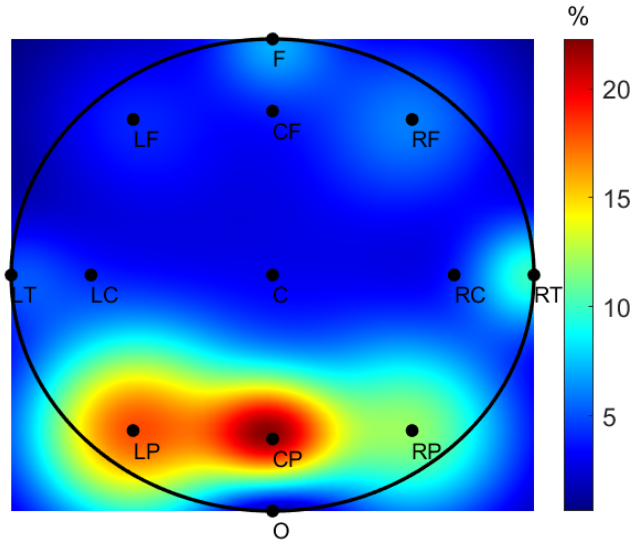


Figure 6.8: Topographic map representing the percentage of the features corresponding to each brain region after feature selection with Kruskal–Wallis H test, for the multiclass scenario Code 1 vs Code 2 vs Code 3 vs Resting Control.

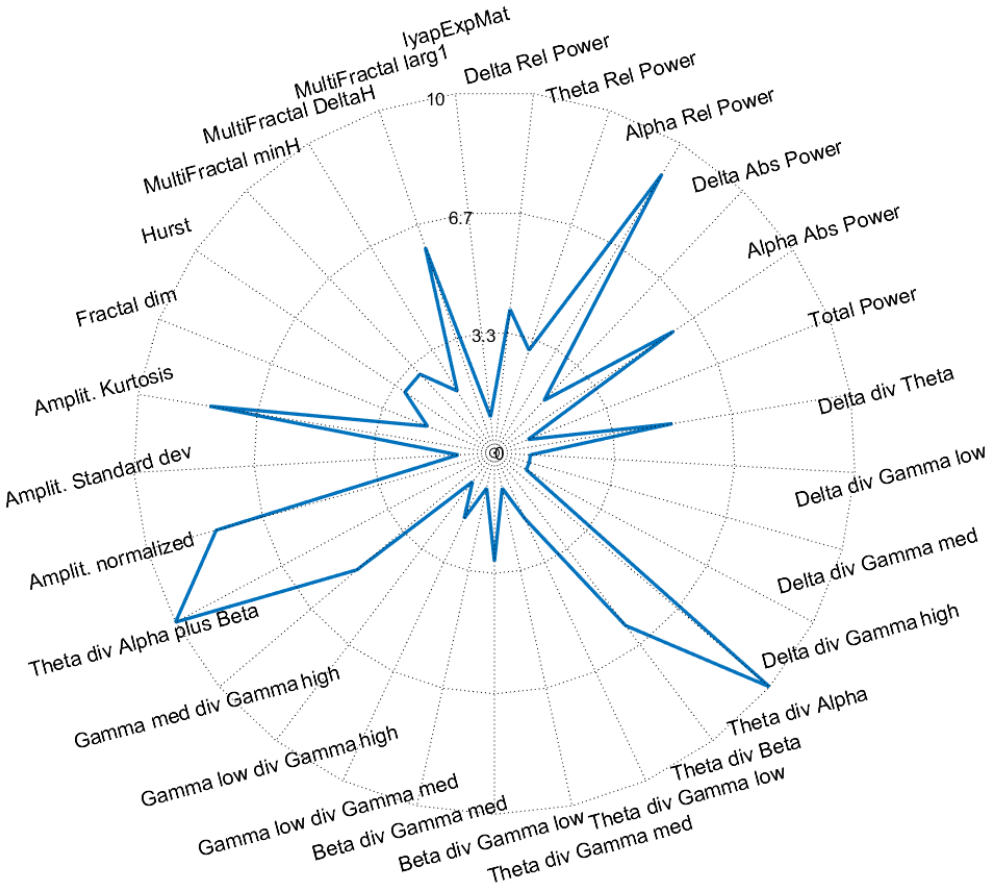


Figure 6.9: Radar plot depicting which type of features are more frequent (in %) in the dataset obtained after feature selection, for the multiclass scenario Code 1 vs Code 2 vs Code 3 vs Resting Control.

In the previous figures 6.8 and 6.9, it is possible to verify that the regions corresponding to the most frequent regions are the parietal lobes, and also that the types of features that stand out comprise Theta, Alpha and Beta band derived features.

From this analysis the value of performance for the resting control task despite being already a satisfactory value, it was not as high as expected, i.e., maximum separation and classification. This fact motivated the analysis of a binary classification of each code task and the respective resting control task. This way, it can be possible to understand if there are any external factors that were not mitigated/addressed with the baseline normalization step (see section 5.2.3). This normalization procedure was meant to correct for any inter or intra-variability present, in any of the three codes when considering a multiclass model.

The performance results corresponding to the three binary classifications (Control 1 vs Code 1, Control 2 vs Code 2 and Control 2 vs Code 3) can be inspected in Table 6.4, for the specific case of PCA. The results for the remaining feature selection methods can be found in Tables C.10 (Mann-Whitney U test), C.11 (ReliefF Algorithm) and C.12 (Normalized Mutual Information), in Appendix C.1.2.

Table 6.4: Performance of the four different classifiers after PCA feature reduction (with 70% variance preserved), for each binary classification scenario: Code task vs Resting Control task.

Classifier	Binary Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^5)	C1 vs Control	75.96 ± 34.26	73.48 ± 29.98	70.19 ± 36.07	70.67 ± 28.92	73.08 ± 22.27
	C2 vs Control	100.00 ± 0.00	99.23 ± 3.92	99.04 ± 4.90	99.57 ± 2.18	99.52 ± 2.45
	C3 vs Control	100.00 ± 0.00	99.23 ± 3.92	99.04 ± 4.90	99.57 ± 2.18	99.52 ± 2.45
FLDA	C1 vs Control	63.46 ± 36.90	72.95 ± 33.87	75.96 ± 33.53	72.95 ± 33.87	69.71 ± 20.97
	C2 vs Control	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	C3 vs Control	100.00 ± 0.00	99.23 ± 3.92	99.04 ± 4.90	99.23 ± 3.92	99.52 ± 2.45
Naive B.	C1 vs Control	63.94 ± 21.31	81.73 ± 26.98	46.15 ± 35.84	56.73 ± 39.65	49.20 ± 35.60
	C2 vs Control	88.94 ± 15.14	93.27 ± 18.11	84.62 ± 22.45	94.38 ± 14.18	87.44 ± 16.81
	C3 vs Control	96.63 ± 7.55	98.08 ± 6.79	95.19 ± 14.18	98.46 ± 5.43	96.03 ± 9.52
k-NN (k=5)	C1 vs Control	61.06 ± 18.48	68.27 ± 32.83	53.85 ± 35.84	56.67 ± 34.99	51.91 ± 31.04
	C2 vs Control	96.63 ± 7.55	96.15 ± 11.60	97.12 ± 10.79	97.18 ± 8.26	96.54 ± 8.13
	C3 vs Control	99.04 ± 3.40	99.04 ± 4.90	99.04 ± 4.90	99.23 ± 3.92	99.02 ± 3.48

It can be observed that the accuracy results returned for the classification of Code 2 vs Control 2 and Code 3 vs Control 3 lie around 100%. However, for the case of Code 1 vs Control 1 the results decrease considerably, which might be explained by the low difficulty of the task, suggesting that there are segments during this task that did not present any additional mental effort for some of the participants in comparison to the mental effort in the resting control tasks.

Additionally, the inspection of the features representation in each brain region was performed on the dataset retrieved by the Mann-Whitney U-test feature selection method (see Figure 6.10).

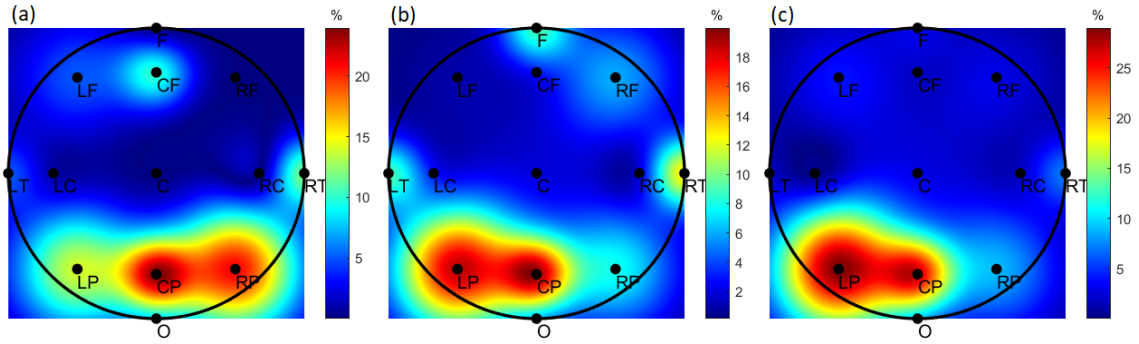


Figure 6.10: Topographic map representing the percentage of the features corresponding to each brain region after feature selection with Mann-Whitney U-test, for the binary classification scenario: (a) Code 1 vs Control 1; (b) Code 2 vs Control 2; (c) Code 3 vs Control 3.

Similarly to the topographic map in Figure 6.8, parietal lobes are also associated with a higher number of features selected by the feature selection method.

In Fig. 6.11, it is represented the highest rank feature, from the selected ones, for each binary classification.

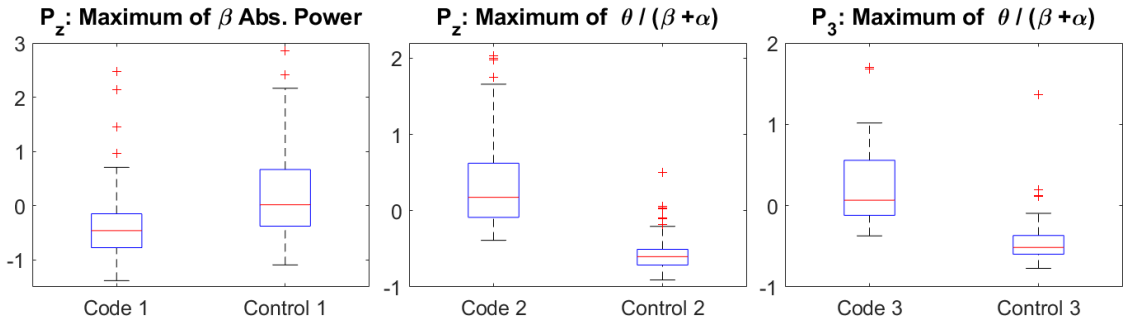


Figure 6.11: Boxplot of the highest rank feature for each binary situation.

The highest rank features for the Codes 2 and 3 against their control task remained to be the previous most common feature , the ratio $\frac{\theta Power}{\beta Power+\alpha Power}$), which, despite some outliers, it shows a clear distinction between the classes. On the other hand, the highest rank feature for the case Code 1 vs Control 1, it was not related to the ratio, but only with one of the variables, the absolute power of Beta, showing an inferior separability than the others situations. This result emphasizes that for

this easier complexity Code, it did not require so much mental effort from some participants during all the task, probably due to its simplicity on comprehension.

6.1.3 Linear and Non Linear Features Performance Analysis

Finally, as already mentioned in Section 5.2, the computational time for the extraction of linear features is considerable lower than the one required for non-linear features extraction (see Appendix B), e.g., for one subject it took around 8 mins for extraction of linear features, while for the non linear it took 23 hours, using a 1 second window and only the 13 region signals. Given that, it was found interesting to create a model using only linear features and compare the performance results with the previous ones, which used both linear and non-linear features.

The results for the different classifiers using PCA, and only linear features, are presented in Table 6.5. Only PCA was considered here, since the principal objective of this analysis was to investigate how the model behaves after removing the non-linear features, and not explore what are the best feature selection/reduction methods.

Table 6.5: Performance of the four different classifiers after PCA feature reduction (with 70% variance preserved), using only linear features, for the multiclass classification scenario: C1 vs C2 vs C3 vs Resting Control.

Classifier	Multiclass Classification	Evaluation Metric				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^5)	C1	88.46 ± 26.67	95.38 ± 19.85	99.36 ± 3.27	90.31 ± 23.41	75.24 ± 12.43
	C2	55.77 ± 36.95	57.84 ± 36.52	87.08 ± 14.62	51.97 ± 30.94	
	C3	60.58 ± 39.48	54.40 ± 33.53	85.40 ± 14.13	53.16 ± 31.45	
	Control	96.15 ± 9.20	87.53 ± 18.80	90.61 ± 15.17	89.99 ± 11.68	
FLDA	C1	75.00 ± 38.73	84.62 ± 36.79	100,00 ± 0,00	77.91 ± 37.12	72.36 ± 16.31
	C2	54.81 ± 41.24	60.00 ± 37.64	85.48 ± 17.12	50.01 ± 31.77	
	C3	60.58 ± 38.19	60.70 ± 36.01	85.74 ± 14.54	56.24 ± 32.03	
	Control	99.04 ± 4.90	81.40 ± 23.22	83.81 ± 21.63	87.30 ± 15.26	
Naive B.	C1	44.23 ± 36.27	66.67 ± 43.97	97.31 ± 6.65	50.84 ± 36.71	48.08 ± 17.12
	C2	25.96 ± 27.82	39.10 ± 40.45	85.21 ± 18.16	28.28 ± 27.98	
	C3	27.88 ± 40.82	26.36 ± 39.22	86.13 ± 17.90	26.48 ± 38.74	
	Control	94.23 ± 12.86	44.94 ± 15.73	40.48 ± 29.64	58.88 ± 13.65	
k-NN (k=3)	C1	65.38 ± 39.42	74.68 ± 36.26	95.65 ± 7.04	67.32 ± 36.76	62.98 ± 14.02
	C2	47.12 ± 34.15	47.60 ± 31.74	82.87 ± 12.62	45.27 ± 29.17	
	C3	52.88 ± 37.63	55.06 ± 33.88	82.28 ± 13.53	49.04 ± 29.97	
	Control	86.54 ± 16.17	69.58 ± 25.42	76.10 ± 23.24	74.45 ± 18.38	

By looking at this table and comparing to the results obtained for the models developed in section 6.1.2 (using linear and non-linear features), the performance of the models designed based only on linear features did not vary significantly for all classifiers (see Figure 6.12)

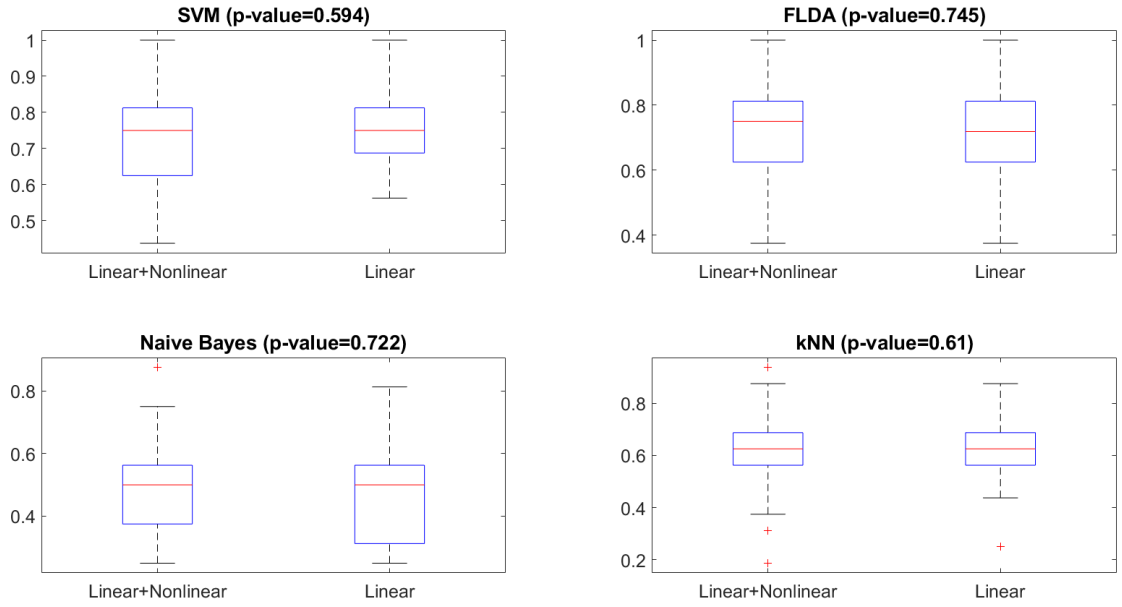


Figure 6.12: Boxplots and respective p-value of the statistical differences between the accuracy results of each models (Linear + Non linear vs Linear) for the different classifiers, after PCA feature reduction. For each classifier case, being only two groups to be compared and all presenting normal distribution, the statistical test performed was the independent t-test.

As can be observed in Fig. 6.12, there are no significant statistical differences between the models using all the information and the models developed using only linear information, considering a significant level of 0.05. For this reason, the following analysis named Study 2 will be conducted using only linear features extracted from all EEG channels. Non-linear features besides not leading to a significant improvement of classifiers' performance, are also more complex features associated with significantly higher computational time.

Similar to previous analyses, it was also investigated what type and region, the most frequent selected features belonged to, by using, as well, the Kruskal–Wallis H test as feature selection method. In Figures 6.13 and 6.14, it is possible to verify that the regions corresponding to the most frequent regions remain being the parietal lobes, and also that the types of features continues to be related to Theta, Alpha and Beta band derived features.

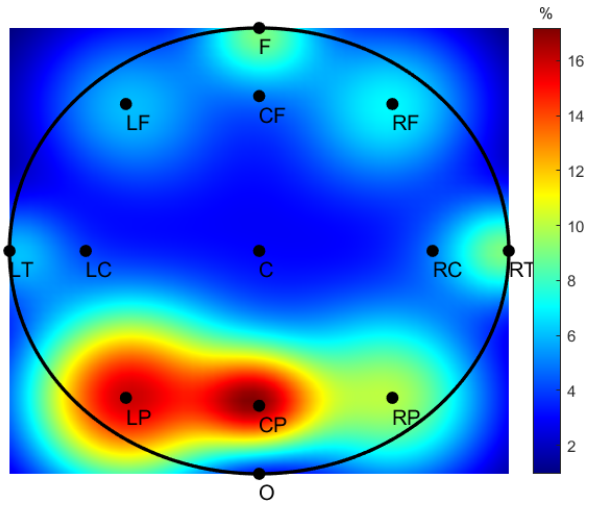


Figure 6.13: Topographic map representing the percentage of the features corresponding to each brain region after feature selection with Kruskal–Wallis H test, for the multiclass scenario using only linear features: Code 1 vs Code 2 vs Code 3 vs Resting Control.

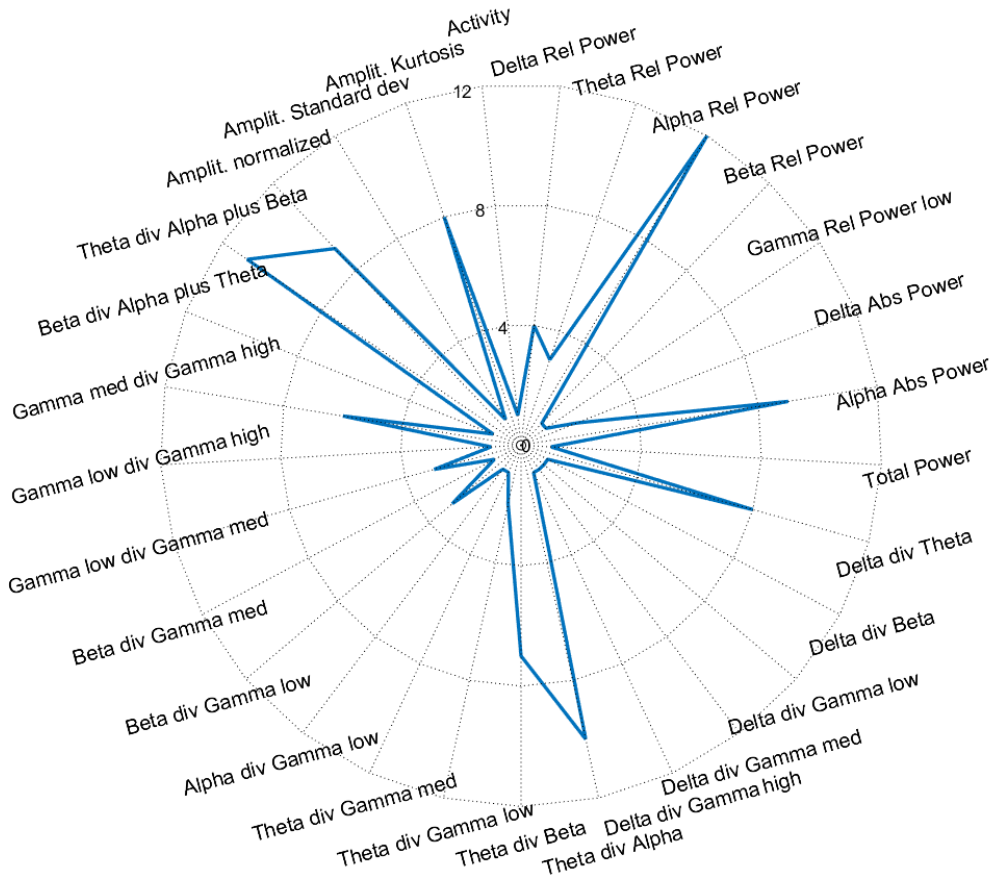


Figure 6.14: Radar plot depicting which type of features are more frequent (in %) in the dataset obtained after feature selection, for the multiclass scenario using only linear features: Code 1 vs Code 2 vs Code 3 vs Resting Control.

6.2 Study 2: Whole Scalp Analysis

Considering that the step of averaging the signals in the different regions might lead to loss of information, it was trained and validated a new multiclass model (Code 1 vs Code 2 vs Code 3 vs Resting Control), but using only linear features, for the already mentioned reasons. This way, it is possible to verify if there is any type of feature prone to differentiate the higher codes complexity, by using all the information of the 60 EEG channels.

6.2.1 Code Complexity and Resting Analysis

Similarly to what was done in Study 1, a statistical analysis was conducted on the accuracy results obtained for the different feature selection/reduction methods obtained for each one of the four classifiers, SVM, FLDA, Naive bayes and k-NN, for this multiclass problem (see Figure 6.15).

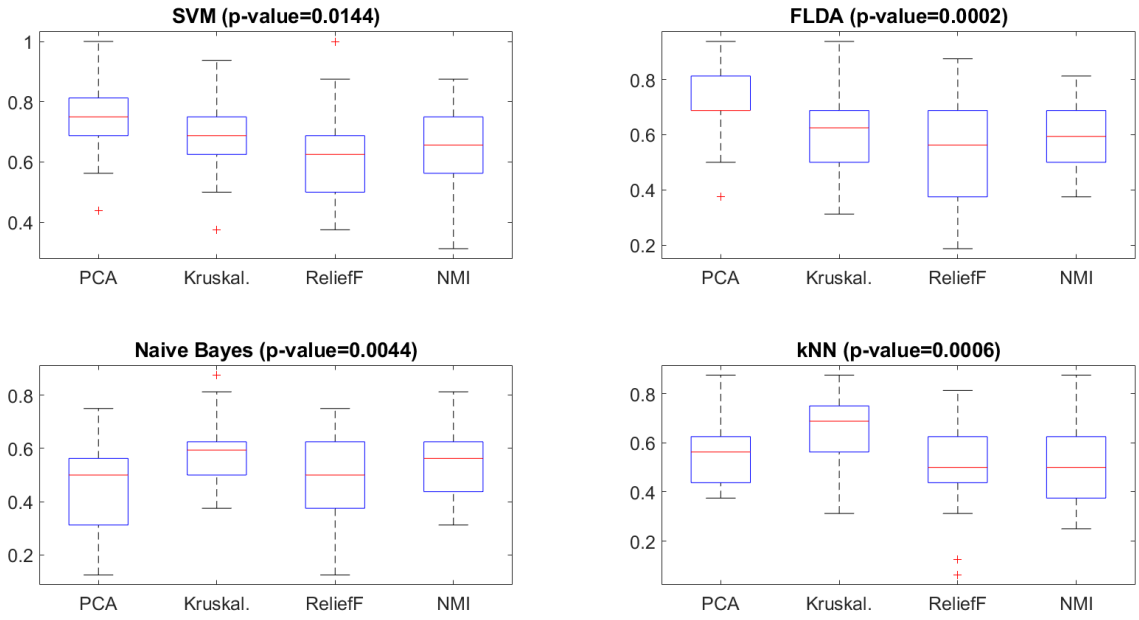


Figure 6.15: Boxplots and respective p-value of the statistical differences between the accuracies values of each method of feature selection/reduction for the different classifiers, as classification models of the three code and resting tasks, using 60 signals information. For each case, being four groups to be compared and all with normal distribution, the statistical test performed was the Analysis of Variance (ANOVA) test.

From this results, it can be observed that there are significant statistical differences between the feature selection/reduction methods for all the different classifiers, considering a significance level of 0.05. Additionally, as can be seen in Figure 6.16, the feature reduction PCA is the method that more often presents statistical significant differences, followed by the feature selection method Kruskall-Wallis H test.

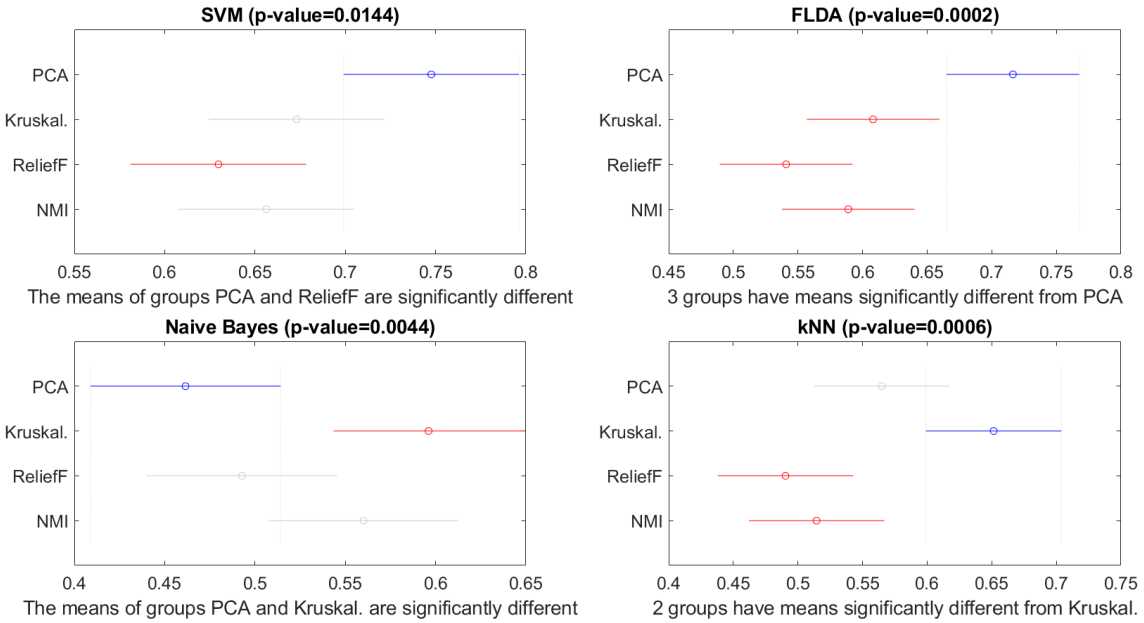


Figure 6.16: Multiple comparison test and respective p-value of the statistical differences between the accuracies values of each method of feature selection/reduction for the different classifiers, as classification models of the three code and control tasks, using 60 signals information.

Based on the above, the results obtained by applying PCA and subsequently the different classifiers are presented in Table 6.6. The results for the remaining feature selection methods can be found in Tables C.13 (Kruskal-Wallis H test), C.14 (ReliefF Algorithm) and C.15 (Normalized Mutual Information), in Appendix C.2.1.

Table 6.6: Performance of the four different classifiers after PCA feature reduction (with 70% variance preserved), for the multiclass classification scenario: C1 vs C2 vs C3 vs Resting Control.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-10})	C1	91.35 ± 19.93	99.23 ± 3.93	99.03 ± 4.73	93.60 ± 14.54	74.76 ± 12.46
	C2	57.69 ± 38.13	51.68 ± 33.39	83.91 ± 12.94	50.60 ± 30.82	
	C3	50.96 ± 36.74	52.06 ± 35.11	85.52 ± 13.70	47.09 ± 30.03	
	Control	99.04 ± 4.90	91.17 ± 14.90	93.75 ± 10.76	94.42 ± 9.67	
FLDA	C1	85.58 ± 25.66	96.15 ± 19.61	100.00 ± 0.00	89.27 ± 22.75	71.63 ± 13.49
	C2	56.73 ± 40.35	48.07 ± 31.21	82.28 ± 14.10	48.79 ± 31.12	
	C3	47.12 ± 38.29	47.39 ± 38.60	85.45 ± 16.64	43.64 ± 33.39	
	Control	97.12 ± 10.79	85.66 ± 17.42	89.30 ± 15.48	89.72 ± 12.71	
Naive B.	C1	45.19 ± 34.65	78.08 ± 40.20	98.29 ± 4.91	53.86 ± 34.06	46.15 ± 16.50
	C2	27.88 ± 32.65	37.05 ± 42.54	84.29 ± 20.17	30.53 ± 34.70	
	C3	25.00 ± 38.08	33.91 ± 45.51	91.87 ± 17.63	26.39 ± 37.49	
	Control	86.54 ± 23.70	35.96 ± 10.24	37.76 ± 22.48	49.64 ± 13.22	
k-NN (k=7)	C1	66.35 ± 35.31	89.49 ± 27.60	98.11 ± 5.44	72.22 ± 31.09	56.73 ± 12.86
	C2	31.73 ± 39.09	30.86 ± 37.17	83.98 ± 15.08	28.94 ± 33.62	
	C3	31.73 ± 35.04	40.77 ± 41.05	88.47 ± 15.69	32.32 ± 31.93	
	Control	97.12 ± 10.79	54.16 ± 19.75	57.37 ± 24.30	67.48 ± 15.57	

From these results, it is possible to observe that despite having more spacial and original information (not averaged), the overall performance regarding, e.g., SVM classifier, remained similar to the first results using the 13 Regions Analysis (Table 6.5): Code 1 and Control with F-Measure around 90% and higher complexity Codes 2 and 3 around 50%, presenting no significant statistical differences on the overall accuracies between each other (p-value of 0.69).

Additionally, it was also explored if there was significant statistical differences between the classifiers for this new model. The boxplots from the statistical differ-

ences between the classifiers' accuracy results for a given feature reduction/selection method are presented in Figure C.5, in Appendix C.2.1. Statistical significant differences between the models were found, more specifically on PCA, where linear SVM and FLDA classifiers stands out from Naive Bayes and k-NN classifiers (see multiple comparison tests in Figure C.6, in Appendix C.2.1). For the case of Normalized Mutual Information feature selection method, there also exists significant statistical differences between the FLDA classifier and the Linear SVM and k-NN classifiers.

Afterwards, it was also investigated what type and electrodes the most frequent selected features belonged to. This was performed using also the Kruskal–Wallis H test feature selection method.

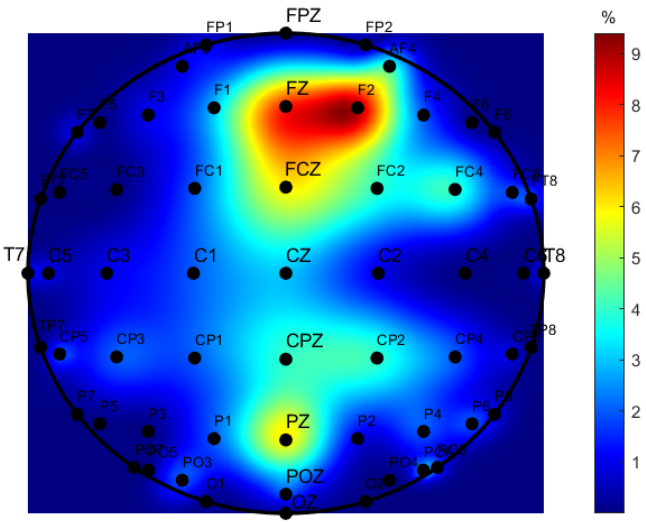


Figure 6.17: Topographic map representing the percentage of the features corresponding to each electrode after feature selection with Kruskal–Wallis H test, for the multiclass scenario Code 1 vs Code 2 vs Code 3 vs Resting Control.

In the previous Fig. 6.17 it is possible to verify that besides some features remained belonging to parietal region (mainly in Pz channel) or near to it, i.e., central parietal (mainly CPz and CP2), it is also revealed a new evident region: frontal region (mainly in Fz, F2 and FCz). These results suggests that in the approach of the first study, there might in fact have been a loss of information due to the step of average of the signals by regions, leading to fading information especially in the two new revealed regions. These new results also comes to emphasize how important the frontal and parietal electrodes can provide information regarding

code comprehension tasks, going in agreement with the findings of recent studies focused in code comprehension tasks [15, 39, 40].

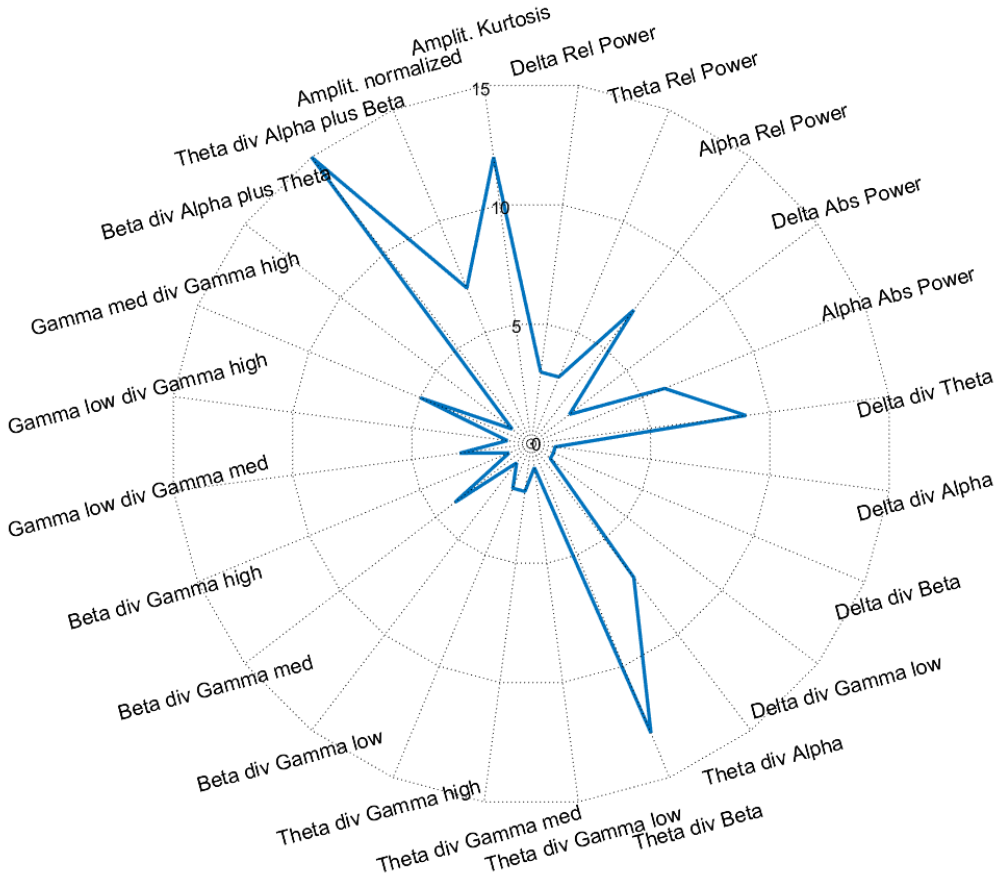


Figure 6.18: Radar plot depicting which type of features are more frequent (in %) in the dataset obtained after feature selection, for the multiclass scenario Code 1 vs Code 2 vs Code 3 vs Resting Control.

By looking to the previous Fig. 6.18, it is possible to confirm that despite new discriminant regions appeared from the analysis, the higher percentage of global type of features remains related to Theta, Alpha and Beta band derived features.

6.2.2 Participant's Proficiency Analysis

Additionally, despite the dataset did not have a balanced number of participants regarding their proficiency, it was inspected the performances of each proficiency class in the results obtained on the previous models from Table 6.6. This was performed in order to have an insight if the proficiency level of the participants had impact on the different tasks and, consequently, on the classification's results.

Given that in the previous analysis it was found that SVM and FLDA were the best classifiers with statistical significant higher performances, it was only considered the results of these two classifiers models for this analysis. Thus, in Table 6.7, it presents the average of the performance's results for each proficiency level of the participants, for the case of the SVM. The FLDA classifier' results are presented in Table C.16 from Appendix C.2.2.

Table 6.7: Performance of Linear SVM classifier after PCA feature reduction, for each different tasks and the three proficiency levels Intermediate, Advanced and Expert). The overall performance of accuracy obtained for each proficiency was 77.60 ± 15.87, 71.25 ± 8.94 and 75.00 ± 13.50, respectively.

Classifier	**Multiclass Classification**	**Evaluation Metric (%)**				
		Recall	**Precision**	**Specificity**	**F-measure**	**Accuracy**
SVM (OAO,C=2^{-4})	**C1**	Intermediate	97.92 ± 7.22	98.33 ± 5.77	97.92 ± 7.22	97.88 ± 4.99
		Advanced	92.50 ± 16.87	100.00 ± 0.00	100.00 ± 0.00	95.24 ± 11.00
		Expert	68.75 ± 37.50	100.00 ± 0.00	100.00 ± 0.00	76.67 ± 29.06
	C2	Intermediate	60.42 ± 43.25	48.65 ± 35.87	86.03 ± 12.01	53.01 ± 37.68
		Advanced	55.00 ± 38.73	50.29 ± 33.58	78.44 ± 12.95	45.45 ± 25.92
		Expert	56.25 ± 42.70	64.29 ± 47.38	91.25 ± 11.81	56.28 ± 38.35
	C3	Intermediate	54.17 ± 39.65	59.72 ± 37.41	86.36 ± 15.00	51.71 ± 33.06
		Advanced	37.50 ± 37.73	33.43 ± 26.48	84.76 ± 12.78	33.15 ± 28.23
		Expert	75.00 ± 35.36	75.71 ± 23.85	84.95 ± 17.32	68.08 ± 24.75
	Control	Intermediate	97.92 ± 7.22	93.33 ± 13.03	95.00 ± 10.40	95.37 ± 10.00
		Advanced	100.00 ± 0.00	94.67 ± 11.67	96.25 ± 8.44	96.89 ± 6.89
		Expert	100.00 ± 0.00	75.95 ± 18.57	83.77 ± 14.01	85.40 ± 11.76

By looking at these results, on one hand, it is possible to observe that the Expert participants were the ones who had the lowest F-measure regarding the Code 1 and the Control tasks. This supports the statement made in the previous analysis, concerning the possibility of the low difficulty that Code 1 presents, having segments that did not require any additional effort in comparison to the control task considered, leading to some overlap between these two classes for classification. On

the other hand, regarding Code 2 and Code 3, the opposite happens: the Expert participants tends to outperform, turning the overall results of the accuracy for each proficiency level similar (around 71-78%, see label of table 6.7). Nevertheless, it should be emphasized that this analysis is very weak and limited by the small number of Expert participants, requiring an increase of acquisitions to be able to perform a more robust valid analysis and generalize the results.

6.2.3 NASA-TLX Labelling Analysis

In the previous analyses, the models were designed considering the class labelling according to the software complexity metrics (see Fig 4.2 in section 4.2). The results obtained reveals that the features' discriminative power stands out when considerably different tasks (in terms of complexity) are being classified (e.g., C1 vs C2 and C1 vs C3), suggesting a saturation with the complexity degree of the software codes (e.g., C2 vs C3). Based on that, it is possible to conclude that the results do not match the complexity levels evaluated with the software metrics, specially the well-known and used MCCabe Cyclomatic Complexity metric. Nevertheless, the results are coherent with the answers to the NASA-TLX survey (described in section 4.2). which also points to such complexity saturation (see Fig.6.19).

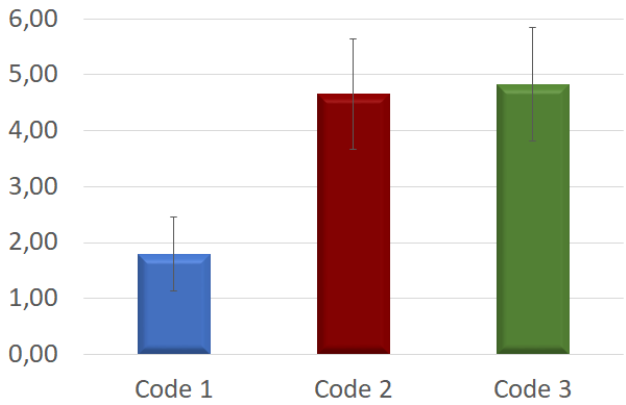


Figure 6.19: Average ratings of the mental effort felt by the participants and written on the NASA-TLX for the three different Code tasks.

In view of the above, it was considered interesting to perform a new labelling of the codes' complexity, according to the mental effort of the NASA-TLX. Afterwards, training and validation were performed in order to obtain a multiclass model able to

distinguish Code 1, Code2/3 and Control. The results obtained for this model that is preceded by PCA feature reduction are presented in Table 6.8. No results were computed for the remaining feature selection methods since the principal objective of this section was to explore how the models performs with the new labelling and compare to the mental effort of the participants, and not to explore what methods are the best.

Table 6.8: Performance of the four different classifiers after PCA feature reduction (with 70% variance preserved), for the multiclass classification scenario (C1 vs C2/C3 vs Resting Control).

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-10})	C1	91.00 ± 20.26	98.40 ± 5.54	99.33 ± 2.31	92.90 ± 14.74	96.50 ± 5.13
	C2/C3	98.50 ± 4.15	100.00 ± 0.00	100.00 ± 0.00	99.20 ± 2.21	
	Control	98.00 ± 6.92	91.62 ± 13.39	96.00 ± 6.85	93.94 ± 8.30	
FLDA	C1	83.00 ± 32.05	92.00 ± 27.69	100.00 ± 0.00	85.79 ± 29.71	93.00 ± 9.77
	C2/C3	96.00 ± 12.35	97.51 ± 6.10	96.16 ± 9.87	96.12 ± 8.23	
	Control	97.00 ± 10.99	86.42 ± 18.87	92.48 ± 11.33	89.93 ± 13.60	
Naive B.	C1	44.00 ± 34.06	75.33 ± 41.07	97.44 ± 6.05	52.17 ± 33.93	54.75 ± 15.23
	C2/C3	43.50 ± 31.48	79.69 ± 30.78	91.20 ± 14.42	52.20 ± 30.35	
	Control	88.00 ± 24.07	36.51 ± 10.50	44.66 ± 21.90	50.44 ± 13.48	
k-NN (k=7)	C1	63.00 ± 36.17	87.20 ± 33.11	99.56 ± 2.22	69.77 ± 33.62	73.25 ± 13.80
	C2/C3	67.50 ± 30.62	85.11 ± 27.51	91.45 ± 12.92	73.06 ± 26.69	
	Control	95.00 ± 12.50	54.93 ± 18.51	68.08 ± 18.91	68.03 ± 15.01	

As expected, the overall performance of the model increased for this situation, achieving for the Linear SVM a maximum accuracy of 96.5%. With respect to class C2/C3 classification performance, a F-Measure of almost 99% was achieved whereas the F-Measure for Code 1 and Control classes remained similar (around 93%), for the reason already mentioned in the previous results.

6.3 Study 3: Space-Temporal Features Analysis

After the previous analysis, it was explored the potential of the most discriminant EEG features as a biomarker to identify complex software sections during the tasks. This was performed by doing a fusion of information with Eye tracker data, allowing a spacial-temporal analysis. Furthermore in this analysis, it was compared the EEG features with features of other two signals: Pupillography and HRV.

In Fig. 6.20, it is depicted one example for the Code task 2 of one expert participant. In the figure, it is represented over the time one of the most discriminant features from HRV and Pupillography and two of the most discriminant features from EEG (bottom - center). Furthermore, regarding Eye tracking data, it is possible to observe the horizontal density of gaze points (positions where the eyes are looking at) along the vertical Y axis of the task (top - left), as well as the clusters of the gaze points over the experience time and the Y axis (top - center). Additionally, it is overlapped the gaze points with the code task figure (top - right), where is also represented the geodesic lines that corresponds to the clusters with higher density of the gaze points. In the code task figure, it is also represented the critical areas considered by the four professionals, marked by different colours, and by a yellow rectangle the overlapping critical areas from different professionals.

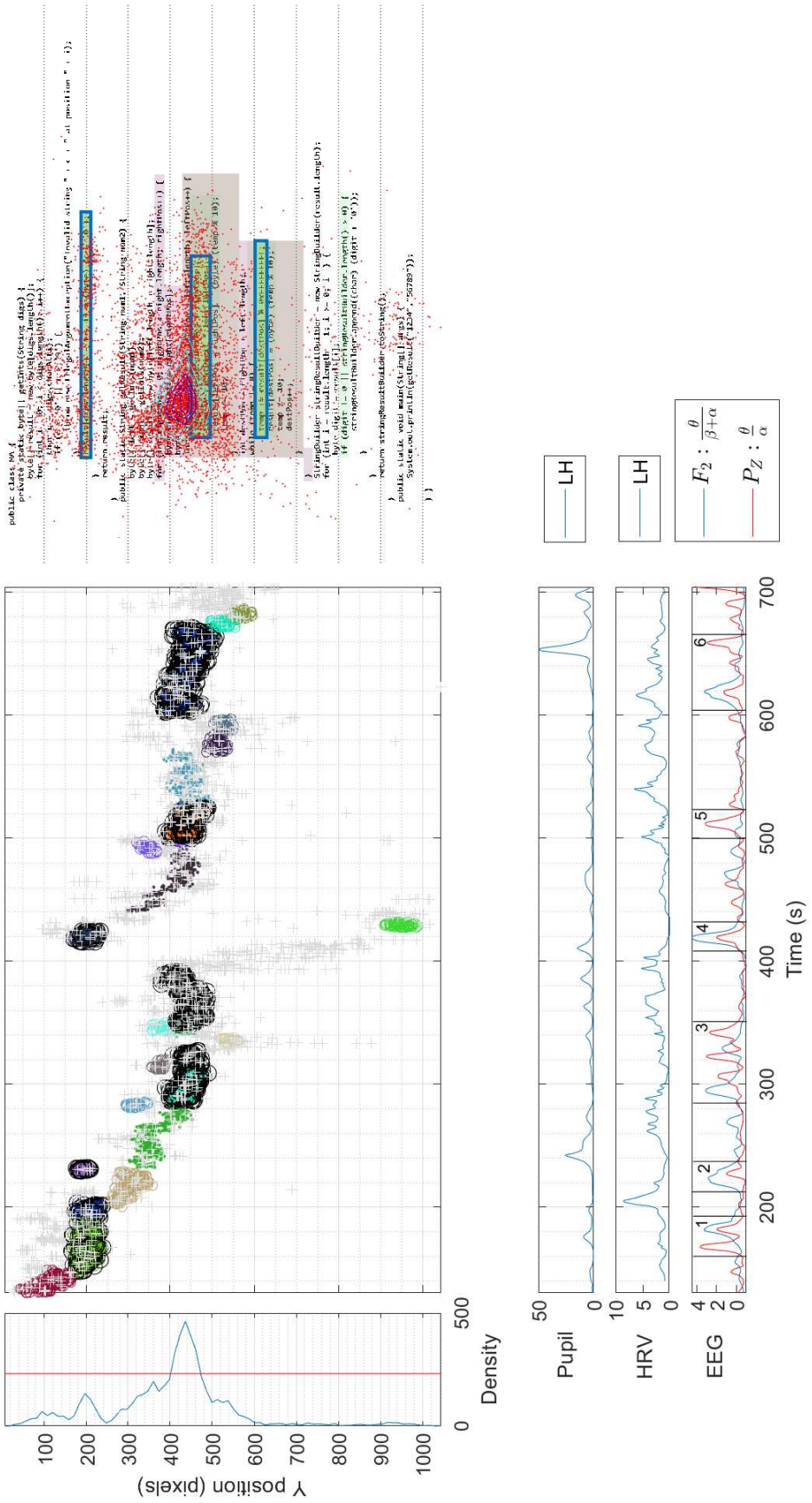


Figure 6.20: Example of the fusion of EEG with HRV, Pupillography and Eye tracking, for an expert participant during the Code task 2.

From the above figure, for the most discriminant EEG features used, i.e., the ratio of $\theta/(\beta + \alpha)$ extracted from the electrode F_2 and the ratio of θ/α extracted from electrode P_Z, it can be observed that are some significant spikes or group of spikes that match with the clusters with higher density of gaze points and with the upper two critical regions marked, more specifically around the instants: (1) 160 - 190; (2) 220-240; (3) 280 - 350; (4) 420; (5) 510 and (6) 600 - 660 seconds. From these instants there are spikes that are common to both features, while there are some spikes corresponding to the high density clusters or critical regions that are only present for one of the features, e.g., the instant represented as (5). This result suggests, that depending on the brain region or the frequency band derived features, it can contribute with the same or additional information about the task being performed. In sum, the overall results of both EEG features, from the two different brain regions highly related with mental workload, reveal possible powerful biomarkers to spot code areas that demand more mental effort or lines of code tending to be critical.

Concerning the relation of the EEG features with the other two biosignals, it is possible to observe that, although there are less spikes presented on the Pupillography and HRV, there still exists some spikes with a delay from the EEG instants represented as (1) and (2), and the instants represented as (6). The spikes of the HRV feature are the ones that presents an higher delay (approximately of 20 seconds) to the EEG spikes. For the case of the feature used from the Pupillography, despite there is a spike that seems delayed from the EEG instant (2), there are other spikes that occur near some of the EEG spikes, e.g., in (1), (4) and (6). Thus, despite having less burst spikes related to the critical regions along the code, these results contribute to reinforce the remarks from others studies [55, 126], using the same dataset, regarding the applicability of using non-invasive and comfortable measurement methods for capture Autonomic Nervous System (ANS) responses, in order to measure the mental effort in software environments.

7

Conclusions

In this work it was assessed cognitive load during code comprehension tasks using EEG. Multiple standard features were compared to standard tools used in software engineering, i.e., complexity metrics, as well as the NASA-TLX tool which is commonly applied in mission critical environments.

One of the main findings is that it is possible to distinguish with high confidence, the simplest code (Code 1) from more complex codes (Code 2 and 3) using EEG. However, Code 2 (middle complexity level code) and Code 3 (highest complexity level code) were not well distinguished. This result suggests that the features only discriminate highly different tasks complexity, revealing an evidence of saturation with the complexity level, being coherent with the answers of the NASA-TLX survey. Nevertheless, the results did not match the evaluations of the codes using the complexity metrics, being proved with the high performances achieved on the analysis performed when considering only two levels of code complexity.

These results suggest that software complexity metrics do not capture cognitive load and, therefore, do not translate a key element in bug production - the human element. Hence, additional research is required to complement current software engineering strategies in order to integrate the programmer in the loop.

Concerning the most discriminant type of features and the regions from which are extracted, the results demonstrate the importance of Theta, Alpha and Beta band derived features for the separability between the efforts in low and high complexity tasks. In addition, the results also show the importance of the contribution of

information, for this purpose, from the electrodes on frontal and parietal regions.

Regarding possible applications, the work developed in this project might potentiate several applications such as: (i) provide feedback to programmers throughout the code about possible critical lines or sections of the code, as shown in the preliminary study of space-temporal analysis; (ii) education and learning, by monitoring the evolution of the mental effort required over time; (iii) criterion for selecting professionals, based on the mental effort on different tasks.

Although the promising results, there are still limitations. Among these limitations is the number of subjects used in the study, that despite being a reasonable number, needs to be increased to see if the findings remain the same, or even to reach further findings. Another limitation is that, as careful as the preprocessing step is, it will never be possible to ensure that all the EEG signals are completely clean and guarantee only the presence of true neural signals.

As future work for this study, it would be interesting to increase the number of subjects for better clarification of the presented results. More specifically, it should be increased the number of expert participants for a more balanced dataset, opening an opportunity to new conclusions regarding expertise levels and respective performances. Another work in mind should be to explore the functional connectivity in the brain using EEG.

As for the project *BASE*, several new data collection studies are being carried out in tasks related to code inspection and code programming. After the acquisition of every subjects is completed, it will be explored the feasibility of the mentioned mental work biomarkers EEG features, obtained in this work, for detection of bugs or even discover new biomarkers, as a possible result from eureka effect.

Bibliography

[1] S. Matteson, "Report: Software failure caused $1.7 trillion in financial losses in 2017." `https://www.techrepublic.com/article/report-software-failure-caused-1-7-trillion-in-financial-losses-in-2017/`, 2018. Last accessed 27 January 2019.

[2] I. Sandu, A. Salceanu, and O. Bejenaru, "New approach of the customer defects per lines of code metric in automotive sw development applications," in *Journal of Physics: Conference Series*, vol. 1065, p. 052006, IOP Publishing, 2018.

[3] R. Minelli, A. Mocci, and M. Lanza, "I know what you did last summer: an investigation of how developers spend their time," in *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*, pp. 25–35, IEEE Press, 2015.

[4] V. Gruhn and R. Laue, "Complexity metrics for business process models," in *9th international conference on business information systems (BIS 2006)*, vol. 85, pp. 1–12, Citeseer, 2006.

[5] S. Scalabrino, G. Bavota, C. Vendome, M. Linares-Vásquez, D. Poshyvanyk, and R. Oliveto, "Automatically assessing code understandability: How far are we?," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pp. 417–427, IEEE Press, 2017.

[6] T. J. McCabe, "A complexity measure," *IEEE Transactions on software Engineering*, no. 4, pp. 308–320, 1976.

[7] N. Kasto and J. Whalley, "Measuring the difficulty of code comprehension tasks using software metrics," in *Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136*, pp. 59–65, Australian Computer Society, Inc., 2013.

[8] S. Scalabrino, M. Linares-Vásquez, D. Poshyvanyk, and R. Oliveto, "Improving code readability models with textual features," in *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, pp. 1–10, IEEE, 2016.

[9] S. Herbold, J. Grabowski, and S. Waack, "Calculation and optimization of thresholds for sets of software metrics," *Empirical Software Engineering*, vol. 16, no. 6, pp. 812–841, 2011.

[10] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information.," *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[11] J. Sweller, "Cognitive load during problem solving: Effects on learning," *Cognitive science*, vol. 12, no. 2, pp. 257–285, 1988.

[12] N. Cowan, "What are the differences between long-term, short-term, and working memory?," *Progress in brain research*, vol. 169, pp. 323–338, 2008.

[13] N. Kumar and J. Kumar, "Measurement of cognitive load in hci systems using eeg power spectrum: an experimental study," *Procedia Computer Science*, vol. 84, pp. 70–78, 2016.

[14] L. Gonçales, K. Farias, B. d. Silva, and J. Fessler, "Measuring the cognitive load of software developers: a systematic mapping study," in *Proceedings of the 27th International Conference on Program Comprehension*, pp. 42–52, IEEE Press, 2019.

[15] M. V. Kosti, K. Georgiadis, D. A. Adamos, N. Laskaris, D. Spinellis, and L. Angelis, "Towards an affordable brain computer interface for the assess-

ment of programmers' mental workload," *International Journal of Human-Computer Studies*, vol. 115, pp. 52–66, 2018.

[16] J. Siegmund, N. Peitek, C. Parnin, S. Apel, J. Hofmeister, C. Kästner, A. Begel, A. Bethmann, and A. Brechmann, "Measuring neural efficiency of program comprehension," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pp. 140–150, ACM, 2017.

[17] OpenStax, "Biology." `http://cnx.org/contents/185cbf87-c72e-48f5-b51e-f14f21b5eabd@11.10`, 2012. Last accessed 25 June 2019.

[18] M. C. F. Fay Evans-martin, Denton A., "Advantages and limitations of time- and time-frequency-domain analyses," in *Your Body. How It Works. The Nervous System*, ch. 3, pp. 31–51, Chelsea House Publishers, 2005.

[19] J. E. Hall in *Guyton and Hall textbook of medical physiology e-Book*, Elsevier Health Sciences, 2015.

[20] S. M. Alarcao and M. J. Fonseca, "Emotions recognition using eeg signals: a survey," *IEEE Transactions on Affective Computing*, 2017.

[21] T. Alotaiby, F. E. A. El-Samie, S. A. Alshebeili, and I. Ahmad, "A review of channel selection algorithms for eeg signal processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2015, no. 1, p. 66, 2015.

[22] L. K. McCorry, "Physiology of the autonomic nervous system," *American journal of pharmaceutical education*, vol. 71, no. 4, p. 78, 2007.

[23] J. Castelhano, I. C. Duarte, C. Ferreira, J. Duraes, H. Madeira, and M. Castelo-Branco, "The role of the insula in intuitive expert bug detection in computer code: an fmri study," *Brain imaging and behavior*, pp. 1–15, 2018.

[24] L. F. Haas, "Hans berger (1873–1941), richard caton (1842–1926), and electroencephalography," *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 74, no. 1, pp. 9–9, 2003.

[25] M. Teplan *et al.*, "Fundamentals of eeg measurement," *Measurement science review*, vol. 2, no. 2, pp. 1–11, 2002.

[26] R. Shriram, M. Sundhararajan, and N. Daimiwal, "Eeg based cognitive workload assessment for maximum efficiency," *Int. Organ. Sci. Res. IOSR*, vol. 7, pp. 34–38, 2013.

[27] C. P. Niemic and K. Warren, "Studies of emotion," *A Theoretical and Empirical Review of Psychophysiological Studies of Emotion.(Department of Clinical and Social Psychology). JUR Rochester*, vol. 1, no. 1, pp. 15–19, 2002.

[28] J. C. Lee and D. S. Tan, "Using a low-cost electroencephalograph for task classification in hci research," in *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pp. 81–90, ACM, 2006.

[29] M. X. Cohen, "Advantages and limitations of time-and time-frequency-domain analyses," in *Analyzing neural time series data: theory and practice*, ch. 2, pp. 24–26, MIT press, 2014.

[30] B. Graimann, B. Z. Allison, and G. Pfurtscheller, *Brain-computer interfaces: Revolutionizing human-computer interaction*. Springer Science & Business Media, 2010.

[31] P. Malmivuo, J. Malmivuo, and R. Plonsey in *Bioelectromagnetism: principles and applications of bioelectric and biomagnetic fields*, Oxford University Press, USA, 1995.

[32] N. Peitek, J. Siegmund, S. Apel, C. Kästner, C. Parnin, A. Bethmann, T. Leich, G. Saake, and A. Brechmann, "A look into programmers' heads," *IEEE Transactions on Software Engineering*, 2018.

[33] K. Brodmann and L. J. Gary, *Brodmann's localization in the cerebral cortex*. Springer, 2006.

[34] S. Katsigiannis and N. Ramzan, "Dreamer: A database for emotion recognition through eeg and ecg signals from wireless low-cost off-the-shelf devices,"

IEEE journal of biomedical and health informatics, vol. 22, no. 1, pp. 98–107, 2017.

[35] J. Siegmund, C. Kästner, S. Apel, C. Parnin, A. Bethmann, T. Leich, G. Saake, and A. Brechmann, "Understanding understanding source code with functional magnetic resonance imaging," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 378–389, ACM, 2014.

[36] T. Nakagawa, Y. Kamei, H. Uwano, A. Monden, K. Matsumoto, and D. M. German, "Quantifying programmers' mental workload during program comprehension based on cerebral blood flow measurement: a controlled experiment," in *Companion Proceedings of the 36th International Conference on Software Engineering*, pp. 448–451, ACM, 2014.

[37] T. Fritz, A. Begel, S. C. Müller, S. Yigit-Elliott, and M. Züger, "Using psycho-physiological measures to assess task difficulty in software development," in *Proceedings of the 36th international conference on software engineering*, pp. 402–413, ACM, 2014.

[38] I. Crk and T. Kluthe, "Toward using alpha and theta brain waves to quantify programmer expertise," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 5373–5376, IEEE, 2014.

[39] S. Lee, A. Matteson, D. Hooshyar, S. Kim, J. Jung, G. Nam, and H. Lim, "Comparing programming language comprehension between novice and expert programmers using eeg analysis," in *2016 IEEE 16th International Conference on Bioinformatics and Bioengineering (BIBE)*, pp. 350–355, IEEE, 2016.

[40] M. K.-C. Yeh, D. Gopstein, Y. Yan, and Y. Zhuang, "Detecting and comparing brain activity in short program comprehension using eeg," in *2017 IEEE Frontiers in Education Conference (FIE)*, pp. 1–5, IEEE, 2017.

[41] S. Lee, D. Hooshyar, H. Ji, K. Nam, and H. Lim, "Mining biometric data to predict programmer expertise and task difficulty," *Cluster Computing*, vol. 21, no. 1, pp. 1097–1107, 2018.

[42] S. Scalabrino, G. Bavota, C. Vendome, D. Poshyvanyk, R. Oliveto, *et al.*, "Automatically assessing code understandability," *IEEE Transactions on Software Engineering*, 2019.

[43] M. R. Wrobel, "Emotions in the software development process," in *2013 6th International Conference on Human System Interactions (HSI)*, pp. 518–523, IEEE, 2013.

[44] I. A. Khan, W.-P. Brinkman, and R. M. Hierons, "Do moods affect programmers' debug performance?," *Cognition, Technology & Work*, vol. 13, no. 4, pp. 245–258, 2011.

[45] A.-M. Brouwer, T. O. Zander, J. B. Van Erp, J. E. Korteling, and A. W. Bronkhorst, "Using neurophysiological signals that reflect cognitive or affective state: six recommendations to avoid common pitfalls," *Frontiers in neuroscience*, vol. 9, p. 136, 2015.

[46] E. Pellouchoud, M. E. Smith, L. McEvoy, and A. Gevins, "Mental effort-related eeg modulation during video-game play: Comparison between juvenile subjects with epilepsy and normal control subjects," *Epilepsia*, vol. 40, pp. 38–43, 1999.

[47] M. E. Smith and A. Gevins, "Neurophysiologic monitoring of mental workload and fatigue during operation of a flight simulator," in *Biomonitoring for Physiological and Cognitive Performance during Military Operations*, vol. 5797, pp. 116–127, International Society for Optics and Photonics, 2005.

[48] C. Berka, D. J. Levendowski, M. N. Lumicao, A. Yau, G. Davis, V. T. Zivkovic, R. E. Olmstead, P. D. Tremoulet, and P. L. Craven, "Eeg correlates of task engagement and mental workload in vigilance, learning, and memory tasks," *Aviation, space, and environmental medicine*, vol. 78, no. 5, pp. B231–B244, 2007.

[49] A. Gevins, M. E. Smith, H. Leong, L. McEvoy, S. Whitfield, R. Du, and G. Rush, "Monitoring working memory load during computer-based tasks with

eeg pattern recognition methods," *Human factors*, vol. 40, no. 1, pp. 79–91, 1998.

[50] W. Klimesch, "Eeg alpha and theta oscillations reflect cognitive and memory performance: a review and analysis," *Brain research reviews*, vol. 29, no. 2-3, pp. 169–195, 1999.

[51] Neurosky, "Neurosky's esense^{TM} meters and detection of mental state." `http://www.brainathlete.jp/pdf/WP-lee-neurosky-esense.pdf`, 2009. Last accessed 22 May 2019.

[52] J. B. Brookings, G. F. Wilson, and C. R. Swain, "Psychophysiological responses to changes in workload during simulated air traffic control," *Biological psychology*, vol. 42, no. 3, pp. 361–377, 1996.

[53] I. Crk and T. Kluthe, "Assessing the contribution of the individual alpha frequency (iaf) in an eeg-based study of program comprehension," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 4601–4604, IEEE, 2016.

[54] G. D. T. Duarte, "Cognitive and autonomic neural manifestations captured using wearable sensors for reliable software development," Master's thesis, University of Coimbra, February 2019.

[55] R. Couceiro, G. Duarte, J. Durães, J. Castelhano, C. Duarte, C. Teixeira, M. Castelo-Branco, P. Carvalho, and H. Madeira, "Pupillography as indicator of programmers' mental effort and cognitive overload," in *IEEE Int. Conf. on Dependable Systems and Networks – DSN 2019*, IEEE, 2019. (Accepted).

[56] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*, vol. 52, pp. 139–183, Elsevier, 1988.

[57] N. E. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Transactions on Software engineering*, vol. 26, no. 8, pp. 797–814, 2000.

[58] A. Delorme and S. Makeig, "Eeglab: an open source toolbox for analysis of single-trial eeg dynamics including independent component analysis," *Journal of neuroscience methods*, vol. 134, no. 1, pp. 9–21, 2004.

[59] A. Delorme, C. Kothe, A. Vankov, N. Bigdely-Shamlo, R. Oostenveld, T. O. Zander, and S. Makeig, "Matlab-based tools for bci research," in *Brain-computer interfaces*, pp. 241–259, Springer, 2010.

[60] A. Widmann, E. Schröger, and B. Maess, "Digital filter design for electrophysiological data–a practical approach," *Journal of neuroscience methods*, vol. 250, pp. 34–46, 2015.

[61] I. Winkler, S. Debener, K.-R. Müller, and M. Tangermann, "On the influence of high-pass filtering on ica-based artifact reduction in eeg-erp," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 4101–4105, IEEE, 2015.

[62] F. Perrin, J. Pernier, O. Bertrand, and J. Echallier, "Spherical splines for scalp potential and current density mapping," *Electroencephalography and clinical neurophysiology*, vol. 72, no. 2, pp. 184–187, 1989.

[63] M. X. Cohen, "Preprocessing steps necessary and useful for advanced data analysis," in *Analyzing neural time series data: theory and practice*, ch. 7, pp. 71–85, MIT press, 2014.

[64] D. Yao, L. Wang, R. Oostenveld, K. D. Nielsen, L. Arendt-Nielsen, and A. C. Chen, "A comparative study of different references for eeg spectral mapping: the issue of the neutral reference and the use of the infinity reference," *Physiological measurement*, vol. 26, no. 3, p. 173, 2005.

[65] N. Bigdely-Shamlo, *Combining EEG Source Dynamics Results across Subjects, Studies and Cognitive Events*. PhD thesis, UC San Diego, 2014.

[66] Q. Liu, J. H. Balsters, M. Baechinger, O. van der Groen, N. Wenderoth, and D. Mantini, "Estimating a neutral reference for electroencephalographic

recordings: the importance of using a high-density montage and a realistic head model," *Journal of neural engineering*, vol. 12, no. 5, p. 056012, 2015.

[67] J.-F. Cardoso, "Blind signal separation: statistical principles," *Proceedings of the IEEE*, vol. 86, no. 10, pp. 2009–2025, 1998.

[68] P. Comon, "Independent component analysis, a new concept?," *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[69] M. X. Cohen, "EEG Artifacts: Their Detection, Influence, and Removal," in *Analyzing neural time series data: theory and practice*, ch. 8, pp. 87–96, MIT press, 2014.

[70] D. Dharmaprani, H. K. Nguyen, T. W. Lewis, D. DeLosAngeles, J. O. Willoughby, and K. J. Pope, "A comparison of independent component analysis algorithms and measures to discriminate between eeg and artifact components," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 825–828, IEEE, 2016.

[71] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural computation*, vol. 9, no. 7, pp. 1483–1492, 1997.

[72] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[73] J. A. Urigüen and B. Garcia-Zapirain, "Eeg artifact removal—state-of-the-art and guidelines," *Journal of neural engineering*, vol. 12, no. 3, p. 031001, 2015.

[74] T.-W. Lee, M. Girolami, and T. J. Sejnowski, "Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources," *Neural computation*, vol. 11, no. 2, pp. 417–441, 1999.

[75] A. Delorme and S. Makeig, "Eeglab wikitorial," *Retrieved from10*, vol. 1016, 2009.

[76] A. Spanos, *Probability theory and statistical inference: econometric modeling with observational data*. Cambridge University Press, 1999.

[77] B. Hjorth, "Eeg analysis based on time domain properties," *Electroencephalography and clinical neurophysiology*, vol. 29, no. 3, pp. 306–310, 1970.

[78] R. Jenke, A. Peer, and M. Buss, "Feature extraction and selection for emotion recognition from eeg," *IEEE Transactions on Affective Computing*, vol. 5, no. 3, pp. 327–339, 2014.

[79] A. T. Pope, E. H. Bogart, and D. S. Bartolome, "Biocybernetic system evaluates indices of operator engagement in automated task," *Biological psychology*, vol. 40, no. 1-2, pp. 187–195, 1995.

[80] F. G. Freeman, P. J. Mikulka, M. W. Scerbo, and L. Scott, "An evaluation of an adaptive automation system using a cognitive vigilance task," *Biological psychology*, vol. 67, no. 3, pp. 283–297, 2004.

[81] S. Lei, *Driver mental states monitoring based on brain signals*. PhD thesis, Ph. D. thesis, TU Berlin, Germany, 2011.

[82] M. Kostyunina and M. Kulikov, "Frequency characteristics of eeg spectra in the emotions," *Neuroscience and Behavioral Physiology*, vol. 26, no. 4, pp. 340–343, 1996.

[83] E. Angelakis, S. Stathopoulou, J. L. Frymiare, D. L. Green, J. F. Lubar, and J. Kounios, "Eeg neurofeedback: a brief overview and an example of peak alpha frequency training for cognitive enhancement in the elderly," *The clinical neuropsychologist*, vol. 21, no. 1, pp. 110–129, 2007.

[84] J. Liu, H. Meng, M. Li, F. Zhang, R. Qin, and A. K. Nandi, "Emotion detection from eeg recordings based on supervised and unsupervised dimension reduction," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 23, p. e4446, 2018.

[85] A. Holm, K. Lukander, J. Korpela, M. Sallinen, and K. M. Müller, "Estimating brain load from the eeg," *The Scientific World Journal*, vol. 9, pp. 639–651, 2009.

[86] A. Varsavsky, I. Mareels, and M. Cook, "Signal Processing in EEG Analysis," in *Epileptic seizures and the EEG: measurement, models, detection and prediction*, ch. 3, pp. 89–137, CRC Press, 2016.

[87] P. Grassberger and I. Procaccia, "Characterization of strange attractors," *Physical review letters*, vol. 50, no. 5, p. 346, 1983.

[88] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, "A practical method for calculating largest lyapunov exponents from small data sets," *Physica D: Nonlinear Phenomena*, vol. 65, no. 1-2, pp. 117–134, 1993.

[89] S. M. Pincus, I. M. Gladstone, and R. A. Ehrenkranz, "A regularity statistic for medical data analysis," *Journal of clinical monitoring*, vol. 7, no. 4, pp. 335–345, 1991.

[90] V. Nigam and R. Priemer, "Accessing heart dynamics to estimate durations of heart sounds," *Physiological measurement*, vol. 26, no. 6, p. 1005, 2005.

[91] D. Kumar, P. d. Carvalho, M. Antunes, J. Henriques, M. Maldonado, R. Schmidt, and J. Habetha, "Wavelet transform and simplicity based heart murmur segmentation," in *2006 Computers in Cardiology*, pp. 173–176, IEEE, 2006.

[92] T. Higuchi, "Approach to an irregular time series on the basis of the fractal theory," *Physica D: Nonlinear Phenomena*, vol. 31, no. 2, pp. 277–283, 1988.

[93] W. Qiang, O. Sourina, and N. M. Khoa, "A fractal dimension based algorithm for neurofeedback games," *CGI 2010*, p. SP25, 2010.

[94] Y. Liu and O. Sourina, "Real-time fractal-based valence level recognition from eeg," in *Transactions on computational science XVIII*, pp. 101–120, Springer, 2013.

[95] T. L. Doyle, E. L. Dugan, B. Humphries, and R. U. Newton, "Discriminating between elderly and young using a fractal dimension analysis of centre of pressure," *International journal of medical sciences*, vol. 1, no. 1, p. 11, 2004.

[96] B. Qian and K. Rasheed, "Hurst exponent and financial market predictability," in *IASTED conference on Financial Engineering and Applications*, pp. 203–209, 2004.

[97] K. Harrar and M. Khider, "Texture analysis using multifractal spectrum," *International Journal of Modeling and Optimization*, vol. 4, no. 4, p. 336, 2014.

[98] H. Wendt and P. Abry, "Multifractality tests using bootstrapped wavelet leaders," *IEEE Transactions on Signal Processing*, vol. 55, no. 10, pp. 4811–4820, 2007.

[99] M. Soleymani, S. Asghari-Esfeden, M. Pantic, and Y. Fu, "Continuous emotion detection using eeg signals and facial expressions," in *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, 2014.

[100] H. Candra, M. Yuwono, R. Chai, A. Handojoseno, I. Elamvazuthi, H. T. Nguyen, and S. Su, "Investigation of window size in classification of eeg-emotion signal with wavelet entropy and support vector machine," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 7250–7253, IEEE, 2015.

[101] H. Abdullah and D. Cvetkovic, "Electrophysiological signals segmentation for eeg frequency bands and heart rate variability analysis," in *The 15th International Conference on Biomedical Engineering*, pp. 695–698, Springer, 2014.

[102] R. M. Rangayyan, *Biomedical signal analysis*, vol. 33. John Wiley & Sons, 2015.

[103] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, 1975.

[104] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[105] D. Novak, M. Mihelj, and M. Munih, "A survey of methods for data fusion and system adaptation using autonomic nervous system responses in physiological

computing," *Interacting with computers*, vol. 24, no. 3, pp. 154–172, 2012.

[106] B. J. Roach and D. H. Mathalon, "Event-related eeg time-frequency analysis: an overview of measures and an analysis of early gamma band phase locking in schizophrenia," *Schizophrenia bulletin*, vol. 34, no. 5, pp. 907–926, 2008.

[107] X.-W. Wang, D. Nie, and B.-L. Lu, "Emotional state classification from eeg data using machine learning approach," *Neurocomputing*, vol. 129, pp. 94–106, 2014.

[108] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[109] M. M. Mukaka, "A guide to appropriate use of correlation coefficient in medical research," *Malawi Medical Journal*, vol. 24, no. 3, pp. 69–71, 2012.

[110] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Aaai*, vol. 2, pp. 129–134, 1992.

[111] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[112] I. Kononenko, M. Robnik-Sikonja, and U. Pompe, "Relieff for estimation and discretization of attributes in classification, regression, and ilp problems," *Artificial intelligence: methodology, systems, applications*, pp. 31–40, 1996.

[113] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on neural networks*, vol. 5, no. 4, pp. 537–550, 1994.

[114] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009.

[115] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Math-*

ematical, Physical and Engineering Sciences, vol. 374, no. 2065, p. 20150202, 2016.

[116] D. Ballabio, "A matlab toolbox for principal component analysis and unsupervised exploration of data structure," *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 1–9, 2015.

[117] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis-a brief tutorial," *Institute for Signal and information Processing*, vol. 18, pp. 1–8, 1998.

[118] C. M. Bishop, "Linear models for classification," in *Pattern recognition and machine learning*, ch. 4, p. 188, springer, 2006.

[119] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[120] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[121] E. García-Gonzalo, Z. Fernández-Muñiz, P. García Nieto, A. Bernardo Sánchez, and M. Menéndez Fernández, "Hard-rock stability analysis for span design in entry-type excavations with learning classifiers," *Materials*, vol. 9, no. 7, p. 531, 2016.

[122] T. M. Cover, P. E. Hart, *et al.*, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[123] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, 2001.

[124] S. Arlot, A. Celisse, *et al.*, "A survey of cross-validation procedures for model selection," *Statistics surveys*, vol. 4, pp. 40–79, 2010.

[125] A. Kielar and M. F. Joanisse, "The role of semantic and phonological factors in word recognition: An erp cross-modal priming study of derivational morphology," *Neuropsychologia*, vol. 49, no. 2, pp. 161–177, 2011.

[126] R. Couceiro, R. Barbosa, G. Durães Duarte, J. Castelhano, C. Duarte, C. Teixeira, N. Laranjeiro, J. Medeiros, M. Castelo-Branco, P. Carvalho, and H. Madeira, "Spotting problematic code lines using nonintrusive programmers' biofeedback," in *ISSRE*, 2019. (Accepted).

Appendices

A

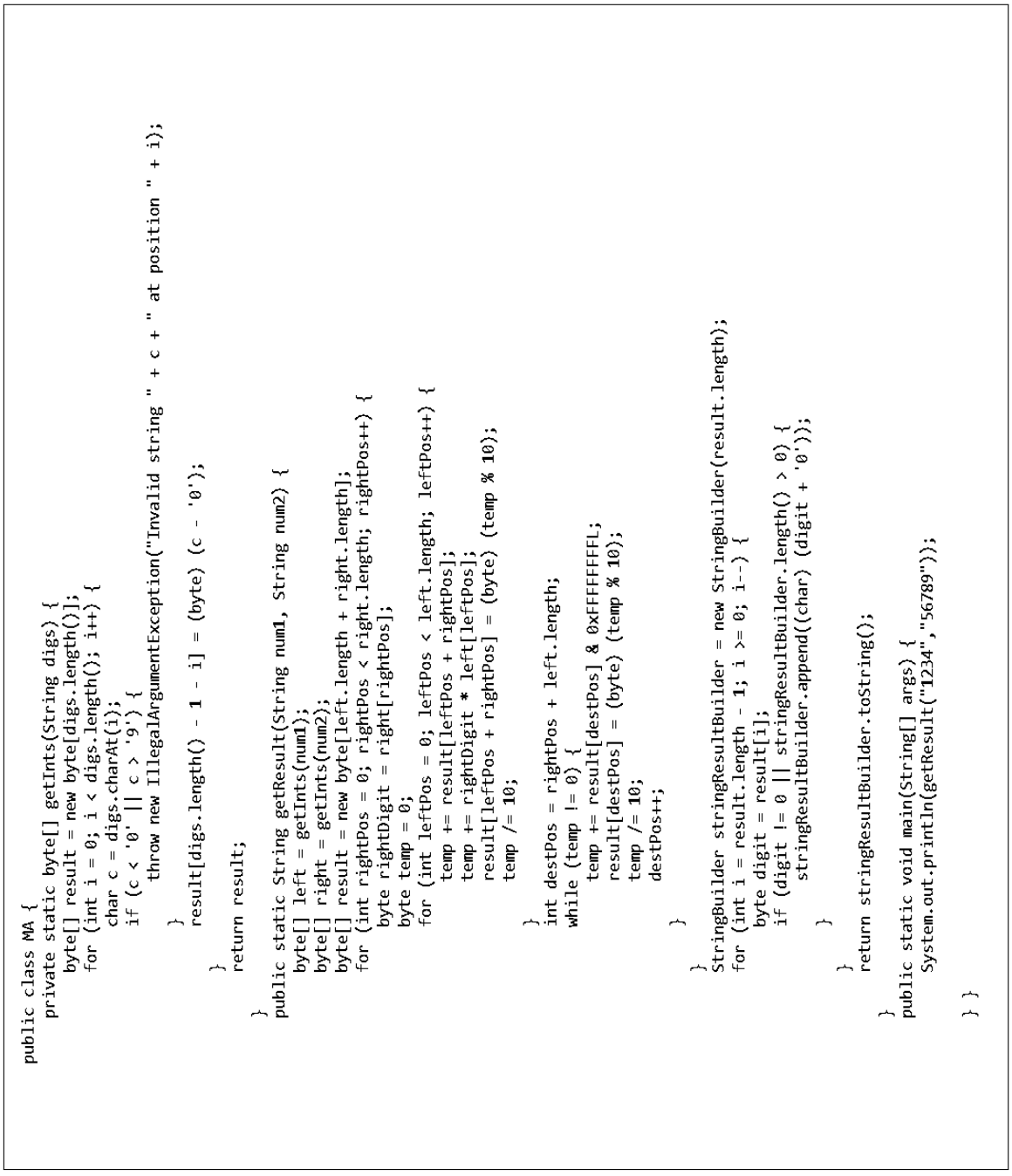
Experimental Protocol - Codes

A.1 Code 1 (Lowest complexity level)

```
public class ProcessNums {
    public static int getResult(int[] sequence, int lower, int upper) {
        int result = 0;
        if (sequence == null)
            return result;
        for (int n : sequence) {
            if (n>=lower && n<=upper)
                result++;
        }
        return result;
    }
    public static void main(String[] args) {
        int[] sequence = {-7, 1, 5, 2, -4, 3, 0};
        int result = getResult(sequence,2,4);
        System.out.println("Result = " + result);
    }
}
```

Figure A.1: Example of the code 1, which corresponds to the lowest complexity level, used in the experimental protocol.

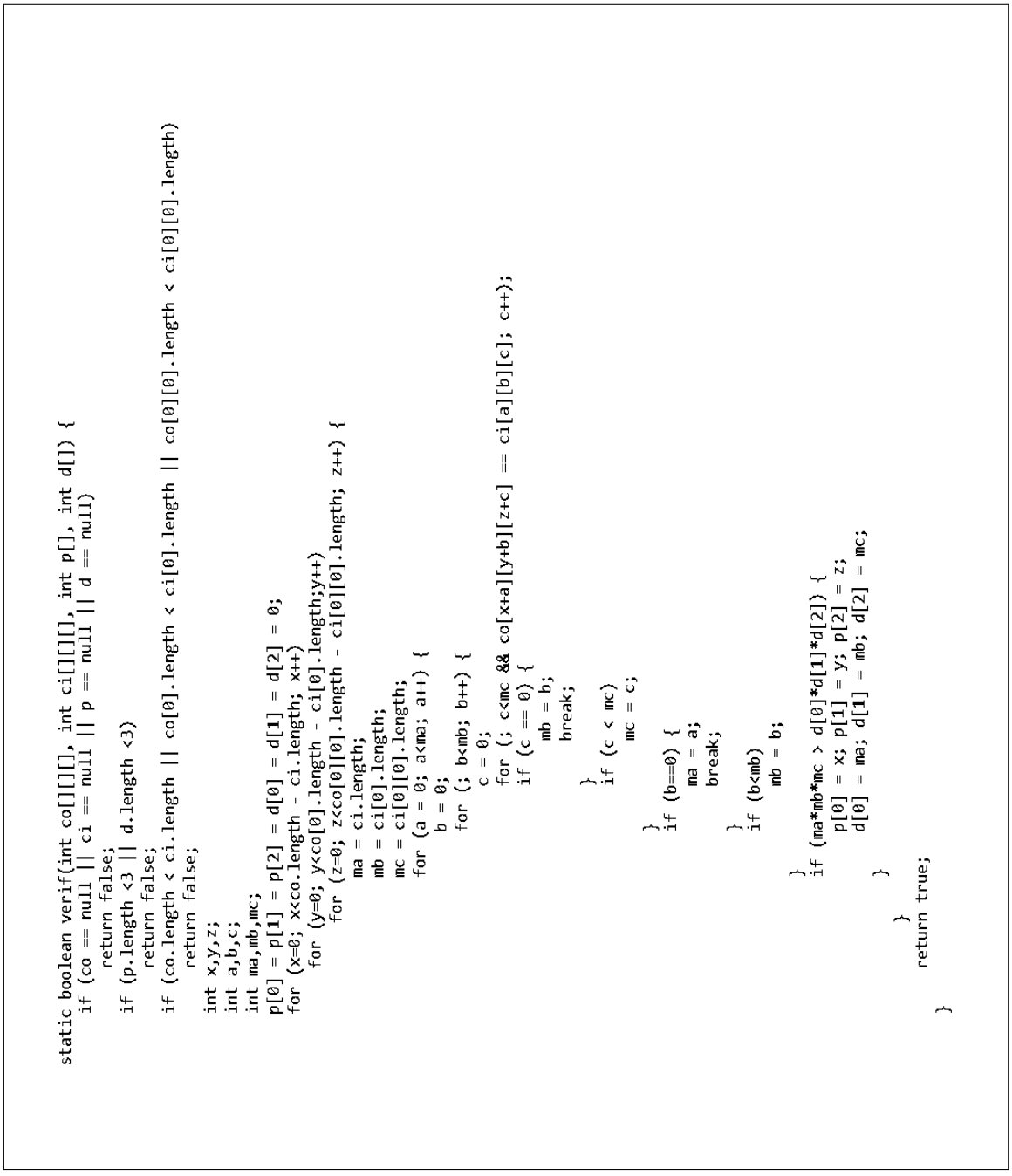
A.2 Code 2 (Middle complexity level)



```
public class MA {
    private static byte[] getInts(String digs) {
        byte[] result = new byte[digs.length()];
        for (int i = 0; i < digs.length(); i++) {
            char c = digs.charAt(i);
            if (c < '0' || c > '9') {
                throw new IllegalArgumentException("Invalid string " + c + " at position " + i);
            }
            result[digs.length() - 1 - i] = (byte) (c - '0');
        }
        return result;
    }
    public static String getResult(String num1, String num2) {
        byte[] left = getInts(num1);
        byte[] right = getInts(num2);
        byte[] result = new byte[left.length + right.length];
        for (int rightPos = 0; rightPos < right.length; rightPos++) {
            byte rightDigit = right[rightPos];
            byte temp = 0;
            for (int leftPos = 0; leftPos < left.length; leftPos++) {
                temp += result[leftPos + rightPos];
                temp += rightDigit * left[leftPos];
                result[leftPos + rightPos] = (byte) (temp % 10);
                temp /= 10;
            }
            int destPos = rightPos + left.length;
            while (temp != 0) {
                temp += result[destPos] & 0xFFFFFFFFL;
                result[destPos] = (byte) (temp % 10);
                temp /= 10;
                destPos++;
            }
        }
        StringBuilder stringResultBuilder = new StringBuilder(result.length);
        for (int i = result.length - 1; i >= 0; i--) {
            byte digit = result[i];
            if (digit != 0 || stringResultBuilder.length() > 0) {
                stringResultBuilder.append((char) (digit + '0'));
            }
        }
        return stringResultBuilder.toString();
    }
    public static void main(String[] args) {
        System.out.println(getResult("1234", "56789"));
    }
}
```

Figure A.2: Example of the code 2, which corresponds to the middle complexity level, used in the experimental protocol.

A.3 Code 3 (Highest complexity level)



```
static boolean verif(int co[][][], int ci[][][], int p[], int d[]) {
    if (co == null || ci == null || p == null || d == null)
        return false;
    if (p.length <3 || d.length <3)
        return false;
    if (co.length < ci.length || co[0].length < ci[0].length || co[0][0].length < ci[0][0].length)
        return false;
    int x,y,z;
    int a,b,c;
    int ma,mb,mc;
    p[0] = p[1] = p[2] = d[0] = d[1] = d[2] = 0;
    for (x=0; x<co.length - ci.length; x++)
        for (y=0; y<co[0].length - ci[0].length;y++)
            for (z=0; z<co[0][0].length - ci[0][0].length; z++) {
                ma = ci.length;
                mb = ci[0].length;
                mc = ci[0][0].length;
                for (a = 0; a<ma; a++) {
                    b = 0;
                    for (; b<mb; b++) {
                        c = 0;
                        for (; c<mc && co[x+a][y+b][z+c] == ci[a][b][c]; c++);
                        if (c == 0) {
                            mb = b;
                            break;
                        }
                        if (c < mc)
                            mc = c;
                    }
                    if (b==0) {
                        ma = a;
                        break;
                    }
                    if (b<mb)
                        mb = b;
                }
                if (ma*mb*mc > d[0]*d[1]*d[2]) {
                    p[0] = x; p[1] = y; p[2] = z;
                    d[0] = ma; d[1] = mb; d[2] = mc;
                }
            }
    return true;
}
```

Figure A.3: Example of the code 3, which corresponds to the highest complexity level, used in the experimental protocol.

B

Computational Time for Features Extraction

Table B.1: Computational time for linear and non-linear feature extraction using ASUS laptop equipped with a 2.2GHZ Intel Core i7 8th Gen processor, 256GB M.20 SATA III SSD and 16 GB RAM.

Window Size (s)	Computational time 13 Regions per Subject		Computational time Whole Scalp per Subject	
	Linear Features (minutes)	Non Linear Features (hours)	Linear Features (minutes)	Non Linear Features (hours)
1	7.71	23.20	35.58	107.09
2	1.55	26.91	7.12	124.21
3	1.58	28.47	7.27	131.40
4	0.68	31.00	3.13	143.10
5	0.64	32.53	2.92	150.13
6	0.61	33.91	2.80	156.50
7	0.52	40.99	2.38	189.19
8	0.48	34.84	2.22	160.79
9	0.45	33.52	2.07	154.71
10	0.64	34.44	2.93	158.95

C

Results

C.1 Study 1: 13 Regions Analysis

C.1.1 Code Complexity Analysis

Table C.1: Performance of the four different classifiers after using Kruskal–Wallis H test as feature selection method (100 features selected), for the multiclass classification scenario: Code 1 vs Code 2 vs Code 3.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^4)	C1	92.31 ± 19.71	98.46 ± 5.43	94.96 ± 19.82	93.72 ± 14.49	64.42 ± 16.76
	C2	59.62 ± 29.22	50.91 ± 21.72	67.51 ± 18.71	52.27 ± 21.25	
	C3	41.35 ± 27.33	52.26 ± 29.78	78.18 ± 20.45	43.36 ± 24.10	
FLDA	C1	94.23 ± 14.68	98.35 ± 8.40	98.35 ± 8.40	95.29 ± 10.58	64.10 ± 12.64
	C2	50.96 ± 30.40	47.30 ± 26.98	71.15 ± 14.48	46.17 ± 24.02	
	C3	47.12 ± 23.80	53.08 ± 22.87	75.16 ± 17.18	47.00 ± 18.77	
Naive B.	C1	93.27 ± 13.34	92.63 ± 15.28	91.86 ± 17.06	92.31 ± 12.71	61.54 ± 11.07
	C2	74.04 ± 31.21	45.13 ± 15.83	58.10 ± 13.06	55.11 ± 20.34	
	C3	17.31 ± 22.10	28.53 ± 36.07	86.70 ± 16.97	19.60 ± 23.19	
k-NN (k=1)	C1	90.38 ± 18.81	98.46 ± 5.43	97.76 ± 8.01	92.81 ± 13.61	62.50 ± 13.39
	C2	46.15 ± 29.74	44.47 ± 29.71	73.10 ± 13.63	43.53 ± 26.69	
	C3	50.96 ± 25.96	46.59 ± 21.92	69.64 ± 16.58	47.00 ± 20.50	

Table C.2: Performance of the four different classifiers after using ReliefF Algorithm as feature selection method (100 features selected), for the multiclass classification scenario: Code 1 vs Code 2 vs Code 3.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-8})	C1	83.65 ± 27.33	91.28 ± 21.40	87.50 ± 27.71	84.98 ± 22.85	59.62 ± 20.51
	C2	46.15 ± 43.41	36.56 ± 36.59	68.88 ± 28.42	38.50 ± 35.70	
	C3	49.04 ± 43.29	39.15 ± 37.44	69.62 ± 25.66	41.43 ± 36.41	
FLDA	C1	76.92 ± 31.56	80.49 ± 31.12	76.95 ± 34.82	75.41 ± 28.30	55.13 ± 20.56
	C2	36.54 ± 40.76	32.88 ± 38.96	70.29 ± 28.28	32.84 ± 36.34	
	C3	51.92 ± 44.12	39.23 ± 35.14	65.66 ± 28.47	43.10 ± 36.53	
Naive B.	C1	51.92 ± 47.92	49.76 ± 45.71	74.79 ± 39.36	49.05 ± 44.78	46.79 ± 29.07
	C2	30.77 ± 43.19	28.21 ± 39.14	67.13 ± 37.09	27.73 ± 38.20	
	C3	57.69 ± 45.15	41.64 ± 36.07	45.01 ± 40.38	44.18 ± 34.61	
k-NN (k=15)	C1	75.00 ± 38.08	71.04 ± 35.52	73.09 ± 32.05	71.57 ± 35.30	53.53 ± 25.62
	C2	28.85 ± 38.53	23.71 ± 32.17	70.86 ± 32.67	25.51 ± 34.00	
	C3	56.73 ± 40.96	42.69 ± 34.35	58.06 ± 33.70	46.19 ± 33.20	

Table C.3: Performance of the four different classifiers after using Normalized Mutual Information as feature selection method (50 features selected), for the multiclass classification scenario: Code 1 vs Code 2 vs Code 3.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-4})	C1	88.46 ± 19.01	98.46 ± 5.43	98.27 ± 6.16	91.71 ± 13.56	66.03 ± 17.47
	C2	69.23 ± 36.27	51.87 ± 28.48	64.77 ± 24.49	55.03 ± 26.32	
	C3	40.38 ± 38.78	49.84 ± 42.32	83.34 ± 20.21	39.81 ± 33.90	
FLDA	C1	91.35 ± 18.63	98.72 ± 6.54	97.44 ± 13.07	93.44 ± 13.87	63.78 ± 15.8
	C2	57.69 ± 42.29	41.57 ± 30.32	66.83 ± 22.06	45.24 ± 30.52	
	C3	42.31 ± 38.58	43.14 ± 39.41	78.36 ± 23.32	38.87 ± 32.16	
Naive B.	C1	82.69 ± 26.24	87.86 ± 23.61	81.30 ± 31.94	82.35 ± 22.13	56.09 ± 18.04
	C2	56.73 ± 37.79	38.68 ± 22.62	57.90 ± 24.39	44.23 ± 25.88	
	C3	28.85 ± 39.17	26.67 ± 34.70	77.92 ± 24.55	25.98 ± 33.79	
k-NN (k=7)	C1	69.23 ± 36.95	81.28 ± 36.61	92.65 ± 21.97	72.47 ± 34.80	59.94 ± 19.86
	C2	60.58 ± 35.48	48.21 ± 29.85	61.45 ± 25.64	49.85 ± 26.26	
	C3	50.00 ± 39.37	49.45 ± 35.48	73.56 ± 28.00	45.84 ± 32.46	

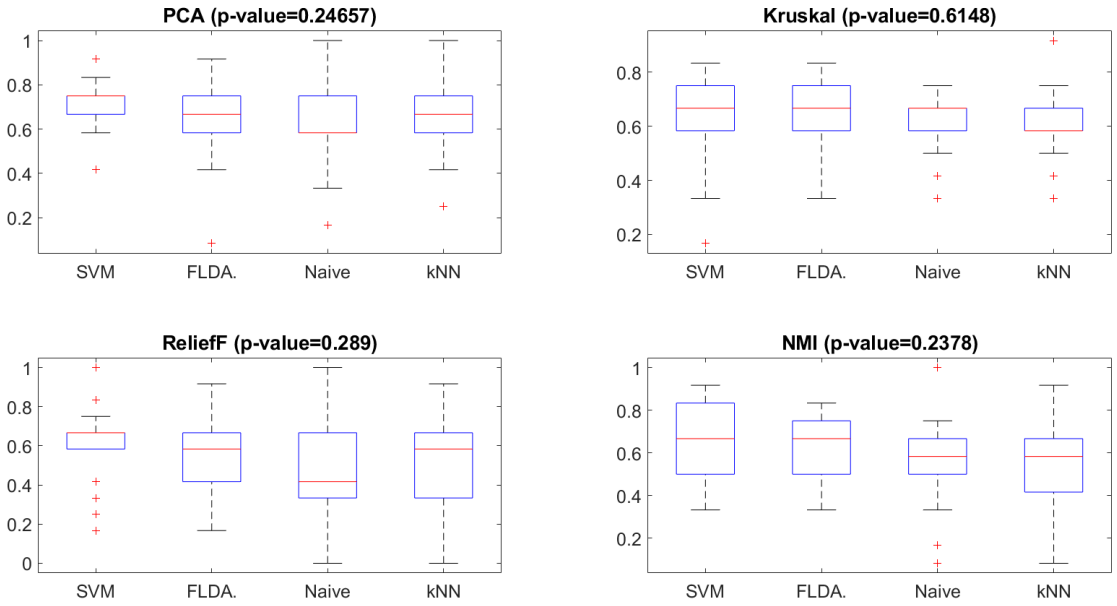


Figure C.1: Boxplots and corresponding Kruskal-Wallis p-value indicating the existence of statistical differences among the four classifiers accuracies. A p-value was obtained for each of the feature selection/reduction methods used to distinguish the three code tasks.

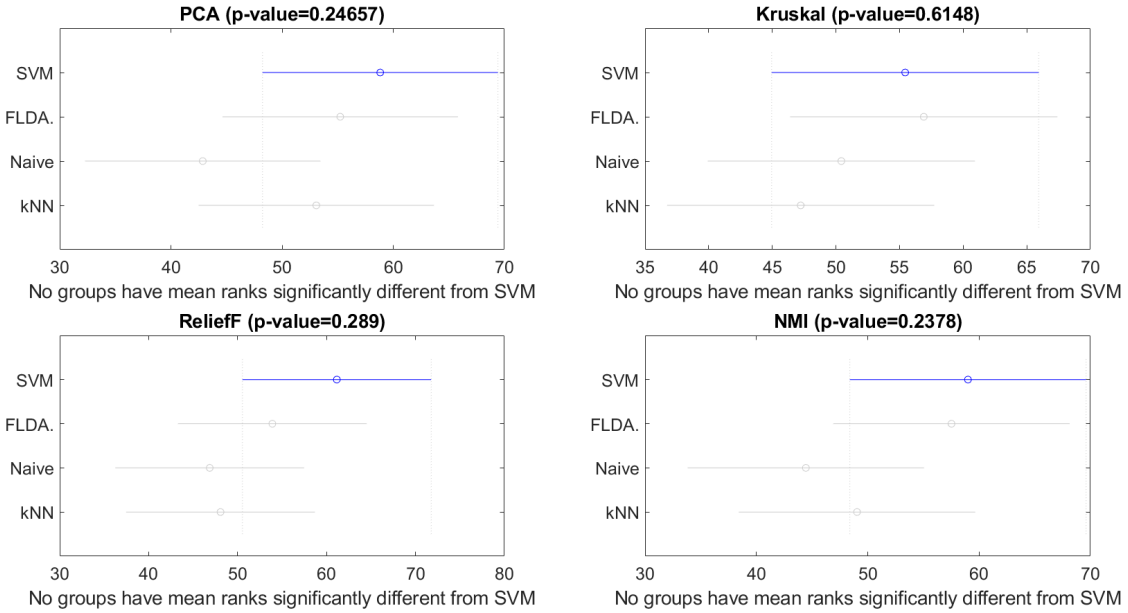


Figure C.2: Multiple comparison test and respective Kruskal-Wallis p-value indicating the existence of statistical differences among the four classifiers accuracies. A p-value was obtained for each of the feature selection/reduction methods used to distinguish the three code tasks.

Table C.4: Performance of the four different classifiers after using Mann-Whiney U-test as feature selection method (100 features selected), for the binary classification scenario: Code task vs Code task.

Classifier	Binary Classification	Evaluation Metric (%)				
		Recall	**Precision**	**Specificity**	**F-measure**	**Accuracy**
SVM (C=2^1)	**C1 vs C2**	95.19 ± 12.29	98.72 ± 6.54	98.08 ± 9.81	96.30 ± 8.41	96.63 ± 7.55
	C1 vs C3	95.19 ± 12.29	99.04 ± 4.90	99.04 ± 4.90	96.66 ± 8.61	97.12 ± 7.34
	C2 vs C3	53.85 ± 37.88	50.60 ± 35.56	51.92 ± 42.97	48.18 ± 31.66	52.88 ± 24.57
FLDA	**C1 vs C2**	94.23 ± 12.86	96.41 ± 8.89	95.19 ± 12.29	94.47 ± 8.72	94.71 ± 8.04
	C1 vs C3	91.35 ± 18.63	93.93 ± 13.86	93.27 ± 16.67	91.64 ± 15.45	92.31 ± 13.27
	C2 vs C3	46.15 ± 43.41	36.92 ± 35.56	50.96 ± 39.67	39.11 ± 36.20	48.56 ± 25.33
Naive B.	**C1 vs C2**	93.27 ± 13.34	97.95 ± 7.49	97.12 ± 10.79	94.77 ± 8.82	95.19 ± 7.97
	C1 vs C3	94.23 ± 12.86	94.81 ± 12.04	94.23 ± 12.86	94.18 ± 11.36	94.23 ± 11.31
	C2 vs C3	50.00 ± 44.72	39.61 ± 35.99	50.00 ± 39.37	41.96 ± 35.84	50.00 ± 24.49
k-NN (k=1)	**C1 vs C2**	92.31 ± 15.44	98.72 ± 6.54	98.08 ± 9.81	94.47 ± 10.28	95.19 ± 8.72
	C1 vs C3	92.31 ± 19.71	98.72 ± 6.54	99.04 ± 4.90	94.21 ± 15.34	95.67 ± 11.15
	C2 vs C3	58.65 ± 34.60	50.74 ± 27.01	50.96 ± 34.26	51.98 ± 26.45	54.81 ± 19.06

Table C.5: Performance of the four different classifiers after using ReliefF Algorithm as feature selection method (100 features selected), for the binary classification scenario: Code task vs Code task.

Classifier	Binary Classification	Evaluation Metric (%)				
		Recall	**Precision**	**Specificity**	**F-measure**	**Accuracy**
SVM ($C=2^1$)	**C1 vs C2**	96.15 ± 15.32	98.46 ± 5.43	98.08 ± 6.79	96.29 ± 12.15	97.12 ± 8.15
	C1 vs C3	94.23 ± 12.86	97.69 ± 8.63	97.12 ± 10.79	95.36 ± 9.79	95.67 ± 9.32
	C2 vs C3	66.35 ± 45.79	41.30 ± 31.24	41.35 ± 39.96	49.99 ± 35.16	53.85 ± 19.93
FLDA	**C1 vs C2**	92.31 ± 19.71	95.93 ± 12.07	93.27 ± 20.69	92.05 ± 15.49	92.79 ± 13.31
	C1 vs C3	95.19 ± 14.18	96.04 ± 10.26	94.23 ± 16.29	94.56 ± 10.53	94.71 ± 10.11
	C2 vs C3	55.77 ± 44.33	45.60 ± 38.33	51.92 ± 42.97	47.81 ± 37.18	53.85 ± 27.56
Naive B.	**C1 vs C2**	90.38 ± 27.46	85.83 ± 26.62	84.62 ± 30.88	86.35 ± 26.36	87.50 ± 21.51
	C1 vs C3	81.73 ± 31.27	85.13 ± 27.33	79.81 ± 36.76	80.33 ± 27.87	80.77 ± 27.67
	C2 vs C3	45.19 ± 47.44	39.78 ± 41.66	57.69 ± 46.24	38.89 ± 39.73	51.44 ± 29.44
k-NN (k=15)	**C1 vs C2**	86.54 ± 32.58	90.26 ± 27.60	97.12 ± 10.79	86.50 ± 30.32	91.83 ± 16.56
	C1 vs C3	86.54 ± 28.49	90.66 ± 27.98	96.15 ± 15.32	87.78 ± 27.41	91.35 ± 16.87
	C2 vs C3	41.35 ± 44.13	28.62 ± 30.03	48.08 ± 41.18	31.74 ± 32.06	44.71 ± 16.27

Table C.6: Performance of the four different classifiers after using Normalized Mutual Information as feature selection method (50 features selected), for the binary classification scenario: Code task vs Code task.

Classifier	Binary Classification	Evaluation Metric (%)				
		Recall	**Precision**	**Specificity**	**F-measure**	**Accuracy**
SVM ($C=2^{-3}$)	**C1 vs C2**	96.15 ± 9.20	97.50 ± 7.11	97.12 ± 8.15	96.54 ± 6.85	96.63 ± 6.67
	C1 vs C3	98.08 ± 6.79	95.93 ± 12.07	93.27 ± 20.69	96.38 ± 8.19	95.67 ± 10.57
	C2 vs C3	61.54 ± 40.14	51.90 ± 33.74	53.85 ± 37.21	53.82 ± 33.48	57.69 ± 24.77
FLDA	**C1 vs C2**	92.31 ± 18.40	96.92 ± 7.36	96.15 ± 9.20	93.05 ± 13.51	94.23 ± 9.51
	C1 vs C3	96.15 ± 11.60	96.81 ± 9.74	95.19 ± 15.84	95.72 ± 9.02	95.67 ± 9.32
	C2 vs C3	58.65 ± 39.33	50.02 ± 33.87	52.88 ± 40.20	51.15 ± 32.40	55.77 ± 23.51
Naive B.	**C1 vs C2**	89.42 ± 17.57	88.41 ± 18.71	80.77 ± 32.64	86.34 ± 13.77	85.10 ± 16.21
	C1 vs C3	95.19 ± 10.05	92.82 ± 11.46	90.38 ± 15.93	93.15 ± 7.32	92.79 ± 8.04
	C2 vs C3	85.58 ± 26.62	50.64 ± 15.66	17.31 ± 28.96	61.83 ± 15.69	51.44 ± 10.20
k-NN (k=5)	**C1 vs C2**	74.04 ± 34.99	92.31 ± 27.17	99.04 ± 4.90	79.05 ± 31.70	86.54 ± 18.68
	C1 vs C3	83.65 ± 27.33	93.33 ± 20.74	96.15 ± 11.60	86.07 ± 23.86	89.90 ± 13.70
	C2 vs C3	45.19 ± 29.17	40.93 ± 27.46	44.23 ± 33.40	40.95 ± 24.94	44.71 ± 16.65

C.1.2 Code Complexity and Resting Analysis

Table C.7: Performance obtained for the four classifiers after using Kruskal–Wallis H test as feature selection method (100 features selected), for the multiclass classification scenario (Code 1 vs Code 2 vs Code 3 vs Resting Control).

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^5)	C1	77.88 ± 26.76	85.45 ± 19.17	91.45 ± 11.23	79.20 ± 21.21	62.98 ± 15.1
	C2	56.73 ± 32.06	50.48 ± 27.45	77.13 ± 14.29	51.03 ± 26.06	
	C3	39.42 ± 32.54	41.79 ± 33.29	81.66 ± 15.92	38.34 ± 29.45	
	Control	77.88 ± 23.80	75.78 ± 19.64	84.09 ± 15.63	73.69 ± 17.97	
FLDA	C1	75.96 ± 25.96	88.35 ± 18.18	92.95 ± 12.11	78.97 ± 20.08	61.54 ± 15.58
	C2	51.92 ± 29.09	53.51 ± 31.13	77.37 ± 16.20	49.57 ± 24.53	
	C3	41.35 ± 28.23	53.11 ± 32.04	82.36 ± 15.89	43.45 ± 25.59	
	Control	76.92 ± 23.37	66.27 ± 22.82	79.13 ± 15.39	69.34 ± 19.85	
Naive B,	C1	70.19 ± 35.37	74.17 ± 32.39	89.95 ± 13.36	69.63 ± 31.51	53.85 ± 13.36
	C2	53.85 ± 39.17	39.98 ± 26.16	72.79 ± 15.73	44.59 ± 30.25	
	C3	17.31 ± 24.26	27.18 ± 37.73	86.06 ± 15.04	18.20 ± 22.87	
	Control	74.04 ± 23.96	56.75 ± 21.39	66.47 ± 25.75	60.74 ± 16.26	
k-NN (k=15)	C1	63.46 ± 38.88	82.12 ± 36.39	98.49 ± 4.28	68.07 ± 36.18	59.62 ± 15.43
	C2	49.04 ± 30.40	42.48 ± 23.33	74.09 ± 16.83	43.39 ± 23.03	
	C3	41.35 ± 32.36	38.77 ± 25.49	76.44 ± 15.03	37.36 ± 24.81	
	Control	84.62 ± 23.53	72.30 ± 24.19	78.06 ± 22.91	76.00 ± 21.25	

Table C.8: Performance obtained for the four classifiers after using ReliefF Algorithm as feature selection method (100 features selected), for the multiclass classification scenario (Code 1 vs Code 2 vs Code 3 vs Resting Control).

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-4})	C1	79.81 ± 31.64	90.77 ± 21.34	96.34 ± 7.38	81.42 ± 26.14	62.74 ± 15.15
	C2	47.12 ± 38.94	45.21 ± 39.01	81.84 ± 16.76	43.14 ± 34.55	
	C3	43.27 ± 39.72	40.60 ± 36.03	80.68 ± 16.44	39.12 ± 33.74	
	Control	80.77 ± 22.70	70.06 ± 28.27	79.87 ± 21.38	72.72 ± 22.55	
FLDA	C1	72.12 ± 31.88	90.38 ± 20.83	96.42 ± 6.58	76.09 ± 25.66	61.54 ± 19.74
	C2	41.35 ± 38.69	41.62 ± 37.90	80.37 ± 20.55	39.28 ± 35.70	
	C3	59.62 ± 43.63	46.76 ± 37.20	76.37 ± 20.92	50.67 ± 37.38	
	Control	73.08 ± 31.56	67.65 ± 31.17	81.22 ± 21.00	67.39 ± 27.68	
Naive B,	C1	76.92 ± 36.00	75.92 ± 31.71	86.20 ± 23.39	72.57 ± 31.95	54.09 ± 14.57
	C2	48.08 ± 43.54	38.65 ± 36.05	75.48 ± 20.00	38.99 ± 33.86	
	C3	31.73 ± 43.33	27.28 ± 36.80	84.39 ± 19.44	28.07 ± 37.37	
	Control	59.62 ± 33.22	49.56 ± 26.57	73.33 ± 20.14	52.15 ± 26.95	
k-NN (k=1)	C1	47.12 ± 38.29	63.01 ± 45.78	96.02 ± 7.89	52.53 ± 40.03	45.91 ± 15.9
	C2	35.58 ± 40.73	23.55 ± 27.96	70.79 ± 25.04	27.18 ± 30.25	
	C3	43.27 ± 37.12	35.89 ± 28.22	64.33 ± 22.86	35.87 ± 26.77	
	Control	57.69 ± 26.24	48.71 ± 21.81	63.12 ± 25.07	50.75 ± 20.81	

Table C.9: Performance obtained for the four classifiers after using Normalized Mutual Information as feature selection method (50 features selected), for the multiclass classification scenario (Code 1 vs Code 2 vs Code 3 vs Resting Control).

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-5})	C1	72.12 ± 35.59	82.82 ± 32.83	95.38 ± 10.58	74.13 ± 32.31	63.94 ± 14.61
	C2	53.85 ± 36.53	48.06 ± 31.42	77.01 ± 19.15	46.62 ± 27.45	
	C3	50.00 ± 38.08	46.97 ± 35.15	82.16 ± 17.14	44.78 ± 31.40	
	Control	79.81 ± 26.48	71.78 ± 25.30	84.02 ± 16.33	73.14 ± 22.36	
FLDA	C1	73.08 ± 38.68	77.24 ± 35.43	93.73 ± 8.91	72.43 ± 35.33	59.62 ± 11.49
	C2	46.15 ± 38.53	34.89 ± 27.26	77.14 ± 14.53	38.44 ± 29.63	
	C3	37.50 ± 33.35	41.95 ± 36.93	82.66 ± 15.45	35.69 ± 28.07	
	Control	81.73 ± 18.11	71.85 ± 24.40	77.91 ± 23.53	73.31 ± 17.10	
Naive B,	C1	62.50 ± 39.53	70.00 ± 38.44	88.54 ± 21.62	63.66 ± 36.90	48.80 ± 18.2
	C2	30.77 ± 37.62	24.75 ± 33.07	70.10 ± 23.09	25.76 ± 31.79	
	C3	25.96 ± 37.07	23.49 ± 32.34	83.96 ± 18.99	23.84 ± 33.19	
	Control	75.96 ± 26.91	56.58 ± 28.21	60.97 ± 33.89	59.42 ± 19.56	
k-NN (k=1)	C1	54.81 ± 30.84	73.85 ± 31.91	90.72 ± 11.34	60.28 ± 28.57	50.72 ± 13.62
	C2	34.62 ± 31.68	32.27 ± 29.33	70.07 ± 16.89	30.62 ± 24.89	
	C3	57.69 ± 32.23	53.74 ± 28.85	70.84 ± 22.73	50.51 ± 23.84	
	Control	55.77 ± 27.67	49.85 ± 21.20	73.27 ± 15.54	51.36 ± 23.05	

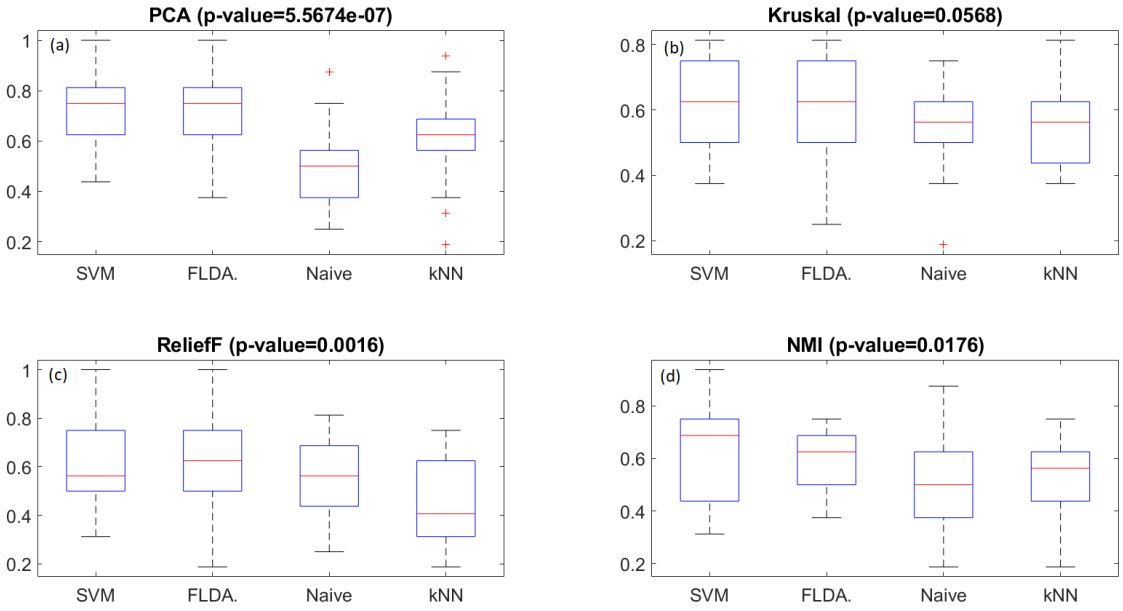


Figure C.3: Boxplots and corresponding statistical test p-value indicating the existence of statistical differences among the four classifiers accuracies. A p-value was obtained for each of the feature selection/reduction methods used to distinguish the three code tasks and the resting task. The statistical tests performed were: (a) and (c) Analysis of Variance (ANOVA); (b) and (d) Kruskal-Wallis test.

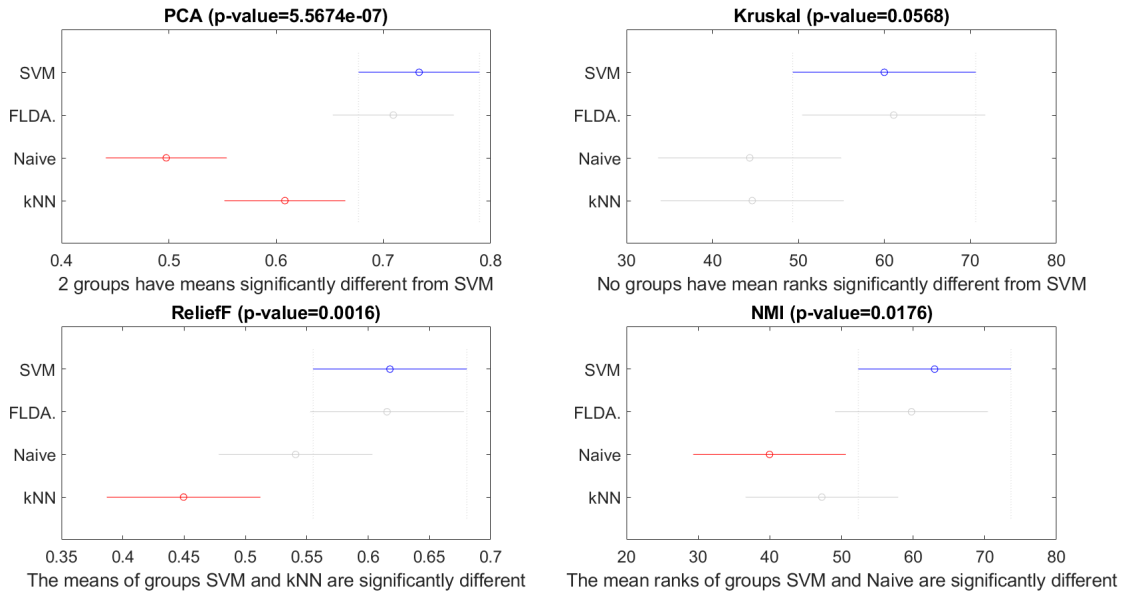


Figure C.4: Multiple comparison test and respective p-value of the statistical differences between the different classifiers for each method of feature selection/reduction, as classification models of the three codes and resting tasks.

Table C.10: Performance of the four different classifiers after using Mann–Whitney U test as feature selection method (100 features selected), for each binary classification scenario: Code task vs respective Resting Control task.

Classifier	Binary Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-8})	C1 vs Control	70.19 ± 36.76	70.50 ± 34.37	70.19 ± 34.65	66.89 ± 32	70.19 ± 25.27
	C2 vs Control	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	C3 vs Control	99.04 ± 4.90	98.46 ± 5.43	98.08 ± 6.79	98.60 ± 4.00	98.56 ± 4.07
FLDA	C1 vs Control	55.77 ± 33.40	57.86 ± 29.10	64.42 ± 27.54	53.89 ± 26.98	60.10 ± 17.33
	C2 vs Control	99.04 ± 4.90	99.23 ± 3.92	99.04 ± 4.90	99.02 ± 3.48	99.04 ± 3.40
	C3 vs Control	95.19 ± 10.05	96.99 ± 8.72	96.15 ± 11.60	95.64 ± 7.67	95.67 ± 7.86
Naive B.	C1 vs Control	72.12 ± 33.41	69.33 ± 32.35	64.42 ± 37.53	67.17 ± 28.67	68.27 ± 24.30
	C2 vs Control	99.04 ± 4.90	100.00 ± 0.00	100.00 ± 0.00	99.45 ± 2.80	99.52 ± 2.45
	C3 vs Control	91.35 ± 21.15	97.18 ± 8.26	96.15 ± 11.60	92.11 ± 16.54	93.75 ± 11.32
k-NN (k=7)	C1 vs Control	68.27 ± 35.75	66.19 ± 32.22	65.38 ± 34.70	64.22 ± 29.96	66.83 ± 24.22
	C2 vs Control	100.00 ± 0.00	99.23 ± 3.92	99.04 ± 4.90	99.57 ± 2.18	99.52 ± 2.45
	C3 vs Control	100.00 ± 0.00	96.92 ± 7.36	96.15 ± 9.20	98.29 ± 4.09	98.08 ± 4.60

Table C.11: Performance of the four different classifiers after using ReliefF Algorithm as feature selection method (100 features selected), for each binary classification scenario: Code task vs respective Resting Control task.

Classifier	Binary Classification	Evaluation Metric (%)				
		Recall	**Precision**	**Specificity**	**F-measure**	**Accuracy**
SVM (OAO,C=2^0)	**C1 vs Control**	55.77 ± 41.42	47.42 ± 37.49	59.62 ± 39.42	49.87 ± 37.36	57.69 ± 26.24
	C2 vs Control	100.00 ± 0.00	99.23 ± 3.92	99.04 ± 4.90	99.57 ± 2.18	99.52 ± 2.45
	C3 vs Control	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
FLDA	**C1 vs Control**	50.00 ± 40.00	49.96 ± 38.67	57.69 ± 39.86	47.32 ± 35.96	53.85 ± 28.67
	C2 vs Control	100.00 ± 0.00	99.23 ± 3.92	99.04 ± 4.90	99.57 ± 2.18	99.52 ± 2.45
	C3 vs Control	99.04 ± 4.90	98.08 ± 9.81	96.15 ± 19.61	98.17 ± 7.01	97.60 ± 10.01
Naive B.	**C1 vs Control**	27.88 ± 38.29	29.94 ± 39.06	56.73 ± 41.57	27.21 ± 36.05	42.31 ± 30.22
	C2 vs Control	95.19 ± 15.84	98.08 ± 9.81	96.15 ± 19.61	95.31 ± 13.49	95.67 ± 12.22
	C3 vs Control	96.15 ± 13.59	100.00 ± 0.00	100.00 ± 0.00	97.44 ± 9.06	98.08 ± 6.79
k-NN (k=1)	**C1 vs Control**	65.38 ± 33.97	51.14 ± 25.03	42.31 ± 30.63	55.78 ± 26.21	53.85 ± 21.73
	C2 vs Control	100.00 ± 0.00	97.58 ± 9.13	96.15 ± 15.32	98.52 ± 5.69	98.08 ± 7.66
	C3 vs Control	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00

Table C.12: Performance of the four different classifiers after using Normalized Mutual Information as feature selection method (50 features selected), for each binary classification scenario: Code task vs respective Resting Control task.

Classifier	Binary Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-5})	C1 vs Control	57.69 ± 38.58	49.62 ± 29.94	52.88 ± 34.88	50.76 ± 30.57	55.29 ± 21.84
	C2 vs Control	100.00 ± 0.00	99.23 ± 3.92	99.04 ± 4.90	99.57 ± 2.18	99.52 ± 2.45
	C3 vs Control	99.04 ± 4.90	100.00 ± 0.00	100.00 ± 0.00	99.45 ± 2.80	99.52 ± 2.45
FLDA	C1 vs Control	61.54 ± 35.52	53.52 ± 30.13	49.04 ± 34.26	54.67 ± 28.82	55.29 ± 24.54
	C2 vs Control	100.00 ± 0.00	99.23 ± 3.92	99.04 ± 4.90	99.57 ± 2.18	99.52 ± 2.45
	C3 vs Control	98.08 ± 9.81	99.23 ± 3.92	99.04 ± 4.90	98.29 ± 6.81	98.56 ± 5.39
Naive B.	C1 vs Control	70.19 ± 30.84	57.74 ± 26.07	46.15 ± 39.81	60.40 ± 23.67	58.17 ± 20.29
	C2 vs Control	98.08 ± 9.81	100.00 ± 0.00	100.00 ± 0.00	98.72 ± 6.54	99.04 ± 4.90
	C3 vs Control	91.35 ± 21.15	100.00 ± 0.00	100.00 ± 0.00	93.74 ± 16.48	95.67 ± 10.57
k-NN (k=15)	C1 vs Control	64.42 ± 30.14	53.32 ± 21.23	39.42 ± 32.54	55.54 ± 20.69	51.92 ± 19.90
	C2 vs Control	100.00 ± 0.00	84.51 ± 12.85	78.85 ± 19.61	91.09 ± 7.70	89.42 ± 9.81
	C3 vs Control	99.04 ± 4.90	90.81 ± 14.12	86.54 ± 22.62	94.02 ± 8.72	92.79 ± 11.28

C.2 Study 2: Whole Scalp Analysis

C.2.1 Code Complexity and Resting Analysis

Table C.13: Performance of the four different classifiers after using Kruskal–Wallis H test as feature selection method (100 features selected), for the multiclass classification scenario: Code 1 vs Code 2 vs Code 3 vs Resting Control.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-5})	C1	76.92 ± 28.22	98.72 ± 6.54	99.52 ± 2.45	83.26 ± 22.27	67.31 ± 12.54
	C2	53.85 ± 32.93	47.10 ± 27.81	79.29 ± 15.71	47.12 ± 24.70	
	C3	46.15 ± 39.81	39.47 ± 30.84	82.36 ± 15.04	40.21 ± 31.20	
	Control	92.31 ± 11.77	79.40 ± 16.02	85.49 ± 12.82	84.45 ± 11.61	
FLDA	C1	70.19 ± 35.37	88.85 ± 28.33	98.01 ± 5.94	75.19 ± 31.33	60.82 ± 13.64
	C2	44.23 ± 34.86	40.40 ± 27	77.49 ± 14.75	39.54 ± 26.81	
	C3	40.38 ± 34.70	40.22 ± 35.48	81.52 ± 15.24	37.64 ± 30.94	
	Control	88.46 ± 12.71	69.16 ± 19.98	74.92 ± 20.17	75.59 ± 13.09	
Naive B,	C1	76.92 ± 25.42	96.54 ± 10.93	98.33 ± 4.76	83.28 ± 18.89	59.62 ± 11.89
	C2	50.96 ± 36.39	43.75 ± 29.22	75.08 ± 17.93	43.25 ± 26.16	
	C3	26.92 ± 35.30	31.28 ± 37.19	88.43 ± 14.85	25.64 ± 29.10	
	Control	83.65 ± 18.63	63.08 ± 22.22	71.72 ± 19.64	68.82 ± 14.85	
k-NN (k=13)	C1	77.88 ± 27.68	99.23 ± 3.92	99.45 ± 2.80	84.04 ± 20.66	65.14 ± 14.38
	C2	55.77 ± 23.78	49.79 ± 22.15	75.89 ± 16.91	49.83 ± 17.68	
	C3	43.27 ± 37.12	39.83 ± 31.51	80.43 ± 12.69	38.73 ± 29.77	
	Control	83.65 ± 22.30	78.42 ± 20.04	86.36 ± 14.28	79.62 ± 18.72	

Table C.14: Performance of the four different classifiers after using ReliefF Algorithm as feature selection method (100 features selected), for the multiclass classification scenario: Code 1 vs Code 2 vs Code 3 vs Control.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-6})	C1	75.00 ± 30.00	86.57 ± 25.95	93.19 ± 14.29	77.43 ± 26.65	62.98 ± 15.4
	C2	50.00 ± 45.83	35.67 ± 35.20	79.29 ± 18.79	40.17 ± 37.08	
	C3	50.96 ± 39.67	46.67 ± 37.84	82.08 ± 16.77	45.88 ± 34.79	
	Control	75.96 ± 26.91	73.08 ± 23.06	83.21 ± 16.09	71.24 ± 20.99	
FLDA	C1	69.23 ± 31.87	84.94 ± 25.06	90.55 ± 16.56	73.12 ± 27.65	54.09 ± 16.86
	C2	44.23 ± 40.81	35.54 ± 36.48	71.60 ± 24.81	36.57 ± 33.35	
	C3	40.38 ± 37.47	40.77 ± 39.31	79.35 ± 18.29	37.05 ± 32.74	
	Control	62.50 ± 35.53	58.24 ± 33.06	77.03 ± 22.11	56.30 ± 29.20	
Naive B,	C1	78.85 ± 31.38	69.46 ± 29.02	76 ± 27.80	70.18 ± 25.86	49.28 ± 15.14
	C2	68.27 ± 43.91	35.49 ± 25.17	55.11 ± 21.41	45.01 ± 29.46	
	C3	10.58 ± 21.42	21.96 ± 39.99	90.37 ± 19.14	12.47 ± 22.82	
	Control	39.42 ± 30.14	56.97 ± 40.22	82.88 ± 24.46	41.95 ± 27.58	
k-NN (k=1)	C1	45.19 ± 38.74	60.00 ± 44.05	91.85 ± 16.80	49.38 ± 38.68	49.04 ± 16.83
	C2	39.42 ± 38.84	38.22 ± 36.13	74.50 ± 22.02	36.47 ± 33.90	
	C3	54.81 ± 39.38	46.39 ± 34.66	70.36 ± 26.64	47.86 ± 33.73	
	Control	56.73 ± 29.63	41.50 ± 22.80	61.90 ± 23.13	45.81 ± 22.02	

Table C.15: Performance of the four different classifiers after using Normalized Mutual Information as feature selection method (50 features selected), for the multiclass classification scenario: Code 1 vs Code 2 vs Code 3 vs Control.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
SVM (OAO,C=2^{-4})	C1	79.81 ± 31.64	86.22 ± 27.42	96.72 ± 5.59	81.04 ± 28.40	65.63 ± 12.41
	C2	64.42 ± 36.18	49.84 ± 30.88	75.74 ± 17.14	53.25 ± 27.76	
	C3	35.58 ± 40.11	31.92 ± 33.48	84.77 ± 13.32	31.54 ± 32.96	
	Control	82.69 ± 18.40	82.84 ± 21.39	86.89 ± 20.44	80.20 ± 15.76	
FLDA	C1	69.23 ± 36.95	83.59 ± 32.51	97.23 ± 6.02	72.75 ± 33.69	58.89 ± 12.27
	C2	51.92 ± 39.95	42.86 ± 31.58	73.61 ± 18.70	42.40 ± 28.48	
	C3	37.50 ± 38.24	35.76 ± 34.23	82.42 ± 15.73	33.80 ± 31.46	
	Control	76.92 ± 26.38	64.87 ± 25.35	77.05 ± 20.84	67.45 ± 21.14	
Naive B,	C1	78.85 ± 30.57	89.49 ± 23.70	95.20 ± 10.49	81.40 ± 26.78	56.01 ± 14.2
	C2	43.27 ± 34.32	47.52 ± 38.41	78.99 ± 19.06	40.54 ± 29.29	
	C3	23.08 ± 36.69	21.35 ± 33.33	88.72 ± 15.50	19.85 ± 29.82	
	Control	78.85 ± 25.19	57.62 ± 26.44	64.75 ± 26.56	61.03 ± 18.93	
k-NN (k=1)	C1	71.15 ± 30.57	88.10 ± 23.77	93.19 ± 13.12	75.01 ± 25.58	51.44 ± 13.95
	C2	47.12 ± 31.09	38.08 ± 26.65	67.94 ± 18.27	39.99 ± 25.30	
	C3	45.19 ± 38.74	32.75 ± 26.63	69.55 ± 15.48	37.22 ± 30.40	
	Control	42.31 ± 25.27	56.03 ± 29.98	79.94 ± 18.55	43.77 ± 20.19	

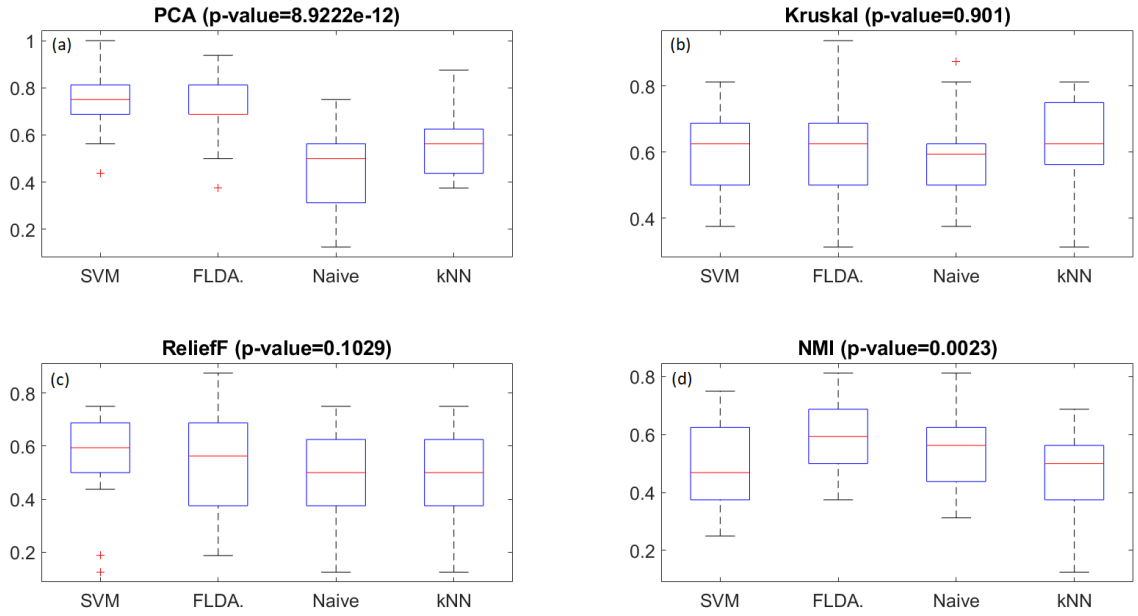


Figure C.5: Boxplots and corresponding statistical test p-value indicating the existence of statistical differences among the four classifiers accuracies. A p-value was obtained for each of the feature selection/reduction methods used to distinguish the three code and the resting tasks. The statistical tests performed were: (a), (b) and (d) Analysis of Variance (ANOVA); (c) Kruskal-Wallis test.

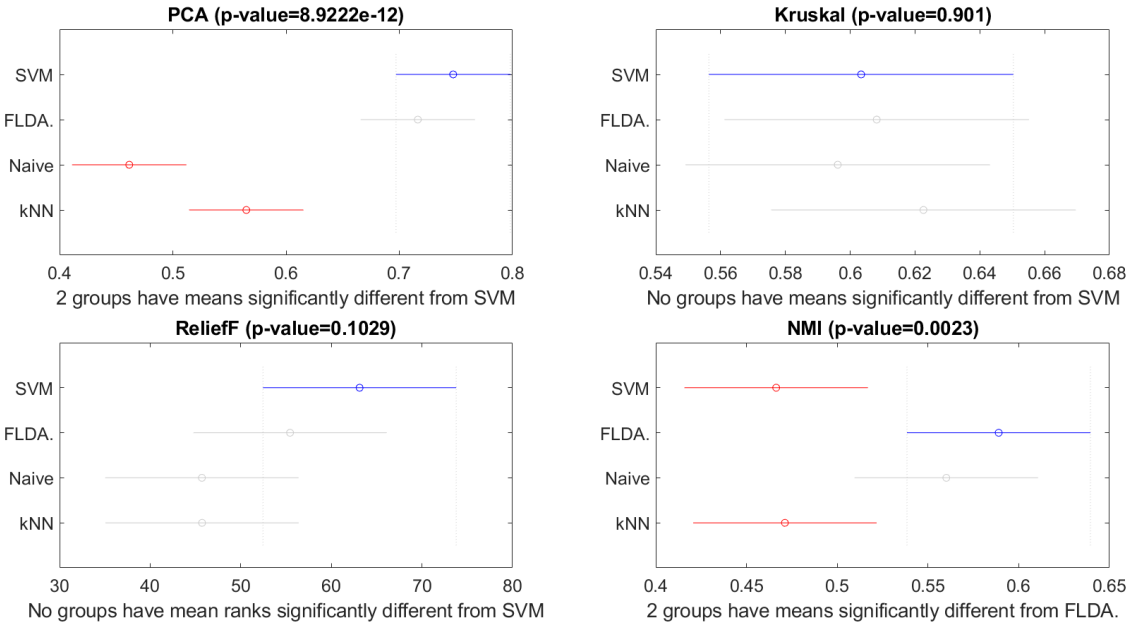


Figure C.6: Multiple comparison test and respective p-value of the statistical differences between the accuracies values of each classifier for the different methods of feature selection/reduction, as classification models of the three code and resting tasks.

C.2.2 Participant's Proficiency Analysis

Table C.16: Performance of FLDA classifier after PCA feature reduction, for each different tasks and the three proficiency levels (Intermediate, Advanced and Expert). The overall performance of accuracy obtained for each proficiency was 72.92 ± 13.41, 71.88 ± 9.88 and 67.19 ± 23.03, respectively.

Classifier	Multiclass Classification	Evaluation Metric (%)				
		Recall	Precision	Specificity	F-measure	Accuracy
FLDA	C1	Intermediate	87.50 ± 16.85	100.00 ± 0.00	100.00 ± 0.00	92.46 ± 10.64
		Advanced	95.00 ± 10.54	100.00 ± 0.00	100.00 ± 0.00	97.14 ± 6.02
		Expert	56.25 ± 51.54	75.00 ± 50.00	100.00 ± 0.00	60.00 ± 48.99
	C2	Intermediate	58.33 ± 44.38	43.64 ± 31.00	84.91 ± 14.43	48.50 ± 35.18
		Advanced	55.00 ± 40.48	48.33 ± 25.40	78.71 ± 12.28	46.05 ± 24.43
		Expert	56.25 ± 37.50	60.71 ± 48.62	83.33 ± 19.25	56.49 ± 40.43
	C3	Intermediate	50.00 ± 38.44	56.81 ± 40.39	86.30 ± 18.03	47.28 ± 32.00
		Advanced	37.50 ± 35.84	33.71 ± 33.50	84.85 ± 13.52	33.83 ± 30.44
		Expert	62.50 ± 47.87	53.33 ± 45.22	84.38 ± 23.66	57.22 ± 46.20
	Control	Intermediate	95.83 ± 14.43	83.93 ± 17.16	87.84 ± 16.05	87.54 ± 12.51
		Advanced	100.00 ± 0.00	96.00 ± 8.43	97.50 ± 5.27	97.78 ± 4.68
		Expert	93.75 ± 12.50	65.00 ± 17.32	73.20 ± 19.79	76.11 ± 15.00