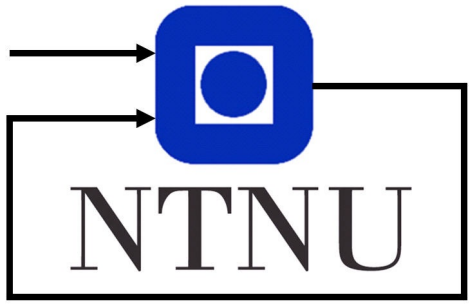
Design and evaluation of an automated stress detection system using Electroencephalography (EEG) signals



Author:
Ida Marie Andreassen

Supervisor:
Prof. Marta Molinas

Master's thesis
Department of Engineering Cybernetics
Norwegian University of Science and Technology

July 3, 2023

Abstract

In a world experiencing an unprecedented wave of declining mental health, it is more important than ever to ensure early detection and prevention of damaging stress. A rising field of study includes the combination of machine learning and Electroencephalography (EEG) signals to detect psychological stress. EEG is a non-invasive, relatively inexpensive diagnostic tool, making it the perfect candidate for widespread clinical use.

This study aims to find a possible solution to these problems, by *designing and evaluating methods of automated stress detection using EEG signals*. Collecting data was necessary to provide relevant data for classification. Natural stressors include taking an exam, therefore participants were selected from students at the Norwegian University of Science and Technology, during and after their exam periods. The period prior to their exams were used as the stressed state, while the baseline was recorded after they were on winter holiday. 28 students, 12 women and 16 men, participated in total, where the ages ranged from 20-28 years old, with a mean age of 23 ± 2 years.

Two methods have been used to label the data, State-Trait Anxiety Inventory for Adults - Y (STAI-Y) and Stress-Scale (SS). Three data sets have been constructed to use for classification. The first one consist of purely raw EEG data, the second has been through a band-pass filter and notch filter prior to filtering with Signal-Space Projection (SSP). The last data set was constructed in a similar matter with band-pass and notch filtering, then decomposing the signal into the Delta frequency band. Four different methods of feature extraction have been explored, time series feautes, entropy features, Hjorth features and Power Spectral Density (PSD) features. All data sets have been segmented into epochs of either 1 second, 2 seconds or 5 seconds. Lastly, the data have been classified using Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbor (KNN), and EEGNet. The highest accuracy was achieved by PSD features with raw data using 5s epochs and SS labels, reaching 77.97% accuracy, 64.22% sensitivity, and 86.21% specificity.

Sammendrag

I en verden som opplever en uforutsigbar bølge av nedadgående mental helse, er det viktigere enn noensinne å sikre tidlig påvisning og forebygging av skadelig stress. Et voksende studiefelt inkluderer kombinasjonen av maskinlæring og Elektroencefalografi-signaler for å oppdage psykologisk stress. EEG er et ikke-invasivt og relativt rimelig diagnostisk verktøy, noe som gjør det til en perfekt kandidat for utbredt klinisk bruk.

Denne studien har som mål å finne en mulig løsning på disse problemene ved å "utforme og evaluere metoder for automatisert stressdeteksjon ved hjelp av EEG-signaler". Innsamling av data var nødvendig for å gi relevant data for klassifisering. Naturlige stressfaktorer inkluderer å ta en eksamen, derfor ble deltakerne valgt blant studenter ved Norges teknisk-naturvitenskapelige universitet, under og etter deres eksamensperioder. Perioden før eksamen ble brukt som den stressede tilstanden, mens referansestilstanden ble målt etter at de hadde hatt juleferie. Totalt deltok 28 studenter, 12 kvinner og 16 menn, i alderen 20-28 år, med et gjennomsnitt på 23 ± 2 år.

To metoder har blitt brukt for å merke dataene, State-Trait Anxiety Inventory for Adults - Y (STAI-Y) og Stress-Scale (SS). Tre datasett har blitt konstruert for å brukes til klassifisering. Det første består av ren, rå EEG-data, det andre har gjennomgått et båndpassfilter og notch-filter før filtrering med Signal-Space Projection (SSP). Det siste datasettet ble konstruert på en lignende måte med båndpass- og notch-filtrering, deretter ble signalet dekomponert til deltafrekvensbåndet. Fire forskjellige metoder for "feature extraction" har blitt utforsket: tidsseriefunksjoner, entropifunksjoner, Hjorth-funksjoner og Power Spectral Density (PSD)-funksjoner. Alle datasettene er blitt segmentert i epoker på enten 1 sekund, 2 sekunder eller 5 sekunder. Til slutt er dataene blitt klassifisert ved hjelp av Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbor (KNN) og EEGNet. Høyest nøyaktighet ble oppnådd med PSD-funksjoner og rådata ved bruk av 5 sekunders epoker og SS-merker, med en treffsikkerhet ("accuracy") på 77,97%, en sensitivitet på 64,22% og en spesifisitet på 86,21%.

Acknowledgements

To begin with, I would like to address whoever finds themselves in a similar situation as me, whether that be a master´s student or a student just starting at university. The journey of writing this thesis has been the most infuriating, stressful, fantastic and joyful experience of my entire education. During this last year I have learned more about myself and my field of study than during my previous four years of higher education combined. While I by no means can call myself an expert on writing a thesis of this magnitude, I implore you to use this time as a learning opportunity and once-in-a-lifetime experience. Then, this time may become one of your best.

I want to thank my close friends and collaboration partners, Anne Joo and Ivar, for all the times and thoughts we have shared. Thank you for all your feedback, laughs, snacks and cups of tea, and for always taking the time to help when it was needed. I would also like to thank Mabel the fluffy corgi for all the cuddles and emotional support during my times of need.

I am deeply grateful for all the help and discussion provided my my supervisor, Prof. Marta Molinas. You have provided a project I am deeply invested in, and I would like to thank you for giving me the opportunity to explore such a fascinating and important field of study. You have helped me create a work that I am immensely proud of through your clear, constructive and thorough feedback. For that I am very thankful.

Lastly, I offer my deepest gratitude to my partner Thomas Schjem. Without whom I would never have been able to finish this thesis. Your constant support and encouraging spirit through the highest of highs and lowest of lows, have carried me across the finish line, I am forever thankful for that. Even though you don't fully understand what I have been babbling about for a year now, you have still listened, and provided feedback in any way you can.

Sincerely,
Ida Marie Andreassen

Table of Contents

List of Tables

List of Figures

Acronyms

EEG Electroencephalography. i, ii, ix, x, 1–8, 10, 14, 20, 22, 24, 27–32, 37, 39–41, 43–45, 65–68

EOG Electrooculography. x, 40–43

KNN K-Nearest Neighbor. i, ii, vii–ix, xi, xii, 2, 13, 17, 19, 20, 46–48, 50, 53, 56, 59, 62, 66, 67, 105, 106, 112, 113, 119, 120, 126, 127

PCG Phonocardiogram. ix, 1, 22, 24, 25, 29

PSD Power Spectral Density. i, ii, viii, x, 2, 3, 12, 13, 22, 27, 29, 31, 32, 37, 40, 46, 47, 50, 60–62, 64, 66, 67

RF Random Forest. i, ii, vii, viii, xi, xii, 2, 13, 16, 17, 46, 47, 50, 52, 55, 58, 61, 64–67, 103, 104, 110, 111, 117, 118, 124, 125

RMS Root Mean Square. 12, 66

SS Stress-Scale. i, ii, x, 2, 23, 33–35, 45, 47, 51–65, 67, 68, 100–129

SSP Signal-Space Projection. i, ii, x–xii, 2–4, 10, 11, 22, 35–37, 40–42, 47, 51–65, 67, 101, 103, 105, 108, 110, 112, 115, 117, 119, 122, 124, 126, 129

STAI-Y State-Trait Anxiety Inventory for Adults - Y. i, ii, x, 2, 9, 23, 24, 33–35, 45, 47, 51–63, 65, 67, 68, 100–129

SVD Singular Value Decomposition. 12

SVM Support Vector Machine. i, ii, vii–xii, 2, 13–16, 46, 47, 50, 51, 54, 57, 60, 66, 67, 100–102, 107–109, 114–116, 121–123

1

Introduction

1.1 Background and motivation

The world is experiencing an unprecedented wave of mental health problems, arising across generations. Accelerated by the global pandemic, and the numerous humanitarian crises across the globe, i.e. war or poverty, mental health is now a hot topic both in the media and research communities. The reasons for stress are numerous and subjective to each individual experiencing it, so now the question is; how can we prevent stress and detect it early enough to reduce the chance of stress evolving into more serious health problems?

Stress is a natural response that is necessary to live a life. Without stress, no one would not be able to do anything. As a response, stress is responsible for making us take actions, stress is the reason we are able to cook when we are hungry, clean when it is needed, and go to work on a regular basis. All actions a human takes is a results of stress (Selye, 1973). However, in recent years the focus has been shifted towards the fact that increased stress, that is abnormal to the daily stress needed for survival, is leading towards increased health problems. Chronic stress has been linked to increased risk of depression, anxiety, cardiovascular diseases, and even neurological disorders (McEwen, 2008). Early detection of increased stress may reduce the risk of it leading to serious health problems, so long as the interventions resulting from detecting the stress are appropriate and feasible for the situation.

1.2 Problem description

The aim of this study is to design and evaluate methods of automated stress detection using Electroencephalography (EEG) and Phonocardiogram (PCG) signals. EEG is a non-invasive, medical tool used to examine the electrical activity in the brain, specifically looking for changes or abnormal patterns, making it a good choice for this study. The thesis

is a part of a larger research group working on the same data, consisting of Anne Joo Yun Marthinsen and Ivar Tesdal Galtung working together, as well as Christian Sletten and Øystein Stavnes Sletta working individually. The first step for this thesis is the collection of data. The participants were recruited from students at the Norwegian University of Science and Technology during their exam period, assuming the exams are a natural stressor that we can use for stress detection. Baseline data was recorded post exams, after the students had been on winter break. Due to the subjective nature of stress, two methods was used for ground truth labeling of the data. State-Trait Anxiety Inventory for Adults - Y (STAI-Y) is a questionnaire developed to reveal anxiety states and traits, meaning short term anxiety and long-term anxiety. The second method denoted Stress-Scale (SS) is a scale from 1-10, where 1 means not stressed. The results of these methods will be compared and analyzed along with the data. Raw EEG data requires pre-processing and filtering, and two different methods are proposed in this thesis. Common for both methods is initial filtering using a bandpass filter and a notch filter, to remove information in the signal that is not related to neurological activity. Filtering using Signal-Space Projection (SSP) is proposed as a way of removing blinking artifacts from the signals, while decomposing the signal into specific frequency bands is proposed to evaluate if the different frequency bands contain information regarding stress. Segmenting the data into epochs is a way of reducing the complexity and increasing the amount of information gathered when extracting features. Three different epoch lengths will be evaluated, namely 1 second, 2 second and 5 second epochs. Feature extraction is an important part when dealing with complex data, thus four feature extraction methods are evaluated here. This includes Time series, Entropy, Hjorth and Power Spectral Density (PSD) features. To classify the data three machine learning classifiers and one neural network will be tested, compared and evaluated. The classifiers are Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbor (KNN) and EEGNet. They will be evaluated based on their accuracy, sensitivity and specificity.

1.3 Related work

Research regarding stress detection using EEG signals have increased in the past years. Katmah et al. (2021) have done an extensive review of 51 articles on EEG and stress, concluding that the lack of consistency in data analyzing methods and experiments protocols, lead to varying and contradictory results for classification of stress. They propose combining different approaches related to studying brain connectivity with network connectivity, and empolying deep learning methods for classification. Khosrowabadi et al. (2011) used 8 EEG channels to classify 26 students before and after exams, in a resting and non-resting state, using Percieved Stress Scale to label the data. Classification was performed with both KNN and SVM, using fractal dimension, Gaussian mixtures of the EEG spectrogram and Magnitude Square Coherence Estimation as features. The results using the latter feature reached over 90% accuracy with inter-object validation. The combination of features is an area of exploration in the research community. Hou et al. (2015) found that combining statistical features (time series features) with fractal features, the accuracy using SVM improved. Using simultaneous EEG with Functional Near Infrared Spectroscopy, Al-shargie et al. (2015) found that there is a correlation between the alpha

frequency band and the change in concentration of oxygenated hemoglobin due to stressful tasks, and that stress decreased the PSD of the signal in the alpha frequency band. The importance of continued research into using EEG as a diagnostic tool in combination with artificial intelligence is proven by Tveit et al. (2023), where they achieved a performance similar to human experts in classifying epilepsy with EEG signals.

1.4 Structure of the thesis

The thesis is divided into six chapters, each serving a purpose to structure the text into meaningful sections. Chapter 2 presents the theory necessary to understand the proposed methods and the discussion evaluating the performance of the methods. The chapter starts with an overview of Electroencephalography (EEG), including placement of the electrodes, explaining artifacts in the signal, how frequency bands are related to EEG, and lastly defining stress and how it presents in EEG signals. The theory continues by introducing how filtering affects signals, while describing the filtering method Signal-Space Projection (SSP). The next section introduces the need for feature extraction when performing classification, and present the theory behind the features used in the thesis. The last section of the theory will explain and describe how the machine learning techniques work.

In Chapter 3 all the materials and methods used in this study are presented. That includes an in depth description of the design of the experiment, an exploration of both the data set and the labels, continuing to a walk-through of the filtering steps and how it affects the data. Then the methods used for feature extraction and classification are presented, with a detailed overview of their functions and hyperparameters. The chapter concludes in a section detailing the performance metrics.

Chapter 4 presents the results following the classification of three data sets. The chapter is structured by feature type, where each of the classifiers are presented with their own tables with results. A brief description of the most notable results will precede each table.

The discussion of the results follows in Chapter 5. The results will be discussed on the basis of the theories and methods mentioned in Chapters 2 and 3. Particularly fascinating results like stand-out accuracies, sensitivities and specificities will be highlighted.

The conclusion to this thesis will be presented in Chapter 6 along with a note on the possibilities of future work.

2

Theory

This chapter will present the relevant theories and background necessary for understanding and discussing this project. The five sections of the chapter cover Electroencephalography (EEG), psychological stress, Signal-Space Projection (SSP), feature extraction, and machine learning.

2.1 Electroencephalography (EEG)

Electroencephalography (EEG) is a method used to measure the brain's electrical activity. It is mostly used for examining the function of the brain and diagnosing seizure disorders such as epilepsy, discovering brain tumors, and for comatose patients to monitor their brains and as a tool to determine whether or not there is brain activity to ascertain death. In recent years, EEG has been used for research as it is a simple, non-invasive method that provides real-time information about the state of the brain. Thus, research into how the brain responds to sleep, depression, eating disorders, and other psychological disorders is ongoing.

The electrical activity in the brain originates from the cells communicating. They communicate through action potentials, which are electrical impulses stemming from the potential differences between the outside and inside of a cell's membrane (Kaada, 2018). Feelings, hormones, movement, or sensory sensations can trigger an action potential. Alone, one action potential is too small to be measured; however, many cells usually work together, and their combined electrical impulses are large enough to be able to measure. Still, the electrical voltage is measured in microvolt (μV) and must be amplified before it can appear on screen in an EEG (Engstrøm and Jansen, 2022).

2.1.1 EEG electrodes

The electrical impulses can be measured by placing electrodes across the scalp. Electrodes are small metallic conductors that can be used dry or with conductive gels. By placing a reference electrode on the ear lobe, each electrode on the scalp measures the voltage difference between the reference and itself, meaning the electrical activity at that point, and transmits this to the recording channels of the EEG. The electrical activity is shown on a graph in a wavelike pattern with peaks and throughs. An example of an EEG is pictured in Figure 2.1.

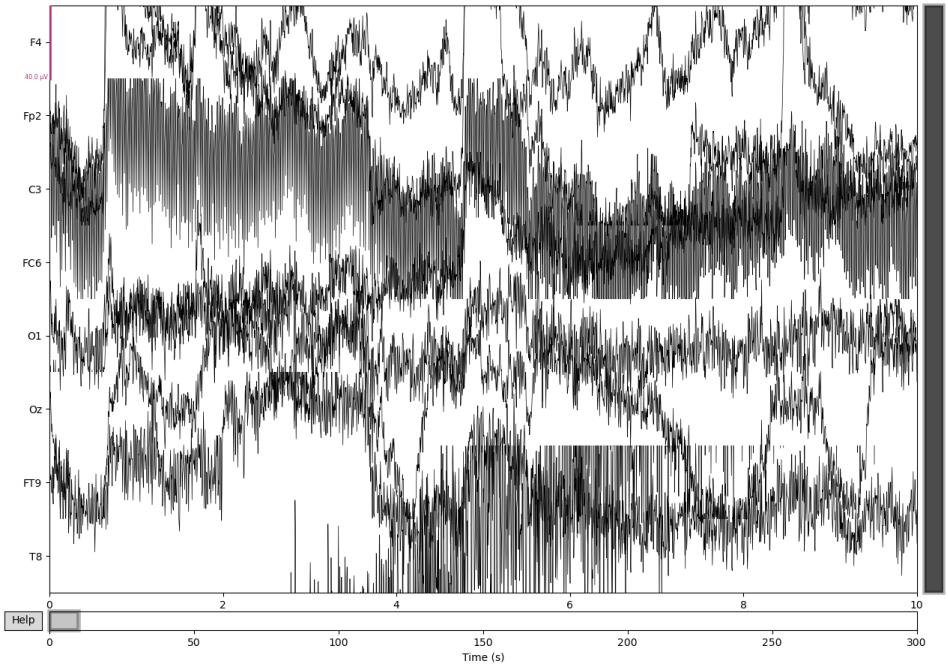


Figure 2.1: An image of a raw, noisy, EEG with 8 channels.

The placement of the electrodes plays a significant role; therefore, an international standard has been developed. The modified 10-20 system developed by Sharbrough et al. (1991) is widely used. The 10-20 system derived its name from the fact that each adjacent electrode is placed within either 10% or 20% of the total distance from the front to the back or the left to the right side of the skull. The placement of 32 electrodes based on the 10-20 method is shown in Figure 2.2. A unique name represents each electrode, with one or two letters in the name accompanied by a number. The numbers represent the sagittal lines, i.e., all placements with the numerical value 3 lie on the same sagittal line, and the value z (zero) lies on the midline sagittal plane of the skull. Even numbers refer to electrodes to the right of the midline sagittal plane, and odd numbers refer to the left. Whereas the lettering denotes which area of the brain the electrode is reading from; pre-frontal (Fp), frontal (F), frontocentral (FC), frontotemporal (FT), central (C), temporal (T), centropari-

etal (CP), parietal (P), posterior temporo-occiptial (PO), and occipital (O). The placement of electrodes is often pairwise and symmetrical. Each electrode corresponds to a channel in the EEG.

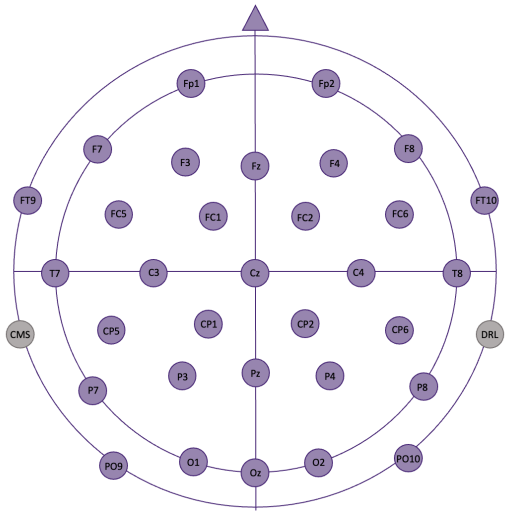


Figure 2.2: A 32 electrode placement map using the 10-20 method. Inspired by Ghosh et al. (2022)

2.1.2 EEG artifacts

Artifacts in EEG are defined as any noise in the data that can be attributed to specific, often physiological sources. The most common forms of artifacts are eye blinks, eye movements and muscle contractions. This type of noise can often overshadow the actual brain activity that we want to measure, thus it is important to filter these artifacts. Artifacts such as blinking often show up in the signal as high amplitude peaks (Jiang et al., 2019). Examples of this can seen in 2.3, where the high amplitude peaks are marked by the red boxes. Other artifacts like movement of the electrodes will often show up in the EEG as drifting, like a slow decline of the signal of a channel. While muscle tension will present itself as an increase in high frequency activity over a short amount of time.

2.1.3 EEG frequency bands

Research implies that the brain operates on different frequency levels depending on the amount of activity. When examining an EEG, it is essential to note the frequency and the amplitude of the brain waves and whether or not there are abnormal patterns—then figure out if the phenomenons happen only in one part of the brain or across the whole

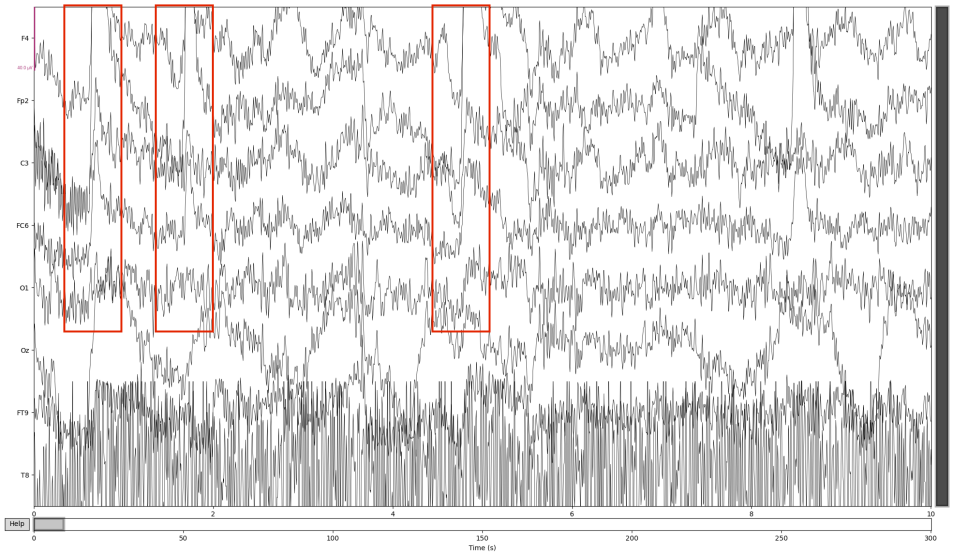


Figure 2.3: Blinking artifact in an EEG. Artifacts are marked with red boxes.

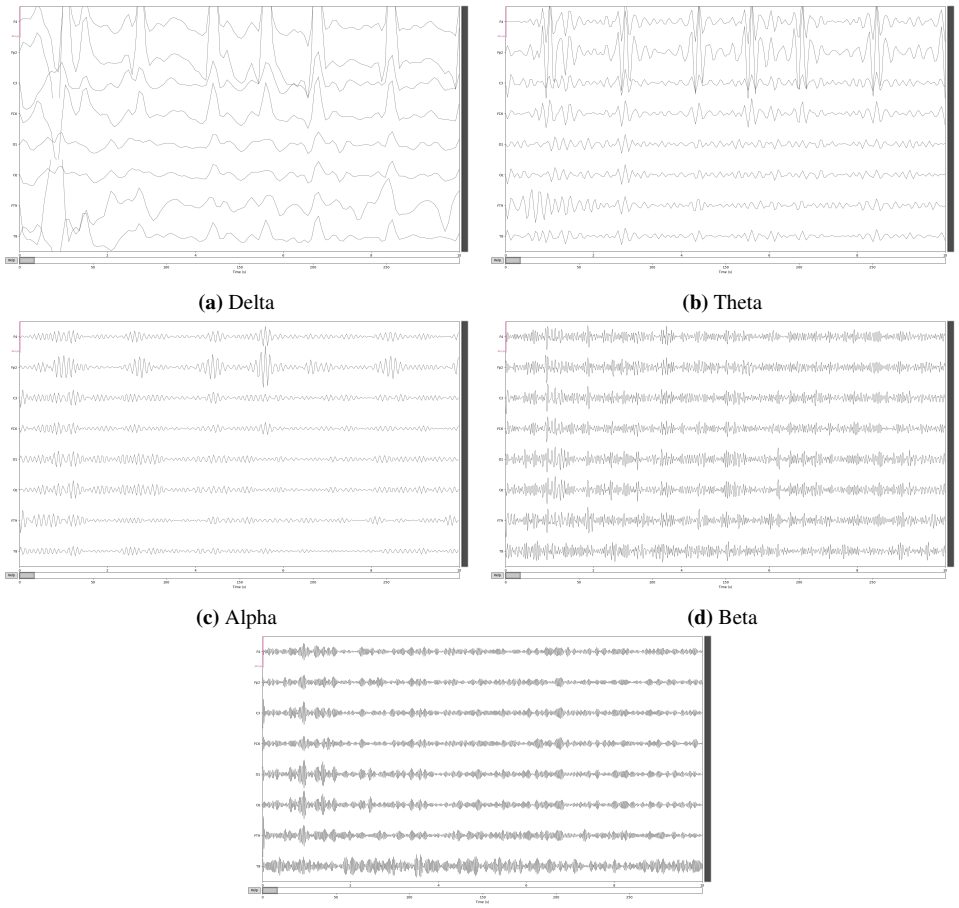
brain (Engstrøm and Jansen, 2022). The frequency levels that the brain operates on can be divided into what are called frequency bands. Usually, the frequency bands are divided into five categories denoted delta, theta, alpha, beta, and gamma, defined in Table 2.1. Figure 2.4 illustrates how the EEG signals appear in the different frequency bands.

Table 2.1: The table details the different frequency bands and their frequency ranges.

Frequency band	Frequency range
Delta δ	0-4 Hz
Theta θ	4-8Hz
Alpha α	8-12 Hz
Beta β	12-30 Hz
Gamma γ	30-40 Hz

2.1.4 Physiology of stress

Stress has perhaps become one of the most difficult phenomena to properly describe and define. In society today, stress is often commented in both media and healthcare, with magazines shouting about 'good' stress and 'bad' stress, how to avoid it, and how people today are more stressed than ever. The first formal definition stems from Hans Selye in 1936, when he observed that although people suffered from different diseases, many of their symptoms were similar. They lost their appetite, muscular strength and their ambition to accomplish anything (Selye, 1973). By injecting rats with hormone extracts and



(a) Delta

(b) Theta

(c) Alpha

(d) Beta

(e) Gamma

Figure 2.4: Illustration of the how the frequency bands appear in EEG data.

toxic drugs, he noticed that the adrenal cortex of the rats became enlarged, the thymus and lymph nodes decreased in size, while ulcers appeared in their stomachs, regardless of what was being injected. This was the basis for the definition of stress and the general adaptation syndrome. Selye defined stress as ”the nonspecific response of the body to any demand made upon it” (Selye, 1973, p. 692). Meaning that the factors that produce stress, denoted *stressors*, can be anything from positive to negative, yet they still produce the same biologic stress response in the body. Stress can be felt from receiving both devastating news like a family member has become ill, and positive new such as a family member having recovered from a disease. They both illicit the same stress-reactions in our body, where the only difference will be in the intensity of the demand. Each scenario yields a demand to the body which then needs to regulate itself back to what is called *homeostasis*, meaning the physiological processes that maintains a steady state of the body. As such, stress is a response that is unavoidable, it is always present even at rest, when the heart keeps beating, the chest rises and falls with each breath, the stomach digests the food consumed, and so on. Stress is necessary to provide enough energy to adapt to the demands stressors place on the body.

The general adaptation syndrome presents the three stages of a stress response. The first stage is the *alarm reaction*, which is a short period where the body becomes activated to respond to the stressor, most commonly it is described as the fight or flight stage. The second stage is named the *stage of resistance*, where the body is in a lower state of activation than the alarm stage, but still activated enough to fight the stressor for some period of time. Lastly, the final stage is *exhaustion*, that happens when the body has spent all its resources, that may happen if the stressor is severe enough and has been applied for a good amount of time. If exhaustion incurs then fatigue, disease or even total collapse may happen (Selye, 1973). This is the point where stress becomes negative. The stress response, although general and fitting for many people, is still relatively subjective given that the same stressor with the same intensity and period, may produce entirely different reactions in different people (Svartdal and Malt, 2022). Which is also why stress can become pathogenic under certain conditions, and not just a mechanism that ensures our survival.

Measuring stress is often not straight forward. As mentioned, stress is a physiological condition as well as psychological, however, the stress response is highly individualistic in terms of stressors, and it can be hard to determine if someone is experiencing stress-related symptoms towards a certain stressor. It is important to define what type of stress that is to be measured. For example, (Crosswell and Lockwood, 2020, p. 3) defines *acute stress*: ”short term, event-based based exposures to threatening, or challenging stimuli that evoke a psychological and/or physiological stress response, such as giving a public speech”. Often, acute stress is measured based on subjective reports such as answering questionnaires, inquiring on levels of anxiety, bodily symptoms, tension, state of mind and so on. Examples of questionnaires are State-Trait Anxiety Inventory for Adults - Y (STAI-Y) (Spielberger et al., 1971) and Perceived Stress Scale (Cohen et al., 1983). Physiologically, there is no one bio-marker that proves stress is present in the body. This makes it difficult to determine ways of physiologically diagnosing stress. There are methods such as measuring the amount of cortisol present in the body, but cortisol is not only released in

negative, health impacting stress-responses (Dickerson and Kemeny, 2004). Which makes it difficult to use cortisol as the only determining factor when diagnosing stress.

EEG has been proposed by several researchers as a tool to identify stress. Many studies and experiments have been conducted, though the results varies, and in some cases the results are even contradicting (Katmah et al., 2021). Currently, there is no standard protocol for either collecting EEG-data, pre-processing, the type of stressor to be researched, what part of the brain to study, feature extraction or what type of classifier to be used. This may lead to the contradictory results. In that way, there is not yet a definitive method for determining stress based on EEG.

2.2 Filtering

Filtering is an important part of signal processing as the sensors that are used for recording can often be prone to noise. The noise in a signal can be a result from environmental noise, frequency disturbances from the electrical grid surrounding us, or even other processes in the body that can be picked up from the sensors. For EEG specifically, noise tends to be a combination of environmental noise, frequency disturbances, heart beats, and muscle movements such as eye blinking or even just tension from the muscles as mentioned in section 2.1.2. The reason for filtering is to ensure that we retain as much of the important information from the signal as possible, without the disturbances. There are many methods available to filter EEG signals. Among them is Signal-Space Projection (SSP). The theory detailing SSP-filtering will be presented in this section.

2.2.1 Signal-Space Projection (SSP)

Projection at its most basic form converts some data points into other data points, usually to convert from a higher dimension to a lower dimension. The same can be said for Signal-Space Projection (SSP). Consider a projection from a three-dimensional space to a two-dimensional plane. In this case, you could think of projection as when you stand in the sun and look at your own shadow, you are the three-dimensional object and your shadow is the two-dimensional projection. This is the general idea behind projection in higher dimensions. In a way, EEG signals can be represented as vectors in space. The number of dimensions in the space is dependent on the amount of recording channels and the sampling time. There is one N-dimensional point for each sampling time. Noise can then be represented as one or more dimensions, for example in a three-dimensional space, the xyz-plane, if you know that the measurements in z-direction is mostly made up of noise (due to your measurement set-up), then you can project your signal to the xy-plane and then the noise component is removed from your signal.

Mathematically, projection is done using projection matrices. By multiplying the signal (the N-dimensional points) with a projection matrix you can project the signal onto the hyperplane (MNE). The important part is to choose the right projection matrix. It needs to be chosen so that the unwanted noise is filtered out while the information in the signal is not compromised, and the hyperplane spanned by the matrix needs to be orthogonal

to the noise-vector. SSP (Uusitalo and Ilmoniemi, 1997) is a method for estimating the projection matrix. It compares the measurements with and without the the signal that is under interest. The best practice is to record special cases of noise that you know will be present during the recordings. Either by recording empty room noises to filter background noise, or adding specific electrodes that measure eye movements or heart rate. In that case it is straight forward to use the noise recordings for calculating the projecting matrices. A more detailed mathematical description of the method can be studied in Uusitalo and Ilmoniemi (1997).

2.3 Feature extraction

As the amount and complexity of information contained in data has increased exponentially with the digital age, it has become necessary to introduce feature extraction when performing machine learning tasks. Feature extraction is meant to ensure that valuable information from data is extrapolated, to hopefully reduce dimensionality and complexity (Guyon and Elisseeff, 2006). Features can be thought of as distinctive markers of the data, for example when meeting a new person you might notice the color of their hair and eyes, what clothes they are wearing, if they have any tattoos or freckles etc., these are all features that can help define a person. Selecting features is often a quite extensive part of machine learning, and the success of the classification depends on the which features are selected. The most important part is to ensure that the features do not suppress any important information, as you may risk losing information if the restrictions in selecting the features is to strict (Guyon and Elisseeff, 2006). Meaning, the better practice is often ”the more, the merrier”. A detailed description of several relevant features will be presented in the following sections.

2.3.1 Time series features

Time-series features refers to features calculated on the time series of the signal. Examples of time-series features can be defined as

Peak-to-peak amplitude: Peak-to-peak amplitude is defined as the difference between the highest peak and lowest through of a signal.

Variance: The variance of a discrete random variable X_i is defined as

$$\text{var} = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2,$$

where N is the number of samples while μ is the expected value of the samples. The variance measure how far from the mean the samples are, it is a measure of dispersion of samples.

Root Mean Square (RMS): Defining the RMS of a set of values x_i as

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

where N is the number of samples, RMS can measure the signal amplitude and energy in the time domain. Giving information about the strength of a signal.

2.3.2 Entropy features

Entropy features captures the amount of information present for a signal and in regards to dynamical systems it measures the rate of information production. Some common entropy features include *approximate entropy*, *sample entorpy*, *spectral entropy* and *singular value decomposition entropy*, described below.

Approximate entropy: Approximate entropy measures the complexity and regularity of a signal, by measuring the likelihood that runs of sequences that are close for some observations will remain close for the next incremental comparison. Meaning it measures the similarities between epochs. The greater the likelihood of remaining close (being similar) means that the signal is more regular, and the lower the value of the approximate entropy will be (Pincus, 1995).

Sample entropy: Sample entropy is a modified version of Approximate entropy, that also measures the complexity and regularity of a signal. The modification from approximate entropy is that sample entropy does not use self-matches when comparing epochs, thus reducing the bias that approximate entropy may introduce to the results leading to suggestions that sequences are more similar than they actually are (Joshua S. Richman, 2000). Sample entropy is also independent on recording length.

Spectral entropy: Spectral entropy is defined as the Shannon entropy of the PSD of a signal. It is able to quantify the spectral complexity of an uncertain system (Zhang et al., 2008). The spectral entropy is mathematically defined as

$$\text{SE} = -\sum_{i=0}^{N} p_i \ln p_i$$

where p_i is the Power Spectral Density (PSD) function of the signal.

Singular Value Decomposition (SVD) entropy: SVD entropy is defined as the Shannon entropy of the SVD of a signal. Mathematically this can be defined as

$$\text{SVD\_entropy} = -\sum_{i=1}^{N} \bar{\sigma_i} \log \sigma_i$$

where σ_i represents the normalized eigenvalues (singular values) resulting from Singular Value Decomposition. A more in depth explanation can be studied in Roberts et al. (1999).

2.3.3 Hjorth features

Hjorth features were specifically developed for use in analyzing EEG (Hjorth, 1970). They are are based on the standard deviation of a signal, including the standard deviation of the amplitude of the signal. There are three types of Hjorth features, activity, mobility and complexity. The latter two are described in more detail below based on the work of Hjorth (1970).

Hjorth mobility parameter: Mobility is defined as the ratio

$$\text{Mobility} = \sqrt{\frac{\text{var}(\frac{df(t)}{dt})}{\text{var} f(t)}},$$

where f(t) is the EEG signal. The ratio will be dependent on the curve shape so that is measures the relative average slope, as both of the variances are dependent on the mean amplitude. The mobility can also be interpreted as the mean frequency of the signal.

Hjorth complexity parameter: Complexity is a dimensionless parameter that measures how similar the signal is to a sine wave. If the signal is similar to the sine wave it is denoted as unity, and any deviation from the sine shape will lead to an increase in unity. Complexity is defined as

$$\text{Complexity} = \frac{\text{Mobility}(\frac{df(t)}{dt})}{\text{Mobility}(f(t))},$$

where f(t) is the signal. Meaning that complexity is defined as the ratio between the mobility of the first derivative of the EEG signal and the mobility of the EEG signal itself.

2.3.4 Power spectral density features

Power Spectral Density (PSD): PSD features are based on computing the PSD for different frequency bands. A more in depth description of the different frequency bands is given in Section 2.1.3 along with an explanation of how the frequency bands are significant for EEG signals.

2.4 Machine learning and neural networks

This section will present an introduction to machine learning, with an in-depth explanation of the machine learning methods Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbor (KNN), including a neural network model called EEGNet.

2.4.1 Introduction

Machine learning is a method where the goal is for a machine to learn the patterns of existing data to predict the outcome when introducing new, previously unseen data. Machine learning is a subset of the broader term Artificial Intelligence (AI). The difference is

that machine learning learns from large amounts of data, improving with experience and making predictions based on the data. AI, among other things, acts on these predictions by responding to a problem or making recommendations based on data. AI is essentially trying to mimic human intelligence, seeing, understanding, or analyzing situations.

Machine learning is a powerful tool when used right. It can be used for decision-making processes by predicting future behavior, i.e., determining which days are more likely to be busier for stores. As a diagnostic tool, we can rely on the experience of machine learning and the enormous amount of data that a machine learning model can process to diagnose new cases of a condition, such as providing medical imaging to a model that is accurately able to determine if a tumor is benign or not. The applications are nearly unlimited as the world progresses into being more data-driven, and the amount of data available is increasing.

In general, we split machine learning further into several categories. Two are called *supervised* and *unsupervised* learning. Supervised learning means that you provide labels along with your data to a machine learning model (Jordan and Mitchell, 2015). A label uniquely corresponds to a data point, providing the answer to which class the data belongs. The model then uses the answers to learn underlying patterns in the data that, when trained, can be used on unseen, unlabeled data where it will predict which class the unseen data belongs to. Meaning that if you want to train a supervised model, you need labels for all your data. For example, if you're going to make a classifier to detect if an EEG recording suggests that someone is under the influence of alcohol, you need the training data to consist of both recordings with and without the influence of alcohol. You need to explicitly know and tell the model which recording was done with or without alcohol. Then predict using the trained model.

On the other hand, unsupervised learning does not need labels to train a machine learning model. Here, the purpose of the model is to detect underlying patterns and structures in the data without the assistance of labels attached. Usually, the model can find clusterings of data points and assigns that cluster a label (Jordan and Mitchell, 2015). It uses this cluster's characteristics to examine the new, unseen data of the test set and determine which group it most likely belongs to.

To validate a machine learning model we often split a data set into two categories called train data and test data. The goal of any machine learning models is to create a model that can accurately predict new, unseen data. When fitting a model you use the training data so that the model can learn the intrinsic patterns of the data. To check how the classifier performs, we keep the test set unseen until it is time to do a performance check. The test set will act as new, unseen data for our model, thus giving an indication of how the model performs on data it has never seen before.

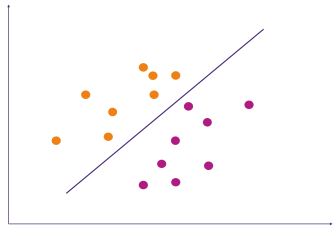
2.4.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a machine learning algorithm that uses the method of supervised learning. Every data point can be represented as a point on a plane, either a

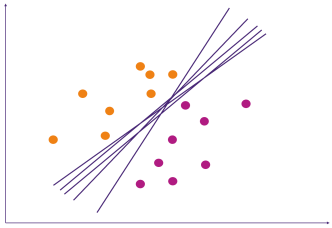
two-dimensional plane or any other dimension. For a two-class classification problem, each data point corresponds to either class. The idea behind SVM is that every point in a certain class will be near other data points of the same class and that there is a separation between data points of different classes. SVM seeks to create a hyperplane that separates the classes so that the model can look at the distance between the new data point to the hyperplane, and where the data lies relative to the plane, and then decide which class the new data belongs to (Noble, 2006). The purpose is to learn what differs between the classes.

To learn the class differences, SVM uses specific parameters. One, already mentioned, is the hyperplane. The hyperplane dictates where the SVM should differentiate between the classes. Figure 2.5a shows how a hyperplane separates two classes. The dimension of the data determines the hyperplane and its form. For example, two-dimensional data will be separated by a line, while three-dimensional data will be separated by a plane spanning the data. As Figure 2.5b shows, there are many possibilities for choosing a hyperplane. The way SVM chooses the hyperplane relies on distance measuring and selects the hyperplane that ”lies in the middle.” If we define the distance from the hyperplane to the nearest data point as the margin, SVM will choose the hyperplane that maximizes the margin between the data and the hyperplane, the maximum margin separating hyperplane (Noble, 2006). To do this we have to assume that the training and testing data are drawn from the same distribution.

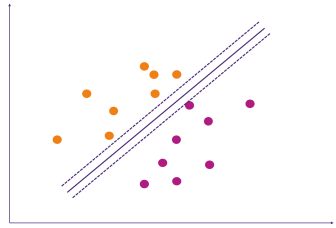
The kernel function of an SVM is used to add dimension to the data. In Figure 2.5d we can see that the structure of the data seem to fit more to a curved hyperplane rather than a linear. The kernel function solves this problem, it is a mathematical trick used to increase the dimension by projecting the data into a higher dimension (Noble, 2006).



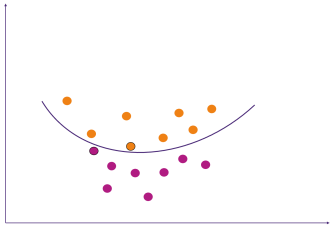
(a) Two-dimensional plot of example data points with a hyperplane separating them.



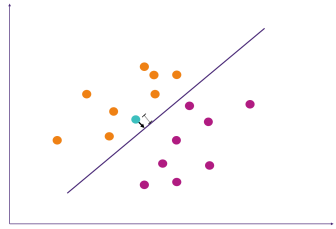
(b) Plot showing that SVM have many options for a hyperplane, and it has to choose the one hyperplane that best maximize the distance from the hyperplane to the data points.



(c) The maximum-margin hyperplane



(d) Choosing a different kernel can lead to a different hyperplane.



(e) When introducing new data, the distance between the point and the hyperplane plays a part in which class SVM predicts the point to be in

Figure 2.5: Showing how SVMs work. Image inspired by (Noble, 2006).

2.4.3 Random Forest (RF)

Random Forest (RF) is a type of decision tree algorithm. Decision tree algorithms are algorithms that decide which class data belongs to by whether or not they satisfy some conditions. Decision trees are made up of nodes called either decision nodes or leaf nodes. Decision nodes often have a logical statement or a test that can either be *true* or *false*, if the statement is true you move down to the left of the tree and to the right if the statement is false. Leaf nodes are the endpoint of the tree, where no further decisions or statements needs to be fulfilled, and they represent the class labels. The leaf nodes are what provides the prediction. For example if you have a basket of fruit and you would like to know what type of fruit is in the basket, you could build a simple decision tree. In Table 2.2 the

chosen features for the fruit basket are diameter and color. Each of the features combined yields a fruit, for example the combination of a size of 6 cm with the color red, will give an apple. In general, a decision tree will contain some decision nodes and some leaf nodes as presented in Figure 2.6a. Specifically the feature set from Table 2.2 can be represented as shown in Figure 2.6b, where the first decision node asks if the diameter is larger than 5 cm. If the answer is no, then the fruit will have to be a grape, since it is the only fruit with a diameter less than 5 cm, this represent a leaf node where we have reached a class label. If the diameter of the fruit is larger than 5 cm the answer to the first decision node is yes and we move down to the left. Since there are two kinds of fruit that have a diameter larger than 5 cm, we need a second decision node to decide which fruit we are dealing with. Here, we make use of the second feature that is color and the decision node asks if the color is orange, if the answer is yes we move to the left and find that the answer is that the fruit is an orange. If the answer is no, then we are moving to the right and are dealing with an apple. Both of the last nodes are also leaf nodes. Now all the fruits have been categorized.

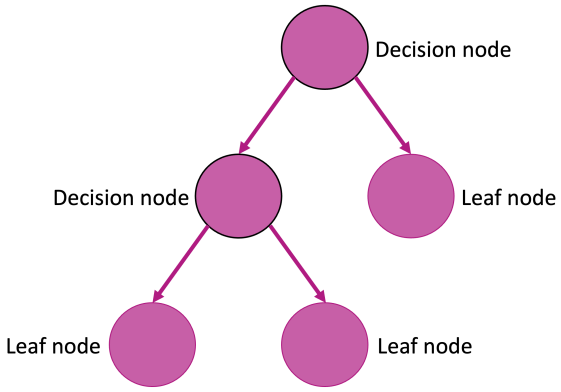
Table 2.2: Fruits represented with features based on diameter and color.

Feature 1 - Diameter	Feature 2 - Color	Label - Fruit
6 cm	Red	Apple
2 cm	Purple	Grape
10 cm	Orange	Orange
8 cm	Orange	Orange

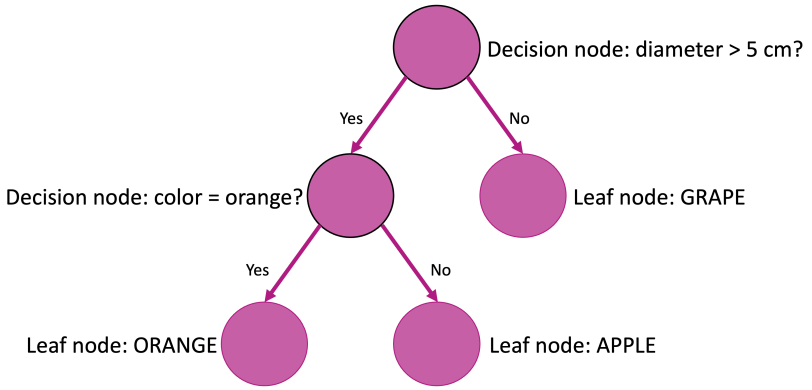
The downside to decision trees are that they are prone to change if the data set is increased or otherwise changed, which will result in high variance. RF was then introduced to minimize the variance and still produce an accurate result. As the name suggest, RF is made up of multiple, random decision trees, thereby making a forest of trees. Each tree is made with a method called bagging, developed by (Breiman, 1996), that generates multiple versions of a predictor (in this case a decision tree). Bagging is a combination of two methods, namely bootstrapping and aggregation. Bootstrapping means that we build each tree by making a random selection from the training set *with replacement*. After building N number of trees, each tree will be fitted to the test set and produce a prediction based on their specific structure and logical statements. By using N trees we get N predictions. Aggregation means that the most 'popular' prediction from all the trees will be the result of the classification (Breiman, 2001).

2.4.4 K-Nearest Neighbor (KNN)

K-Nearest Neighbor (KNN) is a non-parametric, supervised learning algorithm that was first introduced in 1951 by Fix and Hodges Jr (1952). Non-parametric means that no assumption is made regarding the structure of the data. It is based on the concept that the



(a) A simple graphic showing decision nodes and leaf nodes in a Decision Tree



(b) The fruit basket represented with a decision tree

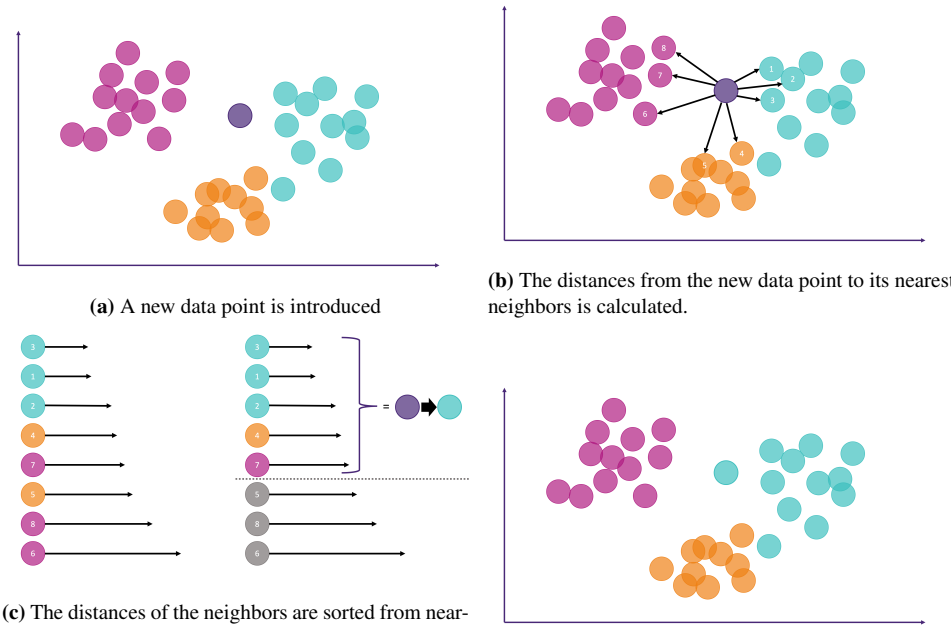
Figure 2.6: Decision trees

classes in the closest proximity to a new, unlabeled data point, deliver useful information about the label (Kramer and Kramer, 2013). The principle can be illustrated as in Figure 2.7. In Figure 2.7a a new point (in purple) is introduced to a data set consisting of three classes; pink, blue and orange. KNN then uses the nearest neighbors by measuring the distance from the new point to certain neighbors as in Figure 2.7b. In Figure 2.7c all the distances have been ordered from nearest to farthest. $K = 5$ neighbors are then chosen, and from those five the class with the most neighbors and the shortest distance, will be the class appointed to the new data point. In this case the blue class has the three close neighbors, compared to only one for each of the other classes, and thus the new data point is classified as blue, Figure 2.7d. Usually, either Minkowski or Euclidean distance is used for the distance calculation, they are respectively, defined as

$$\| \hat{x} - x_j \|^p = \left(\sum_{i=1}^{q} | (\hat{x}) - (x_i)_j |^p \right)^{1/p}$$

$$\| \hat{x} - x_j \| = \sqrt{\sum_{i=1}^{q} (\hat{x}) - (x_i)_j^2} \,,$$

where $\hat{x}$ is the new data point, x_j are the existing data points, and q denotes the dimension we are working in. The *K* in KNN denotes how many neighbors should be apart of the evaluation. Choosing an odd number for K ensures that there will always be majority when the vote comes to stand. Weights are also used to determine how much power each neighbor should have. Either all the neighbors are counted equally as with *uniform* weighting, or the closest neighbors will have a greater influence to those further away, this weighting is called *distance*.



(a) A new data point is introduced

(b) The distances from the new data point to its nearest neighbors is calculated.

(c) The distances of the neighbors are sorted from nearest to farthest. Five neighbors are chosen to decide which class the new point belongs to, and the class with the most near neighbors is the deciding factor.

(d) The new data point has been classified and is now colored blue.

Figure 2.7: An overview of the KNN classification method.

2.4.5 EEGNet

EEGNet was developed by Lawhern et al. (2018) where the objective was to make a compact, convolutional neural network to accurately classify EEG signals from different Brain-Computer Interface paradigms. Meaning that the goal is for EEGNet to generalize across different methods that EEG is being used for, such as controlling prosthetic limbs, controlling a drone, image reconstruction, visual-evoked potentials, and other paradigms. As Lawhern et al. (2018) states, feature extraction and classification methods are often specifically tailored to the characteristics of the EEG signal. The introduction of EEGNet seeks then to generalize this process, by developing a neural network model that adapts to the data at hand. The architecture of the model can be seen in Figure 2.8. The model first uses a 2 dimensional convolutional neural network where the filter length is set to half the sampling rate of the data, outputting feature maps of the EEG at different band-pass frequencies. The next step uses depthwise convolutions to reduce the number of parameters, and decoupling the relationship between the feature maps. Average pooling is performed for dimension reduction. Then the features are used in the classification block where classification is performed using the softmax function (Lawhern et al., 2018).

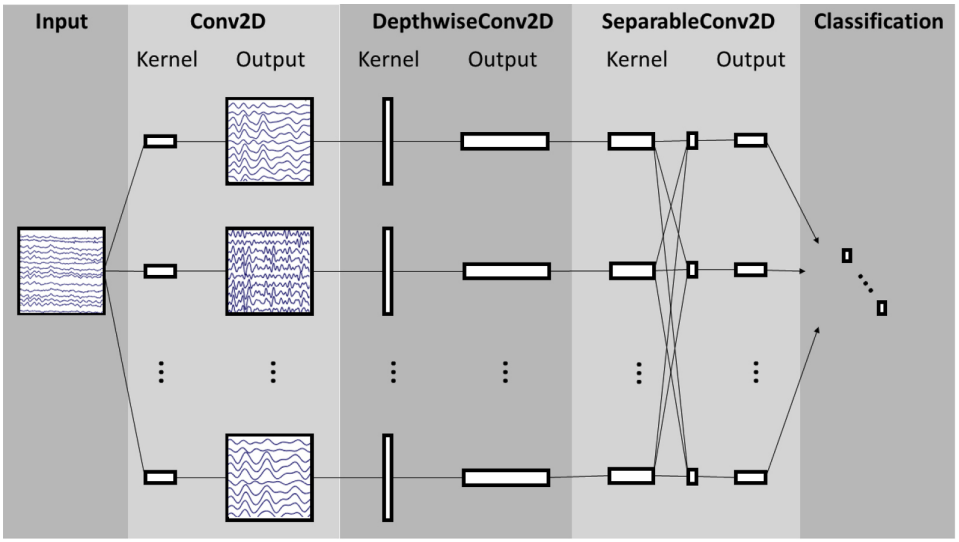


Figure 2.8: The architecture of the EEGNet model, image from Lawhern et al. (2018).

3

Materials and methods

This chapter presents the materials and methods used in the thesis. The chapter starts with design of the experiment, first describing the purpose of the experiment and how we chose the participants. Then an explanation of how the data was labeled, and a statistical analysis of the labels, comparing the two labeling methods. A quick recap of the equipment that was used for the experiment follows, along with a walk-through of the protocol. The next section focuses on exploring the data, both raw data and its Power Spectral Density. In Figure 3.1 an overview of the method used in this thesis is presented. The thesis focuses on three versions of the data, the raw data, data filtered with Signal-Space Projection, and decomposed data. The latter two data sets have also been through an initial filtering process that will be explained in more detail in Section 3.3.1. All three versions of the data is then segmented into epochs of varying length, split into training and test sets, before feature extraction with several types of features. Finally, classification is performed with four different types of classifiers. To summarize; the method consists of three data sets and two sets of labels, that will be classified with different combinations of features and classifiers. Some key parts of the code have been included in the Appendix and will be referenced when necessary throughout the text, the rest of the code and all its helper-functions can be found in this GitHub repository: `https://github.com/idamand/stress_detection_EEG`

3.1 Design of experiment

The data was compiled between December 2022 and January 2023. This project explores whether there is a way to detect psychological stress using EEG and Phonocardiogram (PCG) signals accurately. To detect psychological stress using machine learning, we need to record both stressed and not stressed people (the control group). For this project, it was chosen to use the same participants for the stressed and not stressed parts by recording the participants at two separate times. We assumed it would be easier to compare the same participant's stressed and not stressed states. Considering that in a clinical setting, stress might be ruled unhealthy if the current stress levels are higher than they previously have

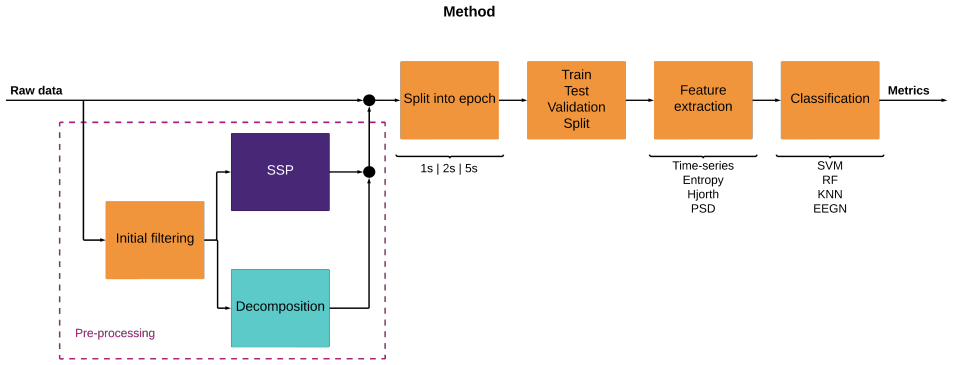


Figure 3.1: A flowchart presenting the steps in the methods of the thesis. From raw data through pre-processing and filtering, then splitting the data into epochs before splitting into training and test sets. Following is feature extraction and classifications methods.

been for a patient. As described in Section 2.1.4, real-life stressors that induce acute stress include taking exams, which gives a perfect opportunity to recruit students with an exam close to the data collection date. The second recording session was chosen to be in January, after the students had been on winter break, assuming that their stress levels had decreased from the period before their exam. There were 28 participants in total, 12 women and 16 men. Their ages ranged from 20 to 28, with a mean age of 23 ± 2 years. Though they were first recorded during their exam period, not all showed signs of stress at their first recording. Therefore, two methods were used to label the participants as stressed or not stressed, the questionnaire State-Trait Anxiety Inventory for Adults - Y (STAI-Y), and the participant's stress rating, denoted Stress-Scale (SS) in this thesis.

Table 3.1 provides an overview of the notations used for the recording sessions that will be used throughout this thesis.

Table 3.1: Overview of the notations used for the recordings in the project.

Recording time/ recording type	December	January
No stimuli	Session 1, Run 1	Session 2, Run 1
Arithmetic test	Session 1, Run 2	Session 2, Run 2

3.1.1 Labeling the data

The subjective aspect of measuring stress is the main challenge of labeling whether stress is active in a participant. As described in Section 2.1.4 people's perceptions of stress may differ daily based on physiological factors, such as sleep quality, food intake, and hydration levels. In addition to psychological factors such as how their day has been, difficult tasks at school or work, or interpersonal problems. One day, which might be interpreted as a level 8 stressor, might be interpreted as a level 4 the next day. By choosing two different methods for labeling the data, there are grounds to compare each method and discover which gives the most accurate picture.

The first method for labeling the data was to use the STAI-Y questionnaire (Spielberger et al., 1971). It has two parts; the STAI-Y1 questionnaire focuses on the current state of stress within the participant, while the STAI-Y2 questionnaire focuses on long-term anxiety traits. This thesis focuses on measuring real-life, current stress, so the STAI-Y1 questionnaire was selected as a basis for one of the label sets. There are 20 questions spanning from "I feel calm" to "I feel upset", where each participant rates their answer either "Not at all," "Somewhat," "Moderately so," or "Very much so." Each question is scored between 1-4, depending on how much the answer is weighted. The more negatively denoted questions (like "I feel nervous") are scored from 1-4, where "not at all" gives a score of 1, while "very much so" yields a score of 4. For the positive questions ("I feel calm") the answer "not at all" yields a score of 4, while "very much so" is scored to 1. The final score is given by summing together the scores for each question, then each participant get a final score between 20 and 80, depending on their current stress level. A higher score corresponds to a greater chance of being stressed. The decision to only use the Y1 questionnaire was decided after the recordings were done, and so each participant filled out both Y1 and Y2 during each session. The participants answered the forms prior to entering the recording room and after the final recording was done for each session giving two scores per session, four in total. The questionnaire can be found in Appendix M.

The Stress-Scale has an interval between 1-10, where 1 corresponds to not stressed and 10 corresponds to very stressed. The participants were asked to rate their stress after each run, giving them two scores for each session, four in total. To convert the scores into labels, where the labels are either 0 for not stressed or 1 for stressed, they were compared to a threshold value chosen to be 4. So that each score below a 4 is denoted as 0, not stressed. While each score above 4, is denoted as 1, stressed.

3.1.2 Equipment

Mentalab's Explore device was used for recording EEG, (Mentalab). The device uses 8 channels meaning that 8 electrodes and 1 reference electrode are used for the experiment, the sampling frequency is 250 Hz. The Mentalab equipment includes two caps, one in size medium and one in size large, and open-source software to record data and check the impedance of the electrodes. EkoDuo digital stethoscope (Eko) was used for recording PCG, including the EkoDuo mobile application. The mobile application was used to check that the audio signals were sufficient while the software AudioCapture (AudioCapture)

was used to record the EkoDuo signals. The Python library PsychoPy (Peirce et al., 2019) was used to display the arithmetic test. Three computers in total were needed to conduct the experiment. One computer was connected via Bluetooth to the Mentalab Explore device, one computer was connected to the EkoDuo with an audiojack while AudioCapture was used for recording the PCG, and the last computer was using PsychoPy to display the arithmetic tests. To stream and synchronize the recordings, we used the open-source software called LabStreamingLayer (Kothe).

3.1.3 Protocol

The participants were selected after going through an initial screening. The initial screening was conducted as a questionnaire with questions regarding their physical and mental health. The participants could not be on any medication regulating their heart rate or cognitive functioning. In addition, they could not be diagnosed with any mental illnesses or illnesses concerning the heart's function.

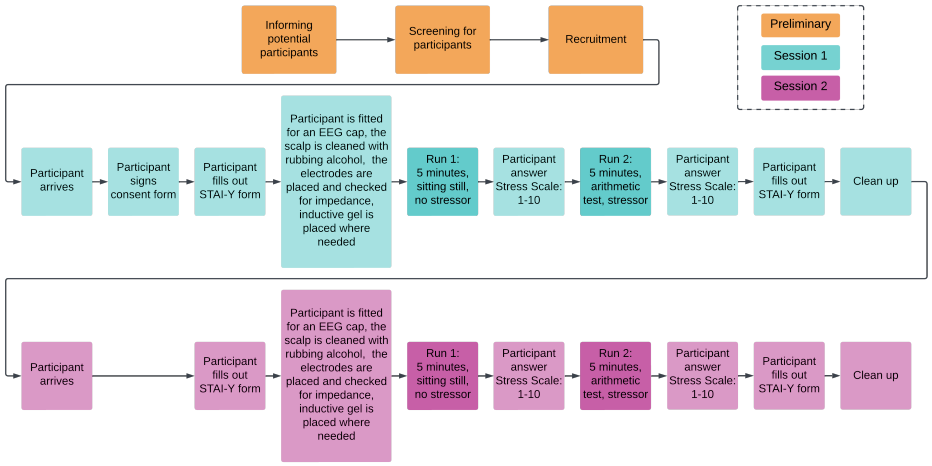


Figure 3.2: A flowchart showing the protocol from the preliminary stages prior to data collection shown in orange, to Session 1 in December shown in blue, and finally Session 2 conducted in January in pink

A graphical overview of the protocol can be studied in Figure 3.2. The data were collected similarly in the two sessions. Session one was at the beginning of December, close to the exams of the participants. Before entering the recording room, the participants were informed of the experiment and signed a consent form, available in Appendix L. Then they filled out an STAI-Y form, both Y1, indicating the current state of anxiety, and the Y2 form, indicating anxiety over time. Only STAI-Y1 was used for the labels as it most accurately captures the current state of anxiety the participants may feel.

Upon entering the recording room, the participant's head was measured, and the correct

Mentalab cap was chosen. Then the Mentalab EEG cap and device were placed upon the participant's head. The scalp, where each of the eight electrodes was placed, was washed with rubbing alcohol. To ensure decent quality recordings, the impedance of each electrode was monitored until it reached an acceptable level which was deemed below 160 kΩ. If the impedance was above this threshold, inductance gel was placed upon the electrode to decrease the impedance. The placement of the eight electrodes are illustrated in Figure 3.3, their positions was determined by genetic algorithm as part of the project thesis of Marthinsen (2022). Figure 3.4 shows the set up of the Mentalab software.

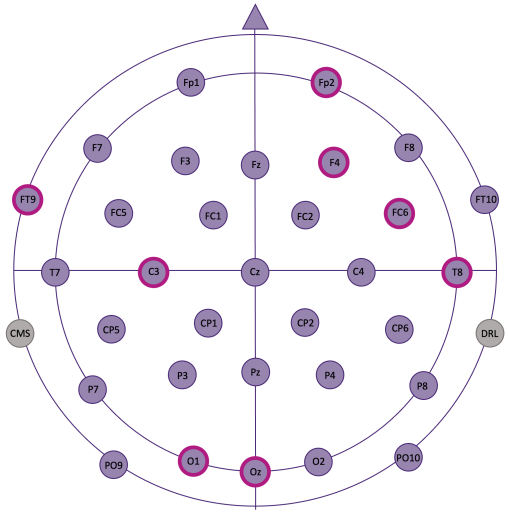


Figure 3.3: The eight electrodes used for the experiment are marked in pink. The channels are namely, Fp2, F4, FC6, T8, Oz, O1, C3 and FT9. Inspired by Ghosh et al. (2022)

After ensuring a good enough connection with the electrodes, the participant moved to the recording station to be fitted for the EkoDuo PCG sensor. The sensor was placed on Erb's point, above and to the right of the heart, as illustrated in Figure 3.5. The EkoDuo records audio through a microphone in the sensor. Thus, it is sensitive to surrounding noise, and it is crucial to obtain a good placement over the heart to get quality recordings. A strap was placed over the sensor to increase the quality, across the participant's chest, over their left shoulder, and under their right armpit. The strap was tightened so that the sensor got good audio signals from the heart, but not to the point of discomfort or pain for the participant. It was not easy to tighten the strap enough to obtain satisfactory recordings on female participants due to breast tissue, so the strap was often adjusted to lay more towards the center, in between the breasts. The recording quality was checked using the EkoDuo app, and when it proved satisfactory, the recordings began.

The recordings were done in two runs per session, each lasting five minutes. During the first run, the participant was seated on a chair without stimulation. The second run

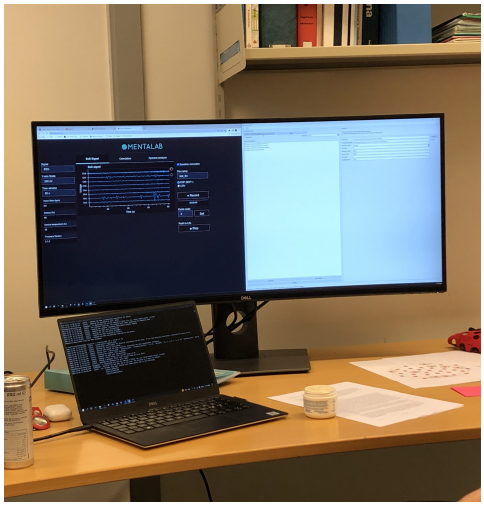


Figure 3.4: Image showing the set-up for EEG. The computer runs the Mentalab software to display and stream the EEG-signals to LabStreamingLayer.

included a cognitive arithmetic test to induce stress in the participant. The arithmetic test was constructed so that the participants sat in front of a screen, as seen in Figure 3.6, which displayed different arithmetic tasks on the form $4 + 1 = 7$ where the goal is for the participant to determine if the arithmetic statement is true or false. If they believe the statement to be true they press ”T” on the keyboard, if the statement is false the press ”F”.

3.2 Exploring the dataset

For this thesis, the data consists of recordings of brain waves from stressed and not stressed participants. Analyzing the data is essential to fully understand how to filter and prepare the data for machine learning and increase our understanding of the data. Firstly, this section will visually analyze the raw, unfiltered data. Looking for potential artifacts such as blinking or saccades and check for abnormal noise levels. To compare the two Sessions and if there are are any differences in noise levels, one recording from Session 1 and one from the same Participant in Session 2 will be examined. Namely, Participant 2, Session 1 - Run 1 and Session 2 - Run 1

Following, I will examine the PSD of two specific recordings. The analysis includes a comparison of the PSD of both a stressed and not stressed participant and the differences in PSD between EEG channels. Including an analysis of the topographic maps of the PSD. The two specific recordings are from Participant 2, like in the raw data analysis, though this time the recordings are from Session 1, both Run 1 and Run 2, as the labels have reported a difference in stress-levels between the two runs.

In general, the dataset consists of 108 recordings from 28 participants.

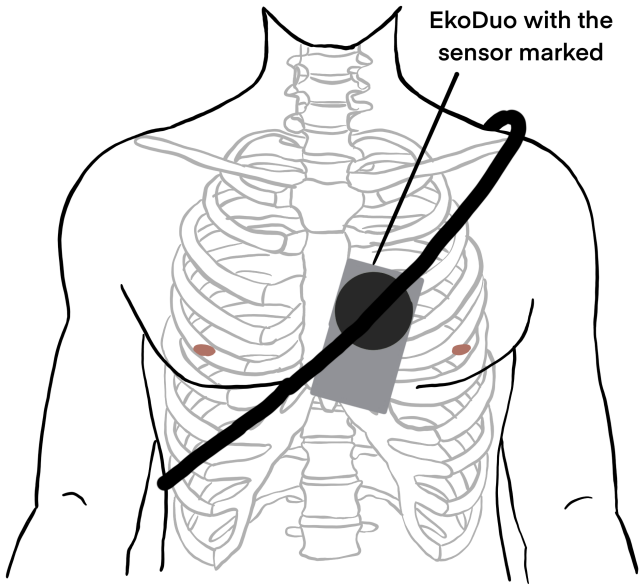


Figure 3.5: Illustrating the placement of the electrodes. Image from Andreassen (2023) based on AMBOSS

3.2.1 Raw data

Figure 3.7 shows 10 seconds of the raw EEG from Participant 2, Session 1, Run 1. With only the raw data available it is impossible to draw conclusions regarding the existence of artifacts in this signal. As the signal is presented here, it is only clear that it inhabits quite some amount of noise. Whether that is present due to head movement, poor connections with the electrodes, the power grid frequency or other sources is difficult to tell. Clearly, pre-processing the signal and filtering need to be applied before any amount of information is to be extracted from the signals.

Construction work was being done to the building where the recordings were taking place in session 2 in January. The environmental noise included drilling, hammering and other disturbing noises that arises from standard construction work. To see how the noise affected the recordings, if it was affected at all, the raw EEG from Participant 2, Session 2, Run 1 was plotted. In addition to comparing it to the Session 1 recording where no outside disturbances were present. Looking at Figure 3.8 it is evident that the recordings were affected by the environmental noise. In this case it is impossible to draw any conclusions

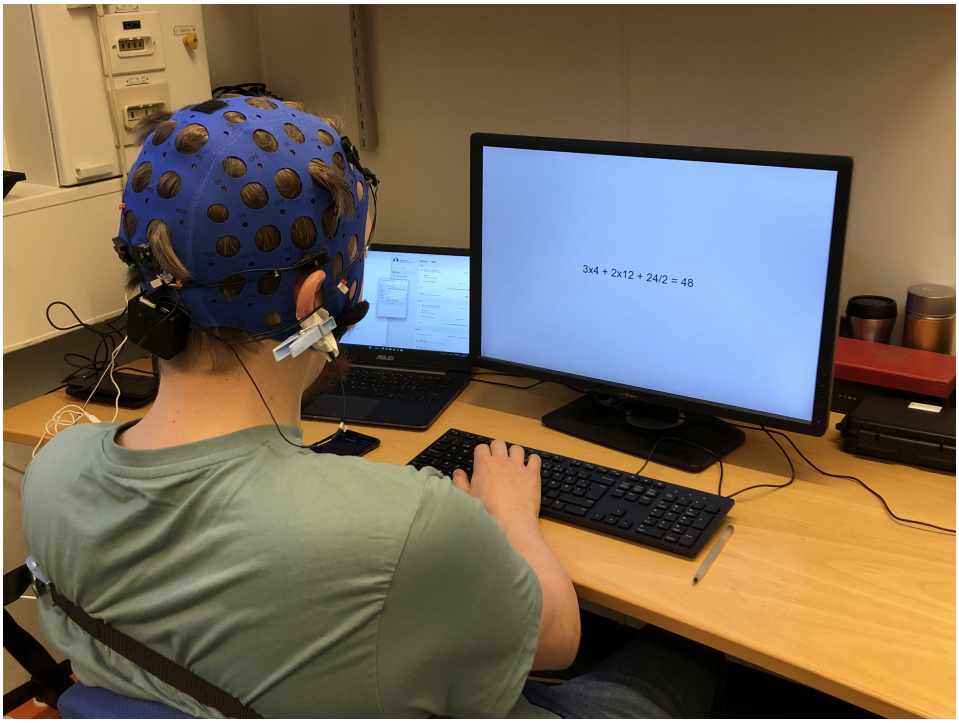


Figure 3.6: Image showing the set-up with a participant during Session 1, run 2. Explicit consent was given from the participant to include the image in this thesis. The image shows the participant with the EEG cap on the head, and the strap holding the PCG-sensor can be seen across the back of the participant. The computer directly in front of the participant displays one of the arithmetic tasks. While the computer to the left streams the PCG-signals.

regarding the state of the data. This noise was evident in most of the recordings from Session 2.

3.2.2 Power Spectral Density

The next step to extract more information from the raw EEG is to compute and plot the Power Spectral Density (PSD) of the signal. By comparing the PSD between a stressed and not stressed run it is possible to examine if there are differences in the power of the EEG-signal due to stress. Figure 3.9 shows the comparison of the PSD across all channels between Participant 2- Session 1- Run 1 (not stressed) and Participant 2 - Session 1- Run 1 (stressed). The first thing to notice is the two high amplitude peaks at 50 Hz and 100 Hz. These peaks are most likely not due to anything stress related, or even brain activity. The power grid frequency in Norway is 50 Hz which explains the peak at 50 Hz. The explanation behind the peak at 100 Hz is not evident, however, this falls outside of the range where neurophysiological information lies, thus it needs to be filtered. By looking at the PSD we have now gained information as to where some of the noise present in the

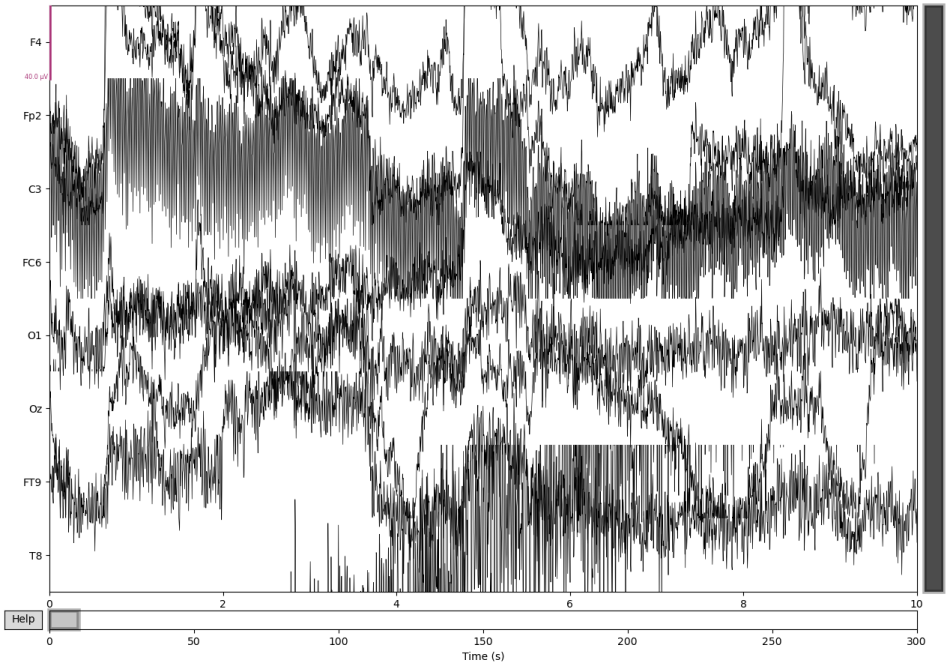


Figure 3.7: A section of a raw EEG, each signal correspond to a channel named on the left.

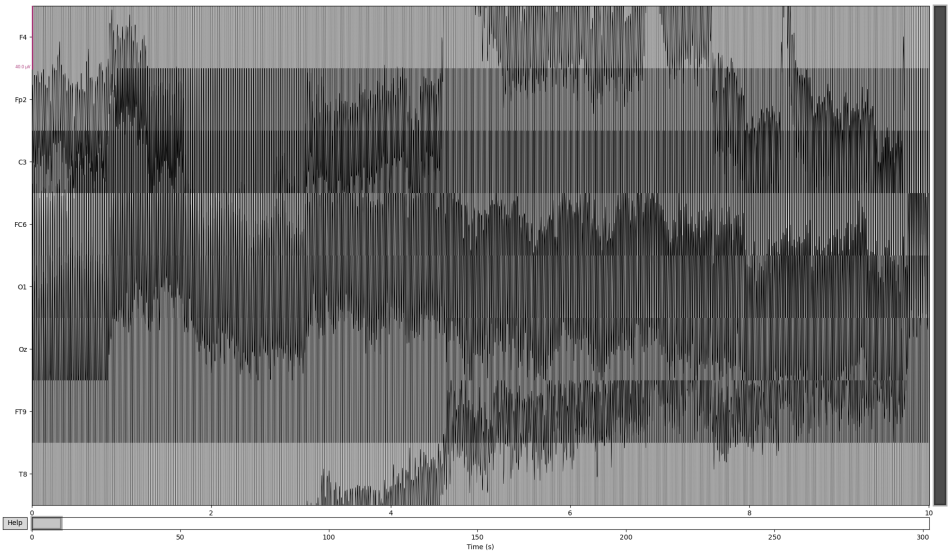


Figure 3.8: 10 second section of the raw EEG from Session 2 during considerable outside disturbances.

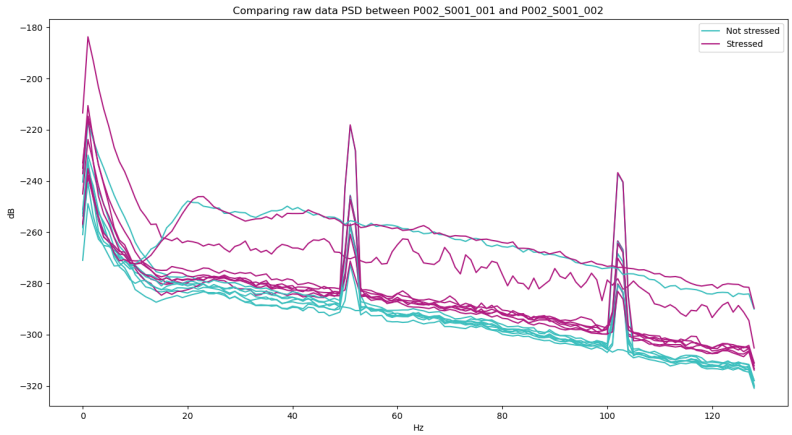


Figure 3.9: The Power Spectral Density (PSD) of two raw data recordings from Participant 2. The pink graph shows the PSD for a stressed recording, while the blue line shows the PSD for a non-stressed recording. The two recordings are from Participant 2, Session 1, Run 1 and Run 2.

raw EEG is coming from and now know that this needs filtering.

There appears to be some clear differences between the PSD of the stressed and not stressed signal in Figure 3.9. To examine this phenomenon closer, each channel have been plotted separately in Figure 3.10. Upon first glance, all channels but **T8** show a difference between not stressed and stressed signals. Channel Fp2 appear to have the greatest difference between not stressed and stress, and that the stress has increased Power Spectral Density compared to not stressed. The Fp2 channel is situated in the pre-frontal cortex.

The next step in the analysis is to examine the topographic maps (topomaps) of the PSD across the different frequency bands. This can give us information regarding which frequencies are most affected by stress. Figure 3.11a shows the topomaps from the non-stressed recording of Participant 2. The Delta and Theta band seem to have more active electrodes on the frontal part of the brain, implying that the frontal part is contributing to the low frequencies of the signal. The Alpha band is active for both the frontal and the temporal electrodes, yielding a larger area. The Beta and Gamma band is concentrated in the temporal part of the brain. Comparing this to Figure 3.11b, we can observe that there are differences in the Delta band where the PSD has a much higher value in the stressed recording. Likewise, in both the Theta and Alpha band the PSD is much higher when stressed. In addition the Alpha band has concentrated in the frontal part of the brain. There is not much of a difference between the stressed and non stressed Beta and Gamma bands.

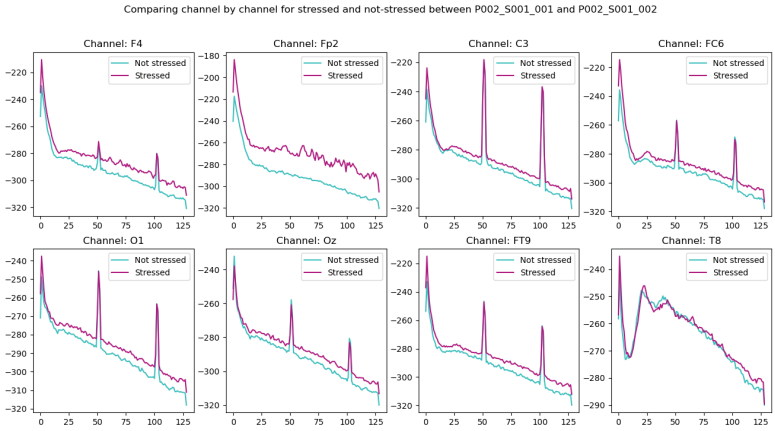
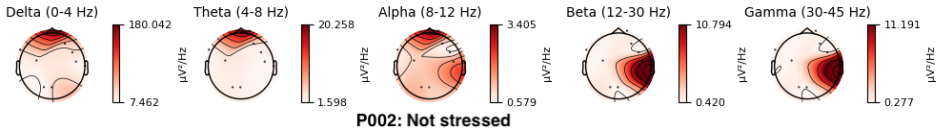
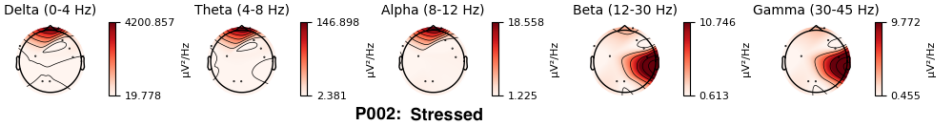


Figure 3.10: The Power Spectral Density (PSD) has been plotted for each channel in the recordings. The pink graph shows the stressed recording, while the blue graph shows the non-stressed recording. Top row, left to right shows the channels; F4, Fp2, C3m FC6, while the bottom row from left to right shows channels; O1, Oz, FT9 and T8.



(a) Topographic map of the PSD of participant 2 while not stressed. The frontal electrodes appear to be most active for both the Delta and the Theta bands, while Alpha is more impacted from the frontal and temporal electrodes. Beta and Gamma are most affected by the temporal electrodes.



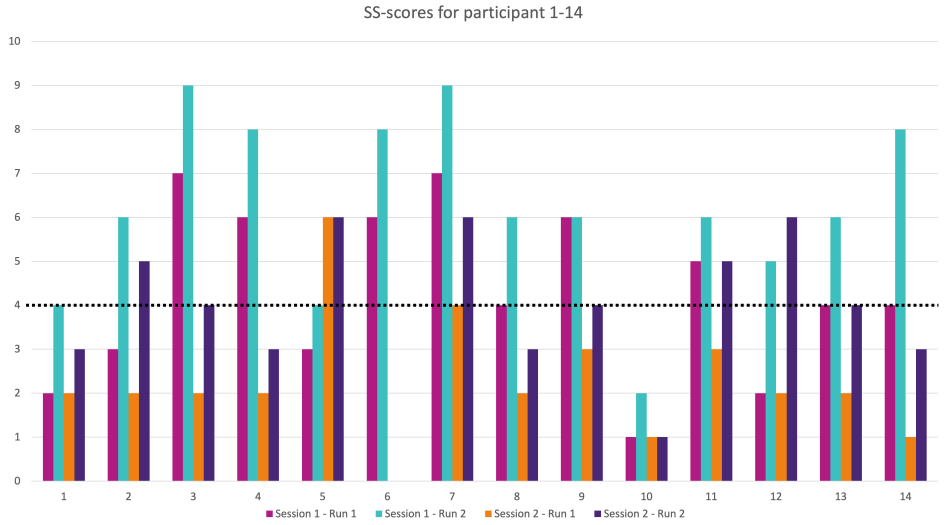
(b) Topographic map of the PSD of Participant 2 while stressed. The Delta, Theta and Alpha bands are most affected by the frontal electrodes, while Beta and Gamma are affected by the temporal electrodes.

Figure 3.11: Topographic maps of the PSD of the different frequency bands from a raw EEG signal.

3.2.3 Exploring the labels

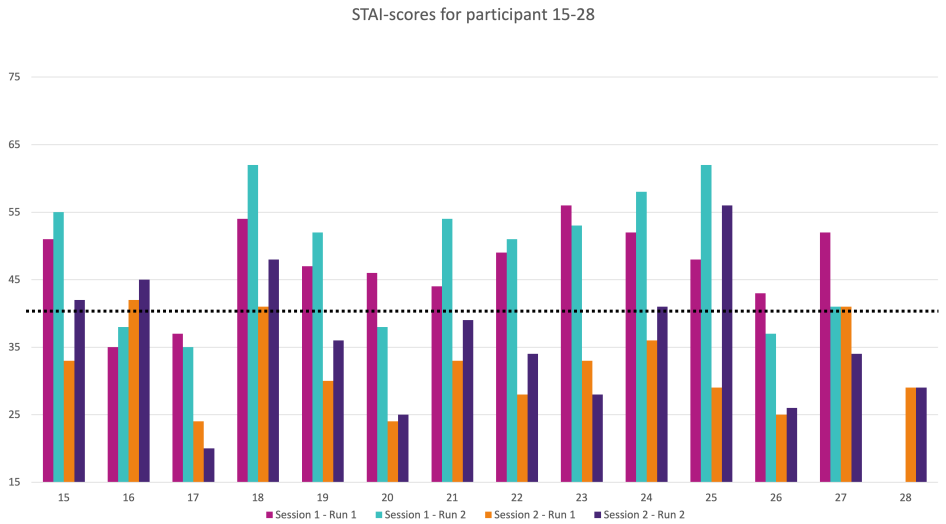
After scoring the questionnaires and determining the threshold values for both STAI-Y and SS, all the scores were plotted. In Figure 3.12 the scores for the first 14 participants are shown, where Figure 3.12a show the STAI-Y scores, and Figure 3.12b shows the SS scores. Likewise, Figure 3.13 shows the scores for participant 15-28. Upon closer analysis there are some discrepancies with how the scores will be labeled. Specifically, for Participant 5, the STAI-Y scores from both of the sessions are below the threshold value of 40, meaning that the recordings will be labeled to 0 meaning Not Stressed. While the SS scores of the same participant scores the two recordings in Session 2 as above the threshold meaning a label of 1 - Stressed. The results of this analysis have culminated in doing a t-test to further the insights into the differences between the label sets. Using `scipy.stats.ttest_ind` (SciPy), the resulting p-value will determine if the two label sets are similar or not. The t-test was done on the labels sets after they had been converted to strictly 0 and 1 labels, and the resulting p-value was 0.78, meaning that we cannot reject the null hypothesis stating that the samples have the same mean. I.e. the labels are similar.

(a) STAI-Y1 scores for Participants 1-14 with a threshold at 40.

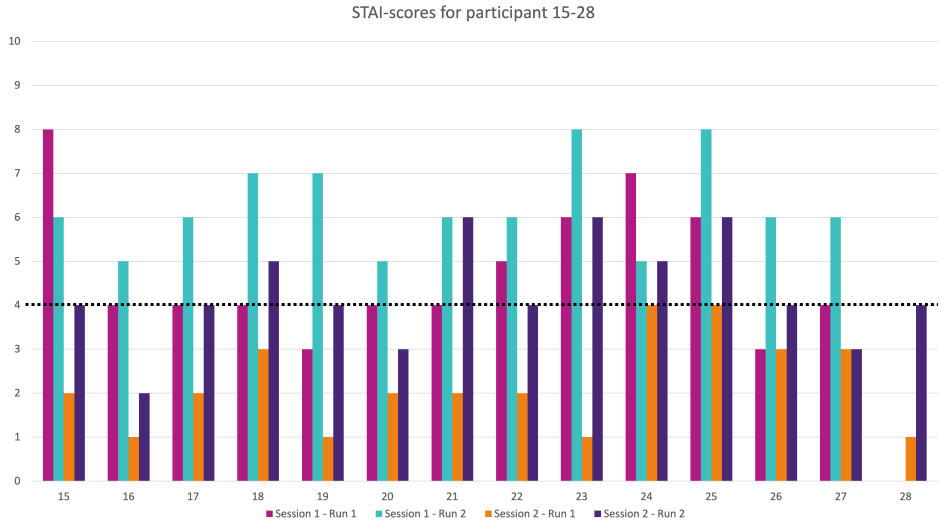


(b) SS scores for Participants 1-14 with a threshold at 4.

Figure 3.12: STAI-Y and SS scores for participants 1-14.



(a) STAI-Y1 scores for Participants 15-28 with a threshold at 40.



(b) SS scores for Participants 15-28 with a threshold at 4.

Figure 3.13: STAI-Y and SS scores for participants 15-28.

3.3 Pre-processing

For simplicity and efficiency, a class named `Filtering` was created to pre-process the data. The code can be found in Appendix E. The class encompasses all aspects of filtering, saving, and plotting the recordings, including loading the data, initial filtering, filtering with Signal-Space Projection (SSP), and decomposing the recordings by frequency bands.

Several functions were implemented in `Filtering`; an overview and description of the functions can be studied in Table 3.2. Most of this chapter uses the MNE Python library created by Larson et al. (2023); Gramfort et al. (2013). The resulting data sets will from now on be called *raw data*, *SSP data*, and *delta data*. Lastly, the function that prepares all the data sets for classification will be presented.

Table 3.2: Listing all the functions implemented in the `Filtering`-class with their descriptions

Functions	Descriptions
`load_data()`	Loads the raw .mat data recording for the given subject, session and run.
`save_ssp_data()`	Saves the filtered SSP data to a .mat-file.
`save_decomp_data()`	Saves the filtered decomposed data to a .mat-file.
`save_psd(data_type)`	Saves the PSD data to a .mat-file.
`init_filter()`	Filters the data with a basic band-pass filter with cutoffs at 1 and 50 Hz, then with a notch-filter at both 50 and 100 Hz, with a bandwidth of 0.5 Hz.
`compute_and_save_psd(data_type)`	Computes the PSD for a given data_type that is either `raw` or `filtered`, then saves the computed PSD using `save_psd(data_type)`. Here, `filtered` means data that has been passed through the `init_filter()`.
`decompose_data (freq_band, method)`	Decomposes the data using a band-pass filter based on the given frequency band and method. The frequency bands are γ, β, α, θ, and δ.
`vizualize_artifacts()`	Visualize the artifacts that will be filtered away with SSP, with built-in functions from the MNE toolbox. The functions create EOG epochs, apply a baseline and plot the artifacts.
`compute_ssp_projectors()`	Computes the SSP projectors.
`plot_eog_projectors()`	Plots the projectors.
`plot_ssp()`	Plots the EEG recordings with and without the projectors.
`plot_psd_comparison (filename1, filename2, title_psd, title_ch_comparison)`	Given two filenames and titles for the plots, this function makes comparison plots between the PSD recordings from the filenames. Used to compare the PSD of stressed and not-stressed recordings.

Before moving on to the detailed explanations of the filtering methods, there are a few steps in the `Filtering`-class that is common for all the methods:

Initialization: The first step is to initialize a `Filtering`-object. This is done by passing `subject_number`, `session_nr` and `run_nr` to `Filtering` and this corresponds to a specific recording.

Global values: `subject_number`, `session_nr` and `run_nr` are set as global variables for the `Filtering`-object.

Loading data: `load_data()` is then used to load the raw .mat-files, corresponding to the given subject number, session number an run number, into a global variable.

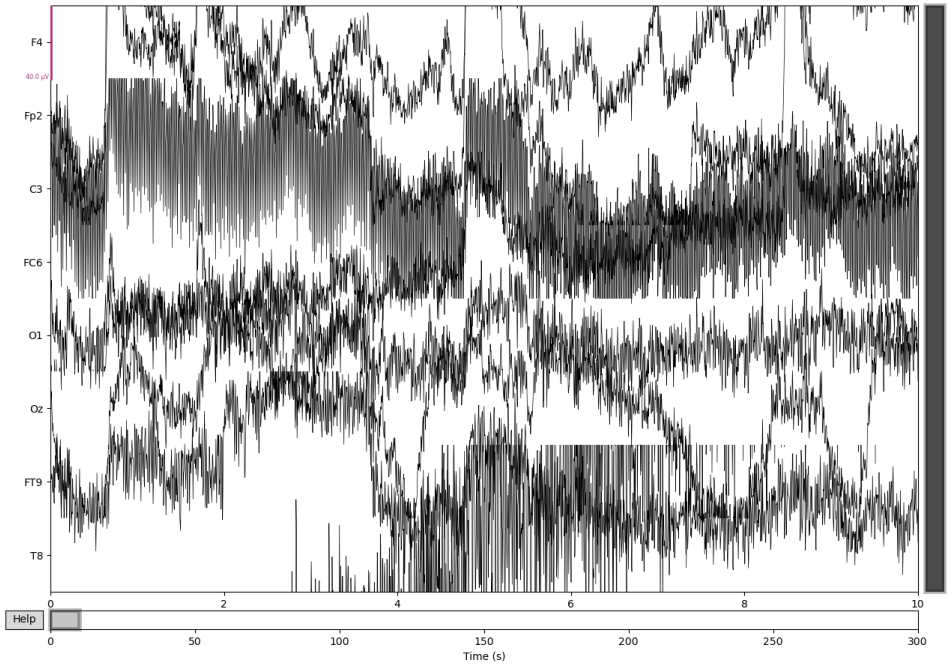
Creating a `RawArray`: `mne.create_info()` creates an info-object to be passed into `mne.io.RawArray()` along with the raw data, where `mne.io.RawArray()` creates a `RawArray`, that is stored in a variable called `raw_arr`.

Rename channels and set montage: To get the proper names of the channels we need to rename them using a built-in MNE function `mne.rename_channels`. Then define the montage that was used (the set-up of the electrodes) with `mne.set_montage`. Here, a 10-20 set-up was used as described in 2.1.1.

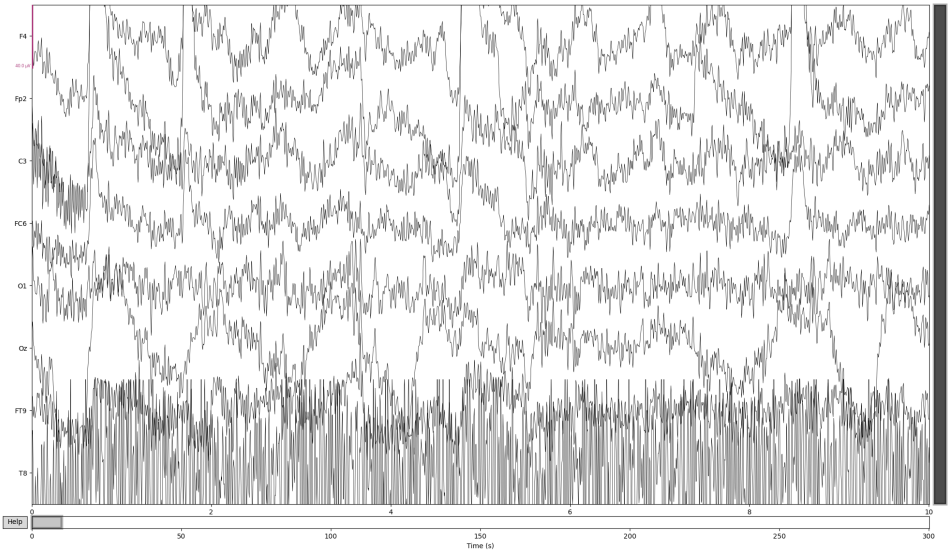
Initial filtering: Initial filtering was saved to a variable called `filtered_arr`. The filtering was done with `init_filter()` as described in Table 3.2 which included a band-pass filter and a notch filter.

3.3.1 Initial filtering

The idea behind doing an initial filtering is to remove both low and high frequency noise that is outside the range of where neurophysiological information lies. Starting with a simple band-pass filter, we remove frequencies below 1 Hz and above 50 Hz. In addition, as has been pointed out during the data exploration in Section 3.2, there are distinct peaks at both 50 and 100 Hz. Power line grids operate at 50 Hz and by creating a notch filter at 50 Hz we remove the noise that originates from the power lines. It is not apparent where the noisy peak at 100 HZ comes from, though we remove that as well with a notch filter. In Figure 3.14 the differences between the raw data and the data that has been filtered can be studied. The noise is noticeably reduced in Figure 3.14b compared to 3.14a, although there is still some noise and high-amplitude peaks (for example around 55 s) visible. Figure 3.15 confirms that the peaks from Figure 3.9 have been reduced by the notch filter.



(a) The raw EEG of Participant 2, Session 1, Run 1



(b) The EEG after going through initial filtering with bandpass filter and notch filter. The signal is less noisy than the raw signal, however there are still some noise present in channel T8.

Figure 3.14: A comparison of the raw EEG signal and the signal after initial filtering with a bandpass filter and a notch filter.

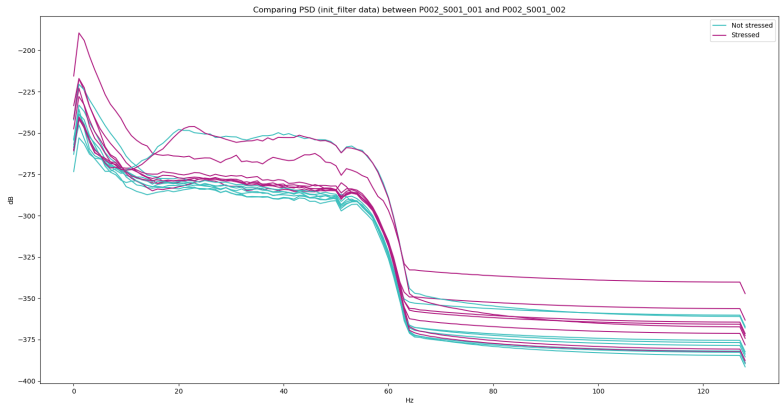


Figure 3.15: The PSD of Participant 2, Session 1, Run 1 and 2, where the high amplitude peaks at 50 Hz and 100 Hz have been reduced by the notch filter from the initial filtering.

3.3.2 Signal-Space Projection (SSP)

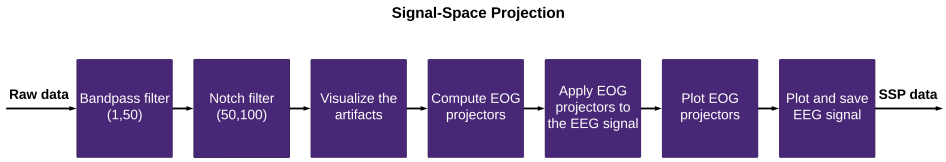
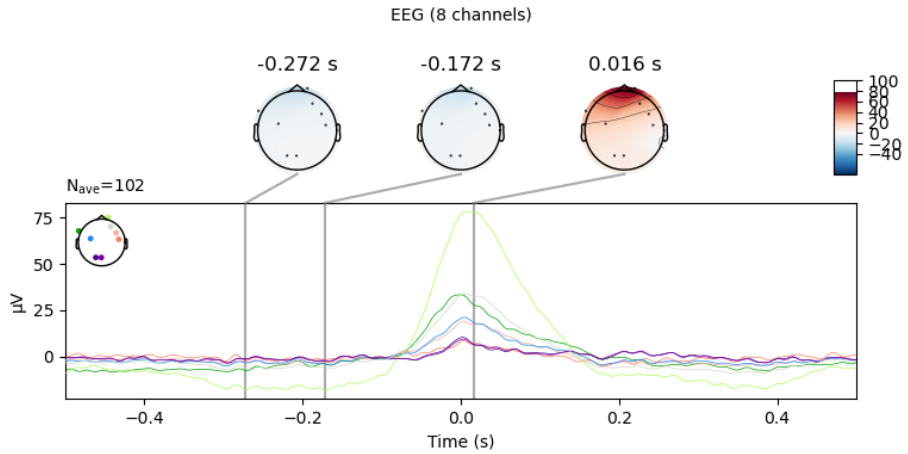


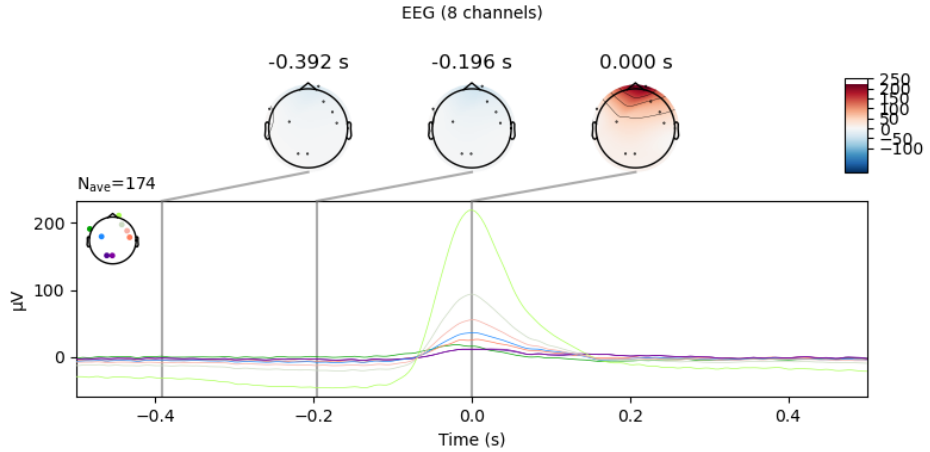
Figure 3.16: An overview of the steps of SSP filtering. The structure of the input, the raw data, is an array of all the recordings. Firstly, the raw data is bandpass filtered and notch filtered, before the artifacts are visualized. Following, the projectors are computed and applied to the EEG signal, then the projectors are plotted, and the new, SSP filtered data is saved as individual .mat-files, ready to be gathered into an array when its time for classification.

This Subsection will include a walk-through of the SSP filtering method on Participant 13, Session 1, Run 2 during the arithmetic test. An overview of the filtering method can be studied in Figure 3.16. The SSP filtering method usually makes use of dedicated Electrooculography (EOG)-channels to filter out eye blinks and other eye-related artifacts. This was not available during the data collection and thus the two electrodes closest to the eye muscles was chosen as a basis for the filtering. The two channels are Fp2 that lies on the pre-frontal cortex, and FT9 which lies on the frontotemporal part of the scalp, their placement can be seen in Figure 3.3. The first step is to visualize the artifact we want to remove. Firstly, we can observe the high-amplitude peaks in Figure 3.14b that resembles the blinking artifact described in Section 2.1.2. Figure 3.17 visualizes how the ocular artifact manifests across the channels. The faint, light green line represent channel Fp2, while dark green line is the channel FT9. The fact that both of these channels are present and

contributes to the artifact means that choosing the two channels for EOG artifacts was a reasonable choice.



(a) A visualization of the artifacts from the Fp2 channel. The faint light green line represents the Fp2 channel, and due to the high amplitude peak, there is clearly some blinking artifacts present.

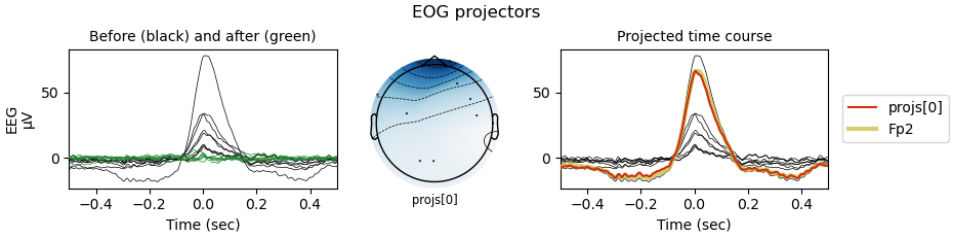


(b) A visualization of the artifacts present in channel FT9. The dark green line represents the channel, it appears as though the blinking artifact is not so present in this channel.

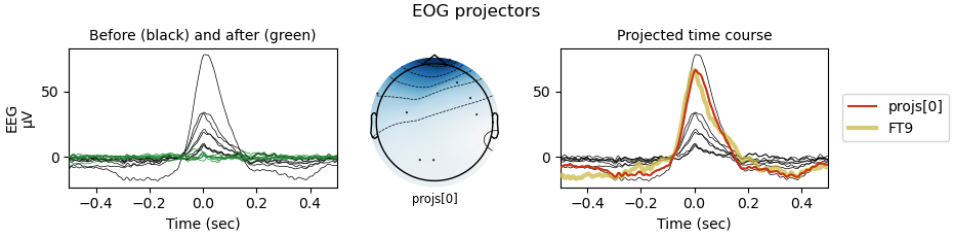
Figure 3.17: A visualization of the artifacts of the EEG signal. The plot shows a short epoch where it detected EOG artifacts, showing the distribution of the artifact across the channels.

Confirming that there are in fact some artifacts that can be removed, the next step is to compute the SSP projectors. In Figure 3.18, the left column shows the data traces before

and after applying the projectors. The center column shows the topographic maps that is associated with each projector, here we see that the projectors are concentrated towards the front of the skull. The right column shows the data traces in black once again, though this time they are also projected onto the projector for each channel, the trace in red. In addition, the graph show a surrogate ground truth for the chosen channel.



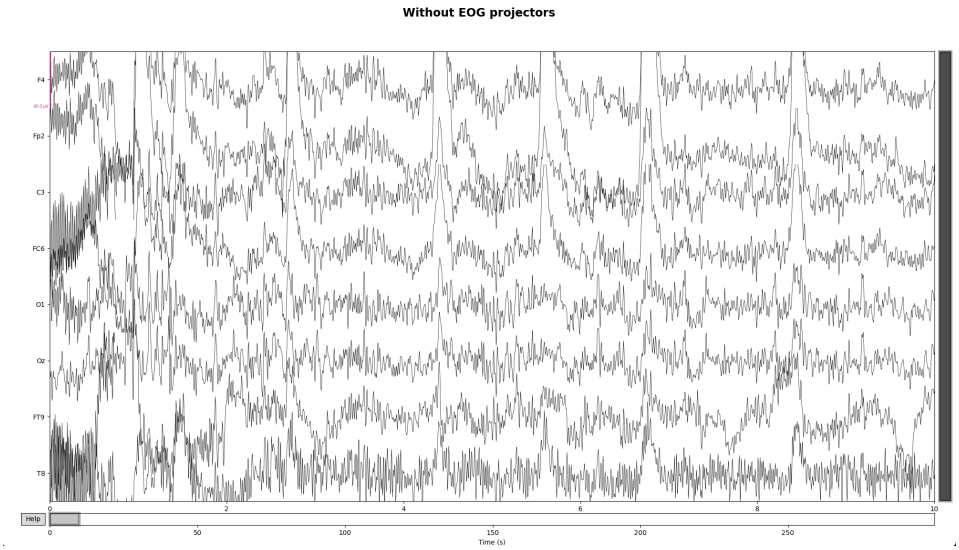
(a) Showing the EOG projectors for channel Fp2.



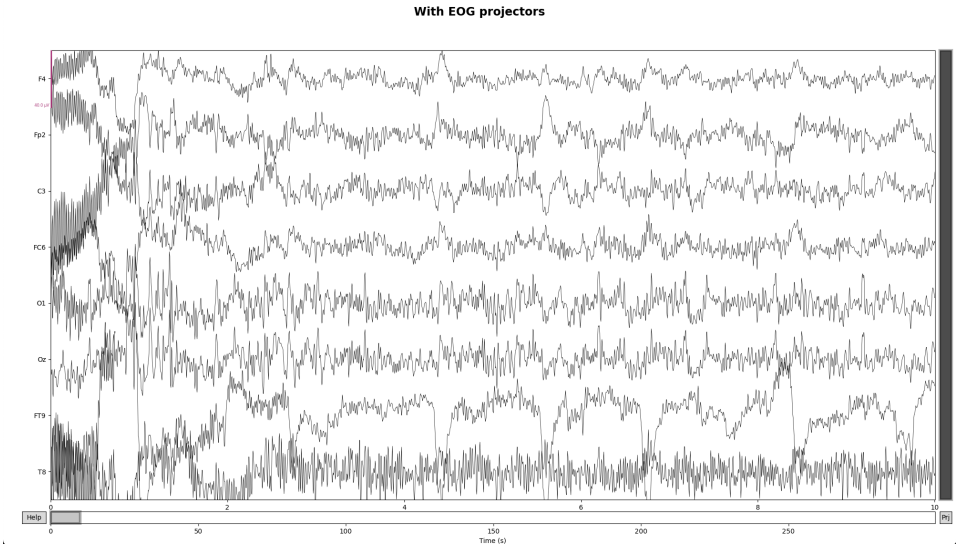
(b) Showing the EOG projectors for channel FT9.

Figure 3.18: A visualization of the projectors showing the data trace before and after projection in the left column, the topographic map of the projectors in the middle column, and the right column shows the data traces in black, along the projectors in red and the yellow represents the ground truth of each channel.

After applying the projectors to the data it is important to visualize the effect that the SSP-filtering had on the data. Figure 3.19 shows the difference between data that has gone through the initial filtering (Figure 3.19a) and data that has been though both initial filtering and SSP-filtering (Figure 3.19b). The most noticeable difference it that the high-amplitude peaks have been reduced. Still, there is significant noise in the signal.



(a) Showing the EEG of the signal withput EOG projectors, note the high amplitude peaks at around 3s, 4.5s, 5.5s 7s, and 8.5s

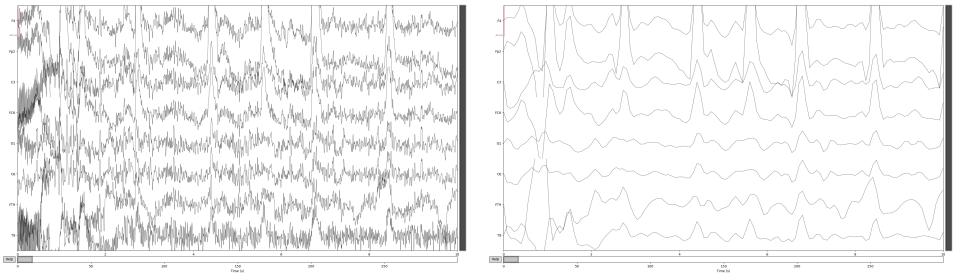


(b) The EEG signals after applying EOG projectors. The high amplitude peaks from the image above have been reduced, though there is still some noise present in the signal.

Figure 3.19: Shows a comparison of EEG signal with and wihtout EOG projectors active.

3.3.3 Signal decomposition

From the exploration of the data in Section 3.2 the conclusion was that stress seems to be relatively active in the Delta (δ) frequency band. Signal decomposition can be obtained by creating a filter bank, using the built in `mne.filter()` function and defining the frequency cutoff. The Delta band ranges from 0-4Hz, so the lower cutoff frequency will be zero and higher cutoff frequency will be 4. Figure 3.20 shows the difference in the initially filtered data and the decomposed Delta band of the EEG data.



(a) The EEG after initial filtering of Participant 13, Session 1, Run 2

(b) The EEG of the Delta-band of Participant 13, Session 1, Run 2

Figure 3.20: A visualization of the difference between initially filtered data and Delta-band data.

3.3.4 Preparing the data for classification

`prepare_data.py` is a script that contains all the necessary steps to prepare the data for classification, depending on which data set, label type, epoch duration, and feature type needed for classification. The script can be found in Appendix A. All the different recordings will be regarded as one set of data, meaning that the classification will not be performed subject wise. `data_type, label_type, epoch_duration` and `feature_type` are all parameters that are passed to the function `prepare_data(...)`. All the Python-functions in the script are listed in Table 3.3. After splitting the data sets into train and test sets, the parameter `feature_type` determines if the features of the train/test sets should be calculated or not, and if so then the features are calculated. The EEGNet classifier does not use features, therefore the option of setting `feature_type = None` exists.

Table 3.3: Listing all the functions implemented in the `prepare_data(data_type, label_type, epoch_duration, feature_type)`-function with their descriptions.

Functions	Descriptions
`get_valid_recordings (data_type)`	Finds all the valid recordings based on the data type.
`extract_eeg_data (valid_recordings, data_type)`	Extracts the EEG-recordings based on the valid recordings and the data type.
`get_stai_labels (valid_recordings, cutoff)`	Fetches the STAI-Y-labels for each of the valid recordings. The cutoff denotes the value at which to separate between stressed and not stressed labels. Default value is 40.
`get_pss_labels (valid_recordings, threshold)`	Fetches SS-labels for the valid recordings, the threshold denotes the value at which to separate the stressed and not stressed labels. Default threshold is 4.
`segment_data(data, labels, epoch_duration)`	Fetches SS-labels for the valid recordings, the threshold denotes the value at which to separate the stressed and not stressed labels.
`create_train_test_split(segmented_data, segmented_labels, epoch_duration)`	Splits the segmented data into train and test sets.

3.4 Features

This section will focus on presenting the methods used for feature extraction. The theory behind each feature is described in more detail in 2.3, where all the features mentioned are implemented in this section, while the code can be found in Appendix B. All the features are a part of the Python library MNE-features (MNE-features). Table 3.4 lists all the functions implemented in `features.py`, the file in which all the features are calculated.

Table 3.4: Listing all the functions implemented in the `features.py`-file with their descriptions.

Functions	Descriptions
`time_series_features(data)`	Calculates the three time-series features using the MNE-features functions `compute_ptp_amp(data)`, `compute_variance`, `compute_rms(data)`
`entropy_features(data)`	Calculates the four different entropy features using `compute_app_entropy(data)`, `compute_samp_entropy(data)`, `compute_spect_entropy(data, sfreq)`, `compute_svd_entropy(data)`
`hjorth_features(data)`	Calculates the two Hjorth features using `compute_hjorth_mobility_spect(sfreq, data)`, `compute_hjorth_complexity_spect(sfreq, data)`
`psd_features(data)`	Calculates the PSD of the five frequency bands mentioned in Section 2.1.3 using `compute_pow_freq_bands(sfreq, data)`

3.5 Machine learning

This section will present the methods for machine learning utilized in this thesis. As the interest and research has increased on the topic of machine learning and neural networks, many open-source libraries have been created that incorporates the most common machine learning models. SVM, RF and KNN uses the Scikit-learn Python library (Pedregosa et al., 2011). All the three machine learning classifiers utilize Scikit-learn's `sklearn.model_selection.GridSearchCV()` to tune their respective hyperparameter grids, to ensure that the most optimal hyperparameters are being used. `GridSearchCV` uses the grid search optimization algorithm with cross-validation, with a 'fit' and 'score' method. It exhaustively search the grid space, meaning that it tests all possible combinations of the parameter grid, and retains the best possible model. The search is run with 10-fold cross-validation, and the refit parameter is set to `True` so the

data will be re-fitted with the best parameters from the search. The specific hyperparameter grids will be presented in the sections to come. EEGNet is a neural network from Lawhern et al. (2018) utilizing the Python-library `Tensoflow` with the high-level API `keras` (Abadi et al., 2015).

Each classifier is being predicted with an array of different input data, label types, features and epoch duration. The classification is done as a general model where the all the recordings are thought of as one data set. There are three different data sets; *raw data* containing the raw data of the recordings, meaning no filtering methods have been applied, *SSP data* containing the data that has been filtered with a bandpass filter, notch filter and by the SSP filtering method, and *delta data* (denoted "decomp" in the code) where the data has been filtered with bandpass filter and notch filter, and then lowpass-filtered from 0 - 4 Hz. There are two types of labels as described in Section 3.2.3; the labels obtained from the STAI-Y-method denoted STAI-Y-labels, and the label set from the SS-method denoted SS-labels. The different features have been introduced in Section 3.4, consisting of *time-series-*, *entropy-*, *Hjorth-*, and *PSD-features*. Lastly, predictions will be computed with three different epoch durations; 1s, 2s, and 5s. The code implementation can be studied in Appendix C.

3.5.1 Support vector machine (SVM)

Support Vector Machine (SVM) was implemented with Scikit-learn's `sklearn.svm.SVC()`. Considering that SVM creates a hyperplane to separate the different classes, it is important that the data is normalized. This was achieved using `sklearn.preprocessing.StandardScaler` which removes the mean and scales the data to unit variance. The regulation parameter `C` and the `kernel` were selected for hyperparameter tuning. `GridSearch()` searches through the parameter grid where `C` iterates through the values $[0.01, 0.1, 1, 10, 100, 1000, 10000]$ and the `kernel` is either `poly, sigmoid, linear,` or `rbf`.

3.5.2 Random forest (RF)

Random Forest (RF) was implemented using `sklearn.ensemble.RandomForestClassifier()`. RF is not sensitive to the distance between data points thus the data does not need to scaled for classification. The parameter grid consist of number of estimators (`n_estimators`) iterating through $[50, 75, 100, 125, 150, 175, 200, 225, 250]$, maximum number of features used for each tree (`max_features`) that is either `auto, sqrt` or `log2`, and the maximum depth of the tree (`max_depth`) iterating through $[3, 5, 7, 9]$. The selected parameters are an important to the performance of the classifier, and thus were chosen in order to find the best representative parameters.

3.5.3 K-Nearest Neighbor (KNN)

K-Nearest Neighbor (KNN), like SVM, is dependant on the distance between the data points. Therefore, the data was scaled using `sklearn.preprocessing.StandardScaler` prior to classification. The selected

hyperparameters of the KNN was number of neighbors that decides how many neighbors are to be taking into account when deciding the nearest class, leaf size which affects the speed of construction of the algorithm used for finding the nearest neighbors, and lastly, the weights parameter which determines how to weigh the neighbors. Number of neighbors (`n_neighbors`) iterates through the values $[1, 5, 9, 15, 19, 25, 29, 35, 39]$, where all the possible values are odd to ensure that there will always be a majority when voting for the nearest neighbors. The leaf size (`leaf_size`) varies between $[5, 10, 20, 30, 40, 50]$. The `weight` parameter only has two option, either `uniform` or `distance`. If the weight is uniform then all neighbors will be weighted equally, if it is distance, then the neighbors closer to the new point will have more influence when deciding the class.

3.5.4 EEGNet

EEGNet uses training, validation and test sets to perform classification. The first step in this classifier is to split the training set returned from `prepare_data` described in Section 3.3.4, into a new training set and a validation set. `sklearn.model_selection.train_test_split()` was used to split into the new training and validation set, using 25% for validation. The EEGNet model is mostly run on the default parameters from the original model. The only parameter that changed in this implementation is the kernel length (`kernLength`. The kernel length was defined as half the sample frequency, thus the parameter was set to 125 in my implementation. The epoch duration was also added as an input parameter to the function to ensure that the model would work with varying epoch lengths. As the model itself creates features, the data did not go through feature extraction prior to classification with EEGNet.

3.6 Classification metrics

To evaluate the performance of the classifiers, certain performance metrics are used. In clinical research the most common metrics are **accuracy**, **sensitivity** and **specificity**. To measure the metrics we need to define a confusion matrix. A confusion matrix is a 2x2 matrix that reports on the number of true positive, true negative, false positives and false negatives when comparing predictions from a classifier with ground truth labels. The **true positives (TP)** are defined as a prediction that correctly indicates the presence of a condition. In this case it would be to correctly indicate that a person is stressed. **True negative (TN)** means that the prediction correctly classifies the absence of a condition, in this case that would be to correctly indicate that a person is not stressed. **False positive (FP)** occurs when the prediction wrongly indicates that a condition is present, like if the prediction stated that the person was stressed, when they were not. **False negative (FN)** means that the prediction indicates that a condition is not present when it is in fact present, meaning that the prediction states that the person is not stressed, when they actually are. These definitions help us to further define the accuracy, sensitivity and specificity of a classifier where

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$\text{Sensitivity} = \frac{TP}{TP + FN}$$
$$\text{Specificity} = \frac{TN}{FP + TN}.$$

The metrics have been calculated using `sklearn.metrics.confusion_matrix` and `sklearn.metrics.ConfusionMatrixDisplay`. The code can be found in Appendix D.

4

Experimental results

The chapter will present all the results from the experiment and the methods detailed in Chapter 3. The structure of this chapter is as follows: the results from the three classifiers SVM, RF and KNN, will first be divided into sections by feature type, then each classifier and their results are presented in tables outlining which data set, epoch duration and label type has been tested. Lastly, the results from EEGNet will have its own section, as this classifier does not use features as input, only the data itself. The tables report on accuracy, sensitivity and specificity. The tbales have been constructed this way to be able to compare the performance of the different label types, epoch durations and data sets. There is an extensive number of results from this experiment, with over 200 unique results. For each combination of classifier, feature type, data set, epoch duration and label type, their respective confusion matrices can be found from Appendix G - K. The structure of the Appendices matches the structure in this Chapter. The complete overview of the result is presented in Appendix F, where all the hyperparameters from the GridSearch is also noted.

Note: the SVM classifier using `sklearn.svm.SVC()` is very sensitive to the size of the data set in regards to fit time. It scales quadratically with number of samples, hence when splitting the data into epochs of both 1 and 2 seconds, the run time of the algorithm is very lengthy. The run-time for PSD-features, which yields the largest amount of samples, with 2 second epochs averages around 12 hours. Therefore, the results for the SVM classifier is not complete. Results that have not been computed have been marked with '-' in the tables.

4.1 Time-series features

This Section will present all the results regarding time-series features for each type of classifier.

4.1.1 Support Vector Machine (SVM)

The results from the SVM with time-series features is presented in Table 4.1. Though the table is not complete due to the reasons stated in the note at the beginning of the chapter, most of the combinations yields a higher accuracy for the SS label type. For the data sets SSP and Delta-band, many of the results yield a low sensitivity and a high specificity, indicating that the classifier is mostly predicting Not Stressed, which is the majority class in the data. Across the epoch duration there are some discrepancies like between raw data with 2 and 5 second epochs with SS labels, where 2s epochs yield 70.67% while 1s epochs yields 55.08%.

Table 4.1: Accuracy, sensitivity and specificity for time-series features classified by SVM. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type used.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	56.15%	55.24%	56.19%
		SS	50.46%	**77.26%**	34.38%
	2s	STAI-Y	69.16%	50.27%	**85.13%**
		SS	**70.67%**	66.29%	**73.29%**
	5s	STAI-Y	69.00%	52.70%	**82.79%**
		SS	55.08%	**73.26%**	44.18%
SSP	1s	STAI-Y	-	-	-
		SS	-	-	-
	2s	STAI-Y	46.62%	19.36%	69.70%
		SS	61.65%	6.38%	**93.24%**
	5s	STAI-Y	45.34%	18.64%	67.93%
		SS	62.56%	34.75%	**78.45%**
Delta-band	1s	STAI-Y	-	-	-
		SS	-	-	-
	2s	STAI-Y	-	-	-
		SS	62.61%	0.75%	**99.73%**
	5s	STAI-Y	60.52%	23.27%	**92.05%**
		SS	62.50%	0.00%	**100.00%**

4.1.2 Random Forest (RF)

The results from the RF classifier with time-series features is shown in Table 4.2. The raw data set is performing at a higher accuracy across all combinations of epoch and labels, than Signal-Space Projection (SSP) and Delta-band. The SS-labels in general yields a higher sensitivity than that of the STAI-Y-labels. The sensitivity is also lower on both SSP and Delta-band data than for raw data. Across epoch duration the results are quite consistent, though a slight drop in accuracy is noted for 2 second epoch compared to the other epochs. The best combination with the highest accuracy, sensitivity and specificity was raw, with 1s epoch and SS-labels, yielding an accuracy of 70.97%, sensitivity of 77.78% and specificity of 66.89%.

Table 4.2: Accuracy, sensitivity and specificity for time-series features classified by RF. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	**70.33%**	59.29%	**79.68 %**
		SS	**70.97%**	**77.78%**	66.89%
	2s	STAI-Y	61.10%	42.34%	**76.97%**
		SS	69.18%	**72,48%**	67.30%
	5s	STAI-Y	**70.62%**	46.38%	**91.13%**
		SS	**70.27%**	**70.06%**	**70,40%**
SSP	1s	STAI-Y	51.21%	14.47%	**82.30%**
		SS	63.56%	23.41%	**86.50%**
	2s	STAI-Y	49.02%	18.91%	**74.50%**
		SS	65.16%	45.72%	**76.27%**
	5s	STAI-Y	48.94%	25.12%	69.10%
		SS	66.87%	40.25%	**82.08%**
Delta-band	1s	STAI-Y	56.59%	17.82%	**89.40%**
		SS	62.71%	19.47%	**88.65%**
	2s	STAI-Y	58.05%	24.65%	**86.32%**
		SS	61.49%	31.84%	**79.28%**
	5s	STAI-Y	57.77%	25.58%	**85.01%**
		SS	65.32%	39.36%	**80.90%**

4.1.3 K-Nearest Neighbor

The results from time-series features classified with KNN classifier is shown in Table 4.3. All the results have a higher accuracy with SS-labels. The same is true for sensitivity except for the combination of raw data and 5s epoch, while for specificity the exception is for SSP data with 1s epoch. The results are quite similar across raw data and SSP data, while Delta-band performs slightly better regarding accuracy and specificity. Best result across the three metrics is Delta-band with 5s epoch and SS labels.

Table 4.3: Accuracy, sensitivity and specificity for time-series features classified by KNN. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	52.20%	47.80%	55.93%
		SS	60.09%	53.81%	63.86%
	2s	STAI-Y	45.13%	45.09%	45.17%
		SS	50.76%	47.05%	52.66%
	5s	STAI-Y	45.55%	50.54%	41.33%
		SS	46.82%	43.88%	48.59%
SSP	1s	STAI-Y	50.60%	31.74%	66.56%
		SS	57.57%	46.70%	63.78%
	2s	STAI-Y	49.27%	36.30%	60.25%
		SS	58.69%	40.94%	68.84%
	5s	STAI-Y	44.77%	36.36%	51.89%
		SS	53.93%	46.61%	58.11%
Delta-band	1s	STAI-Y	54.85%	30.01%	**75.87%**
		SS	63.82%	44.11%	**75.65%**
	2s	STAI-Y	55.23%	32.52%	**74.45%**
		SS	62.53%	45.34%	**72.84%**
	5s	STAI-Y	55.72%	33.90%	**74.19%**
		SS	64.19%	49.72%	**72.88%**

4.2 Entropy features

This Section will present all the results with Entropy features, across the three classifiers.

4.2.1 Support Vector Machine

The results of SVM using Entropy features is presented in Table 4.4. With the exception of raw data with 5s epochs, SS has a higher accuracy than STAI-Y. However, by inspecting the sensitivity it becomes apparent that the SS labels are performing quite poorly. Many combinations yields a very low sensitivity with raw data with 2s and SS-labels has the worst sensitivity of 0.15%. By inspecting the confusion matrix for this combination in Figure H.1e, it is evident that the classifier is mostly predicting Not Stressed.

Table 4.4: Accuracy, sensitivity and specificity for entropy features classified by SVM. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	-	-	-
		SS	55.82%	50.84%	58.82%
	2s	STAI-Y	57.33%	31.97%	**78.78%**
		SS	62.25%	0.15%	**99.51%**
	5s	STAI-Y	58.62%	42.06%	**72.62%**
		SS	46.54%	56.31%	40.68%
SSP	1s	STAI-Y	-	-	-
		SS	-	-	-
	2s	STAI-Y	54.31%	32.46%	**72.79%**
		SS	57.26%	22.06%	**77.37%**
	5s	STAI-Y	53.11%	31.74%	**71.19%**
		SS	59.40%	19.07%	**82.45%**
Delta-band	1s	STAI-Y	-	-	-
		SS	-	-	-
	2s	STAI-Y	-	-	-
		SS	62.56%	0.82%	**99.60%**
	5s	STAI-Y	55.30%	13.25%	**90.87%**
		SS	62.15%	21.66%	**86.44%**

4.2.2 Random Forest

The results from RF classifier with Entropy features is presented in Table 4.5. For raw data the accuracy and specificity is higher for STAI-Y labels compared to SS labels, though the sensitivity is lower. Raw data, in general, performs marginally better than SSP and Delta-band. The accuracy of 1s epoch with SSP data and SS labels is 4 percentage points higher than the other epochs. While the the accuracy decreases with decreasing epohc duration for raw data with SS labels. The specificity is high across all combinations.

Table 4.5: Accuracy, sensitivity and specificity for entropy features classified by RF. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	66.22%	39.84%	**88.55 %**
		SS	57.33%	58.42%	56.68%
	2s	STAI-Y	66.67%	41.31%	**88.13%**
		SS	60.10%	56.90%	61.88%
	5s	STAI-Y	65.75%	43.91%	**84.22%**
		SS	62.85%	68.55%	59.44%
SSP	1s	STAI-Y	52.33%	25.36%	**75.15%**
		SS	60.55%	49.08%	67.10%
	2s	STAI-Y	58.33%	39.96%	**73.88%**
		SS	56.89%	49.16%	61.31%
	5s	STAI-Y	57.06%	40.22%	**71.32%**
		SS	55.32%	45.13%	61.14%
Delta-band	1s	STAI-Y	55.46%	12.92%	**91.46%**
		SS	62.57%	19.81%	**88.23%**
	2s	STAI-Y	56.54%	20.50%	**87.04%**
		SS	61.41%	29.90%	**80.31%**
	5s	STAI-Y	56.85%	30.05%	**79.53%**
		SS	62.43%	41.24%	**75.14%**

4.2.3 K-Nearest Neighbors

The results of classification using KNN and Entropy features are shown in Table 4.6. Raw data is performing better than the other data sets, with both higher accuracy and specificity. The sensitivity is somewhat similar across the three data sets. For different combinations the higher accuracy varies between the different label types. For raw data the best performing epoch is 2s, while 2s is the epoch with the lowest accuracy for the other data sets. The best result is achieved with raw data, 2s epoch and SS labels, with an accuracy of 70.36%, sensitivity of 45.49% and specificity of 85.28%.

Table 4.6: Accuracy, sensitivity and specificity for entropy features classified by KNN. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	63.50%	29.92%	**91.92%**
		SS	67.88%	42.55%	**83.08%**
	2s	STAI-Y	62.92%	29.96%	**90.81%**
		SS	**70.36%**	45.49%	**85.28%**
	5s	STAI-Y	64.34%	39.45%	**85.40%**
		SS	63.21%	50.66%	**70.73%**
SSP	1s	STAI-Y	54.82%	38.67%	68.48%
		SS	55.72%	52.42%	57.60%
	2s	STAI-Y	54.73%	42.40%	65.15%
		SS	51.95%	50.84%	52.59%
	5s	STAI-Y	57.34%	46.53%	66.49%
		SS	52.08%	47.25%	54.84%
Delta-band	1s	STAI-Y	54.25%	24.90%	**79.08%**
		SS	58.47%	37.35%	**71.15%**
	2s	STAI-Y	55.70%	30.02%	**77.44%**
		SS	57.66%	42.88%	66.53%
	5s	STAI-Y	55.65%	38.21%	**70.40%**
		SS	58.83%	57.63%	59.55%

4.3 Hjorth features

The results of classification using Hjorth features are presented in this Section.

4.3.1 Support Vector Machine (SVM)

The results of SVM classification on Hjorth features are presented in Table 4.7. Upon first glance, the specificity across most of the combinations stand out. The specificity is either at a 100% or close to that, while the sensitivity is close to 0.00%. This means that the classifier predicts mostly Not Stressed. Raw data with 1s epoch and SS and SSP data with 5s epoch and SS labels are the only two combinations that are not skewed towards predicting only Not Stressed. SS labels seem to yield more results where the sensitivity and specificity is exactly 0.00% and 100%, respectively.

Table 4.7: Accuracy, sensitivity and specificity for Hjorth features classified by SVM. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	-	-	-
		SS	52.51%	**77.78%**	53.36%
	2s	STAI-Y	51.23%	9.52%	**86.53%**
		SS	62.50%	0.00%	**100.00%**
	5s	STAI-Y	51.98%	1.69%	**94.52%**
		SS	62.50%	0.00%	**100.00%**
SSP	1s	STAI-Y	-	-	-
		SS	-	-	-
	2s	STAI-Y	53.13%	1.83%	**96.54%**
		SS	63.64%	0.00%	**100.00%**
	5s	STAI-Y	52.90%	2.47%	**95.57%**
		SS	42.37%	34.96%	46.61%
Delta-band	1s	STAI-Y	-	-	-
		SS	-	-	-
	2s	STAI-Y	-	-	-
		SS	62.50%	0.00%	**100.00%**
	5s	STAI-Y	57.52%	12.63%	**95.31%**
		SS	62.50%	0.00%	**100.00%**

4.3.2 Random Forest (RF)

The results from the classification using RF with Hjorth features can be studied in Table 4.8. Across the board, raw data with SS labels has the best performance. The accuracies are all around 73%, sensitivity around 77% and the specificity around 71%. There is no major differences between epoch length for the raw data data. Comparatively, the STAI-Y labels are performing quite poorly, yielding accuracies around 54%, sensitivity around 43% and specificity around 63%. Both SSP and Delta-band data have very high specificity, however, the sensitivity of the Delta-band is very low.

Table 4.8: Accuracy, sensitivity and specificity for Hjorth features classified by RF. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	54.08%	43.20%	63.29%
		SS	**73.65%**	**77.33%**	**71.44%**
	2s	STAI-Y	53.13%	42.34%	62.26%
		SS	**73.60%**	**77.63%**	**71.19%**
	5s	STAI-Y	54.80%	43.91%	64.02%
		SS	**73.94%**	**77.78%**	**71.64%**
SSP	1s	STAI-Y	58.49%	31.35%	**81.45%**
		SS	64.06%	22.45%	**87.84%**
	2s	STAI-Y	58.86%	31.79%	**81.78%**
		SS	66.35%	26.93%	**88.88%**
	5s	STAI-Y	56.36%	31.28%	**77.57%**
		SS	62.71%	36.44%	**77.72%**
Delta-band	1s	STAI-Y	54.79%	4.20%	**97.61%**
		SS	63.13%	1.97%	**99.82%**
	2s	STAI-Y	55.01%	5.43%	**96.95%**
		SS	62.82%	1.27%	**99.73%**
	5s	STAI-Y	55.65%	8.23%	**95.70%**
		SS	64.69%	6.59%	**99.55%**

4.3.3 K-Nearest Neighbors (KNN)

The results following classification with KNN using Hjorth features is presented in Table 4.9. The accuracy is ranging from 51% to 62% across all combinations except, for raw data with 5s epoch using SS labels, where the accuracy is as low as 48.80%. There is a 12 percentage point difference in the accuracy between the SS label and STAI-Y label in this case. Generally, Delta-band data performs worse regarding sensitivity than the other data sets, in return the specificity is high. For raw data with STAI-Y labels the accuracy increases with increasing epoch duration, while it decreases for SS labels. For the other two data sets, the accuracy does not change dramatically with varying epochs.

Table 4.9: Accuracy, sensitivity and specificity for Hjorth features classified by KNN. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	52.76%	37.18%	65.94%
		SS	55.23%	58.64%	53.18%
	2s	STAI-Y	58.67%	38.50%	**75.74%**
		SS	51.37%	47.43%	53.74%
	5s	STAI-Y	60.73%	59.01%	62.19%
		SS	48.80%	46.14%	50.40%
SSP	1s	STAI-Y	57.93%	52.30%	62.70%
		SS	59.97%	54.39%	63.16%
	2s	STAI-Y	58.05%	50.40%	64.53%
		SS	59.61%	54.70%	62.42%
	5s	STAI-Y	58.26%	55.78%	60.37%
		SS	57.94%	56.14%	58.96%
Delta-band	1s	STAI-Y	52.08%	21.25%	**78.16%**
		SS	60.91%	37.57%	**74.92%**
	2s	STAI-Y	53.38%	23.73%	**78.47%**
		SS	61.47%	37.29%	**75.97%**
	5s	STAI-Y	54.80%	31.59%	**74.45%**
		SS	61.79%	50.28%	68.70%

4.4 Power Spectral Density (PSD) features

The results from classification with Power Spectral Density (PSD) features will be presented in this Section.

4.4.1 Support Vector Machine (SVM)

The classification results from SVM using PSD features are presented in Table 4.10. Raw data with 1s and 2s epoch perform better with STAI-Y labels than SS labels by a good margin. For 5s epoch SS labels perform marginally better for the accuracy. For SSP and Delta-band data the SS labels perform better regarding accuracy, though for Delta-band data the sensitivity is near zero.

Table 4.10: Accuracy, sensitivity and specificity for PSD features classified by SVM. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	68.90%	35.97%	**96.76%**
		SS	58.95%	7.06%	**90.08%**
	2s	STAI-Y	68.62%	34.90%	**97.16%**
		SS	57.75	4.40%	**89.75%**
	5s	STAI-Y	60.52%	37.90%	**79.66%**
		SS	61.65%	16.38%	**88.81%**
SSP	1s	STAI-Y	-	-	-
		SS	-	-	-
	2s	STAI-Y	52.38%	41.67%	61.44%
		SS	59.64%	57.21%	61.03%
	5s	STAI-Y	50.99%	37.60%	60.32%
		SS	60.02%	66.31%	56.42%
Delta-band	1s	STAI-Y	-	-	-
		SS	-	-	-
	2s	STAI-Y	-	-	-
		SS	62.50%	0.00%	**100.00%**
	5s	STAI-Y	54.17%	1.39%	**98.83%**
		SS	62.50%	0.00%	**100.00%**

4.4.2 Random Forest (RF)

Table 4.11 presents the results from classification using RF with PSD features. The best performing combination is raw data with 5s epoch and SS labels, reaching 77.97% accuracy, 64.22% sensitivity and 86.21% specificity. The accuracy across the epoch lengths in raw data vary considerably. No major differences in accuracy across epochs are detected for SSP or Delta-band data. The sensitivity of the Delta-band data is continuously low while the specificity is high, across labels and epochs.

Table 4.11: Accuracy, sensitivity and specificity for PSD features classified by RF. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	60.35%	26.24%	**89.22%**
		SS	**71.06%**	65.55%	**74.36%**
	2s	STAI-Y	62.19%	30.63%	**88.90%**
		SS	66.83%	65.55%	67.71%
	5s	STAI-Y	63.70%	31.12%	**91.26%**
		SS	**77.97%**	64.22%	**86.21%**
SSP	1s	STAI-Y	55.06%	28.37%	**77.64%**
		SS	60.16%	45.65%	**68.44%**
	2s	STAI-Y	54.84%	27.21%	**78.21%**
		SS	61.47%	46.90%	69.80%
	5s	STAI-Y	55.51%	38.83%	69.62%
		SS	61.56%	54.03%	65.86%
Delta-band	1s	STAI-Y	54.42%	3.56%	**97.45%**
		SS	62.51%	0.04%	**100.00%**
	2s	STAI-Y	54.50%	1.34%	**99.48%**
		SS	62.53%	0.07%	**100.00%**
	5s	STAI-Y	55.86%	14.33%	**91.00%**
		SS	62.15%	0.94%	**98.87%**

4.4.3 K-Nearest Neighbors (KNN)

The results from classification of PSD features with KNN is presented in Table 4.12. The best result with 71.61% accuracy, 73.82% sensitivity and 70.28% specificity is achieved with raw data using 2s epochs and SS labels. SS labels are performing better regarding accuracy and sensitivity across all combinations than STAI-Y labels. There are some differences between epoch lengths across the data sets, where 5s epochs perform better for Delta-band data with STAI-Y labels and for both label types with raw data. SSP has similar results across epoch length for both label types.

Table 4.12: Accuracy, sensitivity and specificity for PSD features classified by KNN. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	58.96%	33.63%	**80.40%**
		SS	66.83%	64.96%	67.96%
	2s	STAI-Y	58.67%	32.52%	**80.80%**
		SS	66.75%	66.29%	66.02%
	5s	STAI-Y	60.66%	36.83%	**80.83%**
		SS	**71.61%**	**73.82%**	**70.28%**
SSP	1s	STAI-Y	53.08%	39.34%	64.70%
		SS	58.47%	54.06%	60.99%
	2s	STAI-Y	54.25%	37.10%	68.77%
		SS	58.45%	48.66%	64.05%
	5s	STAI-Y	52.40%	44.38%	59.19%
		SS	59.71%	63.98%	57.26%
Delta-band	1s	STAI-Y	50.96%	26.54%	**71.62%**
		SS	56.20%	37.42%	67.47%
	2s	STAI-Y	50.59%	28.37%	69.39%
		SS	55.26%	36.61%	66.44%
	5s	STAI-Y	54.03%	32.82%	**71.97%**
		SS	54.73%	43.69%	61.36%

4.5 EEGNet

This Section will present the results of classification on the three different data sets using EEGNet. The best result from this classifier yields 72.24% accuracy, 84.68% sensitivity and 61.54% specificity, achieved by raw data with 1s epoch and STAI-Y labels. On raw data the STAI-Y labels performed significantly better than SS labels. For SSP and Delta-band data the opposite is true. The Delta-band achieved low sensitivity results while specificity was high.

Table 4.13: Accuracy, sensitivity and specificity for classification with EEGNet. The results yielding a percentage above 70 are marked in bold. The first column defines the data set used, the second column states the epoch duration and the third column describes the label type.

Data set	Epoch duration	Label type	Accuracy	Sensitivity	Specificity
Raw	1s	STAI-Y	**72.14%**	**84.68%**	61.54%
		SS	54.17%	33.33%	66.67%
	2s	STAI-Y	62.50%	63.64%	61.54%
		SS	49.58%	33.33%	59.33%
	5s	STAI-Y	63.14%	**73.50%**	54.37%
		SS	58.47%	57.63%	58.98%
SSP	1s	STAI-Y	57.66%	48.13%	65.73%
		SS	**70.28%**	34.32%	**90.83%**
	2s	STAI-Y	54.45%	35.20%	**70.73%**
		SS	58.69%	**73.32%**	50.34%
	5s	STAI-Y	52.61%	28.97%	**72.62%**
		SS	61.56%	37.71%	**75.18%**
Delta-band	1s	STAI-Y	53.04%	1.64%	**96.55%**
		SS	62.47%	9.77%	**94.04%**
	2s	STAI-Y	54.70%	3.54%	**97.99%**
		SS	61.83%	16.70%	**88.90%**
	5s	STAI-Y	55.01%	9.71%	**93.35%**
		SS	61.79%	10.73%	**92.43%**

5

Discussion

This chapter will present a discussion regarding the methods that have been utilized in this work presented in Chapter 3. As well as a discussion regarding the results of the methods presented in Chapter 4. The discussion will touch on the methods of data collection, labeling the data, filtering, and the information gained from classification. To expand on the conclusions drawn from the discussion, a section regarding further work will be introduced with suggestions to the next steps in this research endeavor.

In summation: the best result with an accuracy of 77.97%, sensitivity of 64.14% and specificity of 86.21% from Random Forest (RF) classification of Power Spectral Density (PSD) features using raw data with 5 second epoch and Stress-Scale (SS) labels, will not be sufficient enough to employ in an online automated stress detection system for clinical use. The most consistent results was achieved by Hjorth features using a RF classifier with raw data and SS labels, regardless of epoch length. Where the results for each epoch length was between 71% and 78% across all the metrics. In general, the raw data set performed better across all feature combinations than both the filtered data sets SSP and Delta-band. This may be somewhat surprising considering the amount of noise present in most of the Session 2 recordings. It is difficult to determine if the classifiers are then simply predicting based on noise patterns rather than patterns originating from stress.

There are several methods to reduce noise in recordings that could have been implemented in this experiment. One method is to record empty room noise to use as a basis for filtering with Signal-Space Projection (SSP). This would have been especially helpful during the second session when the construction work was disturbing the data collection. The impedance threshold of 160kΩ that was set prior to recording could have been lowered to ensure better contact with the electrodes. There are some drifting artifacts present, and taking into account the movements made by the participants as they conducted the arithmetic test, there is a high likelihood that the electrodes have shifted and moved during the recording period. Other methods of filtering may be beneficial in reducing the noise and securing more information from the data.

Upon analysis of the results of the Delta-band, it is evident that the classifiers are unable to construct patterns to distinguish between stressed and not stressed characteristics when predicting the Delta-band data. Most of the results for sensitivity are either 0.00% or close. The fact that the specificity in that case is so high is due to the symmetric nature of sensitivity vs. specificity. 0.00% sensitivity and 100% specificity means that the classifier predicted all the samples in the test set as Not stressed. This is a sign that the classifier is unable to detect patterns in the data, and that the Delta-band specifically might not contain any information that can help differentiate between a stressed or not stressed person. Exploration of other frequency bands may lead to better results, as mentioned in Section 3.2, the alpha-band showed differences in the topographic map between stressed and not stressed participants.

The Signal-Space Projection (SSP) data had an average performance compared to the other two data sets. Usually the accuracy was around 50%-60%, where the sensitivity and specificity was neither very low or very high.

Though the Stress-Scale (SS) labels mostly performed better than the STAI-Y labels, there were some notable exceptions. The accuracy of raw data using 1s epochs classified with EEGNet and STAI-Y labels outperformed the SS labels by 18 percentage points, 72% vs. 54%. With a sensitivity of 84.68% vs 33.33% from SS labels. The label sets were deemed to be quite similar from the t-test performed in the data exploration in Section 3.2.3. Which makes it difficult to conclude that one method is definitely better than the other. However, based on the majority of better performances from using SS, it appears that a self perceived rating of stress does have significance. It should be noted that some of the participants struggled with the language used in the STAI-Y form, in understanding the questions, and that may have impacted some of the answers. It should be mentioned that either method does rely on the subjective experience of stress, and thus it is difficult to retain a label set were the labels are guaranteed to be the truth. This is one of the main difficulties in this thesis, as it affects the performance of the classifiers and validity of the results.

Epoch duration is an area of research in the EEG community. The results obtained in this thesis are not conclusive in its exploration of epoch lengths. In some cases a decreased epoch duration led to increased accuracies, while the opposite was true for other cases. This occurred both in the classification of the same features and across data sets. In most cases the performance of different epoch lengths seems arbitrary. Further statistical analysis of the epoch duration is needed to conclude, though in this case it seems that the performance of the classifiers is somewhat independent on epoch duration.

To determine which classifier is better for this task we need to look at the results spanning all the features. Random Forest (RF) appears to be the classifier with the most results above 70% without reaching either 0.00% or 100% for sensitivity or specificity. The randomness introduced in RF may be the reason this classifier performs so well. Randomness will contribute to capturing more complex relationships within the data, and EEG data is

often both complex and non-linear. Though I do not have all the results from Support Vector Machine (SVM) classification making the analysis incomplete, the main patterns in the results is that it is prone to overfitting and only predicting the majority class. The latter scenario happens mostly to Delta-band data regardless of feature type. K-Nearest Neighbor (KNN) performs quite consistent when comparing the results from the different feature types, although not particularly high accuracy results.

Across the feature types Hjorth and Time Series features with RF provides the most consistent, good results, while Power Spectral Density (PSD) features produce the highest accuracy. Hjorth features measures the complexity of a signal based on the standard deviation, and it specifically developed for use on EEG signals. Time Series features calculates the variance of the signal, the peak-to-peak amplitude and the Root Mean Square (RMS), meaning that both Hjorth and Time Series are based on similar principles, especially regarding variance. Therefore, the variance may be a contributing factor to the consistency in results that the two feature types produce. Entropy is based on the principle of regularity, by wanting to calculate the amount of information and chaos in a signal. Due to the fact that many recordings were noisy, it can appear that the entropy is measuring the regularity of the noise instead of the signal itself, as the results for entropy features are somewhat lessened compared to the other features. Even though PSD provided the best result, it has not performed consistently. Consistent results are crucial when introducing new technology to clinical settings, so PSD on its own might not yield satisfactory outcomes, based on these results. This thesis has not combined the features and performed feature importance analysis. This is a natural next step to see if some of the features combined can produce a better result than they achieve on their own.

Using only 8 electrodes may have impacted the results as well as the reasons already mentioned. Stress has not been definitively linked to one specific part of the brain, so it will be beneficial to increase the amount of electrodes to get a better representation of where stress is located in the brain. Section 3.2 showed specifically that channel Fp2 might contribute to stress detection. The difference in PSD from a stressed to a not stressed participant was quite distinguished in channel Fp2. Increasing the number of channels will increase the amount of data, however, it can also increase the amount of relevant information regarding stress. In addition, a more extensive exploration into which part of the brain contributes to stress will be beneficial. Note that EEG does not show a one-to-one correlation on location of the signals in the brain, but it can give indications.

Exploring different positions for the participants as they are being recorded may also improve on the signal quality. In this study the participants were sitting in a chair, and during the second run they had to focus on a screen and move their arm to answer the arithmetic questions on the keyboard. As drifts have been present in the signal, along with noise that can neither be ascribed to construction work or blinking, a possible explanation can simply be movement in the participant. Sometimes muscle contractions and tense muscles can result in noise in an EEG. Some of the participants moved a lot during run 2 with the arithmetic test, either the arm or the head swiveling back and forth, possibly explaining parts of the noise that did not get filtered.

6

Conclusion

The main goal of this thesis was to explore different methods of classifying stress using Electroencephalography (EEG) signals to develop an automated stress detection system. A wide range of classifiers, features, data sets, label types and epoch lengths was used, to gain as much experience and information as possible. The data set was collected by myself and four others, during December and January, using other students during their exam period as participants in the study. Three machine learning classifiers and one neural network was tested, Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbor (KNN), and EEGNet respectively. Four types of features were selected, including Time Series, Entropy, Hjorth and Power Spectral Density (PSD). The data was transformed into three data sets where one was the raw, unfiltered data, the second was filtered using bandpass and notch filters before filtering with Signal-Space Projection (SSP), and the last data set was the Delta frequency band of the original raw data including a bandpass and notch filter for initial filtering. Two sets of labels were collected during the experiment, one based on the State-Trait Anxiety Inventory for Adults - Y (STAI-Y) questionnaire, and the other based on the Stress-Scale (SS). Lastly, three epoch duration were explored, namely 1 second, 2 second and 5 second epochs.

The thesis concluded with over 200 unique results, contributing to valuable insights into what classifier works best with which feature, what label type yielded the best result and how epoch duration affected the accuracy of a classifier. The metrics of the classifiers were measured in accuracy, sensitivity and specificity. The highest accuracy was achieved by PSD features with raw data using 5s epochs and SS labels, reaching 77.97% accuracy, 64.22% sensitivity, and 86.21% specificity. The results with the second highest accuracy had an even better sensitivity, where the results were 73.94% accuracy, 77.78% sensitivity and 71.64% specificity, resulting from from Hjorth features using RF on raw data with 5s epoch and SS labels.

6.1 Future work

This thesis is merely a starting point and a guide to how different methods affect the results of classifiers. As such there are many ways to improve and further develop the ideas and results from this thesis. Exploring the importance of features and the combinations of different features are a natural place to start. Increasing the data set by collecting more data, and increasing the number of channels is also a good way of increasing the size of the data set, and perhaps gain more information regarding how stress manifests in EEG recordings.

To continue the work in examining the significance of labels in this subjective line of research, it is perhaps necessary to do a more in depth analysis of the STAI-Y and SS labels. One way to further examine the relationship between the two label sets is to implement an unsupervised learning algorithm and see how it labels the data compared to the two methods used here. This will again not be an objective measure of the ground truth labels, yet it can bring some insights into whether the data is actually clustered with a difference in stress levels. A different approach might be to collect more data and use experts in psychology to help determine if the participants are stressed or not. This may introduce a layer of objectivity when it comes to the labels.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org/. software available from tensorflow.org.

Al-shargie, F., Tang, T.B., Badruddin, N., Kiguchi, M., 2015. Simultaneous measurement of eeg-fnirs in classifying and localizing brain activation to mental stress, in: 2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), pp. 282–286. doi:10.1109/ICSIPA.2015.7412205.

AMBOSS, . Cardiac auscultation. URL: https://www.amboss.com/us/knowledge/Cardiovascular_examination. (accessed 11.01.2023).

Andreassen, I.M., 2023. Classification of phonocardiogram heart sounds using support vector machines and random forest.

AudioCapture, . Audiocapture. URL: https://github.com/labstreaminglayer/App-AudioCapture/releases. (accessed 15.01.2023).

Breiman, L., 1996. Bagging predictors. Machine learning 24, 123–140.

Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

Cohen, S., Kamarck, T., Mermelstein, R., 1983. A global measure of perceived stress. Journal of health and social behavior , 385–396.

Crosswell, A.D., Lockwood, K.G., 2020. Best practices for stress measurement: How to measure psychological stress in health research. Health psychology open 7.

Dickerson, S.S., Kemeny, M.E., 2004. Acute stressors and cortisol responses: a theoretical integration and synthesis of laboratory research. Psychological bulletin 130, 355.

Eko, . Eko duo ecg + digital stethoscope. URL: https://shop.ekohealth.com/products/duo-ecg-digital-stethoscope?variant=39350415655008. (accessed 14.12.2022).

Engstrøm, M., Jansen, J.K.S., 2022. elektroencefalografi. URL: https://sml.snl.no/elektroencefalografi. (accessed 08.06.2023).

Fix, E., Hodges Jr, J.L., 1952. Discriminatory analysis-nonparametric discrimination: Small sample performance. Technical Report. California Univ Berkeley.

Ghosh, R., Deb, N., Sengupta, K., Phukan, A., Choudhury, N., Kashyap, S., Phadikar, S., Saha, R., Das, P., Sinha, N., et al., 2022. Sam 40: Dataset of 40 subject eeg recordings to monitor the induced-stress while performing stroop color-word test, arithmetic task, and mirror image recognition task. Data in Brief 40, 107772.

Gramfort, A., Luessi, M., Larson, E., Engemann, D.A., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., Hämäläinen, M.S., 2013. MEG and EEG data analysis with MNE-Python. Frontiers in Neuroscience 7, 1–13. doi:10.3389/fnins.2013.00267.

Guyon, I., Elisseeff, A., 2006. An introduction to feature extraction. Feature extraction: foundations and applications , 1–25.

Hjorth, B., 1970. Eeg analysis based on time domain properties. Electroencephalography and Clinical Neurophysiology 29, 306–310. URL: https://www.sciencedirect.com/science/article/pii/0013469470901434, doi:https://doi.org/10.1016/0013-4694(70)90143-4.

Hou, X., Liu, Y., Sourina, O., Tan, Y.R.E., Wang, L., Mueller-Wittig, W., 2015. Eeg based stress monitoring, in: 2015 IEEE International Conference on Systems, Man, and Cybernetics, pp. 3110–3115. doi:10.1109/SMC.2015.540.

Jiang, X., Bian, G.B., Tian, Z., 2019. Removal of artifacts from eeg signals: a review. Sensors 19, 987.

Jordan, M.I., Mitchell, T.M., 2015. Machine learning: Trends, perspectives, and prospects. Science 349, 255–260.

Joshua S. Richman, J.R.M., 2000. Physiological time-series analysis using approximate entropy and sample entropy. American Physiological Society 278, H2039–H2049. doi:https://doi.org/10.1152/ajpheart.2000.278.6.H2039.

Kaada, B., 2018. aksjonspotensial. URL: https://sml.snl.no/aksjonspotensial. (accessed 08.06.2023).

Katmah, R., Al-Shargie, F., Tariq, U., Babiloni, F., Al-Mughairbi, F., Al-Nashash, H., 2021. A review on mental stress assessment methods using eeg signals. Sensors 21, 5043.

Khosrowabadi, R., Quek, C., Ang, K.K., Tung, S.W., Heijnen, M., 2011. A brain-computer interface for classifying eeg correlates of chronic mental stress, in: The 2011 international joint conference on neural networks, IEEE. pp. 757–762.

Kothe, C., . Lab streaming layer. URL: `https://github.com/sccn/labstreaminglayer`. (accessed 06.01.2023).

Kramer, O., Kramer, O., 2013. K-nearest neighbors. Dimensionality reduction with unsupervised nearest neighbors , 13–23.

Larson, E., Gramfort, A., Engemann, D.A., Leppakangas, J., Brodbeck, C., Jas, M., Brooks, T., Sassenhagen, J., Luessi, M., McCloy, D., King, J.R., Goj, R., Favelier, G., Höchenberger, R., Brunner, C., van Vliet, M., Wronkiewicz, M., Holdgraf, C., Massich, J., Bekhti, Y., Rockhill, A., Appelhoff, S., Leggitt, A., Dykstra, A., Luke, R., Trachel, R., De Santis, L., Panda, A., Magnuski, M., Scheltienne, M., Westner, B., Billinger, M., Wakeman, D.G., Strohmeier, D., Bharadwaj, H., Linzen, T., Barachant, A., Ruzich, E., Bailey, C.J., Li, A., Moutard, C., Bloy, L., Raimondo, F., Nurminen, J., Montoya, J., Woodman, M., Lee, I., Schulz, M., Foti, N., Nangini, C., García Alanis, J.C., Hauk, O., Maddox, R., LaPlante, R., Drew, A., Dinh, C., Dumas, G., Hartmann, T., Ort, E., Benerradi, J., Pasler, P., Repplinger, S., Rudiuk, A., Radanovic, A., Buran, B., Massias, M., Hämäläinen, M., Sripad, P., Chirkov, V., Mullins, C., Raimundo, F., Alday, P., Pari, R., Kornblith, S., Halchenko, Y., Luo, Y.H., Kasper, J., Doelling, K., Jensen, M., Gahlot, T., Nunes, A., Gütlin, D., kjs, Weinstein, A., Lamus, C., Galván, C.M., Moënne-Loccoz, C., Heinila, E., Hanna, J., Houck, J., Klein, N., Rantala, A., Maess, B., O'Reilly, C., Peterson, E., Kolkhorst, H., Banville, H., Maksymenko, K., Clarke, M., Anelli, M., Kaneda, M., Bannier, P.A., Choudhary, S., Huberty, S., Kern, S., Forster, C., Kim, C., Klotzsche, F., Wong, F.T., Kojcic, I., Zhang, J., Nielsen, J.D., Lankinen, K., Tabavi, K., Thibault, L., Gayraud, N., Ward, N., dependabot[bot], Quinn, A., Gauthier, A., Pinsard, B., Welke, D., Stephen, E., Hornberger, E., Hathaway, E., Kalenkovich, E., Mamashli, F., Marinato, G., Anevar, H., Sosulski, J., Stout, J., Eisenman, L., Esch, L., Dovgialo, M., Barascud, N., Legrand, N., Falach, R., Deslauriers-Gauthier, S., Cotroneo, S., Matindi, S., Bierer, S., Férat, V., Peterson, V., Tonin, A., Kovrig, A., Pascarella, A., Karekal, A., de la Torre, C., Gohil, C., Zhao, C., Krzemiński, D., Welke, D., Makowski, D., Mikulan, E., Schiratti, J.B., Evans, J., Drew, J., Teves, J., Mathewson, K., Gwilliams, L., Varghese, L., Gemein, L., Hecker, L., Lx37, van Es, M., Boggess, M., Eberlein, M., Sherif, M., Gerster, M., Kozhemiako, N., Srinivasan, N., Wilming, N., Chapochnikov, N., Kozynets, O., Ablin, P., Bertrand, Q., Shoorangiz, R., Hübner, R., Sommariva, S., Er, S., Khan, S., Herbst, S., Datta, S., Jochmann, T., Merk, T., Flak, T., Dupré la Tour, T., Stenner, T., NessAiver, T., akshay0724, sviter, Hindle, A., Koutsou, A., Fecker, A., Wagner, A., Ciok, A., Gramfort, A., Pradhan, A., Padee, A., Dubarry, A.S., Waniek, A.N., Singhal, A., Rokem, A., Hurst, A., Beasley, B., Nicenboim, B., de la Torre, C., Clauss, C., Mista, C., Li, C.H., Braboszcz, C., Schad, D.C., Hasegan, D., Sleiter, D.E., Haslacher, D., Sabbagh, D., Kostas, D., Petkova, D., Issagaliyeva, D., Altukhov, D., Wetzel, D., Eich, E., DuPre, E., Lau, E., Olivetti, E., Varano, E., Altamiranda, E., Brayet, E., de Montalivet, E., Goldstein, E., Zamberlan, F., Weber, F.D., Tan, G., Brookshire, G., Maymandi, H., Sonntag, H., Ye, H., Elmas, H.O., Machairas, I., Kaczmarzyk, J., Zerfowski, J., van den Bosch, J.J.F., Behnke, J., Van Der Donckt, J., van der

Meer, J., Niediek, J., Veillette, J., Koen, J., Bear, J.J., Zhu, J.D., Dammers, J., Galán, J.G.N., Welzel, J., Slama, K., Leinweber, K., Grabot, L., Andersen, L.M., Barbosa, L.S., Hamilton, L., Alfine, L., Hejtmánek, L., Kitzbichler, M., Kumar, M., Kadwani, M., Sutela, M., Koculak, M., Henney, M.A., van Harmelen, M., MartinBaBer, Courtemanche, M., Tucker, M., Visconti di Oleggio Castello, M., Dold, M., Toivonen, M., Shader, M., Cespedes, M., Krause, M., Rybář, M., He, M., Daneshzand, M., Gensollen, N., Proulx, N., Chalas, N., Shubi, O., Sundaram, P., Roujansky, P., Silva, P., Molfese, P.J., Li, Q., Nadkarni, R., Gatti, R., Apariciogarcia, R., Nasri, R., Koehler, R., Stargardsky, R., Oostenveld, R., Seymour, R., Schirrmeister, R.T., Law, R., Pai, S., Perry, S., Ruuskanen, S., Mathot, S., Major, S., Treguer, S., Castaño, S., Deng, S., Antopolskiy, S., Wong, S., Wong, S., Poil, S.S., Foslien, S., Singh, S., Chambon, S., Bethard, S., Gutstein, S.M., Meyer, S.M., Wang, T., Papadopoulo, T., Donoghue, T., Radman, T., Gates, T., Ma, T., Clausner, T., Anijärv, T.E., Xia, X., Zhang, Z., buildqa, luzpaz, 2023. Mne-python. URL: https://doi.org/10.5281/zenodo.7671973, doi:10.5281/zenodo.7671973.

Lawhern, V.J., Solon, A.J., Waytowich, N.R., Gordon, S.M., Hung, C.P., Lance, B.J., 2018. Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. Journal of neural engineering 15, 056013.

Marthinsen, A.J.Y., 2022. Detection of mental stress from eeg data using artificial intelligence. doi:10.13140/RG.2.2.27754.39360.

McEwen, B.S., 2008. Central effects of stress hormones in health and disease: Understanding the protective and damaging effects of stress and stress mediators. European journal of pharmacology 583, 174–185.

Mentalab, . Meet mentalab explore. URL: https://mentalab.com/mobile-eeg/. (accessed 14.12.2022).

MNE, . Background on projectors and projections. URL: https://mne.tools/stable/auto_tutorials/preprocessing/45_projectors_background.html#tut-projectors-background. accessed 26.06.2023.

MNE-features, . Mne-features. URL: https://mne.tools/mne-features/#.

Noble, W.S., 2006. What is a support vector machine? Nature Biotechnology 24, 1565–1567. doi:https://doi.org/10.1038/nbt1206-1565.

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

Peirce, J., Gray, J.R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman, E., Lindeløv, J.K., 2019. Psychopy2: Experiments in behavior made easy. Behavior research methods 51, 195–203.

Pincus, S., 1995. Approximate entropy (apen) as a complexity measure. Chaos: An Interdisciplinary Journal of Nonlinear Science 5, 110–117.

Roberts, S.J., Penny, W., Rezek, I., 1999. Temporal and spatial complexity measures for electroencephalogram based brain-computer interfacing. Medical & biological engineering & computing 37, 93–98.

SciPy, . Scipy. URL: https://docs.scipy.org/doc/scipy/index.html. (accessed 30.05.2023).

Selye, H., 1973. The evolution of the stress concept: The originator of the concept traces its development from the discovery in 1936 of the alarm reaction to modern therapeutic applications of syntoxic and catatoxic hormones. American scientist 61, 692–699.

Sharbrough, F., Chatrian, G., Lesser, R., Luders, H., Nuwer, M., Picton, T., 1991. American electroencephalographic society guidelines for standard electrode position nomenclature. Clinical Neurophysiology 8, 200–202.

Spielberger, C.D., Gonzalez-Reigosa, F., Martinez-Urrutia, A., Natalicio, L.F., Natalicio, D.S., 1971. The state-trait anxiety inventory. Revista Interamericana de Psicologia/Interamerican journal of psychology 5.

Svartdal, F., Malt, U., 2022. stress. URL: https://sml.snl.no/stress. (accessed 27.06.2023).

Tveit, J., Aurlien, H., Plis, S., Calhoun, V.D., Tatum, W.O., Schomer, D.L., Arntsen, V., Cox, F., Fahoum, F., Gallentine, W.B., et al., 2023. Automated interpretation of clinical electroencephalograms using artificial intelligence. JAMA neurology .

Uusitalo, M.A., Ilmoniemi, R.J., 1997. Signal-space projection method for separating meg or eeg into components. Medical and biological engineering and computing 35, 135–140.

Zhang, A., Yang, B., Huang, L., 2008. Feature extraction of eeg signals using power spectral entropy, in: 2008 International Conference on BioMedical Engineering and Informatics, pp. 435–439. doi:10.1109/BMEI.2008.254.

Appendix

A Code: prepare_data.py

```
from utils.data_processing import extract_eeg_data, segment_data,
    create_train_test_split, dict_to_arr_labels, dict_to_arr
from utils.labels import get_stai_labels, get_pss_labels
from utils.valid_recordings import get_valid_recordings
from features import time_series_features, entropy_features,
    hjorth_features, psd_features
import numpy as np

def prepare_data(data_type, label_type, epoch_duration=5, feature_type='
    None'):
    '''
    Prepare data of different types for classification.

    Parameters
    ----------
    data_type : str
        String containing the data type to be classified. Either 'raw', '
            filtered', or 'decomposed'
    label_type : str
        Input can be either 'STAI' or 'PSS', to indicate either STAI-
            labels or PSS-labels.
    epoch_duration : int
        The chosen epoch duration for segmenting the data, in seconds.
            Default is 5 seconds.
    feature_type : str
        The chosen feature type, default is None, available feature_types
            is 'time_series', 'entropy', 'hjorth', or 'psd'


    Returns
    -------
    train_data : nd.array
        An array containing the training data of the given data type
    train_labels : nd.array
        An array containing the training labels of the given data type
    test_data : nd.array
        An array containing the test data of the given data type
    test_labels : nd.array
        An array containing the test labels of the given data type

    '''
    valid_recordings = get_valid_recordings(data_type=data_type)
    data = extract_eeg_data(valid_recordings=valid_recordings, data_type=
        data_type)

    if label_type == 'STAI':
        labels = get_stai_labels(valid_recordings, cutoff=40)
    elif label_type == 'PSS':
```

```
        labels = get_pss_labels(valid_recordings=valid_recordings,
            threshold=4)
    else:
        print('Suggested label_type is not supported.')

    segmented_data, segmented_labels = segment_data(data, labels,
        epoch_duration)

    train_data, train_labels, test_data, test_labels =
        create_train_test_split(segmented_data, segmented_labels,
        epoch_duration=epoch_duration)

    print(f'----------------- CHOSEN FEATURE TYPE: {feature_type
        }-----------------')
    print(f"----- Shape of train data set: {train_data.shape} ---------
        Shape of train labels set: {train_labels.shape} -----")
    #print(f"Shape of train labels set: {train_labels.shape}")

    print(f"----- Shape of test data set:  {test_data.shape}  ---------
        Shape of test labels set:  {test_labels.shape}  -----")
    #print(f"Shape of test labels set: {test_labels.shape}")

    if feature_type == 'time_series':
        train_data_features = time_series_features(train_data)
        test_data_features  = time_series_features(test_data)
        print(f"Shape of train data features set: {train_data_features.
            shape}")
        print(f"Shape of test data features set: {test_data_features.shape
            }")
        return train_data_features, test_data_features, train_labels,
            test_labels

    elif feature_type == 'entropy':
        train_data_features = entropy_features(train_data)
        test_data_features  = entropy_features(test_data)
        print(f"Shape of train data features set: {train_data_features.
            shape}")
        print(f"Shape of test data features set: {test_data_features.shape
            }")
        return train_data_features, test_data_features, train_labels,
            test_labels

    elif feature_type == 'hjorth':
        train_data_features = hjorth_features(train_data)
        test_data_features  = hjorth_features(test_data)
        print(f"Shape of train data features set: {train_data_features.
            shape}")
        print(f"Shape of test data features set: {test_data_features.shape
            }")
        return train_data_features, test_data_features, train_labels,
            test_labels

    elif feature_type == 'psd':
        train_data_features = psd_features(train_data)
        test_data_features  = psd_features(test_data)
        print(f"Shape of train data features set: {train_data_features.
            shape}")
```

```
        print(f"Shape of test data features set: {test_data_features.shape}")
        return train_data_features, test_data_features, train_labels, test_labels

    elif feature_type == 'None':
        return train_data, test_data, train_labels, test_labels




def prepare_data_for_clustering(data_type, epoch_duration):
    '''
    Prepare data for cluster algorithms with time-series features.
    '''

    valid_recordings = get_valid_recordings(data_type=data_type)
    data             = extract_eeg_data(valid_recordings=valid_recordings, data_type=data_type)


    stai_labels = get_stai_labels(valid_recordings, cutoff=40)
    pss_labels  = get_pss_labels(valid_recordings=valid_recordings, threshold=4)

    stai_labels_arr = dict_to_arr_labels(stai_labels)
    pss_labels_arr  = dict_to_arr_labels(pss_labels)

    segmented_data, _ = segment_data(data, stai_labels, epoch_duration) # arbitrary label type for segmenting

    data_arr        = dict_to_arr(data_dict=segmented_data, epoch_duration=epoch_duration)

    data_features   = time_series_features(data=data_arr)
    return data_features, stai_labels_arr, pss_labels_arr
```

B Code: features.py

```
import numpy as np
import mne_features.univariate as mnf
import utils.variables as var


def hjorth_features(data):
    '''
    Computes the features Hjorth mobility (spectral) and Hjorth complexity
        (spectral) using the package mne_features.

    Parameters
    ----------
    data : dict
        data (list of dicts): A list of dictionaries, where each
            dictionary contains EEG data for multiple trials. The keys in
            each dictionary represent trial IDs, and the values are numpy
            arrays of shape (n_channels, n_samples).

    Returns
    -------
    features : list of ndarrays
        list of ndarrays with the computed features.
    '''

    sfreq = var.SFREQ
    features_per_channel = 2
    n_recordings, n_channels, n_samples = data.shape

    features = np.empty([n_recordings, n_channels*features_per_channel])

    for i in range(n_recordings):
        samples          = data[i]
        mobility_spect   = mnf.compute_hjorth_mobility_spect(sfreq=sfreq,
            data=samples)
        complexity_spect = mnf.compute_hjorth_complexity_spect(sfreq=sfreq
            , data=samples)
        features[i]      = np.concatenate([mobility_spect,
            complexity_spect])

    features = features.reshape([n_recordings, n_channels*
        features_per_channel])

    return features

def entropy_features(data):
    '''
    Computes the features Approximate Entropy, Sample Entropy, Spectral
        Entropy and SVD entropy using the package mne_features.

    Parameters
    ----------
    data : dict
        A list of dictionaries, where each dictionary contains EEG data
            for multiple trials. The keys in each dictionary represent
            trial IDs, and the values are numpy arrays of shape (
```

```
            n_channels, n_samples).

    Returns
    -------
    features : list of ndarrays
        list of ndarrays of the computed features.
    '''

    sfreq = var.SFREQ
    features_per_channel = 4
    n_recordings, n_channels, n_samples = data.shape

    features = np.empty([n_recordings, n_channels*features_per_channel])

    for i in range(n_recordings):
        samples          = data[i]
        app_entropy      = mnf.compute_app_entropy(data=samples)
        samp_entropy     = mnf.compute_samp_entropy(data=samples)
        spect_entropy    = mnf.compute_spect_entropy(sfreq=sfreq, data=
            samples)
        svd_entropy      = mnf.compute_svd_entropy(data=samples)
        features[i]      = np.concatenate([app_entropy, samp_entropy,
            spect_entropy, svd_entropy])

    features = features.reshape([n_recordings, n_channels*
        features_per_channel])

    return features

def time_series_features(data):
    '''
    Compute time-series features from data.

    Parameters
    ----------
    data : ndarray, shape(n_recordings, n_channels, n_samples)
        An ndarray containing the data.

    Returns
    -------
    features : ndarray, shape(n_recordings, n_channels*
        features_per_channel)
        An ndarray of the computed features
    '''
    sfreq = var.SFREQ
    features_per_channel = 3
    n_recordings, n_channels, n_samples = data.shape

    features = np.empty([n_recordings, n_channels*features_per_channel])

    for i in range(n_recordings):
        samples      = data[i]
        ptp_amp      = mnf.compute_ptp_amp(data=samples)
        variance     = mnf.compute_variance(data=samples)
        rms          = mnf.compute_rms(data=samples)
        features[i] = np.concatenate([ptp_amp, variance, rms])

```

```
97      features = features.reshape([n_recordings, n_channels*
            features_per_channel])
98
99      return features
100
101 def psd_features(data):
102     sfreq = var.SFREQ
103     n_frequencies = 5 #default number of (freqs-1) for mnf.
            compute_pow_freq_bands
104     n_recordings, n_channels, n_samples = data.shape
105
106     features = np.empty([n_recordings, n_channels*n_frequencies])
107
108     for i in range(n_recordings):
109         samples = data[i]
110         psd     = mnf.compute_pow_freq_bands(sfreq=sfreq, data=samples)
111         features[i] = np.concatenate([psd])
112
113     features = features.reshape([n_recordings, n_channels*n_frequencies])
114
115     return features
116
117 def all_features(data):
118     '''
119     Compute hjorth, entropy and time-series features in one
120
121     '''
122
123     sfreq = var.SFREQ
124     features_per_channel = 10
125     n_recordings, n_channels, n_samples = data.shape
126
127     features = np.empty([n_recordings, n_channels*features_per_channel])
128
129     for i in range(n_recordings):
130         samples              = data[i]
131         mobility_spect       = mnf.compute_hjorth_mobility_spect(sfreq=
                sfreq, data=samples)
132         complexity_spect     = mnf.compute_hjorth_complexity_spect(sfreq=
                sfreq, data=samples)
133         app_entropy          = mnf.compute_app_entropy(data=samples)
134         samp_entropy         = mnf.compute_samp_entropy(data=samples)
135         spect_entropy        = mnf.compute_spect_entropy(sfreq=sfreq, data=
                samples)
136         svd_entropy          = mnf.compute_svd_entropy(data=samples)
137         ptp_amp              = mnf.compute_ptp_amp(data=samples)
138         variance             = mnf.compute_variance(data=samples)
139         rms                  = mnf.compute_rms(data=samples)
140         mean                 = mnf.compute_mean(data=samples)
141         features[i]          = np.concatenate([mobility_spect,
                complexity_spect, app_entropy, samp_entropy, spect_entropy,
                svd_entropy, ptp_amp, variance, rms, mean])
142
143     features = features.reshape([n_recordings, n_channels*
            features_per_channel])
```

C Code: classifiers.py

```
from matplotlib import pyplot as plt
import numpy as np
#from sklearn.metrics import plot_confusion_matrix

from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from metrics import compute_metrics
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import RobustScaler, StandardScaler
from sklearn.model_selection import train_test_split

from EEGNet.EEGModels import EEGNet
from tensorflow.keras import utils as np_utils
from tensorflow.keras.callbacks import ModelCheckpoint
#from keras.utils import np_utils
#from keras.callbacks import ModelCheckpoint
from pyriemann.utils.viz import plot_confusion_matrix

import utils.variables as var
import mne


def svm(train_data, test_data, train_labels, test_labels):
    '''
    Parameters
    ----------
    train_data : dict
        Path to the file to be read.
    test_data : dict
        Test data
    train_labels : dict
        Labels of the train data
    test_labels : dict
        Labels of the test data

    Returns
    -------
    metrics : confusion matrix
        The confusion matrix with the results
    '''

    param_grid = {
        'C': [0.01, 0.1, 1, 10, 100, 1000, 10000],
        'kernel': ['poly', 'sigmoid', 'linear', 'rbf']
    }

    scaler = StandardScaler()
    train_data = scaler.fit_transform(train_data)
    test_data = scaler.transform(test_data)

    #weights = {0:67, 1:33}
    #scaler = RobustScaler()
    #train_data = scaler.fit_transform(train_data)
```

```
    #test_data = scaler.transform(test_data)

    svm_clf = GridSearchCV(SVC(), param_grid=param_grid, refit=True,
        n_jobs=-1, cv=10)
    svm_clf.fit(train_data, train_labels)

    y_pred = svm_clf.predict(test_data)
    y_true = test_labels

    cv_results = svm_clf.cv_results_
    accuracy = cv_results['mean_test_score']
    #print('---------------------- RESULTS FROM GRIDSEARCH
        ---------------------- \n', cv_results)
    print('---------------------- BEST PARAMETERS FROM GRIDSEARCH
        ---------------------- \n', svm_clf.best_params_)
    print(' OVERALL ACCURACY:', np.round(np.sum(accuracy)/len(accuracy)
        *100,2))

    plt.figure(1)
    plt.plot(accuracy)
    plt.xlabel('Fold')
    plt.ylabel('Mean accuracy of test score')
    plt.show()

    metrics = compute_metrics(y_true, y_pred)

    return metrics


def rf(train_data, test_data, train_labels, test_labels):
    '''
    Input: data of shape (n_samples, n_features), and labels of shape (
        n_samples). Performs random forest classification
    '''

    param_grid = {
        'n_estimators' : [50, 75, 100, 125, 150, 175, 200, 225, 250],
        'max_features' : ['auto', 'sqrt', 'log2'],
        'max_depth' : [3, 5, 7, 9]
    }

    #weights = {0:67, 1:33}

    rf_clf = GridSearchCV(RandomForestClassifier(), param_grid=param_grid,
         refit=True, n_jobs=-1, cv=10)
    rf_clf.fit(train_data, train_labels)

    y_pred = rf_clf.predict(test_data)
    y_true = test_labels

    cv_results = rf_clf.cv_results_
    accuracy = cv_results['mean_test_score']
    #print('---------------------- RESULTS FROM GRIDSEARCH
        ---------------------- \n', cv_results)
    print('---------------------- BEST PARAMETERS FROM GRIDSEARCH
        ---------------------- \n', rf_clf.best_params_)
    print(' OVERALL ACCURACY:', np.round(np.sum(accuracy)/len(accuracy)
```

```
        *100,2))

    plt.figure(1)
    plt.plot(accuracy)
    plt.xlabel('Fold')
    plt.ylabel('Mean accuracy of test score')
    plt.show()

    metrics = compute_metrics(y_true, y_pred)


    return metrics

def knn(train_data, test_data, train_labels, test_labels):
    '''
    Explanation

    Parameters
    ----------
    train_data : ndarray
        An array containing the training data, shape(n_recordings,
            n_channels*n_features)
    test_data : ndarray
        An array containing the test data, shape(n_recordings, n_channels*
            n_features)
    train_labels : ndarray
        An array containing the labels of the training set, shape(
            n_recordings, )

    Returns
    -------
    metrics : something

    '''
    param_grid = {
        'n_neighbors': [1, 5, 9, 15, 19, 25, 29, 35, 39],
        'leaf_size': [5, 10, 20, 30, 40, 50],
        'weights' : ['uniform', 'distance']
    }

    scaler = StandardScaler()
    train_data = scaler.fit_transform(train_data)
    test_data = scaler.transform(test_data)

    knn_clf = GridSearchCV(KNeighborsClassifier(), param_grid, refit=True,
         n_jobs=-1, cv = 10)
    knn_clf.fit(train_data, train_labels)

    y_pred = knn_clf.predict(test_data)
    y_true = test_labels

    cv_results = knn_clf.cv_results_
    accuracy = cv_results['mean_test_score']
    #print('---------------------- RESULTS FROM GRIDSEARCH
        ---------------------- \n', cv_results)
    print('---------------------- BEST PARAMETERS FROM GRIDSEARCH
        ---------------------- \n', knn_clf.best_params_)
```

```
155     print(' OVERALL ACCURACY:', np.round(np.sum(accuracy)/len(accuracy)
            *100,2))
156
157     plt.figure(1)
158     plt.plot(accuracy)
159     plt.xlabel('Fold')
160     plt.ylabel('Mean accuracy of test score')
161     plt.show()
162
163     metrics = compute_metrics(y_true, y_pred)
164
165     return metrics
166
167
168 def EEGNet_classifier(train_data, test_data, train_labels, test_labels,
        epoch_duration):
169
170     # Add validation set
171     training_data, validation_data, training_labels, validation_labels =
            train_test_split(train_data, train_labels, test_size=0.25,
            random_state=42, stratify=train_labels)
172
173     # configure the EEGNet-8,2,16 model with kernel length of 32 samples (
            other
174     # model configurations may do better, but this is a good starting
            point)
175     model = EEGNet(nb_classes=2, Chans=var.NUM_CHANNELS, Samples=(
            epoch_duration*var.SFREQ)+1,
176                    dropoutRate = 0.5, kernLength = 125, F1 = 8, D = 2, F2
                        = 16,
177                    dropoutType = 'Dropout')
178
179     # compile the model and set the optimizers
180     model.compile(loss='sparse_categorical_crossentropy', optimizer='adam'
            ,
181                   metrics = ['accuracy'])
182
183     # count number of parameters in the model
184     numParams    = model.count_params()
185
186
187     # set a valid path for your system to record model checkpoints
188     checkpointer = ModelCheckpoint(filepath='/tmp/checkpoint.h5', verbose=
            1,
189                                    save_best_only=True)
190
191     #
            ################################################################################
192     # if the classification task was imbalanced (significantly more trials
             in one
193     # class versus the others) you can assign a weight to each class
            during
194     # optimization to balance it out. This data is approximately balanced
            so we
195     # don't need to do this, but is shown here for illustration/
            completeness.
```

```
196 #
        ################################################################################

197
198 # the syntax is {class_1:weight_1, class_2:weight_2,...}. Here just
        setting
199 # the weights all to be 1
200 class_weights = {0:1, 1:1}
201
202 #
        ################################################################################

203 # fit the model. Due to very small sample sizes this can get
204 # pretty noisy run-to-run, but most runs should be comparable to xDAWN
        +
205 # Riemannian geometry classification (below)
206 #
        ################################################################################

207 fittedModel = model.fit(training_data, training_labels, batch_size =
        64, epochs = 300,
208                         verbose = 2, validation_data=(validation_data,
                                validation_labels),
209                         callbacks=[checkpointer], class_weight =
                                class_weights)
210
211 # load optimal weights
212 model.load_weights('/tmp/checkpoint.h5')
213
214 #
        ################################################################################

215 # can alternatively used the weights provided in the repo. If so it
        should get
216 # you 93% accuracy. Change the WEIGHTS_PATH variable to wherever it is
        on your
217 # system.
218 #
        ################################################################################

219
220 # WEIGHTS_PATH = /path/to/EEGNet-8-2-weights.h5
221 # model.load_weights(WEIGHTS_PATH)
222
223 #
        ################################################################################

224 # make prediction on test set.
225 #
        ################################################################################

226
227 probs       = model.predict(test_data)
228 preds       = probs.argmax(axis = -1)
229 acc         = np.mean(preds == test_labels)
230 print("Classification accuracy: %f " % (acc))
231
```

```
232     #names          = ['Not stressed', 'Stressed']
233     #plt.figure(0)
234     #plot_confusion_matrix(preds, test_labels, names, title = 'EEGNet
            -8,2')
235     compute_metrics(y_true=test_labels, y_pred=preds)
236     return probs
237
238
239 def k_means_clustering(data):
240     '''
241     Perform K-Means clustering on data
242
243     Parameters
244     ----------
245     data : ndarray of shape (n_samples, n_features)
246
247     '''
248     scaler = StandardScaler()
249     data = scaler.fit_transform(data)
250
251     kmeans = KMeans(init='random', n_clusters=2, n_init=10, max_iter=300,
            random_state=42)
252     kmeans.fit(data)
253
254     labels = kmeans.labels_
255
256     return labels
```

D Code: metrics.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
from matplotlib.colors import LinearSegmentedColormap

def compute_metrics(y_true, y_pred):
    '''
    Compute the confusion matrix of the input, accuracy, sensitivity, and specificity.
    Input is y_true and y_predicted.
    Output is display of confusion matrix and an array of accuracy, sensitivity, and specificity.
    '''
    conf_matrix = confusion_matrix(y_true, y_pred)

    true_negative  = conf_matrix[0,0]
    false_negative = conf_matrix[1,0]
    true_positive  = conf_matrix[1,1]
    false_positive = conf_matrix[0,1]

    accuracy = ((true_positive+true_negative)/(true_positive + true_negative + false_positive + false_negative)) *100

    sensitivity = (true_positive/(true_positive + false_negative)) *100
    specificity = (true_negative/(false_positive + true_negative)) *100

    labels = ('Not stressed', 'Stressed')
    colors = ["#ef8114", "#b01b81", "#482776"]
    cmap1 = LinearSegmentedColormap.from_list("mycmap", colors)
    confusion_matrix_display = ConfusionMatrixDisplay.from_predictions(y_true, y_pred, cmap=cmap1, display_labels=labels)

    textstr = f"Accuracy: {round(accuracy,2)}% \nSensitivity: {round(sensitivity,2)}% \nSpecificity: {round(specificity,2)}%"
    props = dict(boxstyle='round', facecolor='white', alpha=0.5)
    xpos = 0.5
    ypos = -0.2
    confusion_matrix_display.ax_.text(xpos, ypos, textstr, transform=confusion_matrix_display.ax_.transAxes, fontsize=16,
            verticalalignment='top', bbox=props, ha='center')

    plt.show
```

E Code: Filtering.ipynb

filtering

July 3, 2023

[1]:
```
import mne
import os
from mne.preprocessing import (create_eog_epochs, create_ecg_epochs,
                               compute_proj_ecg, compute_proj_eog)
import utils.variables as var
import scipy.io as sio
import numpy as np
from utils.valid_recordings import get_valid_recordings
import matplotlib.pyplot as plt
from math import log
%matplotlib qt
```

[2]:
```
class Filtering:
    '''
    A Python class that handles all of the filtering of data, including saving␣
 ↪the filtered data to new .mat-files
    '''
    root         = '/Users/idaandreassen/Desktop/MASTER/stress_detection_EEG/'
    dir_raw      = root + var.DIR_RAW
    dir_ssp      = root + var.DIR_SSP
    dir_psd      = root + var.DIR_PSD
    dir_decomp   = root + var.DIR_DECOMP

    sfreq = var.SFREQ
    ch_type = 'eeg'
    n_channels = var.NUM_CHANNELS

    def __init__(self, sub_nr, ses_nr, run_nr):
        self.sub_nr = sub_nr
        self.ses_nr = ses_nr
        self.run_nr = run_nr

        self.load_data()

        #Create MNE RawArray
        info = mne.create_info(8, sfreq=self.sfreq, ch_types=self.ch_type,␣
 ↪verbose=None)
```

```
        self.raw_arr = mne.io.RawArray(self.data, info)

        mne.rename_channels(self.raw_arr.info, var.MAPPING)

        montage = mne.channels.make_standard_montage('standard_1020')
        self.raw_arr.set_montage(montage)

        self.filtered_arr = self.init_filter()

␣
↪#-----------------------------------------------FUNCTIONS----------------------------------

    def load_data(self):
        dir      = self.dir_raw
        data_key = 'raw_eeg_data'

        # Load one recording
        filename  = f"/sub-{self.sub_nr}_ses-{self.ses_nr}_run-{self.run_nr}.
↪mat"
        f         = dir + filename
        self.data = sio.loadmat(f)[data_key]

    def save_ssp_data(self):
        title = f"sub-{self.sub_nr}_ses-{self.ses_nr}_run-{self.run_nr}"
        self.ssp.apply_proj(True)
        clean_data = self.ssp.to_data_frame(scalings=1)
        clean_data = clean_data.to_numpy()
        clean_data = np.transpose(clean_data)
        clean_dict = {
            "Clean_data" : clean_data[1:, :]  #First columm of dataFrames is␣
↪not data
        }
        sio.savemat(f'{self.dir_ssp}/{title}.mat', clean_dict)

    def save_decomp_data(self):
        title = f"sub-{self.sub_nr}_ses-{self.ses_nr}_run-{self.run_nr}"
        clean_data = self.decomp_data.to_data_frame(scalings=1)
        clean_data = clean_data.to_numpy()
        clean_data = np.transpose(clean_data)
        clean_dict = {
            "Decomp_data" : clean_data[1:, :]  #First column of dataFrames is␣
↪not data
        }
        sio.savemat(f'{self.dir_decomp}/{title}.mat', clean_dict)

    def save_psd(self, data_type):
        title = f"sub-{self.sub_nr}_ses-{self.ses_nr}_run-{self.run_nr}"
```

```
        psd_data = self.psd.get_data()
        psd_dict = {
            "psd_data" : psd_data
        }
        if data_type == 'raw':
            sio.savemat(f'{self.dir_psd}/PSD/{title}.mat', psd_dict)
        elif data_type == 'filtered':
            sio.savemat(f'{self.dir_psd}/PSD_filtered/{title}.mat', psd_dict)
        elif data_type == 'delta':
            sio.savemat(f'{self.dir_psd}/PSD_delta/{title}.mat', psd_dict)
        elif data_type == 'theta':
            sio.savemat(f'{self.dir_psd}/PSD_theta/{title}.mat', psd_dict)
        elif data_type == 'alpha':
            sio.savemat(f'{self.dir_psd}/PSD_alpha/{title}.mat', psd_dict)
        elif data_type == 'beta':
            sio.savemat(f'{self.dir_psd}/PSD_beta/{title}.mat', psd_dict)
        elif data_type == 'gamma':
            sio.savemat(f'{self.dir_psd}/PSD_gamma/{title}.mat', psd_dict)

    def init_filter(self):
        band_pass = self.raw_arr.copy().filter(1, 50)
        #sav_gol = band_pass.copy().savgol_filter(h_freq=35, verbose=False)
        notch = band_pass.copy().notch_filter(freqs=[50,100], trans_bandwidth =␣
↪0.5)
        return notch

    def compute_and_save_filtered_psd(self, data_type):
        if data_type == 'raw':
            self.psd = self.raw_arr.copy().compute_psd()
            #self.psd.plot()
            #self.save_psd(data_type)
        elif data_type == 'filtered':
            self.psd = self.filtered_arr.copy().compute_psd()
            #self.psd.plot()
            self.save_psd(data_type)
        elif data_type == 'delta':
            self.psd = self.decomp_data.copy().compute_psd()
            #self.psd.plot()
            #self.save_psd(data_type)
        elif data_type == 'theta':
            self.psd = self.decomp_data.copy().compute_psd()
            #self.psd.plot()
            #self.save_psd(data_type)
        elif data_type == 'alpha':
            self.psd = self.decomp_data.copy().compute_psd()
            #self.psd.plot()
            #self.save_psd(data_type)
```

```
        elif data_type == 'beta':
            self.psd = self.decomp_data.copy().compute_psd()
            #self.psd.plot()
            #self.save_psd(data_type)
        elif data_type == 'gamma':
            self.psd = self.decomp_data.copy().compute_psd()
            #self.psd.plot()
            #self.save_psd(data_type)
        else:
            print('No data with data_type = {data_type} found')

    def decompose_data(self, freq_band, method):
        '''
        Decompose dataset

        Parameters
        ----------
        freq_band : str
            The frequency band in which to decompose the signal. Either gamma,␣
↪beta, alpha, theta, or delta
        method : str
            Either fir or iir. Decides which filtering method to use.
        '''
        if freq_band == 'gamma':
            self.decomp_data = self.filtered_arr.copy().filter(30, 45,␣
↪method=method)
        elif freq_band == 'beta':
            self.decomp_data = self.filtered_arr.copy().filter(12, 30,␣
↪method=method)
        elif freq_band == 'alpha':
            self.decomp_data = self.filtered_arr.copy().filter(8, 12,␣
↪method=method)
        elif freq_band == 'theta':
            self.decomp_data = self.filtered_arr.copy().filter(4, 8,␣
↪method=method)
        elif freq_band == 'delta':
            self.decomp_data = self.filtered_arr.copy().filter(0, 4,␣
↪method=method)
        else:
            print('No frequency band matching {freq_band} found')

    def vizualize_artifacts(self):
        #kwargs = dict(extrapolate='head')
        channels = ['Fp2', 'FT9']
        for ch_name in channels:
```

```
            self.eog_evoked = create_eog_epochs(self.filtered_arr,␣
↪ch_name=ch_name).average(picks='all')
            self.eog_evoked.apply_baseline((None, None))
            self.eog_evoked.plot_joint()

    def compute_SSP_projectors(self):
        channels = ['Fp2', 'FT9']
        for ch_name in channels:
            self.eog_projs, _ = compute_proj_eog(self.filtered_arr, n_grad=0,␣
↪n_mag=0, n_eeg=1, reject=None,
                                no_proj=True, ch_name=ch_name) #check if grad␣
↪and mag can be 0


    def plot_eog_projectors(self):
        fig = mne.viz.plot_projs_joint(self.eog_projs, self.eog_evoked, 'Fp2')
        fig.suptitle('EOG projectors')

        fig = mne.viz.plot_projs_joint(self.eog_projs, self.eog_evoked, 'FT9')
        fig.suptitle('EOG projectors')


    def plot_ssp(self):
        for title in ('Without', 'With'):
            if title == 'With':
                self.ssp = self.filtered_arr.add_proj(self.eog_projs)
            with mne.viz.use_browser_backend('matplotlib'):
                fig = self.filtered_arr.plot() # original contents of plot␣
↪function: order=artifact_picks, n_channels=len(artifact_picks)
            fig.subplots_adjust(top=0.9)  # make room for title
            fig.suptitle('{} EOG projectors'.format(title), size='xx-large',
                         weight='bold')


    def plot_psd_comparison(self, filename1, filename2, title_psd,␣
↪title_ch_comparison):
        psd1 = sio.loadmat(filename1)
        psd2 = sio.loadmat(filename2)

        fig = plt.figure(1)
        fig.set_figwidth(20)
        fig.set_figheight(10)

        length = len(psd1['psd_data'][0])

        psd1_arr = np.empty([var.NUM_CHANNELS, length])
        psd2_arr = np.empty([var.NUM_CHANNELS, length])
```

```
        for ch in range(var.NUM_CHANNELS):
            for i in range(length):
                db1 = 10*log(psd1['psd_data'][ch][i])
                psd1_arr[ch][i] = db1

                db2 = 10*log(psd2['psd_data'][ch][i])
                psd2_arr[ch][i] = db2

            plt.plot(psd1_arr[ch], color='#3cbfbe')
            plt.plot(psd2_arr[ch], color='#b01b81')
        plt.legend(['Not stressed', 'Stressed'])
        plt.xlabel('Hz')
        plt.ylabel('dB')
        plt.title(title_psd)
        plt.show()

        plt.figure(2)
        plt.suptitle(title_ch_comparison)

        psd1_1 = np.empty([var.NUM_CHANNELS, length])
        psd2_1 = np.empty([var.NUM_CHANNELS, length])

        for i, ch in enumerate(var.CHANNELS):
            for j in range(length):
                db1 = 10*log(psd1['psd_data'][i][j])
                psd1_1[i][j] = db1

                db2 = 10*log(psd2['psd_data'][i][j])
                psd2_1[i][j] = db2

            ax = plt.subplot(2, 4, i+1)
            ax.plot(psd1_1[i], color='#3cbfbe')
            ax.plot(psd2_1[i], color='#b01b81')
            #psd1_1[i].plot(ax=ax, color='#3cbfbe')
            #psd2_1[i].plot(ax=ax, color='#b01b81')
            ax.legend(['Not stressed', 'Stressed'])
            ax.set_title('Channel: %s' %ch)
```

[3]:
```
participant = 'P002'
session = 'S001'
run = '002'
```

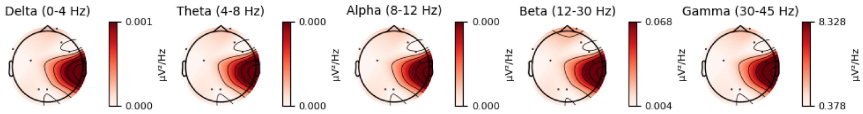
[6]:
```
data_type = 'filtered'
psd_data = Filtering(sub_nr=participant, ses_nr=session, run_nr=run)
psd_data.compute_and_save_filtered_psd(data_type=data_type)
```

```
fname_not_stressed = 'Data/PSD_data/PSD_filtered/sub-P002_ses-S001_run-001.mat'
fname_stressed = 'Data/PSD_data/PSD_filtered/sub-P002_ses-S001_run-002.mat'

title_psd = 'Comparing PSD (init_filter data) between P002_S001_001 and␣
 ↪P002_S001_002'
title_ch_comparison = 'Comparing channel by channel for stressed and␣
 ↪not-stressed between PSD of delta-band for P002_S001_002 and P002_S001_001'

psd_data.plot_psd_comparison(fname_not_stressed, fname_stressed,␣
 ↪title_psd=title_psd, title_ch_comparison=title_ch_comparison)
```

```
Creating RawArray with float64 data, n_channels=8, n_times=75040
    Range : 0 … 75039 =      0.000 …   300.156 secs
Ready.
Filtering raw data in 1 contiguous segment
Setting up band-pass filter from 1 - 50 Hz

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal bandpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 1.00
- Lower transition bandwidth: 1.00 Hz (-6 dB cutoff frequency: 0.50 Hz)
- Upper passband edge: 50.00 Hz
- Upper transition bandwidth: 12.50 Hz (-6 dB cutoff frequency: 56.25 Hz)
- Filter length: 825 samples (3.300 sec)

Setting up band-stop filter

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal bandstop filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower transition bandwidth: 0.25 Hz
- Upper transition bandwidth: 0.25 Hz
- Filter length: 3301 samples (13.204 sec)

Effective window size : 1.024 (s)
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
- Filter order 16 (effective, after forward-backward)
- Cutoffs at 30.00, 45.00 Hz: -6.02, -6.02 dB

Effective window size : 1.024 (s)
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s finished
```

[3]:




```
2023-06-30 12:27:34.375 python[12301:915887] +[CATransaction synchronize] called
within transaction
2023-06-30 12:27:34.507 python[12301:915887] +[CATransaction synchronize] called
within transaction
```

Compute and save decomposed data

[ ]:
```
valid_recordings = get_valid_recordings('raw')
for recordings in valid_recordings:
    sub_nr, ses_nr, run_nr = recordings.split('_')
    test = Filtering(sub_nr, ses_nr, run_nr)
    test.decompose_data('delta', 'iir')
    test.save_decomp_data()
```

[ ]:
```
filt_arr = Filtering(sub_nr=participant, ses_nr=session, run_nr=run)
#filt_arr.vizualize_artifacts()
#filt_arr.compute_SSP_projectors()
#filt_arr.plot_eog_projectors()
#filt_arr.plot_ssp()
#filt_arr.save_ssp_data()
```

[ ]:
```
filt_arr.decompose_data('delta', 'iir')
filt_arr.compute_and_save_filtered_psd('filtered')
```

[ ]:
```
filt_arr.filtered_arr.plot()
```

[ ]:
```
filt_arr.save_decomp_data()
```

[ ]:
```
filt_arr.compute_and_save_filtered_psd('filtered')
filt_arr.psd.plot_topomap()
```

F Complete result overview

Data	Labels	Epoch	Feature	Classifier	Accuracy	Sensitivity	Specificity	Hyperparameters
Raw	STAI	1		EEGNet	72,14%	84,68%	61,54%	
Raw	STAI	2		EEGNet	62,50%	63,64%	61,54%	
Raw	STAI	5		EEGNet	63,14%	73,50%	54,37%	
Raw	PSS	1		EEGNet	54,17%	33,33%	66,67%	
Raw	PSS	2		EEGNet	49,58%	33,33%	59,33%	
Raw	PSS	5		EEGNet	58,47%	57,63%	58,98%	
Raw	STAI	1	Time-series	SVM	56,15%	55,24%	56,91%	C=100, kernel = rbf
Raw	STAI	2	Time-series	SVM	69,16%	50,27%	85,13%	C=100, kernel = poly
Raw	STAI	5	Time-series	SVM	69,00%	52,70%	82,79%	C=100, kernel = poly
Raw	STAI	1	Entropy	SVM				
Raw	STAI	2	Entopy	SVM	57,33%	31,97%	78,78%	C=10, kernel = rbf
Raw	STAI	5	Entropy	SVM	58,62%	42,06%	72,62%	C=10, kernel = rbf
Raw	STAI	1	Hjorth	SVM				
Raw	STAI	2	Hjorth	SVM	51,23%	9,52%	86,53%	C=10000, kernel = poly
Raw	STAI	5	Hjorth	SVM	51,98%	1,69%	94,52%	C=10000, kernel = rbf
Raw	STAI	1	PSD	SVM	68,90%	35,97%	96,76%	C=1, kernel=poly
Raw	STAI	2	PSD	SVM	68,62%	34,90%	97,16%	
Raw	STAI	5	PSD	SVM	60,52%	37,90%	79,66%	C=10, kernel=rbf
Raw	STAI	1	Time-series	RF	70,33%	59,29%	79,68%	max_depth=7, max_features=log2, n_estimators=75
Raw	STAI	2	Time-series	RF	61,10%	42,34%	76,97%	max_depth=7, max_features=auto, n_estimators=50
Raw	STAI	5	Time-series	RF	70,62%	46,38%	91,13%	max_depth = 9, max_features = log2, n_estimators = 125
Raw	STAI	1	Entropy	RF	66,22%	39,83%	88,55%	max_depth=9, max_features=log2, n_estimators=175
Raw	STAI	2	Entopy	RF	66,67%	41,31%	88,13%	max_depth=9, max_features=log2, n_estimators=250
Raw	STAI	5	Entropy	RF	65,75%	43,91%	84,22%	max_depth=9, max_features=sqrt, n_estimators=100
Raw	STAI	1	Hjorth	RF	54,08%	43,20%	63,29%	max_depth=9, max_features=sqrt, n_estimators=125
Raw	STAI	2	Hjorth	RF	53,13%	42,34%	62,26%	max_depth=9, max_features=log2, n_estimators=50
Raw	STAI	5	Hjorth	RF	54,80%	43,91%	64,02%	max_depth=9, max_features=sqrt, n_estimators=125
Raw	STAI	1	PSD	RF	60,35%	26,24%	89,22%	max_depth=9, max_features=auto, n_estimators=225
Raw	STAI	2	PSD	RF	62,19%	30,63%	88,90%	max_depth=9, max_features=log2, n_estimators=75
Raw	STAI	5	PSD	RF	63,70%	31,12%	91,26%	max_depth=9, max_features=log2, n_estimators=100
Raw	STAI	1	Time-series	KNN	52,20%	47,80%	55,93%	leaf_size = 5, n_neighbors = 39, weights = uniform
Raw	STAI	2	Time-series	KNN	45,13%	45,09%	45,17%	leaf_size = 5, neighbors = 15, weights = uniform
Raw	STAI	5	Time-series	KNN	45,55%	50,54%	41,33%	leaf_size = 5, n_neighbors = 35, weights = distance
Raw	STAI	1	Entropy	KNN	63,50%	29,92%	91,92%	
Raw	STAI	2	Entopy	KNN	62,92%	29,96%	90,81%	leaf_size = 5, n_neighbors = 19, weights = distance
Raw	STAI	5	Entropy	KNN	64,34%	39,45%	85,40%	leaf_size = 5, n_neighbors = 15, weights = distance
Raw	STAI	1	Hjorth	KNN	52,76%	37,18%	65,94%	leaf_size = 5, n_neighbors = 39, weights = uniform
Raw	STAI	2	Hjorth	KNN	58,67%	38,50%	75,74%	leaf_size = 5, n_neighbors = 39, weights = uniform
Raw	STAI	5	Hjorth	KNN	60,73%	59,01%	62,19%	leaf_size = 5, n_neighbors = 29, weights = distance
Raw	STAI	1	PSD	KNN	58,96%	33,63%	80,40%	leaf_size = 5, n_neighbors = 5, weights = distance
Raw	STAI	2	PSD	KNN	58,67%	32,52%	80,80%	leaf_size = 5, n_neighbors = 9, weights = distance
Raw	STAI	5	PSD	KNN	60,66%	36,83%	80,83%	leaf_size = 5, n_neighbors = 1, weights = uniform
Raw	PSS	1	Time-series	SVM	50,46%	77,26%	34,38%	C = 0.1, kernel = sigmoid
Raw	PSS	2	Time-series	SVM	70,67%	66,29%	73,29%	c=0.1, kernel = poly
Raw	PSS	5	Time-series	SVM	55,08%	73,26%	44,18%	C=0,1, kernel = sigmoid
Raw	PSS	1	Entropy	SVM	55,82%	50,84%	58,82%	C = 100, kernel = poly
Raw	PSS	2	Entopy	SVM	62,25%	0,15%	99,51%	c=0.01, kernel = poly
Raw	PSS	5	Entropy	SVM	46,54%	56,31%	40,68%	C=0,1, kernel = sigmoid
Raw	PSS	1	Hjorth	SVM	52,51%	77,78%	53,36%	C=0.01, kernel = rbf
Raw	PSS	2	Hjorth	SVM	62,50%	0,00%	100,00%	C=0,01, kernel=sigmoid
Raw	PSS	5	Hjorth	SVM	62,50%	0,00%	100,00%	C=0,01, kernel=sigmoid
Raw	PSS	1	PSD	SVM	58,95%	7,06%	90,08%	c = 0.01, kernel = poly
Raw	PSS	2	PSD	SVM	57,75%	4,40%	89,75%	c=0.01, kernel = poly
Raw	PSS	5	PSD	SVM	61,65%	16,38%	88,81%	C=0,1, kernel = poly
Raw	PSS	1	Time-series	RF	70,97%	77,78%	66,89%	max_depth = 3, max_features = sqrt, n_estimators = 250
Raw	PSS	2	Time-series	RF	69,18%	72,48%	67,20%	max_depth = 3, max_features = auto, n_estimators = 75
Raw	PSS	5	Time-series	RF	70,27%	70,06%	70,40%	max_depth=9, max_features=log2, n_estimators=50
Raw	PSS	1	Entropy	RF	57,33%	58,42%	56,68%	max_depth = 3, max_features = auto, n_estimators = 75
Raw	PSS	2	Entopy	RF	60,10%	56,90%	61,88%	max_depth = 3, max_features = sqrt, n_estimators = 200
Raw	PSS	5	Entropy	RF	62,85%	68,55%	59,44%	max_depth = 7, max_features = log2, n_estimators = 50
Raw	PSS	1	Hjorth	RF	73,65%	77,33%	71,44%	max_depth = 5, max_features = auto, n_estimators = 150
Raw	PSS	2	Hjorth	RF	73,60%	77,63%	71,19%	max_depth = 5, max_features = sqrt, n_estimators = 50
Raw	PSS	5	Hjorth	RF	73,94%	77,78%	71,64%	max_depth = 5, max_features = log2, n_estimators = 250
Raw	PSS	1	PSD	RF	71,06%	65,55%	74,36%	max_depth=7, max_features=auto, n_estimators=175
Raw	PSS	2	PSD	RF	66,83%	65,55%	67,71%	max_depth = 5, max_features = sqrt, n_estimators = 200
Raw	PSS	5	PSD	RF	77,97%	64,22%	86,21%	max_depth = 9, max_features = sqrt, n_estimators = 250
Raw	PSS	1	Time-series	KNN	60,09%	53,81%	63,86%	leaf_size = 5, n_neighbors = 1, weights = uniform
Raw	PSS	2	Time-series	KNN	50,76%	47,05%	52,66%	leaf_size = 5, n_neighbors = 1, weights = uniform
Raw	PSS	5	Time-series	KNN	46,82%	43,88%	48,59%	leaf_size = 5, n_neighbors = 1, weights = uniform
Raw	PSS	1	Entropy	KNN	67,88%	42,55%	83,08%	leaf_size = 5, n_neighbors = 19, weights = distance
Raw	PSS	2	Entopy	KNN	70,36%	45,49%	85,28%	leaf_size = 5, n_neighbors = 35, weights = distance

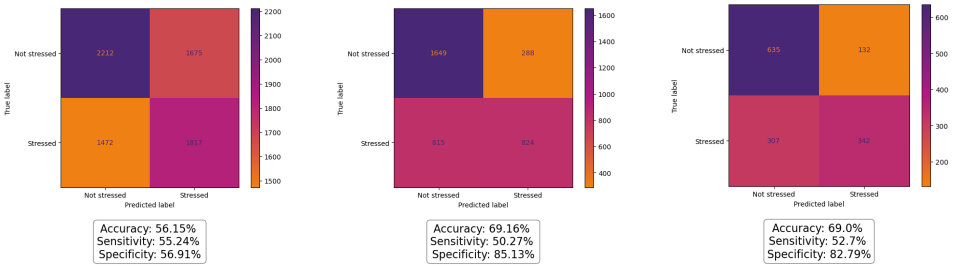
Raw	PSS	5	Entropy	KNN	63,21%	50,66%	70,73%	leaf_size = 5, n_neighbors = 1, weights = uniform
Raw	PSS	1	Hjorth	KNN	55,23%	58,64%	53,18%	leaf_size = 5, n_neighbors = 19, weights = uniform
Raw	PSS	2	Hjorth	KNN	51,37%	47,43%	53,74%	leaf_size = 5, n_neighbors = 1, weights = uniform
Raw	PSS	5	Hjorth	KNN	48,80%	46,14%	50,40%	leaf_size = 5, n_neighbors = 5, weights = uniform
Raw	STAI	1	PSD	KNN	66,83%	64,96%	67,96%	leaf_size = 5, n_neighbors = 1, weights = uniform
Raw	STAI	2	PSD	KNN	66,75%	66,29%	67,02%	leaf_size = 5, n_neighbors = 1, weights = uniform
Raw	STAI	5	PSD	KNN	71,61%	73,82%	70,28%	leaf_size = 5, n_neighbors = 9, weights = distance
SSP	STAI	1		EEGNet	57,66%	48,13%	65,73%	
SSP	STAI	2		EEGNet	54,45%	35,20%	70,73%	
SSP	STAI	5		EEGNet	52,61%	28,97%	72,62%	
SSP	PSS	1		EEGNet	70,28%	34,32%	90,83%	
SSP	PSS	2		EEGNet	58,69%	73,32%	50,34%	
SSP	PSS	5		EEGNet	61,56%	37,71%	75,18%	
SSP	STAI	1	Time-series	SVM				
SSP	STAI	2	Time-series	SVM	46,62%	19,34%	69,70%	C = 10, kernel = linear
SSP	STAI	5	Time-series	SVM	45,34%	18,64%	67,93%	C = 0.01, kernel = linear
SSP	STAI	1	Entropy	SVM				
SSP	STAI	2	Entopy	SVM	54,31%	32,46%	72,79%	C = 10, kernel = poly
SSP	STAI	5	Entropy	SVM	53,11%	31,74%	71,19%	C = 100, kernel = poly
SSP	STAI	1	Hjorth	SVM				
SSP	STAI	2	Hjorth	SVM	53,13%	1,83%	96,54%	C = 10000, kernel = linear
SSP	STAI	5	Hjorth	SVM	52,90%	2,47%	95,57%	C = 100, kernel = linear
SSP	STAI	1	PSD	SVM				
SSP	STAI	2	PSD	SVM	52,38%	41,67%	61,44%	C = 1, kernel = rbf
SSP	STAI	5	PSD	SVM	50,99%	37,60%	62,32%	C = 0.1, kernel = rbf
SSP	STAI	1	Time-series	RF	51,21%	14,47%	82,30%	max_depth = 5, max_features = sqrt, n_estimators = 125
SSP	STAI	2	Time-series	RF	49,02%	18,91%	74,50%	max_depth = 5, max_features = log2, n_estimators = 150
SSP	STAI	5	Time-series	RF	48,94%	25,12%	69,10%	max_depth = 9, max_features = log2, n_estimators = 50
SSP	STAI	1	Entropy	RF	52,33%	25,36%	75,15%	max_depth = 3, max_features = log2, n_estimators = 175
SSP	STAI	2	Entopy	RF	58,33%	39,96%	73,88%	max_depth = 7, max_features = sqrt, n_estimators = 50
SSP	STAI	5	Entropy	RF	57,06%	40,22%	71,32%	max_depth = 5, max_features = auto, n_estimators = 200
SSP	STAI	1	Hjorth	RF	58,49%	31,35%	81,45%	max_depth = 9, max_features = auto, n_estimators = 75
SSP	STAI	2	Hjorth	RF	58,86%	31,79%	81,78%	max_depth = 9, max_features = log2, n_estimators = 75
SSP	STAI	5	Hjorth	RF	56,36%	31,28%	77,57%	max_depth = 7, max_features = auto, n_estimator = 50
SSP	STAI	1	PSD	SVM	55,06%	28,37%	77,64%	max_depth = 9, max_features = sqrt, n_estimators = 250
SSP	STAI	2	PSD	SVM	54,84%	27,21%	78,21%	max_depth = 9, max_features = auto, n_estimators = 150
SSP	STAI	5	PSD	SVM	55,51%	38,83%	69,62%	max_depth = 9, max_features = auto, n_estimators = 150
SSP	STAI	1	Time-series	KNN	50,60%	31,74%	66,56%	leaf_size = 5, n_neighbors = 39, weights = distance
SSP	STAI	2	Time-series	KNN	49,27%	36,30%	60,25%	leaf_size = 5, n_neighbors = 9, weights = distance
SSP	STAI	5	Time-series	KNN	44,77%	36,36%	51,89%	leaf_size = 5, n_neighbors = 15, weights = distance
SSP	STAI	1	Entropy	KNN	54,82%	38,67%	68,48%	leaf_size = 5, n_neighbors = 29, weights = distance
SSP	STAI	2	Entopy	KNN	54,73%	42,40%	65,15%	leaf_size = 5, n_neighbors = 39, weights = distance
SSP	STAI	5	Entropy	KNN	57,34%	46,53%	66,49%	leaf_size = 5, n_neighbors = 39, weights = distance
SSP	STAI	1	Hjorth	KNN	57,93%	52,30%	62,70%	leaf_size = 5, n_neighbors = 9, weights = uniform
SSP	STAI	2	Hjorth	KNN	58,05%	50,40%	64,53%	leaf_size = 5, n_neighbors = 5, weights = uniform
SSP	STAI	5	Hjorth	KNN	58,26%	55,78%	60,37%	leaf_size = 5, n_neighbors = 1, weights = uniform
SSP	STAI	1	PSD	SVM	53,08%	39,34%	64,70%	leaf_size = 5, n_neighbors = 39, weights = distance
SSP	STAI	2	PSD	SVM	54,25%	37,10%	68,77%	leaf_size = 5, n_neighbors = 39, weights = distance
SSP	STAI	5	PSD	SVM	52,40%	44,38%	59,19%	leaf_size = 5, n_neighbors = 39, weights = uniform
SSP	PSS	1	Time-series	SVM				
SSP	PSS	2	Time-series	SVM	61,65%	6,38%	93,24%	c = 1, kernel = poly
SSP	PSS	5	Time-series	SVM	62,56%	34,75%	78,45%	C = 0.1, kernel = sigmoid
SSP	PSS	1	Entropy	SVM				
SSP	PSS	2	Entopy	SVM	57,26%	22,06%	77,37%	C = 0.1, kernel = poly
SSP	PSS	5	Entropy	SVM	59,40%	19,07%	82,45%	C = 0.1, kernel = poly
SSP	PSS	1	Hjorth	SVM				
SSP	PSS	2	Hjorth	SVM	63,64%	0,00%	100,00%	c = 0.01, kernel = rbf
SSP	PSS	5	Hjorth	SVM	42,37%	34,96%	46,61%	C = 10, kernel = sigmoid
SSP	PSS	1	PSD	SVM				
SSP	PSS	2	PSD	SVM	59,64%	57,21%	61,03%	C = 1, kernel = rbf
SSP	PSS	5	PSD	SVM	60,02%	66,31%	56,42%	C = 1, kernel = rbf
SSP	PSS	1	Time-series	RF	63,56%	23,41%	86,50%	max_depth = 3, max_features = sqrt, n_estimators = 150
SSP	PSS	2	Time-series	RF	65,16%	45,72%	76,27%	max_depth = 9, max_features = log2, n_estimators = 50
SSP	PSS	5	Time-series	RF	66,87%	40,25%	82,08%	max_depth = 7, max_features = sqrt, n_estimators = 100
SSP	PSS	1	Entropy	RF	60,55%	49,08%	67,10%	max_depth = 9, max_features = sqrt, n_estimators = 50
SSP	PSS	2	Entopy	RF	56,89%	49,16%	61,31%	max_depth = 9, max_features = log2, n_estimators = 100
SSP	PSS	5	Entropy	RF	55,32%	45,13%	61,14%	max_depth = 9, max_features = sqrt, n_estimators = 175
SSP	PSS	1	Hjorth	RF	64,06%	22,45%	87,84%	max_depth = 3, max_features = auto, n_estimators = 75
SSP	PSS	2	Hjorth	RF	66,35%	26,93%	88,88%	max_depth = 3, max_features = auto, n_estimators = 100
SSP	PSS	5	Hjorth	RF	62,71%	36,44%	77,72%	max_depth = 9, max_features = sqrt, n_estimators = 50
SSP	PSS	1	PSD	RF	60,16%	45,65%	68,44%	max_depth = 9, max_features = sqrt, n_estimators = 50
SSP	PSS	2	PSD	RF	61,47%	46,90%	69,80%	max_depth = 9, max_features = log2, n_estimators = 200

SSP	PSS	5	PSD	RF	61,56%	54,03%	65,86%	max_depth = 9, max_features = log2, n_estimators = 175
SSP	PSS	1	Time-series	KNN	57,57%	46,70%	63,78%	leaf_size = 5, n_neighbors = 5, weights = distance
SSP	PSS	2	Time-series	KNN	58,69%	40,94%	68,84%	leaf_size = 5, n_neighbors = 39, weights = uniform
SSP	PSS	5	Time-series	KNN	53,93%	46,61%	58,11%	leaf_size = 5, n_neighbors = 1, weights = uniform
SSP	PSS	1	Entropy	KNN	55,72%	52,42%	57,60%	leaf_size = 5, n_neighbors = 39, weights = distance
SSP	PSS	2	Entopy	KNN	51,95%	50,84%	52,59%	leaf_size = 5, n_neighbors = 35, weights = distance
SSP	PSS	5	Entropy	KNN	52,08%	47,25%	54,84%	leaf_size = 5, n_neighbors = 39, weights = distance
SSP	PSS	1	Hjorth	KNN	59,97%	54,39%	63,16%	leaf_size = 5, n_neighbors = 39, weights = uniform
SSP	PSS	2	Hjorth	KNN	59,61%	54,70%	62,42%	leaf_size = 5, n_neighbors = 25, weights = uniform
SSP	PSS	5	Hjorth	KNN	57,94%	56,14%	58,96%	leaf_size = 5, n_neighbors = 39, weights = uniform
SSP	PSS	1	PSD	KNN	58,47%	54,06%	60,99%	leaf_size = 5, n_neighbors = 25, weights = uniform
SSP	PSS	2	PSD	KNN	58,45%	48,66%	64,05%	leaf_size = 5, n_neighbors = 39, weights = uniform
SSP	PSS	5	PSD	KNN	59,71%	63,98%	57,26%	leaf_size = 5, n_neighbors = 39, weights = distance
Delta	STAI	1		EEGNet	53,04%	1,64%	96,55%	
Delta	STAI	2		EEGNet	54,70%	3,54%	97,99%	
Delta	STAI	5		EEGNet	55,01%	9,71%	93,35%	
Delta	PSS	1		EEGNet	62,47%	9,77%	94,04%	
Delta	PSS	2		EEGNet	61,83%	16,70%	88.9%	
Delta	PSS	5		EEGNet	61,79%	10,73%	92,43%	
Delta	STAI	1	Time-series	SVM				
Delta	STAI	2	Time-series	SVM				
Delta	STAI	5	Time-series	SVM	60,52%	23,27%	92,05%	C = 10, kernel = rbf
Delta	STAI	1	Entropy	SVM				
Delta	STAI	2	Entropy	SVM				
Delta	STAI	5	Entropy	SVM	55,30%	13,25%	90,87%	c = 1, kernel = poly
Delta	STAI	1	Hjorth	SVM				
Delta	STAI	2	Hjorth	SVM				
Delta	STAI	5	Hjorth	SVM	57,52%	12,63%	95,31%	C = 10000, kernel = rbf
Delta	STAI	1	PSD	SVM				
Delta	STAI	2	PSD	SVM				
Delta	STAI	5	PSD	SVM	54,17%	1,39%	98,83%	c = 0.01, kernel = poly
Delta	STAI	1	Time-series	RF	56,59%	17,82%	89,40%	
Delta	STAI	2	Time-series	RF	58,05%	24,65%	86,32%	max_depth=9, max_features = log2,, n_estimators=200
Delta	STAI	5	Time-series	RF	57,77%	25,58%	85,01%	max_depth=3, max_features = log2, n_estimators =75
Delta	STAI	1	Entropy	RF	55,46%	12,92%	91,46%	max_depth = 3, max_features = log2, n_estimators = 50
Delta	STAI	2	Entropy	RF	56,54%	20,50%	87,04%	max_depth = 3, max_features = auto, n_estimators = 125
Delta	STAI	5	Entropy	RF	56,85%	30,05%	79,53%	max_depth=3, max_features = sqrt, n_estimators = 50
Delta	STAI	1	Hjorth	RF	54,79%	4,20%	97,61%	max_depth=9, max_features = sqrt, n_estimators = 250
Delta	STAI	2	Hjorth	RF	55,01%	5,43%	96,95%	max_depth = 9, max_features = auto, n_estimators = 100
Delta	STAI	5	Hjorth	RF	55,65%	8,32%	95,70%	max_depth = 7, max_features = sqrt, n_estimators = 50
Delta	STAI	1	PSD	RF	54,42%	3,56%	97,45%	max_depth = 5, max_features = auto, n_estimators = 50
Delta	STAI	2	PSD	RF	54,50%	1,34%	99,48%	max_depth=3, max_features = sqrt, n_estimators = 225
Delta	STAI	5	PSD	RF	55,86%	14,33%	91,00%	max_depth = 5, max_features = auto, n_estimators =175
Delta	STAI	1	Time-series	KNN	54,85%	30,01%	75,87%	leaf_size = 5, n_neighbors = 39, weights = distance
Delta	STAI	2	Time-series	KNN	55,23%	32,52%	74,45%	leaf_size = 5, n_neighbors = 35, weights = distance
Delta	STAI	5	Time-series	KNN	55,72%	33,90%	74,19%	leaf_size = 5, n_neighbors = 35, weights = distance
Delta	STAI	1	Entropy	KNN	54,25%	24,90%	79,08%	leaf_size = 5, n_neighbors = 39, weights = uniform
Delta	STAI	2	Entropy	KNN	55,70%	30,02%	77,44%	leaf_size = 5, n_neighbors = 39, weights = uniform
Delta	STAI	5	Entropy	KNN	55,65%	38,21%	70,40%	leaf_size = 5, n_neighbors = 19, weights = distance
Delta	STAI	1	Hjorth	KNN	52,08%	21,25%	78,16%	leaf_size = 5, n_neighbors = 39, weights = uniform
Delta	STAI	2	Hjorth	KNN	53,38%	23,73%	78,47%	leaf_size = 5, n_neighbors = 29, weights = distance
Delta	STAI	5	Hjorth	KNN	54,80%	31,59%	74,45%	leaf_size = 5, n_neighbors = 29, weights = distance
Delta	STAI	1	PSD	KNN	50,96%	26,54%	71,62%	leaf_size = 5, n_neighbors = 39, weights = distance
Delta	STAI	2	PSD	KNN	50,59%	28,37%	69,39%	leaf_size = 5, n_neighbors = 29, weights = uniform
Delta	STAI	5	PSD	KNN	54,03%	32,82%	71,97%	leaf_size = 5, n_neighbors = 35, weights = uniform
Delta	PSS	1	Time-series	SVM				
Delta	PSS	2	Time-series	SVM	62,61%	0,75%	99,73%	c = 0.01, kernel = poly
Delta	PSS	5	Time-series	SVM	62,50%	0,00%	100,00%	c = 0.01, kernel = rbf
Delta	PSS	1	Entropy	SVM				
Delta	PSS	2	Entropy	SVM	62,56%	0,82%	99,60%	c = 0.01, kernel = rbf
Delta	PSS	5	Entropy	SVM	62,15%	21,66%	86,44%	c = 1, kernel = poly
Delta	PSS	1	Hjorth	SVM				
Delta	PSS	2	Hjorth	SVM	62,50%	0,00%	100,00%	c = 0.01, kernel = rbf
Delta	PSS	5	Hjorth	SVM	62,50%	0,00%	100,00%	c = 0.01, kernel = rbf
Delta	PSS	1	PSD	SVM				
Delta	PSS	2	PSD	SVM	62,50%	0,00%	100,00%	C = 0.01, kernel = linear
Delta	PSS	5	PSD	SVM	62,50%	0,00%	100,00%	c = 0.01, kernel = sigmoid
Delta	PSS	1	Time-series	RF	62,71%	19,47%	88,65%	max_depth = 9, max_features = sqrt, n_estimators = 175
Delta	PSS	2	Time-series	RF	61,49%	31,84%	79,28%	max_depth = 7, max_features = auto, n_estimators = 225
Delta	PSS	5	Time-series	RF	65,32%	39,36%	80,90%	max_depth = 7, max_features = sqrt, n_estimators = 50
Delta	PSS	1	Entropy	RF	62,57%	19,81%	88,23%	max_depth = 7, max_features = sqrt, n_estimators = 100
Delta	PSS	2	Entropy	RF	61,41%	29,90%	80,31%	max_depth = 9max_features = sqrt, n_estimators = 125

Delta	PSS	5	Entropy	RF	62,43%	41,24%	75,14%	max_depth = 9, max_features = sqrt, n_estimators = 200
Delta	PSS	1	Hjorth	RF	63,13%	1,97%	99,82%	max_depth = 3, max_features = log2, n_estimators = 50
Delta	PSS	2	Hjorth	RF	62,81%	1,27%	99,73%	max_depth = 3, max_features = auto, n_estimators = 125
Delta	PSS	5	Hjorth	RF	64,69%	6,59%	99,55%	max_depth = 3, max_features = auto, n_estimators = 225
Delta	PSS	1	PSD	RF	62,51%	0,04%	100,00%	max_depth = 3, max_features = sqrt, n_estimators = 100
Delta	PSS	2	PSD	RF	62,53%	0,07%	100,00%	max_depth = 3, max_features = log2, n_estimaators = 125
Delta	PSS	5	PSD	RF	62,15%	0,94%	98,87%	max_depth = 3, max_features = sqrt, n_estimators = 175
Delta	PSS	1	Time-series	KNN	63,82%	44,11%	75,65%	leaf_size = 5, n_neighbors = 35, weights = distance
Delta	PSS	2	Time-series	KNN	62,53%	45,34%	72,84%	leaf_size = 5, n_neighbors = 19, weights = distance
Delta	PSS	5	Time-series	KNN	64,19%	49,72%	72,88%	leaf_size = 5, n_neighbors = 35, weights = uniform
Delta	PSS	1	Entropy	KNN	58,47%	37,35%	71,15%	leaf_size = 5, n_neighbors = 39, weights = uniform
Delta	PSS	2	Entropy	KNN	57,66%	42,88%	66,53%	leaf_size = 5, n_neighbors = 39, weights = uniform
Delta	PSS	5	Entropy	KNN	58,83%	57,63%	59,55%	leaf_size = 5, n_neighbors = 29, weights = distance
Delta	PSS	1	Hjorth	KNN	60,91%	37,57%	74,92%	leaf_size = 5, n_neighbors = 35, weights = distance
Delta	PSS	2	Hjorth	KNN	61,47%	37,29%	75,97%	leaf_size = 5, n_neighbors = 39, weights = uniform
Delta	PSS	5	Hjorth	KNN	61,79%	50,28%	68,70%	leaf_size = 5, n_neighbors = 35, weights = uniform
Delta	PSS	1	PSD	KNN	56,20%	37,42%	67,47%	leaf_size = 5, n_neighbors = 39, weights = distance
Delta	PSS	2	PSD	KNN	55,26%	36,61%	66,44%	leaf_size = 5, n_neighbors = 35, weights = distance
Delta	PSS	5	PSD	KNN	54,73%	43,69%	61,36%	leaf_size = 5, n_neighbors = 39, weights = distance

G Time-series results

Support Vector Machine (SVM)



(a) STAI-Y-labels with 1s epoch.

(b) STAI-Y-labels with 2s epoch.

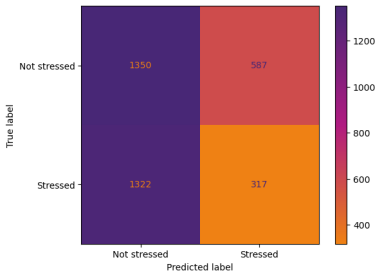
(c) STAI-Y-labels with 5s epoch.

(d) SS-labels with 1s epoch.

(e) SS-labels with 2s epoch.

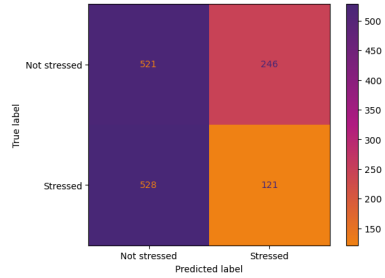
(f) SS-labels with 5s epoch.

Figure G.1: Raw data classified with SVM with both label types and three different epoch lengths



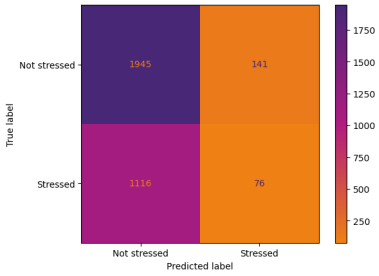
Accuracy: 46.62%
Sensitivity: 19.34%
Specificity: 69.7%

(a) STAI-Y-labels with 2s epoch.



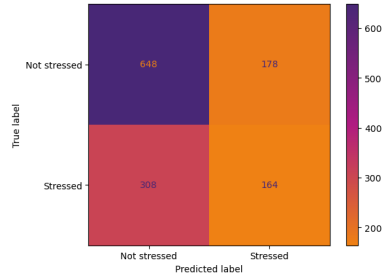
Accuracy: 45.34%
Sensitivity: 18.64%
Specificity: 67.93%

(b) STAI-Y-labels with 5s epoch.



Accuracy: 61.65%
Sensitivity: 6.38%
Specificity: 93.24%

(c) SS-labels with 2s epoch.

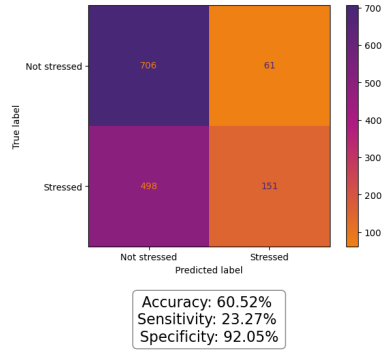


Accuracy: 62.56%
Sensitivity: 34.75%
Specificity: 78.45%

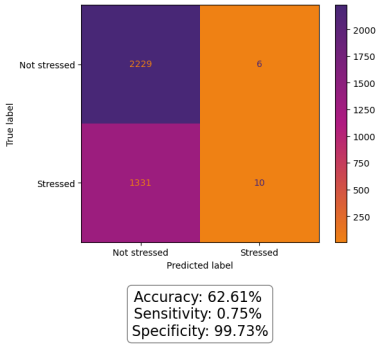
(d) SS-labels with 5s epoch.

Figure G.2: SSP data classified with SVM with both label types and two different epoch lengths

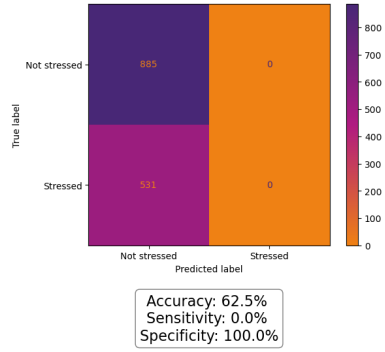
(a) STAI-Y-labels with 2s epoch **not recorded**.



(b) STAI-Y-labels with 5s epoch.



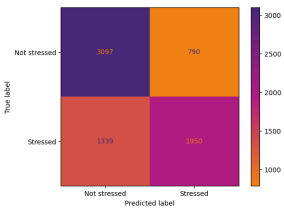
(c) SS-labels with 2s epoch.



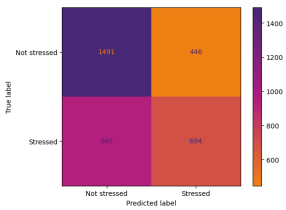
(d) SS-labels with 5s epoch.

Figure G.3: Delta-band data classified with SVM with both label types and two different epoch lengths

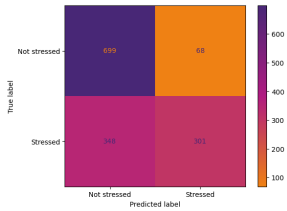
Random Forest (RF)



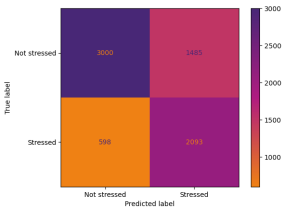
(a) STAI-Y-labels with 1s epoch.



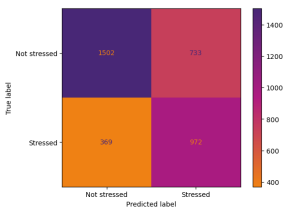
(b) STAI-Y-labels with 2s epoch.



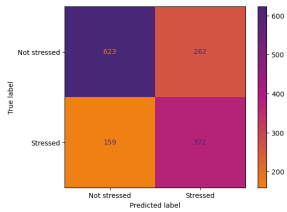
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

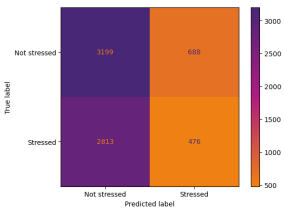


(e) SS-labels with 2s epoch.

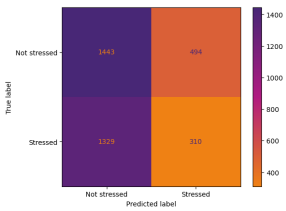


(f) SS-labels with 5s epoch.

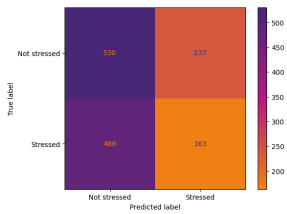
Figure G.4: Raw data classified with RF with both label types and three different epoch lengths



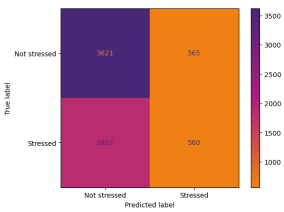
(a) STAI-Y-labels with 1s epoch.



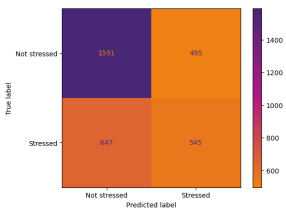
(b) STAI-Y-labels with 2s epoch.



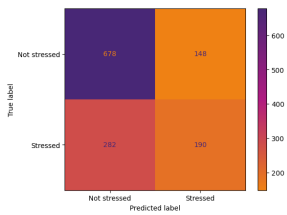
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

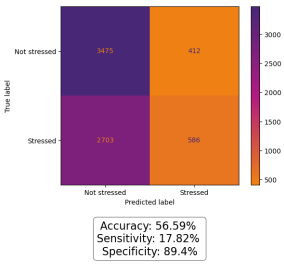


(e) SS-labels with 2s epoch.



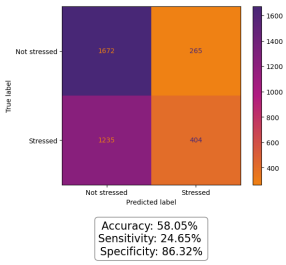
(f) SS-labels with 5s epoch.

Figure G.5: SSP data classified with RF with both label types and three different epoch lengths



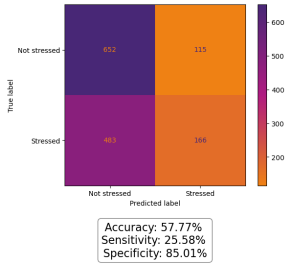
Accuracy: 56.59%
Sensitivity: 17.82%
Specificity: 89.4%

(a) STAI-Y-labels with 1s epoch.



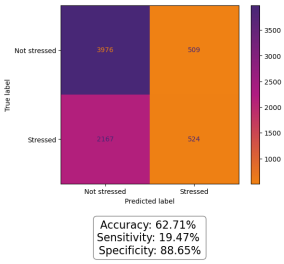
Accuracy: 58.05%
Sensitivity: 24.65%
Specificity: 86.32%

(b) STAI-Y-labels with 2s epoch.



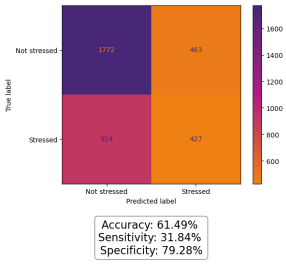
Accuracy: 57.77%
Sensitivity: 25.58%
Specificity: 85.01%

(c) STAI-Y-labels with 5s epoch.



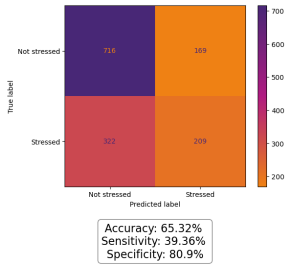
Accuracy: 62.71%
Sensitivity: 19.47%
Specificity: 88.65%

(d) SS-labels with 1s epoch.



Accuracy: 61.49%
Sensitivity: 31.84%
Specificity: 79.28%

(e) SS-labels with 2s epoch.

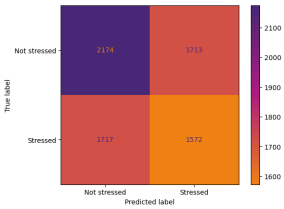


Accuracy: 65.32%
Sensitivity: 39.36%
Specificity: 80.9%

(f) SS-labels with 5s epoch.

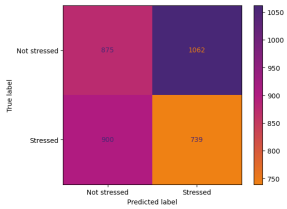
Figure G.6: Delta-band data classified with RF with both label types and three different epoch lengths

K-Nearest Neighbor



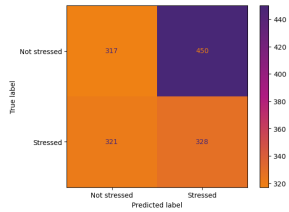
Accuracy: 52.2%
Sensitivity: 47.8%
Specificity: 55.93%

(a) STAI-Y-labels with 1s epoch.



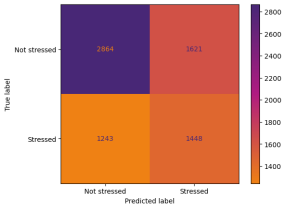
Accuracy: 45.13%
Sensitivity: 45.09%
Specificity: 45.17%

(b) STAI-Y-labels with 2s epoch.



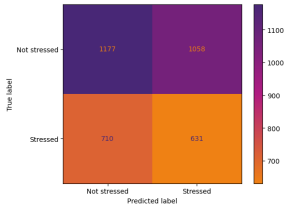
Accuracy: 45.55%
Sensitivity: 50.54%
Specificity: 41.33%

(c) STAI-Y-labels with 5s epoch.



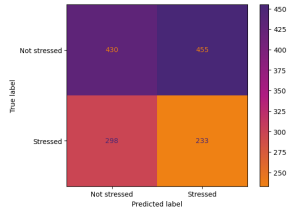
Accuracy: 60.09%
Sensitivity: 53.81%
Specificity: 63.86%

(d) SS-labels with 1s epoch.



Accuracy: 50.56%
Sensitivity: 47.05%
Specificity: 52.66%

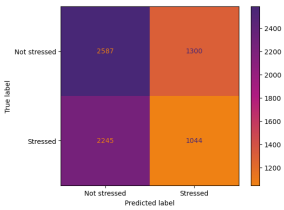
(e) SS-labels with 2s epoch.



Accuracy: 46.82%
Sensitivity: 43.88%
Specificity: 48.59%

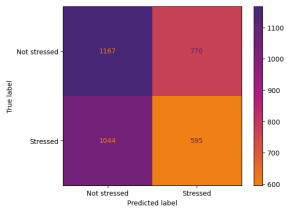
(f) SS-labels with 5s epoch.

Figure G.7: Raw data classified with KNN with both label types and three different epoch lengths



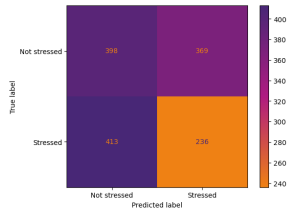
Accuracy: 50.6%
Sensitivity: 31.74%
Specificity: 66.56%

(a) STAI-Y-labels with 1s epoch.



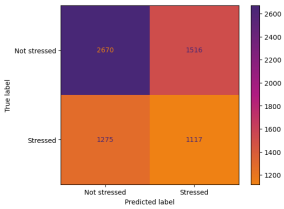
Accuracy: 49.27%
Sensitivity: 36.3%
Specificity: 60.25%

(b) STAI-Y-labels with 2s epoch.



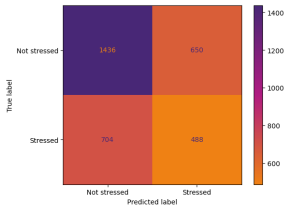
Accuracy: 44.77%
Sensitivity: 36.36%
Specificity: 51.89%

(c) STAI-Y-labels with 5s epoch.



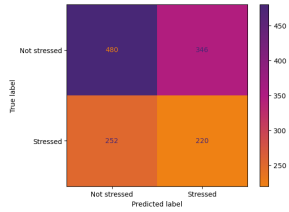
Accuracy: 57.57%
Sensitivity: 46.7%
Specificity: 63.78%

(d) SS-labels with 1s epoch.



Accuracy: 58.69%
Sensitivity: 40.94%
Specificity: 68.84%

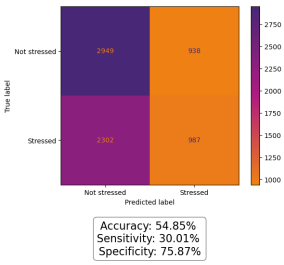
(e) SS-labels with 2s epoch.



Accuracy: 53.93%
Sensitivity: 46.61%
Specificity: 58.11%

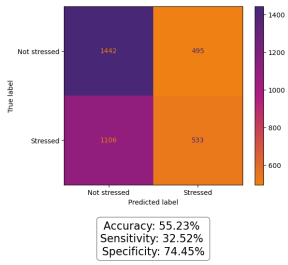
(f) SS-labels with 5s epoch.

Figure G.8: SSP data classified with KNN with both label types and three different epoch lengths



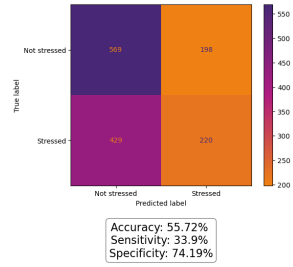
Accuracy: 54.85%
Sensitivity: 30.01%
Specificity: 75.87%

(a) STAI-Y-labels with 1s epoch.



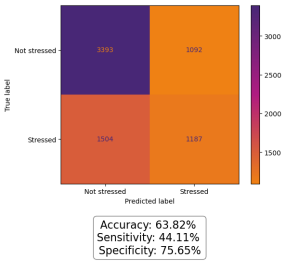
Accuracy: 55.23%
Sensitivity: 32.52%
Specificity: 74.45%

(b) STAI-Y-labels with 2s epoch.



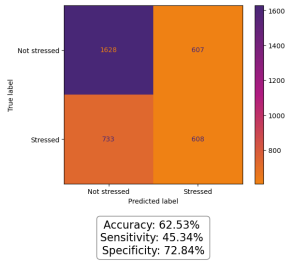
Accuracy: 55.72%
Sensitivity: 33.9%
Specificity: 74.19%

(c) STAI-Y-labels with 5s epoch.



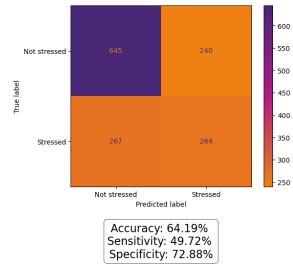
Accuracy: 63.82%
Sensitivity: 44.11%
Specificity: 75.65%

(d) SS-labels with 1s epoch.



Accuracy: 62.53%
Sensitivity: 45.34%
Specificity: 72.84%

(e) SS-labels with 2s epoch.



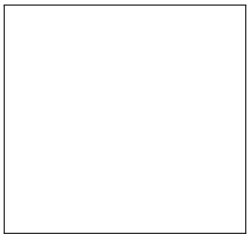
Accuracy: 64.19%
Sensitivity: 49.72%
Specificity: 72.88%

(f) SS-labels with 5s epoch.

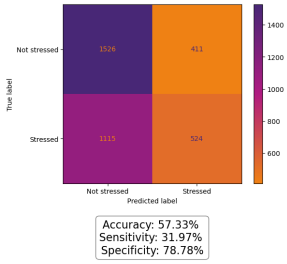
Figure G.9: Delta-band data classified with KNN with both label types and three different epoch lengths

H Entropy

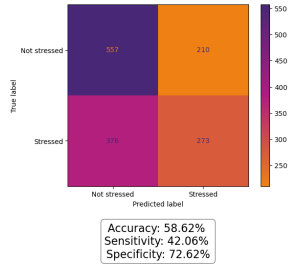
SVM



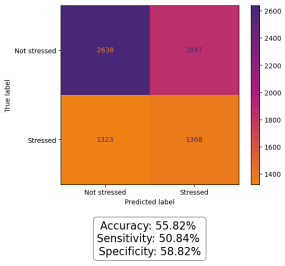
(a) STAI-Y-labels with 1s epoch **not recorded**.



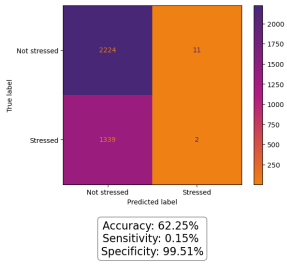
(b) STAI-Y-labels with 2s epoch.



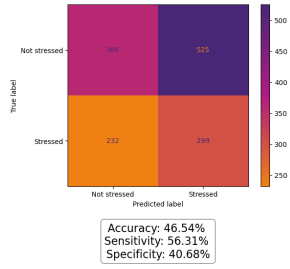
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

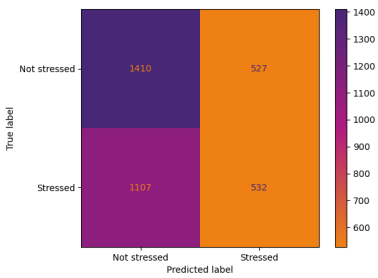


(e) SS-labels with 2s epoch.



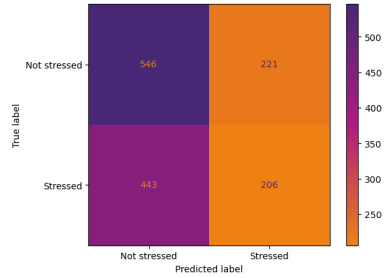
(f) SS-labels with 5s epoch.

Figure H.1: Raw data classified with SVM with both label types and three different epoch lengths



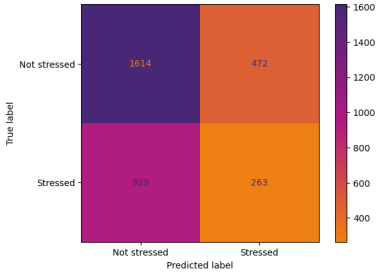
Accuracy: 54.31%
Sensitivity: 32.46%
Specificity: 72.79%

(a) STAI-Y-labels with 2s epoch.



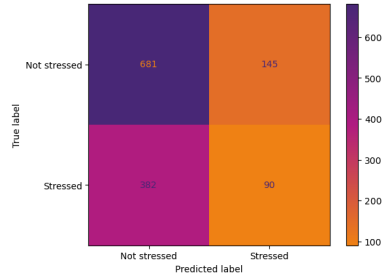
Accuracy: 53.11%
Sensitivity: 31.74%
Specificity: 71.19%

(b) STAI-Y-labels with 5s epoch.



Accuracy: 57.26%
Sensitivity: 22.06%
Specificity: 77.37%

(c) SS-labels with 2s epoch.

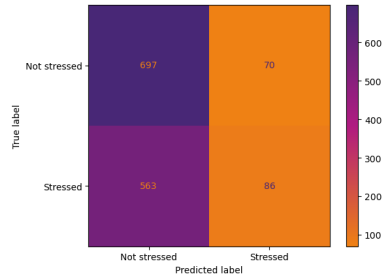


Accuracy: 59.4%
Sensitivity: 19.07%
Specificity: 82.45%

(d) SS-labels with 5s epoch.

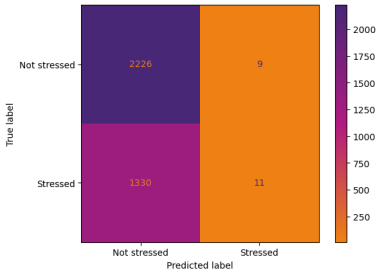
Figure H.2: SSP data classified with SVM with both label types and two different epoch lengths

(a) STAI-Y-labels with 2s epoch **not recorded**.



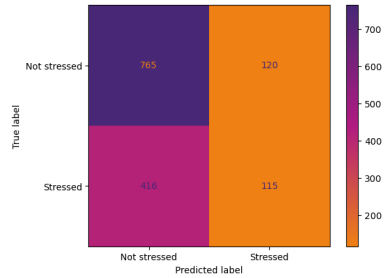
Accuracy: 55.3%
Sensitivity: 13.25%
Specificity: 90.87%

(b) STAI-Y-labels with 5s epoch.



Accuracy: 62.56%
Sensitivity: 0.82%
Specificity: 99.6%

(c) SS-labels with 2s epoch.

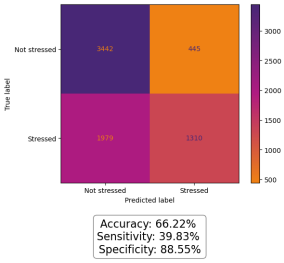


Accuracy: 62.15%
Sensitivity: 21.66%
Specificity: 86.44%

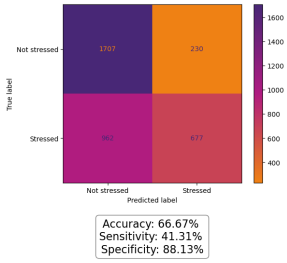
(d) SS-labels with 5s epoch.

Figure H.3: Delta-band data classified with SVM with both label types and two different epoch lengths

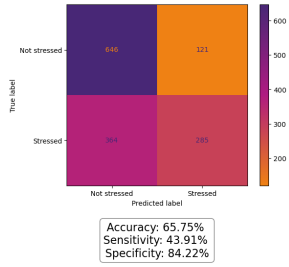
RF



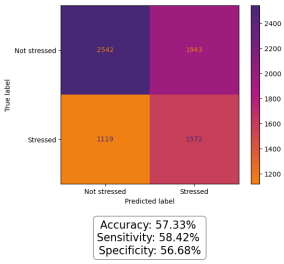
(a) STAI-Y-labels with 1s epoch.



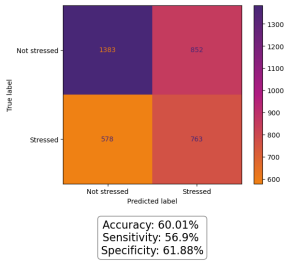
(b) STAI-Y-labels with 2s epoch.



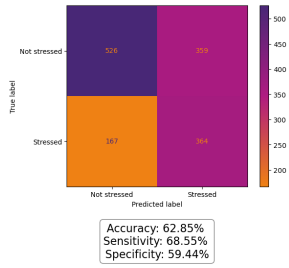
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

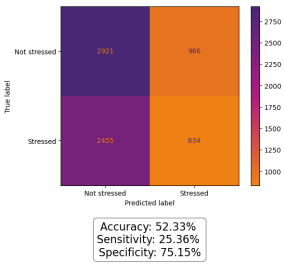


(e) SS-labels with 2s epoch.

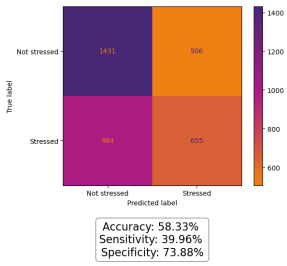


(f) SS-labels with 5s epoch.

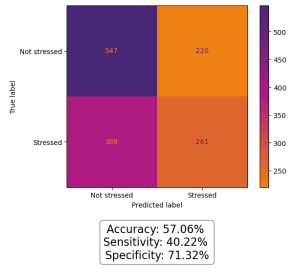
Figure H.4: Raw data classified with RF with both label types and three different epoch lengths



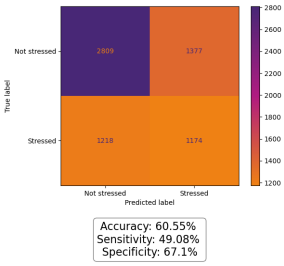
(a) STAI-Y-labels with 1s epoch.



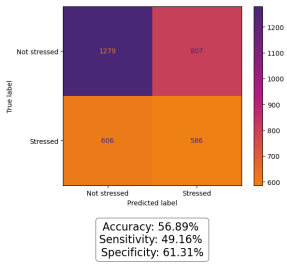
(b) STAI-Y-labels with 2s epoch.



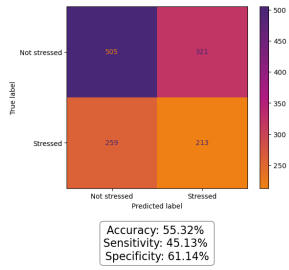
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

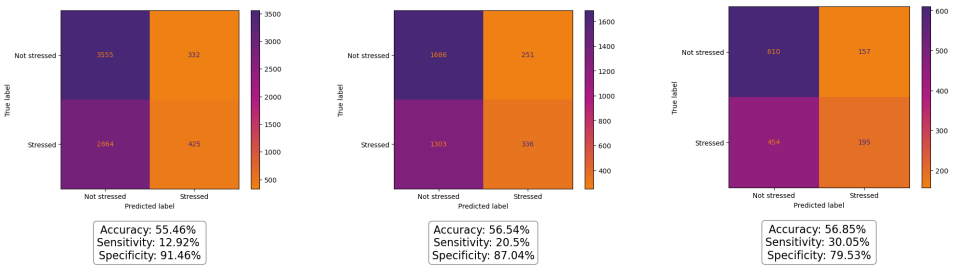


(e) SS-labels with 2s epoch.

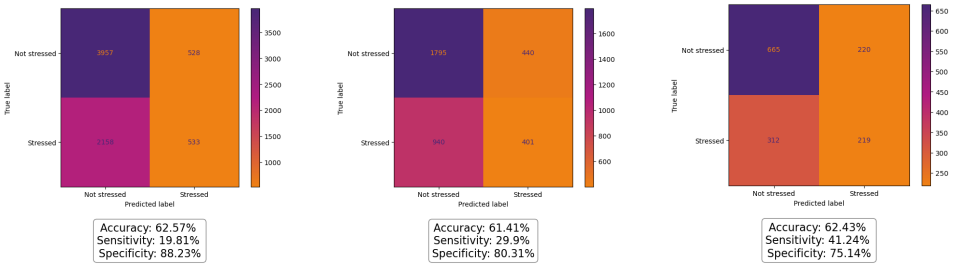


(f) SS-labels with 5s epoch.

Figure H.5: SSP data classified with RF with both label types and three different epoch lengths



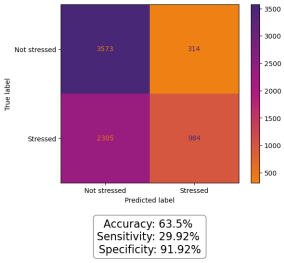
(a) STAI-Y-labels with 1s epoch. **(b)** STAI-Y-labels with 2s epoch. **(c)** STAI-Y-labels with 5s epoch.



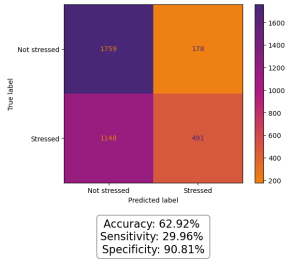
(d) SS-labels with 1s epoch. **(e)** SS-labels with 2s epoch. **(f)** SS-labels with 5s epoch.

Figure H.6: Delta-band data classified with RF with both label types and three different epoch lengths

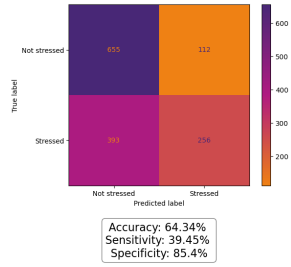
KNN



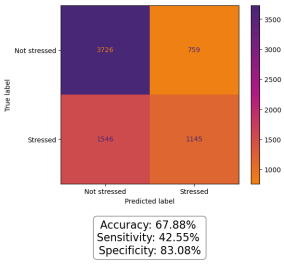
(a) STAI-Y-labels with 1s epoch.



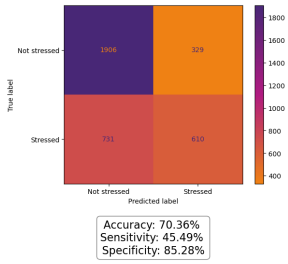
(b) STAI-Y-labels with 2s epoch.



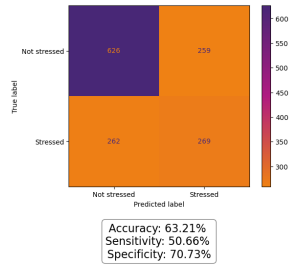
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

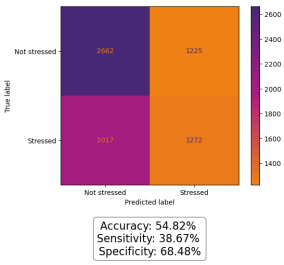


(e) SS-labels with 2s epoch.

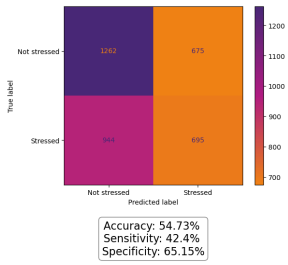


(f) SS-labels with 5s epoch.

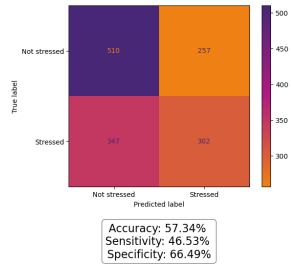
Figure H.7: Raw data classified with KNN with both label types and three different epoch lengths



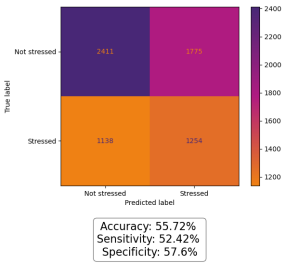
(a) STAI-Y-labels with 1s epoch.



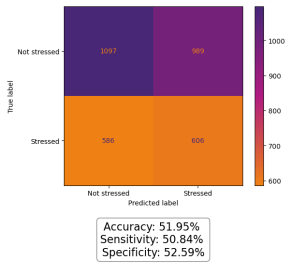
(b) STAI-Y-labels with 2s epoch.



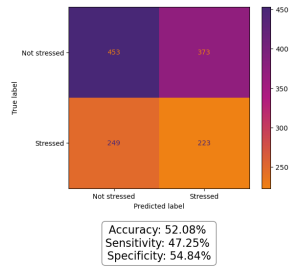
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

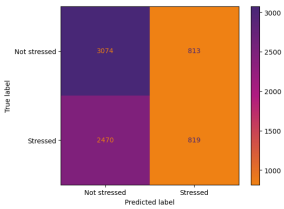


(e) SS-labels with 2s epoch.



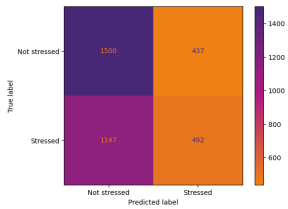
(f) SS-labels with 5s epoch.

Figure H.8: SSP data classified with KNN with both label types and three different epoch lengths



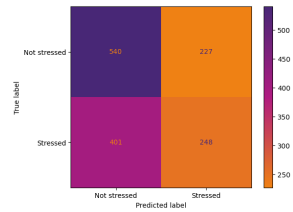
Accuracy: 54.25%
Sensitivity: 24.9%
Specificity: 79.08%

(a) STAI-Y-labels with 1s epoch.



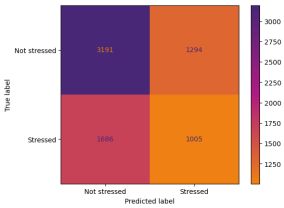
Accuracy: 55.7%
Sensitivity: 30.02%
Specificity: 77.44%

(b) STAI-Y-labels with 2s epoch.



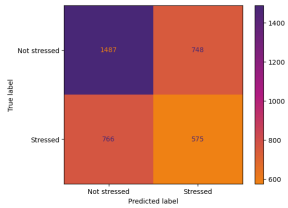
Accuracy: 55.65%
Sensitivity: 38.21%
Specificity: 70.4%

(c) STAI-Y-labels with 5s epoch.



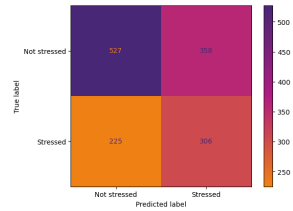
Accuracy: 58.47%
Sensitivity: 37.35%
Specificity: 71.15%

(d) SS-labels with 1s epoch.



Accuracy: 57.66%
Sensitivity: 42.88%
Specificity: 66.53%

(e) SS-labels with 2s epoch.



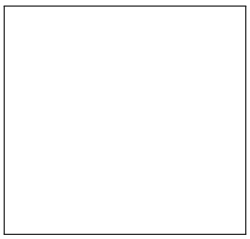
Accuracy: 58.83%
Sensitivity: 57.63%
Specificity: 59.55%

(f) SS-labels with 5s epoch.

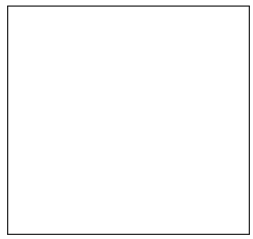
Figure H.9: Delta-band data classified with KNN with both label types and three different epoch lengths

I Hjorth

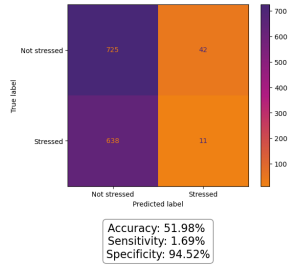
SVM



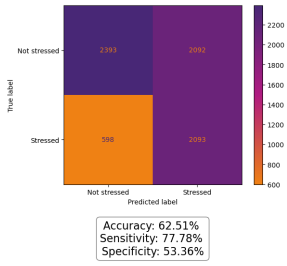
(a) STAI-Y-labels with 1s epoch **not recorded**.



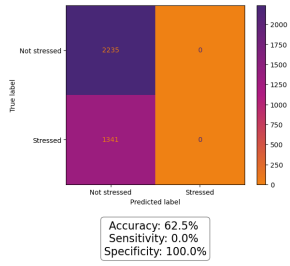
(b) STAI-Y-labels with 2s epoch **not recorded**.



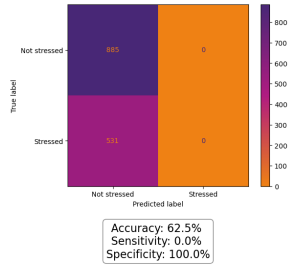
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

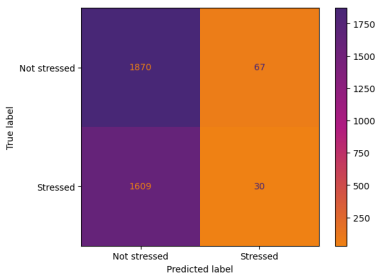


(e) SS-labels with 2s epoch.



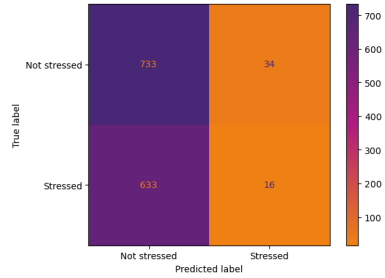
(f) SS-labels with 5s epoch.

Figure I.1: Raw data classified with SVM with both label types and three different epoch lengths



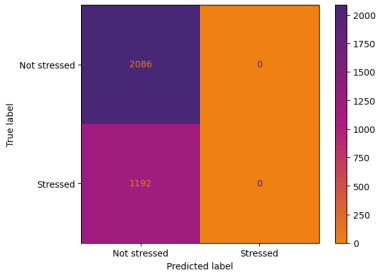
Accuracy: 53.13%
Sensitivity: 1.83%
Specificity: 96.54%

(a) STAI-Y-labels with 2s epoch.



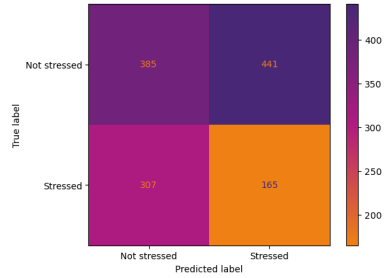
Accuracy: 52.9%
Sensitivity: 2.47%
Specificity: 95.57%

(b) STAI-Y-labels with 5s epoch.



Accuracy: 63.64%
Sensitivity: 0.0%
Specificity: 100.0%

(c) SS-labels with 2s epoch.

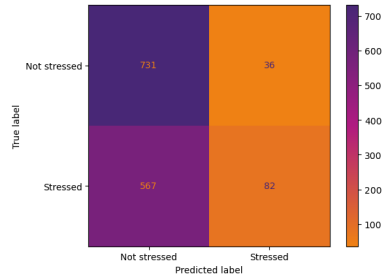


Accuracy: 42.37%
Sensitivity: 34.96%
Specificity: 46.61%

(d) SS-labels with 5s epoch.

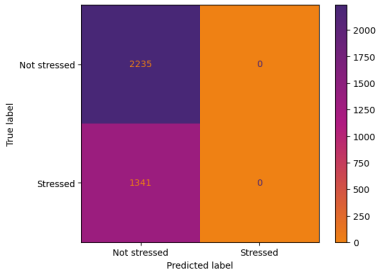
Figure I.2: SSP data classified with SVM with both label types and two different epoch lengths

(a) STAI-Y-labels with 2s epoch **not recorded**.



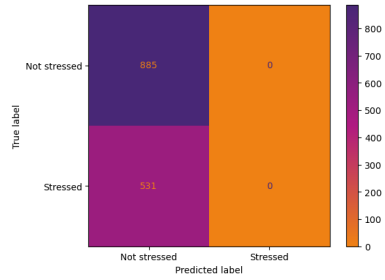
Accuracy: 57.42%
Sensitivity: 12.63%
Specificity: 95.31%

(b) STAI-Y-labels with 5s epoch.



Accuracy: 62.5%
Sensitivity: 0.0%
Specificity: 100.0%

(c) SS-labels with 2s epoch.

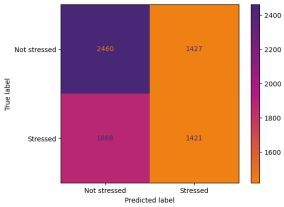


Accuracy: 62.5%
Sensitivity: 0.0%
Specificity: 100.0%

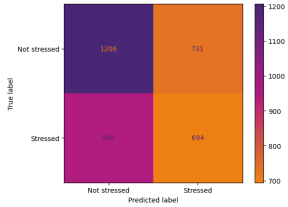
(d) SS-labels with 5s epoch.

Figure I.3: Delta-band data classified with SVM with both label types and two different epoch lengths

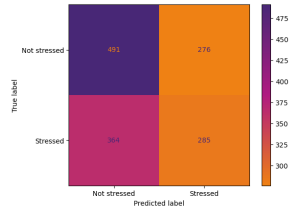
RF



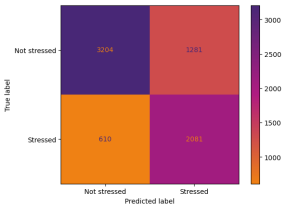
(a) STAI-Y-labels with 1s epoch.



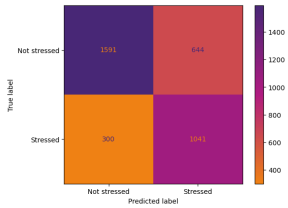
(b) STAI-Y-labels with 2s epoch.



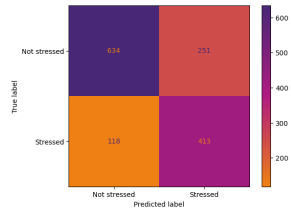
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

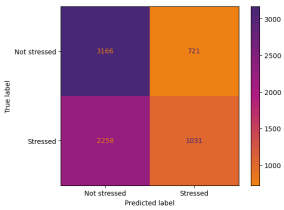


(e) SS-labels with 2s epoch.

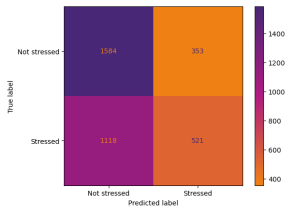


(f) SS-labels with 5s epoch.

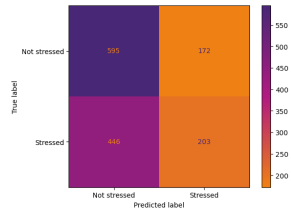
Figure I.4: Raw data classified with RF with both label types and three different epoch lengths



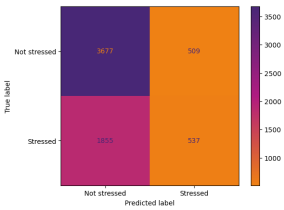
(a) STAI-Y-labels with 1s epoch.



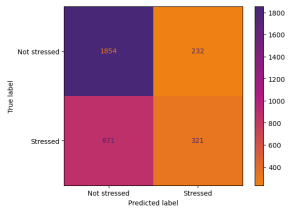
(b) STAI-Y-labels with 2s epoch.



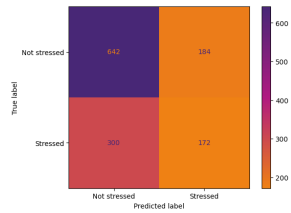
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

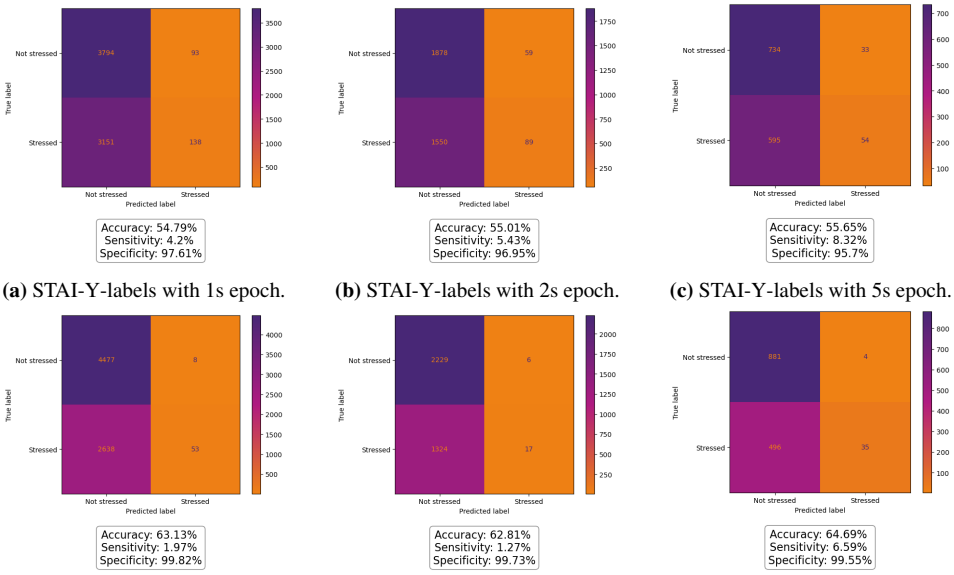


(e) SS-labels with 2s epoch.



(f) SS-labels with 5s epoch.

Figure I.5: SSP data classified with RF with both label types and three different epoch lengths

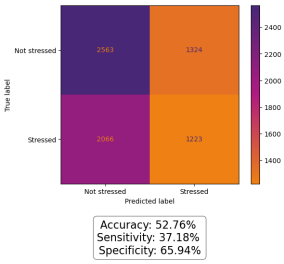


(a) STAI-Y-labels with 1s epoch. **(b)** STAI-Y-labels with 2s epoch. **(c)** STAI-Y-labels with 5s epoch.

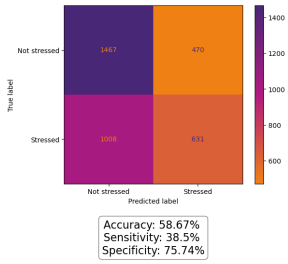
(d) SS-labels with 1s epoch. **(e)** SS-labels with 2s epoch. **(f)** SS-labels with 5s epoch.

Figure I.6: Delta-band data classified with RF with both label types and three different epoch lengths

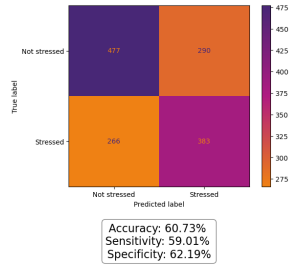
KNN



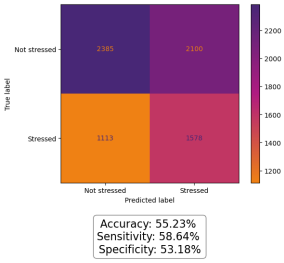
(a) STAI-Y-labels with 1s epoch.



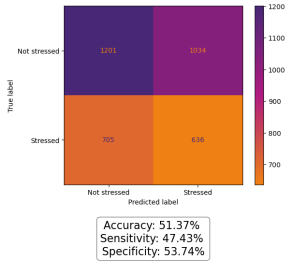
(b) STAI-Y-labels with 2s epoch.



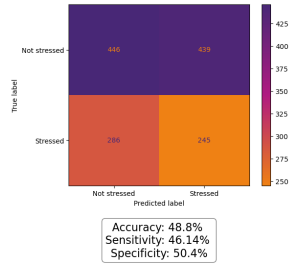
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

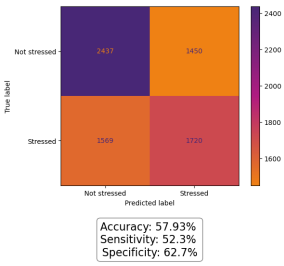


(e) SS-labels with 2s epoch.

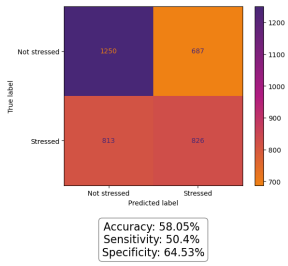


(f) SS-labels with 5s epoch.

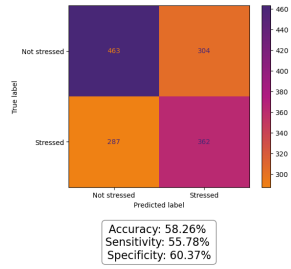
Figure I.7: Raw data classified with KNN with both label types and three different epoch lengths



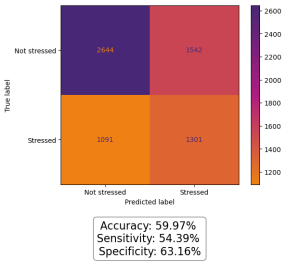
(a) STAI-Y-labels with 1s epoch.



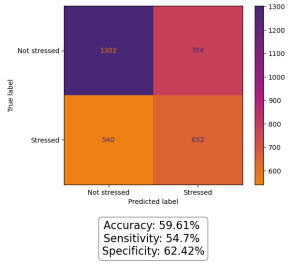
(b) STAI-Y-labels with 2s epoch.



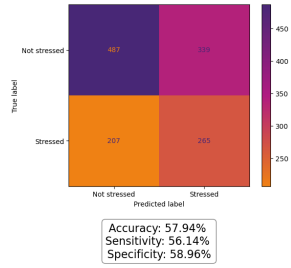
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

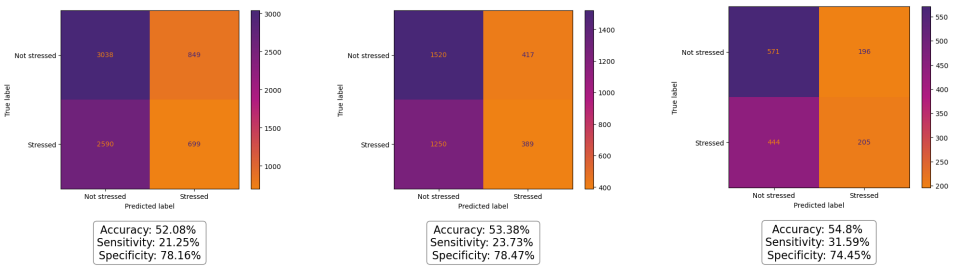


(e) SS-labels with 2s epoch.

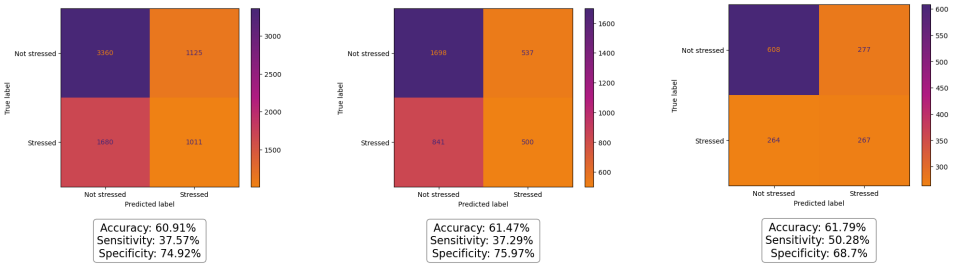


(f) SS-labels with 5s epoch.

Figure I.8: SSP data classified with KNN with both label types and three different epoch lengths



(a) STAI-Y-labels with 1s epoch. **(b)** STAI-Y-labels with 2s epoch. **(c)** STAI-Y-labels with 5s epoch.

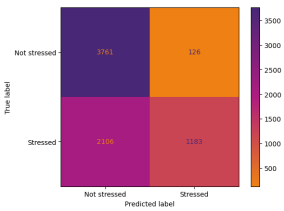


(d) SS-labels with 1s epoch. **(e)** SS-labels with 2s epoch. **(f)** SS-labels with 5s epoch.

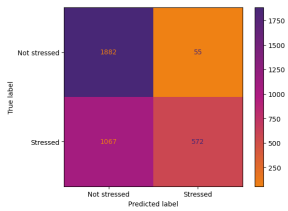
Figure I.9: Delta-band data classified with KNN with both label types and three different epoch lengths

J Power Spectral Density (PSD)

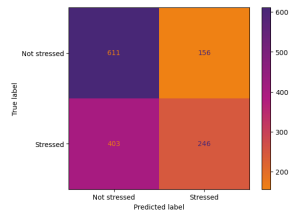
SVM



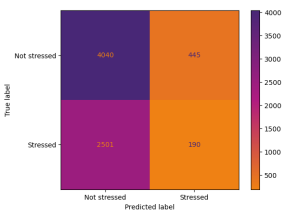
(a) STAI-Y-labels with 1s epoch.



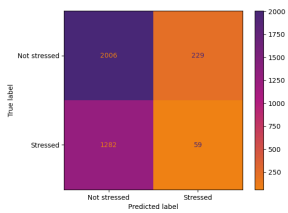
(b) STAI-Y-labels with 2s epoch.



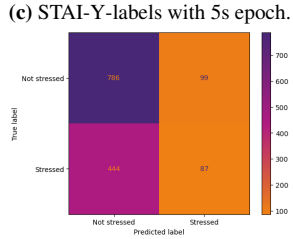
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

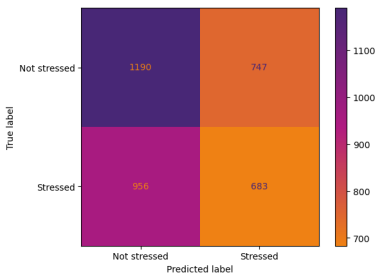


(e) SS-labels with 2s epoch.



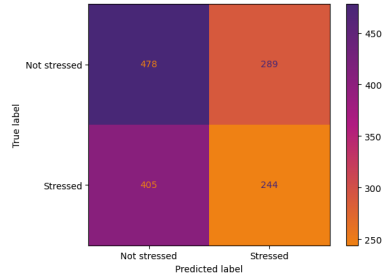
(f) SS-labels with 5s epoch.

Figure J.1: Raw data classified with SVM with both label types and three different epoch lengths



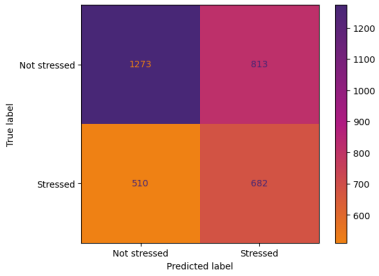
Accuracy: 52.38%
Sensitivity: 41.67%
Specificity: 61.44%

(a) STAI-Y-labels with 2s epoch.



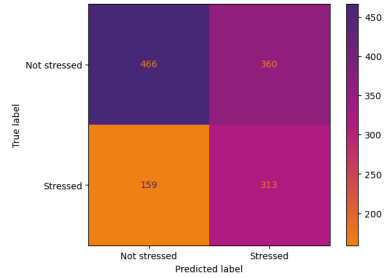
Accuracy: 50.99%
Sensitivity: 37.6%
Specificity: 62.32%

(b) STAI-Y-labels with 5s epoch.



Accuracy: 59.64%
Sensitivity: 57.21%
Specificity: 61.03%

(c) SS-labels with 2s epoch.

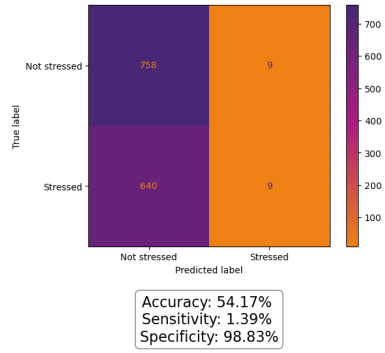


Accuracy: 60.02%
Sensitivity: 66.31%
Specificity: 56.42%

(d) SS-labels with 5s epoch.

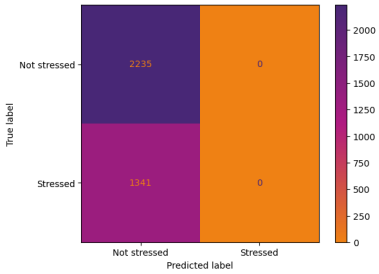
Figure J.2: SSP data classified with SVM with both label types and two different epoch lengths

(a) STAI-Y-labels with 2s epoch **not recorded**.



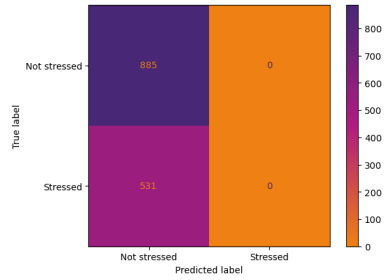
Accuracy: 54.17%
Sensitivity: 1.39%
Specificity: 98.83%

(b) STAI-Y-labels with 5s epoch.



Accuracy: 62.5%
Sensitivity: 0.0%
Specificity: 100.0%

(c) SS-labels with 2s epoch.

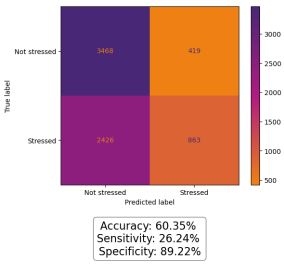


Accuracy: 62.5%
Sensitivity: 0.0%
Specificity: 100.0%

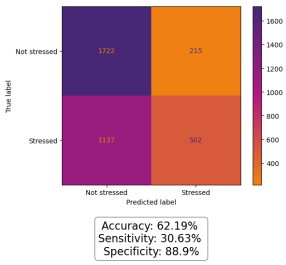
(d) SS-labels with 5s epoch.

Figure J.3: Delta-band data classified with SVM with both label types and two different epoch lengths

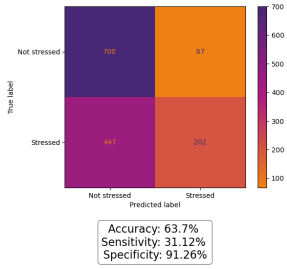
RF



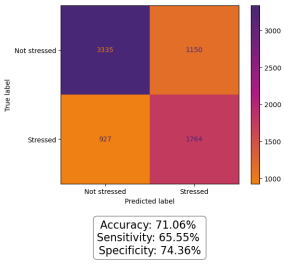
(a) STAI-Y-labels with 1s epoch.



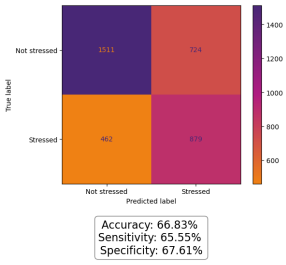
(b) STAI-Y-labels with 2s epoch.



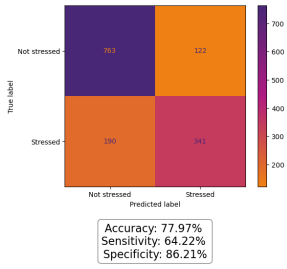
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

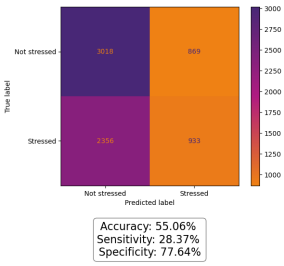


(e) SS-labels with 2s epoch.

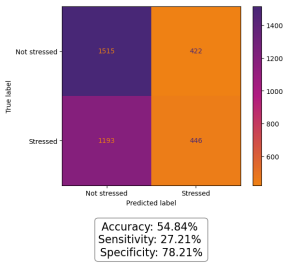


(f) SS-labels with 5s epoch.

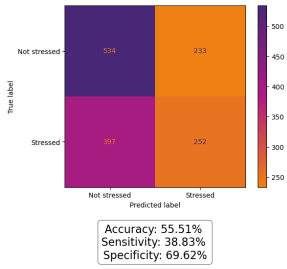
Figure J.4: Raw data classified with RF with both label types and three different epoch lengths



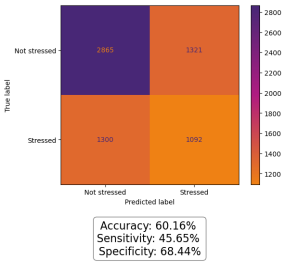
(a) STAI-Y-labels with 1s epoch.



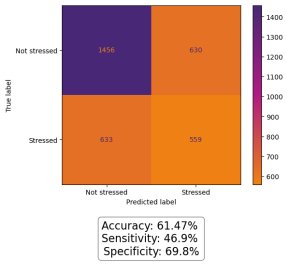
(b) STAI-Y-labels with 2s epoch.



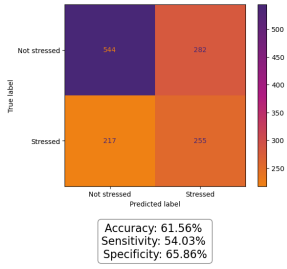
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

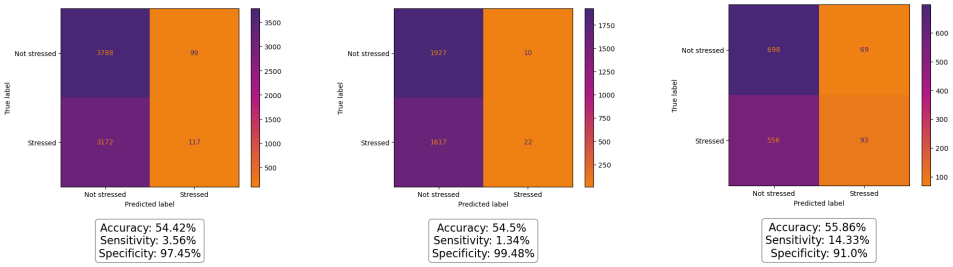


(e) SS-labels with 2s epoch.

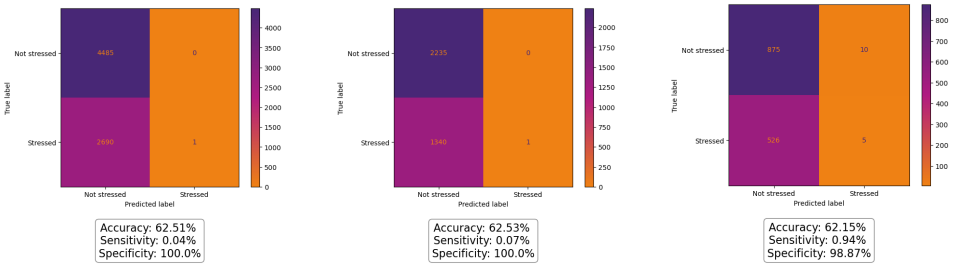


(f) SS-labels with 5s epoch.

Figure J.5: SSP data classified with RF with both label types and three different epoch lengths



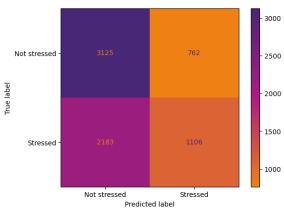
(a) STAI-Y-labels with 1s epoch. **(b)** STAI-Y-labels with 2s epoch. **(c)** STAI-Y-labels with 5s epoch.



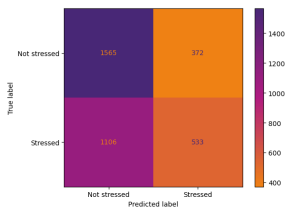
(d) SS-labels with 1s epoch. **(e)** SS-labels with 2s epoch. **(f)** SS-labels with 5s epoch.

Figure J.6: Delta-band data classified with RF with both label types and three different epoch lengths

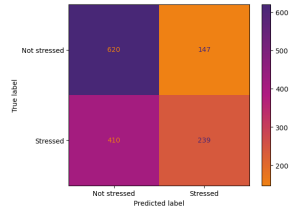
KNN



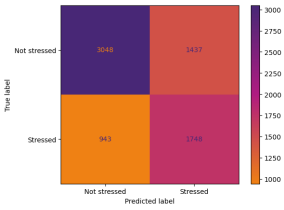
(a) STAI-Y-labels with 1s epoch.



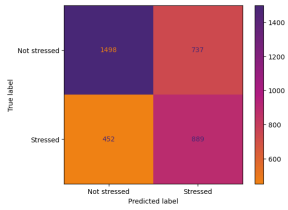
(b) STAI-Y-labels with 2s epoch.



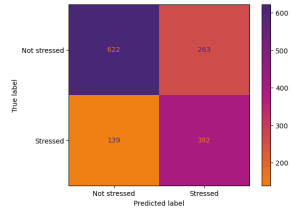
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

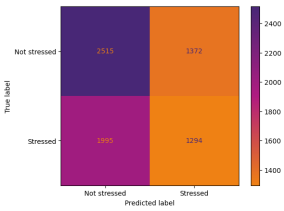


(e) SS-labels with 2s epoch.

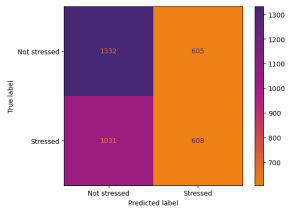


(f) SS-labels with 5s epoch.

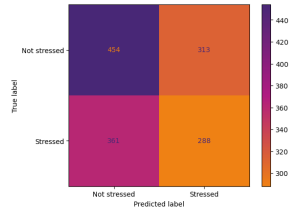
Figure J.7: Raw data classified with KNN with both label types and three different epoch lengths



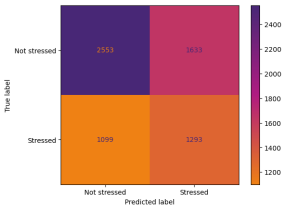
(a) STAI-Y-labels with 1s epoch.



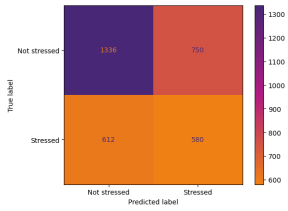
(b) STAI-Y-labels with 2s epoch.



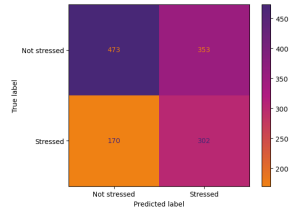
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

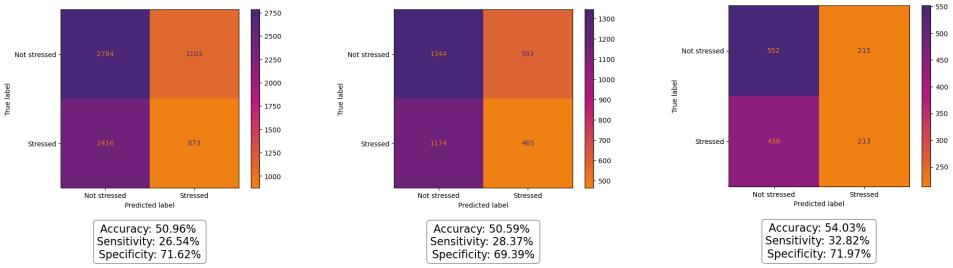


(e) SS-labels with 2s epoch.

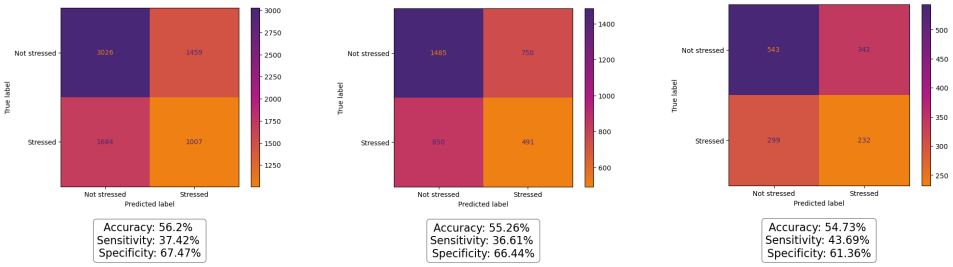


(f) SS-labels with 5s epoch.

Figure J.8: SSP data classified with KNN with both label types and three different epoch lengths



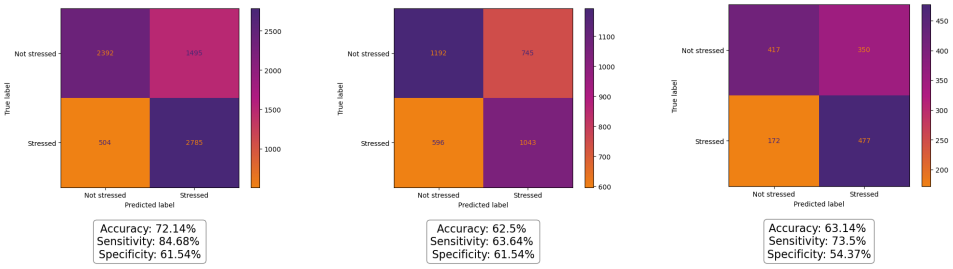
(a) STAI-Y-labels with 1s epoch. **(b)** STAI-Y-labels with 2s epoch. **(c)** STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch. **(e)** SS-labels with 2s epoch. **(f)** SS-labels with 5s epoch.

Figure J.9: Delta-band data classified with KNN with both label types and three different epoch lengths

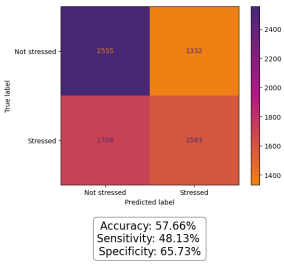
K EEGNet



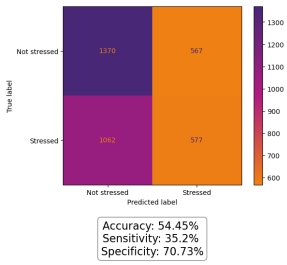
(a) STAI-Y-labels with 1s epoch. **(b)** STAI-Y-labels with 2s epoch. **(c)** STAI-Y-labels with 5s epoch.

(d) SS-labels with 1s epoch. **(e)** SS-labels with 2s epoch. **(f)** SS-labels with 5s epoch.

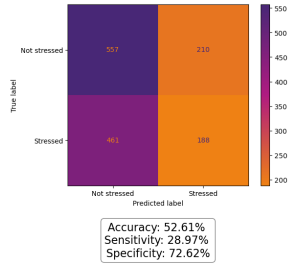
Figure K.1: Raw data classified with EEGNet with both label types and three different epoch lengths



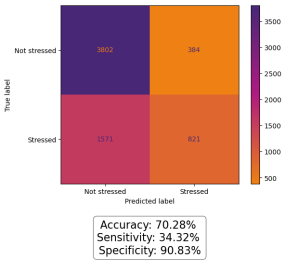
(a) STAI-Y-labels with 1s epoch.



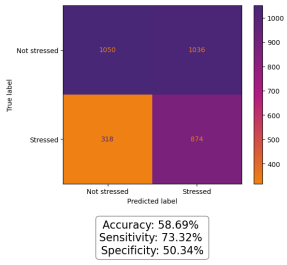
(b) STAI-Y-labels with 2s epoch.



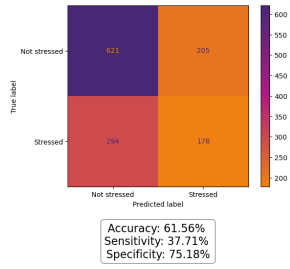
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.

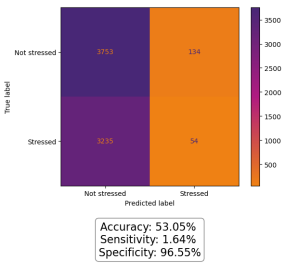


(e) SS-labels with 2s epoch.

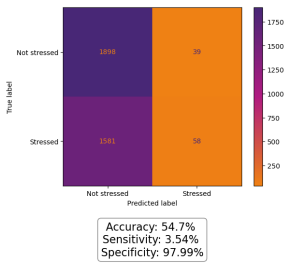


(f) SS-labels with 5s epoch.

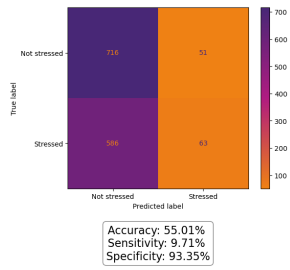
Figure K.2: SSP data classified with EEGNet with both label types and three different epoch lengths



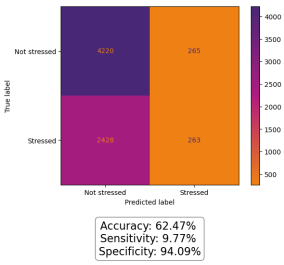
(a) STAI-Y-labels with 1s epoch.



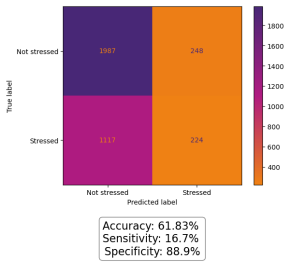
(b) STAI-Y-labels with 2s epoch.



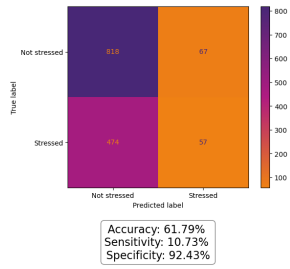
(c) STAI-Y-labels with 5s epoch.



(d) SS-labels with 1s epoch.



(e) SS-labels with 2s epoch.



(f) SS-labels with 5s epoch.

Figure K.3: Delta-band data classified with EEGNet with both label types and three different epoch lengths

L Consent form

Department of Engineering Cybernetics

DATA ACQUISITION CONSENT FORM

You are being invited to participate in a research study, which the Norwegian Center for Research Data (NSD) has reviewed and approved for conduction by the investigators named here. This form is designed to provide you - as a human subject - with information about this study. The investigator or his/her representative will describe this study to you and answer any of your questions. You are entitled to a copy of this form. If you have any questions or complaints about the informed consent process of this research study or your rights as a subject, please contact the PI or Co-PI [redacted]

Project Title: FlexEEG in Mental Health

Principal Investigators: Marta Molinas

Co-investigator: Andres Soler & Mohit Kumar

Thank you for agreeing to participate in this research project. This study involves research aimed at detecting the presence of psychological stress in the human body based on the analysis of EEG and PCG signals. You will participate in two separate data collection sessions. The first session will take place in the exam period of nov-dec 2022, and the second will take place after the holidays, early 2023. Before each session we will ask you to answer a self-evaluation questionnaire called 'State-Trait Anxiety Inventory'. This questionnaire will be used to determine whether you are stressed or not. During both sessions, you will be recorded twice: one five-minute period with no stressor, and one five-minute period with an Arithmetic stressor. You will be asked to rate your stress level on a scale from 1-10 after each recording. The Arithmetic stressor consists of different arithmetic statements presented on a screen. Your task will be to calculate each task in your head and click "T" on the keyboard if the statement is True, and "F" if it is False. This task is supposed to induce stress so please keep this in mind. Each session will last about 30 minutes. 10 of these minutes are for recording of EEG and PCG signals using Mentalab EEG and EkoDuo stethoscope. We will clean the areas of the scalp where the electrodes are placed with isopropyl alcohol. Electrode cap gel will be applied to the areas, but it is easily washed out with water and shampoo.

Participation in this study will take approximately 60 minutes of your time. We warn that the set-up of the EEG cap can lead to some discomfort, and the tasks you are given will (hopefully) induce some stress response. Your participation in this study is completely voluntary. Should you decide to discontinue participation or decline to answer any specific part of the study, you may do so without penalty.

Your participation in this study may help you understand the manifestations of stress on EEG signals. We are not asking you to place your name anywhere on the experimental booklet, so your participation is anonymous. None of your answers can be directly traced back to you. Should you have any further questions, please feel free to contact the study's principal investigator or co-PI, Marta Molinas and Andres Soler at the Department of Engineering Cybernetics. Her office is at Elektro D+B2 room D244, her phone number is [redacted], and her e-mail address is [redacted]

By signing below, I confirm that:

- I give my consent to participate in the research study entitled “FlexEEG in Mental Health”.
- I hereby confirm that I have read the above information and have been informed about the content and purpose of the research.
- I fully understand that I may withdraw from this research project at any time without prejudice or effect on my standing with NTNU.
- I also understand that I am free to ask questions about techniques or procedures that will be undertaken.
- I give my consent for the collection and use of all data of the research “EEG and PCG in mental health” for use in research and teaching purposes.
- I give my consent to use my data for scientific purposes, its documentation and publications (including any exhibitions and further publications)
- I hereby declare that I am currently not diagnosed by a with any heart disease, or neurological disease
- I am also not on any medications affecting heart rate and/or brain wave function
- I hereby declare that I am not officially diagnosed with any mental illness

Date and place: _____ and _____

Participant’s signature: _____

First and last name: _____

Date of Birth and current Age: _____and _____

I hereby certify that I have given an explanation to the above individual of the contemplated study and its risks and potential complications.

29/11/2022

_____ _____

Principal Investigator’s signature Date

M State Trait Anxiety Inventory for Adults

State-Trait Anxiety Inventory for Adults

Self-Evaluation Questionnaire

STAI Form Y-1 and Form Y-2

Developed by Charles D. Spielberger
in collaboration with R.L. Gorsuch, R. Lushene, P.R. Vagg, and G.A. Jacobs

Published by Mind Garden
1690 Woodside Road Suite 202, Redwood City, CA 94061 USA 650-261-3500
www.mindgarden.com

SELF-EVALUATION QUESTIONNAIRE STAI Form Y-1

Please provide the following information:

Name_____Date_____S_____

Age_____ Gender (*Circle*) **M** **F** T_____

DIRECTIONS:

A number of statements which people have used to describe themselves are given below. Read each statement and then circle the appropriate number to the right of the statement to indicate how you feel *right* now, that is, *at this moment*. There are no right or wrong answers. Do not spend too much time on any one statement but give the answer which seems to describe your present feelings best.

	NOT AT ALL	SOMEWHAT	MODERATELY SO	VERY MUCH SO
1. I feel calm	1	2	3	4
2. I feel secure	1	2	3	4
3. I am tense	1	2	3	4
4. I feel strained	1	2	3	4
5. I feel at ease	1	2	3	4
6. I feel upset	1	2	3	4
7. I am presently worrying over possible misfortunes	1	2	3	4
8. I feel satisfied	1	2	3	4
9. I feel frightened	1	2	3	4
10. I feel comfortable	1	2	3	4
11. I feel self-confident	1	2	3	4
12. I feel nervous	1	2	3	4
13. I am jittery	1	2	3	4
14. I feel indecisive	1	2	3	4
15. I am relaxed	1	2	3	4
16. I feel content	1	2	3	4
17. I am worried	1	2	3	4
18. I feel confused	1	2	3	4
19. I feel steady	1	2	3	4
20. I feel pleasant	1	2	3	4

STAIP-AD Test Form Y
www.mindgarden.com

SELF-EVALUATION QUESTIONNAIRE

STAI Form Y-2

Name_____Date_____

DIRECTIONS

A number of statements which people have used to describe themselves are given below. Read each statement and then circle the appropriate number to the right of the statement to indicate how you *generally* feel. There are no right or wrong answers. Do not spend too much time on any one statement but give the answer which seems to describe how you generally feel.

	ALMOST NEVER	SOMETIMES	OFTEN	ALMOST ALWAYS
21. I feel pleasant	1	2	3	4
22. I feel nervous and restless	1	2	3	4
23. I feel satisfied with myself	1	2	3	4
24. I wish I could be as happy as others seem to be	1	2	3	4
25. I feel like a failure	1	2	3	4
26. I feel rested	1	2	3	4
27. I am "calm, cool, and collected"	1	2	3	4
28. I feel that difficulties are piling up so that I cannot overcome them	1	2	3	4
29. I worry too much over something that really doesn't matter	1	2	3	4
30. I am happy	1	2	3	4
31. I have disturbing thoughts	1	2	3	4
32. I lack self-confidence	1	2	3	4
33. I feel secure	1	2	3	4
34. I make decisions easily	1	2	3	4
35. I feel inadequate	1	2	3	4
36. I am content	1	2	3	4
37. Some unimportant thought runs through my mind and bothers me	1	2	3	4
38. I take disappointments so keenly that I can't put them out of my mind	1	2	3	4
39. I am a steady person	1	2	3	4
40. I get in a state of tension or turmoil as I think over my recent concerns and interests	1	2	3	4

STAIP-AD Test Form Y
www.mindgarden.com

State-Trait Anxiety Inventory for Adults Scoring Key (Form Y-1, Y-2)

Developed by Charles D. Spielberger in collaboration with R.L. Gorsuch, R. Lushene, P.R. Vagg, and G.A. Jacobs

To use this stencil, fold this sheet in half and line up with the appropriate test side, either Form Y-1 or Form Y-2. Simply total the scoring **weights** shown on the stencil for each response category. For example, for question # 1, if the respondent marked 3, then the **weight** would be **2**. Refer to the manual for appropriate normative data.

Form Y-1	NOT AT ALL	SOMEWHAT	MODERATELY SO	VERY MUCH SO
1.	4	3	2	1
2.	4	3	2	1
3.	1	2	3	4
4.	1	2	3	4
5.	4	3	2	1
6.	1	2	3	4
7.	1	2	3	4
8.	4	3	2	1
9.	1	2	3	4
10.	4	3	2	1
11.	4	3	2	1
12.	1	2	3	4
13.	1	2	3	4
14.	1	2	3	4
15.	4	3	2	1
16.	4	3	2	1
17.	1	2	3	4
18.	1	2	3	4
19.	4	3	2	1
20.	4	3	2	1

Form Y-2	ALMOST NEVER	SOMETIMES	OFTEN	ALMOST ALWAYS
21.	4	3	2	1
22.	1	2	3	4
23.	4	3	2	1
24.	1	2	3	4
25.	1	2	3	4
26.	4	3	2	1
27.	4	3	2	1
28.	1	2	3	4
29.	1	2	3	4
30.	4	3	2	1
31.	1	2	3	4
32.	1	2	3	4
33.	4	3	2	1
34.	4	3	2	1
35.	1	2	3	4
36.	4	3	2	1
37.	1	2	3	4
38.	1	2	3	4
39.	4	3	2	1
40.	1	2	3	4

Published by Mind Garden, Inc., 1690 Woodside Rd, Suite 202, Redwood City, CA 94061