UCLouvain

École polytechnique de Louvain

EEG for Parkinson's disease detection

Author: **Guglielmo Colombo**
Supervisor: **Benoît Macq, Gabriella Olmo**
Readers: **Nicolas Delinte, Gaetan Rensonnet, Laurence Dricot, Cyril de Bodt**
Academic year 2021–2022
Master [120] in Computer Science and Engineering

List of Figures

Contents

Abstract

Parkinson's disease (PD) is a neurodegenerative disorder that mainly affects dopamine-producing neurons in the substantia nigra region of the brain.
Unfortunately, the conventional screening methods for PD patients are subjective and manual. Hence, to perform automatic screening of PD patients, objective methods are needed. The electroencephalographic (EEG) data has been used to study the differences in brain signals between PD patients and healthy controls and was further developed an automatic screening tool for Parkinson's disease, using Machine Learning to individuate its discriminating pattern.
To permit an easier access and usage of these algorithms, a web interface was created to ease the access to the results. The purpose of the web interface is to give the possibility to doctors and whoever is interested in the prediction of whether a patient is affected by Parkinson's or not,to face a familiar and ready to use interface, rather than a laborious Python IDE.

Acknowledgment

I would like to thank Benoit Macq and Gabriella Olmo for giving me the opportunity to work on this subject during my master's thesis.
I would also like to show my gratitude to Nicolas Delinte who took the time to answer my questions and assisted me for some technical organizational issues.

And thanks to all the people I care about, from A to Zeno.

Part I

Medical and Biological Background

Medical Introduction

In mental healthcare, advances in data and computational science are rapidly changing, with the state of the art of some new technologies, such as the Electroencephalogram. Additionally, the use of machine learning (ML) for automatic classification, has increased.
ML may prospectively test the performance of predictions on unobserved data not used to fit the model using out-of-sample estimations, providing customized information and possibly high clinical translation results. This approach is contrary to classical inference based on null hypothesis tests. ML is expected to help or possibly replace clinical decisions such as diagnosis, prediction, and prognosis or treatment outcome.[7]
Many studies related to these new methods became more and more popular, however, their potential is still not fully exploited in the practical medical diagnosis.

The majority of neurological research applied a supervised Machine Learning approach for diagnostic binary classification between patients and healthy controls, however, EEG studies that include a variety of psychiatric disorders are beginning to emerge. Studies have predominantly focused on Alzheimer's disease, schizophrenia, alcoholism, and depression but have more recently expanded to other diagnostic topics. The literature suggests that ML can be used to discriminate psychiatric disorders using brain data with over 75% accuracy.[1]

Electroencephalography (EEG) measures brain activity and delivers information about the voltage measured through electrodes placed on the scalp. EEG is non-invasive, cost-effective, and suitable for measuring resting-state brain activity in natural settings, allowing easy acquisition of large amounts of data.[2] Considering all these advantages related to this technology, is now widely used and is gaining importance in the studies of neurological diseases.
The aim of this thesis is therefore to identify, using machine learning algorithms, whether EEG data can be used to identify Parkinson's disease. To achieve this objective, ML methods will be applied on the EEG dataset to classify the PD

patients.
Afterwards, the development of a web application that will allow easier access to the classification results through a user-friendly interface.

The project is divided into four parts.

- The first one provides a theoretical background of the EEG data acquisition process and of the main brain structures involved in the process.
- In the second part are presented all the steps implemented to reach the final classification of the EEG signals and for the evaluation of the relative performances.
- The third part contains an inspection of the obtained result with automatic machine learning prediction.
- In the final part is presented the web interface deployed to access the signals classification.

I.1 Electroencephalogram

The main source of data for this thesis project is coming from Electroencephalograms.
In the following sections, the focus will be on explaining what is an electroencephalogram, how is obtained, and what it records. To fully understand how the electroencephalogram can record the activity coming from the brain, a short medical insight is given to the brain functions activity and its fundamental parts.[2]
Electroencephalography (EEG) is a method to record the electrical activity on the scalp, that has been shown to represent the macroscopic activity of the surface layer of the brain underneath.[2] One of the biggest advantages of this technique is that it is non-invasive, with the electrodes placed along the scalp outside the human body.
EEG measures voltage variations produced by ionic current in the brain's neurons. Clinically, it refers to the recording of the brain's spontaneous electrical activity over a period of time, and is recorded from multiple electrodes. [14]
EEG is most often used to diagnose pathologies that cause abnormalities in EEG readings as they affect the normal operation of the brain in a subject with a neurological disease. This technique can be used to be a first-line method of diagnosis for tumors, stroke and other focal brain disorders, even if this use has decreased with the advent of high-resolution anatomical imaging techniques such as magnetic resonance imaging (MRI), which are still more difficult to record with respect to EEG. [15]
However, EEG doesn't record the electrical events that neurons use to communicate with each other, which are called in medical terms "action potentials". Instead, it surveys the summed activity of hundreds of thousands or millions of neurons in the form of oscillatory activity.[8] It's not possible to know which type of information these oscillations carry, as with action potentials; however different frequencies of oscillations correlate with different behavioural states, and for each behavioural state is possible to detect a sort of standard in oscillations for healthy patients, and is on this concept that is based the research of this project. [15]

I.2 Brain structure

The brain is the main source of input features for this project, as the EEG registers the brain activity through the electrodes. For this reason a short insight of the brain structure will be given in the current section.
Three main parts compose the brain structure: the forebrain, midbrain and hindbrain, each composed of multiple subparts.[3]
The forebrain is the most interesting as it comprehends the "cerebrum", the biggest

section of the brain. Also known as the cerebral cortex, the cerebrum is the largest part of the human brain, and it is associated with higher brain functions such as thought and action. The grey matter is made of nerve cells, the white matter of nerve fibers that carry signals between nerve cells in other parts of the brain and body [4]. The white surface is then the actual part of the brain which produces the electrical signals recorded during the EEG experiment.

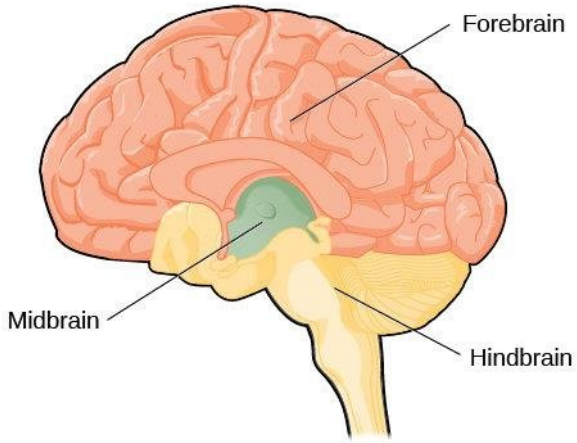

Figure I.2: Illustration of the main parts of human brain, from [6]

I.3 Structure of the Neuron

The neuron is the base cell in the brain. It has three main parts: dendrites, an axon, and a cell body or soma (see figure I.3).
A dendrite is where input from other cells is received by a neuron. Dendrites branch as they move towards their tips create leaf-like structures on them, called spines. The output structure of the neuron is the axon; when a neuron wants to communicate with another neuron, it sends the action potential throughout the entire axon.[8] The soma is where the nucleus lies, where the neuron's DNA is housed, and where proteins are made to be transported throughout the axon and dendrites. [20]

I.4 Action Potentials

Action Potentials are the base component for brain functioning.
With the following four steps can be summarized the neurons' activity of communication between each other through Action Potentials.[19] [16]

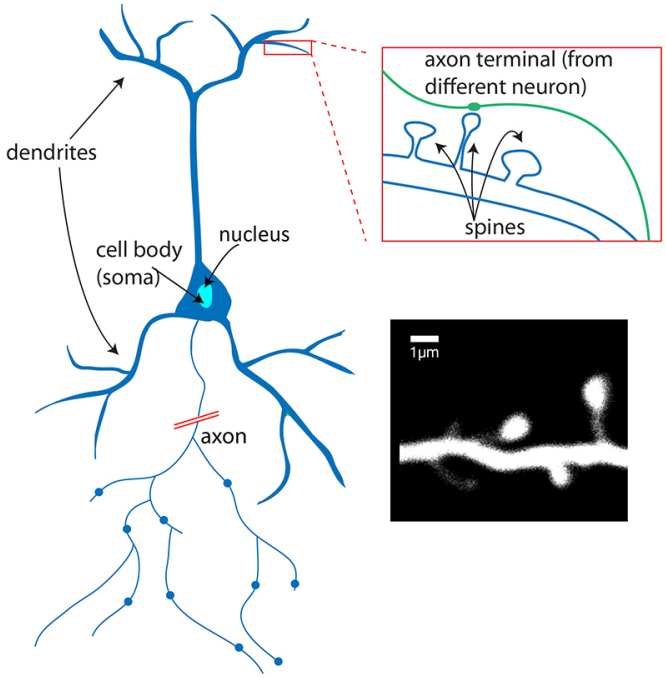

Figure I.3: General Illustration of Dentrites Structure from [8]

1. Through electrical events called 'action potentials' and chemical neurotransmitters, neurons communicate with each other .
2. An action potential causes a neuron to release a chemical neurotransmitter in the synapse, which is the junction between two neurons.
3. The action of the neurotransmitter could either excite or inhibit the receiving neuron, at the other side of the synapse, from firing its own action potential.
4. A combination of hundrends of exciting and inhibiting inputs will determine the result of the Action Potential.

Neurons are essentially electrical devices, their possibility to carry voltage is guaranteed by their structure. The cell membrane, which is the boundary between the inside and outside of the cell, contains a big number of canals that allow the flow of positive and negative ions in both directions of the cells.[8] The constant flow of positive and negative ions, cause the membrane potential to be not static, but it's constantly going up and down, depending on the inputs coming from other neurons. [16]

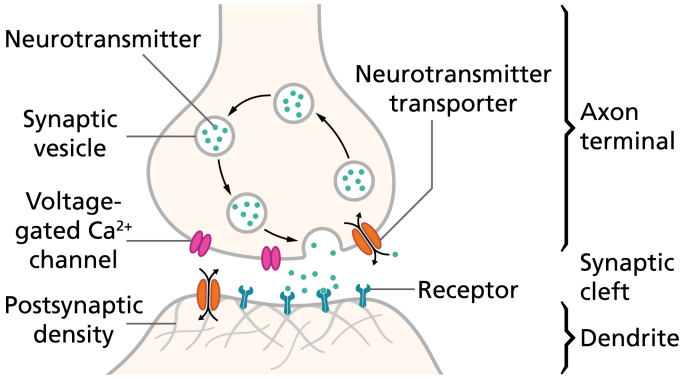

Figure I.4: General Illustration of Synapses from [8]

When the neurons' membrane potentials reach a specific value called the action potential threshold, which is around -50mV, due to the continuous excitatory and inhibitory activity, the Action Potential occurs, generating a spike, as shown in Figure I.4.[16]

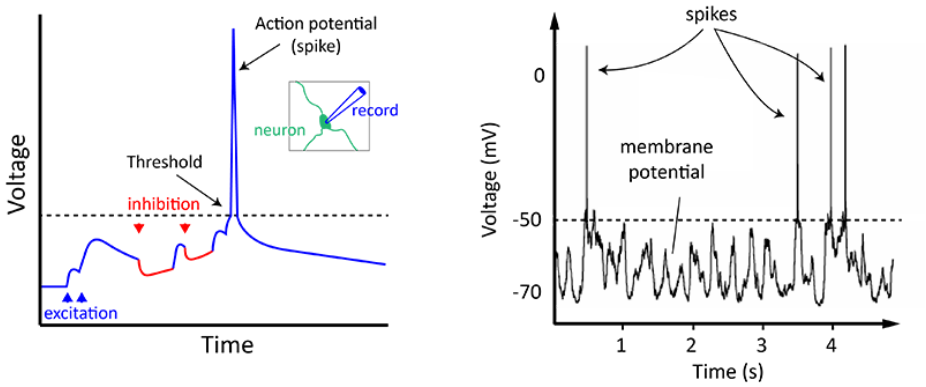

Figure I.4: General Illustraction of Action Potential Spikes after that the voltage value reaches the Action Potential Spike, from [8]

I.5 Synapses

Neurons communicate between each other across synapses. When an action potential reaches the presynaptic terminal, as a conequence the neurotransmitter is released from the neuron into the synaptic cleft, a gap between the presynaptic axon terminal and the postsynaptic dendrite.[19]
The synapses convert the electrical signal (the action potential) into a chemical one, as the neurotransmitter is released in the synaptic cleft (see figure I.4). The receiver

of the chemical neurotransmitter convert then the signal back into an electrical one, with the flow of positive and negative ions in and out the cell membrane of the postsynaptic neuron (the receiver of the signal in the synapse). [18]

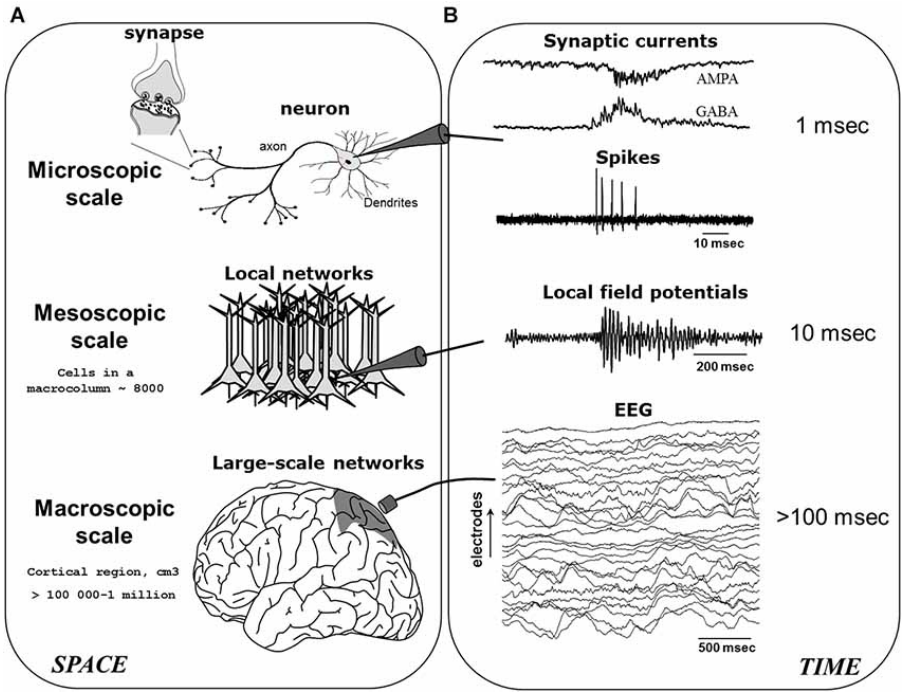

Figure I.5: Illustration of the different stages of a neurological signal in the brain, from the microscopic to the actual EEG rilevation with electrodes, from [5]. It clearly shows how the electrode registers the combined electrical activity of multiple spikes in a precise area of the brain.

I.6 Electrodes Placements

The 10–20 system is a method that describes how to apply the location of scalp electrodes in the context of an EEG exam, and to allow comparable results across different studies internationally recognized. This approach was created to maintain uniform testing methodologies so that a subject's study results may be compiled, reproduced, then examined and compared successfully, using the scientific method. [21]
The system is based on the relationship between an electrode's placement and the brain's underneath area. The "10" and "20" refer to the fact that the actual distances between contiguous electrodes are either 10% or 20% of the overall front–back or right–left distance of the skull. [22] In other words, as can be seen in the figure below, all the electrodes placed on the skull have a distance between each other which corresponds to a specific value based on the skull dimensions. This method allows a standardized approach to the EEG recordings.
Pre-frontal (Fp), frontal (F), temporal (T), parietal (P), occipital (O), and central (C) are the letters assigned to each electrode placement location to indicate which lobe or area of the brain it is reading from.[23]

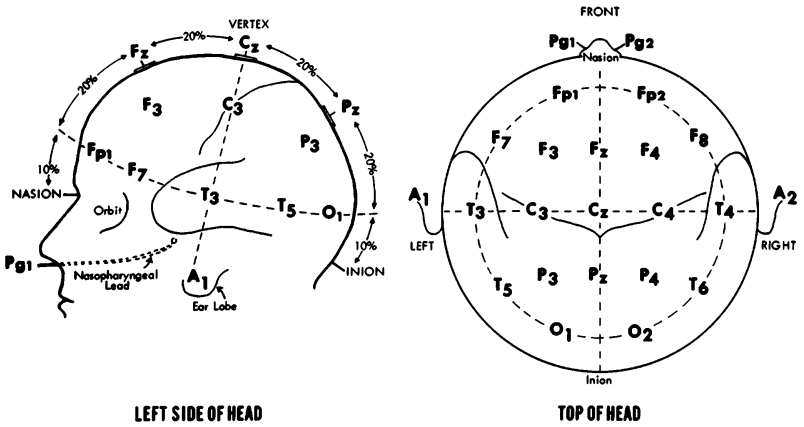

Figure I.6: Illustration of the electrodes positioning of patient scalp following the 10-20 system approach, from [23]

I.7 Bands

Electroencephalograms are recorded during a state of rest of the patient, eyes closed (EC) and eye open (EO) conditions without being involved in any experimental activity, and for this reason can be also named resting EEG (rEEG).
The rEEG data are of composite nature and may be decomposed into frequency

bands such as delta (0.5–4 Hz), theta (4–8 Hz), alpha (8–12 Hz), beta (12–30 Hz) and gamma (> 30 Hz).[17] Each frequency bands carry different physiological information associated with the brain activities.

- Delta and Theta bands are manifested during information encoding in order to create new memory, as are associate to a sleep/drowsiness phase. [26]
- Alpha band indicates the information retrieval from memory and attention. [26]
- Beta and Gamma bands are associated with perception, learning and attention. [27]

In a patient affected by Parkinson, or any other disease that could be detected through the EEG analysis, the values reached by each of these bands present a distortion from the ones of a healthy patient. [25]

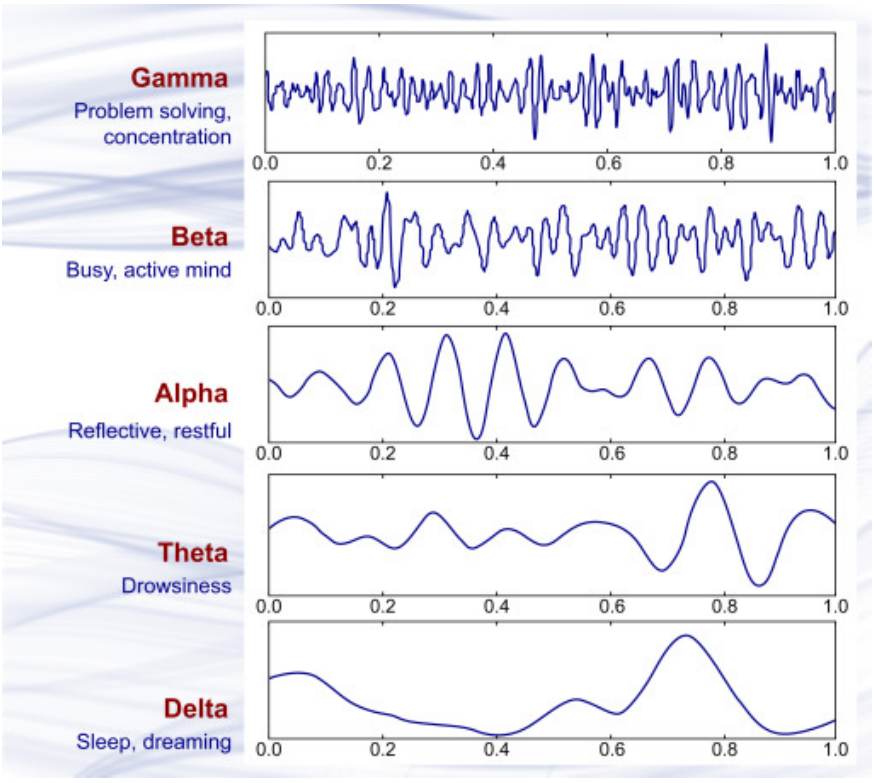

Figure I.7: Bands of interest in the EEG analysis, separated for the different frequency range, from [24]

I.8 Parkinson Disease

Parkinson's disease (PD) is a progressive disorder that affects nerve cells in the brain responsible for body movement. When neurons responsible for the production of dopamine die, symptoms such as tremors, slowness, stiffness, and balance problems occur.[9]
As seen before, when an impulse comes to the end of the axon, it is converted from electrical to chemical, releasing the neurotrasmitter in the synapse's cleft. In patients affected by Parkinson, the dopamine neurotransmitter is no longer released by the axons.
The dopamine-producing nerve cells of the substantia nigra, one of the brain's key dopamine-producing regions, begin to die out in certain people for unknown causes. PD symptoms such as tremors, slowness of movement, stiffness, and balance issues appear after 80% of dopamine is lost.[9]
Body movement is controlled by a complex chain of decisions involving interconnected groups of nerve cells called ganglia.[10]
The basal ganglia and cerebellum are in charge of ensuring that movement is smooth and fluid. When the quantity of neurotransmitters released during the neurons' communications is no longer enough, all this chain is interrupted, and causes tremors and other symptoms in the affected subjects.[11]
The Electroencephalograms record the electrical activity of the basal ganglia, which is the part of the brain highly affected by Parkinson disease.

I.8.1 Parkinson's Symptoms

Symptoms of Parkinson Disease vary from person to person, as does the rate of progression. These are the most common "hallmark" symptoms:[12]

1. Bradykinesia: slowness of movement, impaired dexterity, decreased blinking, drooling, expressionless face.
2. Tremor at rest: involuntary shaking that decreases with purposeful movement. It usually begins on one side of the body, most commonly the hand.
3. Rigidity: stiffness caused by an involuntary increase in muscle tone.
4. Postural instability: a sense of imbalance. Patients frequently compensate by lowering their center of gravity, leading to a stooped posture.

I.8.2 Diagnosis

Because other illnesses and medications can create symptoms similar to Parkinson's disease, it's important to get a proper diagnosis from a physician. Because

symptoms differ from person to person, no single test can validate a diagnosis of PD.
During the physical examination, suspected PD patients are asked to complete certain movements, which can help establish the diagnosis. For example, in people with Parkinson's disease, when they are asked to touch their nose with the hand, the tremor lessens or disappear. People with the condition also have trouble completing rapid alternate movements, such as placing their hands on their thighs and rapidly turning them back and forth numerous times. [13]

Due to this difficulty in diagnosing an actual state of Parkinson's diesease, the analysis of patients' EEG through Machine Learning could be highly effective and useful.

One of the most common technique for the diagnosis of Parkinson is the Unified Parkinson Disease Rating Scale Test, which consists of a series of tests for the patients to gauge the curse of their disease level of Parkinson. This was the same test conducted on the patients of the dataset used for this project.

Unified Parkinson Disease Rating Scale Test (UPDRST)

Several medical organizations have improved the UPDRS scale throughout the years, and it remains one of the therapy and research bases in PD clinics. The UPDRS scale consists of a series of ratings for typical Parkinson's symptoms that cover all movement issues associated with the condition. The UPDRS scale consists of the following five divisions:[28]
1) Mentation, Behavior, and Mood,
2) ADL,
3) Motor sections,
4) Modified Hoehn and Yahr Scale,
5) Schwab and England ADL scale.

During patient interviews, a medical professional specialized in Parkinson's disease evaluates each answer on the scale. On the UPDRS scale, a score of 199 indicates total disability, while a score of zero indicates no disability.[28]

Part II

Computer Science methods applied for EEG Analysis

Methods

This chapter outlines the different steps taken during this work, starting from the raw signals, to the final machine learning prediction. First data should be recorded, and the EEG signal should then be preprocessed. From these preprocessed signals, features can be extracted, and only the most suitable ones are selected. From these features, machine learning algorithms could be applied to predict the binary prediction problem. This is explained in further details in the machine learning section. All the steps are summarized in Figure II.0.

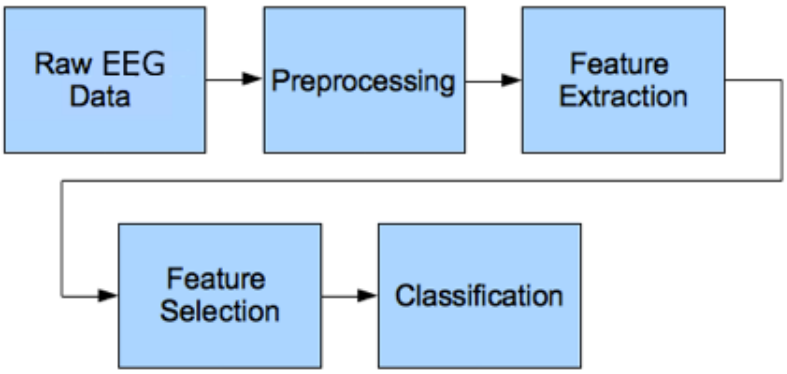

Figure II.0: Illustration of the data processing pipeline that leads to the Machine Learning prediction.

II.1 Data

Data used for this project come from an open-source dataset available online, disposable on [29]. This website makes available several open-source datasets, and their contribution to the study of psychiatric disease's patterns in EEG is significant. Indeed, there are very few online sources that offer patient-specific EEG data, and even fewer are arranged as PRED+CT, for the purpose of patients classification. Particularly other available dataset comprehends a limited number of samples, and do not control patients. This makes the task of EEG classification even more difficult, as it's necessary for a classification algorithm to have at disposal both the labels to classify. The purpose of the PRED+CT is to provide datasets with these characteristics.

The dataset used is composed of ninety-one PD patients (62 men and 29 women), who were recruited from the movement-disorders clinic at the University of Iowa. Their movement-disorders were examined by a physician to verify that they met the diagnostic criteria and tested with the Unified Parkinson Disease Rating Scale Test. Thirty-seven control participants were recruited from the Iowa City community and matched for age, sex, and education.[30] In accordance with the Declaration of Helsinki and the Ethics Committee on Human Research, all PD and control participants were assessed to have the decisional capacity to provide informed permission.
Eighty PD patients participated in the study while assuming their prescribed medications, which could affect interval timing and brain activity of determined bands. These PD patients were tested when they were taking medications as usual, and then they were asked to withhold the dopaminergic medications for 12 h before repeating EEG testing.[30]
Patients whose experimental extraction produced immediately observable artifacts or noise were discarded.
EEG signals were collected from 64 channels of EEG actiCAP (Brain Products GmbH), sampled at a frequency of 500 Hz and high-pass filtered at 0.1Hz. Two of the 64 electrodes recordings were discarded, as the reference was associated with electrode Pz, and the ground to electrode Fpz.[30]
A first pre-processing was performed with Matlab EEGLAB library, where bad channels and bad epochs were identified using the FASTER algorithm, and were then interpolated and rejected. Eye blinks artifacts were removed using independent component analysis (ICA).[30] This means the dataset available on line had already undergone some steps of preprocessing.

II.2 Preprocessing

The preprocessing step, in the context of developing a machine learning algorithm, doesn't have to be underestimated. Indeed, algorithms and computers work with numbers as are just a series of mathematical formulas, and is important to have reliable input values. This is achieved through preprocessing techniques consisting in rescaling the features and deleting the ones are not useful for the prediction.

Why preprocessing is an important step?

For many reasons, preprocessing is necessary for EEG data. The signals picked up by the scalp are not always an accurate representation of the impulses coming from the brain, as the spatial information of the electrodes placement gets lost.[32] Additionally, EEG data has a lot of noise in it, which might obscure the actual signals. Blinking or muscle movement can distort the actual value of the signal, but luckily this kind of artifacs can be detected through some preprocessing techniques, as ICA preprocessing. [33] Other noise could come from the surrounding environment of the experiment or some functioning problems of the electrodes.
Finally, could be necessary to separate the relevant neural signals from random neural activity that occurs during EEG recordings. Indeed, is the patient should stay in a rest condition during the experiment, however eye blinking, heart beats or small muscle contractions could be recorded by the electrodes.

Figure II.2 shows the different types of anomalies that could be detected in EEG signals.

Different types of Preprocessing

There is no commonly accepted EEG preprocessing pipeline as it is still an open research topic, indeed for each different signal, different types of steps could be more suitable.
Some considerations regarding the dataset could help understand which steps could actively improve the validity and meaning of the data.[32]

- Considering which kind of artifacts has to be removed, ICA is a valuable approach to individuate eye blinking and eye movement and remove them from the signals.
- If the preprocessing of data happens as soon as they arrive, it might not be the best approach to use as ICA Preprocessing requires high computational time, and an active interaction of the user to exclude the right components.

However, any pre-processing techniques will be able to restore validity for bad data if these are irremediably compromised. If the experiment was not conducted properly by the doctor or by the subject, or due to a malfunctioning in the equipment, it may be best to simply run the experiment again, rather than trying to salvage the data.

Bad Channels

Sometimes there are 'bad' channels in EEG data that don't provide reliable information. It's important to eliminate those from the analysis as soon as possible because preserving that data will have an impact on subsequent analyses.[32] There are a few reasons why a channel might be excluded [34]:

- A malfunctioning of the channel.
- A wrong placement of the electrode or a loss of contact with the scalp.
- Two or more channels were bridged.
- The electrode got saturated.

Bad channels can be noticed even during the extraction process, noticing for example that one of the electrodes detached from the scalp, can be marked as excluded during the experiment.[32] After data gathering, the most frequent method of detecting bad channels to use, is by visualizing the raw data. Through visual observation, is possible to detect noisy signals, or partially static signals.
This operation could be time spending, considering a visual review by the user for every single signal in each time domain portion. That's why is possible to use pre implemented method, which automatically reject bad portions of the data.

Bad Channels Interpolation

Once the bad channels or bad portions of signals have been detected, it's a standard procedure to interpolate data for bad channels using data from good channels, instead of deleting the bad portion. Interpolation is a technique for filling in missing data using existing data.[32] This approach allows not to waste portions of the data, and as a consequence important features for the classification [35].
However, for the purpose of this thesis of data classification, interpolating portions of the data and changing their actual value mixing with informations coming from other sources, could bias the data and cause a poor performance in the classification algorithm. In this project both the approaches were tried, in one case deleting the bad portion and in the other interpolating.
The most common interpolation technique is by spherical splines. This method consists of the following steps [36]:

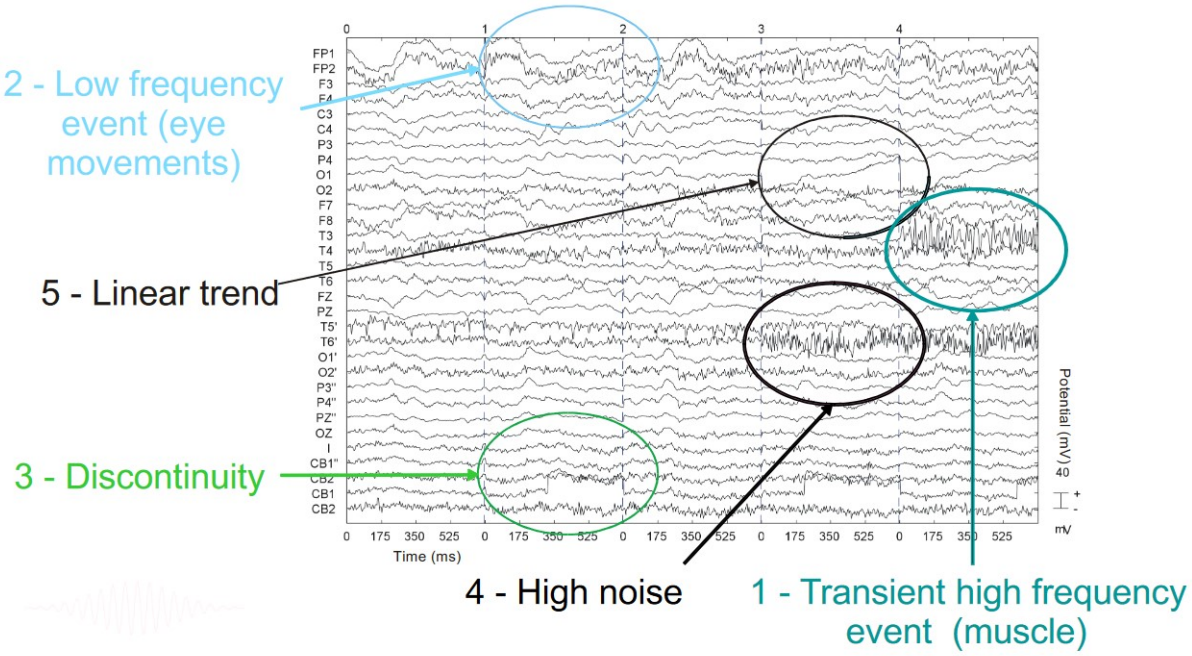

Figure II.2: Illustration of the different types of artifacs (eye bliking and muscle contractions) that could affect the EEG signals as well as the possible environment interferences which cause discontinuity or linear trends in the signals, from [31]

- Project the channel locations onto a sphere which represents the head.
- Describe the relationship between the good and bad channels using a matrix.
- Interpolate the results of the bad channels, using the results coming from the second step.

II.2.1 Filtering

One of the basic operation to be computed on a digital signal, consists in applying a filter. This technique allows acting directly on the frequencies of the signals, removing the ones that are not useful to the analysis [37].
These are all the different kinds of filters that can be applied to the signals:

1. Low-pass filter: frequencies below a certain value are kept ('low' frequencies 'pass'), while high frequencies are removed, considering as high frequencies the ones above a certain threshold.[32]

2. High-pass filter: the same as above, but only high frequencies remain, and only those below a certain value are removed.[32]

3. Band-pass filter: combining the two previous filters, this keeps only frequencies between a lower and upper bound.[32]

4. Notch filter: this filter removes a single frequency. The combination of multiple notch filters can be used to remove a particular set of single frequencies.[32]

In the world of EEG, these are useful for several possible issues when processing.

1. Removing electricity noise: generally the electrical circuits surrounding the measurement will introduce noise in the 50Hz or 60Hz range. To obtain this result a notch filter is applied, to discard that specific frequency.

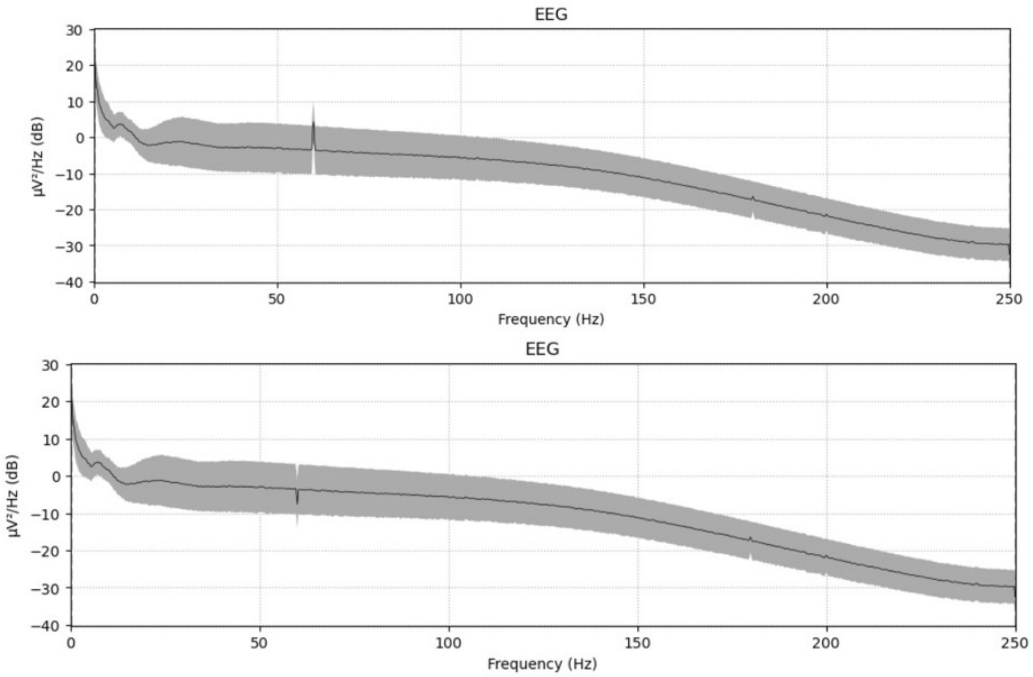

Figure II.2: Illustration realized with the MNE python library using one sample signal of the dataset. Shows the effect of the notch filter applied at the frequency of 60Hz, with a before and after view of the power spectral density of the signal.

2. To perform the analysis separately for each band, performing a band-pass filter between these values removes any noise outside that range. Indeed, the band-pass filter isolates a section of the signal in between two frequencies values. As a consequence it can be applied the band-pass to perform feature extraction for each band frequency.

3. Very low frequency are too slow to originate from the brain, and are usually a sign of long-term drift in the recording environment.[?] For this reason frequencies lower than 1Hz are discarded unsing an high pass-filter.

II.2.2 Artifacts Detection

Artifacts are signals that are picked up by the EEG system but do not actually originate from the brain. The sources of the signals can be both biological or environmental, and sometimes could be difficult to classify their different sources. [38]

1. Outside-world interference, such as power lines, electrodes breaking contact, or other persons moving during the experiment, causes environmental artifacts. Adjusting the environment is the simplest strategy to reduce the impact of those artifacts. A notch filter at 50 or 60 Hz can be used to eliminate power line interference, and some headsets already have this filter built-in. [32]

2. The human body is the source of biological artifacs, and they mix with the actual sources from the brain. Blinks, eye movements, head movements, heart beats, and muscular noises are some of the most prevalent biological artifacts. Having access to other biometric data, such as electrooculogram (EOG) or eye-tracking data for eye movement, accelerometer data for head movement, and electrocardiogram (ECG) data for the heartbeats, it is possible to detect those artifacts.[32]

    - Some of the signals coming from the brain activity can also be considered a source of artifacts. Participants who are tired, will often have large Alpha wave spikes, or due to specific medical treatment of the patients, the activity in specific bands could be biased.[32]

II.2.3 Resampling of Data

Resampling could be useful to reduce computation time without losing informations.[32] Indeed for each EEG, signals are sampled with a certain sampling rate, which indicates the number of recorded points per second. A high number of samples per second, increases the quality and accuracy of the data, but consequently their relative memory space requirements, and time complexity necessary for each operation on them. Applying some of the preprocessing techniques could be highly time demanding, especially if the information for each signal is very accurate and specific, that's why a resampling could be helpful.

II.3 Automatic Preprocessing

If it's necessary to preprocess a big amount of signals could be useful to preprocess automatically all the signals, using several methods available through Matlab software, more specifically using the EEGLAB Library.
EEGLAB is a Matlab toolbox for processing continuous and event-related EEG, MEG, and other electrophysiological data. It includes independent component analysis (ICA), time/frequency analysis, artifact rejection, event-related statistics, and several useful visualisation modes for averaged and single-trial data. For the purpose of this project will be used for the analysis of continuous EEG.

II.3.1 Clean Raw Data

Clean Raw Data is a function available on the EEGLAB library, defined to allow an automatic and reliable preprocessing of the signals. This function allows not only the discard of bad channels and portions of channels, but also a possible interpolation of bad sections using the clean sections of the signals.

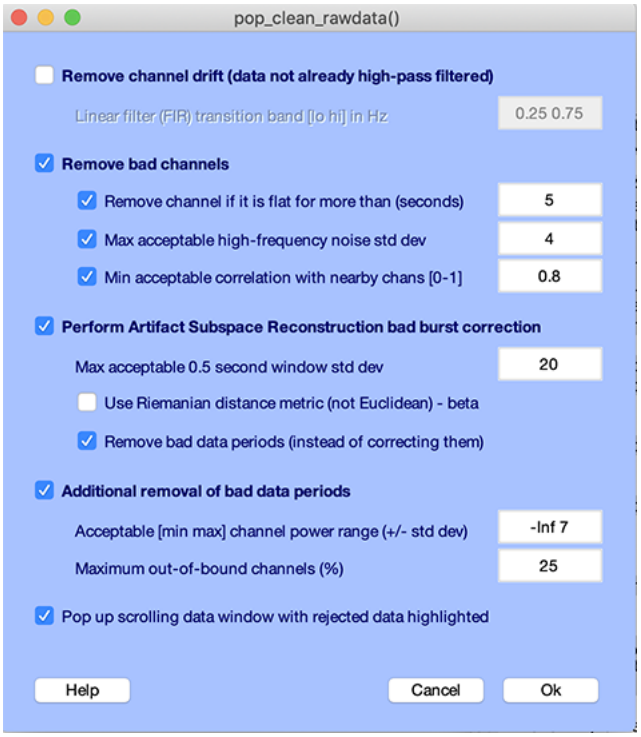

Figure II.3: Screenshot of the Matlab pop up window for the Clean Raw Data window.

Clean Raw Data, methods [41]:

- The top section is about high-pass filtering the data. Usually is applied an high-pass filter to discard all the frequencies below the threshold of 0.5Hz as it would be impossible for the brain to produce so low value frequencies.[41]
- The second option involves the removal of bad channels. Three different options for bad channel removal can be considered.
  - Flat channels may be removed.
  - Channels with a large amount of noise may be removed based on their standard deviation. The standard deviation is a measure of how far the signal fluctuates from the mean.
    The value of the mean for a signal x, composed of x_i from 0 to $N-1$ samples:
    $$\mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i \tag{II.3}$$
    Once computed the mean of the signal, the standard deviation is expressed as:
    $$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1} (x_i - \mu) \tag{II.3}$$
    If the standard deviation is higher than a certain threshold value, it means it's probably affected by some source of noise.
  - Channels, which are poorly correlated with other channels, may be removed.[41] Correlation describes the mutual relationship which exists between two or more signals. Correlation between signals indicates the measure up to which the given signal resembles another signal, and is expressed on a scale [0, 1].
    The threshold for the rejection is channel correlation set to 0.8 by default. Indeed a poor correlation is a symptom that one of the two channels is affected by noise, and this cause the difference between the two signals.
- The Artifact Subspace Reconstruction (ASR) approach is used in the third stage to reject bad data chunks. ASR can be used to either correct or delete bad data segments.ASR determines the standard deviation of PCA-extracted components by finding clean parts of data (calibration data).[41] It discards data regions that exceed more than 20 times the calibration data's standard deviation.
- The fourth option deals with further data rejection based on a predetermined number of channels passing a standard deviation threshold in a given time window, which duration can be chosen. This enables for the rejection of bad data segments that ASR may have missed.[41]

II.3.2 Automatic ICA Label Preprocessing

The ICLabel plugin of Luca PionTonachini is an EEGLAB plugin installed by default with EEGLAB, which provides an estimation of the type of each of the independent components (brain, eye, muscle, line noise, etc.). The ICLabel plugin performs an EEG IC classifier that is reliable and accurate enough to be use in large-scale studies. The current classifier implementation is trained on thousands of manually labelled ICs and hundreds of thousands of unlabeled ICs.

Several Indipendent Components can be detected: [42]

- The activity in brain ICs is thought to arise from locally synchronized activity in one cortical region. The cortical signals are usually tiny and produce dipolar projections onto the scalp that fluctuates smoothly. Power spectrum densities in brain ICs are generally inversely related to frequency and power, with increasing power in frequency bands between 5 and 30 Hz. [42]

- Activity from clusters of muscle motor units is contained in muscle ICs. High broad-band power at frequencies over 20–30 Hz distinguishes them. They can sometimes appear dipolar, similar to brain ICs, but because their origins are outside the skull, their dipolar pattern is considerably more confined than brain sources. [42]

- Eye ICs relate to activity that originates from the eyes and is caused by an electrical dipole within the eye. The projection of this standing dipole to the frontal scalp is shifted by rotating the eyes. [42]

- Heart ICs can be identified in EEG recordings, however they are relatively infrequent. Electrocardiographic (ECG) signals are effectively recorded utilizing scalp EEG electrodes. [42]

- Line Noise ICs capture the impacts of line current noise from surrounding electrical fixtures or EEG amplifiers that aren't properly grounded. Their concentration at 50Hz to 60Hz, depending on voltage local requirements, makes them easily identifiable. However, a single IC is unlikely to be able to represent line noise activity; instead, all components may be contaminated to varying degrees. [42]

- Channel Noise ICs are often an indication of poor signal quality or massive artifacts impacting single channels, and are caused by excessive impedance at the scalp-electrode interface or physical electrode movement. [42]

- Other ICs take into account all ICs that don't fit into any of the other categories. There are two types of ICs in this category: those with indeterminate

noise and those with several signals that ICA decomposition couldn't separate well. [42]

II.3.3 Independent Component Analysis

Independent Component Analysis (ICA) is a technique that separates and localizes independent signals, from different sources, that have been added together by the electrode recording. ICA assumes that signals are static and that signals from different sources are statistically independent, which may not be appropriate for some neural signals [39]. ICA is used to separate components to identify artifacts from eye movements or heartbeats for Artifact Correction. Once performed the ICA separation of certain number of sources, is then possible to individuate the Independent Component of Eye Blinking or Heartbeats quite easily as they present a fixed recognizable shape.[40]
The main steps of ICA Preprocessing are:

1. Centering: the purpose of this step is to center the data by subtracting the mean from all signals.

2. Whitening: this step consists in transforming signals into uncorrelated and then rescaling each signal to be with unit variance. This is done through a Decorrelation step, which makes each signal uncorrelated with the other, followed by the Scaling step.

3. Once the data are whitened, to retrieve the original distributions, is computed a rotation of the axis, obtained with an Uncorrelation Matrix.

In the case of EEG recordings each axis correspond to a electrode, and its value is mixed with the recordings of all the other electrodes.

II.4 Feature Extraction

Features are the most important attribute for a Machine Learning algorithm, the best are the features, the best the machine learning algorithm can learn, and the classification will be then more effective.
From a signal, many different features can be extracted, both in the time domain and frequency domain. For this project was defined that the most suitable feature to extract for signal classification, is the Power associated with the signal.

II.4.1 Power of EEG signals

Spectral analysis is one of the standard methods used for quantification of the EEG. The power spectral density (power spectrum) reflects the 'frequency content' of the signal or the distribution of signal power over frequency. Indeed, the power spectral density of a signal consists in a value, one for each different frequency. This array of values describes the power present in the signal as a function of frequency, per unit frequency.[44]

Spectral analysis of EEG signal is a central part of EEG data analysis, indeed the power spectral density is the main feature used to train the classification Machine Learning algorithm.[43]
One of the most widely used methods to analyze EEG data is to decompose the signal into functionally distinct frequency bands, such as delta (0.5–4 Hz), theta (4–8 Hz), alpha (8–12 Hz), beta (12–30 Hz), and gamma (30–100 Hz).[45] This implies the decomposition of the EEG signal into frequency components, which is achieved through Fourier transforms, and in particular the Fast Fourier Transform, which transform a signal from time domain to its frequency domain. For each frequency corresponds a complex number from which to extract the phase and the amplitude.[46]
To retrieve the power spectral density, is then necessary to compute the squared-magnitude, which returns a periodogram. However this method is not effective with EEG data, as is very unlikely for the signal to be a perfect sum of sine waves, rather changes over time. The classic periodogram requires a stationary signal to perform well, and with EEG signals the results that produce are biased with high variance.[17] The most widely-used method to reduce the effect of variance is Welch's periodogram, which consists of averaging consecutive Fourier transform of small windows of the signal, with or without overlapping. By averaging the periodograms obtained over short segments of the windows, Welch's method allows to drastically reduce this variance. This comes at the cost, however, of a lower frequency resolution. [47]
Once the power spectral density is computed through Welch's periodogram method,

the average power for each different band has to be extracted. The average power consists in the the area occupied by the Welch's periodogram shape in the considered frequencies intervals, for the considered band.

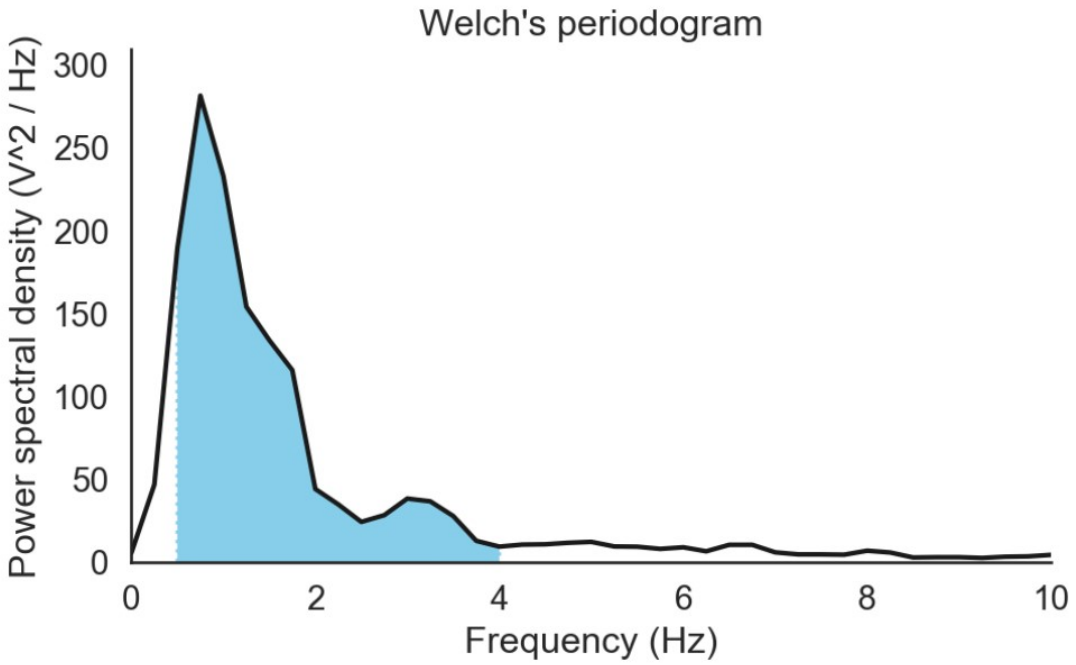

Figure II.4: This illustration, from [17], shows the interval of the delta band in the Welch's power spectral density in the range 0-10Hz. The area with the blue background corresponds to the average total power, and will be computed using the Simpson's method.

The absolute delta power is equal to the blue area of the previous plot, for the delta band, between 1Hz and 4Hz. To approximate the value of this area, is commonly used the Simpson's rule. The area is decomposed into several parabolas, which are summed to approximate its value.[17] This method returns a single value for each different band, which corresponds to the power of the signals for the specified interval. For the Machine Learning purpose of the project, this could be highly effective, as it returns a single value as a key representative feature of an entire band of the signal, which can then easily be fed to the algorithms.

II.5 Machine Learning Methods

To classify the different EEG signals between Parkinson's and healthy patients, some machine learning algorithms were implemented to find the most effective ones. All the Machine Learning methods implemented are supervised. Supervised methods allow the classification of input signals, using a dataset of labelled samples. Through this dataset, the algorithm is trained, in order to recognize and classify correctly in a second time the unseen data.
For the purpose of this project the classification is bimary, as for each sample has to be predicted if is a PD's or healthy subject. In the training dataset is important to have a certain number of samples for each class present, and the number of samples should be balanced, unless the training of the algorithms will be likely biased by the larger one.

II.5.1 How Does it Works a Machine Learning algorithm?

Except for the KNN, all the used algorithms need a training phase during which the corresponding weights and bias have to be tuned in order to reach the smallest cost associated to the prediction, compared to the actual value of the sample. The steps to reach the minimum cost function are described in the section relative to Gradient Descent.
In the following sections will be analyzed the functioning of the machine learning algorithms used and their cost functions.

II.5.2 Logistic Regression

It's a classification algorithm used for classifying categorical response variables. The goal of Logistic Regression is to discover a link between features and the likelihood of a specific outcome.
Binomial Logistic Regression is a type of problem in which the response variable has two values: 0 and 1. The logistic model uses the sigmoid function (denoted by sigma) to estimate the probability that a given sample y belongs to class 1 given inputs X and weights W,

$$P(y = 1|x) = \sigma(W^T X) \tag{II.5}$$

sigma is the sigmoid function, which produces as output a value comprised between 0 and 1 (see Figure II.5), this value corresponds to the probability for the input n to be of class y. Since is a binary classification, the probability for the other class is $1 - p$. If the probability is higher then the threshold value 0.5 the point is considered of class 1, although of class 0. [48]

This formula corresponds to the probability for the sample y_n:

$$y_n = a_n\sigma(a_n) = \frac{1}{1+e^{a_n}} \tag{II.5}$$

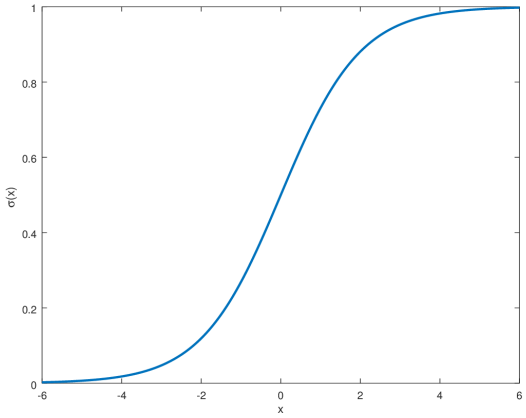

Figure II.5: Shape of the logit function used for the classification in the Logistic Regression method, from [87]

Given that y_n is the prediction for the sample that has to be classified, and t_n the actual value of the sample, the purpose of the algorithm is to maximize the following objective function L:

$$L = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n} \tag{II.5}$$

To avoid numerical exception due to overflows, the logarithm is applied to the function.

$$L = \sum_{n=1}^{N} y_n log(t_n) + (1-y_n)log(1-t_n) \tag{II.5}$$

The purpose of the machine learning algorithm is to find the best set of parameters, this is called Maximum Likelihood Estimation. These sets of regression coefficients can be infinite. The set of regression coefficients for which the probability of getting the data observed is highest is known as the Maximum Likelihood Estimate. Gradient Descent is then applied to the function to retrieve the best values for weights; as Gradient Descent task is to find the minimum of a function, the negative of the objective function is computed. This takes the name of cross-entropy. [48]

$$J = -\sum_{n=1}^{N} y_n log(t_n) + (1-y_n)log(1-t_n) \tag{II.5}$$

II.5.3 K-Nearest Neighbors

K- Nearest Neighbors (KNN) is a Supervised machine learning algorithm as target variable is known. It doesn't make any assumption regarding the distribution data pattern, that's why is considered *nonparametric*.
Differently from all the others presented algorithms, it does not imply any training phase and all the data points will be used only at time of prediction. This means that the prediction step will be more costly, but the training phase very fast.

The K in KNN stands for the most important parameter for this algorithm. When a new sample has to be classified, its label is chosen considering the K closest points in the spatial distribution of the training points. Then the relative label is assigned considering the class with more samples between the K neighbors of the sample to classify. [90]

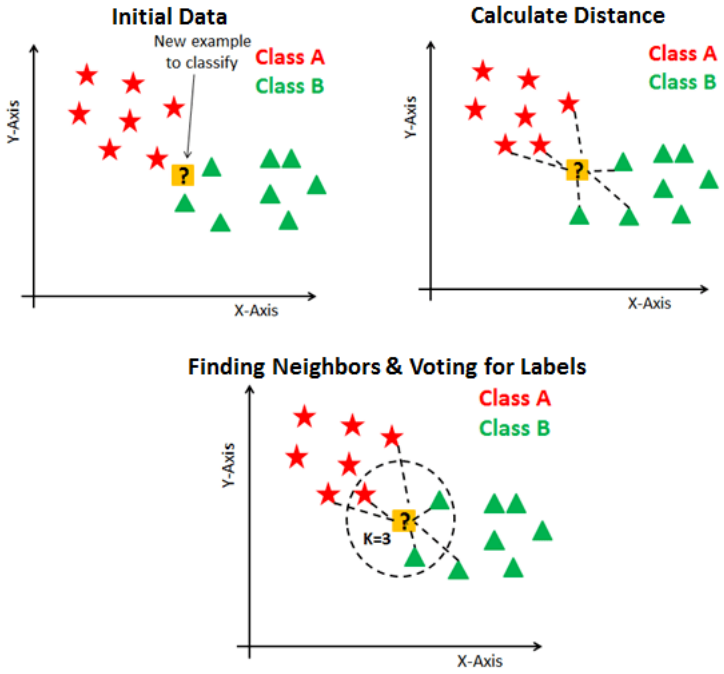

Figure II.5: Illustration of the steps for the classification of a sample using the KNN classification method, using 3 neighbors, from [50]

II.5.4 Support Vector Machine

A Support Vector Machine (SVM) is a supervised discriminative classifier formally defined by a separating hyperplane. Given labeled training data (supervised learning), the algorithm during the training phase computes the best separating hyperplane between the two categorical classes in the feature space.[89]
There are numerous hyperplanes from which to choose in order to separate the two types of data points. The objective of the training phase, through Gradient Descent, is to find a plane that has the maximum distance between data points of both classes.[52] Maximizing the margin distance reinforces the classification, increasing the probability that future data points will be correctly classified. To classify data points, falling on one of the two sides in the spatial representation of the hyperplane, is the discriminative decision boundary.
The hyperplane has the following expression, which corresponds to a line in the feature dimensional space:

$$w^T x + w_0 = 0 \tag{II.5}$$

To maximize the margin of the hyperplane during the training phase, peculiar points of the training data are considered, the support vectors. Support vectors are data points from the two classes that are closer to the hyperplane and have a direct impact on the hyperplane's position and orientation. The margin of the classifier is maximized using these support vectors. [89]
The formula to express the maximization of the margin is

$$max_{w,w_0} \frac{2}{\|w\|} \tag{II.5}$$

Finding the maximum value of weight w and w_0 will give the best margin, given the constraint of correct classification. This constraint is expressed in the following:

$$y^i(w^T x^i + w_0) \geq 1 \tag{II.5}$$

Given that y_i is the class label of a sample (-1 or 1) and $w^T x^i + w_0$ express the relative position of the sample x_i with respect to the hyperplane. The position can be positive, if the sample is spatially located above the separation hyperplane, or negative if it is below.
However is not always possible to find an hyperplane to perfectly separate two classes, as sometimes their spatial representation doesn't allow it, and the constraint cannot be guaranteed. To come over this issue two possible solutions can be adopted: the soft margin and the kernel trick.

Soft Margin

The idea behind the soft margin, is that misclassifications are allowed, but a penalty weight is given to each of them. The new formulation becomes try to minimize:

$$L = \frac{1}{2}\|w\|^2 + C(\#misclassifications) \tag{II.5}$$

This differs from the original objective function in the second term. C is a hyperparameter that determines the trade-off between increasing margin and reducing errors. When C is small, classification errors are less important, and the focus is more on maximising the margin, but when C is large, the focus is on preventing misclassification at the price of maintaining a small margin.
Not all mistakes are equal, indeed data points on the wrong side of the decision boundary that are far should be penalised more than those that are closer. To every data point x_i is associated a variable ξ_i, which is the distance of x_i from the corresponding class's margin. If the point is correctly classified then his value will be zero. As a result, the points on the incorrect side of the margin would be penalised more, based on their distance.

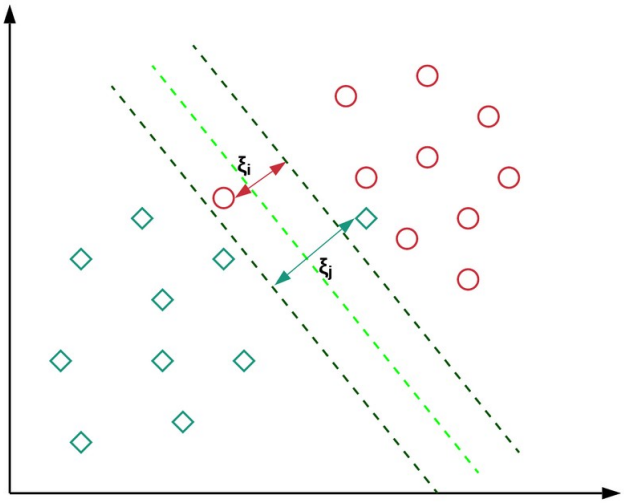

Figure II.5: This illustration shows the best hyperplane that separates two classes in a 2D chart of a non linearly separable dataset. It is shown the penalty cost, xi_i associated to the misclassified points, from [52]

The constraint becomes:

$$y^i(w^T x^i + w_0) \geq 1 - \xi_i \tag{II.5}$$

And the relative function to minimize is:

$$min_w \|w\|^2 + C(\sum_{i \in N^+} \xi_i + \sum_{i \in N^-} \xi_i) \tag{II.5}$$

Kernel trick

The kernel approach enables to work in the original feature space without having to compute the data's coordinates in a higher-dimensional space. The kernel trick offers a more efficient and less expensive way to transform data into higher dimensions, mainly using the dot product between the features of the input data. [53] In an higher dimension then, the same linear separation approach can be used. Basically the kernel function is a dot product of two transformed input vectors.

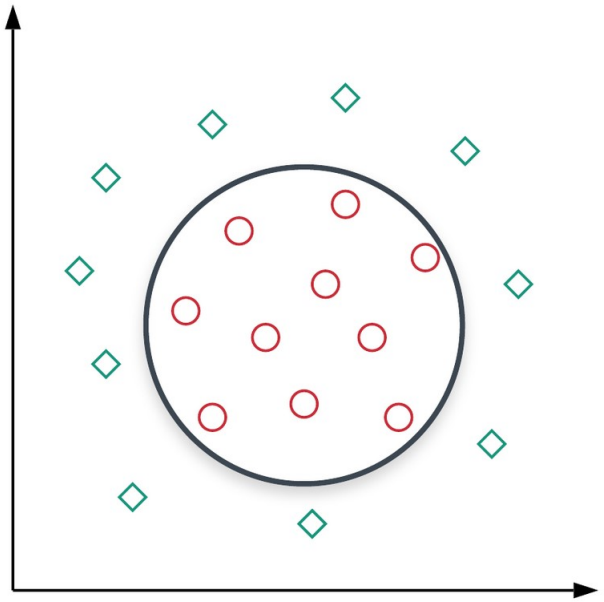

Figure II.5: This illustration, from [52], shows non linearly separable classes, which can be separated using a kernel.

Sometimes, the two approaches of Kernel Trick and Soft Margin has to be mixed to obtain the best solution.

II.5.5 Linear Discriminant Analysis

Linear Discriminant Analysis' purpose is to project features from a higher-dimensional space onto a lower-dimensional space.
This can be achieved in three steps :

1. The first step is to calculate the between-class variance, which is the distance between the mean of different classes, and expresses separability between different classes.

2. Second Step is to calculate the within-class variance, which represents the distance between the mean and sample of each class

3. The third step consists in maximizing the between class variance and minimizing the within class variance, and projecting points on this new lower dimensional space. Indeed the main purpose of LDA is to separate the two classes, projecting the data points in such a way the points of the same class will be close, and the points of different classes far.

The LDA's purpose is basically to maximize the distance between the mean of each class and minimize the spreading within the class itself.[93] However, this formulation is only possible if the dataset has a Normal distribution. This assumption might bring a disadvantage because if the distribution of data is significantly non-Gaussian, the LDA might not perform very well.
This method could be used as well as a dimensionality reduction, like PCA.

II.5.6 Random Forest

Random Forest Classifier is an ensemble algorithm, as it combines more Decision Tree for classifying objects. Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object. The same input will be passed to multiple trees, the output of each tree will be combined, and the class with more than 50% predictions will be the one assigned to the input sample. [54] All the decision trees which compose the forest are trained with different sub-samples of the training data, not to train equal decision trees that will predict all the same output.

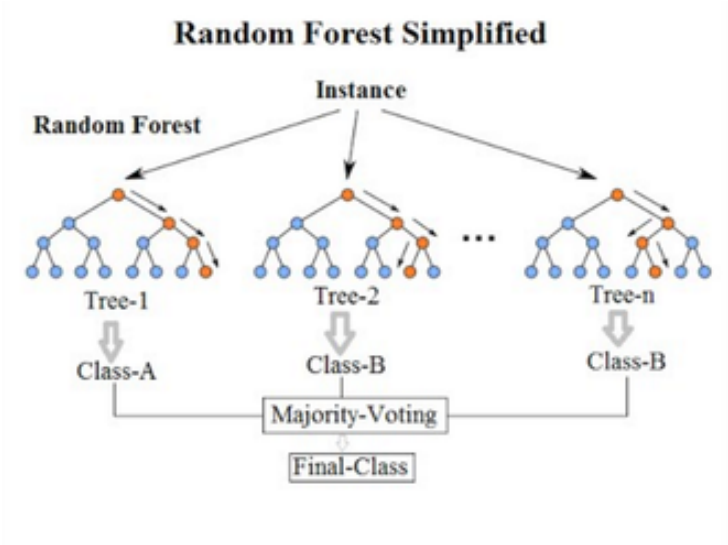

Figure II.5: Random Forest Algorithm mechanism Random Forest Algorithm runs into two stages: the creation and prediction, from [54]

Decision Tree

Decision Tree is a supervised learning technique used for classification problems. Internal nodes represent decisions over dataset features, branches represent decision rules, and each leaf node provides the outcome in this tree-structured classifier.
A decision tree simply asks a question in each decision node, and it further split the tree into subtrees, based on the answer.

The following are the steps to train a decision tree.[57]

1. Begin the tree with the root node which contains the complete dataset.

2. Find the best attribute in the dataset on which to compute the best split. The best feature can be found using an Attribute Selection Measure (Gini Index or Information Gain).

3. The tree is then splitted into two different subtrees, based on the separation feature individuated at the step before. All the samples above a certain threshold value for the chosen feature will be part of a tree, the others will be part of the other.

4. Iteratively makes new decision trees using the subsets of the dataset created in the third step. This process continues until a stage is reached where the nodes contain a number of features below a certain threshold or contain only features of the same class. The node then becomes a leaf, as it corresponds to a specific class.

To classify a sample, it is given as input to the tree, then it follows the path through it considering the splitting made on the features until it reaches a leaf. At that point ,the sample is labelled as the class of that leaf.
The biggest challenge that emerges while developing a Decision tree is how to choose the best attribute for the root node and sub-nodes. A technique known as attribute selection measure, or ASM, can be used to tackle such challenges. The best characteristic for the tree's nodes can be easily determined using this measurement. The two techniques used to select the best features are Information Gain and Gini Index.

Information Gain

Information gain is the calculation of how much information a feature offers about a class by measuring changes in entropy after segmenting a dataset depending on an attribute. Indeed it associates a value to each feature based on:

$$InformationGain_i = EntropyBeforeSplitting - EntropyAfterSplitting_i \quad \text{(II.5)}$$

The attribute of the node that maximizes the value of Information Gain is chosen. [58]

Gini Index

The Gini Index, also known as the Gini Impurity, is calculated by subtracting the sum of each class's squared probability from one and it calculates the probability of a randomly picked feature being erroneously classified. [59]
The Gini Index ranges from 0 to 1, with 0 denoting classification purity and 1 denoting a random distribution of items among multiple classes. A Gini Index of 0.5 indicates that elements are distributed evenly across several classes.

II.6 Feature Scaling

Feature scaling is a method used to normalize the range of features of data of each sample, in the same way.

Why should be considered an important step in data preprocessing?

II.6.1 Gradient Descent Algorithm

All the algorithms previously described, during the training phase try to find the best values for *weights* and *bias* to optimize and reach the minimum of the cost function, iteratively moving in the direction of steepest descent as defined by the negative of the gradient.[60]

The cost function can be represented as a 3D curve, with maximum and minimum values, depending on the parameters, which correspond to the cartesian axes. To find the minimum of the cost function, the derivative of the function tells the direction to follow to move to the minimum, moving for a certain size step. To find the minimum, the value is subtracted from the actual position to find the new one.
The learning rate is the size of these steps. With a high learning rate more ground is covered at each step, however because the hill's slope is always shifting, there's a chance of overshooting the lowest spot. With a very low learning rate, the learning will be slower, but confidently in the direction of the negative gradient since the recalculation is frequent. However, in this second case, there's the risk to be stuck in a local minima, considering it as the global. [61]

Most of the Machine learning algorithms used in this project, use gradient descent as an optimization technique, and so require data to be scaled.
The following is the formula associated to Gradient Descent:

$$\theta_j = \theta_j - \alpha \nabla J(\theta) \tag{II.6}$$

Where J is the associated cost function, and is computed its derivative function. Given the presence of feature value x, which is implied in the formula of the cost function, the step size of the gradient descent will be affected by the variation in feature ranges, which will result in distinct step sizes for each feature.[62] To ensure that the gradient descent progresses smoothly towards the minima and that all of the features' gradient descent steps are updated at the same time, the data are scaled before feeding it to the model. The value α represents the step size. [60] Having features that are on the same scale helps speed up the gradient drop to the minimum.

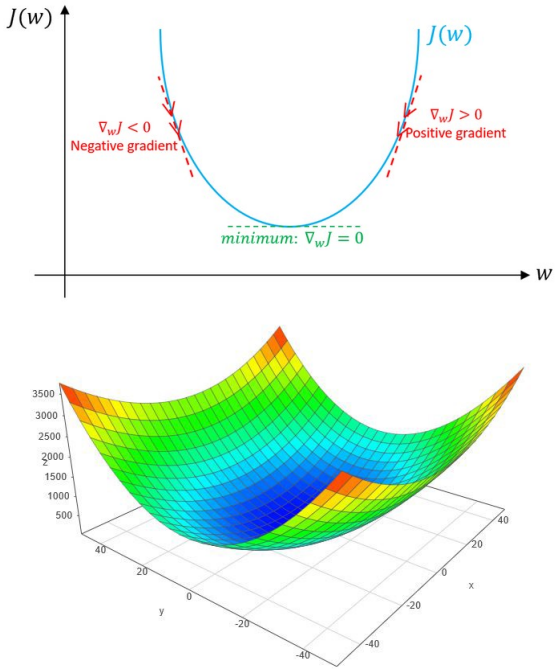

Figure II.6: Illustration in 2D, from [50], and 3D, from [52] of the loss function that the Gradient Descent Algorithm aims to minimize. This task is reached finding the minimum of the function through derivatives.

.

- Scaling the range of features with distance algorithms like KNN, and SVM provides another benefit. This is because these algorithms use distances between data points to infer similarity, and the range of characteristics has the greatest impact on them.[62]

- Tree-based algorithms, on the other hand, are relatively unaffected by feature scale.[62] A decision tree splits a node exclusively on the basis of a single feature.
  The decision tree splits a node based on a feature that increases the node's homogeneity, and this split is unaffected by other features.
  As a result, the remaining features have no effect on the split, which is why they are invariant to the scale of the features.

II.6.2 Normalization

Normalization is a scaling technique that shifts and rescales values to make them range between 0 and 1, it is also known as Min-Max scaler.[62]

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{II.6}$$

Here, x_{max} and x_{min} are the maximum and the minimum values of the feature respectively. [64] When the value of x is the column's minimal value, the numerator is 0, and so x' equals 0. When the value of x is the maximum value in the column, on the other hand, the numerator equals the denominator, and so the value of x' is 1. The value of x' is between 0 and 1 if the value of x is between the lowest and maximum value.
Indeed the Normalization will squash the variables' ranges in between 0 and 1.

II.6.3 Standardization

Another scaling strategy is standardization, in which the values are centered around the mean with a unit standard deviation. As a result, the attribute's mean becomes zero, and the resulting distribution has a unit standard deviation.[62]

$$x' = \frac{x - \mu}{\sigma} \tag{II.6}$$

Feature scaling: *mu* is the mean of the feature values and *sigma* is the standard deviation of the feature values. The values, in this case, are not limited to any fixed range. [64]

II.6.4 Normalization vs Standardization

Which type of feature standardization should be chosen between the two? [64]

- When the data distribution does not follow a Gaussian distribution, normalization is useful. This is useful in algorithms like K-Nearest Neighbors, which do not presuppose any data distribution.
- In situations where the data follows a Gaussian distribution, standardization can be beneficial. Standardization also lacks a boundary range, which means that even if data contains outliers, standardization will has no effect on them.

For this project, the data distribution is not known, then both the standardization and normalization will be taken into consideration.

II.7 Feature Selection

Extracting the power spectral density for each of the 5 bands and each signal, will result in 100 features extracted from each signal. Having a high number of features for each sample could lead to a problem known as course of dimensionality.

II.7.1 Curse of Dimensionality

Curse of dimensionality is an issue related to the presence of a high number of features compared to the total samples available. There is not a strict rule related to the ratio between the number of samples and the number of features for each sample, but is a common approach not to have more features than samples in the dataset. [65]
Having an high number of features for each sample could lead to two main issues:

1. Have more features than observations increases the risk of massively overfitting the model, with consequently in terrible results out of sample performance. [66]

2. When there are too many features, it becomes more difficult to cluster observations because there are too many dimensions, making one observation appear equidistant from the rest.
   This is a serious concern since clustering requires a distance measure like Euclidean distance to estimate the similarity between observations. No meaningful clusters can be created if the distances are all roughly equal. [66]

Feature selection is a process which selects the features contributing the most to classify output features of the machine learning algorithms. The feature selection methods that are proposed in the next section are heuristics since the problem of finding best features as input of machine learning models is NP hard, and there's no specific rule for considering one of the method would perform better than another one. [66]

The perfect Dataset for the prediction of a Machine Learning Algorithm, shows the following features: [66]

1. High Variance: a good model must have features with a lot of variance since they contain a lot of potential information.

2. Uncorrelated: features that are highly correlated with each other are less helpful and, in some situations, outright harmful (when the correlation is so high that it causes multicollinearity); moreover, being correlated means they contain similar information, thus having two for algorithm training is not necessary.

3. Not That Many: low number of features compared to the number of target variable observations. An overfit model that performs badly out of sample would result from having too many features relative to data.

Many methods have been developed to select features and reduce the risk of curse of dimensionality, divided in three categories: Filter Methods, Wrapper Methods, Embedded Methods.

II.7.2 Filter Methods

- This method is done as one of the pre-processing step before passing the data to build a model, so the features are selected blindly to the model.
- Various Statistical test are performed and the feature's are selected on the basis of their score.
- Filter Methods are less accurate but faster to compute, for this reason it is preferable to use filter methods for larger dataset. [70]

Correlation Analysis

To guarantee a low level of correlation between variables this approach is used. Many variables are frequently associated with one another and thus redundant; therefore, deleting one of them will not result in a significant loss of information.[67] The variable to be kept is the one that has a higher correlation coefficient with the target.
The following is the formula for the correlation:

$$r_{xy} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2(y - \bar{y})^2}} \tag{II.7}$$

r refers to to the correlation between features x and y, and N is the total number of samples in the dataset.

Amount of variation

If the feature is mostly all the same value or has a very low variation, then for the model is also hard to learn from it.[68]
In this case the approach is:

1. To account for variables with different scales, standardise all variables or use standard deviation.
2. Drop variables with zero variation (unary).

Correlation with the Target

This method consists in removing variables with a very low connection to the target. If a variable has a very low correlation with the target, it's not going to be useful for building the model and the prediction.[68]

However this approach could miss a useful feature as there might be a feature interaction. Two variables taken singularly could not correlate with the target, however, they are associated with the target after they have been merged into a feature.[69]

II.7.3 Projection Methods

Projection Method is a subcategory of Filter Methods, as it does not take into account the performance of the model in selecting the features. The most common method is Principal Component Analysis, widely used due to his ductility and effectiveness.

Principal Component Analysis

Principal Component Analysis (PCA) is a technique for reducing the number of variables in a data collection while keeping as much information as possible. It creates a set of principal components that are ranked by variance (the variance of the first component is higher than the second, the second component is higher than the third, and so on), uncorrelated, and low in number (lower-ranking components can be discarded because they have minimal signal), as the number of components will correspond to the final number of features kept.[71]
To apply PCA features selection, fixed steps have to be applied on the starting dataset that has to be reduced.

Standardization

This phase is used to normalise the range of continuous initial variables so that their contribution to the analysis is equal. As seen in the previous section, standardization allows creating an equal range for the features of the dataset.[71]

Covariance Matrix Computation

The aim of this step is to check if there is any link between the variables in the input data set and how they differ from the mean with respect to each other. This is achieved by computing the covariance matrix and it helps for the detection of strongly linked variables, as they contain redundant data. [71]
The covariance matrix is symmetric and of shape $p\ x\ p$ (where p is the number of features) and its entries are the covariances associated with the complete set of pairs of the initial variables.

EigenVectors and EigenValues

Principal components are new variables created by combining the initial variables in a linear way. These combinations are made in such a way that the new variables are uncorrelated and most of the information within the initial variables is compressed into the first components.[71]
Principal components, in geometric terms, are the data directions that explain the

most variance, as well as the lines that capture the most data information.[72] The link between variance and information is that the greater the variance carried by a line, the wider the dispersion of data points along it, and the more information it holds.[72]
Principal components are, in fact, those lines on which the projection of dataset points is as large as possible. It's important to note that this technique does not take into consideration the samples' labels. [71]
The eigenvectors of the Covariance matrix are referred to as Principal Components because they are the directions of the axis with the largest variance (information). The coefficients associated with eigenvectors represent the amount of variation held in each Principal Component, and eigenvalues are just the coefficients attached to eigenvectors.
Ranking eigenvectors in order of their eigenvalues, highest to lowest, the principal components are ranked in order of significance too. [71]

Feature Values

The feature vector is essentially a matrix with as many columns as the number of features retained as eigenvectors of the components. This consists in the first step towards dimensionality reduction. In fact, choosing to keep only p eigenvectors (components) out of n, the final dataset will result in having only p dimensions. [71]

Project the Data Along the Principal Components Axes

The aim of this step, which is the last one, is to reorient the data from the original axis to the ones recommended by the principal components using the feature vector formed by the eigenvectors of the covariance matrix. This can be accomplished by multiplying the original dataset's transpose by the feature vector's transpose, resulting in a new feature space with fewer features. [71]

II.7.4 Wrapper Methods

Wrapper methods use greedy search algorithms to examine all possible feature combinations and select the one that delivers the best result for a particular machine learning algorithm.
Testing all possible combinations of the features can be computationally very expensive and this is a drawback of this approach for feature selection, particularly if the feature set is very large, as it is specifically for this project. Moreover this set of features, as it was selected for one specific algorithm, may not be optimal for every other machine learning algorithm.[75]

The wrapper methods inspected in this thesis are of two different types: [74]

- Forward Selection: an iterative method that starts with having no feature in the model. In each iteration, it keeps adding the feature which best improves the model's performances till adding a new variable does not increase the model's performance.[74]
- Backward Elimination: backward elimination starts with the complete set of features and removes the least significant at each iteration, which improves the performance of the model. This process is repeated until no improvement can be shown on the features that have been deleted.[74]

II.7.5 Embedded Methods

Embedded methods combine the advantageous aspects of both Filter and Wrapper methods. Similarly to wrapper methods, select features based on the learning procedure of the Machine Learning model. However, Wrapper methods identify unimportant features iteratively using an evaluation metric, whereas Embedded methods perform feature selection and training of the algorithm in parallel, indeed the feature selection process is an integral part of the classification model. [75]

Wrapper and Filter Methods are discrete processes in the sense that features are either maintained or eliminated; as a result, compared to Embedded Methods, the data variance may be higher.
One example of a widely used embedded algorithm is Lasso.

Least Absolute Shrinkage and Selection Operator

The Least Absolute Shrinkage and Selection Operator (LASSO) is a shrinkage approach that simultaneously conducts variable selection and regularization.
Regularization is a technique for reducing coefficients (weights) to zero and punishing more complex models in order to minimise overfitting. When coefficients are set to zero, a feature selection is performed, and that feature is then eliminated. [75]

II.8 Training and Test Phase

To build a reliable machine learning model, the best approach is to split the whole available samples dataset into the training set, validation set, and test set. Not following this approach means that there will be the risk to face a result biased by the training set.

- The Train Set is the set of data that is used to train and make the model learn the hidden features/patterns in the data.
- The Test Set is a set of data, separate from the training, that is used to finally evaluate the performances of the model on new data, unseen during the training.

It's important that the Test Set is never considered when training the mode, not even for the hyper parameters tuning, and is just used to evaluate the model performances. In this way the model is not affected by the test set, and the results obtained in its predictions are comparable to the ones that could be obtained with data coming from the real world.
In this project the full dataset was splitted with the ratio 80-20, as 80% was used for the training phase and the remaning 20% for the test.

However testing the model directly on the test set, without a previous check of the intermediate result will likely cause a high level of overfitting, as the hyper parameter cannot be properly tuned and the model won't generalize enough.

II.8.1 Generalization

The term "generalization" refers to how well a machine learning model's features apply to specific cases not seen by the model while it was learning. A good machine learning model generalizes well from the training data to any data from the problem domain. This allows making predictions in the future on data the model has never seen.[77]
A model that do not generalize enough leads to overfitting.
When a model learns the information and noise in the training data to the point where it degrades the model's performance on new data, this is known as overfitting. This means that the model picks up on noise or random fluctuations in the training data and learns them as concepts.[77] The issue is that these notions do not apply to new data, limiting the models' ability to generalize.
On the contrary, underfitting refers to a model that can neither model the training data nor generalizes to new data. Overfitting is a much more common error in Machine Learning algorithms, and whether is easy to detect, is not immediate to

solve. The most effective tool to avoid overfitting, in a machine learning algorithm is the k-fold cross-validation.[76]
Another possible approach to mitigate the effect of overfitting is considering not too complex models in the training phase; in this way, the random fluctuations in the training data are not learned too specifically. [76]
To detect a possible overfitting, a common approach consists in testing the prediction of the algorithm not only on the test set, but also on the train set that was used to train the model. In this way, if the performances' metrics for the train are much higher than in the test, could be an evident symptom of over-fit.

II.8.2 k-Fold Cross Validation

To solve the problem related to overfitting, training proceeds on the training set, and evaluation is done on the validation set, which is a portion of the training set, to consider the goodness of chosen hyperparameters, and when the experiment seems to be successful, final evaluation can be done on the test set.[78] This validation process gives information to tune the model's hyperparameters and configurations accordingly, understanding whether the training is moving in the right direction or not.

However, splitting the available data into three sets lowers the number of samples that can be used to train the model, and the results can be influenced by a random selection of the train and validation sets.
Cross-validation (CV for short) is the procedure that can be the solution to this problem. A test set is held out for final evaluation, but there's no more a clear distinction between the validation and the training set, as the latter is split into k smaller sets.
This is the procedure followed for each of the k "folds":[78]

1. A model is trained considering the k-1 of the folds as training data.
2. The resulting model is validated on the remaining fold. It is used as a test set, to check the performance of the model with the considered hyper parameters, and the results are saved.
3. Once all the k folds have been used at least once as a validation set, as overall performance score is considered the average of the different scores.

The average of the values computed in the loop is the performance metric reported by k-fold cross-validation, for each single fold kept as validation. CV is for sure more computationally expensive, but does not waste data as having a fixed validation set size, which is an advantage in this project, where the number

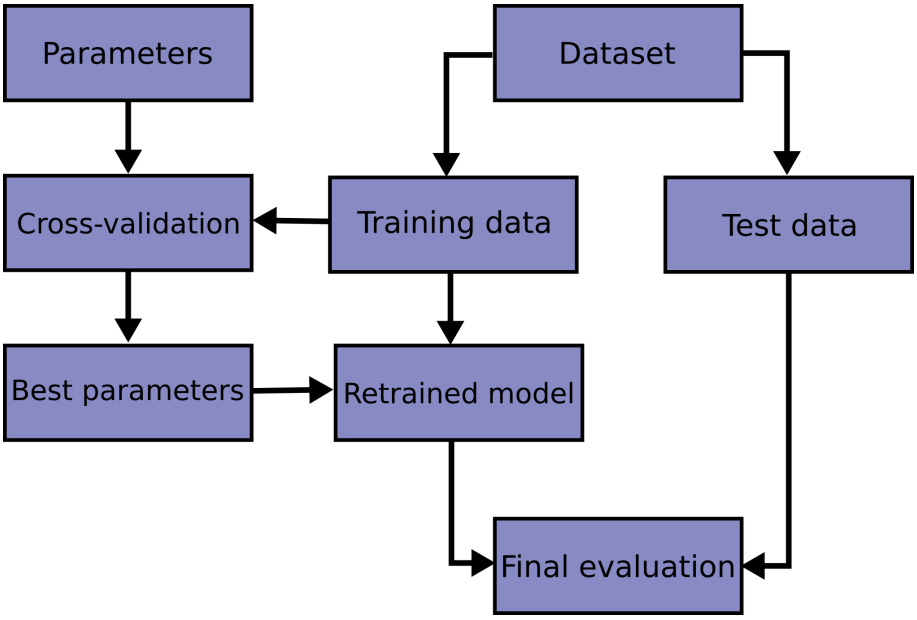

Figure II.8: Steps for the creation of the best possible model to avoid poor performances in the final evaluation due to overfitting or underfitting, using k-fold cross validation for the parameters tuning, from [78]

.

of samples is very small.
Cross-validation is then a powerful resampling procedure used to evaluate machine learning models on a limited data sample. [78]
The process includes only one parameter, k, which specifies the number of groups into which a given data sample should be divided. For this reason, the procedure is also called k-fold cross-validation.
When k is equal to the number of samples in the dataset, it means that at every iteration the model is built on N-1 samples, and evaluated on the remaining one. This leads to a particularly strong algorithm, however this approach increases the computational time required for the training, as a total of N-1 of different models has to be trained, which consequently increases the required computational time. [78]

The models in this project will be trained using a 10-folds cross-validation technique and the Leave One Out approach.

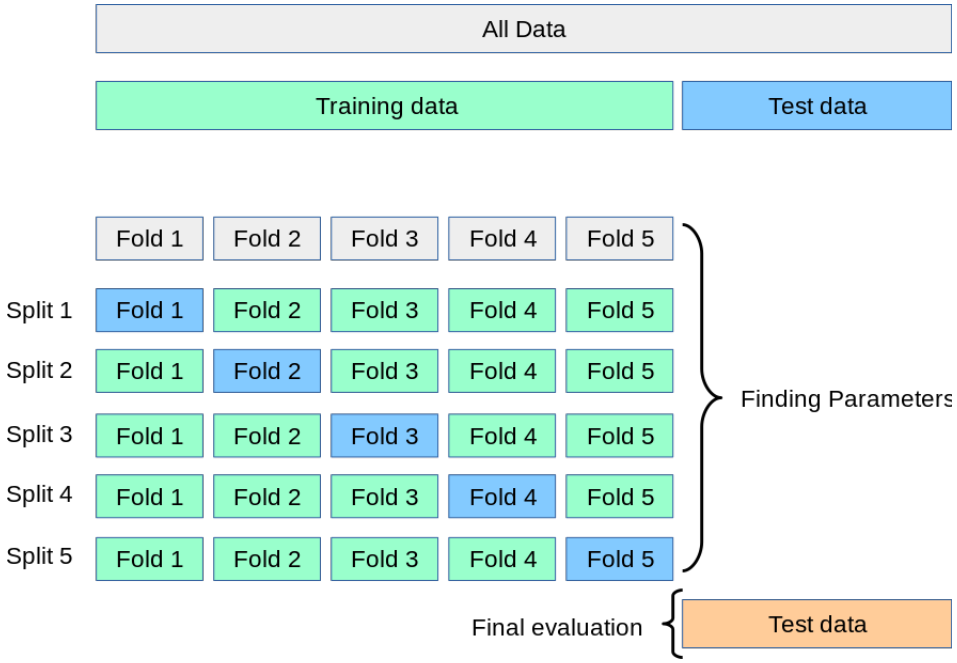

Figure II.8: Illustration of the k-fold cross validation technique for the hyper-parameters tuning, from [78]

II.8.3 Performance Metrics

Once the model has been built, how is it possible to measure its performances first on the validation sets and then on the test set?

All the performance metrics are based on the confusion matrix. A confusion matrix is a technique for summarizing the performance of a classification algorithm. The confusion matrix is divided into 4 different sections, 2 for each possible classification class. Considering one class (A) as the positive one, and the other as negative (B). [79]

1. TP = True Positive, the samples correctly classified as class A
2. TN = True Negative, the samples correctly classified as class B
3. FP = False Positive, the samples wrongly classified as class A
4. FN = False Negative, the samples wrongly classified as class B

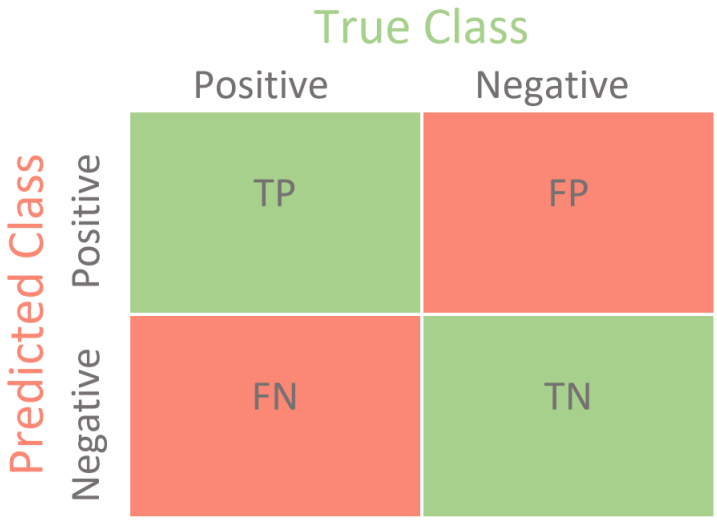

Figure II.8: Visual representation of the confusion matrix, to evaluate the performances of a model, from [79].

A perfect output of the confusion matrix, classifies all the samples as True Positive and True Negative, and none of them is wrongly classified (False Positive and False Negative).
Through the analysis of the confusion matrix and the different spreding of predictions among the four different categories, several scoring functions can be computed.

Accuracy

Accuracy is the starting point for evaluation metrics and the easiest to calculate. It computes the number of correctly predicted samples, above all the possible ones.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{II.8}$$

Classification accuracy alone can be misleading if the training dataset has an unequal number of observations in each class.[82]
Observing the result of accuracy through a confusion matrix can give a better idea of what the classification model is getting right and what types of errors it is making. [81] Considering the example of having an unbalanced dataset between the two classes, it could be achieved an accuracy of 90% or more, but this is not a good score if 90 records for every 100 belong to one class. This means this score could have been achieved by always predicting the most common class value. Observing

the values distribution of the confusion matrix, can be easily detected if a high value of accuracy is obtained due to the imbalanced prediction of the class. Indeed, given that the positive class is the most common one, all the values in the confusion matrix will occupy the True Positive and False Positive positions.
But for an algorithm that only considers numbers, is not easy to detect a wrong configuration in the confusion matrix and will only evaluate the performance on the accuracy value. That's why further performance metrics have to be taken into account.

Precision and Recall

The precision is the ratio:

$$Precision = \frac{TP}{TP + FP} \tag{II.8}$$

where TP is the number of true positives and FP is the number of false positives. Precision is the capacity of the classifier to not label a sample that is negative, as positive. [81]

The recall is the ratio:

$$Recall = \frac{TP}{TP + FN} \tag{II.8}$$

The recall is the measure of the model correctly identifying True Positives.[81]

Precision and Recall are two inverse measures, if the value of one increases, is very likely the other will decrease. That's why is important to choose one of the two metrics and focus on it. In the case of this project, the more useful feature is the precision, given that the positive class is the one of control patients. Indeed is more important to detect as many Parkinson's patients as possible, and keep the value of False Negative classifications low. However it could be interpreted that even the patients wrongly classified as negative (with Parkinson disease), could suffer from another related desease. In this case another performance metric could then be used, to guarantee a precision and recall as high as possible, the F1 score.[83]

F1 score

The F1-score combines the precision and recall of a classifier into a single metric by taking the harmonic mean. It is primarily used to compare the performance of two classifiers. Suppose that classifier A has a higher recall, and classifier B has higher precision. In this case, the F1-scores for both the classifiers can be used to

determine which one produces better results. [84]

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{II.8}$$

Summary

While learning, the machine learning models will tune their hyperparameters considering the one that will produce a better result for the chosen scoring function. To avoid overfitting the whole dataset is splitted in three different sets. Training and Validation set are used to train the model and to chose the best parameters and hyperparameters. If the model parameters, on one hand, are set automatically by the gradient descent technique of the algorithm, on the other hand the model hyperparameters are set manually, in accordance of the performances obtained by the validation set. Once the best performing configuration of hyperparameters is found, the model trained with this ultimate set is tested on the prediction of the Test Set, which was never involved in the construction and validation of the model and will then give the ultimate response on the performance of the algorithm.
Hyperparameters are different for each classification algorithm, and reflect their main characteristics. In the next section will be given an insight on all the possible choices and how to select the best set of hyperparameters among the possible ones for each algorithm.

II.8.4 Hyperparameters Tuning

In this project, to choose the best hyperparameters for each trained model, has been used the method available on Python GridSeacrhCV of the sklearn python library, which will automatically perform a grid search among all the possible hyperparameters, using a cross validation technique on the train set. This method will explore all the combinations of the set of values for hyperparameters specified for a model, and using a k-fold cross-validation technique individuates the best parameters to achieve the best results for the model. These are individuated considering the average score on the chosen metrics, between the k different models evaluated on the trained set. Once the best parameters are found, the model has to be retrained considering the whole training set, not separated from the validation set, and then evaluated on the test set. This allows to reduce the risk of overfitting, using a k-fold cross validation, and automatically performs the validation of the possible parameters.[85]

For each different model several parameters con be chosen, and will be investigated on them in the following.

Class Imbalance

Before inspecting the hyperparameters of each model, it's important to give an insight to this aspect which characterizes almost all the methods. Most machine learning algorithms assume that the data is equally distributed within the two classes. The fundamental issue with class imbalance difficulties is that the algorithm will be biased toward forecasting the majority class and will lack sufficient data to understand the patterns present in the minority class.[80] This problem is common to almost all the algorithms considered for this project, and exists a common solution, which consists in giving different weights, to two different classes, not to bias the learning with the most common one.[86]

Logistic Regression

The following are the main hyper-parameters that is possible to tune, through the sklearn library, for the Logistic Regression: [87]

- Penalty: type of regularization applied to the function, in order to minimize overfitting and reduce its generalization error.
  Adding the regularization parameter to the Logistic Regression, will make its cost function become like this:

$$J = -\sum_{n=1}^{N} y_n log(t_n) + (1 - y_n) log(1 - t_n) + CR(\beta) \tag{II.8}$$

Where R is the regularization function, which depends on *beta*, the coefficients of the function. C is the regularization parameter, and will be investigated in the following.
Two different types of regularization techniques exist, L1 and L2 regularization. For L1 regularization the equation is:

$$L1 = \sum_{i=0}^{N} |\beta_i| \tag{II.8}$$

Whereas in L2 regularization the parameter is:

$$L2 = \frac{1}{2} \sum_{i=0}^{N} \beta_i^2 \tag{II.8}$$

And N is the total number of samples in the dataset.

- C: Inverse of regularization strength. This represents the coefficient of the regularization step seen previously. The higher this value, the smaller contribution is given to the regularization, this means giving a high relevance to the data. On the contrary, with a smaller value of C, more importance is given to the regularization step, and less confidence in the data.

- Solver: options to choose solver algorithm for optimization. Each solver tries to find the parameter weights that minimize a cost function. The default solver often works great in most situations, other solutions can be used for specific occasions, for example in case of very large datasets, but is not the case in this project.

- Class Weight: this parameter is used to overcome the problem of class imbalance. If the option for class weight is *balanced*, the model automatically assigns the class weights inversely proportional to their respective frequencies. To be more precise, the formula to calculate this is:

$$w_j = \frac{N}{\#classes * N_j} \tag{II.8}$$

Where,
w_j is the weight for each class(j signifies the class),
N the total number of samples or rows in the dataset,
$\#classes$ the total number of unique classes in the target,
N_j is the total number of rows of the respective class,

Support Vector Machine Classifier

The following are the main hyper-parameters that are possible to tune, through the sklearn library, for the Support Vector Machine: [88]

- C: as for the previous algorithm, it signifies the inverse of regularization value. The regularization parameter is a value of importance that is given to misclassifications. SVM solves a quadratic optimization problem trying to maximize the margin between both classes and minimizing the number of misclassifications. But in the case of non-separable problems, to find a solution, the misclassification constraint must be relaxed using the relative parameter. This parameter is the coefficient of the value ϵ which corresponds to the distance of the point from its actual margin of the separation hyperplane. The more lambda expands, the fewer incorrectly labelled examples are permitted. When lambda tends to infinite the solution tends to the hard-margin which does not allow miss-classification. When lambda tends to 0 more miss-classification are allowed.[89]
  The formula then becomes:

$$min_w||w||^2 + C(\sum_{i \in C^+} \epsilon^i + \sum_{i \in C^-} \epsilon^i) \tag{II.8}$$

  The same approach is used in the case of a non-linear-kernel. Given this, overfitting could be a consequence of the high value of λ, whereas for lower values of lambda the possibilities of underfitting increase.

- Class Weight: as before class weights can be used to deal with imbalanced dataset, in this case the formula becomes:

$$min_w||w||^2 + C(\beta^+ \sum_{i \in C^+} \epsilon^i + \beta^+ \sum_{i \in C^-} \epsilon^i) \tag{II.8}$$

  Where β are the different weights associated to the classes, based on their relative number of samples in the dataset.

- kernel: Signifies the kernel selection for SVM Machine Learning Algorithm, in order to overcome the problem of non linearly separable class, through the kernel trick. There are several possilbe kernels function that could be used: “rbf” (A very popular kernel, which stands for “radial basis function”), “linear“ (Liner kernel), “poly“ (Polynomial kernel), “sigmoid” (Sigmoid kernel).

K Nearest Neighbours

The following are the main hyper-parameters that are possible to tune, through the sklearn library, for the K-Nearest Neighbours: [90]

- n_neighbor: This is the most important and significant parameter with kNN algorithms as it regulates the number of neighbors that should be checked when an item is being classified, to decide the label based on the most common one represented in between the neighbors.
- weights: This parameter signifies how weights should be distributed between neighbor values, based on their distance from the sample that has to be classified.
    1. "uniform": weights will be distributed equally among all neighbor's values.
    2. "distance": weights will be distributed based on their distance and closer neighbors will have a higher weight in the algorithm.
- algorithm: Signifies the algorithm that will be used to compute nearest neighbors. Choosing one or another of these algorithms won't change the final output but just the speed of the learning process, that's why just one of them will be picked. The difference between the possible algorithms consists in the way training samples and their relative positions are represented, for example with an ordered tree rather than randomly in an array. When a new sample has to be classified, looking for neighbors in an ordered tree is much faster than using a brute force approach in an unordered array.
- metric: The distance metric to use for the distances. The default metric is Minkowski, but can be chosen even Euclidean metric or Manhattan.

Random Forest Classifier

The following are the main hyper-parameters that is possible to tune, through the sklearn library, for the Random Forest Classifier: [91]

- criterion: This parameter allows choosing between two values: gini or entropy. The two parameters usually lead to very similar results, but there is a performance difference as Gini is usually faster since it uses a logarithmic algorithm to compute the entropy.
- splitter: This parameter can be used to define the split strategy and takes two values: best or random.

best will initiate splits on the feature that provides the most information after the split (the best one), instead *random* will chose the splitting on random features. The difference is that best will provide reduced computation and as a consequence more efficiency. Random might be useful when all features provide equal or similar importance to the classification, and the computation needs to be faster.

- max_depth: This parameter is a reference of the maximum depth of the decision tree. When left at default (None), nodes will be expanded until all leaves are pure or they contain samples less than the value of *min_samples_split*. This value can be used to reduce the overfitting of the model, as with a small depth the model is less complex and is likely to generalize more.

- min_samples_split: The minimum number of samples necessary to split an internal node, even this term can be used to reduce or increase the complexity of the model. If the number of samples left on the current node is less than the value of this parameter, the node will be then considered as a leaf, and its label will be the class with a higher number of samples.

- bootstrap: If False, the whole dataset is used to build each tree; in the other case the Bootstrap Aggregation technique is used. Bootstrap Aggregation is a useful procedure to reduce the variance of a decision tree algorithm, as their shape depends on the specific data on which they are trained.[92]
  If the training data is changed the resulting decision tree can be quite different and as a consequence its relative prediction.
  The reason for this lies in the hierarchical nature of the tree classifiers. An error that occurs in a node at a high level of the tree propagates all the way down to the leaves below it.

  And so Bagging (bootstrap aggregating) can reduce the variance and improve the generalization error performance. The basic idea is to create variants of the training set, using bootstrap techniques, by uniformly sampling with replacement. For each of the training set variants, a tree is constructed, and the final decision follows the same approach as the normal decision trees.

- max_features: not all the features should be considered when looking for the best split, but a different subset can be chosen:

  – Auto =
  $$max\_features = \sqrt{n\_features}. \tag{II.8}$$

  – log2 =
  $$max\_features = log_2 n\_features. \tag{II.8}$$

– None =

$$max\_features = n\_features \tag{II.8}$$

- class_weights: same use of previous algorithms, to overcome the problems related to class imbalance.

Linear Discriminant Analysis

The following are the main hyper-parameters that are possible to tune, through the sklearn library, for the Linear Discriminant Analysis: [93]

- solver: solver to use, possible values:

  'svd': Singular value decomposition (default). Does not compute the covariance matrix, therefore this solver is recommended for data with a large number of features. 'lsqr': Least squares solution. Can be combined with shrinkage or custom covariance estimator. 'eigen': Eigenvalue decomposition. Can be combined with shrinkage or custom covariance estimator.

- shrinkage: matrix inversion is a base operation for LDA, and is an extremely sensitive operation. A small amount of noise will be amplified in the matrix inversion, which then is likely to introduce overfitting. Regularizing LDA is a way to overcome this problem. It basically replaces Sw with

  $$S_w = (1 - t)S_w + tI \tag{II.8}$$

  where I is the identity matrix, and t is the regularization parameter (shrinkage constant).

II.9 Summary of Data Management Pipeline

This section contains a summary of the technologies used in the inspected steps for the management and analysis of the data, to lead to the final prediction.
The first step is the preprocessing, and as seen before consists of noise and artifact removal, and ICA preprocessing to individuate the noise sources. These two actions are performed using the EEGLAB Library of MatLab, which allows to perform those two important steps automatically through the functions *Clean_Raw_Data* and *Perform_ICA*.
After these two initial steps on Matlab, each cleaned signal is exported from Matlab in a data format suitable for the MNE library in Python. Indeed, Python is then used for the further steps, through the MNE library, specifically developed for EEG analysis. After the import of the data, the main feature is extracted, which is as seen before, the power spectral density. This is achieved through the MNE function *mne.time_frequency.psd_welch*, which extracts the power spectral density using the Welch method for each signal specified in the function parameter *picks*. This function transforms the signals from the time domain to the frequency domain, computing the total value of the signal for each frequency in the specified frequency interval. The Welch method consists in computing a Fast Fourier Transform with a window of a certain length which moves through the signal length. Using the parameter for the length of the Fast Fourier Transform (FFT) is possible to specify the number of frequencies to consider in the Power Spectral Density.
Once the power spectral density is computed, to retrieve the Average Total Power separately for each band of interest, is used the Simpson method implemented in the NumPy Python library, that for a given array computes the internal covered area, which represents the actual total power. For each patient and for each electrode the total power is calculated for the five different bands of interest. These values are saved in a Data Frame of the pandas library, where each row corresponds to a sample/patient and each column to a feature (i.e. the total power for the specific electrode and the specific band). Before the training of the model, features need to be scaled, and both the approaches are tried with Normalization and Standardization.
After the scaling, to avoid problems related to curse of dimensionality, relevant features are selected. This is done through one of the feature selection methods previously described, comparing the effect of different models' performances.
After the feature selection, the classification models are trained, using the GridSearchCV for the hyperparameters tuning and once the best performing ones are detected, the model is used to predict on the test set, evaluating the actual results of the algorithm, comparing the Precision, Recall, F1-Score and visually analyzing the confusion matrix.

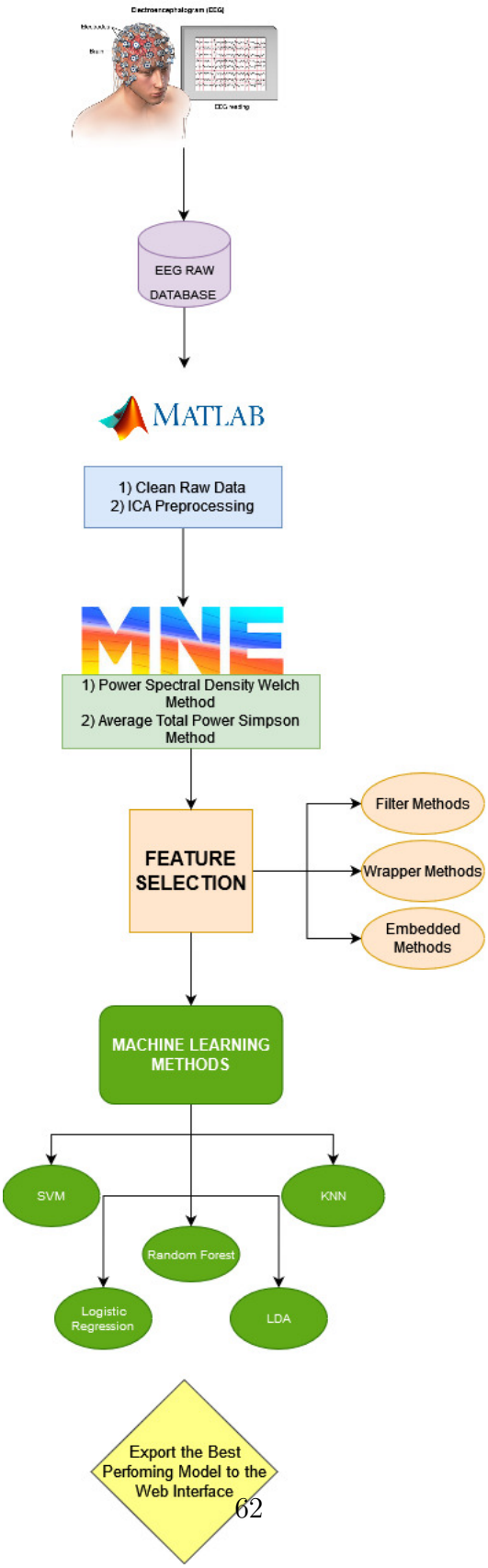

Figure II.9: Visual summary of the pipeline's steps which lead to the creation of the classification model.

Part III

Results of Machine Learning Application

Results

This section contains an overview of the obtained performance's scores, using all the methods previously presented.
Since preprocessing is not a fixed science, is not possible to identify a priori the perfect combination of techniques to achieve the best results. For this reason, a combination of the different techniques is tried, and the results of the best performing ones are presented in the following.

The results presented are the outcome of the algorithm applied on the data already preprocessed with the techniques previously described.
Given the unbalance of the dataset, it is important to correctly label the positive and negative classes, for the evaluation of performances' scores using the confusion matrix and interpreting the accuracy, precision, recall and f1. Indeed in the case of an unbalanced classification, not only the accuracy score, but even the recall and precision would present an high unjustified scores, if the class with more samples in the dataset is considered the positive one. For this reason the positive class is the one associated to control patients (37 samples), and the negative class to the larger dataset of PD patients (91 samples).
Given these assumptions, Precision is considered the most important metric to account for. Indeed, given the formulation of the Precision, a high result implies a small value for False Positive. As the positive class is the one associated to the controls patients, it means having few false predictions for PD subjects, wrongly considered controls.
On the contrary, considering not strictly necessary a high value for the Recall, means having found many False Negative predictions. However wrongly predicting a Parkinson's patient, rather than wrongly predicting him as healthy is less harmful rather than predicting healthy an ill patient. Moreover the wrong classified patients could conceal another illness that caused the alteration of their EEG from the standard recordings of Control patients.

The dataset presents 128 rows (91 PD + 37 Controls) and 301 columns, as it

was computed the average total power for each of the 5 bands for 60 channels + target label column. The channels are just 60 and not 64, which is the number of channels used in the experiment to build the dataset used in this project, because 4 of them were noisy for certain samples and were discarded.
Given the higher number of features compared to the number of samples, to avoid the Curse of Dimensionality presented in the previous chapter, was necessary to proceed with feature selection.

These are the results without applying any sort of dimensionality reduction, using the complete dataset just after a feature range normalization, comparing the prediction's results on the test and train dataset.

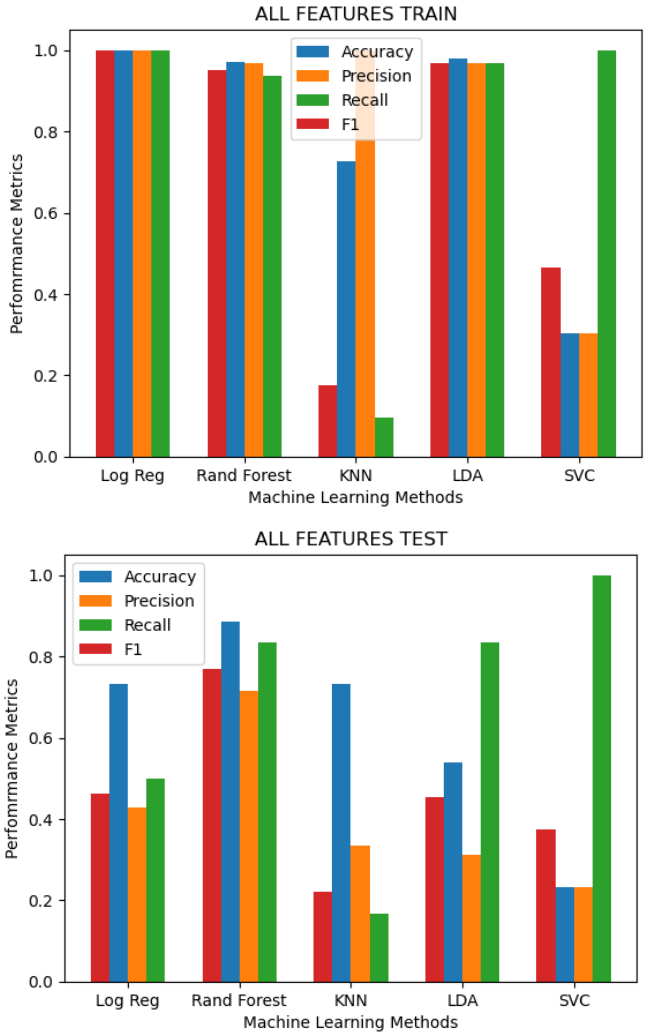

Figure III.0: Resulting char plot of the prediction on the Train and Test set without any technique of dimensionality reduction

As it can be seen from the previous results, an high number of feature leads to an high overfitting, as the results on the train dataset are much better than the ones on the test.
Given the overall performances of Random Forest, accuracy of 85% and F1 close to 80%, it can be considered as a possible model to be uploaded on the application.

In the following sections are presented the three different approaches for feature selection, and the relative performances.

III.1 Filter Methods

III.1.1 PCA Dimensionality Reduction

Even if it's a sub-category of the filters method, PCA is anyway inserted in this section.
The first analysis was made using the PCA method for features selection, both with 50 or 100 features. However it didn't provide any good results, as can be seen in the below figures.

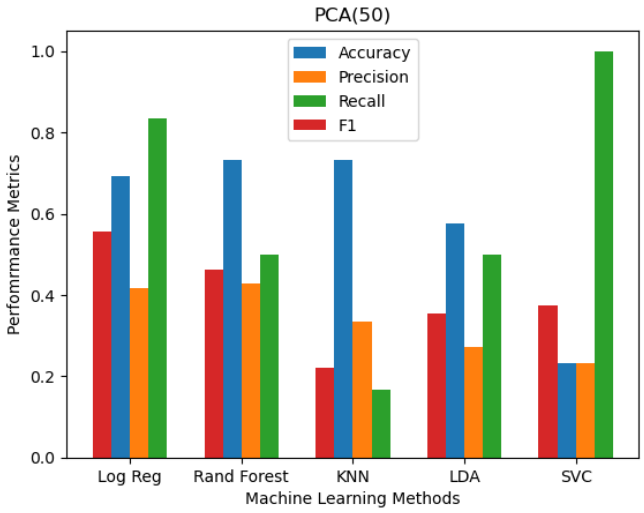

Figure III.1: Illustration of the results obtained with the Principal Component Analysis to reduce the dataset Curse of Dimensionality, comparing the different performance's score metrics, selecting 50 features.

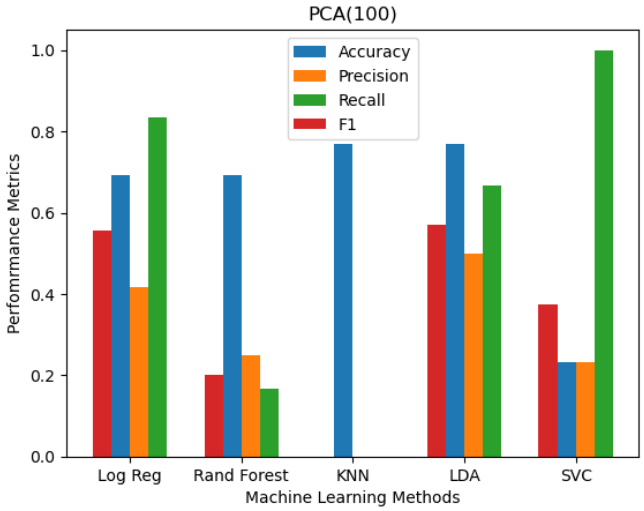

Figure III.1: Illustration of the results obtained with the Principal Component Analysis to reduce the dataset Curse of Dimensionality, comparing the different performance's score metrics, selecting 100 features.

As can be seen, none of the models produced an acceptable result. The higher value for the Recall performances are due to the class unbalance, and due to the fact almost all the controls are misclassified as PD patients.

III.1.2 Remove High Correlated Features

The technique of removing high correlated features is the one that provided the best results in the performance metrics.
First is applied the features reduction technique for removing the ones that are highly correlated within each other, with a threashold value of 90% of correlation for being discarded, after a feature standardization. With this approach, the Random Forest provides good results and the model is one of the selected to be uploaded on the Web App backend predictor. Indeed the accuracy reaches the value of 93% and both Precision and Recall are slightly higher than 80%, as it can be seen in the graph (see Figure III.1), as in the successive relative Confusion Matrix (see Figure III.1). This was the best performance obtained.

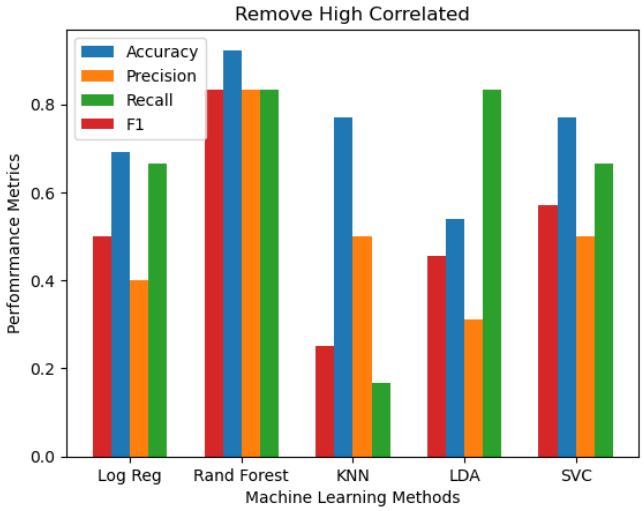

Figure III.1: Illustration of the results applying Standardization and removing the features highly correlated between each other.

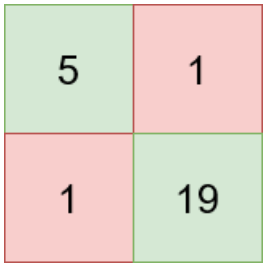

Figure III.1: Representation of the Confusion Matrix obtained for Random Forest. It shows how only two samples were misclassified during the prediction step, one per class.

Further analyses can be done with a previous Normalization/Standardization and a subsequent feature selection, keeping just the top-N variables with an higher correlation with the target. Many trials were made considering different numbers of N features to keep.
In the first chart the kept features were the top 200 with an higher correlation with the target, which provided poor results, due to an high overfitting.

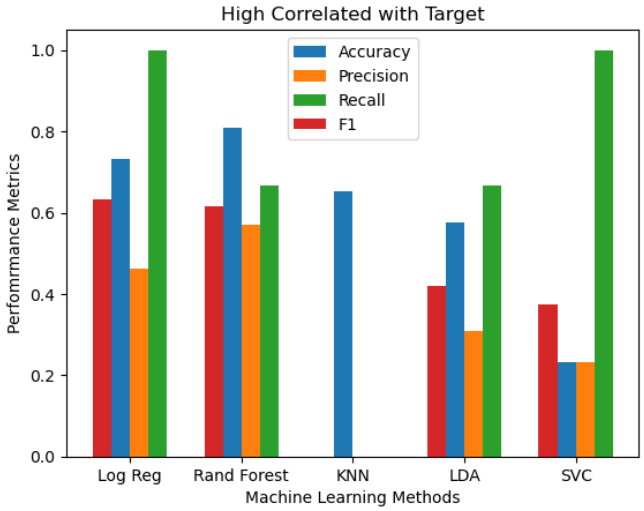

Figure III.1: Char plot of metrics' performances after selecting the 200 features with an higher correlation with the target.

The value of Recall, Precision and F1 of the KNN classifier are at zero such as the value of True Positive (correctly classified Controls), because all the samples are classified as PD subjects. This behaviour is probably due to overfitting in the training phase, which was not possible to solve with the standard anti-overfitting approaches.
Similar results were obtained with 150 features, whereas with 70 the performaces increase, and the overall metrics perform like:

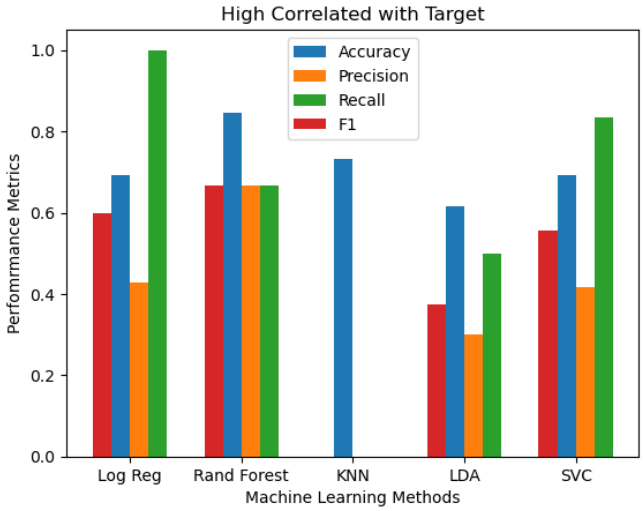

Figure III.1: Char plot of metrics' performances after selecting the 70 features with an higher correlation with the target.

In this case, the values for Precision and Recall for the Random Forest Model are perfectly balanced, even if the overall score is not particularly high.

The hyper parameters of the previous results were tuned using 10 cross-validation techniques. It was even tried the Leave One Out approach, to build a more robust algorithm. It leaded to similar results between all the methods, but not better than with the standard Cross-Validation, and given the higher computational cost associated to this method, it was not further taken into consideration.

The bigger issue related to the the classification was the overfitting. Indeed, many models with different combinations of features selections, gained metrics score >95% on the train set, but poor ones in the test set, even considering all the possible solutions to overcome the overfitting.

III.2 Wrapper Methods

The application of Wrapper Methods is highly time consuming, as it relies on the decision of which features to select, training every time a different model, on the new sub set.
To implement this approach was used the python function "SequentialFeatureSelector" of the sklearn library. This function allows both forward and backward methods changing one parameter. The parameter *n_features_to_select* of the

function is used to select the number of features to consider for the model, which is trained with a k-fold cross validation approach.

III.2.1 Forward and Backward Selection

Given the high computational time required for the training of all the different models, only two models was trained with this approach, the Random Forest Classifier and Logistic Regression, which are the one that provided the best results in the feature selection using filter methods.
Even in this case, as discovered with filter methods, overfitting ruined the final results of the prediction. The scores obtained with the prediction of the train set were at 100%, however the scores in the prediction of the test set were not above 70%, with the 70 best features selected.

III.3 Embedded Methods

For what concerns the embedded methods, Lasso regression was the approach considered to perform the feature selection.
As seen in the section relative to the hyper parameters tuning, the LASSO regression is equivalent to a Logistic Regression with L1 penalty.
The results obtained with however are quite poor, in this case probably due to underfitting. Indeed, both for the training and test datset the results of accuracy are not higher than 75%.

Part IV

Web Application Implementation

Web Interface

Nowadays a huge amount of web interfaces are available on the web, and the personal approach to the world is moved through these. The most important characteristics of an application developed for user interaction, is to dispose of a clear and intuitive interface and the time latency between the user interaction and the application response, is a relevant aspect to take into account.
To allow these qualities to the application, during the implementation the best practices have been followed, in order to provide an application with the best possible experience for the target final users and for its scope.

IV.1 Motivation

To allow a more efficient usage of the developed machine learning algorithm directly by the doctors, part of this project consists in developing a web interface to ease the usage of the potentiality of the computer science tools to someone who is not used to work with raw code, on a Python IDE interface. Indeed, retrieving the results of the prediction for a certain signal through a programming interface and lines of code, could be a challenging task for a doctor, unskilled for this kind of environment. This is the reason why one of the main purposes of this thesis was to develop this tool to ease the access to Machine Learning algorithms, and work with them even if those are hidden behind a easy and ready-to-use user interface. Moreover, this interface is designed in order to be a starting point for the aim of this project, and new tools and technologies can be easily implemented in it. Further research could be done in this field, for example exploring the different levels of stages of Parkinson's disease and not just discerning between PD and healthy subjects.
For these reasons, the technologies used for the development are the most common ones nowadays, with the most powerful development possibilities. This will allow an easy implementation of new features, keeping this interface as the backbone for further projects and studies in this field.

IV.2 Implementation

The web interface consists of 3 different main pages:

1. The first page provides an easy access to the prediction of a signal, in order to predict if the patient who originated that signal is affected by Parkinson's or not. This prediction will be generated using a pre-trained model previously uploaded to the application's backend. This means the signals whose prediction is required, should come from the same source as the ones used to train the model, to retrieve accurate results, as the same features of the model has to be selected even in the new signal.

2. In case a new dataset of signals is available, in the second page is even possible to train new Machine Learning models directly from the web page. Indeed, after uploading all the signals of the patients and healthy subjects, is possible to choose which model to train among the available ones.

3. The own trained models in the second page, are saved in back-end of the application, and is then possible to use them to predict a single signal in the third page of the application.

IV.2.1 Signal Prediction

The first page of the website is the one designed for the prediction of the signals. The user after uploading the patient's EEG, can easily obtain an answer if it's a PD patient or not. Indeed this interface allows to upload a file, in the specified data format, as there are many different equipments for the recording and each of them save the signals in a file with a different extension. The MNE python library, here used to read the files, supports many different data formats, and this is why is necessary to specify it.
The technology used for the file upload is *multer*, which uses the backend of the application as a server and sends there the uploaded files.
Multer is a node.js middleware for handling multipart/form-data, which is primarily used for uploading files. The uploaded files are saved in the backend of the application, in a predefined folder.
After the uploading, is possible to choose if performing or not the ICA preprocessing of the signal. This step will perform the Independent Component Analysis of the signal previously uploaded, using a python code which exploits the MNE library and the pre implemented ICA processing function. At the end of the computation of the ICA preprocessing, two different plots are shown to the user.
The first one shows the distribution of the Independent Components using a visual representation of the scalp, the second shows the shape of the ICs along time. Using

these two plots, the user can then choose which components exclude from the signal, selecting them from the radio buttons that appear in the interface. Differently from the model building, in this case, the Independent Component exclusion is performed manually by the user and not exploiting the automatic steps used with the EEGLAB MatLab library. In fact, is not possible to connect Matlab to a web interface easily and efficiently.

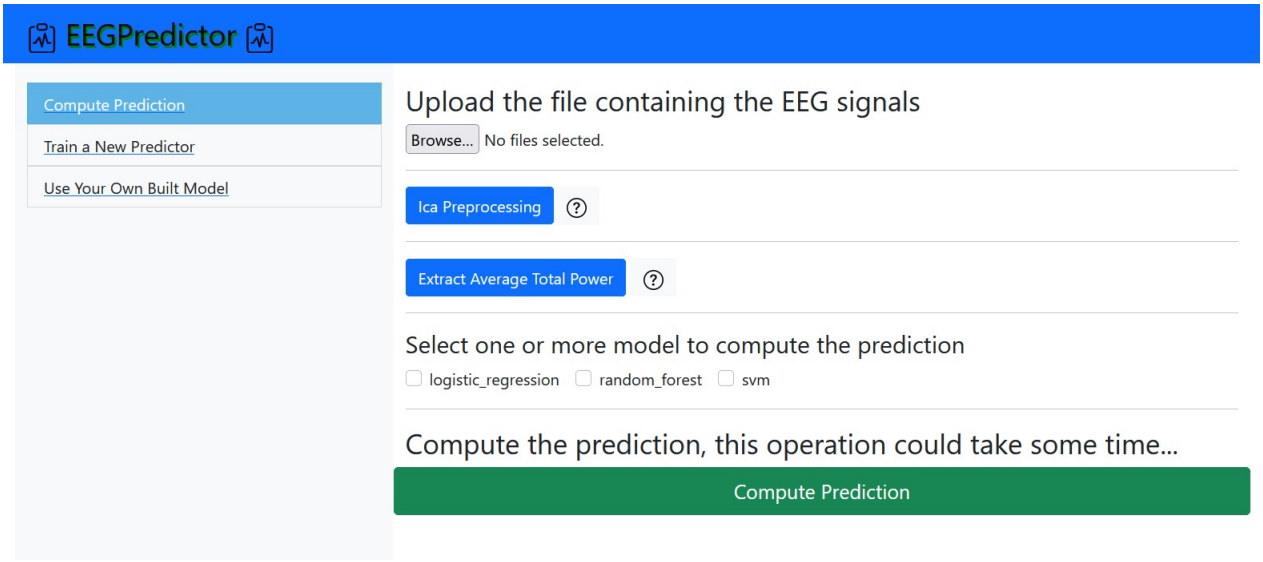

Figure IV.2: Screenshot of the page for the signal prediction, shows all the features implemented.

After this first pre-processing step, is possible to extract the main components of the signal, the average total power for each band, using the power spectral density. This is the feature that will be used by machine learning to discern the nature of the signal. Pressing the button "Extract Power Spectral Density", the signal will be sent to the back-end of the application where a python script will perform the steps to extract the power spectral density. This processing step is still computed using the MNE library. Once the signal has been pre-processed, it's time to compute his prediction. Is possible to compute this step using different pre-trained algorithms already present in the back-end of the application, with a decision of the user. It is important to notice that the used model, must be produced with the training dataset signals coming from the same source of the new signal which was chosen to be analyzed, as the exact same features used to build the model have to be extracted from the signal.
Once the desired models have been chosen is then possible to press the button

"Compute the Prediction", to obtain the expected prediction of the signal, combining the results of the chosen algorithms.
The results will be shown in the frontend, as an overall percentage score comparing the decision of the different algorithms and the single decision for each of them.

IV.2.2 Model Creation

The second page of the web interface allows the creation of new models starting from scratch. In this way is possible to create new models starting from a new dataset of signals, in case different from the one used for the pre-trained models in the application.
Moreover, since the steps for the creation of an algorithm for the analysis and prediction of EEG signals, could be the same even for any type of disease that could be detected through an encephalograms experiment, this page allows creating new algorithms even for the detection of other diseases.
As the previous one, this interface allows the user to upload the controls' and patients' files, using the multer technology. The two different sets of signals, are then saved in two different folders in the backend of the application. On these files will be computed and extracted the Power Spectral Density and subsequently the average power for every different band, to fill the data matrix that will be used to train the algorithm. The user has the possibility to choose the channels of the relative electrodes, for each signal, to analyze or discard. This provides a smaller number of features and a faster algorithm. Some radio check buttons provide the choice of the different algorithms that can be trained, and more than one could be chosen. Once all these decisions have been made, the new models will be computed.

Since MNE does not provide any method for automatic pre-processing of the signals, but each signal should be manually analyzed one after the other, is suggested to pre-process the signals using a Matlab script and the EEGLAB python Library, and then upload them on the application, or use a dataset where a sort of preprocessing was already implemented.
To overcome problems related to the curse of dimensionality, it is set a number of features equal to 50% of the total number of samples. To reduce the number of features down to this value, the used techniques are correlation analysis between the features, and between features and target, which have shown to be the best performing, more reliable and faster.
The new produced models will be then saved in the backend of the application, with the given name and the associated description, inserted in input by the user. To the user is presented as output the performances obtained by these new algorithms, comparing the different metrics score on a char plot. On the base of these results, the user can understand which of the model can be more reliable.
It will be then possible to make use of these new models from the third interface of the web page.

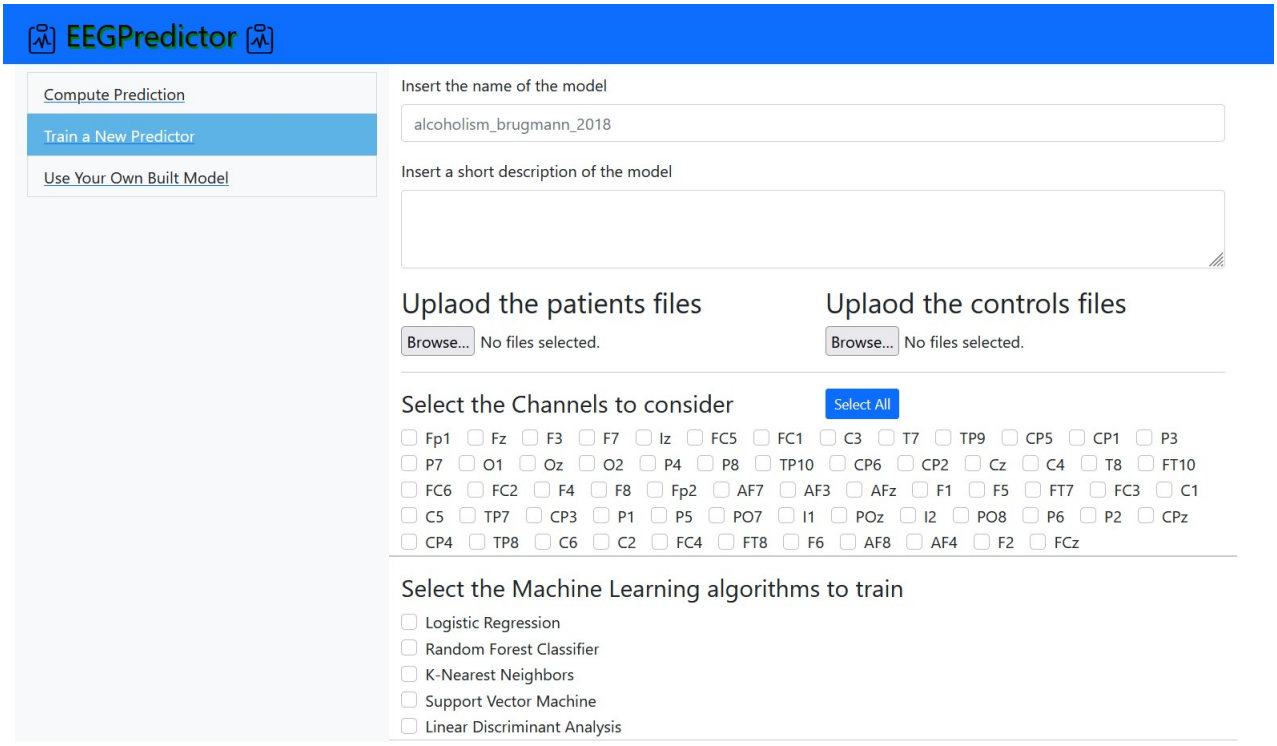

Figure IV.2: Screenshot of the page for the creation of a new model, shows all the features implemented.

IV.2.3 Signal Prediction with Own Models

The models built in the previous section, become available in this section of the web interface. This allows to produce a prediction for a signal, with a model trained with a different dataset and even for a different disease. The interface has the same structure as the section for "Signal Prediction", the only difference is that the available models to retrieve the prediction are the ones produced in the previous section. Again is important to notice that the models and the signals must have the same history, i.e. the dataset used to produce the model must have been created with signals with the same source and structure as the EEG uploaded for the prediction.
A new feature in this interface is the possibility to upload a new model already trained offline on a Python IDE, possibly using more complete techniques of preprocessing and having the possibility of tuning the hyperparameters in a more depth way. Once uploaded, the model can then be used in the application as all the other models.

IV.3 Chosen Technologies

The frontend of the application was developed with Bootrap-React, whereas for the backend the chosen technology was Nodejs. These choices were moved by the necessity to create an application with the best technologies available by now.

IV.3.1 React-Bootstrap

React is a JavaScript-based UI development library run by an open-source developer community. React has to be considered a library rather than a language, but still it is widely used in web development. The library was released in May 2013 and has since grown to become one of the most widely used frontend libraries for web development. [94]

React is today the most widely used front-end development framework. These are the main reasons:

1. Easy creation of dynamic applications: React ease the creation of dynamic web applications because it requires less coding and offers more functionality.

2. Improved performance: React uses Virtual DOM, thereby creating web applications faster. Virtual DOM updates only the items in the Real DOM that were changed, instead of updating all of the components again, speeding the building of the application.

3. Reusable components: Components are the building blocks of any React application, and a single app usually consists of multiple components, which are reusable in the application, and this reduces the time required for the creation of the application.

4. The possibility of using pre implemented objects, makes it easier to understand and learn the language.

Bootstrap is a free and open-source web development framework, created to make the process of developing responsive websites easier by offering a set of syntax for template designs. It is made up of HTML, CSS, and JS-based scripts for a variety of web design functions and components.[94]. Indeed this library contains many different components, optimized for web interfaces, realized with Js, CSS, HTML that can be easily used just importing them from the library and without concerns regarding the way they work.

IV.3.2 Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment. Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser and for this reason very performant.
A Node.js application operates in a single process, rather than starting a new thread for each request, and its standard library includes a set of asynchronous I/O primitives that prevent JavaScript code from blocking.

Instead of blocking the thread and wasting CPU cycles waiting for a response, when Node.js conducts an I/O action, such as reading from the network, accessing a database, or accessing the filesystem, it will restart the operations when the response comes back utilising non blocking techniques. Node.js takes advantage of this behaviour to create faster functions.
This allows Node.js to handle thousands of concurrent connections with a single server without adding the overhead of thread concurrency management, which may be a major source of errors.

Conclusion

In this work, the focus was on finding the links between EEG and a certain psychiatric disease. More specifically, there was an interest in the use of machine learning methods using EEG to classify Parkinson's and healthy subjects. To do so, it was first conducted a literature search on the medical background and functioning of EEG, and on features' bio-markers that could be extracted to study and interpret these diseases in a computer science framework. A dataset containing EEG features studied in the literature research was exploited for the task. Composed of 128 subjects (91 PD and 37 Control subjects), for each subject were recorded 60 channels, and for each signal the power spectral density for the 5 different bands, in total 300 features for each subject. It was then made a research on the classification methods that would have been most suitable for these data.
The first limitation of this work was the number of subjects in the dataset, which if new studies on the same topic were to be initiated, has to be increased, and its unbalance. A balanced dataset would give the possibility of an even learning and avoid the necessity of using a class weight penalty. A higher number of subjects would allow more reliable results and the possibility to train different and more developed learning techniques, such as deep learning methods, that require a higher number of samples to converge.
The highest result obtained for the accuracy was 93% which can be considered an acceptable result. Indeed, even with the mentioned limitations, the results obtained are already very encouraging and strengthen the hypothesis of using machine learning to predict EEG. Indeed this data could be used to help the diagnosis of psychiatric patients and their potential should be exploited to design new treatments and more complex problems.

The web interface developed puts into practice the possibility of using these methods for an automatic diagnosis, with a simple access. The application is intended to be the starting point of this usage and appliance of algorithms, as it even permits the development of new models for different diseases starting from scratch, and applying the best data analysis techniques to train them automatically

without the direct intervention of the user. This second feature can be considered quite ambitious, as the training of a new machine learning model normally requires an active intervention, however results showed how with EEG data this approach can be reliable. Moreover, is an attempt to give at disposal of people without a deep knowledge of data science and machine learning the chance of taking advantage of these technologies autonomously.

Bibliography

[1] Park SM, Jeong B, Oh DY, et al. Identification of Major Psychiatric Disorders From Resting-State Electroencephalography Using a Machine Learning Approach. Front Psychiatry. 2021;12:707581. Published 2021 Aug 18. doi:10.3389/fpsyt.2021.707581

[2] Niedermeyer E.; da Silva F.L. (2004). Electroencephalography: Basic Principles, Clinical Applications, and Related Fields. Lippincott Williams Wilkins. ISBN 978-0-7817-5126-1.[page needed]

[3] https://www.brainlab.org/get-educated/brain-tumors/learn-brain-anatomy-basics/brain-anatomy/

[4] https://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-of-the-brain

[5] Ros, Tomas Baars, Bernard Lanius, Ruth Vuilleumier, Patrik. (2014). Tuning pathological brain oscillations with neurofeedback: A systems neuroscience framework. Frontiers in Human Neuroscience. 8. 10.3389/fnhum.2014.01008.

[6] https://www.simplypsychology.org/forebrain-midbrain-hindbrain.html

[7] Park SM, Jeong B, Oh DY, et al. Identification of Major Psychiatric Disorders From Resting-State Electroencephalography Using a Machine Learning Approach. Front Psychiatry. 2021;12:707581. Published 2021 Aug 18. doi:10.3389/fpsyt.2021.707581

[8] https://qbi.uq.edu.au/brain

[9] https://mayfieldclinic.com/pe-pd.htm

[10] https://www.nia.nih.gov/health/parkinsons-disease

[11] Fahn S, Sulzer D. Neurodegeneration and neuroprotection in Parkinson disease. NeuroRx. 2004;1(1):139-154. doi:10.1602/neurorx.1.1.139

[12] https://www.parkinsonsdaily.com/what-part-of-the-brain-does-parkinsons-affect/

[13] https://www.nhs.uk/conditions/parkinsons-disease/diagnosis/

[14] J. Satheesh Kumar, P. Bhuvaneswari, Analysis of Electroencephalography (EEG) Signals and Its Categorization–A Study, Procedia Engineering, Volume 38, 2012, ISSN 1877-7058, https://doi.org/10.1016/j.proeng.2012.06.298.

[15] E. van Diessen, T. Numan, E. van Dellen, A.W. van der Kooi, M. Boersma, D. Hofman, R. van Lutterveld, B.W. van Dijk, E.C.W. van Straaten, A. Hillebrand, C.J. Stam, Opportunities and methodological challenges in EEG and MEG resting state functional brain network research, Clinical Neurophysiology, Volume 126, Issue 8, 2015, ISSN 1388-2457, https://doi.org/10.1016/j.clinph.2014.11.018.

[16] Kress GJ, Mennerick S. Action potential initiation and propagation: upstream influences on neurotransmission. Neuroscience. 2009;158(1):211-222. doi:10.1016/j.neuroscience.2008.03.021

[17] https://raphaelvallat.com/bandpower.html

[18] Südhof TC, Malenka RC. Understanding synapses: past, present, and future. Neuron. 2008;60(3):469-476. doi:10.1016/j.neuron.2008.10.011

[19] https://qbi.uq.edu.au/brain-basics/brain/brain-physiology/action-potentials-and-synapses

[20] Sidiropoulou K, Pissadaki EK, Poirazi P. Inside the brain of a neuron. EMBO Rep. 2006;7(9):886-892. doi:10.1038/sj.embor.7400789

[21] Klem, GH; Lüders, HO; Jasper, HH; Elger, C (1999). "The ten-twenty electrode system of the International Federation. The International Federation of Clinical Neurophysiology". Electroencephalography and Clinical Neurophysiology. Supplement. 52: 3–6. PMID 10590970.

[22] https://www.ers-education.org/lrmedia/2016/pdf/298830.pdf

[23] https://www.ers-education.org/lrmedia/2016/pdf/298830.pdf

[24] https://eeglab.org/tutorials/ConceptsGuide/Time_frequency_tutorial.html

[25] Nacy, Somer Kbah, Sadeem Jafer, Hind Al-Shaalan, Ibraheem. (2016). Controlling a Servo Motor Using EEG Signals from the Primary Motor Cortex. American Journal of Biomedical Engineering. 2016. 139-146. 10.5923/j.ajbe.20160605.02.

[26] B. Schack, N. Vath, H. Petsche, H.-G. Geissler, E. Möller, Phase-coupling of theta–gamma EEG rhythms during short-term memory processing, International Journal of Psychophysiology, Volume 44, Issue 2, 2002, Pages 143-163, ISSN 0167-8760, https://doi.org/10.1016/S0167-8760(01)00199-4.

[27] Mumtaz W, Vuong PL, Malik AS, Rashid RBA. A review on EEG-based methods for screening and diagnosing alcohol use disorder. Cogn Neurodyn. 2018;12(2):141-156. doi:10.1007/s11571-017-9465-x

[28] https://www.theracycle.com/resources/links-and-additional-resources/updrs-scale/

[29] http://predict.cs.unm.edu/

[30] Singh, A., Cole, R.C., Espinoza, A.I. et al. Timing variability and midfrontal 4 Hz rhythms correlate with cognition in Parkinson's disease. npj Parkinsons Dis. 7, 14 (2021). https://doi.org/10.1038/s41531-021-00158-x

[31] https://sccn.ucsd.edu/githubwiki/files/eeglab2019_aspet_artifact_and_ica.pdf

[32] http://learn.neurotechedu.com/preprocessing/

[33] AUTHOR=Bigdely-Shamlo Nima, Mullen Tim, Kothe Christian, Su Kyung-Min, Robbins Kay A. TITLE=The PREP pipeline: standardized preprocessing for large-scale EEG analysis JOURNAL=Frontiers in Neuroinformatics VOLUME=9 YEAR=2015 URL=https://www.frontiersin.org/article/10.3389/fninf.2015.00016 DOI=10.3389/fninf.2015.00016 ISSN=1662-5196

[34] Kim, SP. (2018). Preprocessing of EEG. In: Im, CH. (eds) Computational EEG Analysis. Biological and Medical Physics, Biomedical Engineering. Springer, Singapore. https://doi.org/10.1007/978-981-13-0908-3_2

[35] https://doi.org/10.48550/arxiv.2009.12244, doi = 10.48550/ARXIV.2009.12244, url = https://arxiv.org/abs/2009.12244, author = Saba-Sadiya, Sari and Alhanai, Tuka and Liu, Taosheng and Ghassemi, Mohammad M., title = EEG Channel Interpolation Using Deep Encoder-decoder Netwoks publisher = arXiv, year = 2020

[36] Dong, L., Zhao, L., Zhang, Y. et al. Reference Electrode Standardization Interpolation Technique (RESIT): A Novel Interpolation Method for Scalp EEG. Brain Topogr 34, 403–414 (2021). https://doi.org/10.1007/s10548-021-00844-2

[37] Alain de Cheveigné, Israel Nelken, Filters: When, Why, and How (Not) to Use Them, Neuron, Volume 102, Issue 2, 2019, Pages 280-293, ISSN 0896-6273, https://doi.org/10.1016/j.neuron.2019.02.039. (https://www.sciencedirect.com/science/article/pii/S0896627319301746) Keywords: filter; distortions; Fourier analysis; time-frequency representation; ringing; causality; impulse response; oscillations

[38] Jiang X, Bian GB, Tian Z. Removal of Artifacts from EEG Signals: A Review. Sensors (Basel). 2019;19(5):987. Published 2019 Feb 26. doi:10.3390/s19050987

[39] N. T. Rachman, H. Tjandrasa and C. Fatichah, "Alcoholism classification based on EEG data using Independent Component Analysis (ICA), Wavelet de-noising and Probabilistic Neural Network (PNN)," 2016 International Seminar on Intelligent Technology and Its Applications (ISITIA), 2016, pp. 17-20, doi: 10.1109/ISITIA.2016.7828626.

[40] http://martinos.org/mne/stable/auto_tutorials/plot_artifacts_correction_ica.html

[**?**] https://www.medicine.mcgill.ca/physio/vlab/biomed_signals/eeg_n.htm

[41] https://eeglab.org/tutorials/06_RejectArtifacts/cleanrawdata.html

[42] Pion-Tonachini, Luca et al. "ICLabel: An automated electroencephalographic independent component classifier, dataset, and website." NeuroImage vol. 198 (2019): 181-197. doi:10.1016/j.neuroimage.2019.05.026

[43] O. Dressler, G. Schneider, G. Stockmanns, E. F. Kochs, Awareness and the EEG power spectrum: analysis of frequencies, BJA: British Journal of Anaesthesia, Volume 93, Issue 6, December 2004, Pages 806–809, https://doi.org/10.1093/bja/aeh270

[44] Zainuddin Lubis, Muhammad Manik, Henry. (2016). SIGNAL PROCESSING FOR POWER SPECTRAL DENSITY (PSD). 10.13140/RG.2.1.2106.2006.

[45] https://eeglab.org/tutorials/ConceptsGuide/Time_frequency_tutorial.html

[46] Hindarto, H., & Sumarno, S. (2016). Feature Extraction of Electroencephalography Signals Using Fast Fourier Transform. CommIT (Communication and Information Technology) Journal, 10(2), 49. https://doi.org/10.21512/commit.v10i2.1548

[47] Z. Sun, H. Chen and Y. Chen, "Application of Periodogram and Welch Based Spectral Estimation to Vortex Frequency Extraction," 2012 Second International Conference on Intelligent System Design and Engineering Application, 2012, pp. 1383-1386, doi: 10.1109/ISdea.2012.689.

[48] Chao-Ying Joanne Peng, Kuk Lida Lee Gary M. Ingersoll (2002) An Introduction to Logistic Regression Analysis and Reporting, The Journal of Educational Research, 96:1, 3-14, DOI: 10.1080/00220670209598786

[49] Pádraig Cunningham and Sarah Jane Delany. 2021. K-Nearest Neighbour Classifiers - A Tutorial. ACM Comput. Surv. 54, 6, Article 128 (July 2022), 25 pages. https://doi.org/10.1145/3459665

[50] https://ai.plainenglish.io/knn-classification-using-scikit-learn-efb34151a8b9

[51] Cristianini N., Ricci E. (2008) Support Vector Machines. In: Kao MY. (eds) Encyclopedia of Algorithms. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30162-4_415

[52] https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe

[53] A. Patle and D. S. Chouhan, "SVM kernel functions for classification," 2013 International Conference on Advances in Technology and Engineering (ICATE), 2013, pp. 1-9, doi: 10.1109/ICAdTE.2013.6524743.

[54] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[55] Azhari, Mourad Alaoui, Altaf Acharoui, Zakia Ettaki, Badia Zerouaoui, Jamal. (2019). Adaptation of the random forest method: solving the problem of pulsar search. SCA '19: Proceedings of the 4th International Conference on Smart City Applications. 1-6. 10.1145/3368756.3369004.

[56] https://towardsdatascience.com/is-lda-a-dimensionality-reduction-technique-or-a-classifier-algorithm-eeed4de9953a

[57] Jijo, Bahzad Mohsin Abdulazeez, Adnan. (2021). Classification Based on Decision Tree Algorithm for Machine Learning. Journal of Applied Science and Technology Trends. 2. 20-28.

[58] Appavu, S., Rajaram, R., Nagammai, M., Priyanga, N., Priyanka, S. (2011). Bayes Theorem and Information Gain Based Feature Selection for Maximizing the Performance of Classifiers. In: Meghanathan, N., Kaushik, B.K., Nagamalai, D. (eds) Advances in Computer Science and Information Technology. CCSIT 2011. Communications in Computer and Information Science, vol 131. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17857-3_49

[59] Frank A. Farris (2010) The Gini Index and Measures of Inequality, The American Mathematical Monthly, 117:10, 851-864, DOI: 10.4169/000298910X523344

[60] https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

[61] P. Baldi, "Gradient descent learning algorithm overview: a general dynamical systems perspective," in IEEE Transactions on Neural Networks, vol. 6, no. 1, pp. 182-195, Jan. 1995, doi: 10.1109/72.363438.

[62] https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

[63] https://medium.com/towards-data-science/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

[64] https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

[65] Venkat, Naveen. (2018). The Curse of Dimensionality: Inside Out. 10.13140/RG.2.2.29631.36006.

[66] https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e

[67] https://www.questionpro.com/features/correlation-analysis.html

[68] https://towardsdatascience.com/how-to-use-variance-thresholding-for-robust-feature-selection-a4503f2b5c3f

[69] https://androidkt.com/find-correlation-between-features-and-target-using-the-correlation-matrix/

[70] Sánchez-Maroño, N., Alonso-Betanzos, A., Tombilla-Sanromán, M. (2007). Filter Methods for Feature Selection – A Comparative Study. In: Yin, H., Tino, P., Corchado, E., Byrne, W., Yao, X. (eds) Intelligent Data Engineering and Automated Learning - IDEAL 2007. IDEAL 2007. Lecture Notes in Computer Science, vol 4881. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-77226-2_19

[71] https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[72] Jolliffe Ian T. and Cadima Jorge 2016 Principal component analysis: a review and recent developmentsPhil. Trans. R. Soc. A.3742015020220150202 http://doi.org/10.1098/rsta.2015.0202

[73] https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d

[74] https://towardsdatascience.com/feature-selection-for-machine-learning-in-python-wrapper-methods-2b5e27d2db31

[75] https://medium.com/analytics-vidhya/feature-selection-for-dimensionality-reduction-embedded-method-e05c74014aa

[76] https://deepai.space/what-is-generalization-in-machine-learning/

[77] https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

[78] https://scikit-learn.org/stable/modules/cross_validation.html

[79] https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

[80] https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights

[81] https://scikit-learn.org/stable/modules/model_evaluation.html

[82] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[83] Ting K.M. (2011) Precision and Recall. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_652

[84] Sokolova, M., Japkowicz, N., Szpakowicz, S. (2006). Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation. In: Sattar, A., Kang, Bh. (eds) AI 2006: Advances in Artificial Intelligence. AI 2006. Lecture Notes in Computer Science(), vol 4304. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11941439_114

[85] https://scikit-learn.org/stable/modules/grid_search.html

[86] https://vitalflux.com/class-imbalance-class-weight-python-sklearn/

[87] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[88] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

[89] https://medium.com/analytics-vidhya/regularization-in- machine-learning-and-deep-learning-f5fa06a3e58a

[90] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors .KNeighborsClassifier.html

[91] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble. RandomForestClassifier.html

[92] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble. BaggingClassifier.html

[93] https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis. LinearDiscriminantAnalysis.html

[94] https://react-bootstrap.github.io/

[95] https://nodejs.org/it/

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**