University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering

Bachelor's thesis

Design of movement detector of measured EEG data

Pilsen 2022 Josef Yassin Saleh

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Josef Yassin SALEH**
Osobní číslo:	**A18B0307P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Návrh detektoru pohybu z naměřených EEG dat**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte literaturu týkající se detekce pohybu z naměřených EEG dat a dále prostudujte funkci rehabilitačního robota dostupného na KIV.
2. Navrhněte scénář pro měření EEG dat s využitím zmíněného rehabilitačního robota.
3. Naměřte EEG záznamy u dostatečného počtu osob (5-10), analyzujte naměřená data a zvolte vhodné charakteristiky EEG signálu, na základě kterých lze detekovat pohyb.
4. Na základě analýzy z předchozího bodu implementujte detektor pohybu z EEG dat, jehož výstupem bude informace, zda došlo k pohybu končetiny či nikoliv.
5. Implementaci ověřte na dostupném rehabilitačním robotovi s využitím EEG zesilovače BV-VAMP a zhodnoťte dosažené výsledky.

Rozsah bakalářské práce:	**doporuč. 30 s. původního textu**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce:	**Ing. Pavel Mautner, Ph.D.** Katedra informatiky a výpočetní techniky
Datum zadání bakalářské práce:	**4. října 2021**
Termín odevzdání bakalářské práce:	**5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D. děkan	**Doc. Ing. Přemysl Brada, MSc., Ph.D.** vedoucí katedry

V Plzni dne 14. října 2021

Declaration

I hereby declare that this bachelor's thesis is completely my own work and that I used only the cited sources.

Pilsen, 5th May 2022

Josef Yassin Saleh

Abstract

Uses of BCIs are very versatile and their possibilities are many. One of the possible uses is to detect movement in EEG data.
Movement detection in EEG data is usually done by extracting feature vectors using *Spatial filters*. They are then passed to classifiers, that attempt to classify using gained feature vectors. Goal of this thesis is to choose suitable *Spatial filter* to gain feature vectors that were passed to classifier to detect movement in EEG data.
For this thesis *Spatial filter CSP* was chosen that passed it's feature vectors to *LDA* and *SVM*. Experiments have shown that classifiers classified well on average but there were some exceptions that were probably caused by subject winking, which resulted in loss of information about movement in EEG data.

Abstrakt

Použití BCI (Brain Computer Interface) je velmi robustní a jejich možnosti jsou velmi různorodé. Jedna z možností je detekce pohybu v EEG datech.
Detekce pohybu v EEG datech se většinou provádí pomocí tzv. *Spatial filters* (volně přeloženo jako prostorové filtry), které jsou schopny vytvořit příznakové vektory, pomocí kterých je možné klasifikovat. Úkolem této práce bylo vybrat vhodné metody pro extrakci příznakových vektorů a poté je klasifikovat a pokusit se nalézt pohyb.
Pro extrakci příznakových vektorů, byla použita technika *CSP* a pro klasifikaci *SVM* a *LDA*. Z výsledků bylo možné usoudit, že klasifikace probíhala vcelku dobře, až na vyjímky, které mohou mít různé důvody, například ztráta informace kvůli mrknutí měřené osoby.

Acknowledgement

I would like to thank my supervisor of my Bachelor's thesis Ph.D Ing. Pavel Mautner for his guidance, help and patience with me. I would also like to thank everyone who participated in EEG measuring, providing data that allowed to make this thesis possible.

Contents

1 Introduction

Artificial Intelligence (from now on as AI) is used for many things such a *Natural language processing*, *Image processing* and so on...
AI in general is used to make work easier and manageable in today's day and age. Instead of letting someone process enormous amount of data, it is more plausible to let a computer do it. AI can also be used in medical sciences, as medical data are very robust and handling them would be very difficult for a single person or even a group of people.
One of the uses for AI is so called *BCI*, which is *Brain-Computer Interface*. Using these BCIs, communication between brain and machines is possible. It is sometimes used as a tool for rehabilitation for paralyzed and disabled[30]. For example creating a movement of prosthetic limbs, restoration of movement for pacients after stroke or possible prediction of seizures and many more.
Goal of this thesis is to attempt to utilize AI to detect movement in given set of EEG data using chosen method and get the best results. Part of this work was measuring EEG of people while they were attempting movement of their hand on the exercise robot, on which measured subjects were attempting to move a lever with their arms in given trajectory and concentrate on the arm movement. While doing so, their brain activity was being measured using EEG and recorded. Recorded data was be processed and then used as a base for movement detection.
Structure of the text is as follows: brief explanation on how human brain works. Then what is EEG, different types of methods for measuring EEG and what is EEG used for. Following chapter is a short description of certain types of AI, *Unsupervised* and *Supervised learning* to be precise, their main differences and importance for this thesis. Next chapter is introduction to BCIs and approaches of using AI in finding information, that is needed for classification of EEG data. Afterwards, scenario for measuring EEG is described, this chapter also contains description of actions subjects were required to perform and components needed for measuring. Following this chapter is analysis of the given problem with finding movement in EEG data. This chapter is an overview of what is goal of this thesis and what approaches were chosen, following that, the practical component of the thesis is written out. This chapter contains explanation of library components, their parameters, how to use library components and why the parameters have been chosen.

2 Theoretical analysis

This chapter is going to describe necessary components for creating a program that detects movement in EEG data.

- EEG
- BCI
- Classifiers
- EEG measurement

2.1 EEG

2.1.1 What is EEG

EEG stands for *Electroencephalography*, which is a method of measuring of electrical activity in the human brain. EEG can be used for diagnosis of the brain to detect some of the neurological diseases such as epilepsy, brain tumours, head injuries, sleep disorders, dementia but also can be used to monitor depth of anesthesia during surgery.
Ascribed procedure for EEG is to put small disks called *electrodes* on the surface of the scalp. Each *electrode* is connected to amplifier and EEG recording device. Signals gathered from brain are converted to wavy forms and displayed on a computer.
Electrodes detect electrical activity of the brain cells and then they pass the information onto amplifier. There can be multiple electrodes at the same time which is called *multichannel* EEG recordings. There are 2 types of EEGs, *intracranial* and *scalp*. For *scalp* EEG, electrodes are put on the scalp. On the other hand, *intracranial* EEG uses special electrodes that are implanted in brain and the EEG is measured from the cortical surface.
There is importance in properly placing the electrodes because each brain lobe processes different types of activities.
One of the approaches is the so-called *10-20 electrode system*. "10" and "20" represent distances between neighbouring electrodes that are either 10 or 20% of the total front-back or right-left distance of the skull. The positions are determined by two points: *nasion*, which is the point between the forehead and the nose on the level with the eyes. Other point is *inion*, which is

the bony protrusion at the base of the skull on the midline at the back of the head. Each location of the electrode uses a letter as a marking of lobe and hemisphere location. There are 5 locations to mark - *Frontal*, *Temporal*, *Central*, *Parietal* and *Occipital* lobes. These are shortened to their first letter, meaning *Frontal* lobe is shortened to F, *Temporal* lobe is shortened to T and so on...

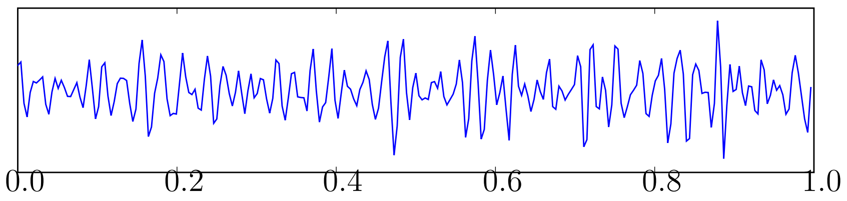

Figure 2.1: Example of EEG, image available here

EEG is a representation of difference between two voltages at two electrodes, that means EEG machine can be set up in several ways for EEG readings. The way electrodes are placed is called a *montage*.

Types of montages

- Bipolar montage
- Referential montage
- Average reference montage

Bipolar montage

For this montage, a channel is represented by a pair of adjacent electrodes. For example "F3-C3" would represent voltage difference between electrode F3 and C3.

Referential montage

Electrodes in this montage use as a differential electrode one designate electrode.

Average reference montage

The outputs of all of the amplifiers are summed and averaged and this average is used as a reference [30].

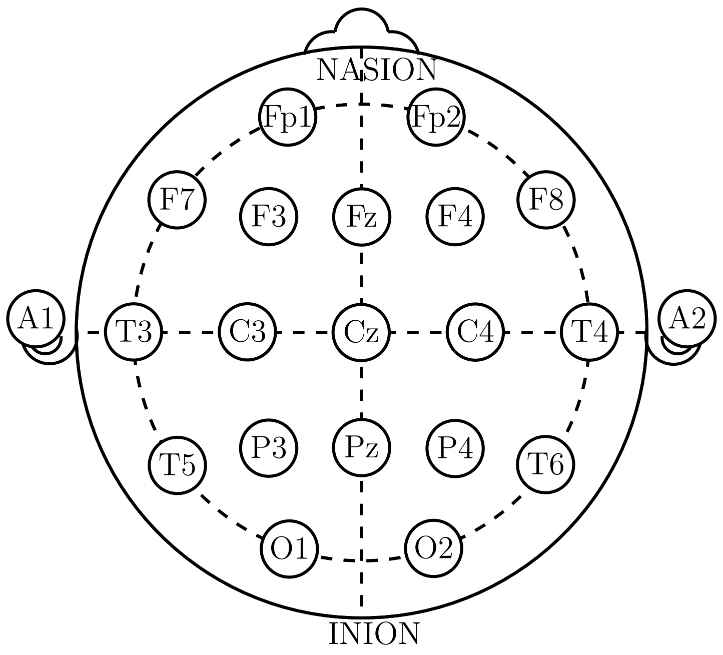

Figure 2.2: Example of montage, image available here

2.2 Classifiers

By definition of Merriam-Webster dictionary *classification* means
Systematic arrangement in groups or categories according to established criteria[2].
Classifying in AI is not different. Classification in AI works on same principle as there are data classification classes AI is classifying into, based on similarity or patterns[20].
There are different types of AI, we will look *supervised* and *unsupervised* learning of AI.

2.2.1 Unsupervised learning

Unsupervised learning AI has to find similarities in data sets and adjusts the parameters from scratch. These approaches can be used on unlabeled data to find any similarities[4].

Types of unsupervised learning

Unsupervised learning can be divided into 3 techniques that are widely used:

- Clustering
- Dimension reduction

- Association

Clustering

Clustering is a data mining technique that allows AI to take data and divide them into groups based on similarities. Clustering algorithms find patterns and similarities and puts them into given amount of groups. There are few types of clustering techniques, some of them being:

- Exclusive
- Hierarchical

Exclusive clustering Exclusive clustering works with assumption that each piece of data can be in only one cluster. Example of that is algorithm *K-means* algorithm.

Hierarchical clustering This approach can be divided into 2 groups and those are *agglomerative* and *divisive* algorithms. Agglomerative algorithms are considered "bottoms-up", that means each data is considered as cluster and most similar pairs are fused into one cluster and this process is repeated until there is one cluster left. Divisive algorithms are opposite of that and are considered "top-down". Divisive algorithms work in an opposite way, algorithm starts with one cluster and divides data into clusters.

Dimension reduction

This approach is used for data, that carry too many informations. This particular technique reduces the amount of information while preserving important factors that AI uses.

Association

Association rules create correlation between variables of given dataset[4].

2.2.2 Supervised learning

Supervised learning is a type of AI that requires a training set. This training set is used to help to adjust AI's parameters more accurately. This data set usually contains data, that is already divided into classification classes, which helps AI to adjust parameters.

Examples of supervised learning

K-Nearest Neighbours

This algorithm places training data to subsets and then takes incoming data and fits them to nearest subset. Assumption for this algorithm is that all data with similar features are in the same subset or near the created subset[3]. Let us assume we have a feature vector $X = x_1, x_2, \ldots, x_n$ and we're going to classify into class with member that has the least amount of differences. Technically this could work but in case our class is build up of anomalies or objects impaired with noise this approach is going to fail. For that reason the K in the name which represents size of a subset. This subset is created from K class members with the least amount of differences. Classification is based on so called *voting*, which means object is classified into class with most amount of representation of class members in the subset.

Classification can be also mapped into N-dimensional space where similarity has to be measured using metrics. That means one of the possible metrics is *Eucledian* distance, which calculates distance between 2 points in N-dimensional space. *Eucledian* distance between point $\mathbf{x} = x_1, x_2, \ldots, x_n$ and $\mathbf{y} = y_1, y_2, \ldots, y_n$ is:

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\Sigma_{i=1}^{n} (x_i - y_i)^2}$$

[25]

Naive Bayes

In this approach AI is making use of class conditional independence of Bayes Theorem. In application probability of features are not dependent on each other. That means that each feature has equivalent value to the result[3]. *Naive Bayes* classifier uses features that are available to predict into which class new object belongs. Let us have a feature vector $X = x_1, x_2, \ldots, x_n$ and a labeled training set, where each object is assigned one of these feature vectors X. Then for n of these classes applies:

$$P(c_i|\mathbf{x}) = \frac{P(\mathbf{x}|c_i)P(c_i)}{P(\mathbf{x})}$$

where $\mathbf{x}$ is the vector describing object that we want to classify and c_i is label of the i-th class. This is how we calculate the probability of object being classified into i-th class based on it's feature vector. In order to classify an object, we need to multiply probabilities (gain product) that attribute would be assigned to class, which would give us:

$$P(\mathbf{x}|c_j) = \prod_{i=1}^{n} P(x_i|c_j)$$

where c_j is a representation of a class j. After that we multiply the product of probabilities of an object $\mathbf{x}$. That gives us:

$$Y_j = P(c_j) \prod_{i=1}^{n} P(x_i|c_j)$$

where Y_j is a probability an object $\mathbf{x}$ is in class j and of course by choosing the highest probability is how classified using *Naive Bayes*[24].

LDA

LDA (Linear Discriminant Analysis) also known as Fisher's linear discriminant is a special type of supervised learning AI as it uses dimensional reduction that is used in unsupervised learning. This algorithm also creates hyperplanes out of N k-dimensional vectors $\boldsymbol{u}$ into a hyperplanes that divide the space and defines the classes, where N is the amount of classes and k is the dimension. This algorithm assumes that each classification class has *Normal distribution* of vectors u and division of classes is done by means of the class values. When classifying a vector y, a linear combination of it's parameters are created based on the existing hyperplanes following this equation:

$$Y = u_1x_1 + u_2x_2 \cdots + u_kx_k[31] \tag{2.1}$$

SVM

Support Vector Machine is a technique that also uses hyperplanes to divide the space to classify the k-dimensional values into classes. Very useful feature of SVM is the concept of mapping to higher dimension meaning if data cannot be separated linearly in given dimension, SVM tries to separate classes linearly in higher dimension. One of the main difference between LDA and SVM is that instead of using means of the classes, like LDA does, SVM uses *support vectors.* Support vectors are values that are found on the boundary of classes. When an optimal hyperplane cannot be found SVM uses *Kernel function* that helps to map values to higher dimension[31].

This thesis will focus on trying to recognize movement from one set of data for each person. For that reason, each data set is going to be used as training and test set. Tests, whether AI was able to recognize the movement, will utilize *markers*, that are going to be explained in *measuring* chapter and their application in epoch chapters. We're going to be working with assumption that data is going to be flowing at all time in the future and AI has to immediately decide if there was movement in data and act according to it. For that reason neural networks haven't been chosen as those can be computationally demanding. This work will attempt to use *LDA* and *SVM* with *CSP* technique explained further in the thesis.

2.2.3 Spatial filters

Of course there is an issue with how to handle the EEG data properly and create feature vectors out of them. For this reason technique called *Spatial filtering* is used to bring out the features in the EEG data[27].

- PCA
- ICA
- CSP

PCA

One of the methods called *Principal Component Analysis* is a dimensionality reduction-based technique. PCA works well under the condition that the number of components is known. Despite limited options of working with data PCA allows user to work with data more quickly and with lesser memory consumption. There are few modifications but one of the simpler ways to perform this method is to calculate mean vector v and covariance matrix d x d A. When these values are computed next step is to calculate eigenvalues $\lambda_1 \dots \lambda_d$ and their eigenvectors $u_1 \dots u_d$. After that k-largest eigenvalues and their assigned eigenvectors are chosen, where k is the amount of components, rest of the values is discarded as noise. After that matrix B of dimension k x k is formed, where its columns are chosen eigenvectors then preprocess data using:

$$X' = B^T(x - u) \tag{2.2}$$

where x is the data point[27, 31].

ICA

Unlike PCA *Independent Component Analysis* searches for components that are not correlated to each other. Let's assume that vector x is result of linearly mixing a set of statistically independent sources inside the brain:

$$x = My \tag{2.3}$$

where M is the unknown mixing matrix and y represents vector of hidden independent sources. ICA attempts to find matrix W, that contains hidden resources:

$$a = Wx \tag{2.4}$$

where a is the feature vector.
Matrix W is sometimes called as *unmixing* matrix to express that it's an inverse matrix to mixing matrix. There are numerous algorithms for computing the matrix W such as *infomax* by Bell-Sejnowski[27, 31].

CSP

Unlike *ICA* and *PCA*, *Common Spatial Patterns* is a supervised method. CSP attempts to find patterns by a process in which that one variance of filtered data from one class is maximized while on the other side second class is minimized. This approach allows CSP to classify only into two classes. However, CSP can be applied recursively which means by classifying it into 2 classes that could be classified into another 2 subclasses. Further mathematical formulation will be written for two classes. We have matrix X that represents one segment of subject doing a task, where X is a N x T matrix, where N is the amount of channels and T is the number of samples in time per channel. Mathematically, CSP attempts to find M spatial filters, given by a N x M matrix W, where each column is a spatial filter that transforms input using:

$$X_{CSP}(t) = W^T x(t) \tag{2.5}$$

where x(t) is the vector of signals at the time t of all channels.
To find the filters, first are two classes covariance matrices estimated as:

$$R_c = 1/K \sum_i X_c^i (X_c^i)^T \tag{2.6}$$

where $c \in \{1, 2\}$, which represents classes and K amount of samples. Then CSP determines matrix W:

$$\Lambda_1 = W^T R_1 W \tag{2.7}$$

$$\Lambda_2 = W^T R_2 W \tag{2.8}$$

where Λ_i are diagonal matrices and $\sum_{i\ =0}^{1}\Lambda_i = I$, where I is the identity matrix. After that CSP solves the problem of generalized eigenvalue:

$$R_1 w = \lambda R_2 w \tag{2.9}$$

and generalized eigenvectors w that satisfy the equation above form the columns of W[18, 27].

This thesis will be focusing on using CSP together with LDA and SVM. After creating feature vectors from CSP, LDA and SVM will attempt to classify EEG data into 3 classes: movement of a left hand, movement of a right hand and resting phase. Reason for choosing CSP for calculating feature vectors is connected to functionality of the other methods, both as PCA and ICA are unsupervised methods[30]. Using of the fact that CSP is supervised, should precise classification accuracy.

2.3 BCI

As mentioned before, BCI stands for *Brain-Computer Interface* which is connecting computer and human brain and allowing brain to control a computer. There are several types of BCIs, some of them being:

- Invasive BCI - Neurons are stimulated or recorded inside the brain.
- Semi-invasive BCI - Recording and stimulating is happening on the surface of the brain or nerves.
- Non-invasive BCI - This type uses techniques that don't penetrate the skin or skull. [27]

2.3.1 Use of BCIs

As mentioned in introduction, BCIs have many uses, one of them being in medical field. Possible use is for example, sensory restoration. One example being *cochlear implants* for deaf people or *retinal implants*.
Another possible use is motor restoration, which focuses on moving prosthetic limbs using neural signals.
Another possible use is rehabilitation. People that have suffered stroke can use BCI to convert neural signal to movement in rehabilitative device, patient can train connecting the neural activity to movement which can speed up recovery[27].

Goal of this thesis is to create non-invasive BCI, that is capable of recognizing movement of hand in EEG data, to add to the research of possible ways to assisted recovery for patients who suffered a stroke and to help accelerate their recovery.

2.3.2 Possible approaches for EEG classification

Classification of EEG can be tricky because as mentioned before, data at hand are N-dimensional, where N is amount of channels (electrodes) in montage. Each channel gives out slightly different values. Not to mention, it is not an easy task for human to detect movement in EEG. For that reason many techniques have been developed for handling EEG data:

- ERD/ERS detection
- ERP detection
- SMR detection

ERD/ERS detection

ERD/ERS is an abbreviation for *Event-Related Desynchronization/Event-Related Synchronization.* As for the description of methods for detection, usually for this method differences of the potentials from electrodes as the event is commencing are calculated [28]. There is already a thesis on arm movement detection using ERD/ERS detection from Ing. Pavel Mochura (available here).

ERP detection

Another type of brain activity that is very useful to track in EEG is after some sort of stimuli. One of the examples would be flashing some kind of light in front of the subject. This is also known as *ERP* which is *Event Related Potential.* One of the examples would be *P300* (or *P3*) which is a positive signal that occurs approximately 300milliseconds after a some sort of stimulus rare or unpredictable stimulus such as flashing bar. Another type of ERP is *N100* (or *N1*) which is negative going potential that is observed approximately 100 milliseconds after an unpredictable stimulus, usually it is followed by a positive wave (*P200* signal) and many more. . . [27].

SMR detection

This thesis focuses on detecting SMR(Sensory Motor Rhythm) also known as *Mu Rhythm* or also *Mu wave.* These waves are possible to detect around 8-13Hz [22, 23, 33] bandpass and around 13-30Hz [22, 23, 33]) bandpass. One definition describes SMR as specific type of brain wave activity correlated with immobility. SMR amplitude decreases with movement and then again increases when movement dims down[19]. Meaning to detect movement, we need to detect a dip of amplitude in brain activity with sudden spike. It has been tested experimentally that vertical movement has been shown in the bandpass of 13-30Hz and horizontal movement in 8-13 bandpass[33]. For this thesis 13-30Hz bandpass hasn't been taken into consideration as movement on the robot (further discussed in the chapter EEG measurement) is predominantly horizontal[27].

There are other methods but I have chosen to list these methods as they are known and used in practice. As to why I have chosen detection of SMR is because thesis on ERD/ERS movement detection has been done and ERP doesn't seem to fit the future prospect of this work very well. During experiments(more about it in chapter EEG measurement) there's a signal to move and so there will be spike of brain activity in data, issue comes when there isn't this kind of signal. SMR detection won't require this kind of signals for nothing other than training set.

2.4 EEG measurement

EEG was measured using custom-created scenario. This scenario had 4 stages with each lasting exactly 10 minutes. Each arm has designated stage with vibrators and one without. These vibrators are irrelevant for this thesis and will be used in different research further on, however data collected will be used nonetheless as vibrators aren't expected to have significant impact on the observed phenomena. Scenario was divided into 2 phases, resting phase and active phase. People that were measured(called patients from now on for EEG measuring), were measured in KIV neuroinformatic laboratory, each patient was measured while sitting and wore a cap with Ag/AgCl electrodes placed using *10-20* system, used electrodes were at locations: Fp1, F3, F4, C3, C4, P3, P4, F7, F8, T3, T4, T5, T6, Cz, Fz, Pz, 17, 18, where 17 is the *reference* and 18 is the *ground*. Reference is designated electrode, in this case electrode that scans brain activity from forehead, and is used as reference against other electrodes as mentioned in chapter EEG. Ground is used for subtracting same voltage from every electrode including referential electrode, doing this will give us differential voltage of electrodes[32].
Patients have moved a lever of an exercise robot that had pre-programmed trajectory of a ring. Patients differentiated resting phase from active phase by red diode that was lit indicating resting phase was active, otherwise active phase was ongoing. During active phase, patients were instructed to move the lever in given direction. The trajectory was visible on the screen and it was in a shape of a ring that was approximately 3 pixels wide. Patient had to follow the trajectory of the ring in their best effort. To ensure correct trajectory was being followed, 2 arrows were shown, one was showing direction patient had to move the lever. Other one showed what direction patient was moving the lever and with how much force. Force was indicated with length of an arrow. In case correct trajectory wasn't followed robot would block the movement of the lever until patient moved lever in the correct direction. Lever could be moved with lightly applying pressure using fingers when patient stuck to correct directions, while on the other hand, lever couldn't be moved so easily if pushed in completely different direction than it is shown on the screen.
This measuring took usually not longer than a hour and subjects were thoroughly acquainted with how the measurement is going to look like and then gave consent to measurement. I hereby want to thank all the participants with their patience and willingness that helped me to make this thesis possible.

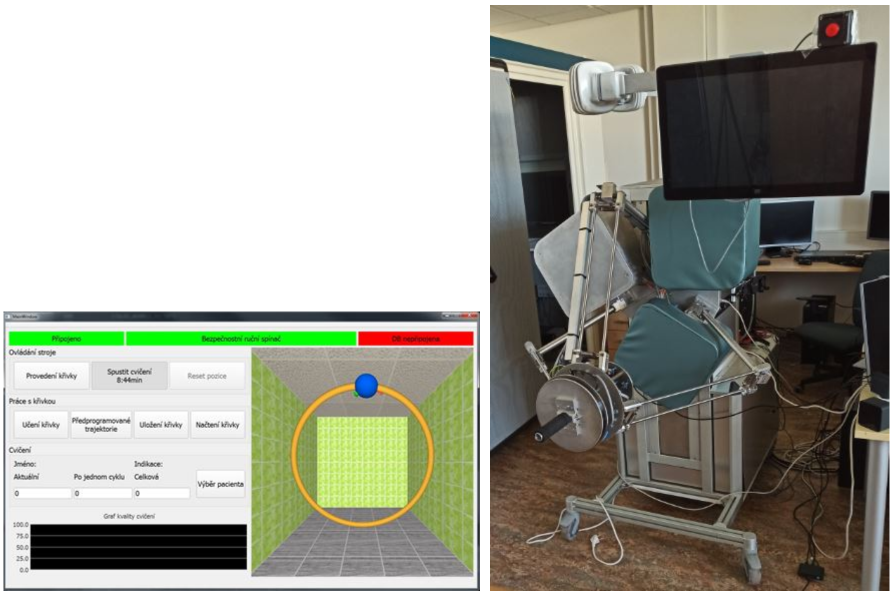

Figure 2.3: Image of the screen with drawn trajectory and a robot that was used to gather data
Picture taken from the work has measured EEG using the same device here

2.4.1 Hardware used

BrainAmp DC amplifier from Brain Products was used. Another used part was a microcontroller STM324F429I-DISCO board. To this board was connected an EKG/EMG shield from Olimex, whose output is an analog signal, which is passed to an A/D converter of the above mentioned STM board. An EEG cap fitted with Ag/AgCl electrodes according to a 10-20 system was used. At the end BrainVision Recorder software was used.

2.4.2 BrainVision

The BrainVision recorder software was used to record EEG data, which saved each EEG record in three individual, dependent files. There are the *.eeg, *.vhdr and *.vmrk file. The *.vhdr file contains all important measurement settings. This means, for example, the number of channels and their designation, the sampling interval (always 1 KHz was used) or the data type in which the EEG data is saved (e. g. INT 16). It is basically a header file, so it refers to the other two files. The *.vmrk file stores all recorded markers to search for epochs. In our case there are four types of markers,which are marked with numbers 1, 2, 4 and 8. Marker with number 1 is in the resting phase, similarly number 2 indicates that the marker is in the middle of the resting phase. Number 4 indicates the beginning of the active phase and at the end the marker with the number 8 indicates the end of the active phase. At last, the *.eeg file is binary and stores all EEG data from all measured chan- nels. Therefore, after only nine minutes of recording, this file size is approximately 20 MB [1]

3 Problem analysis

As mentioned before, the goal of this thesis is to design and implement movement detection in EEG data, using recording devices and pre-existing scenario. Future prospect of this thesis could be to help patients that have suffered injury or disease that limits them in their movement while their brain is still capable of functioning. This kind of detector could be used for rehabilitation of people after stroke that are incapable of properly moving their limbs. As mentioned, post-stroke patients could suffer from motoric control damage which leads to paralysis of parts of the body[5].
Sequence of actions necessary for detector to function is acquiring EEG data from patients in a calm environment, then process data, create feature vectors and then use those vectors for classification.

3.1 Data acquisition

As mentioned in chapter dedicated to measuring EEG, data were measured in neuroinformatical laboratory. This data have been saved on computer that was connected to amplifier and together with software data has been kept in proper format that could've been passed to classifier that could further process EEG data. At the time of measuring, subject was accompanied by someone who observed recorded data to ensure everything is going as planned.

3.2 Working with data

EEG data that were processed using Brain software is in divided into 3 main files that are *.eeg, *.vmrk and *.vhdr (contents of these files were described in section BrainVision). It goes without saying that all these files are necessary for the EEG data be usable. For this thesis library named *MNE* (will be covered later in section MNE) has been chosen for working with EEG data. Reason for choosing MNE is that MNE has comprehensive documentation and many tutorials on how to use MNE components unlike library named *eeglib* for example, that doesn't have sufficient documentation. MNE takes *.vhdr file and creates an object that represents *RAW* data in *.fif* format. This is divided into so called *Epochs.* MNE represents Epoch as an object of chosen time span of EEG data around a marker that has many other important information. Two important information are being used are data and labels. Data are *ndarray*, which is N-dimensional array, that has 3 dimensions. Where the dimensions represent: amount of epochs, picked channels and data in chosen time interval. Labels are 1-dimensional array that contain markers for given epochs. It goes without saying that size of first dimension of data array must be equal to the size of the label array. For further manipulation with data, it is necessary to work with these two arrays. As mentioned before data and labels are going to be divided into training set and test set in ratio of 1:4.

3.3 Feature vectors

In this thesis classifiers LDA and SVM are trained on raw data without any modification and then trained on data with CSP modification. Training data are taken and are used 4 times for classifiers 2 times for SVM and 2 times for LDA. From section about SMV, we know that there are kernels that can be used for mapping to higher dimensions. For that reason *GridSearchCV* from library *scikit-learn* is used. GridSearchCV is an object that tests given classifier and all of it's parameters on labeled data. This way we can test kernels and other parameters (later in chapter about scikit-learn SVM). With that being said raw data are passed to:

- GridSearchCV with SVM as classifier and a list of parameters
- LDA
- CSP that passes those data to GridSearchCV with SVM and a list of parameters
- CSP that passes those data to LDA

Reason for this is to test whether LDA and SVM work better with feature vectors created by CSP.

3.4 Results

Results of classification can tell us a lot about not only how accurate classifier was but also what did the classifier labeled data as. This can be useful for later analysis and possible improvements of either selection of classifier or feature vectors. Statistics are printed into file called *statistics.txt*. Metrics used to determine accuracy are *precision*, *recall*, *f1-score* and *accuracy* (all of those metrics will be explained in following chapters). Reason for choosing these metrics is to know, what was classified, accuracy would be of no use for us if classifier had high accuracy but classified only data from resting phase.

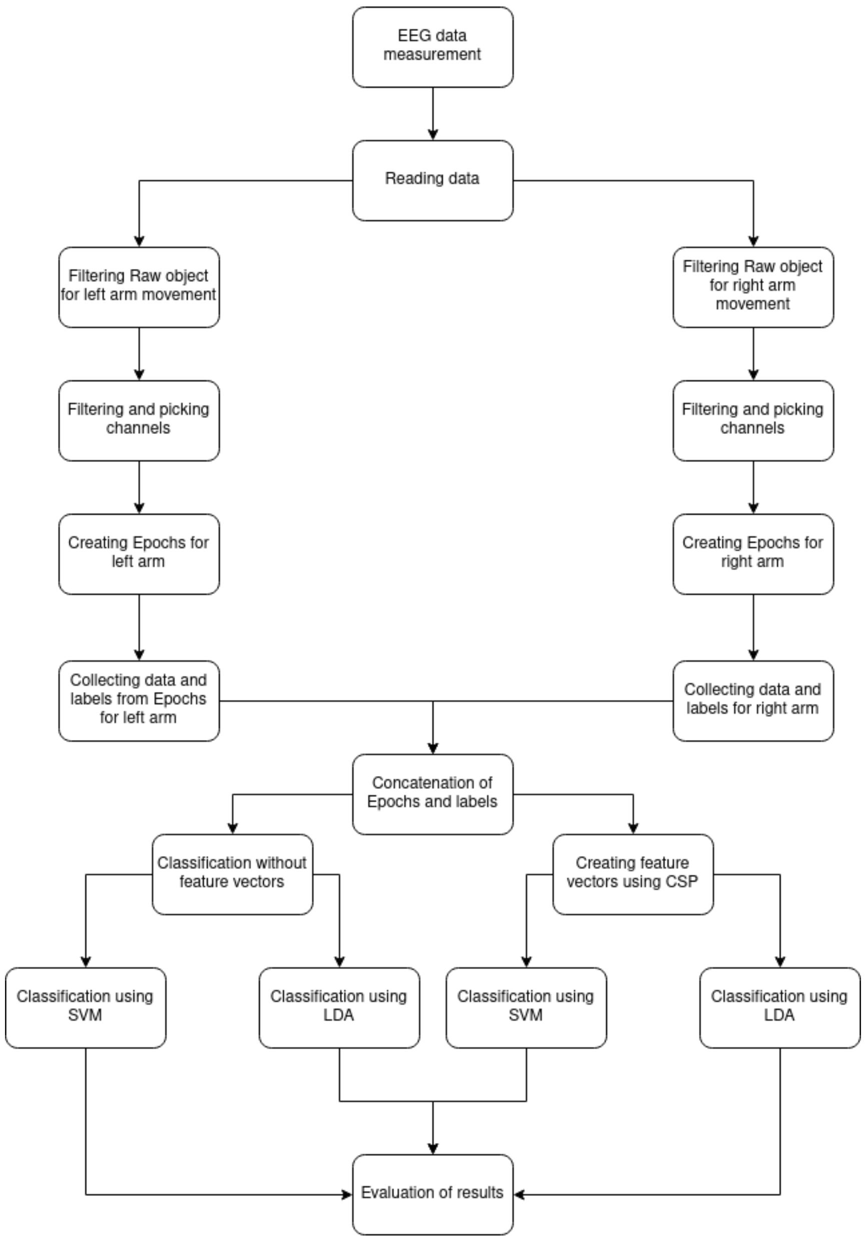
EEG data measurement
Reading data
Filtering Raw object for left arm movement
Filtering Raw object for right arm movement
Filtering and picking channels
Filtering and picking channels
Creating Epochs for left arm
Creating Epochs for right arm
Collecting data and labels from Epochs for left arm
Collecting data and labels for right arm
Concatenation of Epochs and labels
Classification without feature vectors
Creating feature vectors using CSP
Classification using SVM
Classification using LDA
Classification using SVM
Classification using LDA
Evaluation of results

Figure 3.1: Workflow diagram of this thesis

4 Implementation

In this chapter we will describe used libraries, their components that were outlined in previous chapter, then structure of program and description of implementation. Finally, we will sum up achieved results.

4.1 Libraries used

Libraries that have been used in this thesis were *MNE*, *Scikit-learn*, *NumPy* and *Matplotlib*. Description of libraries is as follows: to start with we describe Matplotlib and NumPy, then Scikit-learn and then MNE. Reason for this being that, Matplotlib library has been used when working with data initially and then the code was deleted. Despite that, various methods to plot data were used and this library is thus worth mentioning. Each of listed libraries is using NumPy library as their support pillar, this library is going to be briefly described to clarify it's contribution to this thesis. Scikit-learn library provides classifiers (for example SVM and it's kernels which are going to be explained in this chapter) and metrics that were used to classify data. Library MNE is used to handle EEG data, from loading data, creating epochs, to using CSP.

4.1.1 Matplotlib

This library was used to visualize information from EEG data and EEG data itself. Even though visualisation is not available anymore, this library played helpful role in understanding oversee data in a understandable form. Matplotlib was used to visualize EEG data and to check if data doesn't have any abnormal spikes of potentials. Another use was to display markers at designated times, when learning MNE library. Also another use was to display accuracy of classifiers from Scikit-learn library when classifying EEG data.

4.1.2 NumPy

NumPy is a library that allows most of the libraries to work with N-dimensional arrays and operations happening on them. Listed libraries are not an exception of that. For example, Matplotlib uses these N-dimensional arrays to visualize information. When user reads EEG using MNE, despite EEG being represented as an object, EEG data are represented as N-dimensional array. Scikit-learn takes N-dimensional arrays as arguments to classify.

4.1.3 Scikit-learn

Scikit-learn provides us with many AI tools, one of them being classifiers that were used in this thesis. Classifiers LDA and SVM were taken from this library and are going to be described in following chapters. Before an explanation of LDA and SVM from Scikit-learn it is necessary to go over a *Pipeline* from Scikit-learn, which is a structure that creates a sequence of data transformations and final estimator. Reason for it that is that classifiers from Scikit-learn accept data in certain format. After that, *GridSearchCV* and it's uses and it's relevance for this thesis is going to be explained. Another part of this library that was used in this thesis are *accuracy metrics*, that are going to be described.

Pipeline

As briefly explained, Pipeline is an object that represents sequence of data transformations and a final estimator. That means LDA can be used for it's ability to reduce dimensions of data and then other classifier can use those data to classify. There are 2 ways to create a Pipeline using Scikit-learn. One is using constructor *Pipeline* and as first parameter pass list of transformations where the last element has to be a final estimator. Another approach to create a Pipeline is to use method *make_pipeline*. Difference between these two approaches is that *make_pipeline* assigns name for transformations and final estimator which is further used for manipulation of their parameters. This thesis used the second approach as there wasn't much need for any special name assigned to estimator or transformations. Only necessary use of these assigned names would be when passing parameters to *GridSearch* (will be explained in further sections). Syntax to access parameters of transformation or estimator is to
name_of_estimator/transformation__parameter
Definition of a transformation, that can be section of a Pipeline, is an object that implements functions *fit* and *transform*. Similarly enough for object to be an estimator, function *fit* needs to be implemented. Purpose of these two methods is as their name suggests. Function *fit* takes data and labels and creates a model. *Transform* function transforms the data[13, 15].

LDA

Theoretical background of LDA has been lightly touched in the chapter about LDA. As mentioned in previous sections, EEG data are three dimensional NumPy arrays. Classifiers from Scikit-learn need 2-dimensional data. For that reason one of the transformations that was used was a *Vectorizer* from MNE library that takes N-dimensional NumPy array and transforms it into 2-dimensional NumPy array, that contains amount of features and other dimension represents amount of samples[6]. LDA classifier is created by calling it's constructor *LinearDiscriminantAnalysis*. As strange as it seems this constructor is called without parameters in this thesis, which means default parameters are used. Parameter that could be considered suitable for modification is *solver*. Parameter *solver* sets how LDA predicts new element. Possible options are *svd*(*Single Value Decomposition*), *lsqr*(*Least Square Roots*), and *eigen*(Eigenvalue Decomposition). Default option for *solver* parameter is *svd*.
Reason as why to pick *svd* is that this method doesn't calculate covariance matrix unlike other options. For EEG data that are robust and have a lot of features, this method should be a good choice[9].

SVM

Support Vector Machine from library Scikit-learn has a bit of problems as there are two main parameters that have to be chosen. One of them being, as foreshadowed before, *Kernel function* and the other one is C-parameter. Main issue is in this case choosing parameters, for this reason *GridSearchCV* was used (will be explained in following sections).
Creation of an object representing SVM classifier is done by calling constructor *SVC*. Here is where parameters of *Kernel function* and C-parameter come into play. Possible options for *Kernel function*(*SVC* parameter *kernel*) are *linear*, *poly*, *rbf* and *sigmoid*. Each of the *kernel*s is going to be very briefly explained in further sections but before that C-parameter is going to be briefly explained.
Scikit-learn offers other implementations for SVM which are for example *LinearSVC* and *NuSVC* for example. Reason for selection of *SVC* object instead of *LinearSVC* is rather simple and lies in selection of *kernel* parameter of *SVC*. *LinearSVC* is very similar to *SVC* with *linear kernel*, differences lie in underlying implementations. While it is true that *LinearSVC* offers more precise tuning of parameters but for simpler use of *GridSearchCV* and collecting metrics *SVC* was chosen[12].
As to why *NuSVC* wasn't chosen, this version of classifier requires a *Nu* as

a parameter, which represents amount of support vectors that are going to be used to divide classes. While it might be useful to control amount of support vectors, for this case where data are robust, letting SVM calculate amount of support vectors dynamically might be better instead of randomly trying to guess amount of support vectors[14].

C-parameter

Value of this parameter is a positive integer that represents penalty score for *SVM* classifier. Scikit-learn uses this value to refine training of the classifier model by allowing higher/lower margin hyperplanes that SVM creates[**?**]. In this thesis range of values 10^x, where $x \in \{-3, -2, -1, 0, 1, 2, 3\}$.

Kernel

SVC uses *rbf* as a default kernel option. Each kernel uses different function to map into higher dimension and those are:

- *rbf* that uses exponential function $\mathrm{e}^{-\gamma\|x - x'\|^2}$
- *linear* that uses linear function (x, x')
- *poly* that uses polynomial function $(\gamma(x, x') + r)^d$
- *sigmoid* that uses sigmoid function $tanh(\gamma(x, x') + r)$

where r is one of the parameteres possibly defined by *coef0* in *SVC*, d is parameter *degree* also in *SVC* and γ is parameter *gamma* in *SVC*. Notation (value, another_value) represents *dot product* of value and another_value[8, 17].

GridSearchCV

As was mentioned in previous section, SVM is tested on chosen parameters. To do so, *GridSearchCV* was used. It is an object that iterates over given *Pipeline* or an estimator with list of parameters inputted as a *map*. In order for something to be an estimator, implementation of estimator interface is required. Either estimator object implements *score* function or it has to be provided as a parameter *scoring*. Both *score* function and *scoring* parameter is a strategy that evaluates performance of the cross-validated model on the test set.
One of the parameters is *cv* which is *cross-validation*. Cross-validation is a technique used to prevent something called *overfitting*, which is creating a

model that can classify data on which it has been trained perfectly but can fail when it comes to classifying new data. For that reason cross-validation offers a simple solution, which is dividing training data into k-folds. One fold is chosen as a test set and the rest of the folds is used as training sets and then classifier is trained on these combinations of training/test sets. After each test is done average of all values is returned as result. *GridSearchCV* wasn't used on *LDA* classifier as there aren't parameters to test. *Cross-validation* wasn't used on *LDA* classifier since with each EEG dataset, new model is created so *overfitting* shouldn't occur [10, 11].
GridSearchCV uses cross-validation and tests each parameter it is passed of an estimator, which makes it perfect for testing which parameters to choose.

Accuracy metrics

To gather statistics from classifiers *classification_report*, *accuracy_score* and *precision_recall_fscore_support* functions were used. As is mostly conveyed in the names of the methods, metrics that were collected are: *accuracy*, *precision*, *recall* and *f-score*.
First method *classification_report* creates a string (or a dictionary) that contains already computed metrics that were mentioned, for further use.
Next method *accuracy_score* computes accuracy, which tells us how well was the classifier able to classify the data, of given estimator.
Last method is called *precision_recall_fscore_support*. It returns last three of the metrics that are *precision*, *recall* and *f-score*. *Support* from this function tells us, how many values from each class were attempted to classify .
Precision is defined as the ratio of true positive data, which are data that were correctly classified to sum of true positives and false positives, which are data that were incorrectly classified. Mathematical formula would be:

$$pr = tp/(tp + fp)$$

where *pr* is precision, *tp* is amount of true positive data and *fp* is amount of false positive data. This metric tells us how well was classifier able to classify into correct class. Recall is defined as a ratio of true positive data to sum of true positive data and false negative data. Mathematical formula is:

$$rec = tp/(tp + fn)$$

where *rec* is recall and *fn* is amount of false negative data. The meaning of this metric is to find ratio of data that was classified into a class as positive. Last metric is *f-score* that is defined as harmonic mean of precision and recall. Mathematical formula is:

$$fs = 2(prrec)/(pr + rec)$$

where *fs* is f-score, pr is precision and rec is recall. This metric is used to determine the quality of precision and recall[16, 26, 29].

4.2 MNE

This library has been used to handle EEG data and work with them, visualise data using library *Matplotlib* and manipulates with data for classifiers. As mentioned in previous chapter about Scikit-learn, *Vectorizer*, which is a transformation used to change dimensionality of EEG data is going to be briefly explained in this chapter. This library also includes a *CSP* transformation, which is a method for extracting feature vectors that is passed to classifiers. Main component for working with EEG data is *Epoch*, which is also going to be explained as to how was it used and some of it's parameters. Explanation of the the library components is going to be used to be divided into sections *Reading data*, explanation on working with *Epochs* and then will briefly touch on the topic on *CSP*, *Vectorizer*.

4.2.1 Reading data

In order to work with EEG data, it is necessary to read them first, which is done using *read_raw_brainvision* function. This function reads EEG data and saves them as the *RawBrainVision* object. This object offers information about EEG. For example the electrodes(this library uses *channels* to describe which electrodes were used) that were used to measure EEG, type of montage and of course EEG data. Parameters of function *read_raw_brainvision* that were relevant to this thesis were *vhdr_fname* and *preload*. The first parameter is a path (can be relative or absolute) to *vhdr* file (more in section BrainVision).
In order to detect *Mu wave*, EEG data were filtered by *Firwin* filter on an interval of 8-13Hz (more about *Mu wave* in section SMR). This was done using *BrainVision Raw* object method *filter*, where passed parameters were what kind of filter is going to be used (*firwin*) and in what bandpass should the EEG data be kept.
As mentioned in previous chapters, subjects were measured while using their left and right hand. For that reason both files have been loaded and saved into *Epochs* objects.

In this thesis there was an issue with setting up *lowpass* and *highpass* of EEG data, which resulted in breaking sampling rate and which resulted in wrongly set up sampling frequency. For that reason another function called *create_info* was used. This function creates an *Info* object that contains information about channels that were used, date of measurement and sampling rate, which was used to replace former *Info* attribute of *Raw* object.
Another function worth mentioning is *events_from_annotations* that extracts markers (more about markers in chapter BrainVision). This function returns at what time which marker (MNE library calls markers *events*) has appeared. It also returns a map of marker explanations mapped to their respective numeric labels. Parameter passed to this function is *Raw* object we want to extract events from. This is important in next section about *Epochs*[7, 21].

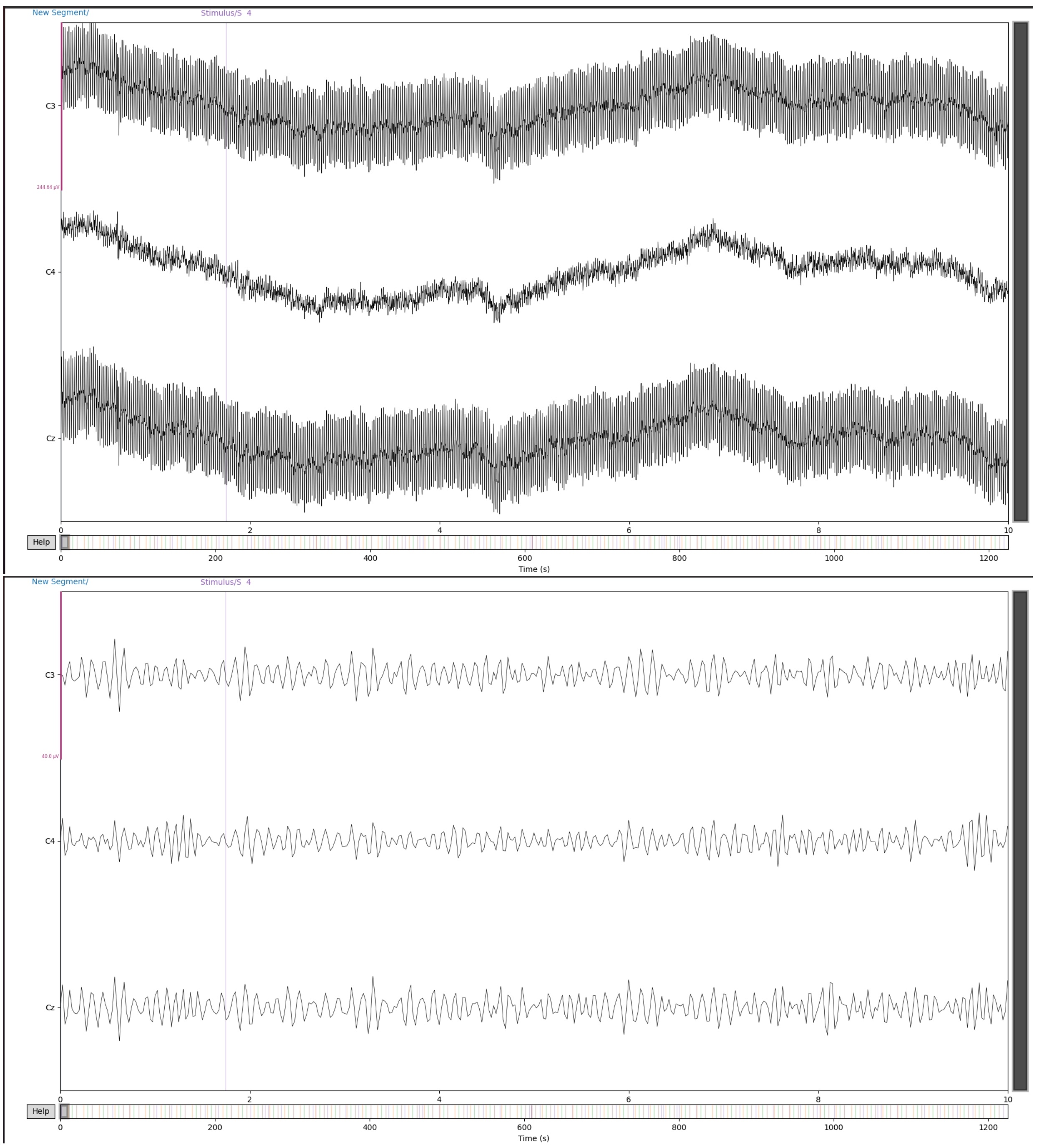

Figure 4.1: Comparison of raw data and filtered raw data

4.2.2 Epochs

Before explaining working with *Epochs* object it is necessary to explain what *Epoch* object represents. As mentioned in previous chapters, *Epochs* are clusters of data in given time intervals. There are a few options to create an *Epochs* object, two of them being a) using constructor *Epochs* or b) reading them from a file. Approach b) is not used in this thesis and is mentioned only as an example for completeness.
In order to create *Epochs* object using constructor it is compulsory to pass parameters: *Raw* object, to extract necessary informations for *Epochs*, array of events (first returned value of method
events_from_annotations). Another parameter is a map of events (other returned value of *events_from_annotations* or it's modification). Based on this map, every event, that is mentioned in map, will be taken into consideration and there will be a representation of it. Last relevant parameters are *tmin* and *tmax*, which are values that represent start and end of the time interval respectively.
Use and their manipulation *Epochs* takes up most of this thesis, since it contains most of the necessary information, one of them being representation of EEG data. EEG data as mentioned before are a 3-dimensional *ndarray* and their dimensions represent: amount of epochs, picked channels and data in chosen intervals. As mentioned in previous section, both files of EEG data with subjects using right and left arm were read and saved as *Epochs* objects. This way access to data of both arms was made easier. First step was to take data from *Epochs* object with event number two and six. In section BrainVision we mentioned that only existing markers were 1, 2, 4 and 5. Event number 6 is actually event 5 but all mapping on event 5 has been artificially changed into mapping on event 6. This is essential as in both files event number 5 represents movement but doesn't distinguish which hand was moved. As for reason why event number 2 was chosen, is because we're assuming that person is more relaxed in the middle of rest phase than at the start of resting. Moving the lever of the robot and suddenly stopping takes some toll and would show on EEG data. As for choosing event number 5, we're assuming that intention of movement is highest at the start of the movement phase because intention of the movement occurs when subjects realise that rest phase ended. After [] EEG event labels and are collected. Then EEG data of file with left hand is collected together with their labels. To work with both these arrays of EEG data and their labels they were concatenated respectively.

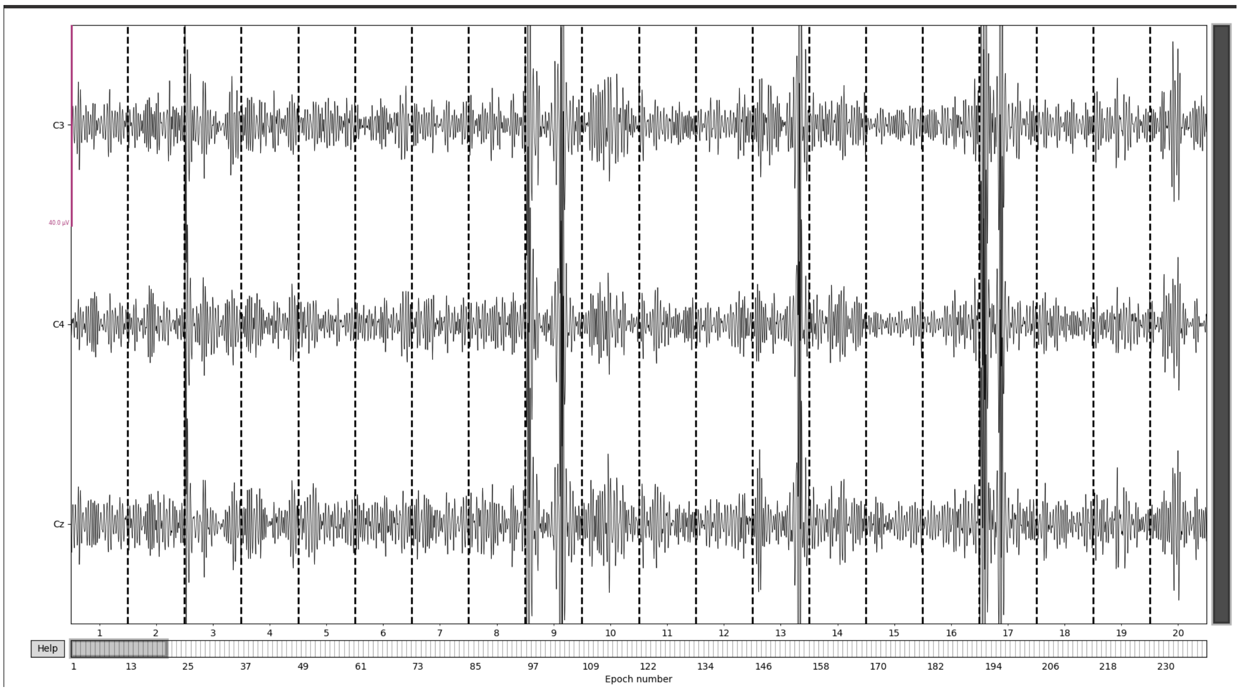

Figure 4.2: Example of EEG data Epochs

4.2.3 CSP

CSP from library MNE has been used as a transformation in *Pipeline*. To create an object representing *CSP* it is necessary to call construction *CSP* and passs it's parameters. Parameter to pass is called *n_components* and it defines how many components to decompose EEG signals into. *N_components* parameter for *SVM* using *CSP* has been chosen to be 5. According to MNE documentation, this value should be chosen based on *cross-validation* value. *CSP* for *LDA* is 4, which is a default value. As mentioned in previous chapters *cross-validation* wasn't used for *LDA*. As it is used as a transformation, function *transform* returns 2-dimensional: array of amount of epochs and amount of sources. These epochs are averaged by the power of CSP features.

4.2.4 Vectorizer

Vectorizer is used to convert N-dimensional data into 2-dimensional data, where first dimension is amount of samples and the other one is array of features. Since *CSP* is creating a 2-dimensional data there is no need to use *Vectorizer* on classifier using *CSP* to extract feature vectors.

5 Achieved results

Results were collected from two datasets, one dataset is from last year (2021). This dataset was used for another thesis by Pavel Mochura (available here). This dataset has been used in this thesis only to test classifiers and have more data to test with. Second dataset is a dataset collected this year using scenario explained in section EEG measurement. Results should show comparison of classifiers accuracies. In the first dataset, data were measured from 14 people, 9 of them were female and 5 of the male. Their average ages were 19.6 and 20.6 years respectively (taken from Pavel Mochura's thesis at chapter 8, available here). In the second dataset 10 people were measured, from which 4 were female and 6 were male.
From graphs it seems that classifiers with *CSP* have better performance in average. What is also apparent is that classifiers using *CSP* are on average better than their respective counterparts without *CSP*. This means that choosing *CSP* has proven to be beneficial for classifying EEG data but also when it come to *precision*, *recall* and *f1-score.* Tables with information about old data are in their respective file attached to this thesis as *table_old_data.pdf.* Reason for a separate file for table data is that there is too much of information and it would be better to have a different file designated for it.

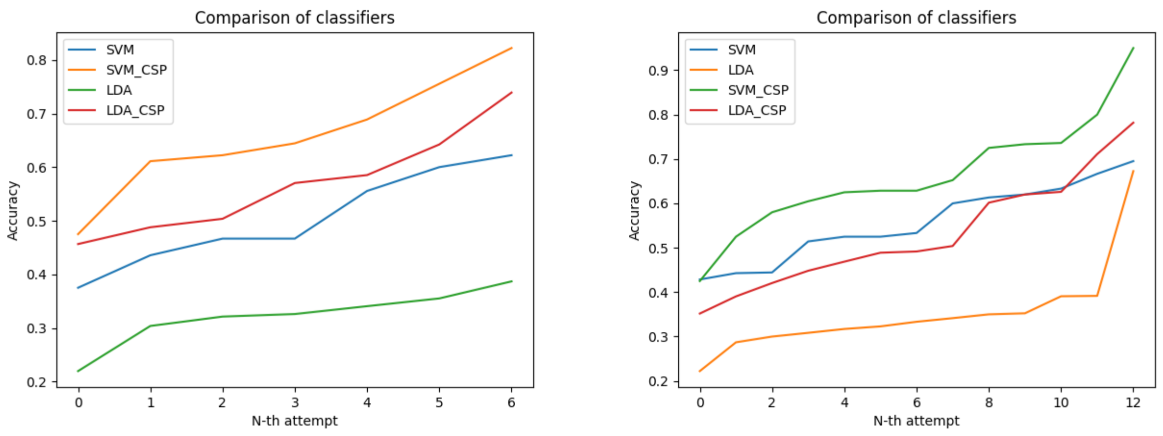

Figure 5.1: Comparison of performance of classifiers on new data and old data respectively. Accuracies are sorted from worst to best

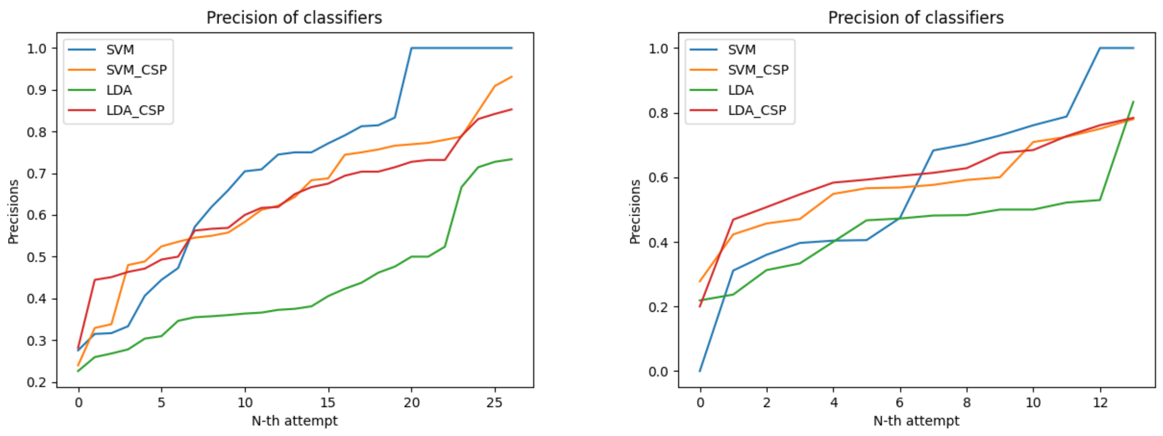

Figure 5.2: Precisions of the classifiers for new data and old data respectively. Precisions are also sorted from worst to best

It is pretty apparent from graphs depicted in pictures 5.2 and results from table 5.1, results are kind of different. This has been done on purpose to keep the results as robust as they can be because with each run, data are divided randomly into training set and test set. That means each time there are different results. In spite of this data are different each run, classifiers using *CSP* end up with better results except for some of the anomalies. For example SVM classifier without feature vectors classifies sometimes with better accuracy than SVM using *CSP* but SVM using *CSP* classifies better in average.
From the table and graphs depicted, it is uncertain why is the interval of accuracies so wide. Classifier with accuracy lower than 50% is essentially a

Name	***Accuracy***	***Precision***	***Recall***	***F1-score***	***Support***	***C - parameter***	***Kernel***
SVM-Normal	0.4667	[0.5741-0.5455-0.4]	[0.6739-0.4-0.4348]	[0.62-0.4615-0.4167]	[46-45-46]	10	rbf
SVM-CSP	0.5778	[0.6667-0.4318-0.3889]	[0.6047-0.4043-0.4468]	[0.6341-0.4176-0.4158]	[43-47-47]	10	rbf
LDA-Normal	0.4307	[0.5484-0.3929-0.4]	[0.3542-0.5238-0.4255]	[0.4304-0.449-0.4124]	[48-42-47]	X	
LDA-CSP	0.4526	[0.5484-0.3929-0.4]	[0.3542-0.5238-0.4255]	[0.4304-0.449-0.4124]	[48-42-47]	X	
SVM-Normal	0.4444	[0.6607-0.4074-0.0]	[0.8605-0.7021-0.0]	[0.7475-0.5156-0.0]	[43-47-48]	1	rbf
SVM-CSP	0.8	[0.7115-0.6-0.7805]	[0.7708-0.6279-0.6809]	[0.74-0.6136-0.7273]	[48-43-47]	1	rbf
LDA-Normal	0.3478	[0.3611-0.3714-0.3284]	[0.2955-0.26-0.5]	[0.325-0.3059-0.3964]	[44-50-44]	X	
LDA-CSP	0.6739	[0.3611-0.3714-0.3284]	[0.2955-0.26-0.5]	[0.325-0.3059-0.3964]	[44-50-44]	X	
SVM-Normal	0.6	[0.3115-0.5294-0.3333]	[0.4634-0.18-0.4318]	[0.3725-0.2687-0.3762]	[41-50-44]	10	rbf
SVM-CSP	0.6	[0.4773-0.5714-0.7857]	[0.4884-0.4444-0.9362]	[0.4828-0.5-0.8544]	[43-45-47]	10	rbf
LDA-Normal	0.3778	[0.3871-0.3333-0.4571]	[0.2609-0.5476-0.3404]	[0.3117-0.4144-0.3902]	[46-42-47]	X	
LDA-CSP	0.5704	[0.3871-0.3333-0.4571]	[0.2609-0.5476-0.3404]	[0.3117-0.4144-0.3902]	[46-42-47]	X	
SVM-Normal	0.4667	[0.7143-0.3571-0.38]	[0.3488-0.1064-0.8444]	[0.4688-0.1639-0.5241]	[43-47-45]	10	rbf
SVM-CSP	0.5333	[0.6429-0.4194-0.8571]	[0.2045-0.9286-0.4898]	[0.3103-0.5778-0.6234]	[44-42-49]	10	rbf
LDA-Normal	0.3111	[0.3529-0.2609-0.3333]	[0.1364-0.2667-0.5217]	[0.1967-0.2637-0.4068]	[44-45-46]	X	
LDA-CSP	0.637	[0.3529-0.2609-0.3333]	[0.1364-0.2667-0.5217]	[0.1967-0.2637-0.4068]	[44-45-46]	X	
SVM-Normal	0.475	[1.0-0.2927-1.0]	[0.0-1.0-0.0]	[0.0-0.4528-0.0]	[47-36-40]	0.001	linear
SVM-CSP	0.625	[0.5741-0.5385-0.3488]	[0.6739-0.3333-0.4286]	[0.62-0.4118-0.3846]	[46-42-35]	0.001	linear
LDA-Normal	0.3496	[0.3333-0.4054-0.3158]	[0.381-0.3571-0.3077]	[0.3556-0.3797-0.3117]	[42-42-39]	X	
LDA-CSP	0.4797	[0.3333-0.4054-0.3158]	[0.381-0.3571-0.3077]	[0.3556-0.3797-0.3117]	[42-42-39]	X	
SVM-Normal	0.4578	[0.5455-0.2712-0.2647]	[0.125-0.3404-0.4186]	[0.2034-0.3019-0.3243]	[48-47-43]	1	sigmoid
SVM-CSP	0.4844	[0.4074-0.3021-0.6]	[0.2444-0.6744-0.18]	[0.3056-0.4173-0.2769]	[45-43-50]	1	sigmoid
LDA-Normal	0.3188	[0.3056-0.3-0.3387]	[0.25-0.2609-0.4375]	[0.275-0.2791-0.3818]	[44-46-48]	X	
LDA-CSP	0.4275	[0.3056-0.3-0.3387]	[0.25-0.2609-0.4375]	[0.275-0.2791-0.3818]	[44-46-48]	X	
SVM-Normal	0.6	[0.7907-1.0-0.4255]	[0.7727-0.0-0.9302]	[0.7816-0.0-0.5839]	[44-50-43]	1	rbf
SVM-CSP	0.6889	[0.6667-0.7209-0.675]	[0.8-0.6889-0.5745]	[0.7273-0.7045-0.6207]	[45-45-47]	1	rbf
LDA-Normal	0.365	[0.381-0.4359-0.3036]	[0.3721-0.3542-0.3696]	[0.3765-0.3908-0.3333]	[43-48-46]	X	
LDA-CSP	0.7372	[0.381-0.4359-0.3036]	[0.3721-0.3542-0.3696]	[0.3765-0.3908-0.3333]	[43-48-46]	X	

Table 5.1: Table of metric accuracies for data measured this year. Three values in table represent metric value of *neutral state*, *right arm movement* and *left arm movement*

coin flip machine or worse, while on the other hand, classifier with accuracy of 80%, can be considered acceptable. There could be many reasons for that, one of them being invalid data. This thesis attempted to search for movement in data but if the subject winked as they moved lever of a robot, information about movement got lost in the EEG data. There could've also been an issue finding movement. This could be because when phases switch from resting to action phase, before subject moves the lever there is lag when subject realises it is action phase. Other reason could be poorly chosen classifier, another approach for classifying could be using *logistic regression*. Another possibility could be some kind of pre-processing of data if it is possible. From *precisions* of classes we can conclude that when classifier has a high accuracy, classifier found movement of left and right arm as well as resting phase.
Another useful metric would be confusion matrix, that would tell us how the values were classified. Using this metric could be useful for analysis of EEG data.
Problem with knowing what the best approach is that there isn't a sure way to test if the classifier was correct other than analysis of EEG data, which requires an extensive experience and professional expertise.

6 Conclusion

In this bachelor's thesis I tackled the task of classifying EEG data, choosing proper feature vectors and collecting the necessary data. One of the items in the assignment wasn't done solely by me. That task being creating the scenario, which wasn't done only by me, as I was only contributing to it.
In order to work with EEG data I had to get acquainted with what EEG is, montages, BCIs, some of the approaches to BCI's, different kinds of classifying algorithms and methods and some of the methods to extract features from EEG data. It was of course necessary to find people that would be willing to be measured and provide EEG data for this thesis.
While collecting data, it was our utmost priority to ensure for the quality of the to be data are as good as possible. For that reason the environment was calm and we made sure subjects were relaxed and ready for measurement. After data was collected library *MNE* was used so EEG data were handled and using *Matplotlib* they were displayed. For further work with EEG data, it was necessary to learn some of the BCIs and methods of working with EEG data.
BCIs have a variety of different specializations and many are for detecting movement in EEG data using many other approaches. Some of them are using a stimulus and are training their classifier models based on timestamps, where the occured happened. In this thesis detection of SMR was chosen because there wasn't any additional stimulus needed for detecting movement. In order to classify SMR, it was necessary to get acquainted with what SMR is, how to detect it, at which bandwidth can SMR be detected. Using *MNE*'s functions and object methods helped creating required bandwidth and allowed further work with it.
In order to classify data it was necessary to use some of the classifiers available from library *Scikit-learn* which provides many classifiers (for this thesis *SVM* and *LDA* were chosen), their functionality and many statistical metrics. There was an issue of working with data as *Scikit-learn* requires data in a specifi format. This was fixed using dimension reductions of *Vectorizer* and *CSP* (this component is from *MNE*).
After classifying, metrics were needed in order to figure out how efficient classifier was. The used metrics were *accuracy*, *precision*, *recall* and *F1-score*. It is necessary to have another metric other than *accuracy* as classifier could be classifying into one class perfectly but not into other ones.
Accuracies of classifiers was between 20% - 90%. Reason for such low accur-

acy could've been because of some disruption in EEG data, another reason could've been not enough of data and many more discussed in previous chapter.
This thesis could be analysed even further and probably different approaches could be used or even try the classifiers on another dataset. Another addition could be classifying different movement and not just the movement of hands.

Bibliography

[1] [online]. README of the data for measurement. Available at: https://drive.google.com/drive/u/1/folders/18KRukxcBD-YS5Rn5uRISIBQL-a5XddDC.

[2] *Classification* [online]. Merriam-Webster. (n.d.). Available at: https://www.merriam-webster.com/dictionary/classification.

[3] *IBM - Supervised learning* [online]. Supervised learning. Available at: https://www.ibm.com/cloud/learn/supervised-learning.

[4] *IBM - Unsupervised learning* [online]. Unsupervised learning. Available at: https://www.ibm.com/cloud/learn/supervised-learning.

[5] Post-Stroke Rehabilitation Fact Sheet, Nov 2021. Available at: https://www.student.unsw.edu.au/how-do-i-cite-electronic-sources.

[6] MNE Vectorizer Documentation. Available at: https://mne.tools/stable/generated/mne.decoding.Vectorizer.html.

[7] Mne.io.raw. Available at: https://mne.tools/stable/generated/mne.io.Raw.html.

[8] Support Vector Machines Explained, . Available at: https://scikit-learn.org/stable/modules/svm.html.

[9] Scikit-learn - Linear discriminant analysis, . Available at: https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html?highlight=linear+discriminant#sklearn.discriminant_analysis.LinearDiscriminantAnalysis.

[10] Cross-validation: Evaluating estimator performance, . Available at: https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation.

[11] GridSearchCV, . Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html?highlight=gridsearch#sklearn.model_selection.GridSearchCV.

[12] LinearSVC, . Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC.

[13] Sklearn.pipeline.make_pipeline, . Available at: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.make_pipeline.html#sklearn.pipeline.make_pipeline.

[14] NuSVC, . Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html#sklearn.svm.NuSVC.

[15] Sklearn.pipeline.pipeline, . Available at: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline.

[16] Sklearn.metrics.precision_recall_fscore_support, . Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html#sklearn.metrics.precision_recall_fscore_support.

[17] SVC, . Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html?highlight=svc#sklearn.svm.SVC.

[18] Blankertz, B. et al. Optimizing Spatial filters for Robust EEG Single-Trial Analysis. *IEEE Signal Processing Magazine*. 2008, 25, 1, s. 41–56. doi: 10.1109/MSP.2008.4408441.

[19] Dahan, A. – Ryder, C. H. – Reiner, M. Components of Motor Deficiencies in ADHD and Possible Interventions. *Neuroscience*. 2018, 378, s. 34–53. ISSN 0306-4522. doi: https://doi.org/10.1016/j.neuroscience.2016.05.040. Available at: https://www.sciencedirect.com/science/article/pii/S0306452216301993. Neurofeedback and Functional Enhancement: Mechanisms, Methodology, Behavioral and Clinical Applications.

[20] Ing. Pavel Král. Classfiers, recognition and clusterization. Available at: http://home.zcu.cz/~pkral/uir/pr5-materialy/FThema4-select.pdf.

[21] Gramfort, A. et al. MEG and EEG Data Analysis with MNE-Python. *Frontiers in Neuroscience*. 2013, 7, 267, s. 1–13. doi: 10.3389/fnins.2013.00267.

[22] Jeunet, C. et al. Using EEG-based brain computer interface and neurofeedback targeting sensorimotor rhythms to improve motor skills: Theoretical background, applications and prospects. *Neurophysiologie Clinique*. 2019, 49, 2, s. 125–136. ISSN 0987-7053. doi: https://doi.org/10.1016/j.neucli.2018.10.068. Available at: https://www.sciencedirect.com/science/article/pii/S0987705318302594. Neurophysiology of Movement: From preparation to action.

[23] Kim, Y. et al. Motor Imagery Classification Using Mu and Beta Rhythms of EEG with Strong Uncorrelating Transform Based Complex Common Spatial Patterns. *Computational Intelligence and Neuroscience*. Oct 2016, 2016, s. 1489692. ISSN 1687-5265. doi: 10.1155/2016/1489692. Available at: `https://doi.org/10.1155/2016/1489692`.

[24] Kubat, M. *Probabilities: Bayesian Classifiers*, p. 19–26. Springer, 1 edition, .

[25] Kubat, M. *Similarities: Nearest-Neighbor Classifiers*, p. 43–46. Springer, 1 edition, .

[26] Pedregosa, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, 12, s. 2825–2830.

[27] Rao, R. P. N. Brain-Computer Interfacing an Introduction. 2013, p. 101, 239. doi: https://doi.org/10.1017/CBO9781139032803.

[28] Relvas, V. – Sanches, J. M. – Figueiredo, P. Scalp EEG Continuous Space ERD/ERS Quantification. In Sanches, J. M. – Micó, L. – Cardoso, J. S. (Ed.) *Pattern Recognition and Image Analysis*, p. 616–623, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-38628-2.

[29] Sasaki, Y. The truth of the F-measure. *Teach Tutor Mater*. 01 2007.

[30] Siuly, S. – Li, Y. – Zhang, Y. EEG Signal Analysis and Classification: Techniques and Applications. 01 2016, p. 3–5. doi: 10.1007/978-3-319-47653-7.

[31] Subasi, A. – Ismail Gursoy, M. EEG signal classification using PCA, ICA, LDA and support vector machines. *Expert Systems with Applications*. 2010, 37, 12, s. 8659–8666. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2010.06.065. Available at: `https://www.sciencedirect.com/science/article/pii/S0957417410005695`.

[32] Teplan, M. Fundamental of EEG Measurement. *MEASUREMENT SCIENCE REVIEW*. 01 2002, 2.

[33] Wolpaw, J. R. – McFarland, D. J. Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proceedings of the National Academy of Sciences of the United States of America*. Dec 2004, 101, 51, s. 17849–17854. ISSN 0027-8424. doi: 10.1073/pnas.0403504101. Available at: `https://pubmed.ncbi.nlm.nih.gov/15585584`. 15585584[pmid].

A The contents of the attached ZIP file

Folder structure is divided into 4 parts "Aplication_and_libraries", "Input_data", "Results" and "Text thesis"

- "Text thesis" contains .pdf file that is eletronic version of thesis on topic of detecting movement in EEG. It also contains folder LaTex, that contains build files for LaTex version of thesis.
- "Aplication_and_libraries" contains script main.py, which is starting point of application
- "Input_data" contains measured data from EEG
- "Results" contains .pdf file that contains table with results generated after classifying old data

Zip file also contains *main.py* which is a source file for this thesis.

B User guide

In order for correct functionality of script *main.py Python3* is needed and these *Python3* libraries:

- MNE
- Scikit-learn
- Matplotlib
- NumPy

Script has 3 functions *classify_old_data*, *classify_new_data* and *main*. Main function calls first 2 mentioned functions. To run the script command *python3 main.py* has to be typed.