Pathology detection mechanisms through continuous acquisition of biological signals

by

Jorge Sánchez Casanova

A dissertation submitted by in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Electrical Engineering, Electronics and Automation

Universidad Carlos III de Madrid

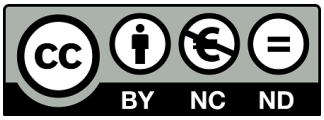
Advisors:

Raúl Sánchez Reíllo
Judith Liu Jiménez

Tutor:

Raúl Sánchez Reíllo

December 2021



A todas las personas que, frente a la adversidad, eligen la pastilla roja;
a la causalidad, sin la cual, nadie podría estar leyendo esto;
a la curiosidad, de no ser por ella, aún contemplaríamos sombras en una caverna.

ACKNOWLEDGEMENTS

Lo sé, quien lo diría. Pero henos aquí, igual que en las grandes historias. Después de casi 50000 palabras (Sanderson, sé que para ti esto no es nada), menos sensato, menos joven y quizá un poco menos indocto. Lo normal en unos agradecimientos es mencionar a las personas que, a lo largo del viaje, te han ayudado de una forma u otra, empero, esto no va a suceder aquí. Si esperas eso, entonces quizá este texto no sea para ti. Esto no va a ser una lista nominativa, es algo que no va conmigo; la gente que me ha ayudado lo sabe. Esta ayuda ha sido en diferentes formas: oportunidades, consejos, ideas, revisiones de texto (tremenda turra), escucharme divagar (otra turra), firmas, cafés, conciertos, abrazos y mil millones de cosas más. A todos vosotros, que en mayor o menor medida habéis participado, que sois el et. al. de esta tesis, gracias.

Bueno dicho esto, allá voy; ¡tremendo viaje! Es una de esas experiencias que te hacen ver que el viaje es mucho más importante que el destino. Ciertamente no ha sido un periplo corto, ha sido un viaje de muchas horas, kilómetros, libros, whiles(1), historias de Ture, malets y mangos que roncan.

Ha sido un viaje que me ha permitido viajar a otros países, además de permitirme vivir durante 6 meses en otro país. Yo, viviendo en otro país y hablando ingles todo el día. Yo, el mismo que dijo malets en inglés, cuando quería decir maletas. En fin, si lo pienso poniéndome en la piel de mi yo de hace 5 años, me lo dicen y no me lo creo. Gracias a eso, conocí a mucha gente increíble, aprendí a insultar en coreano, a decir teta en turco y como no podía ser de otra manera en mí, hice una compra por internet y la mandé a Corea del Norte.

La gente viene y va, y muchos de los amigos de hoy son los conocidos de mañana. Empero, mucha gente perdura beyond the distance. Creo que se me va de intensidad esto, bajo el nivel un poco. Simplemente quiero decir que hay personas que merecen la pena, y que muchas veces, sin pretenderlo te ayudan. Al señor que cree que me puede ganar en una pelea mientras empuño la llama del oeste, aunque en muchas cosas somos opuestos, eres un gran amigo y jamás te libraras de mí, a ti te digo: Nai tiruvantel ar varyuvantel i Valar tielyanna nu vil. A el/la persona/persono del lenguaje/lenguaja neutro/neutra que no veas que turras/turros has aguantado/aguantade, te llevo/lleva en tol peso/pesha. La edad de las tardes de cafetería ha terminado, el tiempo de la diversión ha llegado. Señor POC, TODIL y más nombres absurdos que vendrán, poco que decir que no sepas, así que simplemente diré que, te debo una canción.

Esto podría terminar aquí, but not yet, not yet. quería aprovechar la ocasión de mencionar a mi familia que son todos increíbles (menos tú), yo de mayor quiero ser como ellos. Me hace especial ilusión dedicar unas líneas a mi abuelo, el cual me enseñó muchas cosas y al que parcamente pude ganar jamás al ajedrez. Por lo visto nos parecemos incluso más de lo que alcanzo a saber, ojalá no me equivoque diciendo que estarías orgulloso de

mi, aunque estoy seguro de que seguiría siendo incapaz de ganarte al ajedrez.

Bueno y finalmente mencionar a mi madre, mi padre y mi hermano. Personas que siempre me lo han dado todo a lo largo del espacio y del tiempo. Buena parte del ser que soy hoy es gracias a ellos. Eso es algo que no se puede agradecer lo suficiente, así que simplemente diré que esta tesis es vuestra.

Y hasta aquí he llegado, terminaré esto simplemente diciendo que, aunque a veces no lo parezca, todo lo que tiene un principio, tiene un final.

PUBLISHED AND SUBMITTED CONTENT

- Jorge Sanchez-Casanova, Judith Liu-Jimenez, Paloma Tirado-Martin, Raul Sanchez-Reillo, "Unsupervised and scalable low train pathology detection system based on neural network" in Heliyon, Volume 7, Issue 2.
  - Published.
  - Role: Design and development of both the database and the algorithm and writing of the contribution.
  - Wholly included in the Thesis in Chapter IV.
  - The material from this source included in this thesis is not singled out with typographic means and references.
  - URL: https://doi.org/10.1016/j.heliyon.2021.e06270
- Jorge Sanchez-Casanova, Judith Liu-Jimenez, Pablo Fernandez-Lopez, Raul Sanchez-Reillo, "Recurrent Neural Network for Gait pathology detection" in International Joint Conference of Biomedical Engineering Systems and Technologies.
  - Published.
  - Role: Design and development of both the database and the algorithm and writing of the contribution.
  - Wholly included in the Thesis in Chapter IV.
  - The material from this source included in this thesis is not singled out with typographic means and references.
  - URL: https://www.scitepress.org/Papers/2020/89106/89106.pdf
- Evgenii Kim, Eloise Anguluan, Jeungeun Kum, Jorge Sanchez-Casanova, Tae Young Park, Jae Gwan Kim, Hyungmin Kim, "Wearable transcranial ultrasound system for remote stimulation of freely moving animal" in IEEE Transactions on Biomedical Engineering.
  - Published.
  - Role: Hardware design and sample acquisition.
  - Not included in the Thesis.
  - URL: https://doi.org/10.1109/TBME.2020.3038018
- Paloma Tirado-Martin, Judith Liu-Jimenez, Jorge Sanchez-Casanova, Raul Sanchez-Reillo, "QRS Differentiation to Improve ECG Biometrics under Different Physical Scenarios Using Multilayer Perceptron" in Applied Sciences, 2020; 10(19).

  - Published.
  - Role: Assistance in the problem statement.
  - Not included in the Thesis.
  - URL: https://doi.org/10.3390/app10196896

- Evgenii Kim, Jorge Sanchez-Casanova, Eloise Anguluan; Hyungmin Kim, Jae Gwan Kim, "Mobile Wireless Low-intensity Transcranial Ultrasound Stimulation System for Freely Behaving Small Animals" in International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).
  - Published.
  - Role: Hardware design and sample acquisition.
  - Not included in the Thesis.
  - URL: https://doi.org/10.1109/TBME.2020.3038018

- Jorge Sanchez-Casanova, Ines Goicoechea-Telleria, Judith Liu-Jimenez, Raul Sanchez-Reillo, "Performing a Presentation Attack Detection on Voice Biometrics" in International Carnahan Conference on Security Technology (ICCST)
  - Published.
  - Role: audio samples creation, test execution and partial writing of the contribution.
  - Not included in the Thesis.
  - URL: https://doi.org/10.1109/CCST.2018.8585464

- Jorge Sanchez-Casanova, Antonio Miranda-Escalada, Raul Sanchez-Reillo, Pablo Bartolome-Molina, "ECG biosignals in biometric recognition" in International Carnahan Conference on Security Technology (ICCST), 2017.
  - Published.
  - Role: research and writing of the contribution.
  - Not included in the Thesis.
  - URL: https://doi.org/10.1109/CCST.2017.8167817

- Pablo Fernandez-Lopez, Jorge Sanchez-Casanova, Judith Liu-Jimenez, Carlos Morcillo-Marin, "Influence of walking in groups in gait recognition" in International Carnahan Conference on Security Technology (ICCST), 2017.
  - Published.
  - Role: Assistance in the problem statement.
  - Not included in the Thesis.

  - URL: https://doi.org/10.1109/CCST.2017.8167842

- Pablo Fernandez-Lopez, Jorge Sanchez-Casanova, Paloma Tirado-Martín, Judith Liu-Jimenez, "Optimizing resources on smartphone gait recognition" in IEEE International Joint Conference on Biometrics (IJCB), 2017.

  - Published.
  - Role: Assistance in the problem statement.
  - Not included in the Thesis.
  - URL: https://doi.org/10.1109/BTAS.2017.8272679

ABSTRACT

Pattern identification is a widely known technology, which is used on a daily basis for both identification and authentication. Examples include biometric identification (fingerprint or facial), number plate recognition or voice recognition.

However, when we move into the world of medical diagnostics this changes substantially. This field applies many of the recent innovations and technologies, but it is more difficult to see cases of pattern recognition applied to diagnostics. In addition, the cases where they do occur are always supervised by a specialist and performed in controlled environments. This behaviour is expected, as in this field, a false negative (failure to identify pathology when it does exists) can be critical and lead to serious consequences for the patient. This can be mitigated by configuring the algorithm to be safe against false negatives, however, this will raise the false positive rate, which may increase the workload of the specialist in the best case scenario or even result in a treatment being given to a patient who does not need it. This means that, in many cases, validation of the algorithm's decision by a specialist is necessary, however, there may be cases where this validation is not so essential, or where this first identification can be treated as a guideline to help the specialist. With this objective in mind, this thesis focuses on the development of an algorithm for the identification of lower body pathologies.

This identification is carried out by means of the way people walk (gait). People's gait differs from one person to another, even making biometric identification possible through its use. however, when the people has a pathology, both physical or psychological, the gait is affected. This alteration generates a common pattern depending on the type of pathology. However, this thesis focuses exclusively on the identification of physical pathologies. Another important aspect in this thesis is that the different algorithms are created with the idea of portability in mind, avoiding the obligation of the user to carry out the walks with excessive restrictions (both in terms of clothing and location).

First, different algorithms are developed using different configurations of smartphones for database acquisition. In particular, configurations using 1, 2 and 4 phones are used. The phones are placed on the legs using special holders so that they cannot move freely. Once all the walks have been captured, the first step is to filter the signals to remove possible noise. The signals are then processed to extract the different gait cycles (corresponding to two steps) that make up the walks. Once the feature extraction process is finished, part of the features are used to train different machine learning algorithms, which are then used to classify the remaining features. However, the evidence obtained through the experiments with the different configurations and algorithms indicates that it is not feasible to perform pathology identification using smartphones. This can be mainly attributed to three factors: the quality of the signals captured by the phones, the unstable sampling frequency and the lack of synchrony between the phones.

Secondly, due to the poor results obtained using smartphones, the capture device is changed to a professional motion acquisition system. In addition, two types of algorithm are proposed, one based on neural networks and the other based on the algorithms used previously. Firstly, the acquisition of a new database is proposed. To facilitate the capture of the data, a procedure is established, which is proposed to be in an environment of freedom for the user. Once all the data are available, the preprocessing to be carried out is similar to that applied previously. The signals are filtered to remove noise and the different gait cycles that make up the walks are extracted. However, as we have information from several sensors and several locations for the capture device, instead of using a common cut-off frequency, we empirically set a cut-off frequency for each signal and position. Since we already have the data ready, a recurrent neural network is created based on the literature, so we can have a first approximation to the problem. Given the feasibility of the neural network, different experiments are carried out with the aim of improving the performance of the neural network.

Finally, the other algorithm picks up the legacy of what was seen in the first part of the thesis. As before, this algorithm is based on the parameterisation of the gait cycles for its subsequent use and employs algorithms based on machine learning. Unlike the use of time signals, by parameterising the cycles, spurious data can be generated. To eliminate this data, the dataset undergoes a preparation phase (cleaning and scaling). Once a prepared dataset has been obtained, it is split in two, one part is used to train the algorithms, which are used to classify the remaining samples. The results of these experiments validate the feasibility of this algorithm for pathology detection. Next, different experiments are carried out with the aim of reducing the amount of information needed to identify a pathology, without compromising accuracy. As a result of these experiments, it can be concluded that it is feasible to detect pathologies using only 2 sensors placed on a leg.

RESUMEN

La identificación de patrones es una tecnología ampliamente conocida, la cual se emplea diariamente tanto para identificación como para autenticación. Algunos ejemplos de ello pueden ser la identificación biométrica (dactilar o facial), el reconocimiento de matrículas o el reconocimiento de voz.

Sin embargo, cuando nos movemos al mundo del diagnóstico médico esto cambia sustancialmente. Este campo aplica muchas de las innovaciones y tecnologías recientes, pero es más difícil ver casos de reconocimiento de patrones aplicados al diagnóstico. Además, los casos donde se dan siempre están supervisados por un especialista y se realizan en ambientes controlados. Este comportamiento es algo esperado, ya que, en este campo, un falso negativo (no identificar la patología cuando esta existe) puede ser crítico y provocar consecuencias graves para el paciente. Esto se puede intentar paliar, configurando el algoritmo para que sea seguro frente a los falsos negativos, no obstante, esto aumentará la tasa de falsos positivos, lo cual puede aumentar el trabajo del especialista en el mejor de los casos o incluso puede provocar que se suministre un tratamiento a un paciente que no lo necesita.

Esto hace que, en muchos casos sea necesaria la validación de la decisión del algoritmo por un especialista, sin embargo, pueden darse casos donde esta validación no sea tan esencial, o que se pueda tratar a esta primera identificación como una orientación de cara a ayudar al especialista. Con este objetivo en mente, esta tesis se centra en el desarrollo de un algoritmo para la identificación de patologías del tren inferior. Esta identificación se lleva a cabo mediante la forma de caminar de la gente (gait, en inglés). La forma de caminar de la gente difiere entre unas personas y otras, haciendo posible incluso la identificación biométrica mediante su uso. Sin embargo, esta también se ve afectada cuando se presenta una patología, tanto física como psíquica, que afecta a las personas. Esta alteración, genera un patrón común dependiendo del tipo de patología. No obstante, esta tesis se centra exclusivamente la identificación de patologías físicas. Otro aspecto importante en esta tesis es que los diferentes algoritmos se crean con la idea de la portabilidad en mente, evitando la obligación del usuario de realizar los paseos con excesivas restricciones (tanto de vestimenta como de localización).

En primer lugar, se desarrollan diferentes algoritmos empleando diferentes configuraciones de teléfonos inteligentes para la adquisición de la base de datos. En concreto se usan configuraciones empleando 1, 2 y 4 teléfonos. Los teléfonos se colocan en las piernas empleando sujeciones especiales, de tal modo que no se puedan mover libremente. Una vez que se han capturado todos los paseos, el primer paso es filtrar las señales para eliminar el posible ruido que contengan. Seguidamente las señales se procesan para extraer los diferentes ciclos de la marcha (que corresponden a dos pasos) que componen los paseos. Una vez terminado el proceso de extracción de

características, parte de estas se emplean para entrenar diferentes algoritmos de machine learning, los cuales luego son empleados para clasificar las restantes características. Sin embargo, las evidencias obtenidas a través de la realización de los experimentos con las diferentes configuración y algoritmos indican que no es viable realizar una identificación de patologías empleando teléfonos inteligentes. Principalmente esto se puede achacar a tres factores: la calidad de las señales capturadas por los teléfonos, la frecuencia de muestreo inestable y la falta de sincronía entre los teléfonos.

Por otro lado, a raíz de los pobres resultados obtenidos empleado teléfonos inteligentes se cambia el dispositivo de captura a un sistema profesional de adquisición de movimiento. Además, se plantea crear dos tipos de algoritmo, uno basado en redes neuronales y otro basado en los algoritmos empleados anteriormente. Primeramente, se plantea la adquisición de una nueva base de datos. Para ellos se establece un procedimiento para facilitar la captura de los datos, los cuales se plantea han de ser en un entorno de libertad para el usuario. Una vez que se tienen todos los datos, el preprocesado que se realizar es similar al aplicado anteriormente. Las señales se filtran para eliminar el ruido y se extraen los diferentes ciclos de la marcha que componen los paseos. Sin embargo, como para el dispositivo de captura tenemos información de varios sensores y varias localizaciones, el lugar de emplear una frecuencia de corte común, empíricamente se establece una frecuencia de corte para cada señal y posición. Dado que ya tenemos los datos listos, se crea una red neuronal recurrente basada en la literatura, de este modo podemos tener una primera aproximación al problema. Vista la viabilidad de la red neuronal, se realizan diferentes experimentos con el objetivo de mejorar el rendimiento de esta.

Finalmente, el otro algoritmo recoge el legado de lo visto en la primera parte de la tesis. Al igual que antes, este algoritmo se basa en la parametrización de los ciclos de la marcha, para su posterior utilización y emplea algoritmos basado en machine learning. A diferencia del uso de señales temporales, al parametrizar los ciclos, se pueden generar datos espurios. Para eliminar estos datos, el conjunto de datos se somete a una fase de preparación (limpieza y escalado). Una vez que se ha obtenido un conjunto de datos preparado, este se divide en dos, una parte se usa para entrenar los algoritmos, los cuales se emplean para clasificar las muestras restantes. Los resultados de estos experimentos validan la viabilidad de este algoritmo para la detección de patologías. A continuación, se realizan diferentes experimentos con el objetivo de reducir la cantidad de información necesaria para identificar una patología, sin perjudicar a la precisión. Resultado de estos experimentos, se puede concluir que es viable detectar patologías empleando únicamente 2 sensores colocados en una pierna.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AI - Artificial Intelligence
ANN - Artificial Neural Network
CART - Classification And Regression Trees
CNN - Convolutional Neural Network
DB - Database
DCM - Direct Cosine Matrix
DL - Deep Learning
ECG - Electrocardiogram
EEG - Electroencephalogram
EKF - Extended Kalman Filter
EMG - Electromyogram
FC - Fully Connected
FLR - False Limp Rate
FNR - False Negative Rate
FPR - False Positive Rate
FS - Sampling Frequency
GC - Gait Cycle
GDI - Gait Deviation Index
GRU - Gathered Recurrent Unit
GS - Grid Search
ICA - Independent Component Analysis
IG - Information Gain
IMU - Inertial Measurement Unit
kNN - k Nearest Neighbours
LASSO - Least Absolute Shrinkage and Selection Operator
LDA - Linear Discriminant Analysis
LSTM - Long-Short Term Memory
MDA - Mixture Discriminant Analysis
MEMS - Micro Electro-Mechanical Systems
ML - Machine Learning
NI - Normal Index
NMF - Non-negative Matrix Factorization
NN - Neural Networks
OS - Operating System
PC - Principal Component
PCA - Principal Component Analysis
ReLU - Rectified Linear Unit
RFE - Recursive Feature Elimination

RNN - Recurrent Neural Networks
RS - Random Search
RTFS - Random Trees Feature Selection
RTOS - Real-Time Operating System
SDE - Spectral Density Estimation
SVM - Support Vector Machines

1. INTRODUCTION

Nowadays, many things can be done telematically without leaving home, such as banking, working or attending to conferences. However, when it comes to medical procedures, there is still a long way to go. Indeed, in recent years, several telediagnostic services have become available, but they are not free of limitations. Any pathology that requires more than a mere observation is practically impossible to diagnose, requiring the patient to move to the medical centre. Thanks to the development of technology, specific pathologies can be identified before an expert can diagnose them, as in the case of lower body pathologies.

Most of the research in this field has been carried out in the study of gait and the rehabilitation of a specific pathology. Through these studies, it has been possible to quantify the diseases progression and the success degree of rehabilitation . However, the advance of technology has opened the door for non-medical researchers to investigate the detection of pathologies. This is partly possible thanks to the methods used for this purpose, which are mostly based on machine learning algorithms. These algorithms allow the identification of patterns as long as we have a sufficiently large database with the pathologies represented. In contrast, when gait analysis used to be performed, it had to be carried out by an expert, as the values of the parameters being evaluated had to be understood at all times.

If the algorithms are capable of detecting gait problems, they can even be used in the rehabilitation process of high-performance athletes, where both the recovery process and the athlete reintegration must be as fast as possible. These systems would constantly monitor the athlete activity, assessing whether the movements they perform are within the ranges considered acceptable by the algorithm. In the event of a deviation from these values, the algorithm could issue an alert for the athlete to stop training or rehabilitation.

Although this approach is interesting, in order to achieve this goal, first of all, a pathology detection algorithm must be developed that is as portable as possible. The research in this Thesis has been carried out with this objective in mind. For a better understanding of the research carried out, this document has been structured as follows:

- The first part of the document focuses on giving an overview of the state of the art of gait analysis and the description of the artificial intelligence algorithms used in this Thesis.
  - *Chapter 1* introduces the topics covered in the thesis.
  - *Chapter 2* focuses on exposing the different utilities with which gait analysis has been used, as well as its evolution up to the present.
  - *Chapter 3* explains what artificial intelligence is, as well as the two subfields

of machine learning and deep learning. In addition, the different algorithms used throughout the thesis are presented.

- Following, the document focuses on the creation of a pathology detection algorithm using mobile phones as a signal acquisition system. For this purpose, different machine learning algorithms are evaluated to find the one that offers the best results with the dataset.
  - *Chapter 4* addresses the problem of pathology detection using a single smartphone. Throughout the chapter many smartphone configurations are assessed. Additionally, the database used, the signal processing and the results obtained are presented.
- Next, it is presented a pathology detection algorithm but using a professional motion capture system to acquire the data.
  - *Chapter 5* focuses on the creation of an algorithm based on deep learning for the detection of pathologies using temporal signals. Throughout the chapter, the procedures for the creation of the database, the optimisation of the algorithm as well as the decisions taken to make the algorithm scalable are presented.
  - *Chapter 6* aims to create a pathology detection algorithm based on machine learning by parameterising the signals obtained by the sensors. Throughout the chapter, the experiments carried out with the different machine learning algorithms used are presented with the aim of both increasing the accuracy of the algorithm and reducing to a minimum the information necessary for the detection of pathologies.
- Finally presents the overall conclusions of the Thesis as well as possible future research directions are shown in *Chapter 7*

2. STATE OF THE ART

2.1. Gait

Gait is the name given to the way people walk, which is different from each other. This difference is due to the biological variations between people, such as height, weight, feet size, etc., so we can say that each person walks differently from the others. However, when it comes to single person walks since walking is one of the most performed motor skills, it has an intra-user variability lower than 3 % [1], [2]. Nevertheless, this is only true in cases of normal gait, when it comes to pathological gait, it is different. When a physical or psychological pathology is present, it can lead to changes in gait, which causes an increase in intra-user variability. However, this increased variability can be helpful, as the changes introduced by specific pathologies describe a typical pattern, which can be used to identify them.

Diseases may cause these changes in gait [3] as well as a surgery [4]. On the other hand, they can also be caused by disorders such as Parkinson's [5] or Alzheimer's [6] disease (patients with Parkinson's disease tend to walk with reduced gait speed and sorter strides). In addition, the change in the gait can also be used to predict the risk of falling in older adults [7].

Due to the small variability of the gait, we can state that walking is a pseudo-periodic movement, which means that we repeat almost the same movement all the time. Specifically, walking consists of taking a right step, a left step and repeating; this movement is known as the Gait Cycle (GC). The GCs can be divided into different phases, which can be seen in Figure 1.

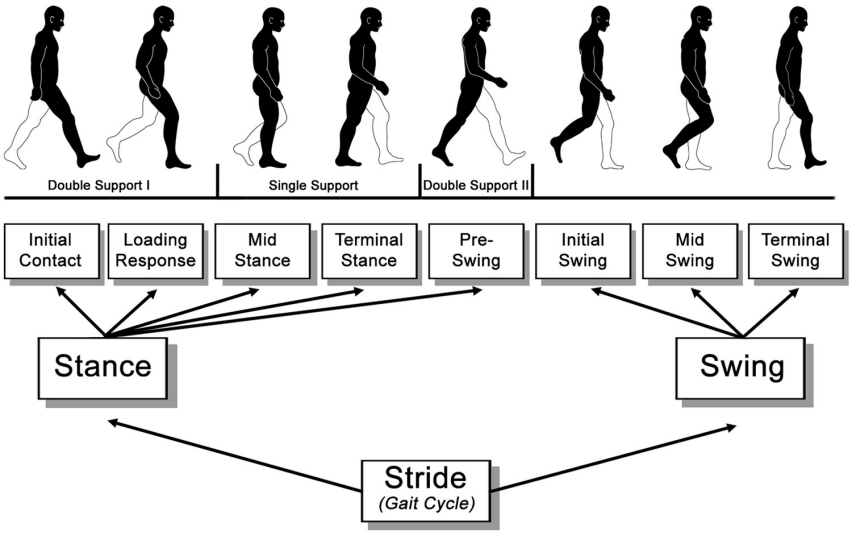


Figure 1: Gait Cycle phases [8].

As it may be seen in Figure 1, CGs have two main phases, the stance phase and the swing phase. The stance phase is when the foot is on the ground, and the swing phase is when the foot is in the air. Another way of dividing the CG is according to the type of support being performed, single support (which can be of the left and right foot) that is when only one foot is on the floor or double support (of which there are two types) that is when both feet are on the floor . Although the GC could be explained in more detail, it is not relevant to this Thesis beyond the explanations already made.

2.2. Gait Analysis

Gait analysis consists of obtaining the characteristic parameters of gait such as speed, stride length, cadence, or muscle strength. These parameters can be used in different research areas such as security, medicine, or sports. There exist diverse technologies used to capture and measure the gait information; according to [9] we can divide them into image-based, floor sensors and wearable sensors.

Image-based systems are the most expensive of all, as they are composed of several cameras to capture the ride from all angles. In this group are the traditional image processing system as well as the based on infrared technology. One technology that brought down the cost of these systems is structured light; one of the best-known systems using this technology is the Kinect sensor, which has been used for a complete gait analysis [10]. The problem with these systems is that they are fixed in place, so they require constant space and compel the user to move to the laboratory to carry out the gait analysis. These systems can give a wide variety of features as velocity, joint angles, cadence, stride length, etc.

The next group of systems are those based on tread pressure. These systems are composed of pressure sensors that are placed along a path over which users walk. These systems are used to obtain the tread pressure. Although these systems are more portable than those based on the video, they are often used in conjunction with the latter, as they do not provide much information on their own.

The last group are wearable sensors, where a wide range of technologies can be included. Here we can find technologies as strips attached to the body to measure the force of the stride or insoles that detect the pressure of the footprint (like those in the previous group). We also can find the electromyogram (EMG), which measure muscle contraction (both voluntary and involuntary); however, the most relevant and used is based on inertial sensors. This technology uses inertial measurement units to measure acceleration, orientation and gravitational forces of the body, which is done using an accelerometer, a gyroscope and a magnetometer. These systems eliminate the restriction of moving to a specific location to perform the analysis.

Of all the gait analysis systems, video-based systems, both infrared-based and image processing-based, are arguably the most widely used in a clinical and professional setting

[11], [12].

Although the gait analysis can be applied to different areas, not all parameters are helpful in all areas. For example, when used for security purposes, the silhouette [13], the floor pressure [14], joint kinetics [15] or even the sound of walking [16] are used. On the other hand, when it comes to sport, joint angles [17] and the muscle forces [18] are more frequently used. Finally, the field of medicine is the least restrictive with the parameters as it uses speed, length, angles, etc., for both physical and psychological problems [19], [20]. Focusing on medicine (since this is the subject of the thesis), we can identify two main fields of gait analysis: pathology rehabilitation and pathology identification.

In the case of rehabilitation, gait analysis has been applied for more than 30 years [21], and there are different areas where it is applied. On the one hand, it is used to evaluate the improvement in the gait of people with some problem or disease that affects the way they walk. This group includes cases such as strokes [22], [23] osteoarthritis [24], [25] or prosthetic implants [26], [27]. On the other hand, this rehabilitation is also given to measure the worsening of gait in degenerative diseases such as Parkinson's [28], Alzheimer's [29], or ageing [30].

In order to efficiently evaluate how the gait changes (either for the better or for the worse), the healthy gait needs to be quantified. There are many studies that perform the feature extraction on the CG [31], [32]; however, there exist other studies that propose the feature extraction after a gait phase detection [33], in this way, more information can be obtained.

Furthermore, the gait analysis in pathology detection can be divided by one fact: the rise of Machine Learning (ML) algorithm popularity. If we search for articles with the keyword "gait pathology detectio" between 2000 and 2017, we get about the same number of articles as if we search between 2017 and 2021, so we can place the turning point in 2017.

Focusing on the period leading up to 2017, we find that there is no standard line of research and the few papers that do, address the problem in different ways. In the first place, we can see the method presented in [34], which proposes a pathology detection system based on the gait phase detection. In this paper, the authors demonstrate that by obtain features from the different GC phases, they can detect the presence of abnormalities in the walk.

On the other hand, Yu et al. [35] presented a method based on the activation patterns of the lower extremity muscles. In this way, they are capable of detecting locomotor control system disorders suggesting a neurological disease. Moreover, Schutte et al. [36] proposed the normalcy index (NI) to characterize the gait of the people. It was stated that the NI could be used to classify the walks depending on whether they were healthy or not; however, the groups to identify different pathologies were overlapped. Finally, Schwartz [37] presents an index to measure the pathological gait called Gait Deviation Index (GDI). They show that using the GDI is possible to classify the pathological

Table 1: State of the art of gait pathology detection using ML algorithms.

Paper	Pathology	Data Used	DDBB	ML algorithm	Accuracy (%)
Bei et al.[38]	True	Features	120	Bayesian	94.2
Guo et al.[39]	True	Features	101	Random Forest	97.5
Dolatabadi et al.[40]	True	Features	40	kNN	>95
Seifert et al.[41]	True	Features	10	Logistic Regression	82.2
Teufl et al.[42]	True	GCs	46	SVM	100
Khoklova et al.[43]	True	GCs	27	RNN	91
Jain et al.[44]	True	GCs	-	CNN	95
Gao et al.[45]	Simulated	GCs	25	LSTM-CNN	93.1
Yin et al.[46]	Simulated	Fragments (1024 points)	15	CNN	93
Potluri et al.[47]	Simulated	Features	10	RNN	97.5
Albuquerque et al.[48]	Simulated	GCs	31	CNN+RNN	91.4

walk from the healthy. However, this feature is only helpful to identify the Hemiplegia, Diplegia, Triplegia and Quadriplegia.

As we can see, few studies focus on detecting gait pathologies, and those that do, do so from a specific approach. In this period, what we found the most are papers presenting gait phases detection with different techniques.

Furthermore, moving to the period beyond 2017, there is a trend towards applying ML algorithms in gait pathology detection problems. These new papers can be divided into two groups: those using a real pathology and those using a simulated pathology. Unlike the above, ML-based papers have a common approach: acquire the data and train the ML algorithm. The main differences between them are the acquisition system used, the features used and the ML algorithm. Among all the articles, we can distinguish between those that extract features and those that use GCs directly to classify walks. The Table 1 shows an overview of the primary gait pathology detection papers.

Analysing the Table 1, it can be seen that the accuracy for the detection of pathologies is in almost all cases higher than 90 %. On the other hand, we can see that neural networks-based algorithms shows higher accuracy. However, one of the biggest problems is the databases, which in many cases do not have enough samples to consider the results presented as valid. Although most of the papers work with real pathologies, the incursion of papers using simulated pathologies implies that they have been done from a more technical profile than the traditional medical profile.

3. ARTIFICIAL INTELLIGENCE ALGORITHMS USED

Artificial intelligence (AI) can be defined as the ability of a machine to achieve cognitive functions associated with the human brain, such as learning, sensing, reasoning, or interacting with the environment. However, as shown in Figure 2 both Machine Learning and Deep Learning (DL) are subfields within AI. On the other hand, we can define ML as the machine's ability to learn by themselves using a set of data, changing and adjusting the parameters as they process the information and learn about it. Deep learning is a group of ML algorithms composed of several layers through which high-level information is extracted to make a decision finally.

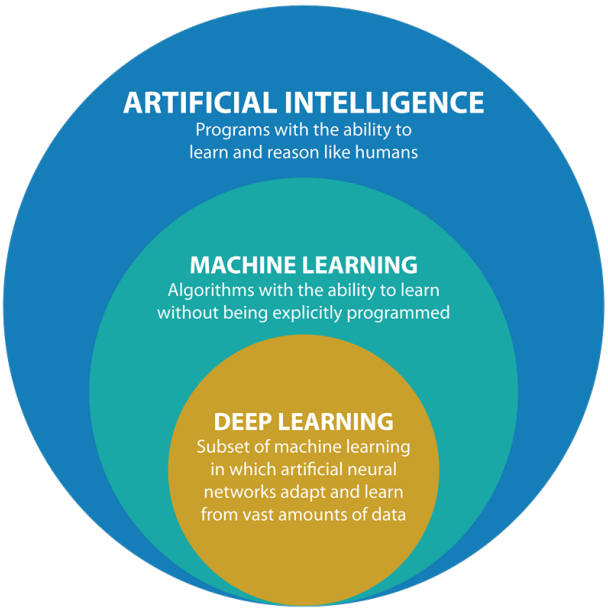


Figure 2: Differences between artificial intelligence, machine learning and deep learning.

3.1. Machine Learning

Machine Learning is an AI discipline that provides algorithms that automatically improve through the experience gained from studying a dataset. These algorithms have the ability to make decisions (e.g., data predictions or classifications) without being explicitly programmed to do so. Depending on the data, ML algorithms can be grouped into three categories; an overview of the different ML algorithms and their use is shown below:

- **Supervised learning** is when the information used to train is labelled, and the objective is to learn the pattern of the input data. This algorithm produces an inferred function that makes predictions based on the input data. Depending on the nature of the output, the algorithm can be used for classification or regression.

- **Unsupervised learning** is when the input data has no labels. The algorithm on its own must find a way of grouping the data. The method can be used for clustering or anomaly detection.

- **Reinforcement learning** is based on an agent that interacts with its environment making decisions. Depending on the decisions made, the agent receives a reward that can be positive or negative. The rewards are positive when the agent behaves as desired and negative when he does not, the objective of the agent is to maximize the positive rewards.

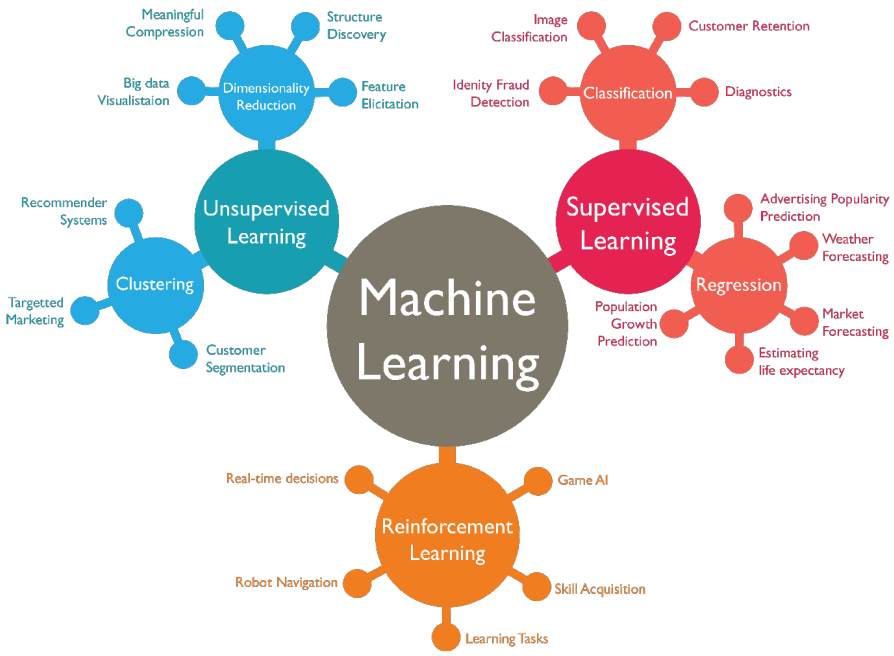


Figure 3: Types of machine learning algorithms [49].

As we have seen, different ML algorithms can be used depending on the problem nature to be dealt with. This thesis aims to create an algorithm to classify walks according to whether they present a pathology or not, so according to Figure 3 we should use supervised learning. Even though there are many classification algorithms, beforehand, it is complicated to determine which one is better than the others. It depends on the nature of the dataset and the application objective. Different algorithms are used to find the best one to solve the problem, both machine learning and deep learning algorithms. Below are explained the ML algorithms used throughout the thesis.

3.1.1. Classification and Regression Trees

Classification And Regression Trees (CARTs) are decision trees algorithms-based used in both classification and regression problems.A decision tree is a machine learning algorithm in which classification is performed by partitioning the data based on a feature

comparison. These methods have already proven to be useful in problems of disease diagnosis [50], [51], water quality prediction [52] and classification problems [53], [54].

The flowchart representation of this algorithm has a tree structure where a comparison is made between a feature and a given value at each node Figure 4. Each comparison has a result that is either false or true, which determines the path of the tree to the right or the left.

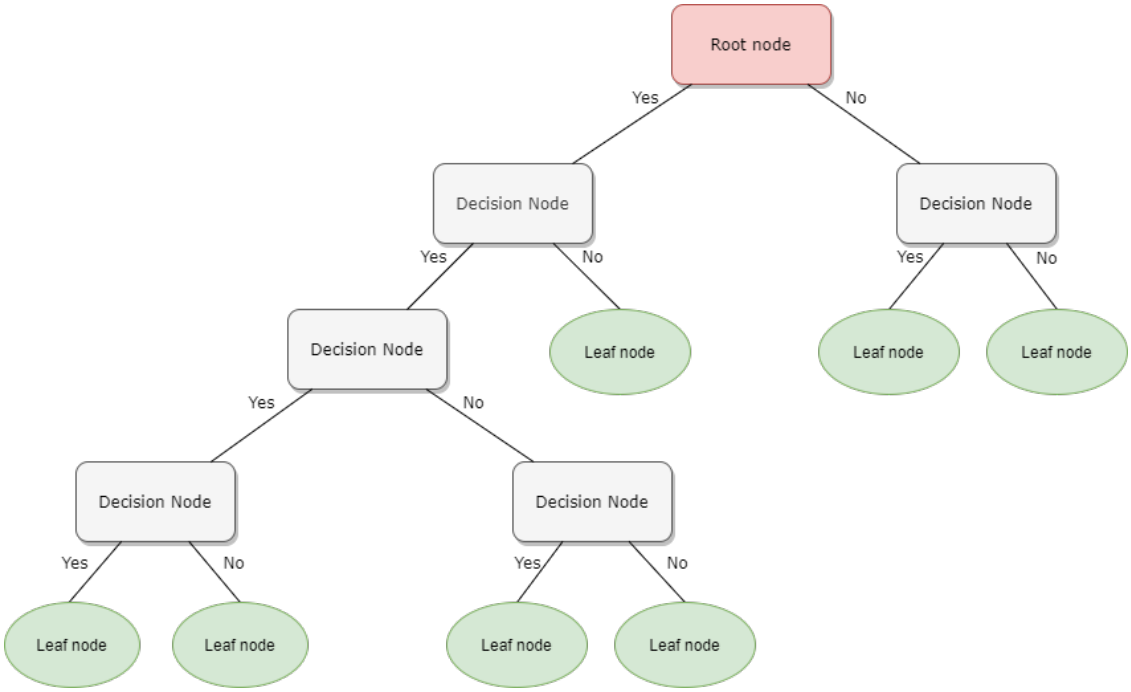


Figure 4: Decision tree structure

As mentioned above, the operation of the algorithm is based on splitting the dataset. These divisions are made by prioritising those divisions that reduce the entropy of the dataset. This process is performed iteratively at each decision node until the entropy exceeds a certain threshold or all instances have the same class, and the node becomes a leaf.

This type of algorithm has the advantage of working as a white box, i.e., the decision-making logic of the algorithm can be followed. Other advantages are the simplicity of implementation and the speed of classification once the tree has been constructed. However, these algorithms are susceptible to overtraining (mainly due to the depth of the tree), and for massive datasets, it can be challenging to follow the decision-making logic.

3.1.2. k-Nearest Neighbour Classifiers

The k-Nearest Neighbour (kNN) is a nonparametric and lazy machine learning algorithm that works under the premise that similar objects belong to a similar class, e.g., it is expected that two birds that look very much alike belong to the same species. A

nonparametric algorithm means that the algorithm does not have any parameter to learn, it just takes the data and gives an output based on the votes of the k nearest neighbours [55]. Moreover, an algorithm is considered lazy when it remembers the training dataset instead of learning a discriminative function from it [56]. In kNN algorithms, the k represents the number of nearest neighbours considered when classifying a point. The distance metric is used to calculate the closest neighbours; however, the most optimal depends on the dataset.

Imagine the case shown in Figure 5 in which we want to classify the green dot using the kNN method. As a first approach, we set $k = 3$ (solid circle), and it labels the input as a red triangle. However, if we set the $k = 5$ (dotted circle), the input is labelled as a blue square with three blue squares vs two red triangles.

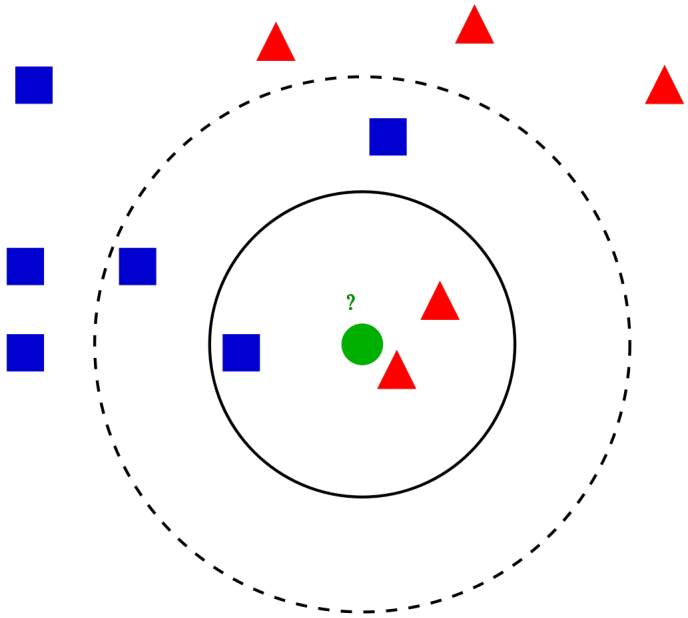


Figure 5: Example of kNN classification [57].

The original kNN algorithm [58] is based on voting in which the vote of all neighbours is worth the same. However, a new weighted algorithm based on the distance between the neighbour and the new point was proposed by [59]; the closer the dots, the higher the weight of the point.

3.1.3. Logistic Regression

Logistic regression is a supervised machine learning algorithm based on a logistic function to determine the different classes. To classify the data, the algorithm finds the line that best separates the classes. Based on where the predicted point is, the algorithm uses the next probabilistic function to give a result:

$$f(x) = \frac{1}{1 + e^{-cx}} \tag{1}$$

This function is a sigmoid that produces a value ranging over [0,1]. This algorithm maps the classes using the values between 0 and 1, being the worst case when $f(0) = \frac{1}{2}$, meaning that the classification of that class is like toss a coin.

Classification problems are usually non-binary and have several classes to choose from. A proposed solution was giving numbers to the different classes, starting on the one and so on until all the classes have a number. Then, perform a linear regression to predict the output. Although this solution would work beforehand, it is a bad idea, as the numbers have an ordinal scale that usually classes do not have.

The correct way to perform multiclass classification with logistic regression is to use as many one-vs.-rest classifiers as there are classes. For each class, a logistic classifier is trained to distinguish one class from the others. Once all the classifiers are trained, to classify a new element, it is classified by all the classifiers, returning the class with the highest probability.

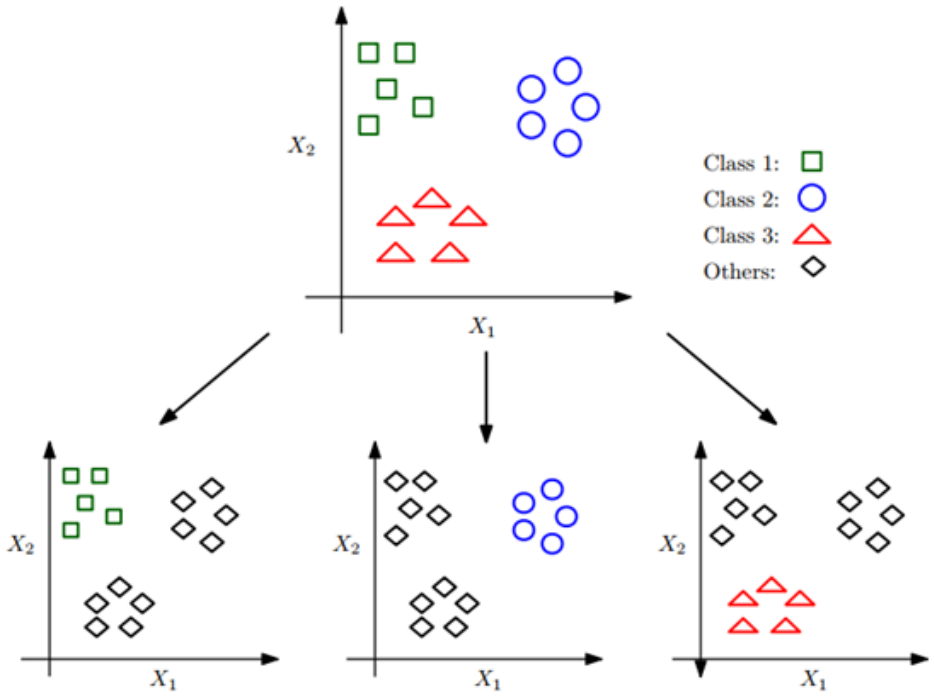


Figure 6: Example of breakdown into 3 one-vs-rest classifiers [60].

3.1.4. Naïve Bayes

Naïve Bayes is a supervised machine learning algorithm used in classification problems. However, naïve Bayes assumes that there is no correlation between the features, that is, they are independent of each other. This assumption is difficult to find in real datasets, however, this is why the algorithm is called naive. This algorithm is based on Bayer's

theorem, which formula is as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

The above formula describes the probability of A occurring, given that B has already occurred. This algorithm is mainly used in sentiment analysis and recommendations systems. It has the advantage of being a fast algorithm as well as being easy to implement; however, the main disadvantage is the assumption of the data being independent.

3.1.5. Support Vector Machines

Support Vector Machines (SVM) is a supervised machine learning algorithm with the objective to find the hyperplane that classifies the data in the different classes. Before continuing, it is essential to define what a hyperplane is. In a p-dimensional space (where p is the number of input variables), a hyperplane is defined as a plane subspace with p-1 dimensions. The mathematical definition of the hyperplane is as follows:

$$w^T x + b = 0 \tag{3}$$

Where:
w is a weight vector.
x is the input vector.
b is the bias.

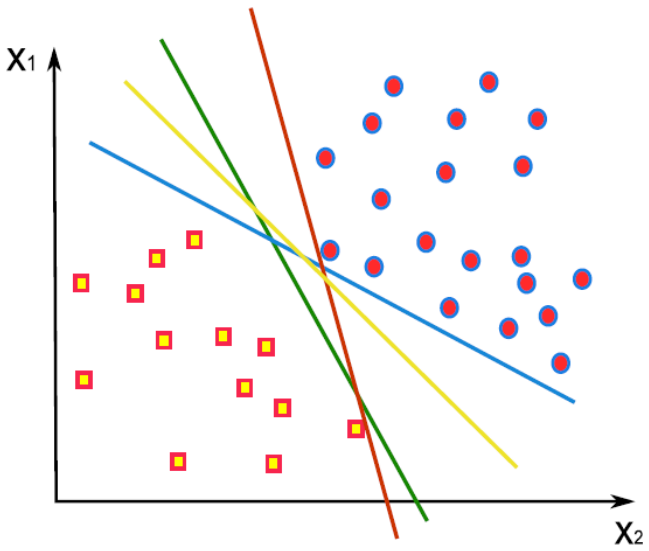


Figure 7: Possible separation hyperplanes [61]

If we are working in a two-dimensional space, the hyperplane is just a line, and on the other side, if the number of input features is three, a hyperplane is a two-dimensional

plane. However, to separate the data, there exist many possible hyperplanes that achieve the requirement, Figure 7 shows an example of several hyperplanes that correctly divide the data. So there exist an unlimited number of possible hyperplanes, but which is the optimal one?

The objective of the SVM algorithm is to find a hyperplane that maximises the margins (or support vectors) and the number of points correctly classified. The margins are the distance between the hyperplane and the closest point to the hyperplane. To find the optimal hyperplane, the algorithm uses the support vectors, which are the data that are closer to the hyperplane and influence its position and orientation [60]. Utilising the support vectors, the two support hyperplanes H1 and H2, are calculated, and the hyperplane H is the one that is equidistant to the two. Figure 8 shows a representation of the optimal hyperplane and the hyperplanes calculated through the support vectors.

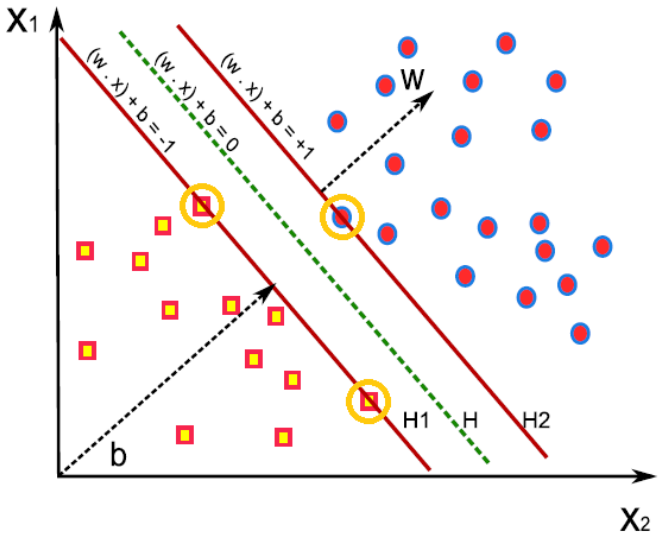


Figure 8: Optimal hyperplane. The yellow circle shows the support vector points used to calculate H1 and H2 [61]

The images above show a linearly separable dataset, nevertheless, the data are usually not presented that way. Tipically, the data cannot be separated just by drawing a straight line, an example of no linearly separable data is shown in Figure 9a. As the dataset is not linearly separable, the algorithm adds a new z dimension by transforming the data (Figure 9b). Within this new dimension, the algorithm calculates a new hyperplane (Figure 9c). The linear hyperplane estimated using the third dimension becomes nonlinear if we represent it in the original space (Figure 9d). This procedure is known as the kernel trick.

The SVM algorithm is effective with high dimensions datasets. One advantage of the SVM algorithm is the robustness to data changes, i.e., if the support vectors are still unchanged, the SVM algorithm would calculate the same hyperplane. On the other hand, SVM presents a high computational cost for large datasets. SVM algorithms are widely used in energy forecasting [63], electroencephalogram (EEG) classification [64]

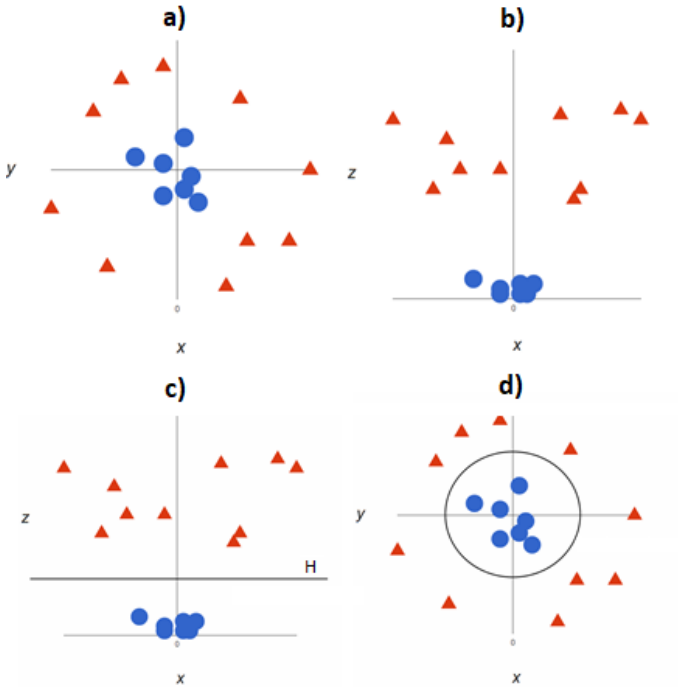


Figure 9: No linearly separable dataset [62]

and diagnosis applications [65] among others.

3.2. Deep Learning

Deep learning features algorithms analyse data with a logical structure similar to that of a human does. For this purpose, DL algorithms are based on structures called Artificial Neural Networks (ANN). These algorithms mimic the mechanism of learning of biological organisms by using artificial neurons. The natural neurons are connected to one another by synapses; however, the artificial ones are connected through weights, which serve the same purpose. Figure 10 shows an example of an artificial neuron.

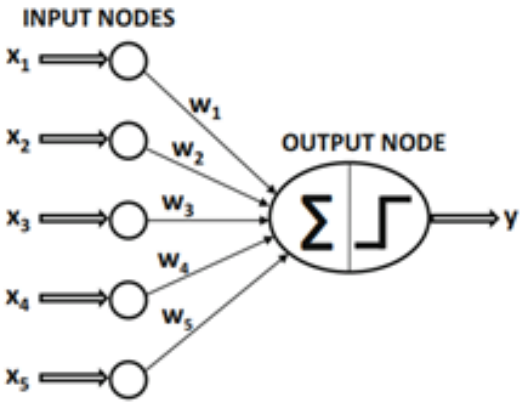


Figure 10: Artificial neuron [66]

Although the minimum unit used by ANNs is the neuron, these networks are formed

using layers, which are nothing more than clusters of neurons. The more neurons, the larger the layer. When creating a neural network, we need different layers, which can be grouped into:

- **Input layer**: this layer transmits the data to the network without performing any calculations on it. This layer must have as many neurons as features are used.
- **Hidden layers**: these are the intermediate layers between the input and output layers. They are called hidden layers because the calculations they perform are not visible to the user.
- **Output layer**: is the layer that produces the output of the neural network, i.e., that makes the prediction.

As more layers the network has, the deeper it is. The deeper the network becomes, the more information can be inferred from the data so that a shallow network may detect lines, and as the depth increases, it may be able to distinguish shapes or silhouettes. An ANN estimates the output function by propagating input neuron values to output neurons and using weights as intermediate parameters [66]. Learning is achieved by altering the weights connecting neurons in a process known as training.

Multilayer neural networks are usually trained using a back propagation algorithm, which consists of two main phases, forward and backward phases.

1. **Forward phase**: The first phase consists of feeding the neural network with the training data, which triggers a cascade of calculations through the layers using the current weights. The predicted value is compared with the training instance, and the derivative of the loss concerning the output is computed. This derivative is then computed in the backward phase.
2. **Backward phase**: The objective of this step is to learn the gradient of the loss function concerning the different weights of the network. These gradients are used to update the weights. This process is known as the backwards phase since the gradients are learned from back to front (starting from the output layer).

Once the neural network has been adequately trained, it can be used to predict or classify data. Depending on the layers used in a network, we can find three different types of networks: simples, recurrent and convolutional. The simplest networks are those based exclusively in fully connected layers, that are layers in which all the outputs are connected to the inputs of the next layer. The remaining networks are more complex, so a brief explanation of these is given below.

3.2.1. Recurrent Neural Networks

Neural Networks (NN) learn about the input dataset at each iteration; however, what happens when the input data is a temporal sequence, weather, or stock forecasting? Classical neural networks have no memory about the information that has been processed, this makes it impossible to predict future values without knowing the previous values.

To overcome this, Recurrent Neural Networks (RNN) maintain information from past interactions, so while classical or forward networks make decisions based on current input only, recurrent networks make decisions based on current and previous inputs. A comparison of both networks is shown in Figure 11.

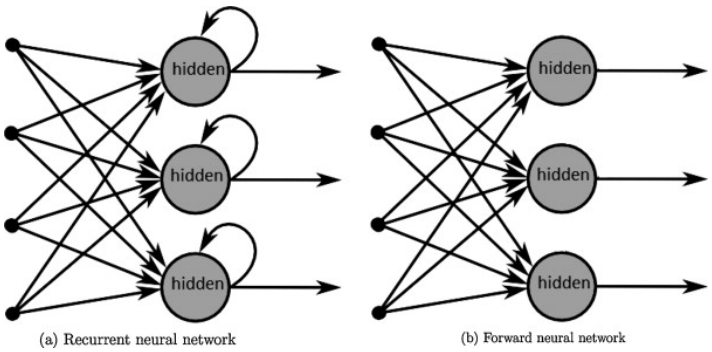


Figure 11: Recurrent vs forward neural networks [67]

In traditional NNs, we speak of neurons, however, for RNNs, the term changes to cells, which are a combination of neurons and activation gates. The three best-known cell types are: simple RNN, Long Short-Term Memory (LSTM) cells, and Gathered Recurrent Units (GRU). The image Figure 12 shows an example of all of them.

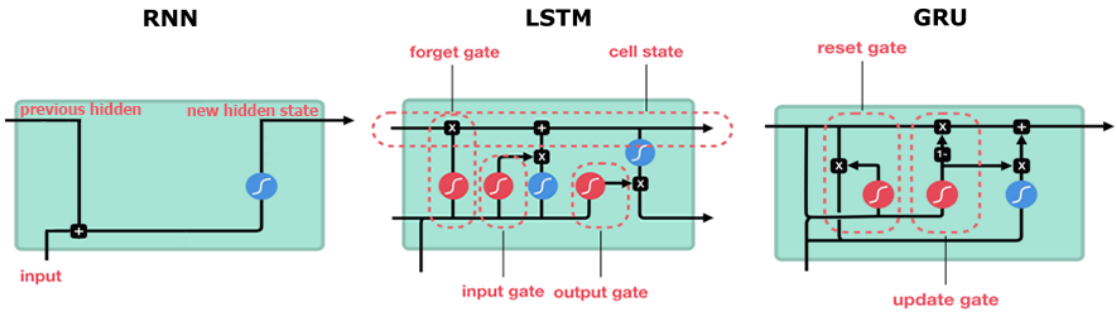


Figure 12: Types of RNN cells [68]

The RNN first combines the current input and the hidden state (which acts as the memory from the previous cell) forming a vector, which contains not only the current information but also the previous inputs information. The new vector goes through the tanh activation, regulating the values between -1 and 1. This vector is the hidden state for the next RNN cell. Though the basic RNN cells are very simple, they work pretty well, however, due to the decay of the loss function gradient it is hard to solve long-term temporal dependencies [69]. Despite this, these cells perform well in short sequences.

The LSTM cells have the same workflow as RNN cells: the data is processed forward the cells, however, LSTM shows more operations which allow the LSTM to forget or hold information. Compared to RNN, LSTM cells have an additional input (C_{t-1}) and output (C_t). This is call cell state and acts like an information highway, which transfers the information through the whole network. In this way, the network can maintain the information for long periods, reducing the short-term memory problem caused by gradient decay. In the LSTM, not all the previous information is feed to the next cell as the forget gate controls the amount of information

The GRU cell is a newer, simpler version of the LSTM, which uses the hidden state instead of the cell state to transfer information. Like the LSTM, the upgrade gate decides which information to keep and which to forget. The reset gate controls the amount of past information that is forgotten. This type of cell works similar to LSTM, however as the operations performed by the cells are pretty lower, they are faster than the LSTM.

3.2.2. Convolutional Neural Networks

As mentioned before, when creating a neural network, in the input layer, you need as many neurons as features you use, but what happens when you want to use an image? In this case, a layer of as many neurons as pixels in the image should be created. With the quality of the current cameras, both photo and video, this fact is unfeasible (a typical image can have around twelve million pixels). If we were to create a network of such a size, the time needed to train it would be enormous. To solve this problem, Convolutional Neural Networks (CNN) are used.

The CNNs are neural networks that are applied to image classification. They work by extracting the representative characteristics of the image and classifying them. In the Figure 13 can be seen an example of a CNN, the part before the flattened is the one in charge of extracting the characteristics, and the part beyond it is the one in the task of classifying them.

In the classification part, we find convolutional and max-pooling layers. The purpose of the convolutional layers is to extract high-level features (such as edges or shapes). The max-pooling layers oversee extracting the dominant features and thus reduce the computational cost. Once the most relevant features have been extracted, they are flattened in order to classify them using a network of fully connected layers.

3.3. Optimization Algorithms

When working with both machine and deep learning algorithms, we have two types of parameters:

- **Hyperparameters**: these are the parameters that the user can set before training.

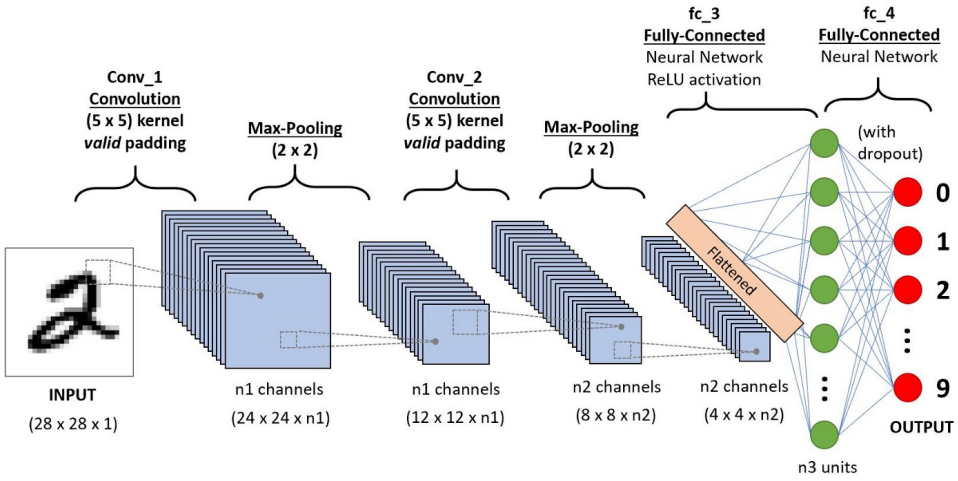


Figure 13: CNN structure [70]

- **Model parameters**: these are the intrinsic parameters of the model that are learned by the model during training and cannot be modified by the user.

The hyperparameters affect the initial structure of the model, so they influence the learning process of the data. The values of these hyperparameters differ from problem to problem, and finding the best value for them is an optimisation problem. To find an optimal solution for the hyperparameters, we can use both Random Search (RS) and Grid Search (GS). Both methods work similarly; you select values for the hyperparameters you evaluate and train and test the model. The training and testing process is usually performed several times, randomly choosing the training and testing datasets (k-fold) and giving the final result as the average of all the runs. Once the model accuracy with these parameters is known, it is recorded in a table, and the hyperparameters are changed. In the end, the combination of hyperparameters that gives the best accuracy is chosen.

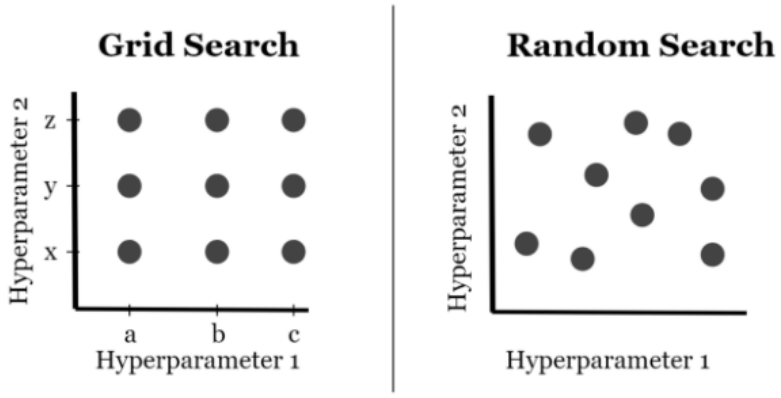


Figure 14: Random Search vs Gird Search [71]

The difference between RS and GS is the choice of hyperparameter values. In the case of RS, a range of values is set, and the algorithm randomly chooses a value within the range. On the other hand, GS requires the hyperparameter values to be detailed and

it chooses random combinations of these values. The Figure 14 shows an example of RS against RS.

Both algorithms work on a limited set of values, however, as in each iteration, RS uses different values, more hyperparameters combinations are tested, and it is possible to find a combination that we would not have chosen beforehand. On the other hand, when trying with random values, it can happen that the combinations tested do not offer good performance. Both algorithms solve the problem of optimising the models, and with sufficient iterations, both find the optimal solution.

4. SMARTPHONE-BASED ALGORITHMS

This chapter aims to study the viability of using machine learning-based algorithms for gait pathologies detection using a smartphone to acquire the database. This idea stems from the work [72] showing that it is possible to identify people using a smartphone, which leads to the question of whether it is also possible to locate pathologies. To answer this question, three different smartphone configurations were used. The use of the 1-smartphone configuration has the advantage of portability and usability, as actually everyone owns a smartphone. In contrast, the 4-smartphone configuration acquires more information about the stride, and the structure is closer to the used by the professional motion capture systems.

Using smartphones for data acquisition, in addition to the small cost, offers enormous portability since, as seen in state of the art, most systems require the user to move to the laboratory. The use of smartphones opens up the possibility of performing a gait analysis anywhere and by anyone since everyone owns a smartphone nowadays.

4.1. Pathology Detection Using a Single Smartphone

Due to its easiness and its simplicity, the configuration using a single smartphone is evaluated first.

4.1.1. Algorithm

In this section, the algorithm used to detect the lower limb pathologies is explained. The structure of the algorithm is as follows: the first step is the pre-processing stage, in which the signals are prepared; cycle extraction phase, in which the signals are trimmed into gait cycles; once the GC are extracted, they are parametrized in the feature extraction phase; lastly, the parameters are used to feed the machine learning algorithm. A schema of the algorithm can be seen in Figure 15. To find the ML algorithm that fits the best to this problem, five ML algorithms are used.

4.1.1.1 Pre-processing

When using a phone to acquire the data, it can be placed in different orientations each time used. If we do not take into account the phone orientation, the magnitude of the axes will change between different positions, making it impossible to compare different walks. A possible solution is to set the smartphone position for all walks, however, this would add difficulty to the acquisition of the database, so another solution is needed. An

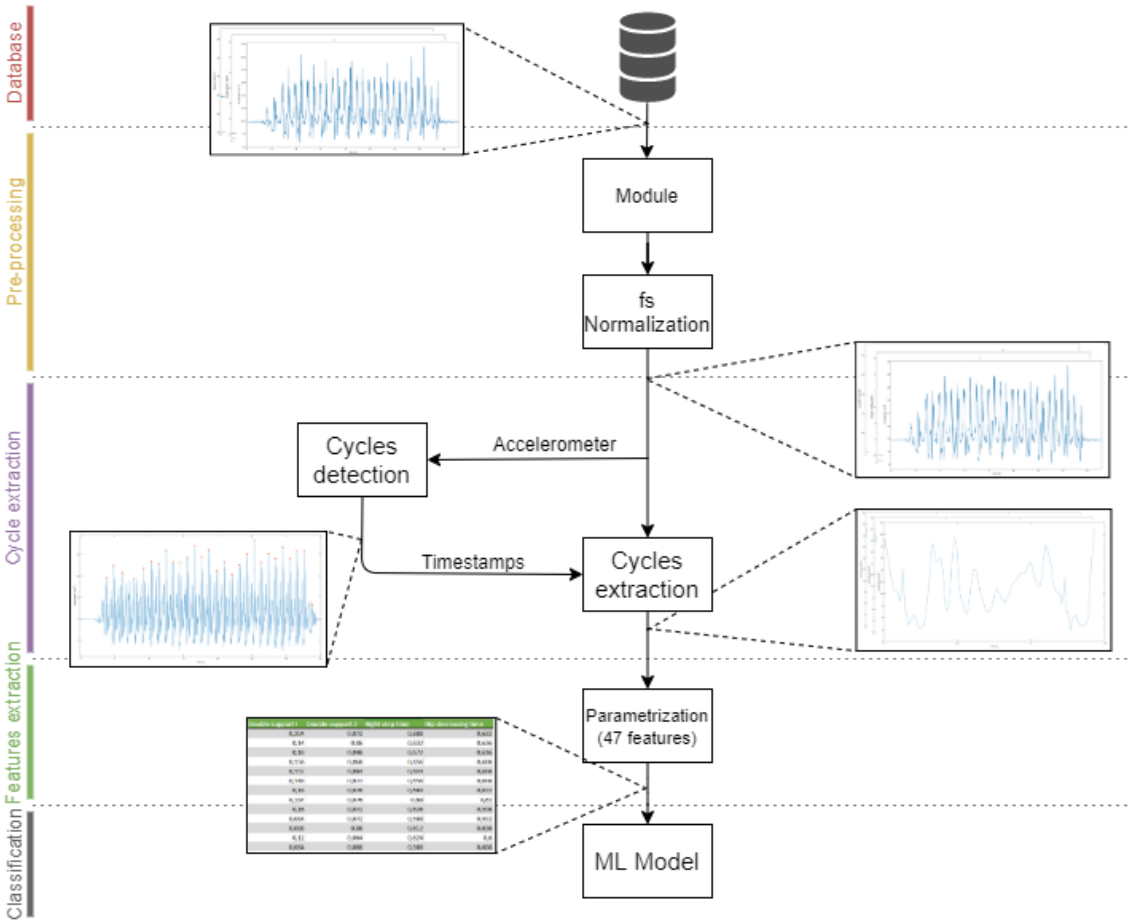


Figure 15: One-smartphone configuration algorithm schema

alternative solution is to perform the module vector (Equation 4) of the axis as the module value is not affected by the phone orientation, allowing to place it in any orientation.

$$module = \sqrt{x^2 + y^2 + z^2} \tag{4}$$

As told before, the smartphone works with regular OS, which cannot guarantee a constant sampling frequency (f_s). To solve this, we have to interpolate the samples and make the f_s constant, the process is performed as follows:

- By checking the time vector, we can find out when a sample is ahead or behind, as by using the f_s we know that the samples have to be at 0.01 s distance.
- If a sample is out of place, we calculate the value at the correct position by interpolating the value of the adjacent samples.

4.1.1.2 Cycle Extraction

After solving the inherited problems of data acquisition, we can now perform the cycle extraction. By extracting the GCs, the amount of data is increased as from each walk,

we can obtain around 25-30 cycles. Since the signals from the different sensors are synchronised, obtaining the starting point of the cycles is performed once per walk. As the accelerometer signals show the clearest cycles are used to finde the starting points. The process of finding the starting points of the cycles is described below.

1. First of all, the algorithm looks for all the maximum peaks without any restriction.
2. Then the average of these peaks is obtained, and the peaks below it are discarded.
3. Finally, the points that are less than 0.9 seconds away are eliminated.

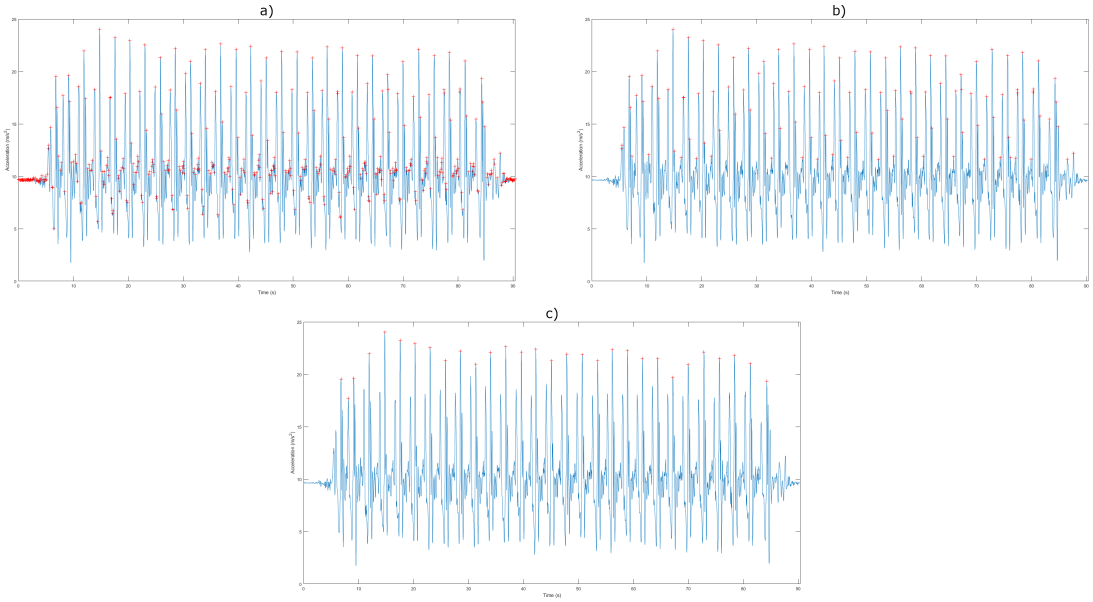


Figure 16: Cycles extraction process. a) All the peaks without restriction; b) the peaks above the average of the peaks; c) GCs starting points.

It can be seen that throughout the process of finding the starting points (Figure 3), the different rules remove the incorrect points until the correct ones are seen (Figure 3c). Once the points are obtained, the next step is to divide the signals into gait cycles. Once the GCs have been extracted, the following step is to parametrize them, thus increasing the amount of information we have for each cycle.

4.1.1.3 Feature Extraction

The features used can be divided into time features and frequency features. These features should characterise the GC so that information about it is not lost. Below are presented the description of the features used.

Average value This feature measures the mean value of the signals, which corresponds to the central value about which they oscillate.

Maximum and minimum values These features measure the highest and the lowest values of the signals.

Rise time This feature measures the time it takes for the signals to rise from 10 % to 90 %.

High/width ratio This feature measures the high/width ratio of the accelerometer central peak. It can be seen in Figure 17.

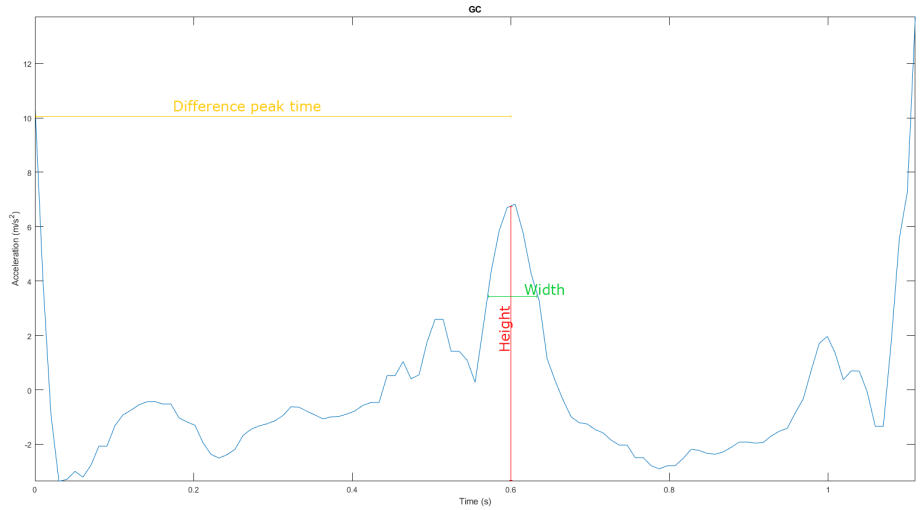


Figure 17: Example of features.

Difference peaks time Time difference between the start of the signal to the central peak of the accelerometer signal. An example can be seen in Figure 17.

Zero-crossing rate The zero-crossing rate is the ratio at which the signal changes from a positive to a negative value, i.e., each time it passes through zero.

Harmonics position and value The position in frequency and the power in dB of the first five harmonics of the signals.

Along the signal parametrization process, 14 parameters have been used and obtaining 47 different features from each GC, an overall view of the features is shown in Figure 18. Once the features are obtained, they should be prepared in order to feed the machine learning algorithms.

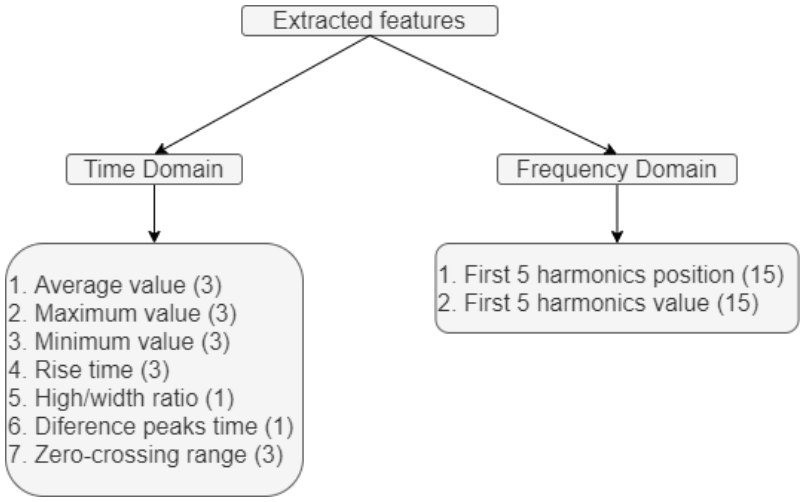


Figure 18: Features obtained in the one-smartphone configuration. In brackets the number of values obtained from each feature.

4.1.1.4 Data Preparation

When using a predictive model, for example, a machine learning algorithm, raw data is not recommended to use directly. There exist some reasons such as statistical noise and errors in the data (incomplete data instances or outliers) or requirements of the algorithms (some ML algorithms impose data constraints. Data preparation can be defined as transforming the raw data into a form more suitable for ML algorithms. This process is specific to the database, the used algorithms, and the goals of the project, the methods applied throughout the data preparation process are detailed below.

- **Data cleansing**: Consist of identifying and correcting or deleting errors, missing values, or inconsistent data in the database.
- **Data formatting**: Consist of preparing and sorting the data to feed the ML algorithm.

4.1.1.4.1 Data Cleansing

When ML algorithms are used, data plays an important role. It is through the data that the algorithm learns, so the presence of incorrect data in the dataset can tend to an inappropriate learning process. To avoid these problems, before feeding the ML algorithm, the data must be cleansed. Data cleansing is the process of preparing the dataset by removing incoherent, missing, or duplicated data. This process increases the quality of the dataset, resulting in an increase in algorithm accuracy [73]. At the case in point, the data have been treated as follows:

1. In the first place, the missing and zero values are discarded.

2. After that, the incoherent data is removed, as negative times or ranges, or unusually large amplitudes.

3. The last step is to analyse the data to determine whether data contains outliers and, if so, eliminate them.

Steps 1 and 2 are carried out by directly analysing the data and removing the detected values, however, the outlier detection needs a statistical method to be performed. As we are working with a dataset that follows a normal distribution, we can use a criterion based on z-score [74] that is the most common method. The z-score is a standardization method to transform normal variables into standard score variables. The new variables will have a variance of 1 and a mean of 0. The z-score is defined by the Equation 5:

$$Z = \frac{z - \mu}{\sigma} \tag{5}$$

Where:
μ is the mean.
σ is the standard deviation.

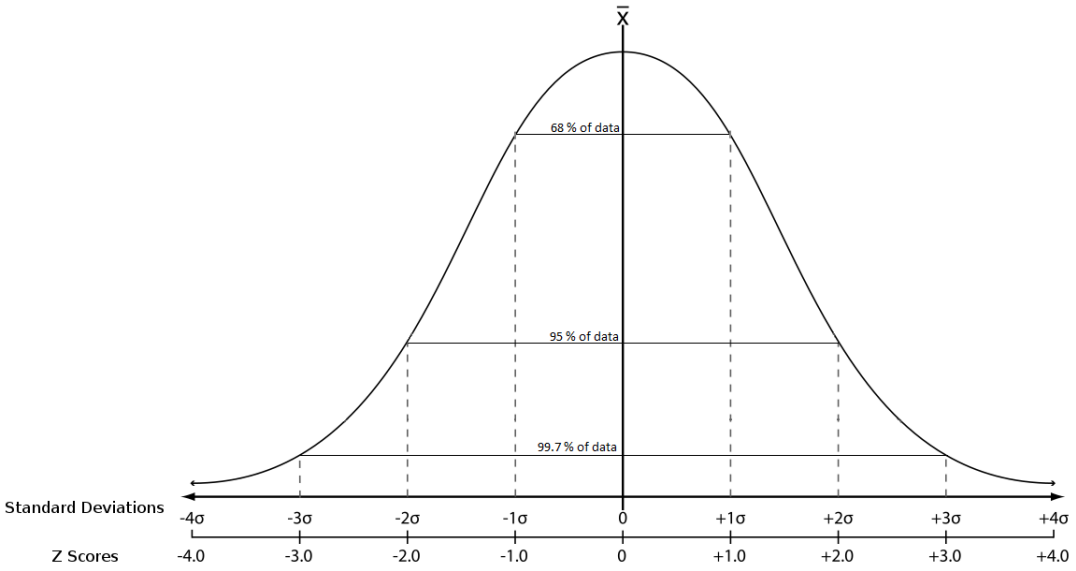


Figure 19: Relationship between z-score, standard deviation, and the percentage of data.

As the new data have a mean of 0, the z-score can be used to measure how far a value is from the mean, measured in standard deviation units (Figure 19). This distance can be positive or negative, meaning that the value is above or below the mean. Based on the literature, a value of 3σ is used in the outlier elimination.

4.1.1.4.2 Data Formatting

After removing the noisy samples of the dataset, we must short the data to feed the ML algorithms. The data is arranged in a matrix, where the columns represent the features,

and the rows represent the GCs. Following is presented an example of a matrix:

$$M = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_n^2 \\ x_1^3 & x_2^3 & x_3^3 & \cdots & x_n^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^m & x_2^m & x_3^m & \cdots & x_n^m \end{bmatrix} \tag{6}$$

Where:
n is the number of features.
m is the number of GCs.

4.1.1.5 Feature Classification

Once the data is prepared, we can use the matrices to feed the ML algorithms. As told above, in order to find an optimal algorithm, five different ML algorithms are used: CART, kNN, Linear Regression, Naïve Bayes and SVM.

Different ML algorithms employ different hyperparameters to control the learning process. Finding the correct values for them can be an arduous and time-consuming task. Instead of trying different values to find those that give the best results, the values of configurations that have already proven their worth are used.

The CART configuration is based on the works [51], [75], which focuses on the detection of heart problems. For kNN, we rely on works on fall detection [76] and lung pathology detection [77]. On the other hand, the works consulted for Logistic Regression focus on the detection of voice and [78] brain pathologies [79]. For Naive Bayes, we rely on papers on the detection of brain [80] and spine pathologies [81]. Finally, for SVM, we rely on voice pathology detection studies [82], [83] .

Once we have decided on the hyperparameter configuration for all the ML algorithms, we can use the data to feed them. The database is divided into 30 % for testing and the remaining 70 % for training to feed the algorithms. Each algorithm is run five times, and the data is randomly divided each time. In this way, we avoid the algorithm getting stuck on a particular set of data.

4.1.2. Evaluation Database

In every machine learning project, one of the essential requirements is the database. It needs to be diverse, i.e., across category differences should be significant while within categories should be insignificant [84] and scalable, i.e., new data must be readily incorporated into the database. With this in mind, we created a database with healthy and pathological users. Since getting access to people with a pathology can be a difficult task, and as this is the first approach to the problem, the pathology walks in the database

have been simulated. The smartphone used, the database procedure and the final dataset are explained below.

4.1.2.1 Capture Device

As mentioned above, the database is acquired using one smartphone, specifically the BQ Aquaris. By using the smartphone, the acceleration (m/s^2), angular acceleration (rad/s) and the magnetic field (μT) of the walk can be acquired, all of them in 3D. Nevertheless, a problem exists when acquiring the data whit the smartphone; it works with standard Operating System (OS). Unlike Real-Time Operating Systems (RTOS), which provide a hard real-time response without buffer delays, OS provide a soft real-time response, which does not guarantee the absence of delays.

Due to system delays, the sampling frequency of the system oscillates within the range 90 ± 10 Hz instead of remaining constant at 100 Hz. As the acquired signals have inconstant frequency, they need to be resampled to normalize the sampling frequency to 100 Hz.

4.1.2.2 Acquisition Procedure

When acquiring a database, some non-planned problems can interfere. In order to minimize those problems, a suitable acquisition procedure is needed. A detailed procedure facilitates the acquisition process as well as making the acquisition replicable at any time.

As mentioned above, a first approach to the detection of pathologies using a smartphone is presented. For this reason, instead of looking for a complex pathology to simulate, it is decided to simulate a problem that at first sight seems more straightforward. For this reason, it was decided to create an imbalance in the way of walking by using a standard shoe on one foot and a boot with a heel on the other.

A reference algorithm can be created and gradually improved by starting with something simple to detect and increasing the number of pathologies or removing the restriction for certain variables. However, as it is the first attempt to create the algorithm, we want to avoid the influence of non-considered variables, so the following rules are established:

- To perform the walks, the users wear the same shoes and the same trousers. The shoes used are flat and flexible so as not to cause discomfort. On the other hand, the trousers are short and wide, to avoid discomfort when walking. Users could choose between several sizes of shoes and trousers to find the one that best suits them.
- The path is a flat surface 40 metres long with markings at the beginning and end. This route is always the same for all the walks, in this case, it takes place in a university corridor.

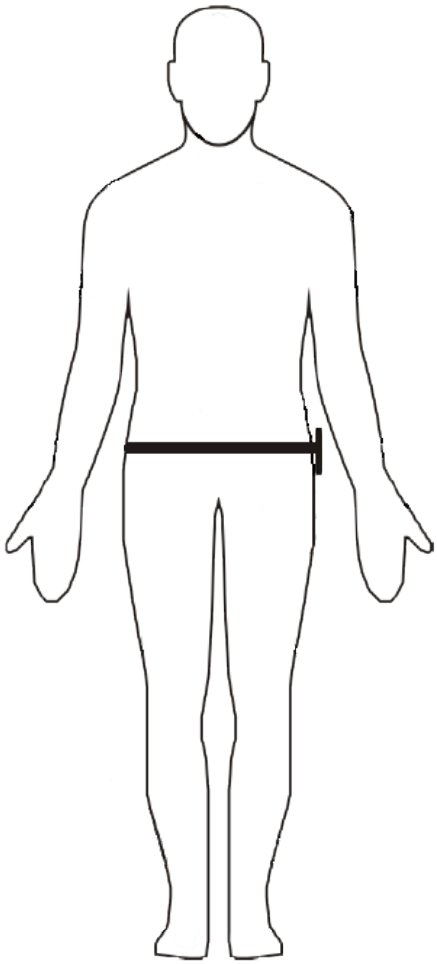


Figure 20: Smartphone position in the one-smartphone configuration.

- As the users wear wide trousers, the phone cannot be placed in the pocket, as it would be susceptible to shaking. Because of this, the phone is placed at the waist, using a belt and a holder. Figure 20 shows a schematic of the positioning of the phone.
- Each user performs 12 walks: 4 healthy, 4 left-pathology and 4 right-pathology.

Following this procedure, nine people were recruited for the experiment. Although this is a small DB for an ML algorithm, as it is a first approximation, it is considered enough to evaluate the performance of the configuration. Table 2 presents an overall of the dataset (to obtain the number of cycles, we consider an average of 28 cycles per walk).

4.1.3. Results

To evaluate the performance of the different algorithms, we use the confusion matrices, through which we can obtain metrics such as the false positive or the false-negative rate. In our case, we are looking to classify healthy GCs as “healthy” and pathology GCs as

Table 2: One-smartphone dataset details.

Users	Number of walks		Number of cycles
	Pathological	Healthy	
9	72	36	3024

"Left" or "Right", depending on the side of the limp. Since this problem can be considered a medical problem, the metrics differ slightly from those traditionally used. Below is a brief review of the evaluation used metrics for machine learning multiclass classification models.

- **False Positive Rate (FPR)**: is the proportion of healthy walks wrongly identified as pathological. Although, in this case, pathology has been diagnosed when the patient was healthy, this does not have a serious effect on the patient, beyond the fact that the patient is treated and does not need it. Therefore, we can consider this metric as a low-risk error, it can be considered as such as this algorithm is intended to aid in diagnosing gait problems, however, in a case where treatment is detrimental to the patient (e.g., cancer screening), this metric should be considered of vital importance.

- **False Negative Rate (FNR)**: is the proportion of pathological walks (both left and right) wrongly classified as healthy. As the algorithm presented here is intended to aid in diagnosing pathologies, we need this metric to be as low as possible, otherwise, a person with a pathology might not be diagnosed. Therefore, the patient would not receive any treatment for the pathology.

- **False Limp Rate (FLR)**: is the proportion of pathological walks that are misclassified as other pathology (e.g., left limp classified as right and vice versa). This metric has the worst of both FPR and FNR, as it can result in a patient receiving treatment that is at best harmless to the patient. It can also be that the pathology is left untreated, for example, when a knee pathology is classified when it is actually in the ankle.

Analysing the confusion matrices in Figure 21, it can be seen that the algorithms show a different response to each other. In the case of CART, kNN and Linear Regression algorithms, they predicted almost all instances as healthy, SVM does it as left pathology. Lastly, Naïve Bayes classifies all cases equally. To better analyse these behaviours, the metrics presented above are used, which are calculated using the confusion matrices, these metrics are shown in the Table 3.

Analysing the results, we can see that all the algorithms have poor outcomes, being kNN the best one with an accuracy of 59 %, however, it has an FNR of almost 20 %. In

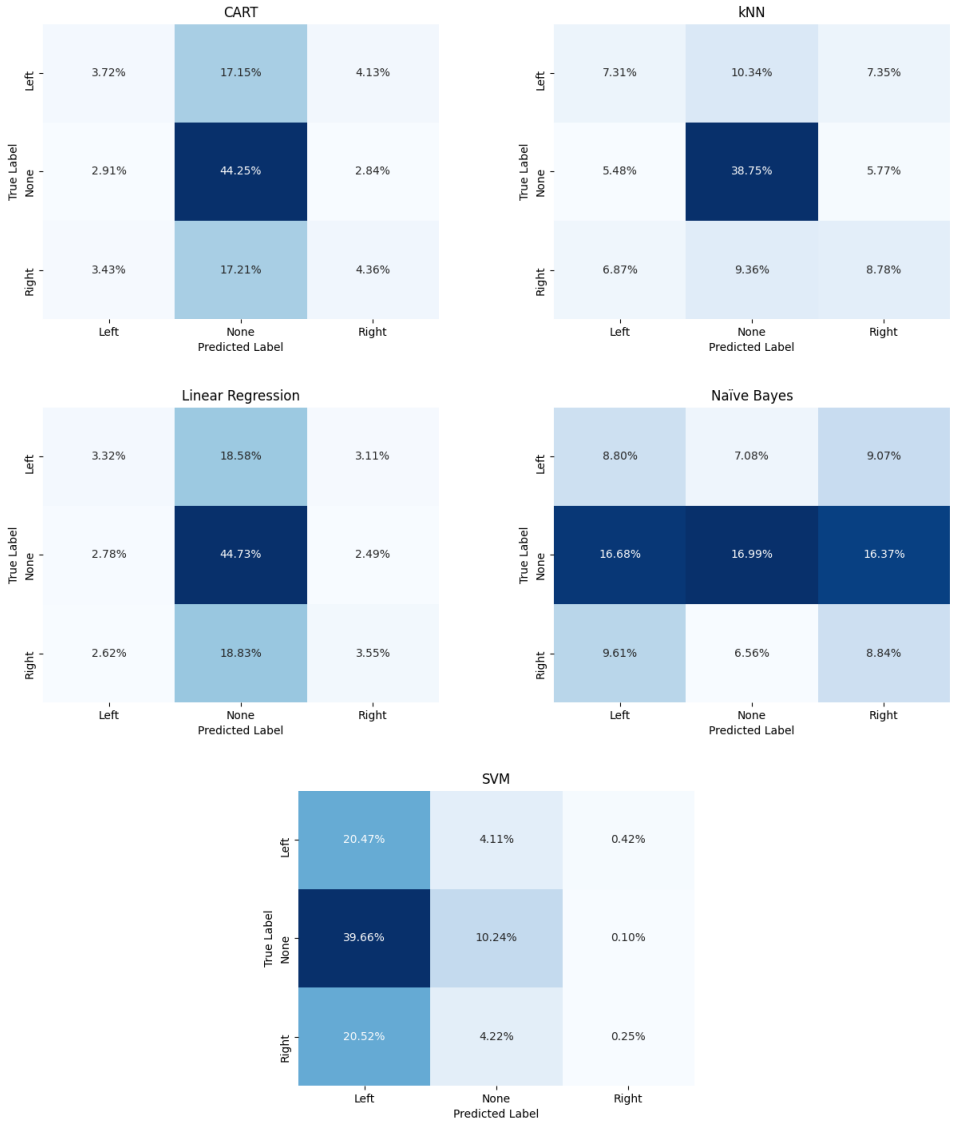


Figure 21: Confusion matrices for one-smartphone configuration (left: left pathology, none: healthy, right: right pathology).

the light of the results, we can conclude that it is not viable to use the data acquired by one smartphone to perform pathology detection accurately.

4.1.4. Conclusions on using one smartphone for pathology detection

As we have seen, the different algorithms used in this chapter do not give good results, so that we can discard the configuration presented here as valid for pathology detection. However, although some algorithms have shown an accuracy of 30 %, others are around 60 %. Though beforehand, these results do not seem very useful, they open the door to future research. Is it the lack of information that obstructs the results? Are the cycles sufficiently represented with the features used? To answer these questions, the following experiment will follow the same procedure explained above, but using two smartphones

Table 3: ML algorithms metrics for 1-smartphone configuration.

ML Algorithm	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)
CART	37,36	2,35	3,16	57,13
kNN	19,74	12,42	8,79	59,05
Logistic Regression	37,4	5,27	5,73	51,6
Naïve Bayes	13,64	33,05	18,7	34,61
SVM	8,33	39,76	20,95	30,96

instead of one in acquiring the database.

4.2. Pathology Detection With The Joint Use Of Two Smartphones

This section aims to continue the experiment presented above but using two smartphones for data acquisition. This problem is inherited from the previous section, as it seems that a single phone does not provide enough information for pathology detection. The algorithm used here is firmly based on the above explained, however it has some new processes that aim to improve the algorithm, however, the ML algorithms are the same.

4.2.1. Algorithm

As it can be seen in Figure 22, the schema of the new algorithm is quite similar to the one presented previously, as the objective of both is the same. The main difference between them is the time alignment and the new low-pass filtering stages in the current algorithm. The other distinction is the number of features used, as we now have information from both legs, more features are calculated. The remaining stages are still unchanged.

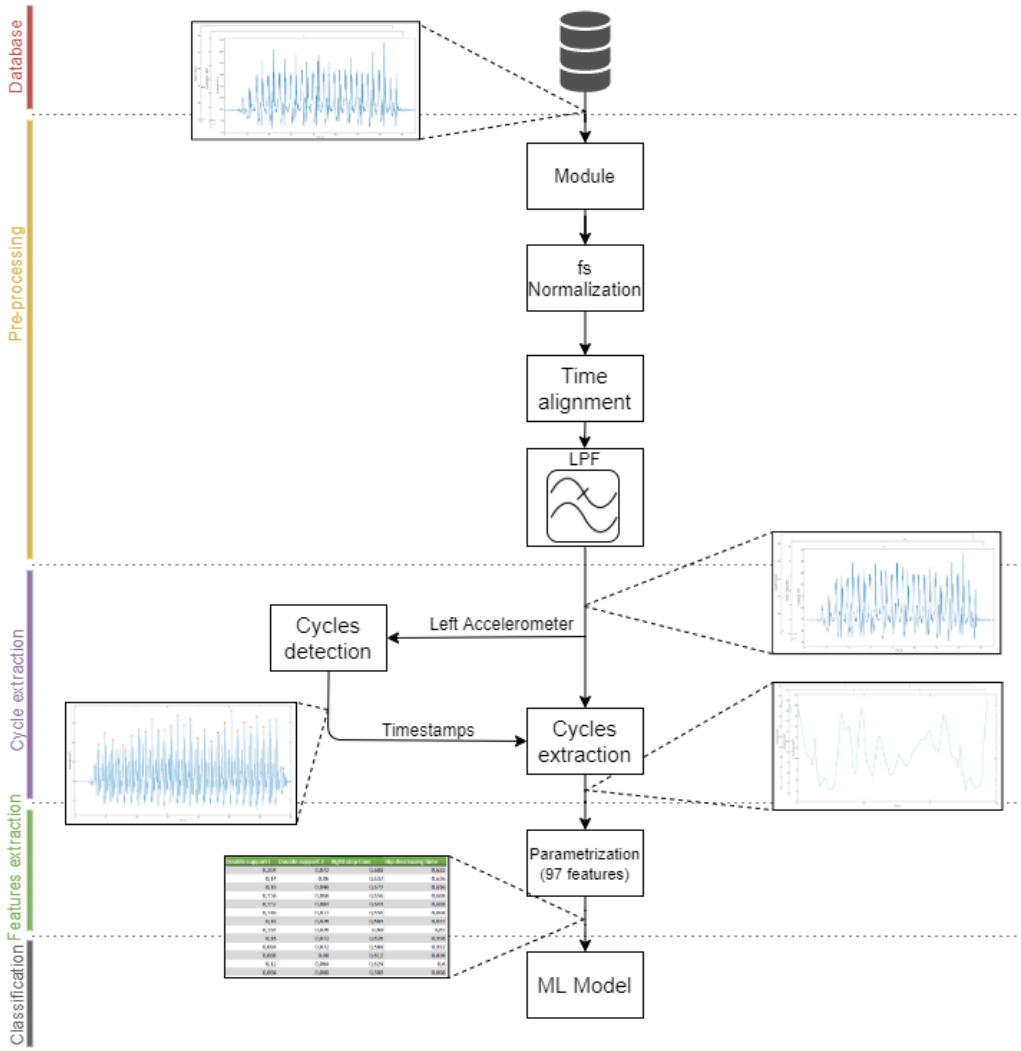


Figure 22: Two-smartphones configuration algorithm schema.

4.2.1.1 Pre-processing

Below are explained the new processes added to the algorithm: the time alignment and the signal filtering stages. The time alignment process becomes necessary since we work with two different devices (whose purpose is not the data acquisition). Even if the data acquisition is performed simultaneously, there may be a small-time delay between the signals. To find the time delay between the signals, we use cross-correlation. When calculating the cross-correlation of the two signals, the maximum value indicates the point where the two signals are aligned. Thus, using this measurement, we can obtain the delay of the signals and synchronise both smartphones signals.

From the frequency feature used in the above algorithm, it has been observed that the frequency of the cycles is about 1 Hz so that the signals can be filtered to remove unnecessary information. All the signals are filtered using a Butterworth low-pass filter to remove the frequencies higher than 15 Hz. This frequency is empirically set, allowing a high margin to ensure not to extract essential information. As it can be seen in Figure 23,

by low pass filtering the signals, the shape is not affected however, the high-frequency noise is removed.

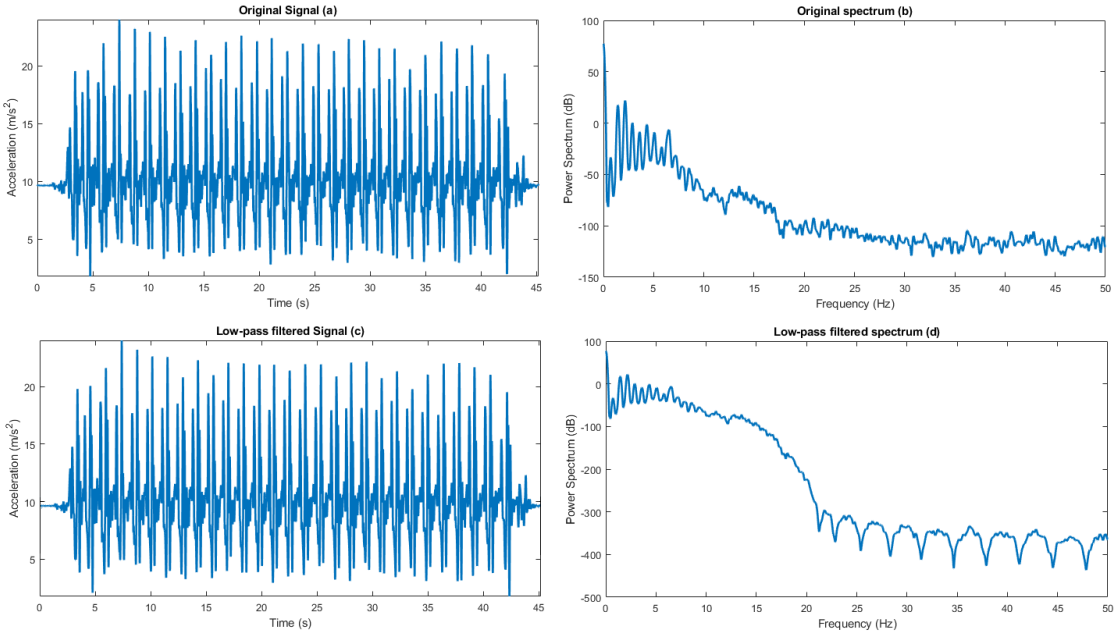


Figure 23: Accelerometer signal before (up) and after (down) filtering.

The cycle extraction process is performed in the same way as in the above algorithm. We use the accelerometer signals from the right phone to calculate the start and end points of the cycles. Once all the points are obtained, all the signals have identical timestamps so that we can divide them into GCs.

4.2.1.2 Feature Extraction

As already mentioned, this algorithm is based on the one presented above, so the features used in that algorithm are also used in the current one. However, some additional features are obtained by using the information from both legs. The new features are described below.

GCs similarity This parameter measures the similarity of the GC from both phones using cross-correlation. The value used is the maximum value of the process.

GC distance This parameter measures the CGs distance, which is another measure to quantify the likeness of the GCs. It is performed using the Dynamic Time Wrapping (DTW) [85]

Middle peaks difference This parameter measures the time difference between the central peak of both GCs. An example of it is shown in Figure 24.

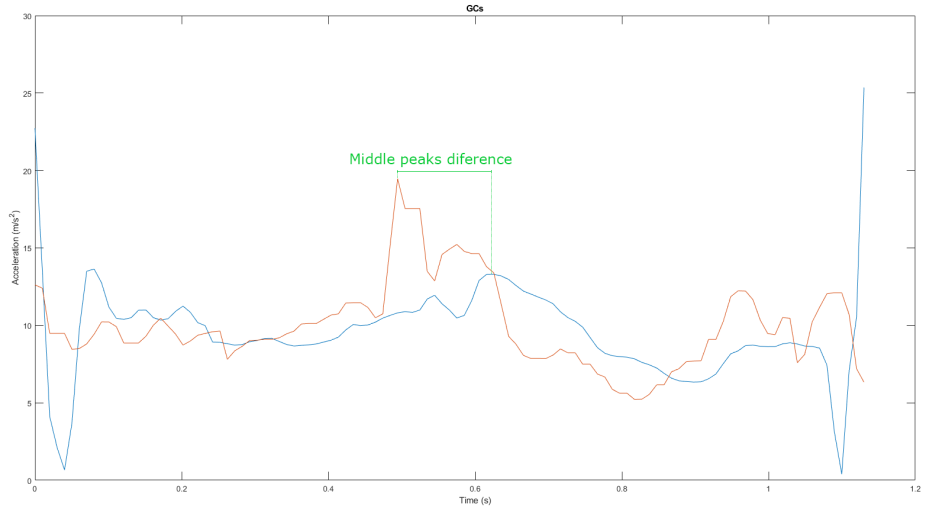


Figure 24: Accelerometer signals from both phones.

In this case, as we have duplicated the number of phones, the number of features is also doubled. In addition, we have calculated new features using the information from both phones. Result of it, in the current algorithm, the number of features increases up to 97. An overall of the features is shown in Figure 25.

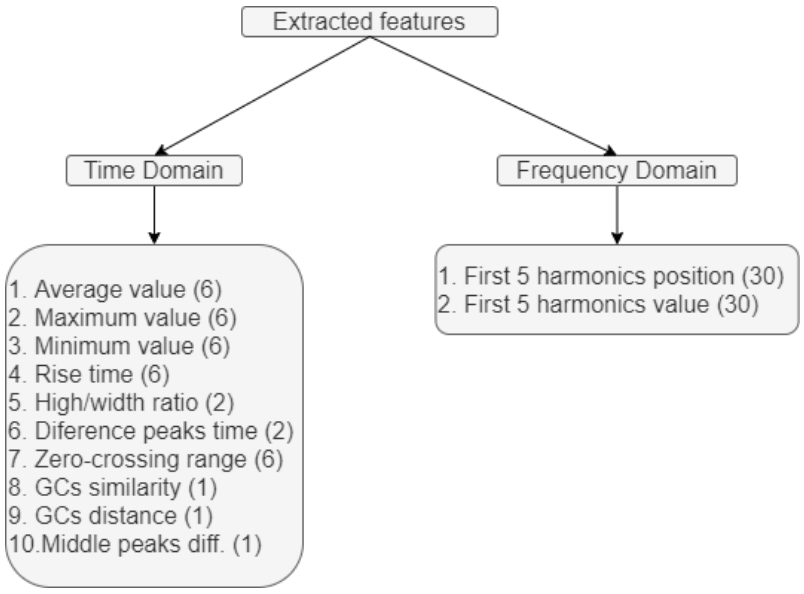


Figure 25: Features obtained in the two-smartphone configuration. In brackets the number of values obtained from each feature.

4.2.2. Evaluation Database

The database used in this experiment follows the same procedures as in section 4.1, as well as the same smartphone, only the number of smartphones and the simulated pathology change.

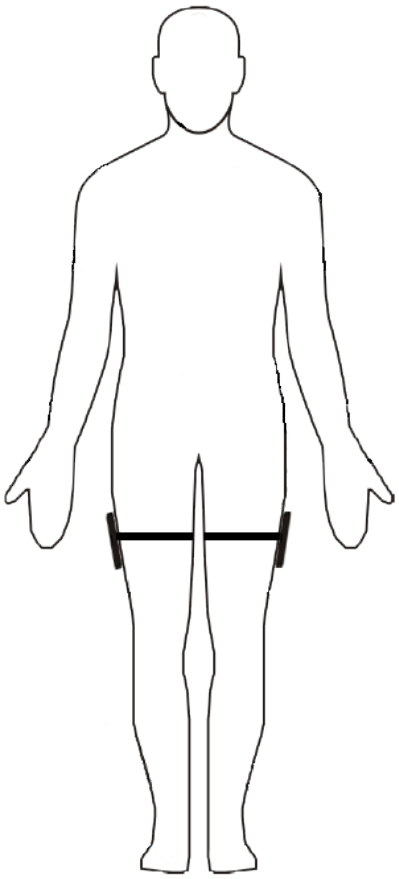


Figure 26: Smartphones position in the two-smartphones configuration.

Placing the phone on the waist does not provide specific information about the legs, so instead of putting the smartphones on the waist, as in the previous case, in this database, they are attached to the thighs. The Figure 26 shows the new positions for the smartphones.

On the other hand, as we now have more information about the strides, a new fake pathology is used instead of unbalanced walking. In particular, bandages are used on both the knee and the ankle to immobilise the joint, to simulate an injury. Figure 27 shows an example of the bandages used.

As there is now an extra pathology, the number of walks the users do has changed. There exist four types of fake pathologies for this experiment: knee left, knee right, ankle left, and ankle right. However, the fake pathologies of the knee and the ankle are different, regarding the algorithm, they are considered alike, as the algorithm is in an early stage to add the difficulty of adding two more cases to identify.

As one of the requirements of the machine learning algorithms is a balanced database, instead of performing four healthy walks as in the previous database, the users make eight healthy walks. In overview, each user must perform eight right pathology walks, eight left pathology walks and eight healthy walks. With these requirements, 12 people were

Figure 27: Knee bandage used to simulate the pathology.

recruited for the database, Table 4 shows an overview of the dataset obtained.

Table 4: Two-smartphones dataset details.

Users	Number of walks		Number of cycles
	Pathological	Healthy	
12	192	96	8064

We must highlight that even if the data acquisition is performed with two smartphones, the smartphones data counts as one since it is used together, i.e., for each cycle measured, there are one of each smartphone, including all the signals from the different sensors.

4.2.3. Results

As mentioned above, the ML algorithms used for this algorithm are the same as before, plus the percentage of training/test data, on the other hand, the algorithm is still run five times, and the results shown here are the average of those runs. After feeding them with the new data, we obtain the confusion matrices shown in Figure 28.

Observing the confusion matrices indicate that the relationship between the different cases remains stable for the CART, kNN and Logistic Regression algorithms. For the SVM case, it can be observed that the highest percentage is now in the correctly classified

healthy walks, so the overall accuracy of the algorithm increases. Finally, the Naïve Bayes algorithm goes from classifying GCs almost randomly to having a similar relationship to the other algorithms. To analyse the results deeply, the Table 5 shows the metrics obtained from the matrices.

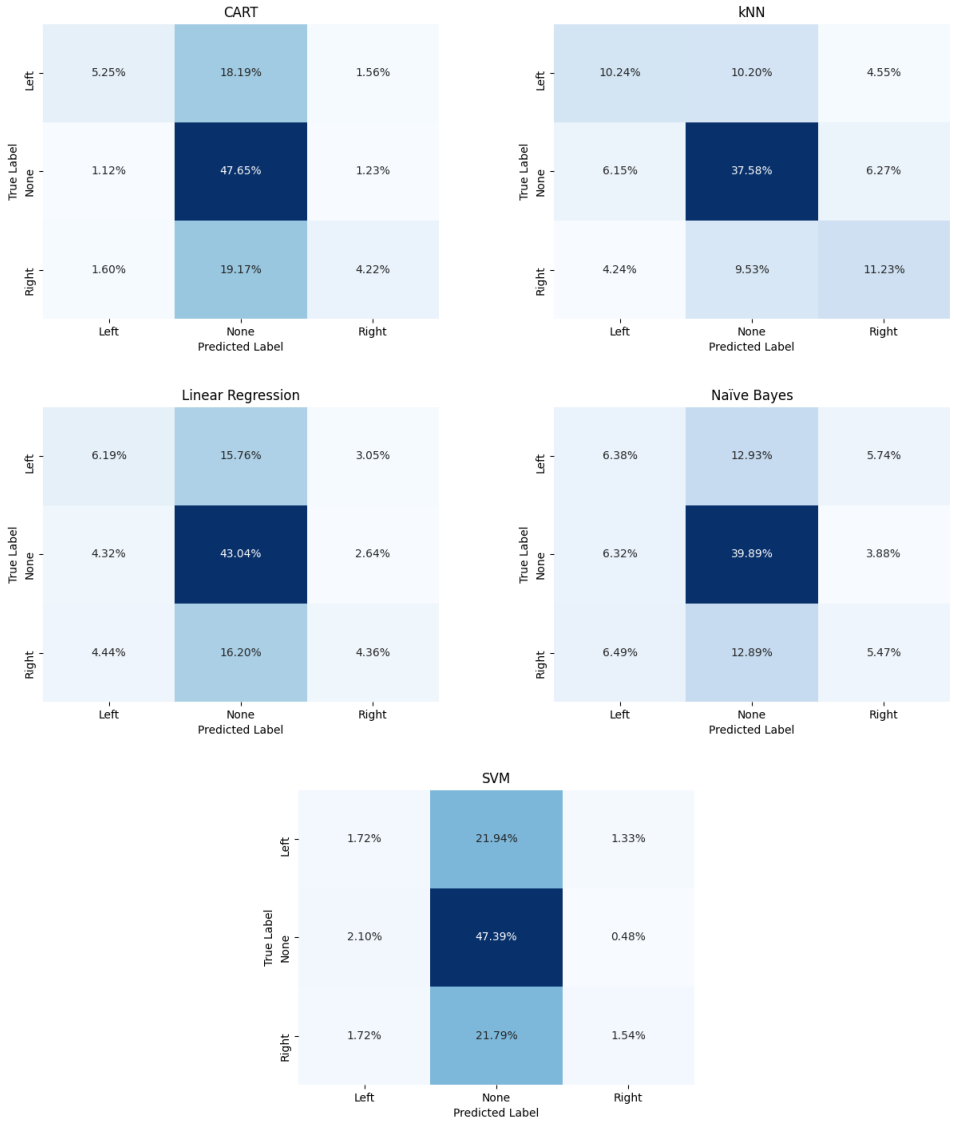


Figure 28: Confusion matrices for two-smartphone configuration.

Focusing on the accuracy, it may be seen that all five algorithms perform better than when using a single phone to acquire the database. This is expected behaviour, as the more the data, the better the ML algorithms perform. However, the increase in accuracy is not as high as expected. Moreover, all five algorithms have a considerable FNR, which rules them out as valid algorithms. Nevertheless, we can take this as a starting point to improve the algorithm. Increasing the number of smartphones in the database has increased the algorithms accuracy by about 10 %. However, instead of directly acquiring a new database, an experiment is conducted by eliminating the most disparate cycles since they may reduce the algorithms learning capacity.

Table 5: ML algorithms metrics for 2-smartphone configuration.

ML algorithm	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)
CART	37,36	2,35	3,16	57,13
kNN	19,74	12,42	8,79	59,05
Logistic Regression	31,96	6,96	7,49	53,59
Naïve Bayes	25,82	10,11	12,28	51,79
SVM	43,68	2,61	3,06	50,65

To obtain the most similar peaks, cross-correlation, which measures the similarity of the signals, is used. The cross-correlation is performed between all the cycles of the same walk, including all the signals of a single smartphone, and these values are stored in the matrix C. Then, all the rows are added, resulting in the vector C'. The last step is to sort the GCs in the function of the score, from highest to lowest.

Once the GCs have been ordered by similarity, it remains to decide how many will be used and how many will be discarded. Discarding too many cycles can lead to a decrease in the accuracy of the algorithm by drastically reducing the number of valuable data. Conversely, not dumping enough cycles may not solve the problem at hand.

In order to decide on an acceptable threshold for the problem, the following is considered:

- On a walk, a user performs on average 28 cycles.
- Empirically, it has been found that both the first two and the last two cycles tend to have both an abnormal shape and a reduced amplitude.

Considering this, we can set the limit of eliminated cycles at 14 % so that we would be left with 86 % of the cycles. It should be clarified that the most logical solution would be to set the threshold to the value of the cross-correlation, however, it has been observed that when performing pathological walks, the similarity values between the cycles are very different from one person to another. Therefore, the most optimal solution found is the one mentioned above. Once the GCs have been removed from the dataset, we proceed to rerun the algorithms, Table 6 shows the metrics of this new execution.

Analysing the new values, it may be seen that, by removing the most dissimilar GCs, the algorithm accuracy rises by an average of 4 %. Currently, all the algorithms have an accuracy higher than 50 %, reaching the 63 % in the case of kNN algorithm. Even though the algorithms have been improved, this improvement is not good enough to consider the algorithms as valid.

Table 6: ML algorithms metrics for 2-smartphone configuration after removing the dissimilar GCs.

ML algorithm	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)
CART	34,74	2,02	2,83	60,41
kNN	18,18	10,86	8,01	62,95
Logistic Regression	29,87	6,91	7,44	55,79
Naïve Bayes	24,40	9,54	11,43	54,64
SVM	36,63	2,30	2,59	58,48

4.2.4. Conclusions on the use of two smartphone for pathology detection

In the previous section, we questioned whether the lack of information was limiting the results of the algorithms. By adding another phone to the DB acquisition, we have increased the amount of information in the database and been able to extract new features. As a result, we have seen how the algorithms have improved by an average of 14 %, although the two algorithms that have benefited the most are Naïve Bayes and SVM, which have improved by 20 % and 27 % respectively. We also have brought up to light that using the most alike GCs increases the accuracy, so it is not only the quantity of data that matters but the quality of the data. Although the increase in accuracy is not significant enough to consider the algorithms as valid, this opens a new door as it confirms that the information was insufficient. We proceed with the 4-smartphones configuration.

4.3. Quad-smartphone Pathology Detection

As we have seen, increasing the amount of data in the database by using more smartphones increases the algorithm accuracy. In this chapter, we employ four smartphones to acquire the database. By this change, we lose entirely the possibility for a user to detect pathologies by himself. However, if good results are obtained with this configuration, we can conclude that the bad results of the previous configurations are due to the lack of information. On the contrary,even if the amount of information received by the algorithm is increased, an accurate result is not obtained, we can deduce that it is due to the smartphones fault, as they do not manage to acquire the signals with sufficient quality.

4.3.1. Algorithm

The current algorithm structure is the same as the one used above, including discarding the dissimilar cycles. A schema of the algorithm can be seen in Figure 29. As happened previously, the features used in the previous algorithms remain in the present one.

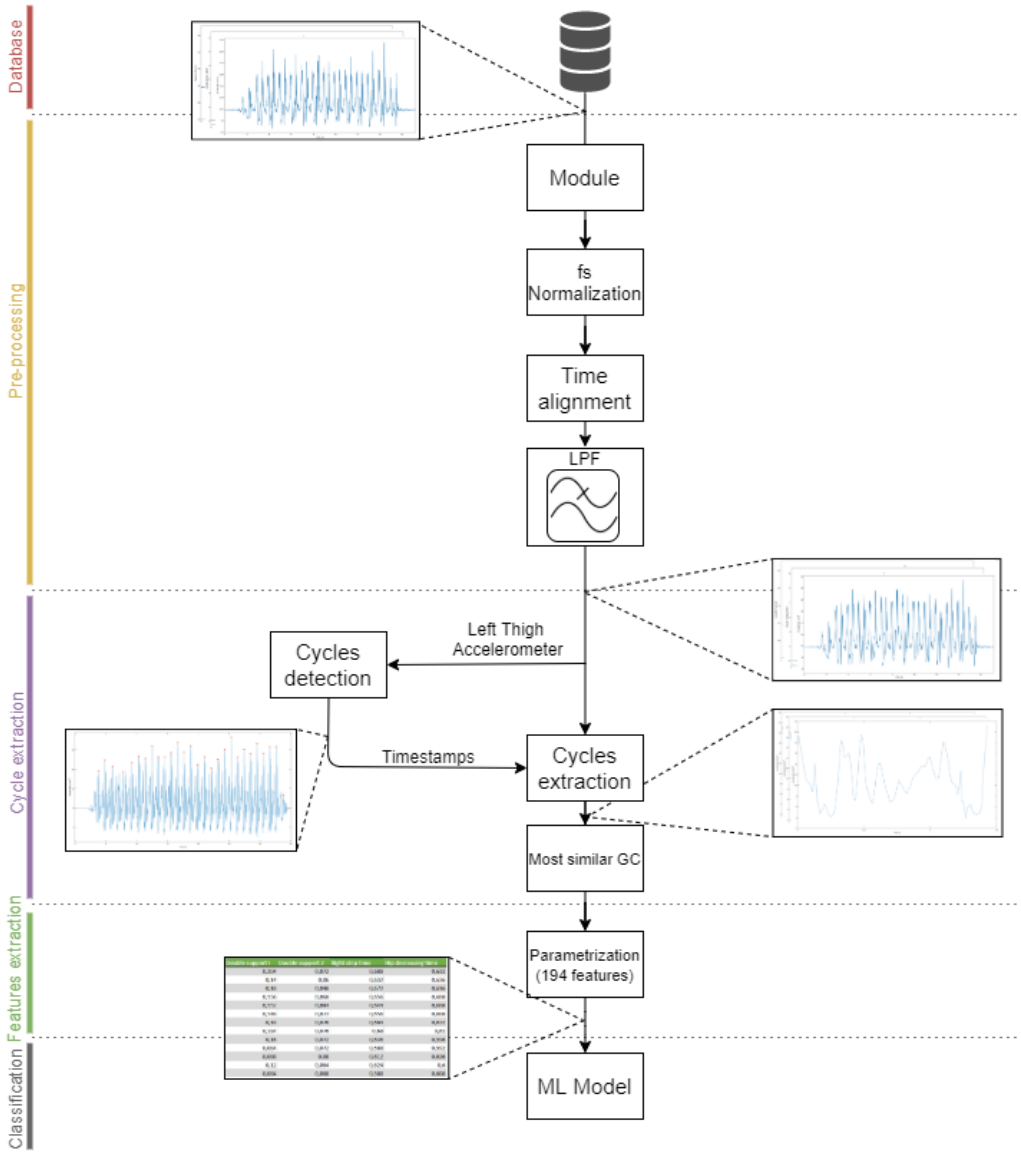


Figure 29: Four-smartphones configuration algorithm schema.

4.3.1.1 Feature Extraction

As in the past, as the number of smartphones increases, the number of features obtained increases. In this case, no new features are calculated, but the existing ones are applied to the new signals. A summary of the features can be seen in Figure 30.

4.3.2. Evaluation Database

This database is acquired the same way as the previous one, following it is shown an overview of the proceedings:

- The users wear the same short trousers and the same shoes. In this way, we can ensure the non-influence of unwanted variables.

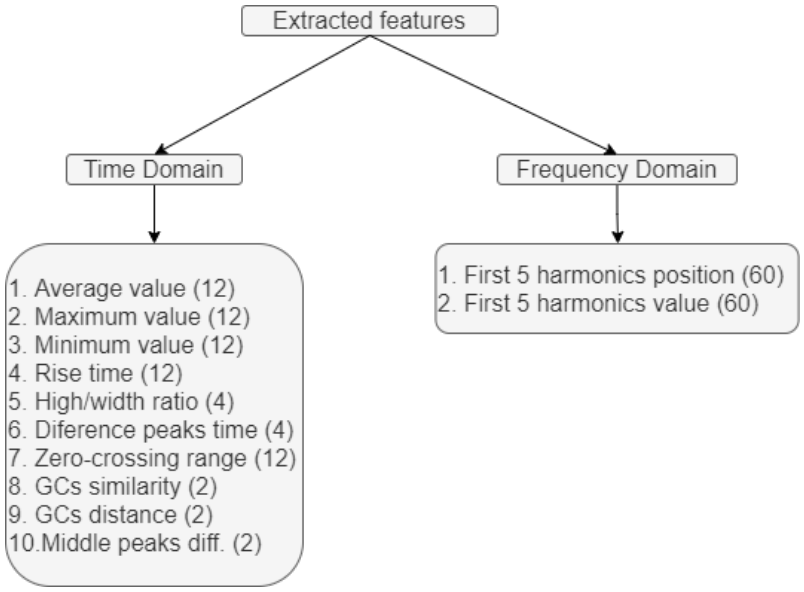


Figure 30: Features obtained in the four-smartphone configuration. In brackets the number of values obtained from each feature.

Table 7: Four-smartphones dataset details.

Users	**Number of walks**		**Number of cycles**
	Pathological	Healthy	
14	224	112	9408

- The users perform the acquisition in a 40 meters flat corridor, with marking and the beginning and end of the path.
- The smartphones are placed vertically on the outer side of tights and shin using a holder. A schema of the positions can be seen in Figure 31.
- The pathology used is a fake pathology, simulated by bandages on the knee or the ankle.
- Each user must perform 16 walks: 8 healthy walks, 8 right pathology walks, and 8 left pathology walks.

With these requirements, a database of 14 users was acquired. A summary of the obtained datasets is shown in Table 7.

4.3.3. Results

As in the previous experiment, the machine learning algorithms used to classify the data are the same as used above. The data has still been prepared the same way, the dataset is divided into 30 % for training, and 70 % for testing and the algorithm run five times. The results of the execution of the algorithms with the new data are shown in Figure 32.

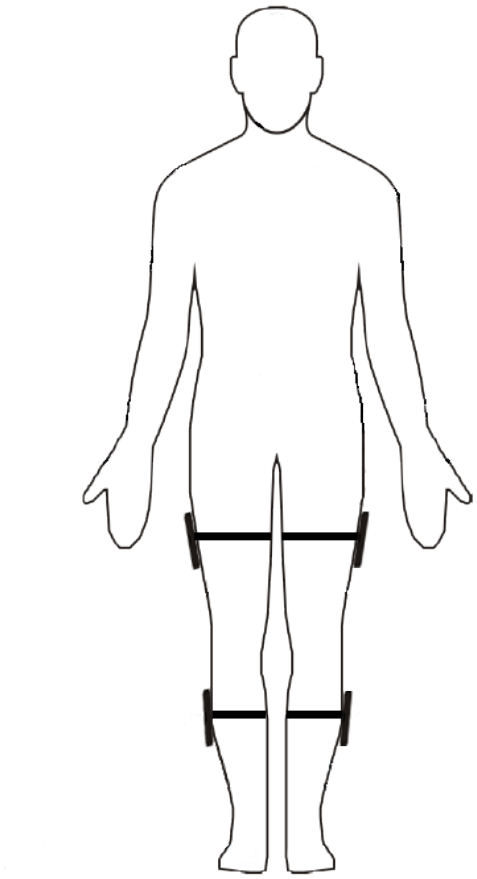


Figure 31: Smartphones position in the four-smartphones configuration.

Looking into the colour distribution, we see that except Naïve Bayes, the other algorithms have reduced the vertical line in the middle. This can be interpreted as the algorithms now learn the pattern more precisely it produces an accuracy increase. Moreover, the kNN algorithm has changed the vertical line for a diagonal one, which means that the number of pathological walks classified as healthy has been reduced. As always, to obtain a deeper understanding of the algorithms, the metrics are displayed in Table 8.

Table 8: ML algorithms metrics for 4-smartphone configuration.

ML algorithms	**FNR (%)**	**FPR (%)**	**FLR (%)**	**Accuracy (%)**
CART	25,31	2,64	6,85	65,20
kNN	10,61	14,33	7,35	67,72
Logistic Regression	24,35	5,46	8,95	61,24
Naïve Bayes	35,93	3,13	4,28	56,67
SVM	10,35	8,43	14,92	66,30

As it can be appreciated, the accuracy has risen for all the algorithms; however, SVM

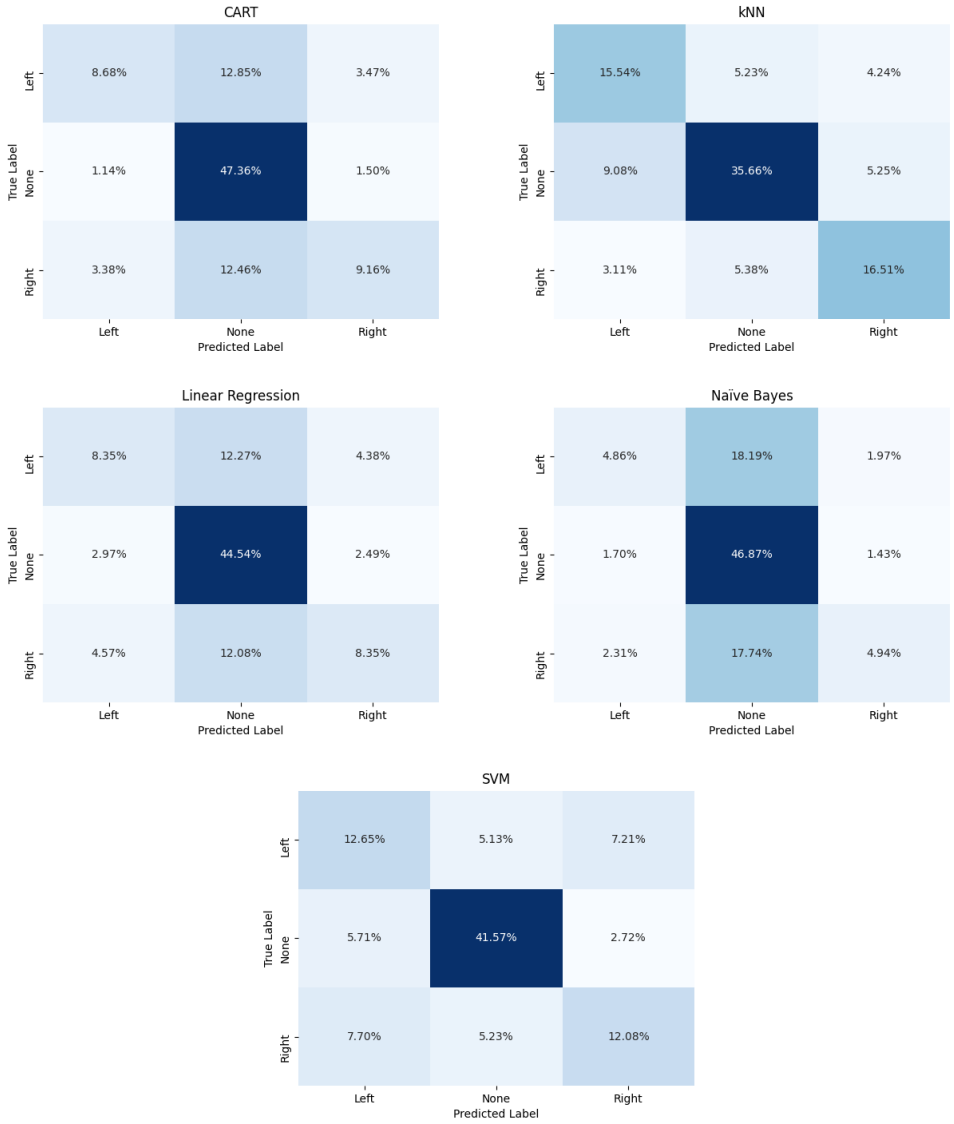


Figure 32: Confusion matrices for four-smartphone configuration.

most takes advantage of adding more data. This behaviour is coherent with the way SVM works, as the more the data it uses, the better it can calculate the support vectors to divide the regions. The other algorithms have an improvement of about 3 %. In addition, the FNR has decreased significantly, except for Naïve Bayes, which has increased, resulting in safer algorithms, so we have reduced the number of pathological walks classified as healthy. Once again, the most improved algorithm (in the case of FNR) is SVM, though it is tied with kNN with 10.5 % in global terms.

4.3.4. Conclusions on quad-smartphone for Pathology Detection

Analysing the results, it can be seen that the performance of the algorithms increases as the number of devices in the database increases. However, this increase cannot be considered good enough to justify the use of these additional phones. From the poor growth in

accuracy that comes with the rise in the number of smartphones, it can be deduced that the poor results of the algorithm are due to the quality of the signals obtained by the phones and not to the amount of information used.

4.4. Conclusions on the Use of Smartphones for Pathology Detection

Throughout this chapter, we have used different smartphone configurations to acquire a database and to create an algorithm capable of identifying the pathologies being a portable system.

The first attempt to create an algorithm was carried out using one single smartphone, under the premise that using one smartphone is possible to perform a biometric identification. However, by the results shown by the different machine learning algorithms used, it can be concluded that it is not possible to identify pathologies using one smartphone. We can deduce that the information gap about the strides can be the consequence of the poor results, so we decide to use a 2-smartphone configuration to acquire the database.

In the 2-smartphone configurations, we have lost the system portability, as not every user could do it by themselves, however, we have doubled the amount of information. We included a low pass filtering and a selector of the most alike cycles in walk stages along with this configuration. With the new amount of data and the new process in the algorithm, we increased the accuracy of all the algorithms. However, it was not enough to consider the configuration as valid. As the extra information has improved the accuracy of the algorithms, we decided to make a new configuration using four smartphones based on the professional data acquisition systems. As the number of smartphones increases, the algorithms again increase the accuracy, however, this increase is not high enough to consider the configuration valid.

Analysing all the results together, we can conclude that it is true that the more the data we use, the best the algorithms perform, this improvement is not good enough, and it is due to the inherent problems of smartphones. These problems can be defined as follows: firstly, the varying sampling frequency, which compels us to interpolate data to correct it; following, the delay introduced by the phones when starting the acquisition in the configuration with more than one phone, that even though it is fixed in the pre-processing stage, the signals could be sightly misaligned, which can affect to the algorithm; and last but not least, the quality of the signals, which are not precise enough to detail the pathologies. For these reasons, we conclude that smartphones are not an appropriate tool for data acquisition in pathology detection algorithms, and from now on, a professional acquisition system is used.

On the other hand, we have seen that the different algorithms have evolved differently as more information has been introduced into the database, although it is the CART, kNN and SVM algorithms that have finally presented the best results. Based on this fact, to

reduce the complexity of future experiments, only these three algorithms will be used instead of the five used here.

5. RECURRENT NEURAL NETWORK APPLIED TO PROFESSIONAL MOTION CAPTURE SYSTEM

In the last chapter, we have seen that using different configurations of smartphones to acquire a database has resulted in a poor performance to recognise pathology walks. Given these results, it was decided to change the data acquisition system to a dedicated professional system. Using a professional motion capture system, both the resolution of the signals and the number of signals increase significantly. In addition, the problem of synchronisation between the different devices disappears.

Since in the previous chapter we have worked with GCs, this algorithm will approach the problem differently to cover the possibility that dividing the signal into cycles is not the optimal way for pathology detection.

This chapter aims to present an algorithm based on neural networks capable of distinguishing lower body pathologies with the requirement of not having to pre-register to use the system. In addition to this, it is also intended to create a system as portable and less restrictive as possible, solving the problem of going to a specialised centre to perform the motion capture.

The chapter is divided into four main blocks. In the beginning, the database is presented: the capture system used, the acquisition procedure and the final dataset. The following is the proposed algorithm: the pre-processing of the data, the data extraction and the neural network used. Once the main algorithm has been presented, we continue performing experiments with it to evaluate and improve it. Lastly, the final algorithm configuration and the conclusions close this chapter.

5.1. Evaluation Database

A proper database is an essential requirement to evaluate an algorithm as the quality of the database influences the results. Due to the absence of databases with healthy and pathological walk signals, we collected our database for this study. Although any pathology could be used, due to the difficult access to people with one common pathology, this study uses clubfoot walk as simulated pathology to evaluate the system. In this section, the device used to acquire the signals, the protocol and the final dataset are explained.

5.1.1. Capture System

The device used to acquire the database is the Technaid Tech-MCS v3 (Figure 33) [86], a professional tool to register the movement and the orientation of the human body. The

Tech-MCS can be divided into the Tech-HUB and the Tech-IMUs. The Tech-HUB is the device that collects and stores all the data from the IMUs (Inertial Measurement Unit). The IMUs are small electronic devices based on MEMS (Micro Electro-Mechanical Systems) technology. Inside each IMU, there is an accelerometer, a gyroscope, and a magnetometer, all working in 3D. This system allows capturing the acceleration (m/s^2), angular acceleration (rad/s) and the magnetic field (µT) for each IMU. The device has a sampling rate range from 10 to 500 Hz. Table 9 shows the dynamic range and the sensibility of each sensor.

Table 9: Technical specifications of the sensors.

Sensor	Dynamic Range	Sensibility
Accelerometer	±34.9 m/s^2	0.06 mV/o/S
Gyroscope	± 39.22 – 156.88 rad/s	0.122 mg
Magnetometer	± 810 µT	0.092 V/gauss

Figure 33: Capture system Tech-MCS.

The signals obtained by the IMUs are merged together using an Extended Kalman Filter (EKF) [87] in order to obtain the orientation of each IMU, represented by the Direction Cosine Matrix (DCM). By using the DCM, it is possible to calculate the IMUs orientation in all the axis. This process is performed in two steps:

1. First, the initial alignment of the IMUs is estimated through the accelerometer and magnetometer measurements. This process is performed only once when the IMUs are not moving (before motion capture starts).

2. Lastly, the orientation estimation of the IMUs is calculated when the motion starts. This process is performed by integrating the angular velocity signals obtained by the gyroscope.

These steps are used in a sensory fusion process to estimate the orientation from the inertial information executed in each IMU. The Figure 34 shows the schema of the process.

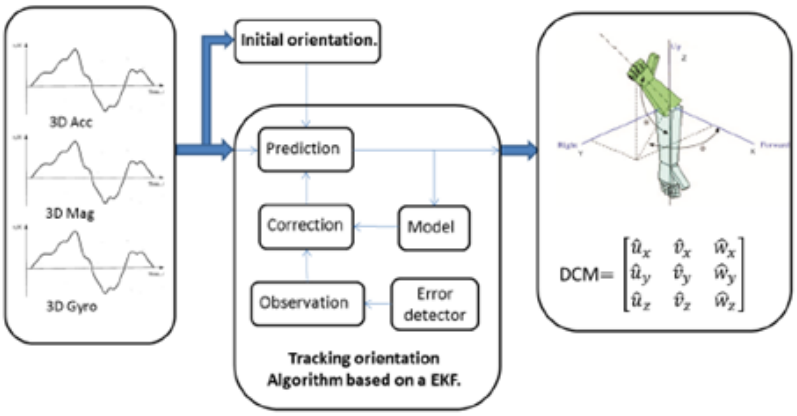


Figure 34: Schema of the process to obtain the 3D orientation.

5.1.2. Acquisition Procedure

As we are interested in the movements of the lower part of the body, the IMUs are placed on both legs as follows: 2 at each foot, 2 at the middle of each shin, 2 at the centre of each thigh and one at the middle of the lumbar. The exact position of the IMUs on the body can be seen in Figure 35. As explained above, the IMUs are calibrated by the Tech-HUB before each walk, obtaining the relative position between them, thereby eliminating differences in the IMUs positions on different visits.

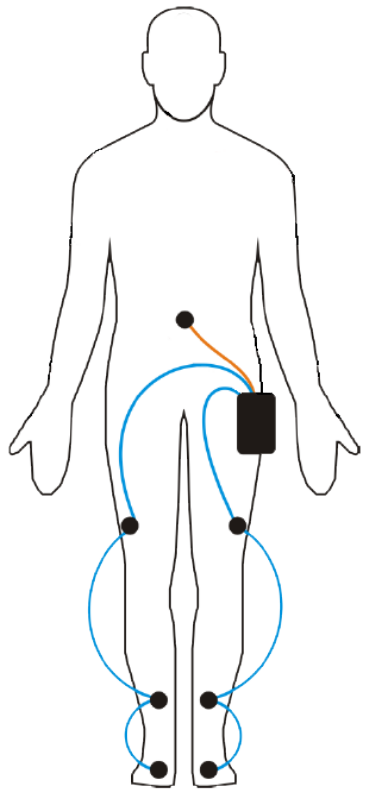


Figure 35: Position of the IMUs on the body

Although Fernandez-Lopez et al. [72] place the gait frequency at around 1 Hz, this is the frequency of the entire movement, however, high-frequency movements can occur in different joints or planes. A sampling rate of 250 Hz is used to account for these movements so that frequencies up to 125 Hz are captured, which is well above the expected.

When designing a database, it is necessary to consider that machine learning algorithms need a large amount of data, furthermore, the data must be balanced, i.e., the number of samples for the different cases must be approximately the same. Having access to healthy people (in the way they do not have any problems in the way of walking) is not a problem however accessing people with a common pathology is another matter.

Because of the difficulty of accessing people with some pathology and the reduced number of people with a common pathology, the database has been created using a simulated pathology. Since a pathology has been simulated, the persons who participated in the database could not have any problems in their walking, i.e., no person in the database could have any gait impairment (sprains, surgeries, flat food, etc.).

In order to choose a pathology to simulate, some aspects must be taken into account, such as:

- **Easiness and comfortability for the users**: it must be easy and comfortable to walk while simulating the pathology. It cannot be painful as the pain affects the gait

parameters [88], [89].

- **Replicability among users**: must be possible to simulate the pathology in all the users, regardless of physiological differences
- **Similarity with real pathology**: to get a database as close to a real problem as possible, the simulated pathology should mimic a real pathology.

A sprain can be simulated with bandages; however, it can be painful for users, and the bandage may be different between users or visits. In addition, clubfoot walk can be simulated using sole padding, which fulfils all the above requirements.

As it is told above, one of the goals of the experiment is to create a pathology detection system for a non-closed environment, however as this is a first approach to creating an algorithm, specific rules have been established to reduce the variability of the data, as well as to facilitate the acquisition of it. The requirements for the acquisition are below.

- The walks are performed by the users wearing their own clothes and footwear at a self-selected speed, with the only restriction of not wearing neither heels nor slippers, as it is not possible to wear sole padding with them.
- The users can choose the place to make the acquisition, but it must be a flat, solid surface of 20 meters long with markings at the start and end of it.
- To avoid spurious data and maximise the amount of data, each user must do three visits within an established schedule: At least fifteen days should elapse between the first and second visit and two months between the second and third visit.
- Each visit consists of 16 walks: eight healthy walks, four left pathology walks, and four right pathology walks.

5.1.3. Dataset

Following the above procedure, a database of 51 healthy people was acquired. Focusing on the number of users, it can be stated that they are too few for an NN algorithm, as the dataset has to be divided into testing and training datasets. However, the amount of data increases substantially due to the number of visits and the number of walks per visit. Of the 51 recruited users, only 32 made the second visit, and only 21 completed all visits, making a total of 104 different visits. As stated above, each visit consists of 16 walks, so taking into account that there are 104 visits, we have 1664 walks. The Table 10 shows a summary of the dataset.

Table 10: Dataset details.

Visit	Users	Number of walks		Number of signals			
		Pathological	Healthy	Accelerometer	Gyroscope	Magnetometer	Angle
1^{st}	51	408	408	5712	5712	5712	14688
2^{nd}	32	256	256	3584	3584	3584	9216
3^{rd}	21	168	168	2352	2352	2352	6048
Total		832	832	11648	11648	11648	29952

On the other hand, 81 different signals are acquired in each walk with a sampling frequency of 250 Hz. Sixty-three of these signals correspond to the information from the kinematic sensors (accelerometer, gyroscope and magnetometer), the other 18 are the angle values from the joints and correspond to different movements of the leg. An example of those movements is shown in Figure 36. The first one belongs to the sagittal plane (x-axis), in which the flexion and extension take place. The coronal plane (y-axis) is the next one; this consists of the movement of the leg from right to left and vice versa. The last one is the rotation of the joints over themselves.

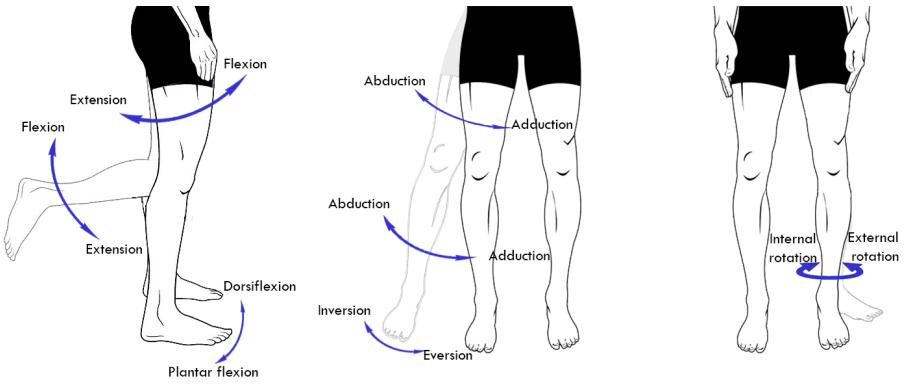


Figure 36: Angle movements acquired by the system.

If we analyse the database distribution (Figure 37), we can see that the gender of the users is balanced with 43 % male users and 57 % female users, however, the age distribution is unbalanced. The main group of users is between 18 and 30 years old, followed by a second group between 50 and 59 years old. This is because the recruitment has been carried out at university, so the main group is made up of people of university age.

5.2. Proposed Algorithm

The objective of the system is to create a system capable of distinguishing between healthy walks, right pathology walks and left pathology walks. As it is shown in Figure 38 the

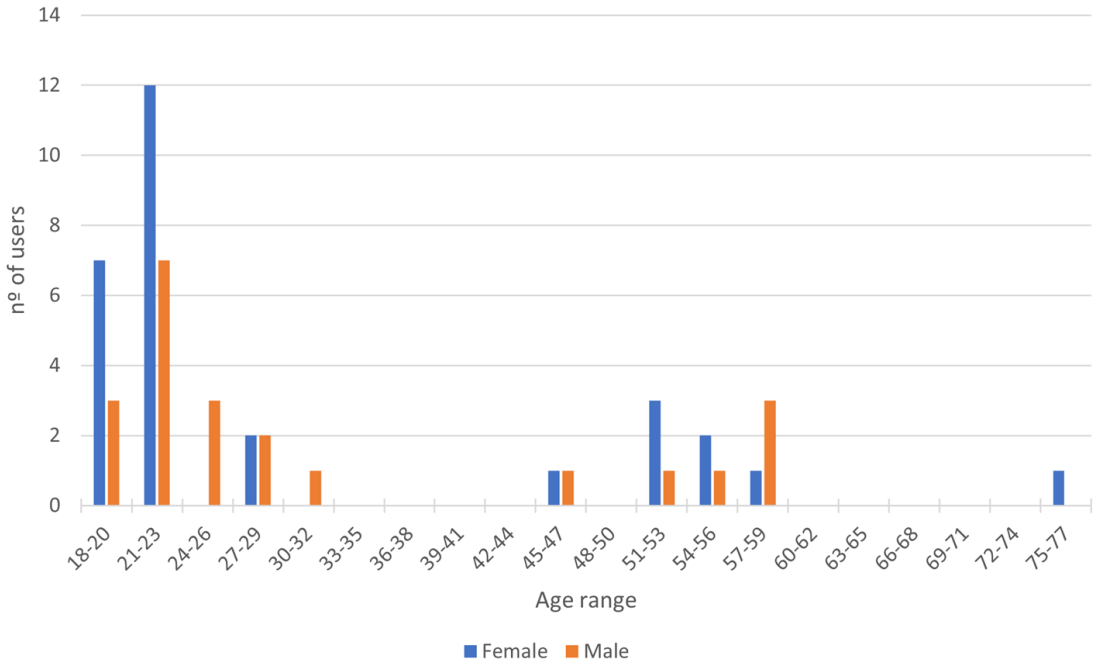


Figure 37: Gender and age distributions of dataset.

algorithm is divided into four different sections: pre-processing, fragment extraction, data preparation and classification. The pre-processing consists of filtering the signals to remove the noise. After the filtering process, the signals are trimmed in order to obtain fragments and using those fragments, the feature matrices are created. Finally, the NN is feed using the matrices. In the following subsections, all the processes are explained in detail.

5.2.1. Pre-processing

Walking is a repetitive movement, right-step, left-step and repeat, these repetitions are called gait cycles, the frequency of those movements depends on the person, however, it can be stated that it is approximately 1 Hz. As a result of the low frequency of the movements, the signals can be filtered in order to remove unnecessary information and noise. Due to the various positions and sensors, the central frequency can vary from one to another. All the signals are evaluated in the frequency domain instead of setting a standard cut-off frequency to prevent losing valuable information. After some testing, we heuristically set the cut-off frequencies at the point where the power spectrum drops approximately below -40 dB. A third-order Butterworth filter is used due to its simplicity and its flat response. In Table 11 the cut-off frequencies that accomplish the mentioned requirement are presented. As we can see, the values from the shin and feet are higher than those from the lumbar and thigh. This is because the shin and the feet make small movements with a higher frequency.

To see how filtering affects the signals, Figure 39 shows the signal (both time and

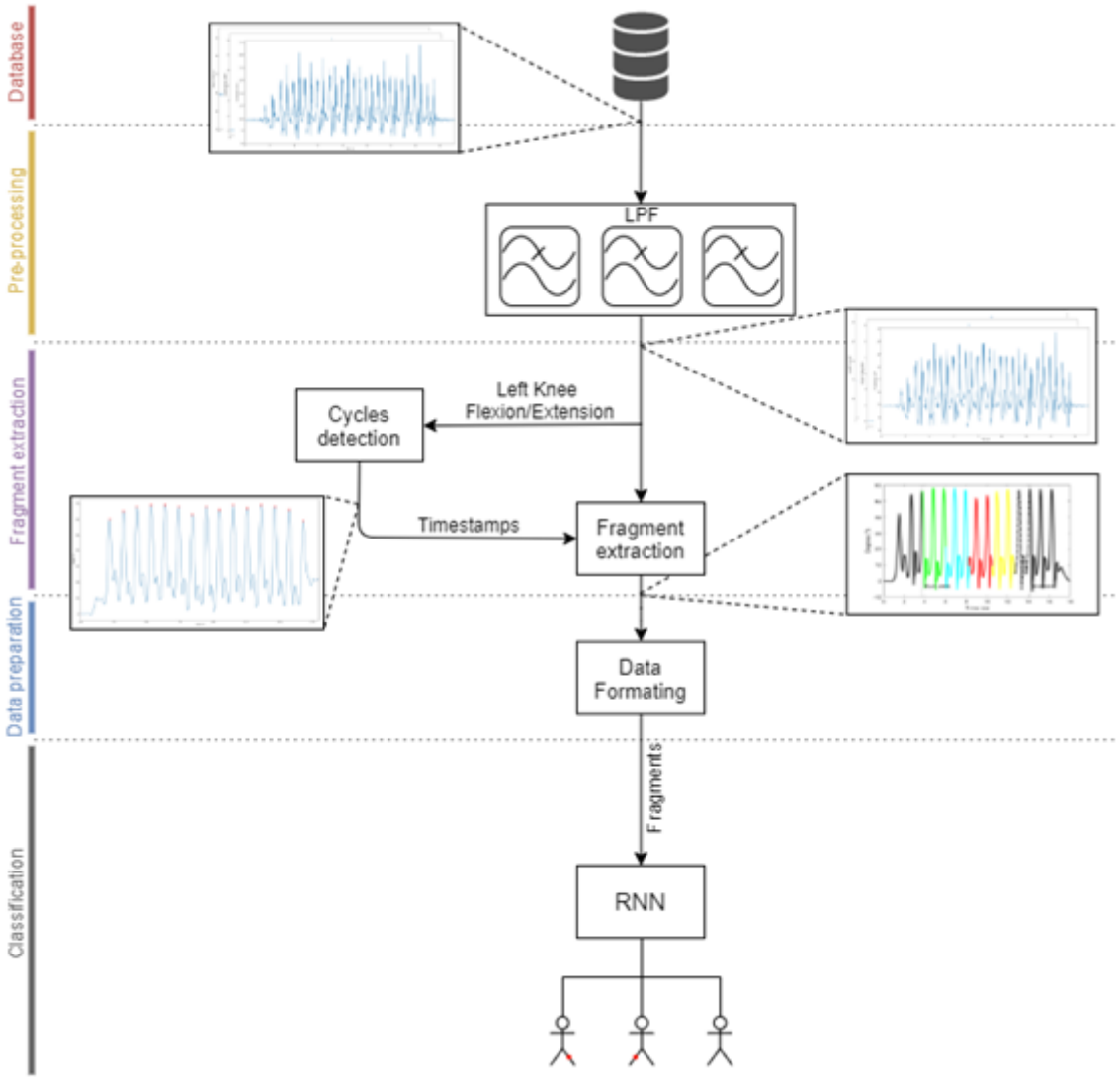


Figure 38: Algorithm schema.

frequency domain) of the left knee before (up) and after (bottom) filtering. As it can be appreciated through this process, the frequency information is reduced, but the signal looks the same in the time domain. Therefore, we can say that the filtering of the signals has removed unnecessary high-frequency information.

5.2.2. Data Extraction

The whole signals could be used to feed the algorithm; however, this would lead to an increase in computational cost. To avoid this and do not use GCs, the signal is divided into fragments, used as the basic unit to feed the algorithm. This not only reduces the computational cost but also increases the amount of data. By using fragments, the evolution from the right to the left and then from the left to the right step can be observed, having in advance much information that using GCs.

The process of finding the start and end points is performed once for all signals

Table 11: Cut-off frequencies for the different signals.

Position	Signal	Frequency (Hz)
All	Angle	20
Lumbar and thigh	Accelerometer	20
	Gyroscope	10
	Magnetometer	10
Shin and feet	Accelerometer	40
	Gyroscope	20
	Magnetometer	20

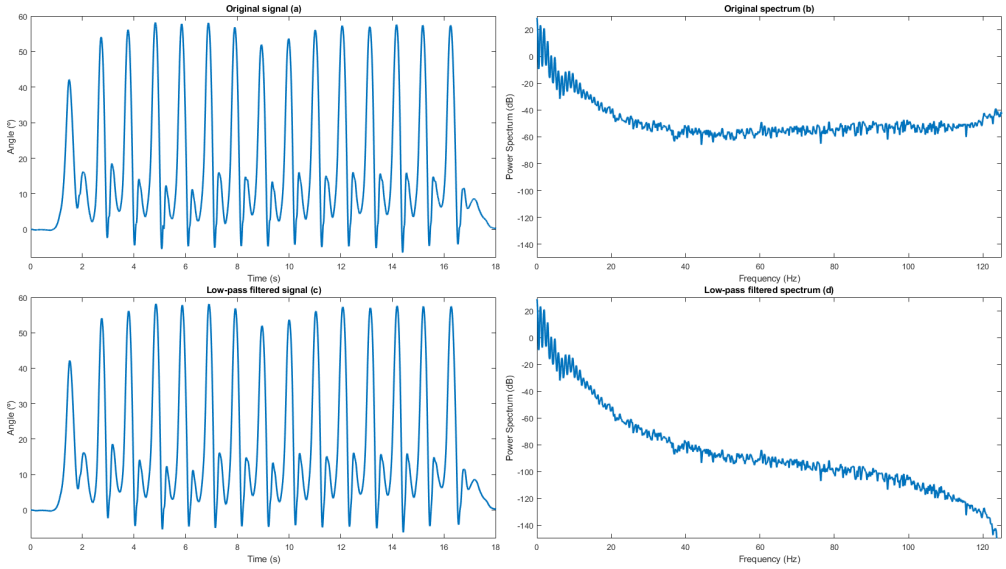


Figure 39: Left knee flexion/extension signal before (up) and after (down) low-pass filtering.

coming from the same walk, as they are all synchronised. To choose the signal that would best achieve the extraction of these points, all signals are evaluated. The signals with the clearest cycles are the hip and knee signals in the z-axis, there are no noticeable differences between the left and right signals, so the left signals were selected. Once the laterality of the signals was established, the hip and knee signals were inspected. The one with the clearest cycles is the knee signal, which corresponds to the flexion/extension movement. Thus, the z-axis signal of the left knee (x_{LK_z}) is used as the reference signal through the data extraction process. The process of obtaining the start and end points is as follows:

1. To begin with, the algorithm looks for all the peaks that accomplish the next relationship:

$$V_{pk} > max(x_{LK_z}) - \frac{max(x_{LK_z}) + \left|min(x_{LK_z})\right|}{3} \tag{7}$$

This means that only the values placed in the upper third of the signal are considered. These points correspond with the leg totally extended forward.

2. The maximum points overlap with other local maxima, so to avoid them, the distance between peaks must be bigger than 0.9 seconds.

$$P_{pk_{n+1}} - P_{pk_{n+1}} > 0.9 f_s \tag{8}$$

3. To avoid instabilities due to the start and end of the walk, there must be at least three seconds left at the beginning and the end of the signal, which corresponds to:

$$P_{start} < 3 f_s \tag{9}$$

$$P_{end} > length(x_{LK_z}) - 3 f_s \tag{10}$$

4. Once the maximum points have been found, the first and the last are used as starting and ending points to trim the signals off. As all the signals have a common timestamp, all of them are trimmed using the same points.

5. It is important pointing that not all the signals have the same duration. Signals from different walks or different users may take shorter or longer. As zero-padding would only affect the last fragment, all the signals are reduced to the length of the shortest one.

6. Finally, after shrinking all the signals, they are divided into four fragments used as the basic unit. An example of the final fragments is shown in Figure 40.

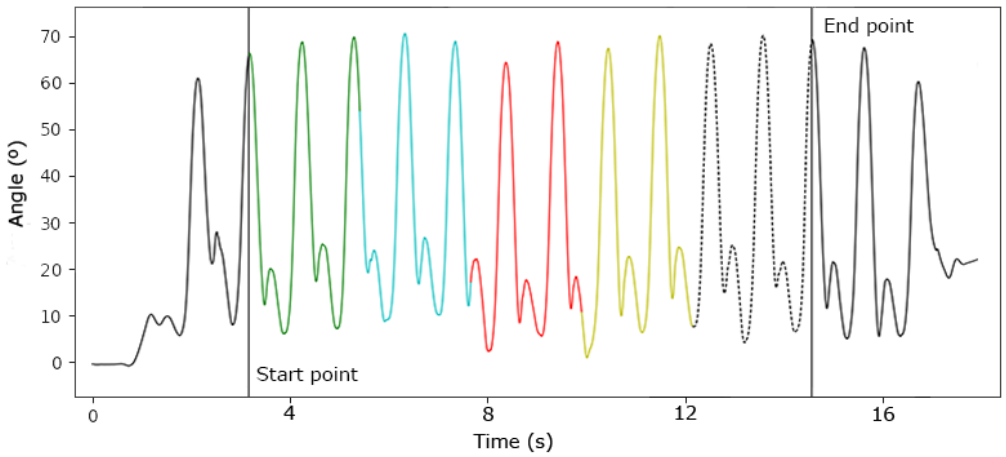


Figure 40: Final fragments differentiated by colours (black is discarded).

The last step is to sort all the data in a matrix of L x 81, where the rows correspond with the time samples of the fragments and the columns with the different sensors. The matrices are used as samples to feed the neural network.

$$V_{pk} = \begin{bmatrix} An_0 & Ac_0 & Gy_0 & Mg_0 \\ An_0 & Ac_1 & Gy_1 & Mg_1 \\ \vdots & \vdots & \vdots & \vdots \\ An_L & Ac_L & Gy_L & Mg_L \end{bmatrix} \tag{11}$$

Where:

$$An_n = \begin{bmatrix} L_{Hip_n} & R_{Hip_n} & L_{Knee_n} & R_{Knee_n} & L_{Ankle_n} & R_{Ankle_n} \end{bmatrix}'$$

$$Ac_n = \begin{bmatrix} Lumbar_n & L_{Thigh_n} & R_{Thigh_n} & L_{Shin_n} & R_{Shin_n} & L_{Foot_n} & R_{Foot_n} \end{bmatrix}'$$

L is the number of samples of the fragments.

The structure of the matrix is as follows. An_n is a vector formed by information from the left hip (L_{Hip_n}), right hip (R_{Hip_n}), left knee (L_{Knee_n}), right knee (R_{Knee_n}), left ankle (L_{Ankle_n}) and right ankle (R_{Ankle_n}) from the n^{th} GC. This information corresponds to the joint angles shown in Figure 36. Similarly, Ac_n corresponds to the accelerometer information, and it is formed by information from the Lumbar ($Lumbar_n$), left thigh (L_{Thigh_n}), right thigh (R_{Thigh_n}), left shin (L_{Shin_n}), right shin (L_{Shin_n}), left foot (L_{Foot_n}) and right foot (L_{Foot_n}). The structure of Gy_n and Mg_n are the same as Ac_n, but they contain the gyroscope and magnetometer information instead.

5.2.3. Classifier: neural network

Neural networks are a set of algorithms that mimics the way the human brain operates to recognise patterns [90]. These algorithms have different attributes, such as adaptative learning, real-time operation, and prognosis, making NNs a powerful tool to solve various problems. The capability of NN algorithms detecting health problems has been widely demonstrated [91], [92].

As we are working with sequential signals, a RNN [93] is used, and in special a LSTM layer [94]. RNNs have a high configuration capability, the problem is that for a new approximation, there is no reference of the number of layers, number of neurons per layer, activation algorithm, filters per layer and so on. These adjustable parameters are called hyperparameters, although finding the best configuration of these parameters to solve the problem can be an arduous task, some techniques facilitate the process, such as GS or RS. To develop an RNN, the python deep learning library Keras [95] is utilised.

As this is the first approach to a neural network for gait pathology detection, rather than creating an NN from scratch, which would be very time consuming (due to the possibilities and computational cost of these), we have relied on previous studies that have used RNNs to detect problems in temporal signals, such as [45], [96], [97] where the RNN are used to detect abnormal electrocardiogram (ECG) and gait signals. As a result of the inspection, the scheme of the Figure 41 is created.

An LSTM layer is used as an input layer, and fully connected layers are used as hidden layers. The number of fully connected layers is established in the hyperparameter optimization process. The output layer is also a fully connected layer with three neurons, coinciding with the number of classes (healthy, right pathology and left pathology). A dropout of 0.2 is performed before every fully connected layer to prevent overfitting and

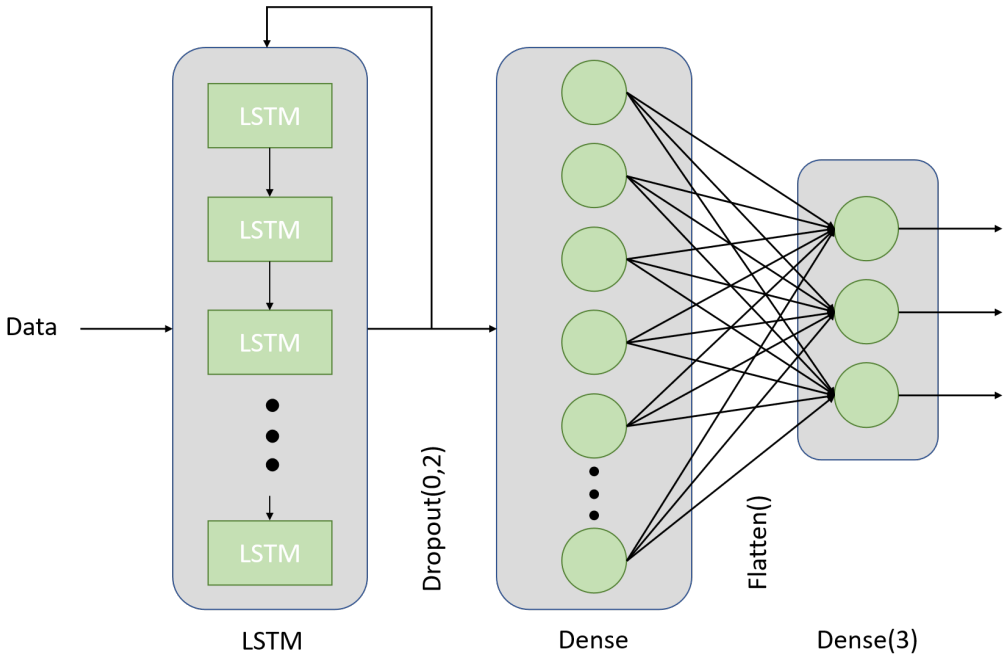


Figure 41: Neural network schema.

increase the accuracy, excluding the output layer.

After the fully connected layers, we use the ReLU activation function to maintain linearity, and after the LSTM layer, a TanH activation function is used. However, for the output layer, we use a sigmoid function to get a probabilistic output. As we are developing a multi-class classification algorithm, the RNN uses a categorical cross-entropy loss function. Finally, adam is used as an optimization algorithm.

Once we have the procedures next step is to find the optimal RNN hyperparameter configuration.

5.2.3.1 RNN optimization

There is no fixed rule about how many layers should be used, although three layers are the minimum of layers to create a neural network. More layers can give better results, but it will be harder to train. Two different configurations are used to find the optimal configuration to solve the problem presented here, one with tree layers and the other one with four. The number of neurons per layer is also studied. To perform this evaluation, we created a relationship between the layers and the number of neurons. The input layer will have 2^n neurons and the fully connected layers (FC) 2^{n-1} neurons. To find the best performing configuration, the value of n starts in 2, increasing by one until the result stops improving. Thus, we can observe how the change in those hyperparameters affects the final accuracy.

Another value to consider is the dataset split ratio. In the state of the art, there is no fixed training/testing ratio for the data, even though it is recommended 20-40 % for testing

and 80-60 % for training. To solve this problem and provide a broader representation of how the algorithm works, we trained the network with different percentages of training data, between 10 % and 90 %. For each ratio of data, the algorithm has been trained and tested ten times, and the final accuracy is the mean value of all of them, furthermore, in each execution of the algorithm, the datasets are randomly built, this ensures that the algorithm is valid for different data and that it is not learning to recognise the same data all the time. The accuracy results obtained in the executions are plotted in Figure 42.

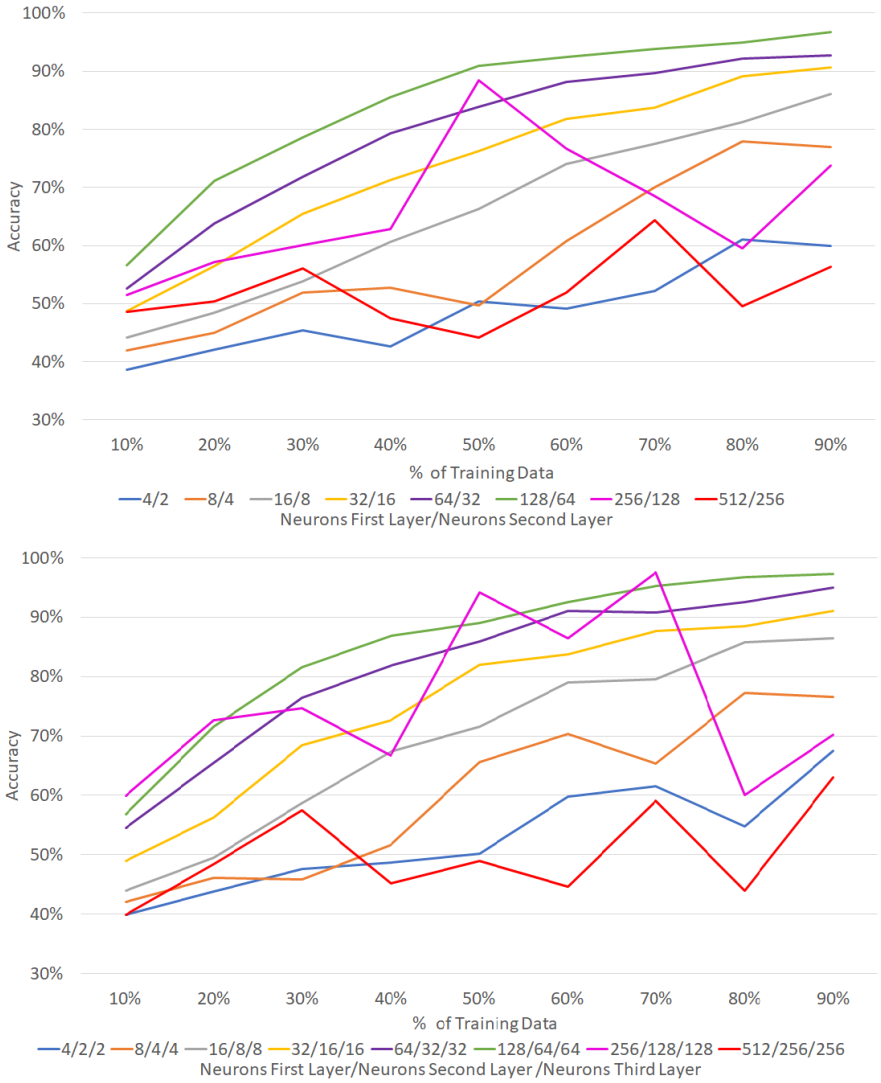


Figure 42: Accuracy results of 3 layers (up) and 4 layers (down) NN. The different neurons per layer are represented by colours.

After the execution, it can be seen that the networks with fewer neurons per layer have the lowest accuracy, however, the configurations with a high number of neurons tend to have unpredictable results. This behaviour is due to the complexity of the RNN: the designs with a low number of neurons do not have enough complexity, and the RNN is not learning the pattern to discern the different classes; in the other hand, the RNNs with many neurons have too much complexity, and them do not generalise well from the training set, tending to overfit. For the case at hand, it can be said that the RNNs that give

consistent results are those with a value of $n = [4, 7]$. Overall, it can be observed that the more neurons the layers have, the better the results, therefore, the best configurations are those with $n = 7$.

The last hyperparameter to check is the number of fully connected layers. To have a clearer view, the two best RNNs are compared in Figure 43. As it can be seen, the results of both RNN are similar, however, the one with more layers presents about 3 % better results. In terms of time, the *RNN_FC2* is the fastest up to 50 % of the training data and the slowest after that, although the time difference is about 10 ms. Because of the results, it can be stated that the best performing RNN is the one with $n = 7$ and $FC = 2$.

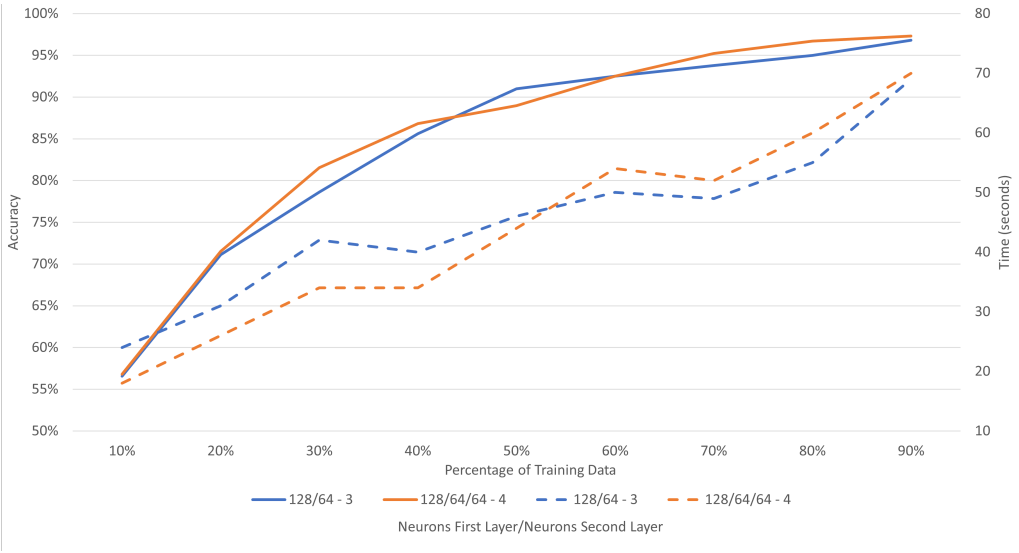


Figure 43: Accuracy results of the RNNs with $n = 7$ and $FC = [1, 2]$.

Once we get the best RNN configuration, it is interesting to try whether it can classify the different cases uniformly. To do this, the network is trained using all the data and evaluated using only the three classes individually (one at a time). As it can be seen in Figure 44, the classification accuracy results of the individual classes show similar behaviour to when all classes are classified at the same time, with a maximum difference of 6 %. Despite these differences, it can be stated that the system is balanced.

At this point, we have achieved an RNN capable of distinguishing between healthy walks and pathological (both right and left) walks, however, there is still room for improvement. In the following section, the different experiments carried out in order to test the algorithm are presented.

5.3. Experimentation

Throughout this section, the different performed experiments are explained. The experiments aim to evaluate the algorithm behaviour in different scenarios and improve it if possible. In the case of the changes made during an experiment enhance the performance of the algorithm, them would be incorporated into the algorithm, and future

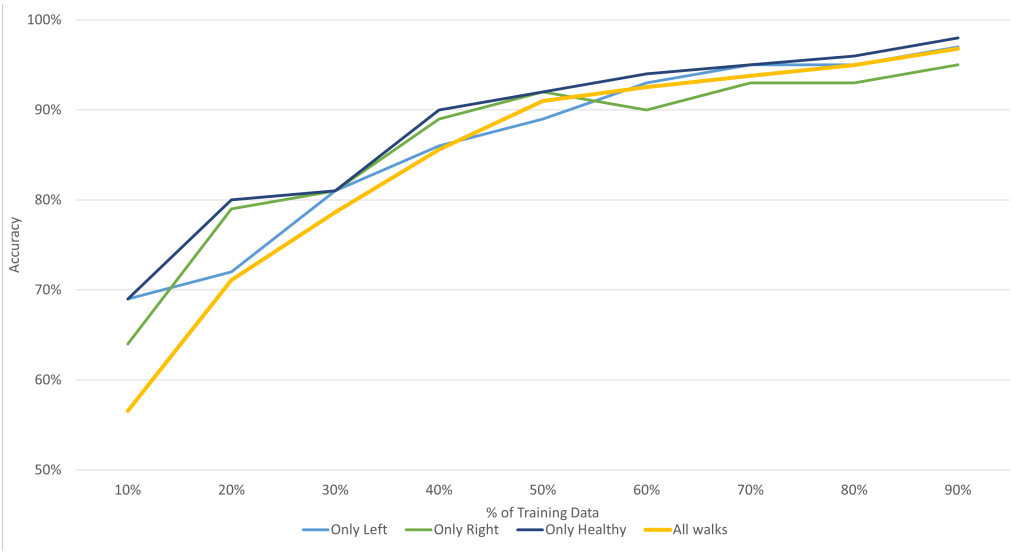


Figure 44: Accuracy results when classifying the cases one at time.

experiments will be performed with the new configuration. The experiments that have been conducted are below.

1. Evaluating the influence of the data origin and the way of dividing it.
2. Studying the influence of the cut-off frequencies.
3. Testing whether it is possible to train the RNN with some users and test with the others.
4. Evaluating the influence adding the first and last 3 seconds, and physiological information.
5. Testing whether it is possible to reduce the number of signals without worsening the algorithm.
6. Evaluating whether all the cycles in a walk have the same importance when classifying the walks.

5.3.1. Influence of the input data

The current algorithm uses signal fragments as the basic unit to feed the RNN, however, there is no evidence that this is the best way to split the data. This experiment aims to evaluate different methods of splitting the data to find the optimal one, which optimises the accuracy of the RNN. To find the best way to split the data, the following methods are evaluated.

- Using the whole signal.
- Using halves of the signal.

- Using quarters of the signal.
- Using a different number of gait cycles.

When referring to the entire signal or any proportion thereof, this refers to the signal contained between the start and end points calculated in subsection 5.2.2. By splitting the signals in different ways, we can see how the amount of data used affects the algorithm.

5.3.1.1 Methodology

Once the signal has been split, the current algorithm considers all fragments as equal, regardless of whether they are from the same or different walks. By doing this, it is possible that fragments from the same walk can be used for both training and testing, making the algorithm incoherent. To avoid this behaviour, a different approach to sorting the data is proposed, in which data from the same walk is not shared between datasets. The Figure 45 shows the schema of the new method of data organization.

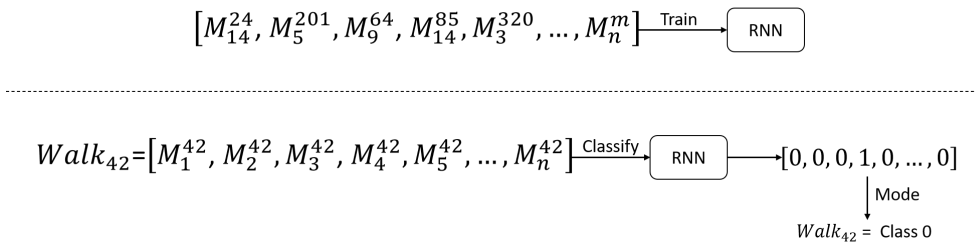


Figure 45: Dataset organization schema. Above for training, below for testing. m is the number of walks; n is the number of cycles in a walk.

In the new method, the process of creating the training and testing datasets is different. The data from the walks is extracted and used individually as an input to train the network in the training dataset. Moreover, to create the testing dataset, the data is extracted and grouped depending on the walk they come from. In the classification process, the data of the same walk is classified individually, and the mode of the result of classifying all data is used as the result of the walk.

At this point, it has been explained how to calculate the start and end points, and from them, the halves and quarters can be calculated directly by dividing the signal into two or four fragments. However, obtaining the GCs requires a more complex process, although there are specific steps in common with getting the fragments, the process is slightly different. The method of extracting the cycles is detailed below.

1. As done previously we look for the peaks that accomplish the next relationship:

$$V_{pk} > max(x_{LK_z}) - 2\frac{max(x_{LK_z}) + \left|min(x_{LK_z})\right|}{5} \tag{12}$$

In this way, only the peaks placed in the upper part of the signal are considered, however, this formula is less restrictive than Equation 7.

2. The next step is to guarantee that the distance between the peaks is bigger than 0.9 seconds between the points (f_s = sampling rate, P_{pk_n}= position of the n^th peak).

$$P_{pk_{n+1}} - P_{pk_{n+1}} > 0.9 f_s \tag{13}$$

3. The following step consists in discarding all the cycles that are longer than a threshold empirically found, which corresponds to:

$$P_{pk_{n+1}} - P_{pk_n} \leq \frac{\sum_{i=1}^{n} P_{pk_i}}{5n} \tag{14}$$

The purpose of this is to avoid those cycles that due to an error were not correctly acquired, an example of one of those cycles can be seen in Figure 46d.

4. The last step is to discard all the points that are in the first and last 3 seconds to avoid GCs with an uncommon waveform.

$$P_{start} < 3 f_s \tag{15}$$
$$P_{end} > length(x_{LK_z}) - 3 f_s \tag{16}$$

It should be noted that not all the GCs have the same length, even between cycles in the same walk. In order to solve the problem and avoid losing information, the length of all the cycles is increased up to the length of the longer one by zero padding.

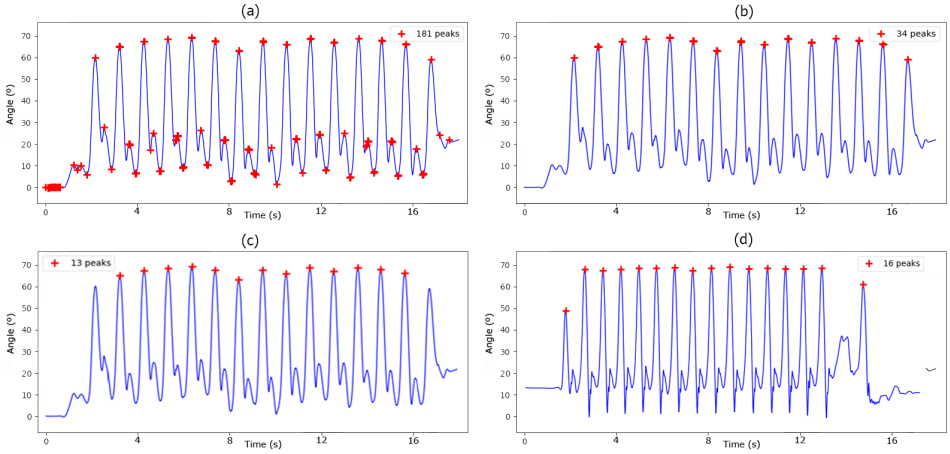


Figure 46: Data extraction process: a) peaks obtained without restriction. b) peaks obtained considering Equation 12. c) peaks obtained considering equations 12 - 16. d)example of a signal with an incorrect cycle.

Figure 46 shows the process of extracting the GCs. In Figure 46a the result of finding the peaks with any restriction can be observed. Figure 46b shows the process of looking for the peaks considering the Equation 12. It is hard to appreciate with the naked eye, but two peaks are detected at each maximum point. To solve the problem, the minimal distance between peaks (Equation 13) is also applied in Figure 46c. In Figure 46c the peaks without the three first and last seconds (Equation 15 and Equation 16) are removed.

5.3.1.2 Results

After splitting the data with different strategies, they are used to feed the RNN. Figure 47 shows the accuracy results of those executions. Using the whole signal, the algorithm has less than 50 % accuracy, making it the worst strategy to split the data. Oppositely, the best results are obtained when using quarters of the signal, so we can state that the larger the data to feed the NN, the less accurate it becomes. Interestingly, when using signal quarters, those from the beginning produce better results than those from the middle or the end.

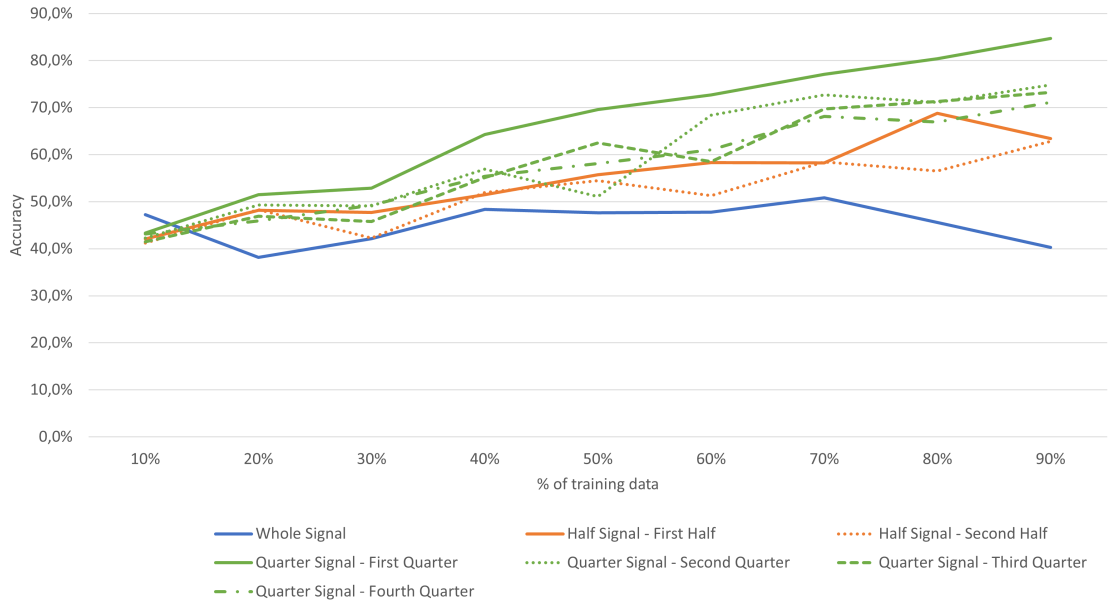


Figure 47: Accuracy results using different strategies to split the data.

The Figure 48 shows the accuracy results when using a different number of GCs. Analysing the accuracy results using GCs, we can see that, as happen above, when using more cycles, the accuracy worsens, so the best configuration is using one single GC as a basic unit to feed the algorithm.

5.3.1.3 Conclusions on the influence of input data

Comparing the accuracy results before and after applying the new dataset organization, it is clear, the new method to create the datasets reduces the accuracy, although, in this way, we have a more realistic system, as all data of the same walk are used to classify it as well.

On the other hand, it can be seen that the smaller the unit used as a sample, the better the results obtained. This behaviour is due to the fact that dividing the data into larger units reduces the number of samples obtained (e.g., using 2 GCs instead of 1 minimises the number of samples by half). As each walk is formed by several GCs, extracting, and using those GCs instead of the whole signal reduces the computational cost of the

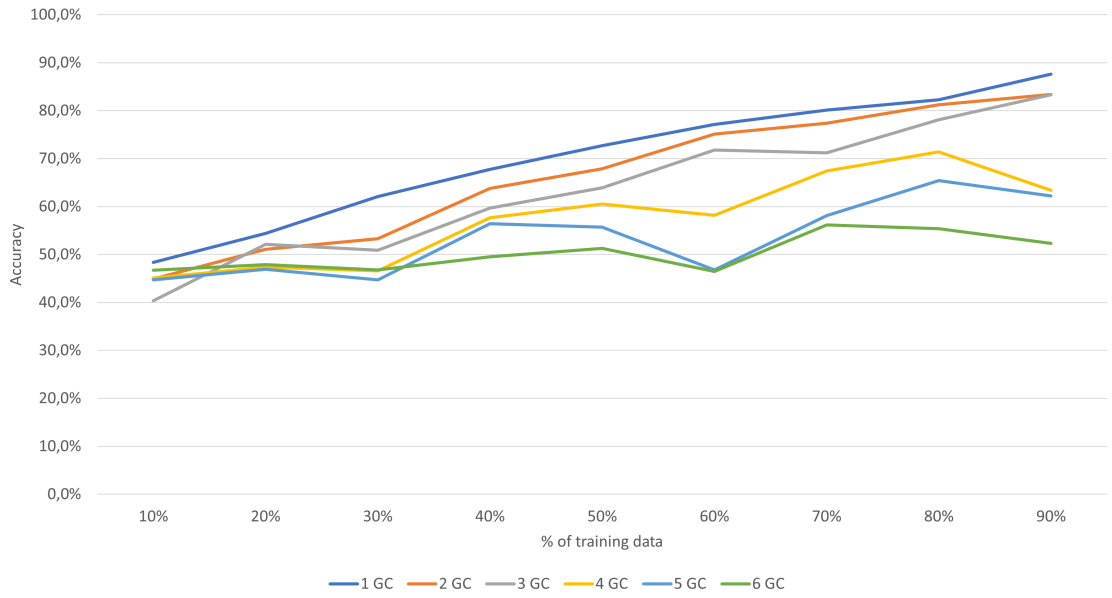


Figure 48: Accuracy results using different number of GCs .

algorithm and increases the data.

In the light of the results, it can be concluded that the smaller the division of the signals, the better the results, so the GCs are the best signal split method. On the other hand, to overcome the worsen due to the new approach to create the datasets, we must optimize the hyperparameters again, however, as we now have a configuration to use as a baseline, this time the optimisation will be done in more depth. The following section details the hyperparameter optimisation process.

5.3.2. Optimising the RNN hyperparameters

Based on the current RNN, hyperparameters can be optimised to increase performance, as they control the network's learning ability. Although finding the optimal configuration of the hyperparameters can be a difficult task, there are different approaches that facilitate the process. In this section, GS has been used to find the optimal hyperparameter configuration.

5.3.2.1 Methodology

As mentioned above, GS is used to find the best hyperparameter configuration. GS works by creating a grid in which upper and lower limits for each hyperparameter are set, and at each run, a value within that range is chosen. When the process has been executed enough times, hyperparameters limits can be reduced by discarding those values which provide lower accuracy, and the process is repeated. Once the resulting grid is small enough, instead of repeating the process, every value is evaluated.

Table 12: Hyperparameters ranges for the first execution of GS optimization.

Hyperparameters	Values
Number of fully connected layers (FC)	[1,5]
Number of filters (NF)	$2^{[4,10]}$
Learning rate (LR)	$10^{[-1,-4]}$
Relationship between layers (RL)	[1,4]
First layer activation	tanh
Fully connected layers activation	relu
Output layer activation	softmax
Optimizer	adam

Table 13: Different relationships used. n is the value under study; m is the number of the fully connected layer.

Number	Relationship
1	2^n
2	2^{n-m}
3	2^{n+m}
4	$2^{n-(m+1)}$

To establish the hyperparameters limits for the optimization process, the current values of the hyperparameters are used as a reference. In that way, the number of hidden layers goes from 1 to 5. As both the number of layers and the number of neurons/filters per layer are under study, their combinations grow exponentially. To reduce the complexity of the study, instead of trying all the possibilities, four different relationships between the number of filters of the first layer and the number of neurons of the fully connected layers are studied, the relationships used are shown in Table 13. Finally, the learning rate goes from 0.1 to 0.001. All other hyperparameters remain untouched. In Table 12 the ranges of values of the hyperparameters are shown.

In the first place, to obtain a general idea about the hyperparameter optimization, the algorithm is run 6 % of the total cases. Once the grid is reduced, the number of executions grows up to 20 % of the new possibilities. After the second execution of GS, the hyperparameters are fine-tuned to find the best configuration. To prevent the algorithm from learning about the same data, in each configuration, the algorithm is run ten times on random data each time, and the result displayed is the average of these runs.

5.3.2.2 Results

After executing the GS algorithm a few times, we have an idea about which configurations perform the best. The accuracy results of these executions are presented in Table 14. It

may be seen that the configurations with one and five FC layers do not show the best accuracy. Furthermore, a low number of neurons per layer does not give a proper result, nor does the higher one. So, we can see that the central configurations are the most optimal. The LR and RL parameters do not show a clear relation about which is better. Because of the results, we set the limit of FC between 2 and 4 and NF from 2^7 to 2^9, in addition, because of its outstanding results, the cases $FC = 1$ and $RL = 4$ are also included. In the case of the LR and the RL, we maintain the same limits. The ranges of the hyperparameters for the second GS execution are in Table 15.

Table 14: Accuracy results of the first run of the GS optimisation (%). The red rectangle delimits the new boundaries for the second run of the GS algorithm. FC: fully connected layers; RL: relationship between layers; NF: number of filters of the first layer; LR: Learning rate.

	FC	**1**	**1**	**1**	**1**	**2**	**2**	**2**	**2**	**3**	**3**	**3**	**3**	**4**	**4**	**4**	**4**	**5**	**5**	**5**	**5**
LR	**NF/RL**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
10^{-1}	2^4															62,5					
10^{-2}	2^4			51,6							44,7			69,7							
10^{-3}	2^4																		56,5		
10^{-4}	2^4					61,2															
10^{-1}	2^5														55,7						
10^{-2}	2^5									60,3											
10^{-3}	2^5																		63,3		
10^{-4}	2^5						65,3						71,6								
10^{-1}	2^6			46,9																	
10^{-2}	2^6																				
10^{-3}	2^6									68,5					78,5					72,5	
10^{-4}	2^6																				
10^{-1}	2^6												85,8				82,6				
10^{-2}	2^7	72,5																			
10^{-3}	2^7								81,6												
10^{-4}	2^7			67,8																	
10^{-1}	2^8														72,2						
10^{-2}	2^8				81,6																
10^{-3}	2^8																				66,3
10^{-4}	2^8							81,5									72,6				
10^{-1}	2^9											79,6			81,5						
10^{-2}	2^9					78,3															
10^{-3}	2^9										75,8										
10^{-4}	2^9		65,7																		
10^{-1}	2^{10}																71,8				
10^{-2}	2^{10}			46,8															68,4		
10^{-3}	2^{10}													65,2							
10^{-4}	2^{10}								41,6												

After establishing the new limits for the hyperparameters, the GS algorithm is rerun. Table 16 shows the accuracy results of the second-round executions.

Analysing the results of the second GS execution, we can see that the configurations with $NF = 2^7$ perform the best. Referring to the relationship between layers, we can see that relations 2 and 4 have the best results. Furthermore, as far as fully connected layers are concerned, the 2 and 3 FC configurations offer the best results. In the case of the learning rate, there is no evidence of what the optimal value is. In order to have a broader understanding of the behaviour of the best performing configurations, they are analysed in deep below.

Due to the changes in LR does not impact the accuracy, the value is fixed to the one in the previous configuration. In the case of the NF, the accuracy is higher with $NF = 2^7$,

Table 15: Hyperparameters ranges for the second execution of GS optimization.

Hyperparameter	Values
Number of fully connected layers (FC)	[2,4]
Number of filters (NF)	$2^{[7,9]}$
Learning rate (LR)	$10^{[-1,-4]}$
Relationship between layers (RL)	[1,4]
First layer activation	tanh
Fully connected layers activation	relu
Output layer activation	softmax
Optimizer	adam

Table 16: Accuracy results of the second run of the GS optimisation (%). The underlined values are those obtained in the first run. FC: fully connected layers; RL: relationship between layers; NF: number of filters of the first layer; LR: Learning rate.

	FC	**2**	**2**	**2**	**2**	**3**	**3**	**3**	**3**	**4**	**4**	**4**	**4**
LR	**NF/RL**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
$\mathbf{10^{-1}}$	$\mathbf{2^7}$								85,8			79,12	82,6
$\mathbf{10^{-2}}$	$\mathbf{2^7}$	79,5		80,91			85,1		86,9		82,5		
$\mathbf{10^{-3}}$	$\mathbf{2^7}$		84,2		85,6	78,5				79,5			81,9
$\mathbf{10^{-4}}$	$\mathbf{2^7}$			79,2				82,6					
$\mathbf{10^{-1}}$	$\mathbf{2^8}$	71,5				72,6					72,2	72,9	
$\mathbf{10^{-2}}$	$\mathbf{2^8}$				80,9					73,6			
$\mathbf{10^{-3}}$	$\mathbf{2^8}$												
$\mathbf{10^{-4}}$	$\mathbf{2^8}$	73,5		81,5				75,2				73,9	72,6
$\mathbf{10^{-1}}$	$\mathbf{2^9}$							79,6			81,5	80,2	
$\mathbf{10^{-2}}$	$\mathbf{2^9}$	78,3		78,2			73,6						
$\mathbf{10^{-3}}$	$\mathbf{2^9}$						75,8		75,9	80,2			83,7
$\mathbf{10^{-4}}$	$\mathbf{2^9}$	76,2			79,15							82,2	

so in the fine-tuning process, it remains unchanged. The relationships with the better performance are the $RL = 2$ and $RL = 4$, so only them are used in the second execution. As the best value for FC cannot be discerned, the values of FC goes from 2 to 4, as in the last GS execution. Finally, the training/testing ratio is added as a parameter under study to the fine-tuning process; in that way, a better behaviour of the algorithm can be observed.

In Figure 49 the accuracy results of best-performing configurations are shown. It can be seen that the worst configurations are those using four fully connected layers. The $NF2^7\_FC4\_RL4\_LR10^{-2}$ and $NF2^7\_FC2\_RL4\_LR10^{-2}$ are the best performing configurations, if we compare them, with a low percentage of training data, the configurations give similar results, but beyond the 40 % of training data $NF2^7\_FC4\_RL4\_LR10^{-2}$ is the best configuration. In the light of the outcomes, we can say that the RNN offering the best results is the one with $NF2^7\_FC4\_RL4\_LR10^{-2}$.

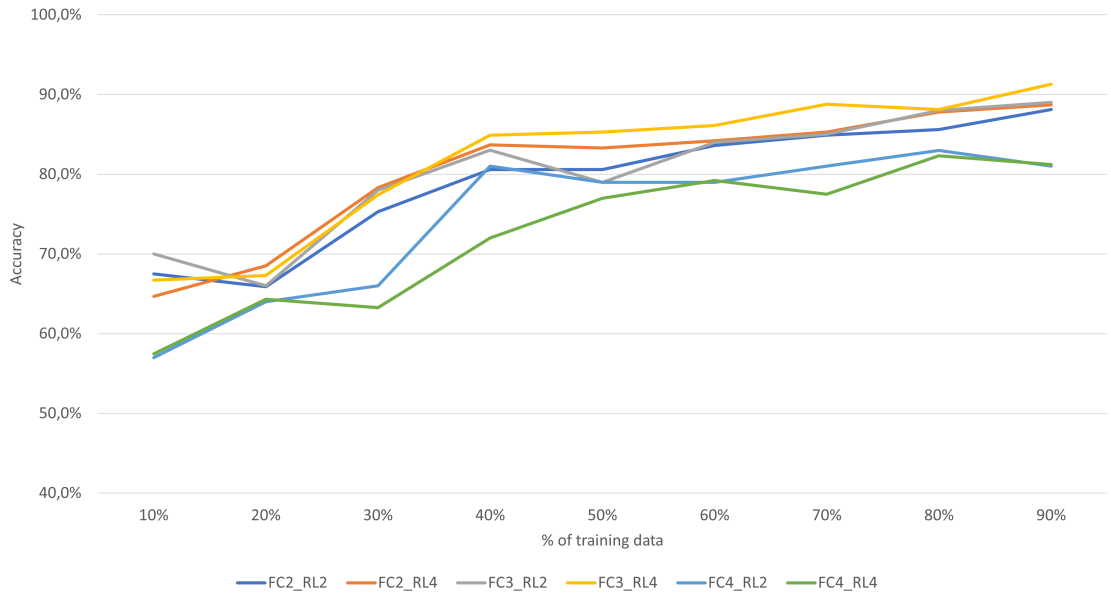


Figure 49: Accuracy results of the best performing configurations after GS optimisation.

Throughout this section, we have tested various hyperparameter configurations, and as a result of these tests, the optimal values for the hyperparameters have been found. In that way, we can see that, although a previous optimisation of the hyperparameters had already been carried out, this had been very superficial, so carrying out a new, more in-depth optimisation, we found the optimum values for the problem.

5.3.3. Influence of filtering the signals

The current algorithm filters the signals in the pre-processing phase. This filtering is based on the premise that the primary information of gait signals is in low frequency, so high-frequency information is not characteristic of the gait. However, this fact has not been tested, so we do not know if the cut-off frequency used is optimal or even if filtering the signals worsens the algorithm performance. To clarify these questions, in this section, we filter the signals with many cut-off frequencies to see the impact of filtering on the algorithm accuracy.

5.3.3.1 Methodology

As mentioned before, when working with multiple signals and multiple positions, a standard cut-off frequency would not work, as each signal has a different nature from the other. In order to choose the cut-off frequency of the signals, a common method is established for all signals: the cut-off frequency is the point where the power spectrum of the signal drops below -40 dB. Thus, to evaluate how to affect the filtering to the performance of the algorithm, the signals have been filtered using different limits for the power spectrum, resulting in different cut-off frequencies. The different cut-off

Table 17: Cut-off frequencies used in the experiment.

Position	**Signal**	**Cut-off limit (dB)**			
		-40	-20	-10	-5
		Cut-off frequency (Hz)			
All	Angle	20	10	6	4
Lumbar and thigh	Accelerometer	20	10	8	3
	Gyroscope	10	5	4	2
	Magnetometer	10	5	5	3
Shin and feet	Accelerometer	40	20	13	6
	Gyroscope	20	10	7	5
	Magnetometer	20	10	5	4

frequencies used are shown in Table 17.

The Figure 50 shows an angular and an accelerometer signals throughout the filtering process with different cut-off frequencies. Comparing the raw signals, the accelerometer signal is slightly noisier than the angle signal, so throughout the filtering process, we appreciate that the filtering affects the accelerometer more than the angle signal. We can observe that as the cutoff frequency is reduced, the signals become smoother, this is especially noticeable in the accelerometer signal, as it has more high frequency information. When filtering with the -40 dB cut-off frequencies, the shape of the signals does not change, but the minor variations of the accelerometer signals are reduced. By switching to the -20 dB cut-off frequencies, we see that the accelerometer signal is completely smoothed out, i.e., the small oscillations have been removed. We see that filtering using the -10 dB cut-off frequencies starts to affect the shape of the signal, and not only the tiny oscillations are removed but also more significant oscillations (some up to 2 m/s^2). Lastly, when filtering with the -5dB cut-off frequencies, we can see that the accelerometer signal has lost relevant information, as well as the shape differs from the original one.

After filtering the signals with different cut-off frequencies, it remains to see how it affects the algorithm performance.

5.3.3.2 Results

To get an idea of the algorithm performance using the filtered signals, the algorithm is trained and tested using the new signals. As mentioned above, for each accuracy value, the algorithm is trained ten times with random data each time. The Figure 51 shows the accuracy results of the algorithm using the different filtered signals. When analysing the results, it is clear that the filtering of the signals affects the algorithm performance. Focusing on the different configurations, we can see that the worst-performing are those

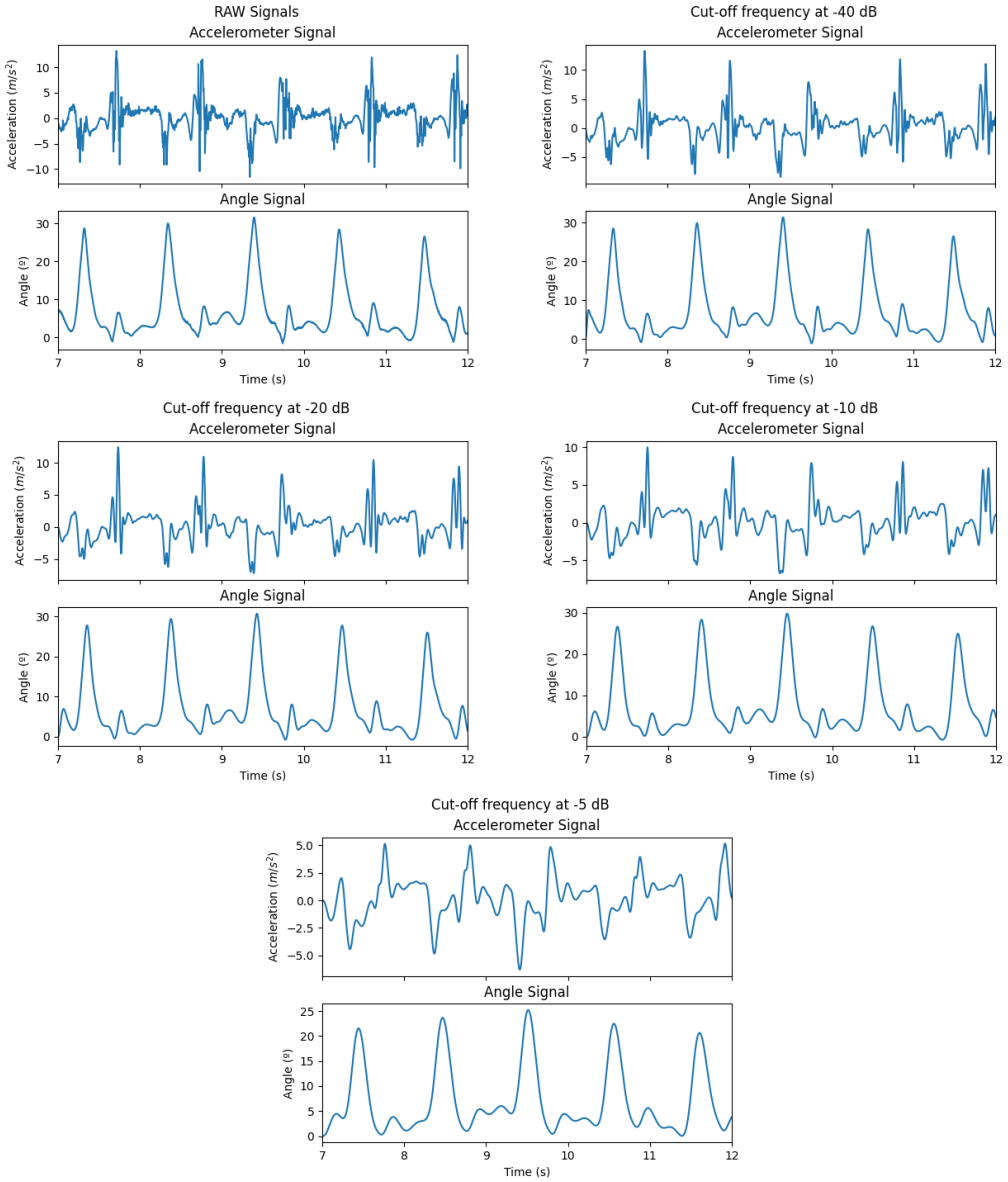


Figure 50: Signals filtered using different cut-off frequencies.

using the raw signal and the -5dB frequencies. The first one contains noise, and in the second one, crucial information has been removed. By contrast, the best configuration is the one filtering the signals with the -20dB frequencies. The configurations using -10dB and -40dB cut-off frequencies perform about 4.5 % worse due to information deletion in the former and noise in the latter.

5.3.3.3 Conclusions on signal filtering

Analysing the results, it can be deduced that the small oscillations in the raw signals were noise, hence the poor performance of using the raw signal. This noise is still present in the -40dB setting, as although the accuracy increases, it is still not the most optimal setting. On the other hand, the settings using the -10dB and -5dB frequencies filter the signals too

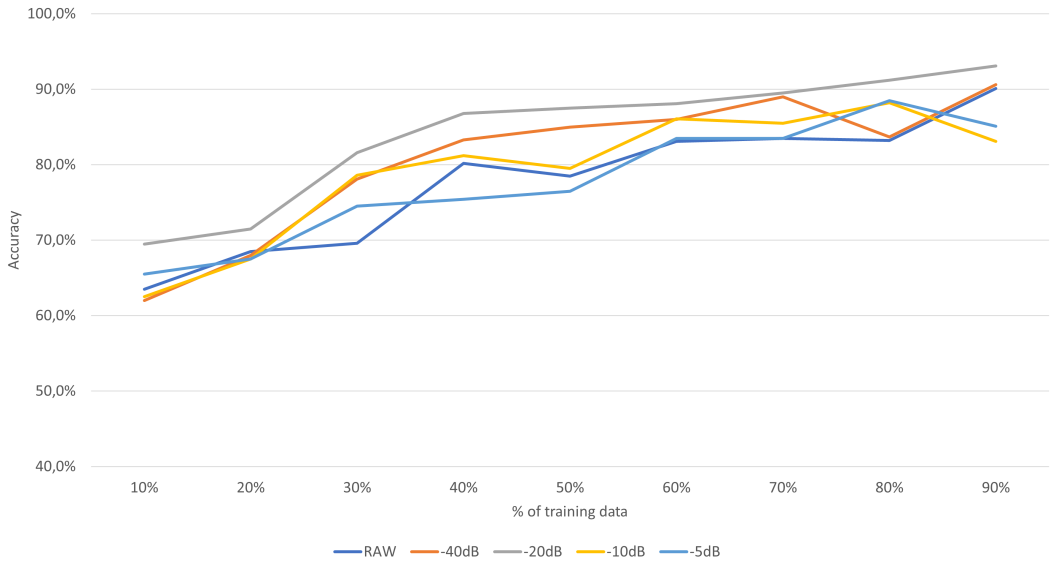


Figure 51: Accuracy results using different cut-off frequencies.

much, so we can say that the small details that have been removed in these settings are distinctive for pathology detection. In this case, the balance between eliminating noise and keeping the essential information is obtained with the -20dB frequencies.

Given the findings, we can conclude that signal filtering can increase the performance of the algorithm, but the cut-off frequencies have to be chosen carefully, as it can remove crucial information for pathology detection.

5.3.4. Improving the scalability

The most common problem in NN-based pattern recognition systems is that the whole system must be retrained when a new user registers. This is the expected behaviour in other fields, such as biometric recognition, as you cannot be identified if you are not inside the system. However, it is unfeasible for medical purposes to have samples of the users before the pathology identification. To solve this problem, the main objective of this section is to create a valid system that uses different users in the training and testing dataset. Thus, from this experiment, all datasets would not share users in the training and testing datasets, and the pathology pattern would be learned by the RNN.

5.3.4.1 Methodology

To achieve the objective, the first step is to create the training and testing datasets. To avoid the system constantly training with the same users, these are chosen randomly in each iteration. In addition, as not all users have the same number of walks, for each training/testing ratio, the algorithm is run 20 times instead of 10 as done before. The method for creating the datasets remains the same as in Figure 45.

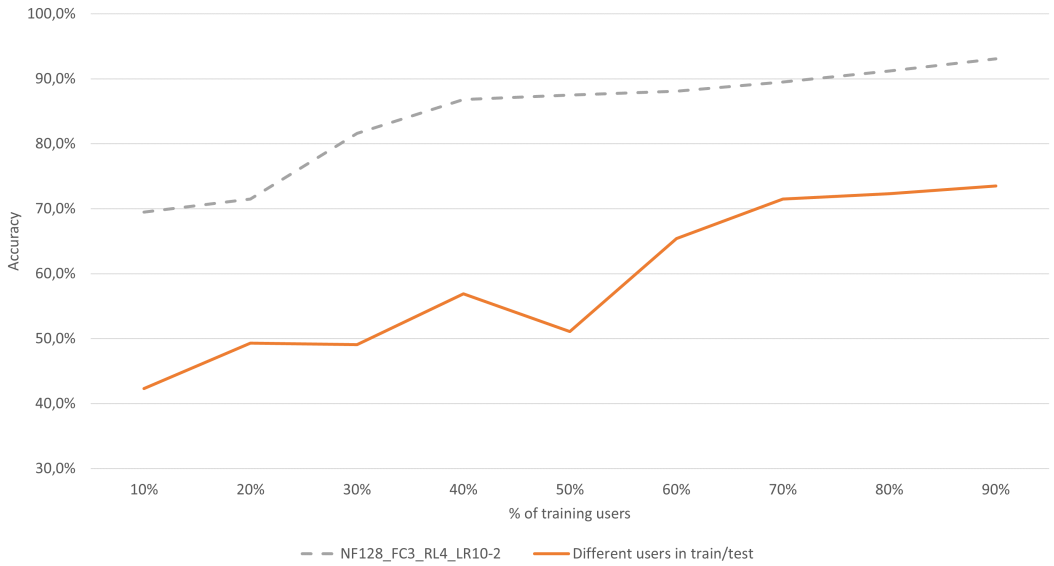


Figure 52: Accuracy results for the RNN with different users in training and testing dataset. In grey the results of the last configuration.

5.3.4.2 Results

In the Figure 52 we can see the accuracy results of the new approach, and it is clear that the accuracy drops sharply. This drop in precision was expected since now the algorithm has to learn the pathology pattern independently of the users, which seems to be more complex. The following experiments aim to improve the results of the algorithm.

Although the results of this approach clearly worsen the performance of the algorithm, this configuration is maintained in future experiments, as this way the scalability of the system increases, i.e., to classify the walks of a new user, it is not necessary to retrain the algorithm with the user data.

5.3.5. Influence of the extra data

This experiment aims to evaluate whether adding more information and GCs improve the accuracy of the algorithm. With the purpose of adding more GCs, the data extraction process is changed: the limitation of removing the first and last three seconds is removed (Equation 15 and Equation 16), furthermore the physiological information is added. As we are adding the physiological information, the matrices M are modified to include the following information about the users: height (Ht), weight (Wt), foot size (Fz), gender (Gd), age(Ag) and sport activity(Sa). The new information is introduced as new columns

in the matrix, so the structure of the new matrix is as follows:

$$V_{pk} = \begin{bmatrix} An_0 & Ac_0 & Gy_0 & Mg_0 & Ht & Wt & Fz & Gd & Ag & Sa \\ An_0 & Ac_1 & Gy_1 & Mg_1 & Ht & Wt & Fz & Gd & Ag & Sa \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ An_L & Ac_L & Gy_L & Mg_L & Ht & Wt & Fz & Gd & Ag & Sa \end{bmatrix} \tag{17}$$

Where:

$$Sa = \begin{cases} N = none \\ M = once\ a\ month \\ W = once\ a\ week \\ T = twice\ a\ week\ or\ more \\ D = daily \end{cases}$$

Figure 53 shows the accuracy results of running the algorithm with the new approaches. The grey dashed line stands for the configuration before any changes were applied and are used as a baseline to compare the new configurations. The blue line shows the results of the system when the extra GCs are added, and as it can be seen, increasing the number of GCs enhance the accuracy of the system by about 4 %, as well as having a flatter response. Furthermore, the yellow line shows the accuracy results when including the physiological information plus the GCs. In this case, the system accuracy rises 19 % compared to the configuration not using the extra information. By employing the physiological data of the users, the algorithm can better identify the stride and thus more easily identify the pathology pattern.

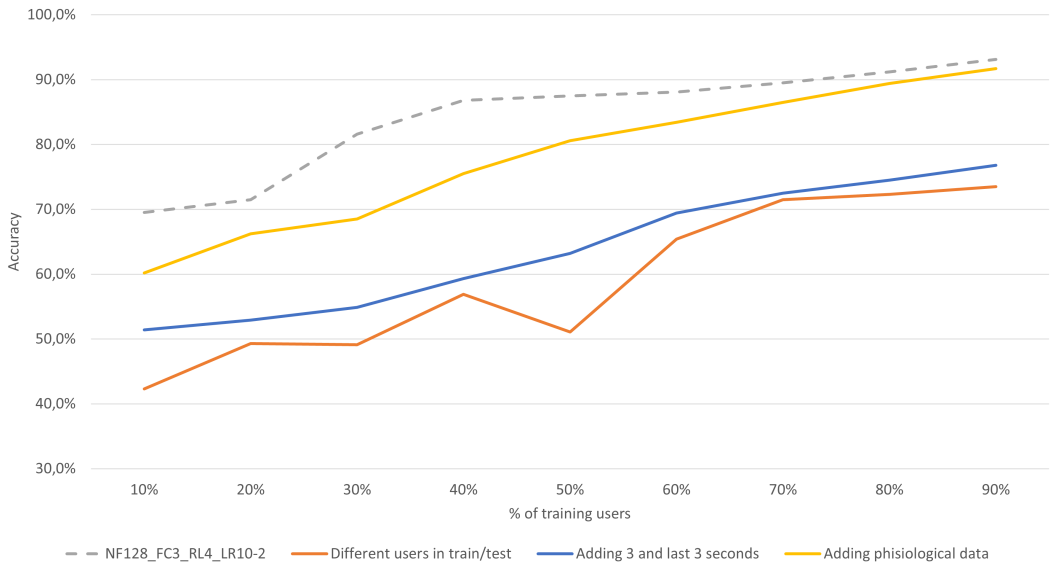


Figure 53: Accuracy results for the configurations with the extra GCs and the physiological information.

Given the findings, by removing the constraint of discarding the information of the first and last three seconds, the amount of data in the database increases, leading to an

increase in the system accuracy. So contrary to what has been said before, this information does not harm the system. On the other hand, by adding the physiological data of the users, the system drastically improves accuracy. Although the last changes have increased the algorithm performance, it is still lower than the baseline one.

5.3.6. Sensor Discrimination

As told above, the performance of the current configuration is still lower than the baseline one, so there is still room for improvement. In the current database, we have information from the accelerometer, gyroscope, magnetometer, and angles in 3D space. This means that we got 81 signals for each walk. From all this information could be that some of the signals are not useful or lead to the misidentification of the pathologies. To see how the different signals behave during a walk, the Figure 54 shows an example of every signal captured by the system.

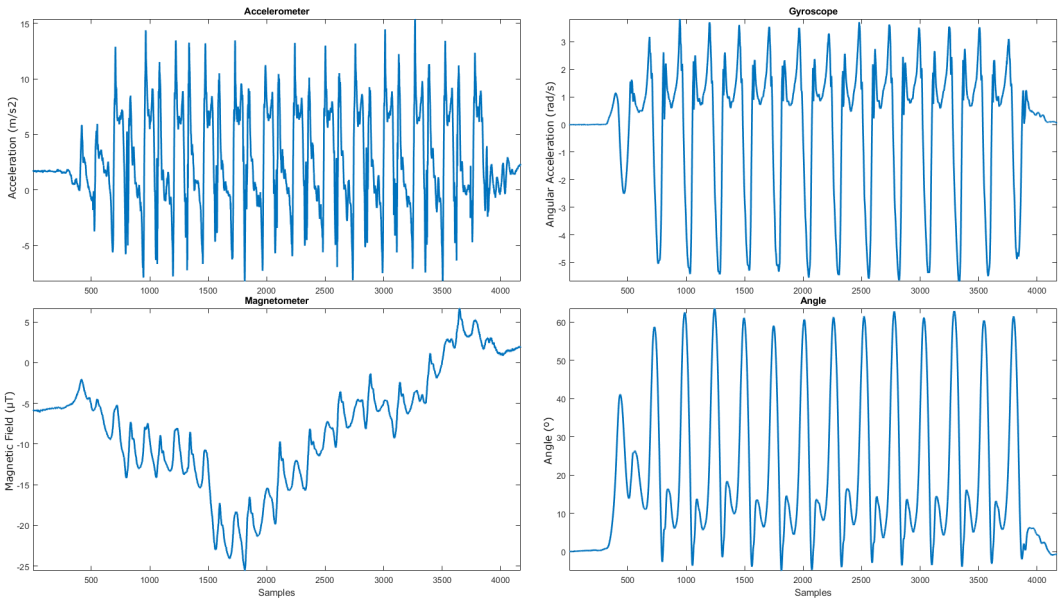


Figure 54: Signals for the different system's sensors. Top-left accelerometer. Top-right gyroscope. Bottom-left magnetometer. Bottom-right angle.

In this section, the validity of different signals combinations of the dataset is evaluated. At this moment, the different axes from the sensors are used individually; however, to study the viability of reducing the number of signals, the module (Equation 4) of the three-axis is performed for each sensor.

For this experiment, all the different combinations of the signals with and without using the module are tested, making a total of 29 different configurations. In order the make the analysis easier, first are presented the results of the configurations using only the 60/40 ratio of training/testing users. The Table 18 shows the accuracy results of the different configurations.

Before starting with the analysis of the results, let us explain the terminology used to

refer to the configurations. The different signals are accelerometer (Ac), gyroscope (Gy), magnetometer (Mg) and angles (An). The "1" after the name indicates that the signal is used, and "0" indicates the opposite. The last parameter specifies whether the magnitude is applied (1) or not (0). We can see that the configurations are ordered in pairs (the one that does involve the magnitude and the one that does not). If we focus on this, we can see that if the angle signals are used, performing the magnitude improves the accuracy by about 2 %, however, if the angle signals are not used, it worsens the accuracy by about 5 %.

Table 18: Accuracy results for the different sensor combination. In red the best six configurations.

Configuration	Accuracy
Ac1_Gy1_Mg1_An1_0	84,7
Ac1_Gy1_Mg1_An1_1	87,5
Ac0_Gy1_Mg1_An1_0	82,4
Ac0_Gy1_Mg1_An1_1	85,2
Ac1_Gy0_Mg1_An1_0	84,5
Ac1_Gy0_Mg1_An1_1	85,1
Ac0_Gy0_Mg1_An1_0	81,1
Ac0_Gy0_Mg1_An1_1	84,6
Ac1_Gy1_Mg0_An1_0	87,0
Ac1_Gy1_Mg0_An1_1	89,6
Ac0_Gy1_Mg0_An1_0	86,6
Ac0_Gy1_Mg0_An1_1	86,7
Ac1_Gy0_Mg0_An1_0	85,3
Ac1_Gy0_Mg0_An1_1	85,4
Ac0_Gy0_Mg0_An1_0	85,9
Ac1_Gy1_Mg1_An0_0	78,9
Ac1_Gy1_Mg1_An0_1	77,2
Ac0_Gy1_Mg1_An0_0	65,7
Ac0_Gy1_Mg1_An0_1	51,3
Ac1_Gy0_Mg1_An0_0	75,2
Ac1_Gy0_Mg1_An0_1	74,5
Ac0_Gy0_Mg1_An0_0	54,5
Ac0_Gy0_Mg1_An0_1	49,3
Ac1_Gy1_Mg0_An0_0	81,1
Ac1_Gy1_Mg0_An0_1	80,1
Ac0_Gy1_Mg0_An0_0	63,5
Ac0_Gy1_Mg0_An0_1	54,3
Ac1_Gy0_Mg0_An0_0	81,5
Ac1_Gy0_Mg0_An0_1	76,5

On the one hand, configurations that use angular signals perform 17 % better than those that do not. In contrast, magnetometer signals make the system perform 5 % worse when used. On the other hand, accelerometer signals, when used, improve results by 11 %. Finally, the gyroscope signals do not seem to affect the performance of the algorithm.

To get a clearer picture of the behaviour of the algorithm with the different configurations, as there are too many configurations to compare them all, only the six best configurations are compared in detail below.

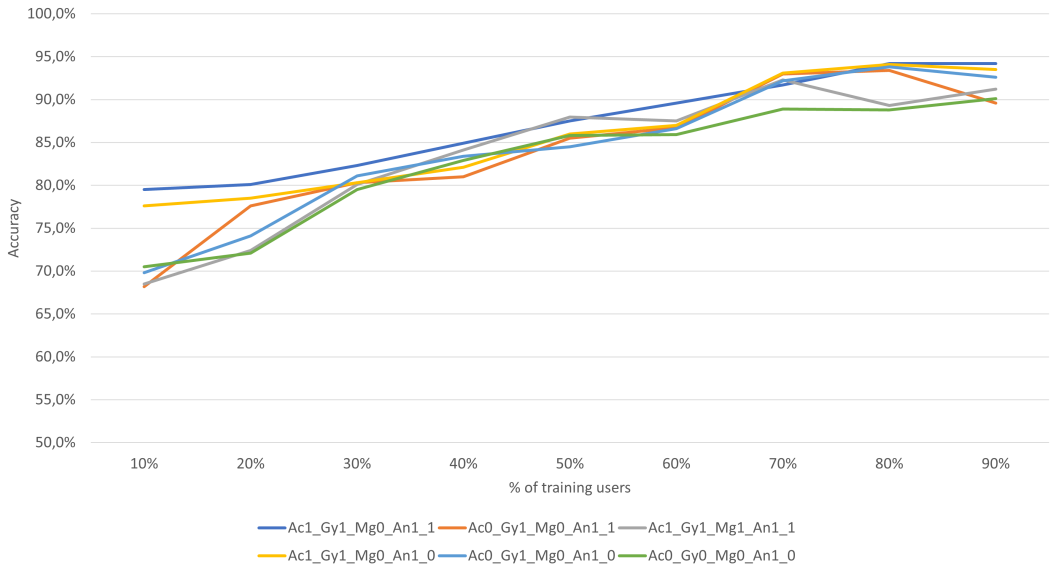


Figure 55: Accuracy results for the 6 best performing configurations in sensor discrimination.

The Figure 55 shows the accuracy results for the six best performing configurations. Analysing those configurations, it can be appreciated that none of them uses the magnetometer signals, and the angle signals are in all the configurations. Concerning the accelerometer and gyroscope, the first one is in 50 % of the configurations, and the other is in 83 % of them. In the light of these results, we can say that the more crucial signals are the angles of the joints followed by the gyroscope signals. The accelerometer signals are less relevant, however, they improve the results slightly. Regarding the signal module, we can see that for the configuration *Ac1_Gy_1_Mg0_An1*, performing the module increases the accuracy, however, for the configuration *Ac0_Gy1_Mg0_An1* this improvement is negligible.

One configuration to note is *Ac0_Gy0_Mg0_An1_0*, which only using the angle signals, is on the top 6 best configurations. *Ac1_Gy1_Mg0_An1_1* and *Ac1_Gy1_Mg0_An1_0* are the best configurations, with the difference that one does the module and the other does not. The *Ac1_Gy1_Mg0_An1_1* configuration has higher accuracy in almost all ratios, so we take it as the best configuration.

Once the best signal combination has been determined, let us compare it with the previous configuration, the Figure 56 shows the comparison of them.

As we can see, the configuration with signal discrimination performs better than

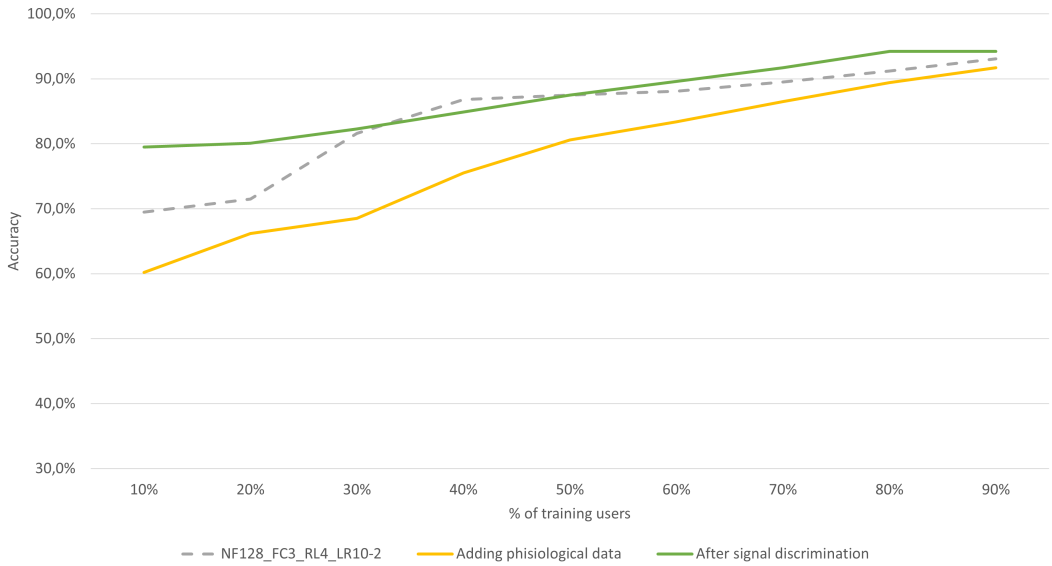


Figure 56: Accuracy results adding the signal discrimination results.

the previous one, mainly with a low percentage of training users. However, the most important fact is that the signal discrimination configuration has outperformed the baseline configuration. At this point, we have reached an algorithm able to classify the 90 % of the cases when trained with 60 % of the users. Although this is a remarkable result, the remaining experiments are conducted to see whether the algorithm accuracy can be increased.

5.3.7. Influence of the origin of cycles

When classifying the walks, the GCs of a walk are individually classified, and the mean value is used as a result of the walk. Doing this, all the cycles have the same importance in the final result, but it could be that some CGs are more significant than others. This idea comes from the experiment in subsection 5.3.1, where we saw that using the parts of the signal closer to the start of the walk gave better results, so this experiment aims to check if there are some CGs that provide better results than others, and if necessary, to create a weighing schema better than the mode.

5.3.7.1 Methodology

To test whether the position of the cycles in the walk influences prediction accuracy, all the cycles of a walk are classified, but instead of calculating the mode, the accuracy values of each GC are stored. After classifying all walks, the average accuracy of all cycles with the same position (i.e., all first, all second, etc.) is calculated. The Figure 57 shows the average accuracy of each cycle as a function of its position.

As it can be seen in Figure 57 there exist some differences in accuracy depending on the position of the cycle. To compensate for the non-linearity of the cycles, a

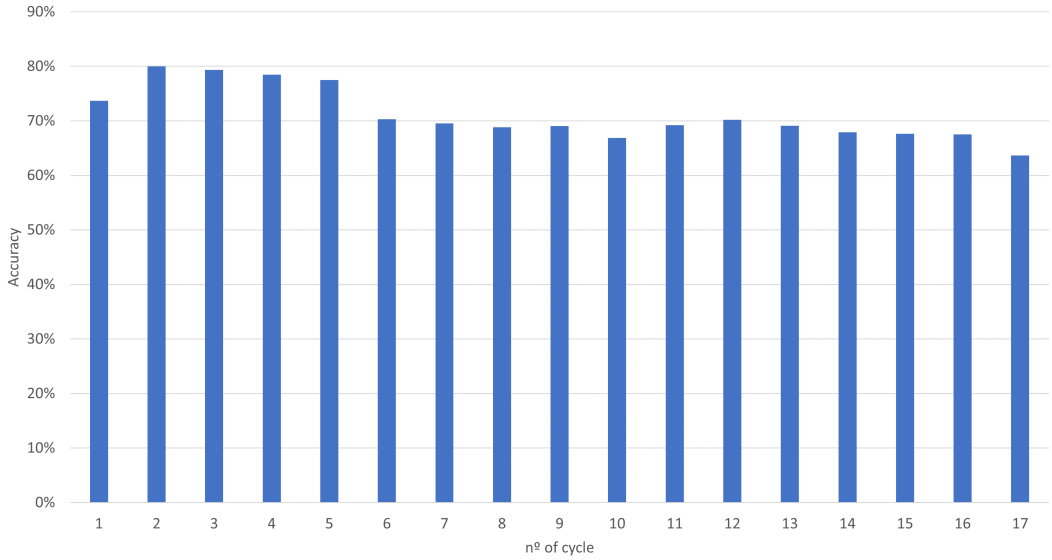


Figure 57: Accuracy results of the GCs depending on their position.

weighting rule is created: First of all, the average accuracy value for all cycles is obtained (ac_{mean}). Then, the average value is used as a reference, the cycles above it are weighted positively, and those below it are weighted negatively. This is done using Equation 18 and Equation 19.

$$ac_{mean} = \frac{\sum_{j=1}^{n} ac_i}{n} \tag{18}$$

$$w_i = 1 + (ac_i + ac_{mean}) \tag{19}$$

Where:
ac_{mean} is the mean accuracy of all the cycles
ac_i is the accuracy of the i^{th} cycle
w_i is the weight of the i^{th} cycle

As we can see in Figure 58, the first five cycles have a positive weight while the rest have a negative weight, so it is clear that the cycles from the begging contain much relevant information. As now weighting curve has to be applied to the results of the cycles, we no longer perform the mode of the cycles to obtain the results of a walk, the new process is as follows.

Given the vectors r_i and w, which contain the predicted classes of the GCs and weights, the class of the walk is obtained by adding the weights with the same predicted class. The highest value indicates the class of the walk. An example of the process is shown in Figure 59. It can be seen that for class 1, the first, fifth, eighth and ninth weights are summed since those GCs has been predicted as 1. This is done for all classes, and lastly, the maximum of the added values is calculated, obtaining that the current walk corresponds to class 0. If no weighting were used in this particular case, the mode would give a tie between classes 0 and 1.

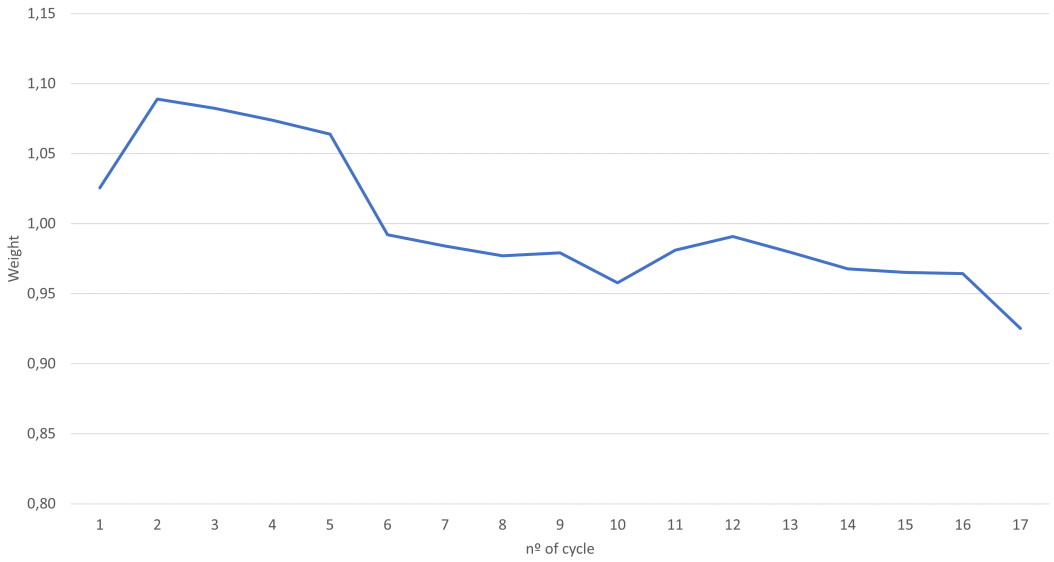


Figure 58: GCs weighting curve.

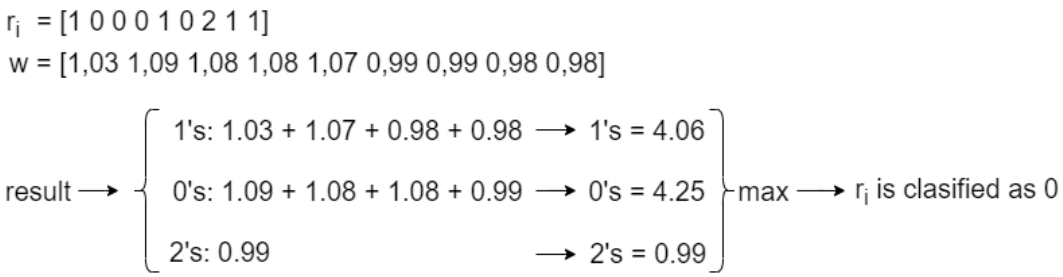


Figure 59: Schema for the new decision method.

5.3.7.2 Results

Once the weighting curve is obtained, it is incorporated into the algorithm, and the algorithm is run again. The algorithm accuracy results are shown in Figure 60. As can be seen, using the new weighted decision method, the accuracy increases by about 2 %. Although not a significant increase, this is because the algorithm is already optimised, and the cases where indeterminacy between two classes occurs are rare. Even so, from the improvement in the accuracy, it can be deduced that they are not null, and by using the weighting decision method, they are entirely removed.

5.3.7.3 Conclusions on cycles origin

As mentioned above, when classifying pathologies, there are specific cycles that perform better than the others. For the detection of pathologies, contrary to what happens in gait recognition, where the best cycles are those at the end [72], the cycles that give the best results are those at the beginning. From this behaviour, it can be concluded that, at the beginning of the walk, the pathology affects the steps the most, and as the walk becomes longer, it influences less and less, contrary to the characteristic parameters of the gait of

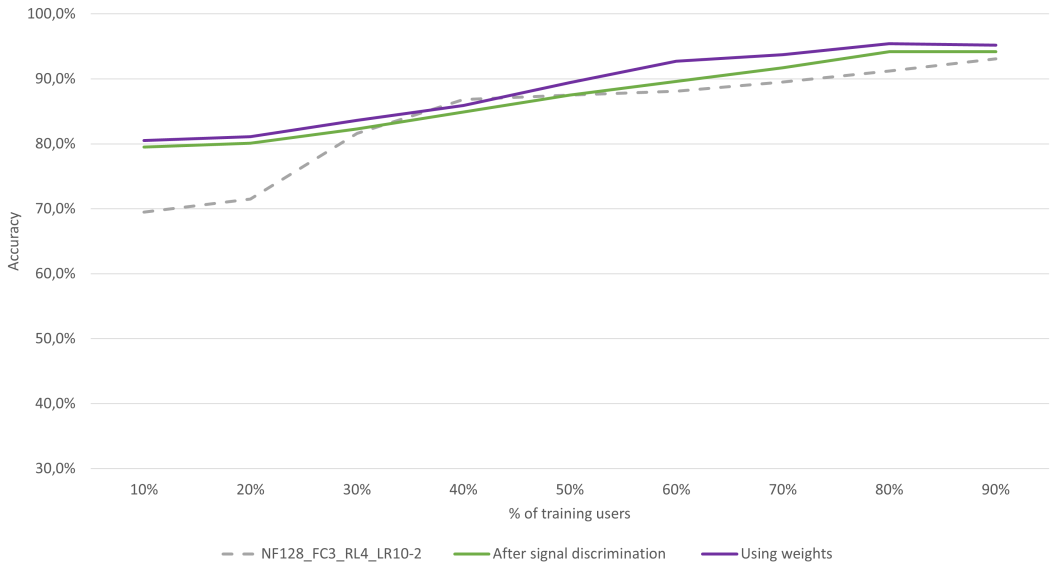


Figure 60: Accuracy results using the weighted decision method.

each user.

5.4. Final Algorithm State

Throughout this chapter, different experiments have been carried out to evaluate the proposed algorithm, and in case one of these experiments improved the algorithm, they were incorporated into the main algorithm. The current state of the algorithm after all changes have been made is given below.

Starting with pre-processing stage, in the first place, the magnetometer signals are discarded, and the module of the three axes of the accelerometer and gyroscope is performed. The angle signals are still being used. So, instead of having 81 signals, we reduce the number of them to 38. On the other hand, the cut-off frequencies have changed, and now they are the frequencies at the point where the spectrum falls under -20 dB.

Following with the data extraction process, currently, GCs are used instead of splitting the signal into four fragments, and the restriction of not using the information from the beginning and end of the signal is removed. Moreover, when creating the training and testing datasets, they are made considering the users instead of the GCs, so that no users are shared between the two datasets.

Finally, the algorithm is still RNN-based, however, the configuration of the hyperparameters has changed. The current configuration for the RNN is $NF128\_HL3\_FC4\_LR10^{-2}$. Furthermore, in an attempt to make the system as realistic as possible, instead of classifying the GCs individually, all the cycles corresponding to a walk are grouped together, and a result is given for the walk, which depends on the result of all the GCs that compose it. In order to obtain the result of a walk, a weighting curve

is used for the different cycles of the walk. A schema of the final algorithm state can be found in Figure 61.

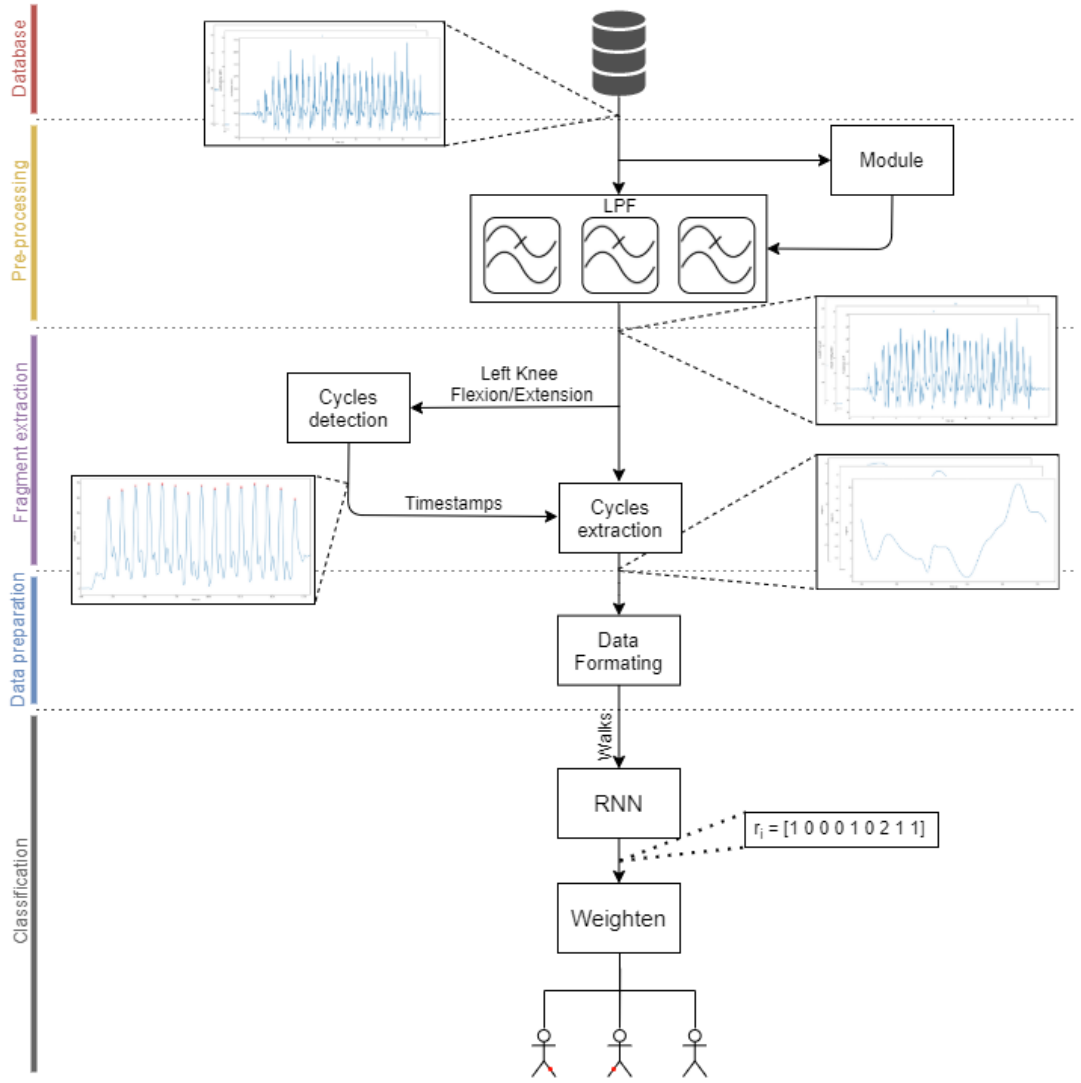


Figure 61: Final algorithm schema.

Once the final algorithm is determined, it is necessary to evaluate whether all the cases are equally classified. To perform this evaluation, the system is trained using all data, however, it is evaluated three times using only data from the one current class (right limp, left limp, or healthy) each time. By doing this, whether the system classifies equally, all the walks can be observed. As it can be appreciated in Figure 62, there are no noticeable differences between the different classes, the maximum difference is around 5 %. With these results, we can assume that the algorithm classifies all the cases equally.

5.5. Conclusions on the Recurrent Neural Network-based Algorithm

This chapter presents an RNN-based algorithm for the classification of lower body pathologies with the restriction of not sharing users between the training and testing

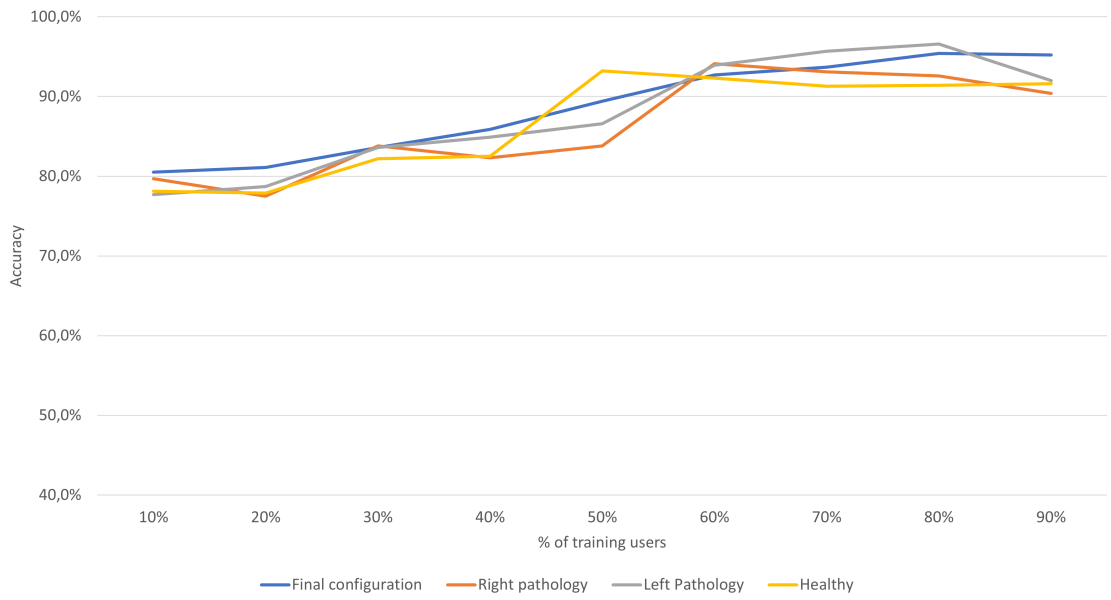


Figure 62: Accuracy results when classifying the cases one at time.

datasets, in this way, there is no need to retrain the system when a new user arrives which gives the algorithm scalability. For this purpose, a database with healthy and fake pathological walks was collected. The system works with the kinematic signals and the joint angles that are processed to extract the GCs. The information of the different sensors corresponding with a GC is sorted in a matrix, and those matrices are used as a basic unit to feed the RNN.

Throughout the chapter, different experiments have been carried out to evaluate the algorithm and try to improve it. At the beginning of the chapter, a rule was presented to eliminate the information related to the first and last seconds of a walk; however, it has been shown that not only does this information not harm the algorithm, but it also improves it. In this case, it can be said that this improvement is due both to the increase in the number of GCs and to the importance of the first cycles of the walk.

On the one hand, it has been proven that the magnetometer signals do not contribute to detecting pathological conditions and that the angles between the joints are the most influential signals. On the other hand, it has been observed that by adding physiological information from the users, the algorithm considerably increases the accuracy, as the RNN can better isolate the pathology pattern.

After all the experimentation, it has been possible to create an algorithm with an accuracy of 93.7 % classifying pathologies, which does not need to be retrained to classify new users.

6. MOTION CAPTURE SYSTEM WITH FEATURE-BASED ALGORITHM

We have evaluated different machine learning algorithms for pathology detection along this thesis, but none have obtained viable results. The lack of feasible results has been blamed on the quality of the signals obtained by smartphones. To verify this, this chapter presents a pathology detection algorithm based on machine learning. As in chapter 4, this algorithm uses features to identify pathologies.

This chapter is divided into five main blocks. The first one shows the initial structure of the algorithm, the process of calculating the features and the data preparations phase. Secondly, the influence of the data preparation is shown. The following section deals with the optimization process, optimising the hyperparameters for the different ML algorithms. The following block applies feature extraction methods to the database to reduce the size of the dataset used. Finally, the conclusions and the final state of the algorithm are presented.

6.1. Database

To create an algorithm capable of distinguishing pathologies, we need a database with healthy and pathological walks. As the objective of this algorithm remains the same as that of the RNN algorithm, the same database is used. By way of reminder, the database is acquired with a portable motion acquisition system based on kinematic sensors, obtaining the acceleration (m/s^2), angular acceleration (rad/s), magnetic field (μT) and the angles of the joints (degree). The database has 51 users walking on a 20 meters flat surface. For more detailed information, see section 5.1.

Although magnetometer signals are present in the database, it has been shown in the previous chapter that they do not provide information for pathology detection. Since this has been proven, magnetometer signals are not used in this algorithm. In addition, the magnitude of the accelerometer and gyroscope axes is performed.

6.2. Proposed Algorithm

As presented in Figure 63, the general structure of the ML algorithm is as follows: a pre-processing phase, in which the signals are low-pass filtered and the module of the accelerometer and gyroscope signal is performed; cycle extraction phase, in which the signal is divided into GCs; feature extraction phase, in which new representative features are obtained by processing the signals; a data preparation phase, in which the features are adapted to the need of the ML algorithm; and lastly, the classification phase, in which the

ML model is trained and the features are classified. As done in chapter 4, different ML algorithms are used to find which best perform with the dataset, however, from what we learnt in that part, we use three of the five algorithms initially used, which are: CART, SVM and kNN.

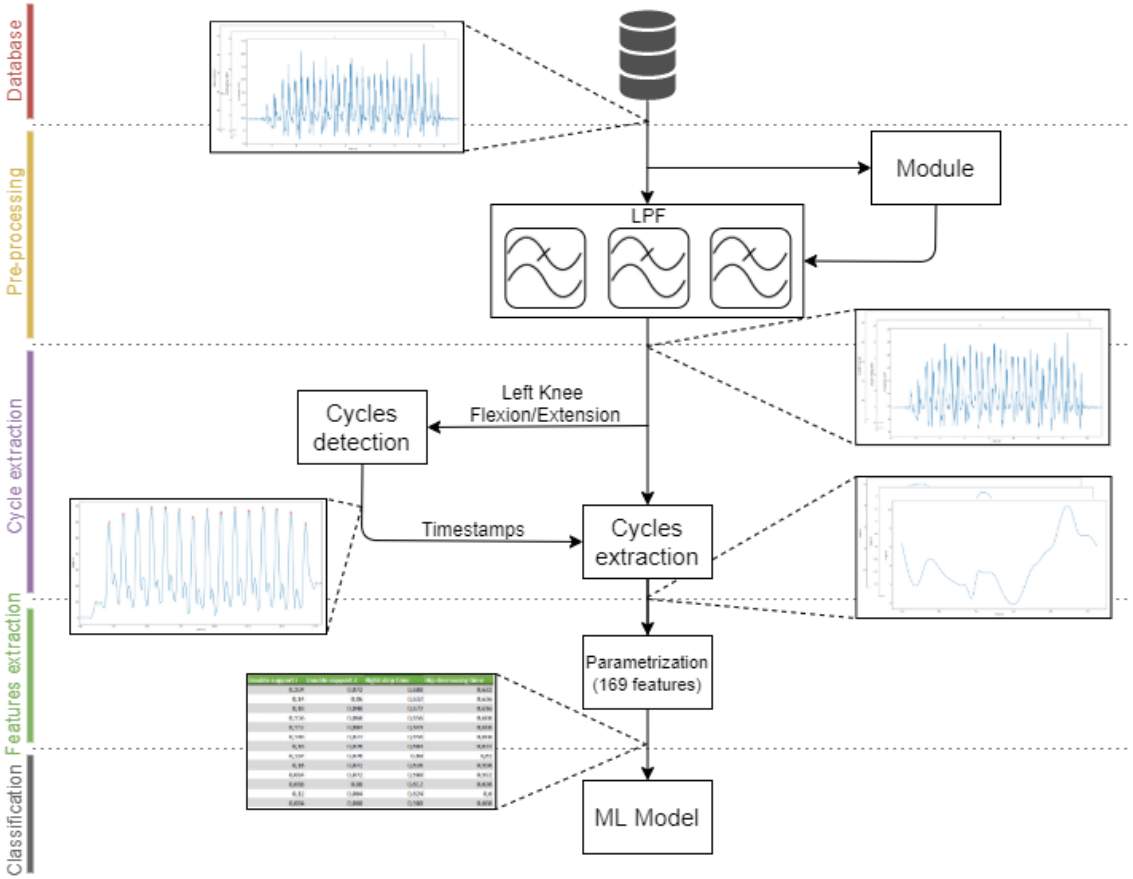


Figure 63: Algorithm general schema.

6.2.1. Pre-processing and Cycle Extraction

The pre-processing and the cycles extraction processes are the same as in the RNN algorithm, although both are briefly explained below.

Walking is a low-frequency movement, thus the signals can be low-pass filtered to remove unnecessary information as well as noise without losing the main information. There are different signals in the database. Instead of filtering all signals with a single cut-off frequency, the signals are filtered depending on the position (foot, leg, or thigh) and the type (accelerometer, gyroscope or angles) of the signal. On the other hand, the cycle extraction process is performed by looking for the maximum positive peaks of the knee flexion/extension signal. Each pair of consecutive peaks determines a cycle. For more details about the pre-processing and cycle extraction process, refer to subsection 5.2.1

and subsection 5.2.2.

As in the previous one, this algorithm uses different users in the training and testing dataset. This way, there is no need to retrain the algorithm in case of a new user, thus increasing the system scalability. An explanation of how the training and testing dataset are created can be found in subsection 5.3.4.

6.2.2. Feature Extraction

As done in the previous chapter, GCs can be used directly to feed the algorithm. However, due to the size of the signals, the computational cost of the algorithm and the memory required to handle the data increases noticeably. One solution to this dilemma is to use parameters to characterize the GCs, which aim to represent the cycles behaviour faithfully.

When calculating the parameters, there are two types: those that are calculated using all signals (such as frequency or zero-crossing rate) and those that are obtained from a specific signal (such as double support times). The parameters used are as follows.

Main Frequency value and peak Although [72] has demonstrated that the main frequency of gait is close to 1 Hz, it corresponds to a healthy gait. Furthermore, the database not only contains information on gait, but also on the different parts of the lower body in different axes, so different frequencies can occur. The frequency is calculated using the Spectral Density Estimation (SDE) with Welch's method [98].

Zero-crossing range A zero-crossing is when the signal changes from negative to a positive value, and it is helpful to identify how oscillating the signal is.

Double Support Time I and II The support time is the time during gait when both feet are on the ground, to calculate them, it is necessary first to obtain the contact and release points. The release point is the moment at which the toe loses contact with the floor, on the other hand, the contact point is the point at which the heel comes into contact with the ground.

To find the release and contact points, the ankle plantar flexion movement is used. The release point corresponds with the global maxima of the signal in which the ankle is totally extended, the contact point corresponds with a very first the local minima before the release point. Once contact and release points are obtained, double support time can be obtained by calculating the time difference between the two points. Figure 64 shows the location of the points in the ankle plantar flexion signal as well as the double support times.

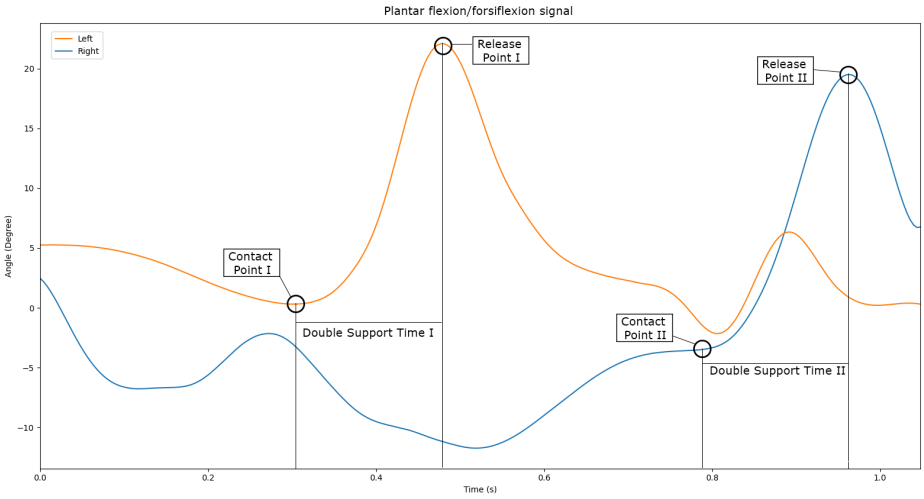


Figure 64: Plantar flexion/dorsiflexion signals.

Left and Right Step Duration Above has been obtained the release and contact points, so they can be used to calculate the length of the step. The right step time corresponds to the time between the release point I and the contact point II, and the left step time is the remaining time (without considering the double contact times) of the signal.

Maximum and Minimum Instant The maximum and minimum instant are when the signal reaches the maximum and minimum values, respectively.

Average Value This parameter measures the value on which the signal is centred.

Right and Left Leg Forward Move Time The left leg forward move time is the time that lasts the left flexion/extension signal to go from the beginning to the first local minimum point. This time corresponds to the time to extend the left leg from the back to the front after the release point. On the other hand, the right leg advance time goes from the global minimum to the end of the signal. This time corresponds to the time it takes for the right leg to advance and step. In Figure 65 both times are shown.

Left Ankle Support Range The left ankle support range measures the set of values that the left ankle takes during the support phase. An example of this can be found in Figure 65.

Right Knee Range The right knee range is the peak-to-peak value of the right knee, that is, all the right knee values taken during the gait. Figure 66 shows an example of it.

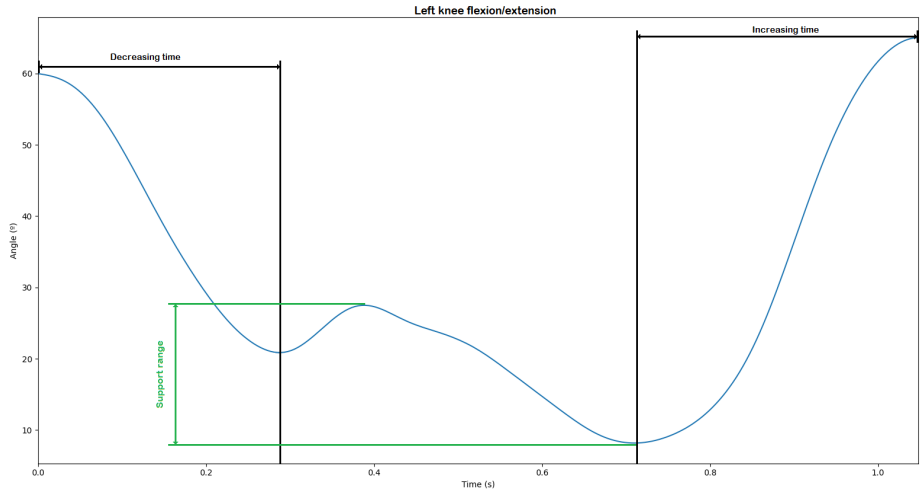


Figure 65: Right and left leg forward move time (black) and left support amplitude (green) in the left knee signal.

Right Knee Peak Width This parameter measures the width of the main peak of the right knee peak at 50 % of the value of the peak. An example of this parameter is shown in Figure 66.

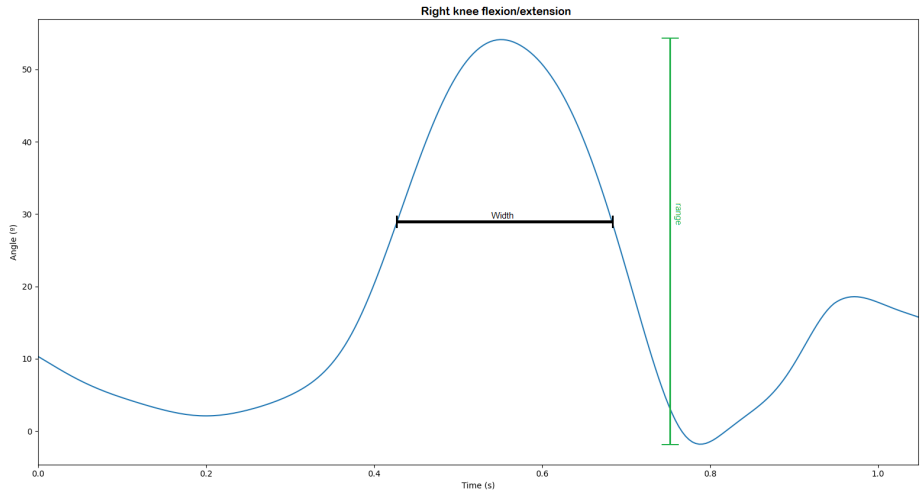


Figure 66: Right knee signal with the measures of the with and range.

In the feature extraction process, 14 features have been used, obtaining a total of 201 features from each GC. An overview of the used parameters and the number of features obtained can be found in Figure 67.

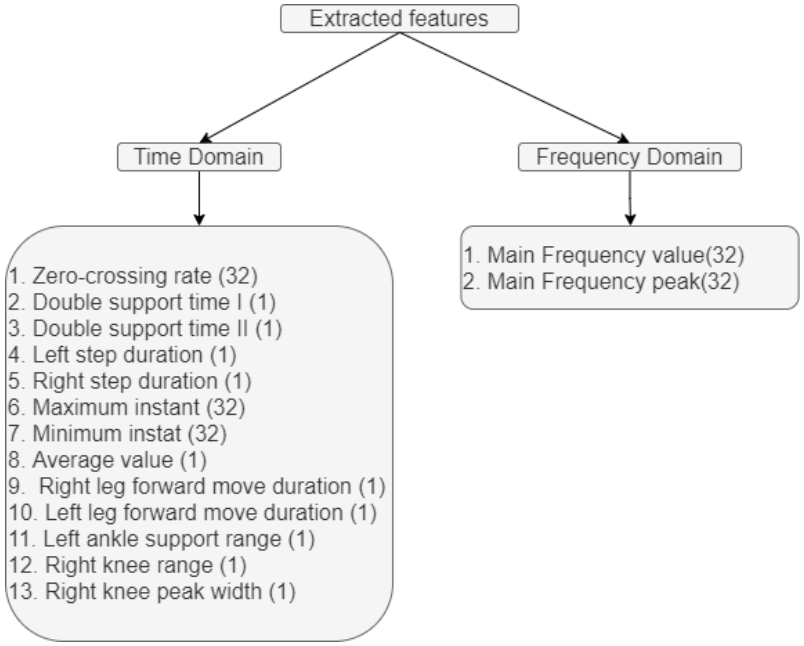


Figure 67: Features obtained. In brackets the number of values obtained from each feature.

6.3. Data preparation

As told above, when using a predictive model, it is essential to prepare the data, as could be some errors in the dataset, such as statistical noise or outliers. In the subsubsection 4.1.1.4 we presented two data preparation approaches: data cleansing and data formatting (refer to subsubsection 4.1.1.4 for a detailed explanation of the processes). In addition to those processes, in this dataset, we also applied feature scaling techniques, which can be defined as:

- **Feature scaling:** Consist of changing the scale of the features to normalize the range of the values.

6.3.1. Data Cleansing

As explained above, the missing and zero values are discarded directly as well as the incoherent data, however, to detect and remove the outliers, we need a statistical tool. In this case, we use the z-score (Equation 12). However, the criteria for choosing the boundary of what is an outlier or not is a matter of discussion. Choosing too low a boundary will result in fair values being eliminated, conversely, a bigger boundary will allow outliers to be considered as genuine values. As choosing a good boundary depends on the data, we can support the choice by visualising the edges in the data. The Figure 68 shows different boundaries using different values of σ.

It can be seen that by using 1 and 2 standard deviations as a threshold, many valid data are discarded as outliers, while by setting the threshold at 4σ some outliers sneak in.

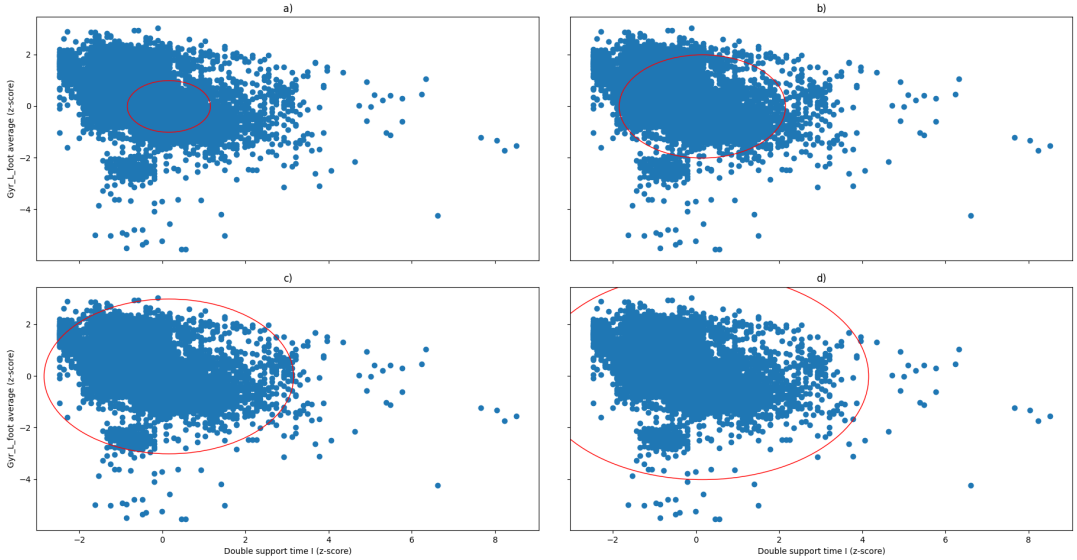


Figure 68: Different σ values to remove the outliers. a) σ=1 b) σ=2 c) σ=3 d) σ=4.

In the case of using 3σ as limit for the values, the main cluster is taken as valid values, although there are specific values attached to it that are discarded. However, according to Figure 68, in a normal distribution 99.7 % of the data is within 3σ, so, as we can consider, the current dataset follows a normal distribution, we set the threshold for the outlier detection at 3σ.

Table 19: GCs removed in the data cleansing process.

Process	**GCs removed**
Zeros	180
Incoherent	28
Outliers	934
Total	1142

The current dataset consists of 25351 GCs, of which 1142 have been removed in the data cleansing process (Table 19). Of these 1142 GC, 934 have been removed because they have been considered as outliers, this corresponds to 3.7 % of the data, which is about ten times more than the stated in theory for a normal distribution. From this, we can note that the data is sparse, however, this does not impede using the database.

6.3.2. Feature Scaling

Feature scaling is a method used to rescale the input variables, as they may have different units (e.g., seconds, km, or mV), which result in different scales. Such differences in scale may affect the ML algorithm, resulting in an unstable or slow learning process, feature scaling eliminates these differences, allowing the model to converge to better weights,

resulting in better accuracies. The two main methods of data scaling are standardization and normalization.

- **Standardization** consists of rescaling the data, so the mean of the data is 0, and the standard deviation is 1. The z-score is a data standardization method.
- • **Normalization**, also known as min-max normalization, consists of rescaling the data into a range of values between 0 and 1.

Scaling the data can improve the results of specific algorithms (e.g., distance-based algorithms, such as kNN, SVM, etc., are the most affected by the scaling) and having no impact on others (e.g., decision tree algorithms).

Figure 69 shows the feature distribution of the original, standardised, and normalised features, and it is still the same in all three cases, only changing the value of the features. In order to see which method fits better with each algorithm, both approaches are compared below.

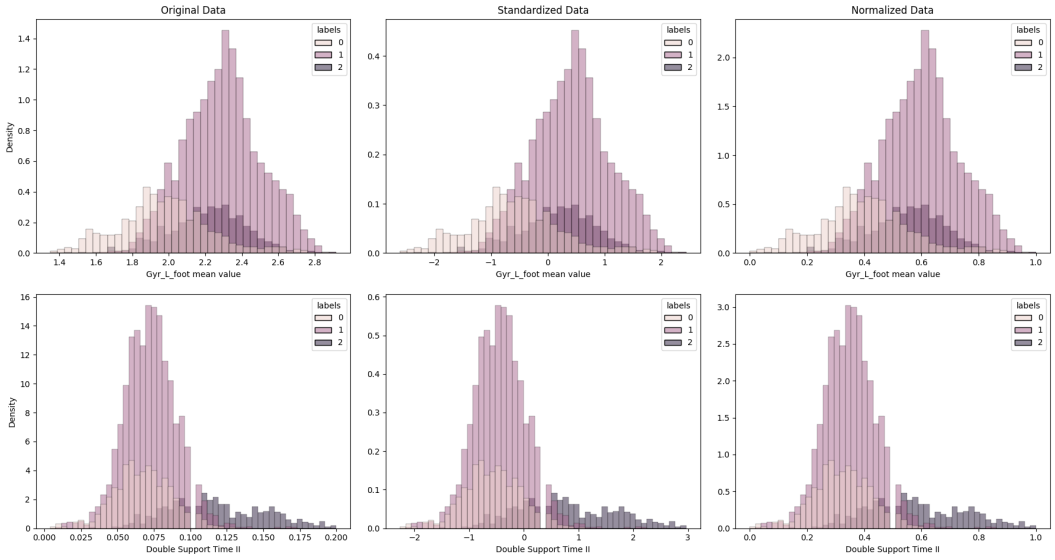


Figure 69: Effects of normalisation and standardisation on the data. Up and down different signals. Right the original dataset. Middle standardized dataset. Left normalized dataset.

6.3.3. Data Formatting

Once the data have been cleansed and formatted, the last step is to sort it in a viable way to feed the algorithm. As above, the data are arranged in a matrix, where the columns represent the different characteristics, and the rows represent the different GCs. An example of the matrix can be seen at Equation 6.

6.4. Experimentation

Above, the structure of the algorithm has been exposed, as well as the data pre-processing, however, although in theory, these changes should improve the performance of the algorithm, the opposite may be the case. Thus, although Figure 63 shows the general structure of the algorithm, some parts have been established by empirical experimentation. Furthermore, the ML algorithms can be optimised to obtain better results. The different experiments carried out to optimise both the ML algorithm and the data used are presented throughout this section. Furthermore, through these experiments, we can determine which of the three ML algorithms used (CART, SVM and kNN) performs better. The experiments performed can be grouped into:

1. Study the influence of the data preparation process, as well as to find out which data scaling performs best.
2. Optimise the hyperparameters of the different ML algorithms.
3. Evaluate the effect of applying dimensionality reduction techniques to the dataset.

In order to find which ML algorithm gives the best results, we need a broader view of how the algorithm works in terms of training/testing ratios. When using any ML algorithm, the ratio of training/testing data is an important decision, the bigger the dataset to train, the more accurate the algorithm can be. However, it can tend to overfit, resulting in the algorithm having learned only to identify the training data. On the contrary, a low training data ratio offers a large amount of data to test the algorithm, however, it may result in the algorithm not learning correctly from the data, i.e., underfitting. To have a more detailed behaviour of the algorithms, different training/testing ratios are used (from 10 % to 90 % in increments of 10 %). As a reminder, when we talk about 10 % of the data, we mean the data obtained using 10 % of the users. In this way, users are not shared between datasets.

On the other hand, for each pair of training/testing users, the algorithm is run twenty times, with random users each time, and the result is the average of those executions. Thus, we can ensure that the algorithm is learning correctly to recognise the patterns, no matter the data.

6.4.1. Influence of Data Preparation

As explained above, data preparation is an important phase in ML algorithms that increase the quality of the dataset, leading to an increase in the algorithm efficiency. To evaluate how the different data preparation processes affect the ML algorithms, the algorithms are executed with data processed differently. Thus, the effect of applying them can be observed. In case that one process improves the results, it will be maintained after

that. Furthermore, as said above, both methods of data scaling (normalization and standardization) are used to see which one offers the best results.

Figure 70: Accuracy results of the ML algorithms with different data preparation methods.

Figure 70 shows the accuracy results of the different ML algorithms when using differently prepared data. It may be seen, the more the training data, the better the results, however, this behaviour is much more noticeable in kNN and SVM than in CART. Working with raw data, up to 50 % of training data CART algorithm performs the best,

but from this point, kNN and SVM significantly increase the accuracy. In terms of data cleansing, we can see that all three algorithms increase the accuracy, with kNN and SVM benefiting the most. As data cleaning has improved accuracy, accuracy is maintained from now on. Finally, scaling the data by normalisation improves the results, while the standardisation worsens them. The exception to this is the CART algorithm, which is not affected by data scaling. A more detailed analysis of the different algorithms is given below.

The CART algorithm has the most stable results, being close to 81 % of accuracy for all configurations of both the training/testing ratio and the data preparation processes. To easily compare the results, the 60/40 % ratio of training/testing users is used. With raw data, the CART algorithm has 81.4 % accuracy, increasing by 2 % after data cleansing. Furthermore, data scaling, neither normalisation nor standardization, seems to improve the accuracy algorithm, although this was expected behaviour since it is not affected by data scaling [99].

Moreover, using the raw data, the SVM algorithm correctly classifies 79 % of the walks and is improved by 5.4 % when data cleansing is applied, reaching a total accuracy of 84.4 %. It can be seen that for this algorithm, data normalization does offer an improvement in accuracy while data standardization does not. It is essential to point out that as the data cleansing has improved the algorithm, the comparison to see if data scaling improves the accuracy is made against the configuration with cleansed data. Looking at the results of the two approaches, it can be discerned that while normalising the data increases the accuracy by an average of 6 %, standardising the data only worsens it.

Finally, the kNN algorithm behaves similarly to the SVM algorithm. With the raw data, it has 83.3 % of accuracy, and it is increased by 4.1 % when using cleansed data. Standardising the data for kNN also worsens the performance, while normalising the data slightly improves the results by 0.5 %.

Based on the results, we can say that normalize the data leads to an improvement of the results, while the standardization causes a worsening of these. In the following, where reference is made to data scaling, it is referred to normalisation. On the other hand, we see that depending on the algorithm, the response to the different data preparation processes is different. All the algorithms improve the accuracy when the data is cleansed, however only the SVM and kNN algorithms improve when the data is normalized. So, from now on, data cleansing and scaling are used in SVM and kNN algorithms and only data cleansing in CART.

Another way to see how the algorithm changes depending on the data are the decisions boundaries. The decisions boundaries are hypersurfaces that separate the space into the decision surfaces for the different classes. To plot the decision boundaries, only two features are needed, in this case, the support time I (x-axis) and II (y-axis) are used. Figure 71 shows the changes in decision boundaries between the use of raw (left) and prepared (right) data.

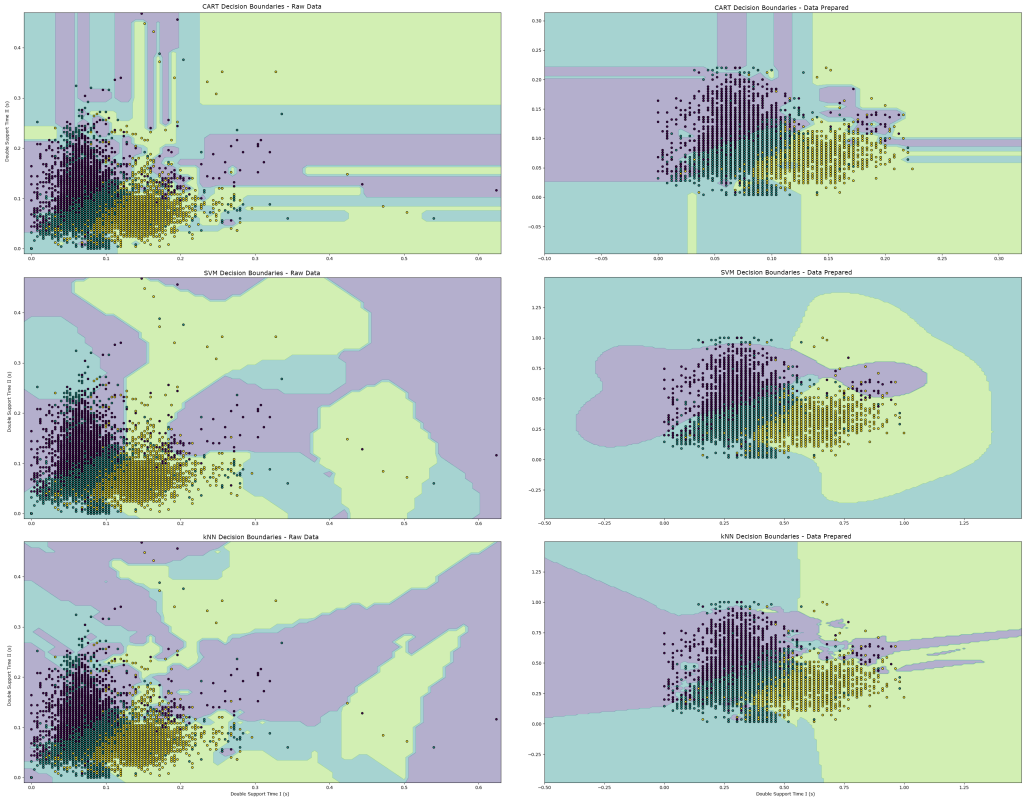


Figure 71: Comparison of the algorithms decision borders with raw (left) and prepared data (right).

As it can be seen in Figure 71, when using raw data, the algorithms create various decisions surfaces for each class. It is due to the algorithms tend to classify every point, creating very intricate surfaces, or in other words, producing overfitting. Otherwise, when using prepared data, the algorithm tends to generalise, creating larger surfaces. In the case of kNN and CART, there are small regions, but the SVM algorithm shows clear surfaces. This can be solved by optimising the hyperparameters of the ML algorithms.

Time is also an essential factor to take into account, in the case of the ML algorithms, it can be divided into two: training and testing time. Table 20 shows the times for the ML algorithms before and after the data preparation. These values are obtained using 60 % of the users to train the algorithms and the remaining 40 % to test.

Table 20: Training and testing times with and without data preparation.

Algorithm	Raw data		Prepared Data	
	Training (ms)	Testing (ms)	Training (ms)	Testing (ms)
CART	678.39	4.03	368.87	3.51
SVM	2713.26	3511.56	1413.5	2309.93
kNN	3.50	1276.31	2.76	652.50

It may be seen that the data preparation reduces times significantly. For the CART and kNN, the times are reduced by around 50 %. However, in the case of the SVM algorithm, the time reduction is smaller. The fact that the CART testing time and the kNN training time are so low is because of the properties of the algorithms, explained in section 3.1.

After optimising the dataset for every ML algorithm, we can compare them to see which one performs better. So far, we have looked at accuracy and decision borders to evaluate the performance of the algorithm. However, accuracy gives us an overall idea of how the system works and the decisions borders give us an idea of how well the algorithm works with the dataset. These metrics provide an idea about the algorithm performance and any clue about the interclass classification so that the algorithm could be unbalanced.

To get more details on the algorithm performance, we can use the confusion matrices, through which we can obtain metrics such as the false positive rate or the false negative. In this case, the walks are classified as either healthy or pathological (right or left). As discussed in section 4.1, the metrics used differ slightly as this is a medical problem. Specifically, the following metrics are used, for more details on these metrics, see subsection 4.1.3.

1. **False Positive Rate (FPR)**: is the rate of healthy walks wrongly identified as pathological.
2. **False Negative Rate (FNR)**: is the rate of pathological walks wrongly classified as healthy
3. **False Limp Rate (FLR)**: is the ratio of pathological walks that are misclassified as other pathology.

By using the confusion matrices shown in Figure 72, we can calculate the metrics explained above, which are presented in Table 21.

Table 21: Comparison of algorithms metrics.

Algorithms	**FNR (%)**	**FPR (%)**	**FLR (%)**	**Accuracy (%)**
CART	9.55	5.76	1.21	83.48
SVM	2.17	6.57	0.09	91.17
kNN	12.07	0.13	0.22	87.58

Analysing the results, we can see that the best algorithm is the SVM, as it has an accuracy of 91.17 % and the lowest FNR and FLR values. On the contrary, it is the algorithm with the worst FPR value, however, for the current problem, the FPR is the least important metric.

On the other hand, as far as accuracy is concerned, the CART algorithm is the worst, as well as having poor ratios. Finally, we can see that the kNN algorithm has an accuracy

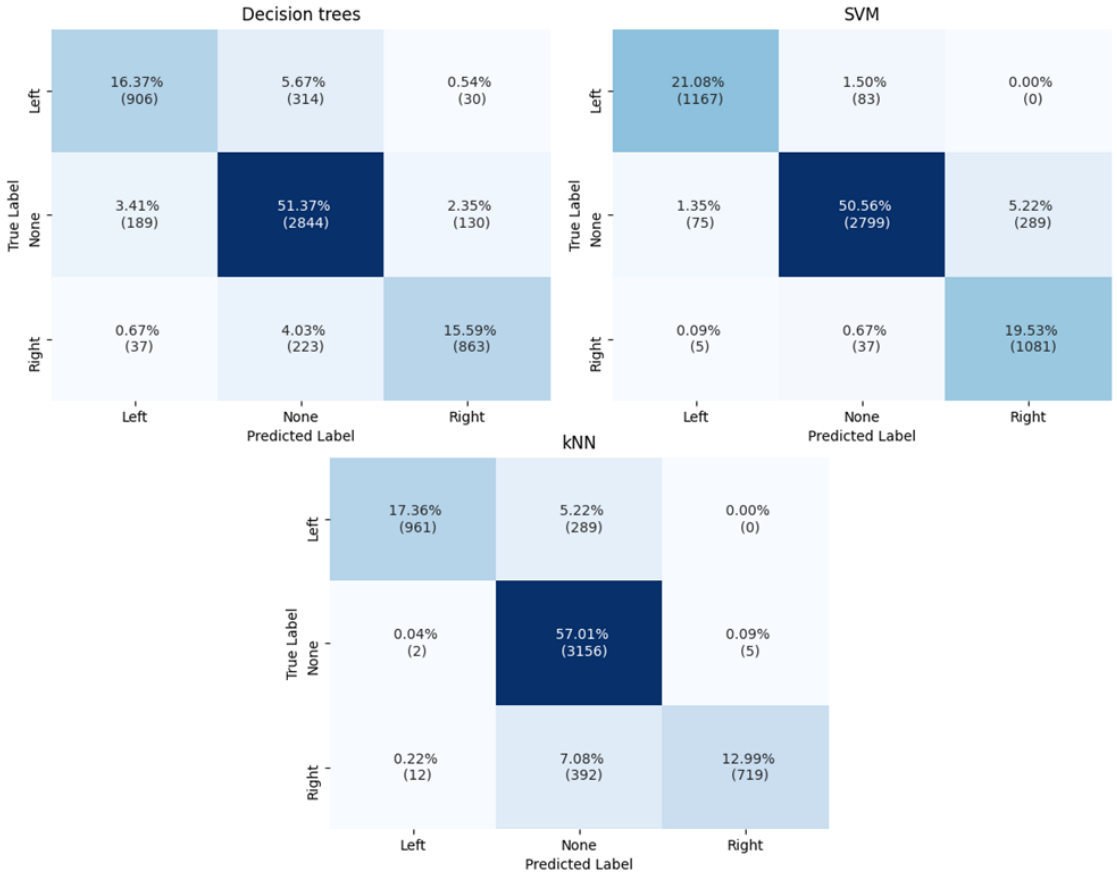


Figure 72: Confusion matrices of the ML algorithms using prepared data.

of 87.58 %, however, it has an FNR of 12.07 %, being the worst of the three algorithms. On the other hand, it has an FPR of 0.13, being the most robust algorithm against false positives.

At this point, let us summarise the current condition of the different algorithms.

- The CART algorithm has 83.48 % accuracy in classifying the walks, which is the worst of the three algorithms, however, once the algorithm has been trained, it is the fastest one. Focusing on decision edges, we can see that there are still few isolated zones for different classes, which means the algorithm needs to be optimised. Focusing on the metrics, we see that it has an FPR of 5.76 % and an FNR of 9.55 %.

- The SVM algorithm has an accuracy of 91.17 %, which makes it the best one. The decision borders are clear and uniform, they seem to have learned how to generalize from training data. Additionally, they present an FNR of 2.17 %. It also shows an FLR of 0.09 %, which is the lower of the three algorithms. In terms of time, this algorithm is the slowest one both in training and testing.

- kNN algorithms correctly classify 87.58 % of the walks. The decision surfaces created by the algorithm behaves similarly way to the CART's surfaces. The metrics

show an FPR of 0.13 % (the lowest) and an FPR of 12.3 % (the highest). Because kNN is a lazy algorithm, it is the fastest in the training phase.

In light of the results, it can be concluded that SVM is the best algorithm at this point. However, the results could be improved by optimising the algorithms. The following section focuses on algorithms optimizations.

6.4.2. Hyperparameter Optimization

In the last section, we have demonstrated that data preparation can improve the accuracy of the algorithms, however, there is still room for the improvement of the algorithms. It is possible to enhance the algorithm by tuning the hyperparameters. As done before in the optimization of the RNN, GS optimisation is used to find the optimal configuration. It is important to note that the optimization is performed by training the algorithms with 60 % of the users and testing with the other 40 %. Although we lose the full view of how the algorithms perform, it makes the optimization much faster. As done above, the algorithm is run twenty times with randomized data each time for each value combination. Such we have three algorithms the optimization process is explained separately.

6.4.2.1 CART Optimization

Below are explained the hyperparameters used for the CART optimization.

- **Criterion**: Function to measure the quality of the splits. The possible options are the Gini Impurity function (gini), which uses the divergences between the probability distributions, and the Information Gain function (IG), which uses the entropy.
- **Splitter**: The approach used to decide the split at each node. The possible options are best (B), which evaluate all the possible splits to choose the best and random (R), which splits the nodes using a random uniform function.
- **Max depth**: Parameter to set the maximum depth of the tree.
- **Min samples split**: This parameter establishes the minimum number of samples needed to split a node.
- **Min samples leaf**: Parameter to set the minimum number of samples of a leaf node.

After every hyperparameter has been defined, they are analysed to establish the borders for the optimization grid. In the case of criterion, we must consider the work published by Laura Elena et al. [100] in which they demonstrate that only in 2 % of the cases there exists a discrepancy between using Gini impurity and information gain. So, to reduce the complexity of the optimization process, the information gain function is used.

In the case of the splitter, there is no clue about which is better, so both values are used in the optimization process.

Choosing a max depth for the algorithm is a complex task. The deeper a tree is, the more complicated it becomes. However, a too deep tree can lead to overfitting, whereas a too shallow tree can lead to underfitting. A common approach to have an idea about max depth value is to run the algorithm without any restriction and check what value the model uses. The non-optimized model has a max depth value of 16, so we build the grid around this value.

The minimum number of samples in the splits and the leaf can be used to control the overfitting. A small value produces a very thorny tree, which would be strictly specific for the training data. Conversely, a high value would produce underfitting, not granting the model to create enough splits. According to [101] the best min samples split value is between 1-40 and between 1-20 for min samples leaf. The limits for the first round of random grid are shown in Table 22.

Table 22: Hyperparameters ranges for the first execution of GS optimization for CART.

Hyperparameters	**Values**
Splitter	[‘best’, ’random’]
Max depth	[8, 24]
Min samples split	[5, 40]
Min samples leaf	[5,20]

After establishing the limits for the hyperparameters, the algorithm is tested with 50 different random combinations that correspond with 15 % of the total combinations. The results of those executions are shown in Table 23.

Table 23: Accuracy results from first execution of GS optimization for CART (%). DP: max depth, mSL: min samples leaf, mSS: min samples split, SP: Splitter. B: best, R: random. In red the area with the best results.

DP	SP mSL/ mSS	B 5	B 10	B 15	B 20	B 25	B 30	B 35	B 40	R 5	R 10	R 15	R 20	R 25	R 30	R 35	R 40
8	**5**	86,2		87,7													83,4
8	**10**							85,1			86,7						
8	**15**				86,15										84,1	85,3	
8	**20**		87,71				86,2						85,5				
12	**5**									85,8							
12	**10**			86,52										84,7			86,1
12	**15**	85,7					86,2					85,3					
12	**20**		86,22		87,15						84,1				86,1		
16	**5**								84,0								
16	**10**											86,4		85,9			
16	**15**						84,6			83,8						83,1	
16	**20**			86,6													83,9
20	**5**							86,1					85,9				
20	**10**		86,4			85,2											
20	**15**														84,3		
20	**20**						84,3		86,1		85,0		83,5				
24	**5**		83,4		85,9												
24	**10**									86,6						82,2	85,9
24	**15**		85,0										85,0				
24	**20**	86,1					85,7							86,3			84,4

By analysing the results, it may be seen that all the configurations have pretty similar results. Overall, the configurations using the "best" splitter and with small depth performs better. In the case of the mSL, the algorithm works better within the range [10-20]. Finally, the results do not give a clear pattern about which value is better for the mSS.

To continue the search for the best configuration, the limits of the hyperparameters are reduced. It can be observed that the configurations with max depth between 8 and 12 perform better, so the max depth is constrained to this range. Moreover, the mSL range is also reduced to 10-20. As told above, the splitter best has better results, only is used in the next round of GS optimization. Finally, the mSS range is still unchanged. The new limitations are shown in Table 24.

Table 24: Hyperparameters ranges for the second execution of GS optimization for CART.

Hyperparameters	Values
Splitter	['best']
Max depth	[8,12]
Min samples split	[10,15,20]
Min samples leaf	[5,10,15,20]

After setting the new hyperparameters limits, the optimization algorithm is rerun. However, because the casuistry has been reduced, the algorithm is run 35 % of the possible

combinations, which is 20 times. The result of those executions is shown in Table 25.

Table 25: Accuracy results of second execution of GS optimization for CART (%). DP: max depth. mSL: min samples leaf. mSS: min samples split. SP: Splitter. In red the area with the best results.

DP	SP mSL/ mSS	B 10	B 15	B 20
8	**5**		87,7	
8	**10**	86,15		
8	**15**			86,15
8	**20**	87,71		
9	**5**			87,4
9	**10**		87,64	
9	**15**			
9	**20**			88,18
10	**5**	86,23		
10	**10**		86,2	89,62
10	**15**	87,90		
10	**20**	87,69		88,44
11	**5**	87,39		
11	**10**			
11	**15**		87,85	87,18
11	**20**			
12	**5**		87,85	
12	**10**	86,4	86,52	
12	**15**			87,78
12	**20**	86,22		87,15

It can be seen that all the configurations have a similar result, however, the best performing configurations are those with mSS = 20 and DP = 10. However, the mSS parameter does not seem to converge to an optimal parameter. The configurations inside the red rectangle are evaluated in-depth to obtain a clearer understanding of their behaviour.

6.4.2.1.1 Comparison of the best performing CART configurations

Throughout the optimization, only one training and testing ratio has been used, however, as the number of possible configurations has been significantly reduced, and to compare the best performing configurations, all ratios are used. The accuracy results of all the configurations are shown is Figure 73. To begin with, it is necessary to explain the nomenclature of the configurations used, which corresponds to the following:

- **DP**9 refers to the maximum depth of the tree, in this case, 9.
- **mSL**10 refers to the minimum samples to the split a node, in this case, 10.
- **mSS2**0 refers to the minimum samples of a leaf node, in this case, 20.
- **SP**B refers to the strategy used to split the nodes, in this case, the "best" strategy.

Once the nomenclature of the configurations has been given, we can focus on analysing them. It is seen that all the configurations have similar results, with a difference between the best and the worst of only 2 %. So, to identify the best performing configuration, these are compared using the metrics, which can be seen in Table 26.

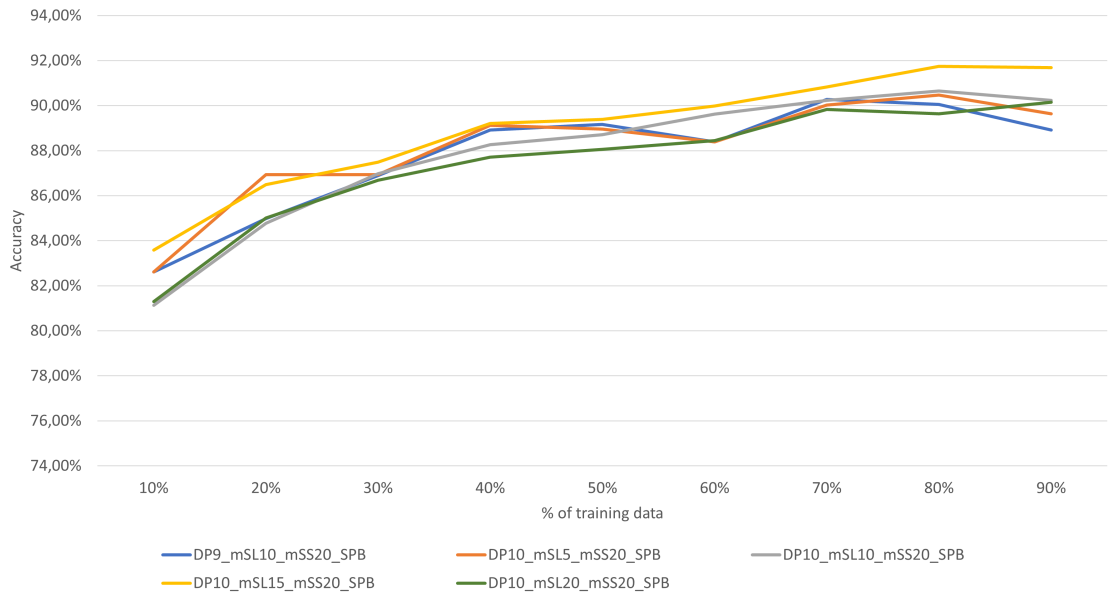


Figure 73: Accuracy results for the best CART configurations.

Table 26: Metrics of the best performing CART configurations.

Configurations	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)	Training Time (ms)	Predict Time (ms)
DP9_mSL10_mSS20_SPB	5,79	4,90	1,12	88,18	292,65	4,23
DP10_mSL5_mSS20_SPB	5,82	5,09	0,69	88,39	290,39	3,99
DP10_mSL10_mSS20_SPB	**4,70**	**5,07**	**0,60**	**89,62**	**292,54**	**3,82**
DP10_mSL15_mSS20_SPB	5,34	4,04	0,63	89,99	295,00	4,14
DP10_mSL20_mSS20_SPB	5,19	5,50	0,87	88,44	292,51	4,04

By analysing the metrics, we see that as with the accuracy results, the metrics have very similar values, however, *DP10_mSL10_mSS20_SPB* is slightly better than the others. Although this configuration does not have the highest accuracy, it does have the best FNR as well as FLR. In terms of training and prediction times, the differences between the configurations are negligible. Due to the above, we can conclude that the best performing CART configuration is *DP10_mSL10_mSS20_SPB*.

6.4.2.2 SVM Optimization

As before, we first present the definition of the hyperparameters for the SVM algorithm.

- **C**: is the regularization parameter, and it controls the magnitude of the penalty for each misclassified point.SVM tries to find a hyperplane that maximises the correctly classified points and the support vectors. The trade-off between these two parameters is controlled by C. A small C value results in a low penalty for the misclassified point, so the algorithm tends to choose hyperplanes with large support vectors. Whereas if C is large, the algorithm tries to reduce the number of misclassified points due to the high penalty.
- **Kernel**: Parameter to select the kernel to be used by the algorithm. Possible options are *linear*, which uses a linear function; *poly* which uses a polynomial function; *rbf* which uses a Gaussian function and *sigmoid*, which uses a sigmoid function.
- **Degree**: Parameter to define the degree of the polynomial kernel. The other kernels ignore it.
- **Gamma**: Kernel coefficient for *rbf*, *poly* and *sigmoid*. This parameter controls both the curvature of the decision borders and the distance of influence of isolated points to the decision borders. With high gamma values, the curvatures of the borders increase, and the isolated points have less influence on the borders. In contrast, a low value of gamma produces straighter borders and the influence of isolated points increases. Therefore, models with low gamma tend to underfit, and high values of gamma can produce overfitting.

All the kernels are used in GS optimization. However, the degree parameter is only needed when using the polynomial kernel. As a degree of 1 produces a linear function and a degree greater than 5 implies an overly complex function with a tendency to overfitting, the range for the degree hyperparameter is set between 2 and 5. The C and gamma values are established considering the recommendation by [102]. For the linear kernel, as the linear kernel does not produce curvatures, the gamma parameter has no effect, so only C is used. Table 27 shows the hyperparameter range for the GS optimization.

Table 27: Hyperparameters ranges for the first execution of GS optimization for SVM.

Hyperparameters	Values
kernel	[linear, poly, rbf, sigmoid]
C	$10^{[-1,2]}$
gamma	$10^{[-4,1]}$
degree	[2, 5]

After establishing the limits for the hyperparameters, the algorithm is tested with 22 different random combinations corresponding with 15 % of the total combinations. The results of those executions are shown in Table 28.

Table 28: Accuracy results of the first execution of GS optimization for SVM (%).

C	kernel Degree/ gamma	rbf -	sigmoid -	linear -	poly 2	poly 3	poly 4	poly 5
0.1	$\mathbf{10^{1}}$							
0.1	$\mathbf{10^{0}}$			-				
0.1	$\mathbf{10^{-1}}$	89,44		-				41,86
0.1	$\mathbf{10^{-2}}$			-		84,51		
0.1	$\mathbf{10^{-3}}$		50,86	-				
0.1	$\mathbf{10^{-4}}$			-	85,23			
1	$\mathbf{10^{1}}$							
1	$\mathbf{10^{0}}$			-			41,86	
1	$\mathbf{10^{-1}}$			-				
1	$\mathbf{10^{-2}}$	71,32		-				
1	$\mathbf{10^{-3}}$			-		83,68		
1	$\mathbf{10^{-4}}$			-				41,86
10	$\mathbf{10^{1}}$							
10	$\mathbf{10^{0}}$		51,86	-		85,27		63,92
10	$\mathbf{10^{-1}}$	95,58		-	89,56			
10	$\mathbf{10^{-2}}$			-				
10	$\mathbf{10^{-3}}$			-				41,86
10	$\mathbf{10^{-4}}$		51,86	-				
100	$\mathbf{10^{1}}$			81,46	91,03			
100	$\mathbf{10^{0}}$	89,02	51,86	-			84,23	
100	$\mathbf{10^{-1}}$			-				
100	$\mathbf{10^{-2}}$			-				
100	$\mathbf{10^{-3}}$			-				
100	$\mathbf{10^{-4}}$			-		41,86		

In the light of the outcomes, it can be seen that the gamma values with the best results are between 10^{1} and 10^{-1}. On the other hand, the different values of C do not show significant changes in the accuracy, although it tends to improve the result for the range between 10 and 100. Finally, the sigmoid kernel has the worst results, followed by the poly kernel with a degree equal to and bigger than 4. The linear kernel is not noted for its accuracy, although it has only one value. The rbf and poly ($degree \geq 3$) have the best results, so they are used for the second round of GS. After analysing the results, the borders are recalculated. Table 29 shows the new values for the second execution of the GS optimization.

Once the new borders have been established, the algorithm is rerun, as done before, the percentage of runs is increased up to 30 %. The results of the new executions are in Table 30.

Table 29: Hyperparameters ranges for the second execution of RS optimization for SVM

Hyperparameters	Values
kernel	[poly, rbf]
C	[10,20,40,60,80,100]
gamma	[0.1,0.2,0.4,0.8,1]
degree	[2,3]

Table 30: Accuracy results of the second execution of GS optimization for SVM (%). In red are highlighted the best performing configurations

C	kernel Degree/ gamma	rbf -	poly 2	poly 3
10	**0.1**			94,09
10	**0.2**	93,98	93,84	
10	**0.4**			
10	**0.6**			
10	**0.8**		93,51	
10	**1**			
20	**0.1**		92,44	92,68
20	**0.2**	95,71		94,16
20	**0.4**	94,99		
20	**0.6**	93,86		
20	**0.8**			
20	**1**	81,54	92,52	
40	**0.1**			
40	**0.2**	93,58	92,24	
40	**0.4**			
40	**0.6**			92,47
40	**0.8**			
40	**1**	80,34		
60	**0.1**		92,24	92,48
60	**0.2**			
60	**0.4**		94,06	
60	**0.6**			
60	**0.8**			
60	**1**		92,57	
80	**0.1**			
80	**0.2**	93,08		
80	**0.4**	94,76		
80	**0.6**		93,08	93,32
80	**0.8**			
80	**1**	81,64	92,70	92,94
100	**0.1**	85,51		93,53
100	**0.2**		92,35	
100	**0.4**		92,34	
100	**0.6**			
100	**0.8**	89,63		
100	**1**			

From the new execution, it is clear that the configurations with the best accuracy are those with a value of gamma lower than 0.6. The C parameter does not seem to affect the performance of the algorithm. Furthermore, we see that the rbf kernel is the best performing, though some configurations with polynomial kernel are also outstanding. In order to obtain the best hyperparameter configuration, the best performing configurations (those marked in red) are thoroughly compared to see which one has the best performance.

6.4.2.2.1 Comparison of the best performing SVM configurations

As in the previous case, to get a broader picture of how the configurations behave, all training/test users ratios are used. In first place the nomenclature of the configurations is explained.

- **KL**rbf refers to the used kernel, in this case rbf.
- **DG**2 refers to the degree using when using polynomial kernel, in this case 2.
- **GM.**2 refers to the gamma used, in this case 2.
- **C**20 refers to the regularization parameter used, in this case 20.

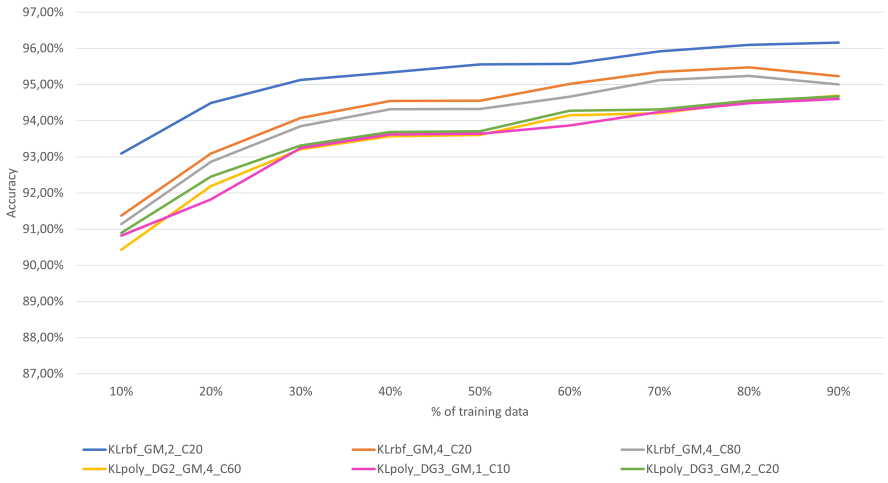


Figure 74: Accuracy results for the best SVM configurations.

Although analysing the accuracy results (Figure 74) of the configurations, it can be seen that the *KLrbf_GM.2_C20* configuration stands out from the rest, it is only about 2 % better. On the other hand, focusing on the metrics shown in Table 31 we have a broader knowledge of the configurations. The configurations *KLrbf_GM.2_C20* has the best accuracy, but it has a poor FNR compared to the other rbf configurations, on the other hand, *KLrbf_GM,4_C80* is 1 % worse in accuracy, but it has the best FNR and FLR, that are crucial metrics. The configurations with polynomial kernel have good accuracy, however, they have high FNR. Focusing on the time, the polynomial kernel configurations are about five times faster than the rbf kernel. Considering this, we can choose *KLrbf_GM0.4_C80* as the best configuration for the SVM algorithm.

Table 31: Accuracy results for the best SVM configurations.

Configurations	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)	Training Time (ms)	Predict Time (ms)
KLrbf_GM.2_C20	1,23	2,91	0,29	95,57	954,73	1078,37
KLrbf_GM.4_C20	0,78	3,96	0,24	95,02	1008,53	1630,91
KLrbf_GM.4_C80	**0,60**	**4,57**	**0,16**	**94,66**	**903,05**	**1652,14**
KLpoly_DG2_GM.4_C60	3,11	2,52	0,22	94,15	202,73	132,03
KLpoly_DG3_GM.1_C10	2,57	3,35	0,21	93,87	168,70	148,30
KLpoly_DG3_GM.2_C20	2,46	3,04	0,22	94,28	180,96	160,15

6.4.2.3 kNN Optimization

As usual, first of all the corresponding hyperparameters of the ML algorithm are explained below.

- **The number of neighbours (k)**: Number of neighbours taken into account when classifying a new input.
- **Weights (WH)**: This parameter establishes the distance-weights used in the prediction. It can be *uniform* (un), which considers all the points equally, or *distance (di)*, which weight the points by the inverse of the distance.
- **Distance metric (DM)**: The type of distance used to calculate. It can be *Euclidean* (ec), *Manhattan* (mh), *Minkowski* (mk). Table 32 shows the equations of the different distances.

Table 32: Distance equations for kNN algorithms.

Distance	Equation
Euclidean	$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$
Manhattan	$d(p,q) = \lvert p_1 - q_1 \rvert + \lvert p_2 - q_2 \rvert$
Minkowski	$d(p,q) = \left(\lvert p_1 - q_1 \rvert^3 + \lvert p_2 - q_2 \rvert^3\right)^{\frac{1}{3}}$

As this algorithm has few hyperparameters, and two (weight and distance metric) have limited options, we can select a broader range for the remaining hyperparameter. The range for the number of neighbours has been set from 1 to 25. Regarding weights and distance metrics, all the possibilities are used.

Table 33: Hyperparameters ranges for the first execution of RS optimization for kNN.

Hyperparameter	Values
Number of neighbours (k)	[1,25]
Weights (WH)	[uniform, distance]
Distance metric (DM)	[Euclidean, Manhattan, Minkowski]

Table 34: Accuracy results of the first execution of RS optimization for kNN (%). In red the area with the best results.

distance metric k/weights	Manhattan un	Manhattan di	Euclidean un	Euclidean di	Minkowski un	Minkowski di
1			94,40			
2					88,16	
3		95,76				
4						
5	95,41			94,04		
6				94,19		89,62
7		94,28				
8				93,28	87,18	
9						
10						87,92
11				91,45		
12			90,54		85,14	
13	92,52					
14		93,38				
15						
16						
17				89,62		
18					82,52	
19	90,19					
20						
21						
22						
23			86,09			
24				87,52	79,96	
25	87,90					

As done before, once the values of the hyperparameters are set, in the first round of the GS optimization, the algorithm is run 15 % of all possibilities. The accuracy results of the executions are shown in Table 34.

Analysing the results, it is observed that increasing the number of neighbours (k) worsen the accuracy, so the new k values are set within the range [1, 7]. On the other hand, the Manhattan and Euclidean distance metrics show the best performance, so they are maintained for the next round. Focusing on the weights, there is no evidence of which one performs better, so it remains untouched. After selecting the new ranges for the hyperparameters, eight different random configurations are executed in the second round of GS optimization. The latest results can be seen in Table 35. The underlined values are from the first round of the GS.

From the new results, it is clear that Manhattan distance metrics perform better, however, it is it remains uncertain which weight works the best. Furthermore, we can state that the values of k that give the best results are between 1 and 5. To find

Table 35: Accuracy results of the second execution of GS optimization for kNN (%). In red the area with the best results.

distance metric k/weights	Manhattan un	Manhattan di	Euclidean un	Euclidean di
1		95,18	92,40	
2	95,34		93,54	
3		95,76		94,32
4		95,98		
5	95,41	95,86	93,92	94,04
6	94,15			94,19
7		94,28		

the best performing configuration, a more in-depth analysis of the six best performing configurations is done.

6.4.2.3.1 Comparison of the best performing kNN configurations

Just as in previous sections, to get a more detailed understanding of the behaviour of the algorithm, the different configurations are compared by using different training/testing ratios in addition to the metrics obtained from the confusion matrices. As done previously, for each training/testing ratio, the algorithm has been executed twenty times, and the value showed is the mean value of all of them. A brief explanation of this notation is given below to clarify the notation used for naming the configurations.

- **k**3 refers to the number of neighbours used, in this case 3.
- **DM**mh refers to distance metric used, in this case Manhattan.
- **W**un refers to the weight applied to the distance, in this case uniform.

As can be seen in Figure 75, all configurations have similar behaviour, with the most significant differences found when training with 10 to 40 % of training users. This behaviour is because the algorithm with little data does not generalise, so the data in the configuration produces more noticeable changes.

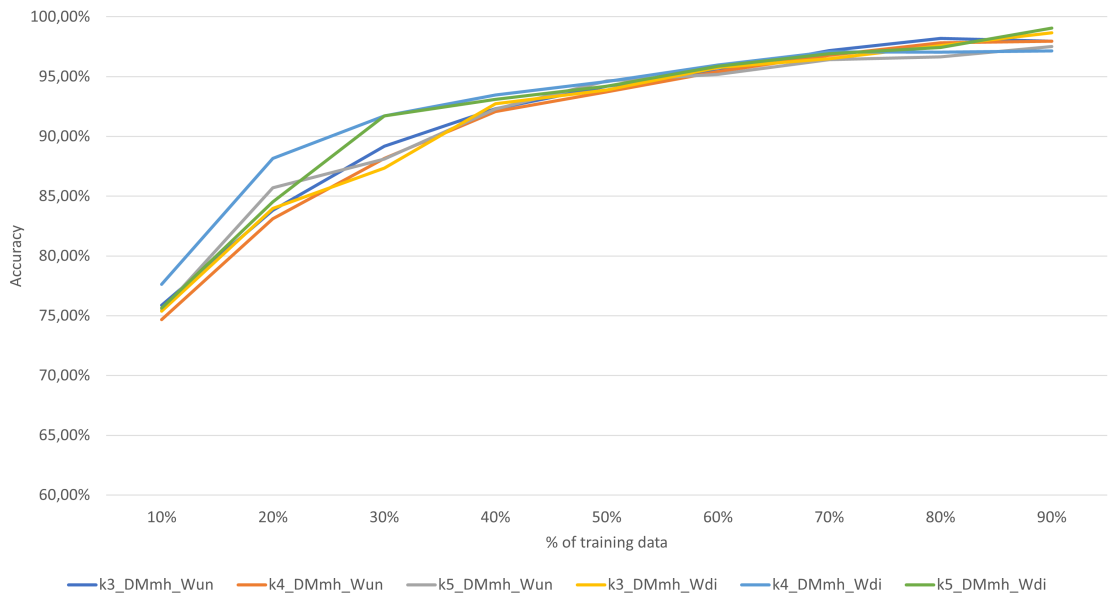


Figure 75: Accuracy results for the best kNN configurations.

By analysing the metrics of the configurations (Table 36), we can observe that, as happens above, the different configurations show similar results. It can be seen that the configurations using distance weight perform slightly better. Although all configurations achieve similar results, the *k4_DMmh_Wdi* configuration has a lower FNR and FLR, making it the best of all configurations.

Table 36: Metrics of the best performing kNN configurations..

Configurations	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)	Training Time (ms)	Predict Time (ms)
k3_DMmh_Wun	3,98	0,35	0,20	95,47	2,55	544,00
k4_DMmh_Wun	4,07	0,38	0,14	95,41	2,47	593,34
k5_DMmh_Wun	4,46	0,24	0,12	95,18	2,64	399,39
k3_DMmh_Wdi	3,78	0,38	0,08	95,76	2,53	425,56
k4_DMmh_Wdi	**3,72**	**0,22**	**0,08**	**95,98**	**2,47**	**493,12**
k5_DMmh_Wdi	3,74	0,22	0,18	95,86	2,66	493,67

6.4.2.4 Results of Hyperparameter Optimization

By optimising the hyperparameter, we can adjust the performance of the algorithms to the current dataset, which results in increased accuracy. In order to measure the impact that the optimisation process has on the algorithms, Table 37 shows the metrics before and after hyperparameter optimisation.

Analysing the accuracy results of the algorithms, we see that in all the cases, the accuracy has improved. If we focus on the CART algorithm, it has been enhanced by 6.14 %, being the main improvement the FNR which is reduced by 4.79 %. Moreover, the SVM algorithm has improved by 3.85 %, which is reflected in all metrics. Lastly, the

Table 37: Metrics and times comparison before (up) and after (down) optimization.

	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)	Training Time (ms)	Predict Time (ms)
			Non-optimised algorithm			
CART	9.55	5.76	1.21	83.48	368.87	3.51
SVM	2.17	6.57	0.09	91.17	1413.5	2309.93
kNN	12.07	0.13	0.22	87.58	2.76	652.50
			Optimised algorithm			
CART	4.76	5.13	0.90	89.21	292.53	3.85
SVM	0.78	3.96	0.24	95.02	1008.53	1630.91
kNN	3.72	0.22	0.08	95.98	2.47	493.12

algorithm with the most considerable improvement is kNN, which has been enhanced by 8.4 %, with FNR showing the most remarkable improvement, reducing its value by three times. Related to the times, we can see that all the algorithms have reduced the training and testing duration, with SVM being the most affected by this reduction, although it is still the slowest.

Considering the results, it can be seen that all algorithms have improved due to hyperparameter optimisation. However, although kNN has higher accuracy, the value of FNR is five times higher than that of SVM, and since we are classifying pathological walks, this is a decisive parameter. For these reasons, it is considered that SVM is still the best choice to classify the pathologies.

After obtaining an optimized algorithm, we can study the viability of optimising the dataset. The following section presents various dimensionality reduction techniques to reduce the dataset.

6.4.3. Dimensionality Reduction Approaches

The datasets with high dimensionality may be prone to having redundant or irrelevant data, which results in the increase of the difficulty of the algorithms learning process, in addition to computational costs and storage issues produced by them [103]. To alleviate this, dimensionality reduction techniques can be used, which can reduce the dimensionality of the dataset by removing the correlated and redundant features maintaining the characteristics of the original space. There exist two main techniques of dimensionality reduction: feature extraction and feature selection [104].

- **Feature extraction** transforms the original features into others that are more significant, so the new features are a linear combination of the original. This method can visibly reduce the dataset by grouping the information into a few variables, however, the new features lose the units (e.g., kilometres, seconds, angles, etc.).
- **Feature selection** ranks the features by their importance so the most relevant features can be chosen, however, a feature that is ranked as non-important by itself

can have high performance when considered with others, or two variables classified as unnecessary on their own might be useful together [105].

Concerning which technique is better, it can be affirmed that no single dimensionality reduction method performs the best in all the datasets. Thus, in order to determine which method is best suited to the current dataset, various methods and approaches are used to reduce the dimensionality of the dataset.

6.4.3.1 Feature Extraction

As said above, feature extraction reduces the dimensionality of a dataset by transforming it into a new one maintaining meaningful information. Reducing the number of variables tends to a reduction in the accuracy, however, it increases the simplicity of the algorithm. There are two main approaches of feature extraction techniques, supervised and unsupervised approaches [106]. The supervised approaches consider the labels in the calculation, while unsupervised approaches do not use them at all.

The most known unsupervised dimensionality reduction technique is Principal Component Analysis (PCA), however, there exist other methods such as Independent Component Analysis (ICA) or Non-negative Matrix Factorization (NMF). On the other hand, Mixture Discriminant Analysis (MDA) and Linear Discriminant Analysis (LDA) are supervised approaches, being LDA the most famous one.

In order to compare which method produces better results, LDA and PCA are used separately, and the results are compared. A brief explanation of the approaches and the effects of applying them to the data are explained below.

6.4.3.1.1 Principal Component Analysis

PCA is a non-supervised dimensionality reduction technique with the objective of finding a new space that represents the direction of the maximum variance [107]. So, given the dataset $X = x_1, x_2, \ldots, x_n$, the objective of PCA is to find a new dimensional space (W) dimensionality lower than the original, which maximise the variance in the dataset. The new features are called Principal Components (PCs) and are sorted from PC_1 to PC_n. The direction of the new space is given by the first PC, which is the one with the highest variance. Below are the steps to apply the PCA transformation.

1. The dataset is standardized (so that all the variables have mean = 0 and variance = 1).

2. The covariance matrix is obtained (this helps to identify the variables that are correlated).

3. The Eigenvectors (principal components) and eigenvalues are obtained from the covariance matrix. The eigenvectors and the magnitude define the new feature space direction by the eigenvalues (e.g., the eigenvalues show the data variance in the new feature space).

4. The eigenvectors are sorted using the eigenvalues from highest to lowest, and the k first are selected.

5. The new data matrix is created by applying the equation $Y = XxW$, where W is the k-dimensional Eigenvector matrix.

PCA is a dimensionality reduction technique, however, how many components should we use to represent the original data without a lack of information? A standard method to identify the optimal number of PCs is to use the cumulative variance and see when the increase is no longer significant. Figure 76 shows the cumulative variance against the number of PC for the current dataset.

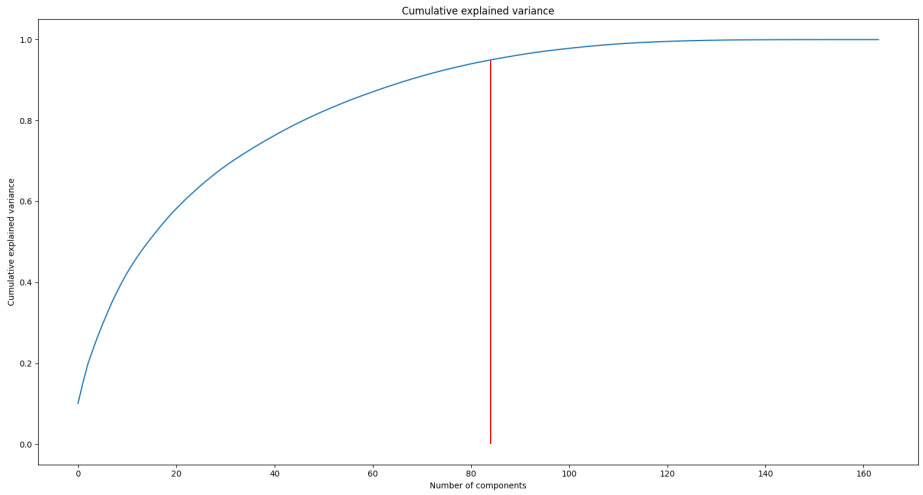


Figure 76: Cumulative variance by PCA.

As it can be seen the 95 % of the variance is reached using 84 PCs, i.e., 95 % of the original information is kept using only 84 PCs. On the other hand, to reach 100 % of the variance, we need to add the remaining 80 PCs, that is, the last 80 PCs contain only 5 % of the original information. After studying the optimal number of PCs, we can explore the behaviour of the algorithm using different numbers of PCs. To obtain a broad knowledge of how the algorithms behave, the number of PCs used goes from 1 to 100.

Figure 77 presents the accuracy results of the algorithms using a different number of PCs. It may be noticed that the three of them have a logarithmic shape, rapidly increasing the accuracy with the first PCs. This behaviour is expected since PCA ranks the principal components with the highest variance (i.e., the ones with the most information)

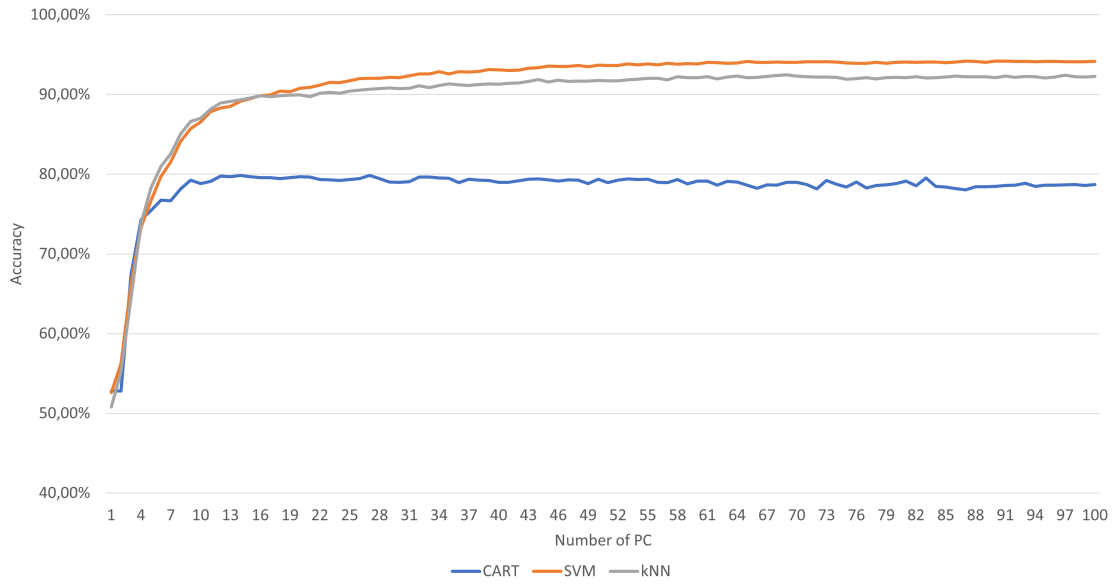


Figure 77: Accuracy results against different number of PCs.

at the beginning, therefore, the first PCs produce a more significant increase in accuracy than those at the middle-end. Evaluating the algorithms separately, the CART algorithm reaches the maximum accuracy of 80.01 % with 24 PCs, beyond this point, the accuracy decreases up to 78.21 % with 72 PC. So, using PCA with CART worsen the results of the algorithm.

In the case of the SVM algorithm, the best accuracy is obtained using 90 PCs, reaching 94.18 % accuracy. However, it can be seen that beyond 53 PCs, the increase of the accuracy when adding a new PC is smaller than 0.5 %. The difference in accuracy when using 90 PCs instead of 53 PCs is 0.34 %. As the difference between the two settings is small, it is impossible to decide which is better using accuracy alone, so both configurations are compared in detail below.

Lastly, the kNN algorithm reaches an accuracy of 92.45 % with 69 PCs. As happens with SVM, there is a point at which using more data produces a minor increase in accuracy. kNN algorithm is able to correctly classify 91.88 % of the cases using 44 PCs. The extra 25 PCs increase the accuracy by 0.57 %, that is, each additional PC produces an increase of 0.00228 % in the accuracy. Both configurations are also discussed in detail below.

The Table 38 shows the metrics of the algorithms before and after applying PCA, as CART does not benefit from using PCA is not shown in the table. Analysing the metrics in detail, it can be noticed that the SVM algorithm has reduced the accuracy by approximately 1.5 %, which is reflected in the FNR, i.e., false negatives worsen, which means that applying PCA worsens the algorithm. Comparing the 50 and 90 PC configurations, we can see that both have very similar results, with the differences in accuracy and FNR being less than 0.5 %.

Otherwise, the kNN algorithm also reduces the accuracy, in this case, by about 3 %. Contrary to SVM, kNN reduces the FNR, but the FPR increases considerably. If we compare the kNN configurations using PCA, we see that they have a difference in the accuracy of about 2 %, although the FNR and FLR values are very similar in both.

Regarding to times, it can be seen that using PCA reduces the training and predict times, although this reduction is more noticeable in SVM than in kNN. However, this was an expected consequence, as less data is used than in the original configurations.

Table 38: Metrics using the original and PCA features.

Algorithm	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)	Training Time (ms)	Predict Time (ms)
SVM	0,78	3,96	0,24	95,02	1008,53	1630,91
SVM_53PC	2,51	3,44	0,16	93,89	322,26	907,81
SVM_90PC	2,87	2,61	0,24	94,27	620,31	1224,49
kNN	3,72	0,22	0,08	95,98	2,47	493,12
kNN_44PC	2,41	5,53	0,28	91,78	0,44	367,69
kNN_69PC	2,73	3,88	0,24	93,15	0,57	368,10

Based on the results, it can be seen that data transformation using PCA does not improve the results, but it improves the algorithms speed. In the case of kNN and SVM, they have similar, albeit worse, results, however, CART performs poorly when using PCA. Comparing the configurations with fewer PCs, we can say that they behave similarly to configurations with more PCs, and in the case of SVM is faster.

Finally, it can be concluded that, although there are configurations with good performance, due to the worsening of both accuracy and FNR (for SVM), it is not recommended to apply PCA. As a consequence of the weak performance offered by PCA, we can conclude that the characteristics in the dataset are poorly correlated.

6.4.3.1.2 Fisher Linear Discriminant Analysis

Fisher Linear Discriminant Analysis or Linear Discriminant Analysis is a supervised dimensionality reduction technique that reduces the dimensions in the dataset by projecting the original data onto a lower-dimensional space. It is close to PCA since both are based on linear transformations, however, the objective of LDA is to maximize the ratio of between-class variance to within-class variance in order to increase the separation between classes [108]. LDA works by projecting the data onto a lower-dimensional space, so given a N-dimensional dataset with C classes, the resulting new space would have C-1 dimensions (or features). The process of finding the new subspace can be divided into the following steps:

1. Calculate the between-class variance (i.e., the separability between the different classes)

2. Calculate the within-class variance (i.e., the distance between the mean and the samples of each class).

3. A new lower-dimensional space is calculated, which minimize the within-class variance and maximize the between-class variance.

This is a short explanation of how LDA works, in case further knowledge is needed, the papers [108], [109] are very useful. Once how LDA works has been briefly explained, we can show the results of applying it to the dataset. Since the used dataset has three different classes (healthy, right limp and left limp), LDA has transformed it into a new one with two features. As it can be seen in Figure 78, the new features are grouped into clusters according to the three classes.

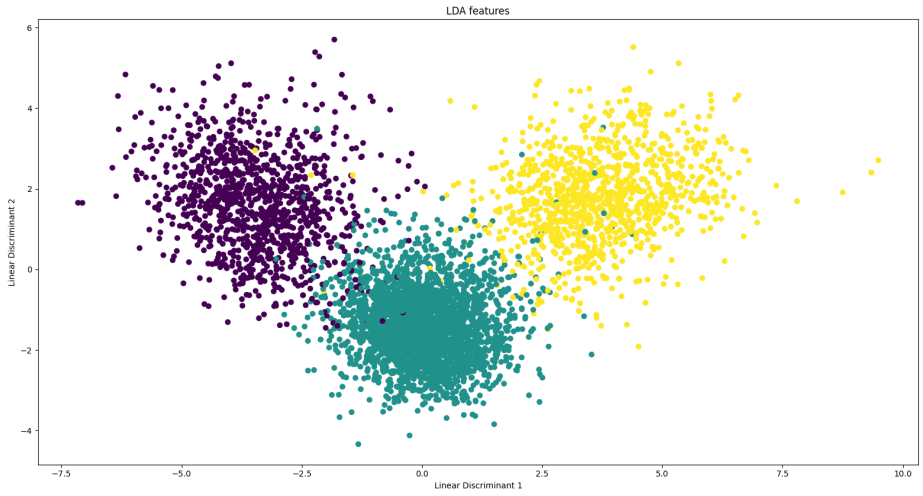


Figure 78: Representation of LDA features.

After the LDA transformation, the new data is used to feed the ML algorithms. As the new dataset has only two features, these are used entirety. The results of the execution are shown in Table 39.

Table 39: Metrics using the original and LDA features.

Algorithm	**FNR (%)**	**FPR (%)**	**FLR (%)**	**Accuracy (%)**	**Training Time (ms)**	**Predict Time (ms)**
CART	4,76	5,13	0,90	89,21	292,53	3,85
CART_LDA	2,62	2,01	0,78	94,59	4,92	0,32
SVM	0,78	3,96	0,24	95,02	1008,53	1630,91
SVM_LDA	1,62	1,96	0,79	95,63	109,16	149,94
kNN	3,72	0,22	0,08	95,98	2,47	493,12
kNN_LDA	1,66	2,26	0,68	95,40	1,60	114,63

By analysing the metrics, it can be seen that all the algorithms using LDA data show an accuracy close to 95 %. If we focus on CART, we can see the most significant

improvement, increasing its accuracy by about 6 % and reducing the FNR and FPR by about 2,5 %. Those changes make CART viable for classifying pathologies. In the case of SVM, the most significant change comes from the increase in FNR, which has doubled, and FLR, which has increased by about three times, lastly, the FPR has reduced by half. Although the accuracy of the SVM has risen by 0.6 %, the FNR and FLR has increased, which can be translated into a worsening of the algorithm performance. In the case of kNN, the accuracy has worsened by 0.5 %, however the FNR has improved by 2.06 %. On the other hand, the FPR and FLR have increased significantly. Those changes can be interpreted as an improvement of the algorithm, which is now more sensitive to false negatives.

As far as training and test times are concerned, the algorithms are a lot faster because LDA reduces the dataset into two features.

It can be concluded that the transformation of the dataset using LDA positively impacts the algorithms. The CART algorithm has greatly improved its performance, making it viable for pathology classification. On the other hand, although the FNR and FLR of SVM have increased slightly, the algorithm is now more balanced between false positives and false negatives. Finally, the kNN algorithm reduces the FNR, which is considered an improvement. Although both kNN and SVM take advantage of the LDA transformation of the data, SVM can be viewed as the best performing algorithm.

6.4.3.1.3 Overview of feature extraction

The goal of feature extraction is to summarize the information from a dataset into a few new features, maintaining the original dataset information. However, when transforming the data, the new features lose the connection to a physical quantity, making them not intuitive to interpret. The approach to transforming the dataset depends on the nature of the original dataset. In this case, as the dataset features do not have a strong correlation between them, PCA is not the optimal method.

When using PCA, you can select the number of PCs you want to use, thus making the method more flexible, in the case of LDA, as the dataset has three classes, LDA transforms it into a new one with two features, i.e., it summarises the information of 201 characteristics in 2. From the above results, we can say that transforming the dataset can improve the performance of the ML algorithm, not only making them faster but increasing their accuracy.

6.4.3.2 Feature Selection

Feature selection is a dimensionality reduction technique to select a subset of features from the input dataset, which efficiently describe the original dataset while minimizing the impact of irrelevant or redundant variables [110]. The different feature selection methods can be grouped into three, depending on the approach used, the groups are as follows:

- **Filter methods** use statistical methods to rank the features by their relevance (or correlation) against the output variable.
- **Wrapper methods** work by training an ML model in a subset of features and modifying (adding or removing) features to it depending on whether they make the model better or worse. Due to these methods use an ML model, and it must be trained repeatedly, these methods are computationally expensive.
- **Embedded methods** are like wrapper methods since they also use an ML model to obtain the best features, however, the main difference is embedded methods use model metrics to rank the best features.

Several methods could be used to select the best features of the dataset, however, in this study, only three are used. These methods are briefly explained below.

1. **Recursive Feature Elimination (RFE)** is a wrapper feature selection algorithm. RFE works by building a model with all the features and then recursively removing features and building the model again. Using the accuracy of those executions, the algorithm can identify which attributes (or combination of them) are more relevant and contribute more to predict the output.
2. **Random Trees Feature Selection (RTFS)** is an embedded method that uses a Random Forest algorithm that is trained with the dataset, next the feature importance is calculated using the Gini impurity of each node. This method allows plotting the cumulative importance of the features, as shown in Figure 79 the 95 % of the importance are obtained using 99 features.
3. **LASSO** is an acronym for Least Absolute Shrinkage and Selection Operator, this embedded method applies a shrinking process in which the coefficients of the regression variables are penalised, reducing the value of some of them to zero. At the end of the process, the parameters reduced to zero are discarded from the model, and the others are used. The paper [111] provides a deeper insight into the LASSO feature selection process.

The Figure 80 ,Figure 81, Figure 82 show the accuracy result of the different ML algorithms using a different number of features selected by the feature selection methods. In general, all three algorithms rapidly increase accuracy with the first few features before levelling off at a certain point. This behave, similar to PCA, is because the feature selection methods rank the features by the information they provide, with the first features having the most information. In any case, the best performance is achieved with RTFS features, however, with very low numbers of features (less than 10), LASSO is the best performing feature selection method.

In the case of the CART algorithm, RTFS features provide approximately 2 % better results than the other feature selection algorithms. The highest accuracy of *CART_RTFS*

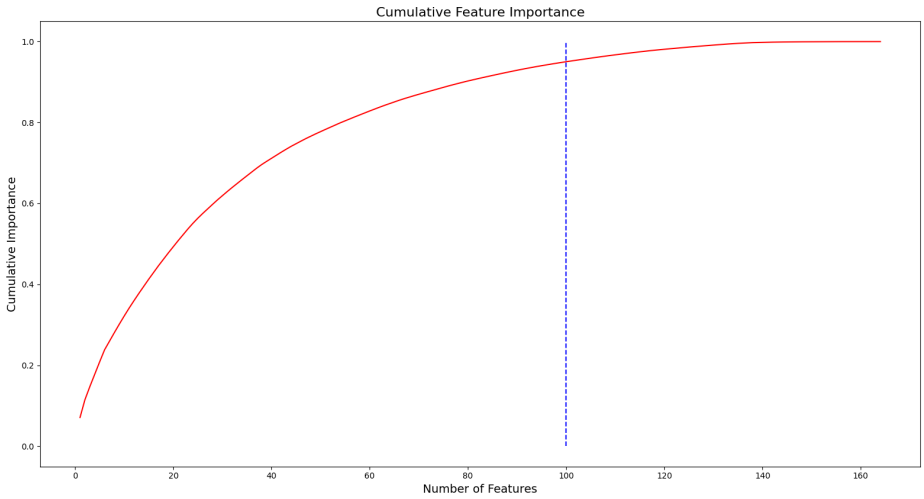


Figure 79: Cumulative importance using tree-based feature selection.

is achieved when using 84 features, for which the algorithm reaches an accuracy of 89.38 %, however, using only 31 features, the algorithm correctly classifies 88.30 % of the cases. That is, a 1 % increase in accuracy is achieved by using 53 more features, i.e., each extra feature increases the accuracy by 0.02

In the case of the SVM algorithm, it reaches up to 96.38 % accuracy with 94 features, which corresponds to the most accurate configuration. However, as indicated above, there is a point where the accuracy tends to stabilize, so the improvement becomes weak. In the case of *SVM_RTFS*, the stabilization starts with 45 features, so that with about half of the features of the best configuration, 95.45 % accuracy is achieved, i.e., the additional 49 features only increase the accuracy by about 1 %. Surprisingly, using only the first 20 RTFS features, the SVM algorithm is able to correctly classify almost 92 % of the cases.

Finally, the highest accuracy using kNN is obtained with 97 RTFS features and reaches 96.26 % of accuracy. In this case, the accuracy tends to stabilise using 34 features, where the algorithm has an accuracy of 95,16 %. Like the SVM algorithm, kNN also achieves a remarkable accuracy, using only 20 features, in this case, kNN correctly classifies 93.34 % of the cases.

Some configurations have been pointed out above, some for being the best performing and others for achieving remarkable accuracy with few features. To study these configurations in-depth, Table 40 shows the metrics of these configurations, as well as the original configurations. By analysing the metrics, it can be seen that using the RTFS features, all the algorithms are more accurate and faster.

The improvement in the accuracy of the CART algorithm is negligible, on the contrary, the value of FNR worsens, so we can state that feature selection is not helpful when using the CART algorithm. In the case of SVM, if we compare the original configuration with *SVM_RTFS_94* we see that the accuracy has improved by 1.3 %, but the FNR

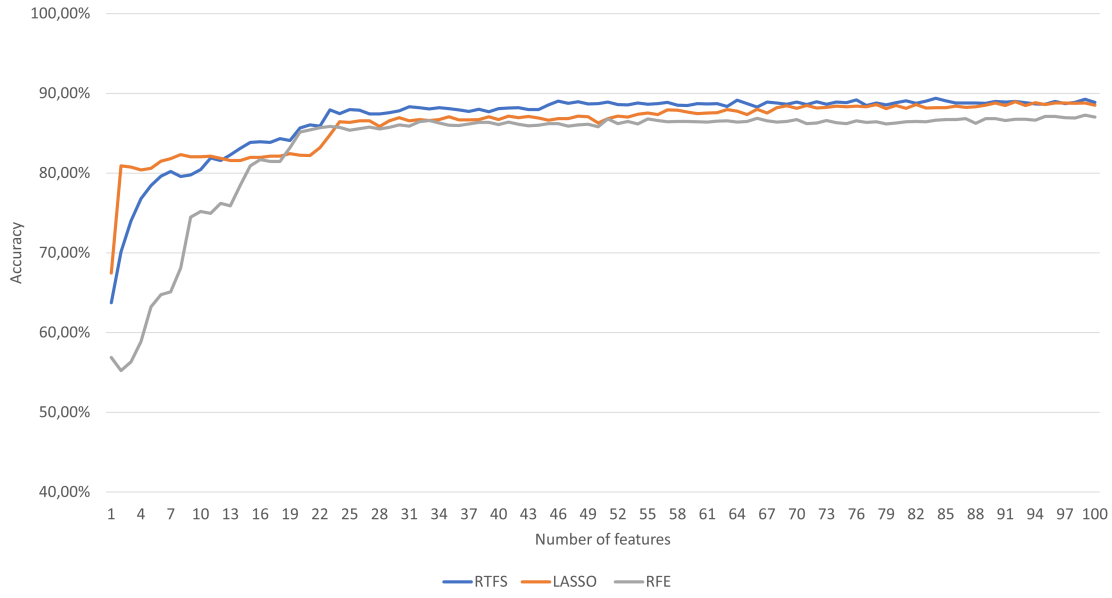


Figure 80: Accuracy results using different number of features for the CART algorithm.

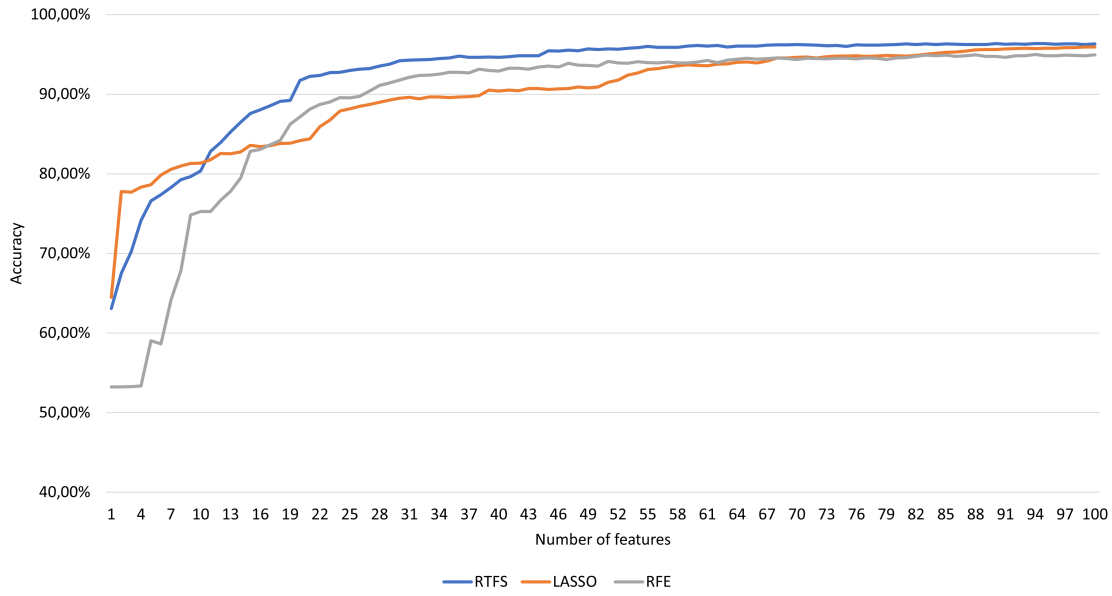


Figure 81: Accuracy results using different number of features for the SVM algorithm.

Table 40: Metrics using the original and RTFS features.

Algorithm	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)	Training Time (ms)	Predict Time (ms)
CART	4,76	5,13	0,90	89,21	292,53	3,85
CART_RTFS_84	5,99	3,50	0,88	89,63	199,89	2,80
SVM	0,78	3,96	0,24	95,02	1008,53	1630,91
SVM_RTFS_94	1,35	2,01	0,32	96,32	295,22	830,64
SVM_RTFS_49	1,51	2,69	0,28	95,52	131,50	369,91
SVM_RTFS_20	2,69	5,29	0,30	91,72	113,02	315,23
kNN	3,72	0,22	0,08	95,98	2,47	493,12
kNN_RTFS_97	1,65	1,83	0,22	96,30	1,89	469,92
kNN_RTFS_34	1,63	3,06	0,16	95,16	1,43	358,15
kNN_RTFS_20	2,09	3,72	0,20	93,99	1,35	360,41

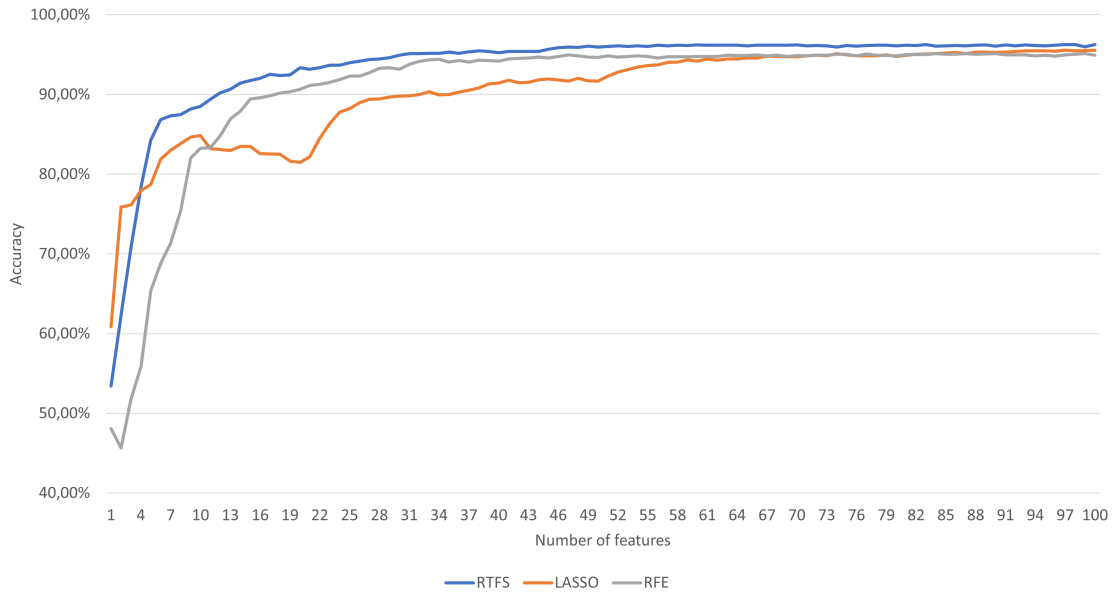


Figure 82: Accuracy results using different number of features for the kNN algorithm.

has increased by 0.5 %, as well as the FPR has halved. Although this configuration has worsened in FNR, it is more balanced across the different ratios. Comparing *SVM_RTFS_94* and *SVM_RTFS_49*, we can say that both have similar results, the former being slightly better and the latter slightly faster. Finally, the kNN algorithm has not only increased the accuracy but also reduced the FNR by about 2 %, although the FPR and FLR have increased. As happens with SVM, using feature selection makes the algorithm more balanced in terms of false positives and false negatives. The *kNN_RTFS_97* and *kNN_RTFS_34* configurations behave similarly, and as with SVM, the one with more features performs a little better, and the one with fewer a little faster.

At this point, it only remains to comment on the configurations using 20 features. If we compare the results of the best accurate configurations (*SVM_RTFS_94* and *kNN_RTFS_97*) with the configurations using 20 features, it is clear that to use 20 features offers a less accurate resultS. However, these configurations have the advantage of classifying walks with an accuracy of up to 94 % using only 20 features. To find out what these features are and to see the possibilities of these configurations, Table 41 shows the first 20 features selected by RTFS.

If we observe the signals, we can see that 6 of them correspond to the average value of the signals, 7 of them correspond to the moment at which the signal reaches its maximum value, 6 of the corresponding to the zero-crossing rate and 1 of them to the double support time. On the other hand, we can see that the signals from the left side are distributed among the different positions, but the signals from the right side are all from the IMU placed on the thigh, and they are either from the accelerometer or the gyroscope. Moreover, of those 20 signals, the first 15 correspond to the left side of the body, and only four are from the right side. In the light of those results, we can reduce not only the features used but also the number of IMUs to collect the database. Analysing the signals,

Table 41: First 20 features selected by RTFS.

Feature	RTFS
1^{st}	Left Foot Gyroscope average value
2^{nd}	Double Support Time I
3^{rd}	Left Hip x plane average value
4^{th}	Left Ankle z plane maximum momentum
5^{th}	Left Knee z plane average value
6^{th}	Left Thigh Accelerometer maximum momentum
7^{th}	Left Ankle x plane average value
8^{th}	Left Hip z plane zero-crossing rate
9^{th}	Left Foot Gyroscope zero-crossing rate
10^{th}	Lumbar Accelerometer maximum momentum
11^{th}	Left Knee z plane zero-crossing rate
12^{th}	Left Leg Gyroscope maximum momentum
13^{th}	Left Leg Gyroscope average value
14^{th}	Left Ankle z plane zero-crossing rate
15^{th}	Right Thigh Gyroscope average value
16^{th}	Left Hip Accelerometer maximum momentum
17^{th}	Right Thigh Gyroscope maximum momentum
18^{th}	Left Leg Accelerometer maximum momentum
19^{th}	Right Thigh Accelerometer zero-crossing rate
20^{th}	Right Thigh Gyroscope zero-crossing rate

we can see that none of the first 14 are from the IMUs placed on the right leg and ankle.

Table 42: Metrics for the configurations using 14 RTFS features.

Algorithm	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)	Training Time (ms)	Predict Time (ms)
SVM_RTFS_14	5,83	6,96	0,72	86,49	144,87	410,40
kNN_RTFS_14	3,18	5,15	0,25	91,42	1,37	364,19

Table 42 shows the metrics of the algorithms using the first 14 features. It can be seen that with 14 features, the performance of the SVM algorithm drops substantially. However, the kNN algorithm still being suitable, with 91,42 % of accuracy. The FNR and FPR have indeed increased, although the FLR remains stable, however it should be noted that this configuration uses only 14 features.

In this section, some feature selection algorithms based on different approaches have been tested. In the light of these results, we can conclude that for all tested methods, the best results have been obtained using the features selected by RTFS.

Although the SVM and kNN algorithms have similar results, it may be observed that compared to the original configurations, kNN shows an improvement in both accuracy and

FNR. In contrast, SVM improves accuracy but worsens the FNR. On the other hand, it is observed that by reducing the number of features, kNN maintains more consistent results compared to SVM, where a low number of features produces a significant worsening of the results. Considering this behaviour, we can state that when using RTFS feature selection, kNN is the best performing algorithm.

Finally, it can be concluded that the application of feature reduction techniques not only reduces the processing time of the algorithms but can also increase the accuracy of the algorithm. This behaviour is due to many of the variables being correlated or having noise, which hinders the learning process of the algorithm.

6.4.3.3 Feature Extraction vs Feature Selection

Although both approaches aim to reduce the dimensionality of the dataset, feature extraction works by transforming the features by linear combinations of the original features. On the other hand, feature selection works by ordering the variables according to their importance, the calculation of importance varies according to the method used.

Table 43: Results of the best feature extraction and feature selection configurations (feature extraction up, feature selection down).

Algorithm	FNR (%)	FPR (%)	FLR (%)	Accuracy (%)	Training Time (ms)	Predict Time (ms)
SVM_LDA	1,37	1,71	0,29	96,63	109,16	149,94
kNN_RTFS_97	1,65	1,83	0,22	96,30	1,89	469,92

Comparing the results of the best performing feature extraction and feature selection configurations (Table 43), it is clear that the two configurations offer very similar results. A detailed analysis of the two methods reveals the following results.

- *SVM_LDA* performs 0.33 % better, however, all 201 features have been used to obtain the new two linearly separable features. This implies that when a new user is registered in the database, all the features must be obtained, and the LDA transformation has to be calculated, so the pre-processing time would be longer, even if the classification would be faster afterwards.
- On the other hand, *kNN_RTFS_97* uses the 97 features that have been chosen previously, thus reducing pre-processing times, i.e., not all features need to be calculated, and also allows for a future study of sensor reduction.

Given the results, it can be concluded that feature extraction is more suitable for the current experiment. This can be stated not only because of the results obtained (similar results can be obtained with feature extraction) but also because of the reduction of the features used (which are reduced from 201 to 97) and the possibility to further reduce this number, as kNN still gives an effective result with few features, resulting in a reduction

of the number of sensors. In addition, when using feature selection, the features keep the physical quantity, with the possibility of studying them by an expert.

6.5. Final Algorithm State

To obtain the best performing configuration for the algorithm, three different ML algorithms have been used: CART, SVM and kNN. In this way, it has been possible to determine which algorithm performs best throughout the various experiments that have been carried out. After all the experiments performed, the algorithm differs a bit from the one presented in Figure 63, the new structure of the algorithm is shown in Figure 83. At the beginning of this chapter, a basic algorithm was presented, through the experiments, it can be seen that a data preparation phase has been added as well as the number of features used has been reduced. It has also been shown that the ML algorithm best suited to this problem is kNN.

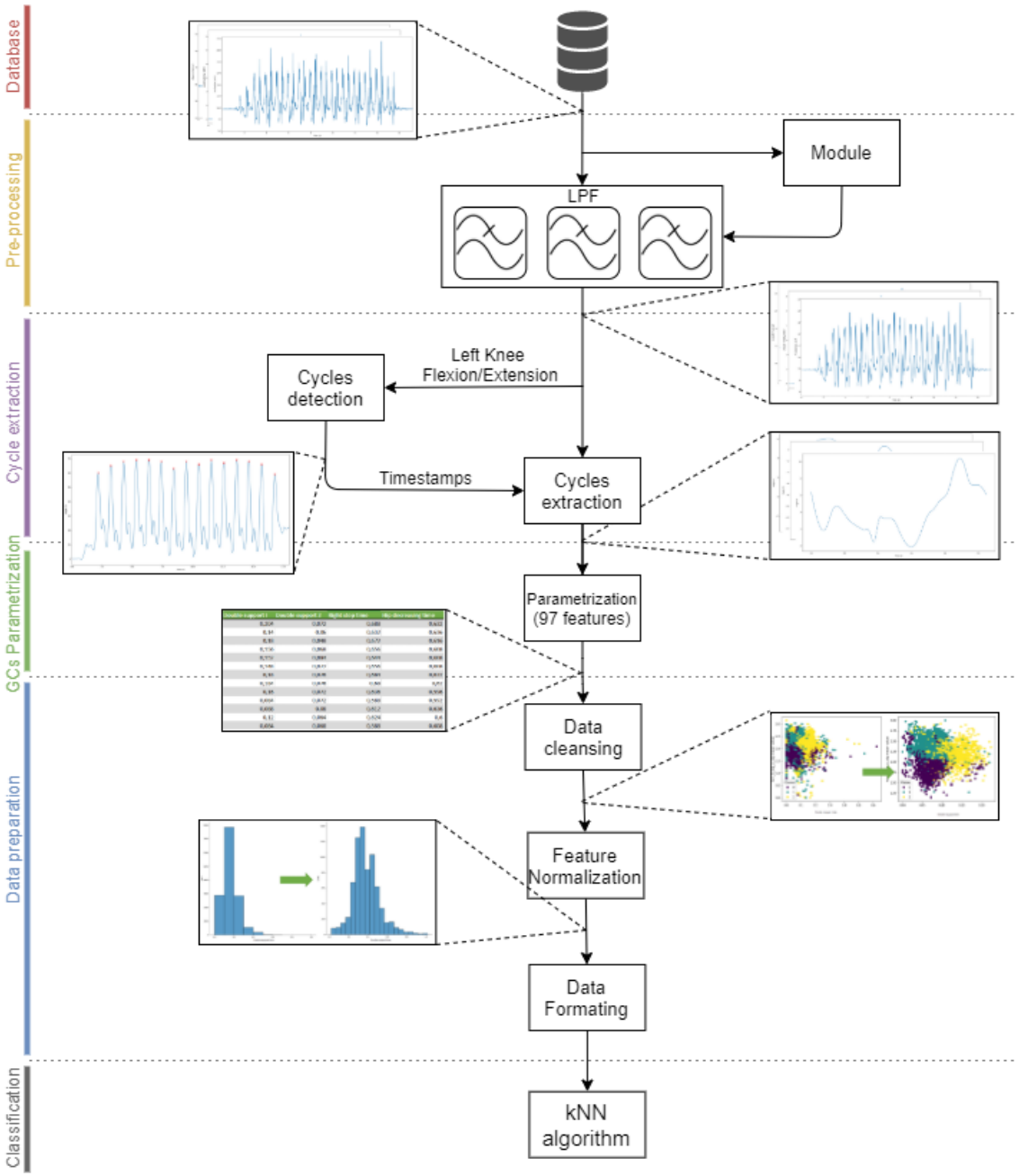


Figure 83: Final algorithm schema.

6.6. Conclusions on feature-based algorithm for pathology detection

This chapter presents a machine learning-based algorithm for pathology classification, as well as its evolution. Like the algorithm presented in the previous section, this algorithm uses kinematic signals as well as joint angles, from which the GCs are extracted. However, this algorithm aims to parameterise these GCs to obtain characteristic features describing the GCs, thereby reducing the amount of data. The algorithm uses 15 different parameters, resulting in 201 features.

From the results of the different experiments, we can state that the best ML algorithm

for creating a pathology detection algorithm is kNN with 96.30 % accuracy using only 97 features out of the 201 initially obtained.

The two first experiments aim to optimise both the dataset and the ML algorithms; however, the main contribution of this chapter is the dimensionality reduction experiment. Not only for having reduced the proposed dataset by almost half but also for having demonstrated that it is possible to reduce the number of IMUs when collecting the database. This can be used as a basis for further studies with the aim of creating a portable and reduced pathology detection system.

7. CONCLUSIONS AND FUTURE RESEARCH LINES

Having described all the work carried out throughout this thesis, this section presents the general conclusions of the work.

7.1. Conclusions

Throughout this thesis, different approaches to pathology detection have been presented. The methods used have in common the need to be able to perform the walks in a place without oppressive restrictions. Although the algorithm based on smartphones is more restrictive, it is due is a first approach to the problem. However, the algorithms based on the professional motion capture system offer complete freedom of location as well as clothing. This is considered an important point, since one of the main objectives we had in mind during the development of this thesis was the portability of the system, since there are currently systems capable of detecting pathologies, but they do not offer such a level of freedom.

Another important point in the creation of algorithms is scalability, i.e., the ability of the algorithm to be used by new people without limitations. Although it may sound strange at first, this has to be seen in the context of artificial intelligence-based algorithms, which have to be trained before they can be used. Without this feature, it would not be possible to use an AI-based algorithm for pathology detection, as would require information about the patient gait before he/she has a pathology.

With this in mind, we can divide the algorithms according to the device used for database acquisition, a professional device and smartphones. The different smartphone-based configurations offer high portability (especially the 1-smartphone configuration) as everyone has a phone. This portability is lost in favour of data when more phones are added to the configuration. However, due to the findings made through the experiments conducted, we can conclude that, due to the quality of the signals and the synchronisation between the phones, it is not possible to perform pathology detection using smartphones.

On the other hand, we have the professional motion acquisition device, which solves the problems found in the smartphone, although it is more cumbersome to set up and carry. Using this device, two pathology detection algorithms based on deep learning and machine learning have been presented. Throughout their corresponding chapters, both algorithms have been optimised; however, the final version of the two algorithms will be used for the comparison of the two algorithms.

Both algorithms use the same signals (angle, accelerometer and gyroscope), and both extract the GCs from these signals. However, while the RNN-based algorithm uses the

GCs for pathology detection, the ML-based algorithm parameterises them.

In terms of results, the RNN-based algorithm achieves 93.7 % accuracy in classifying pathologies while the ML-based algorithm achieves 96.3 %. We can see that both do a great job in classifying pathologies, although the ML-based algorithm offers 2.6 % better results. Based on Figure 84, we can see that the more data used, the better the results of DL versus ML, however, the results are not so disparate as to suggest that this is a problem of dataset size.

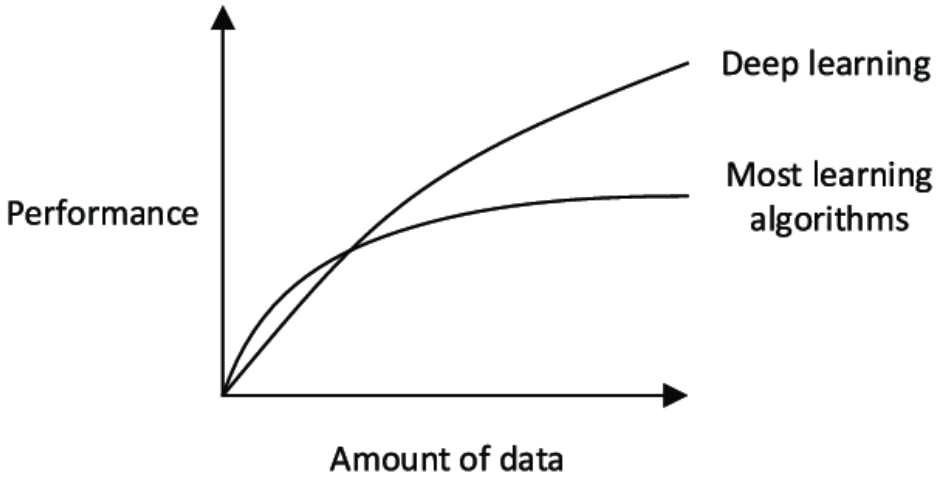


Figure 84: Comparison of the performance of ML against DL [112].

This value disparity is only since the ML-based algorithm fits better with the type of problem presented here, i.e., pathology classification. However, given that the larger the dataset, the better the DL results, if the database were to be increased considerably, it could be the case that the RNN algorithm outperforms the ML algorithm.

On the other hand, comparing the complexity of the algorithms on which they are based (RNN and kNN) we can say that the algorithm based on RNN is much more complex than the one based on kNN. This will imply longer training and testing times, especially considering that kNN is a nonparametric and lazy algorithm.Furthermore, the kNN algorithm not only performs better, but has been proven to be feasible with a reduced number of features. For these reasons, it can be stated that the kNN-based algorithm is the best algorithm for pathology detection.

If we compare the two approaches performed (smartphones vs. Professional system) we can see that there are approaches that are performed in one and not in the other. One reason for this is that the experiments were conducted in the same order as presented in this paper. That is, as more experiments were conducted, the need arose to add different methods, for example, low-pass filtering is added from the two-phone algorithm and remains until the end. However, this filtering is reconsidered in chapter 5, when the capture device is changed. However, this is not the main reason, but during the experiments, certain configurations have been discarded for different reasons. For

example, it would have been possible to evaluate how the RNN-based algorithm would work with signals from smartphones, but these were discarded because the quality of the signals was not high enough.

7.2. Future Work

Although the validity of the algorithms for pathology detection has been demonstrated throughout the thesis, the evaluation of the algorithms has only been carried out with one type of pathology. This is not sufficient to provide a fully functional algorithm, as there are still many improvements and experiments to be carried out. Some of these are detailed below.

First, and perhaps the most important of these, is the evaluation of the algorithm with real scenarios. Right now, the algorithm is trained and optimised to detect only one type of pathology. The next step is to create a database with real pathologies, in order to test the validity of the algorithm.

Another line of work would be to implement the algorithm in a production scenario, i.e., to use it to identify pathologies in real time, rather than through a previously acquired database. In this way, future advantages and disadvantages of the algorithm could be investigated.

Beyond the identification of pathologies as such, the algorithm could be used for monitoring rehabilitation. This could be done by continuously monitoring the activity of, for example, a basketball player recovering from an injury. The system would give constant feedback on the player's condition and, if the player starts to show signs of injury, it would warn him to stop training.

BIBLIOGRAPHY

[1] J. M. Hausdorff, P. L. Purdon, C.-K. Peng, Z. V. I. Ladin, J. Y. Wei, and A. L. Goldberger, "Fractal dynamics of human gait: stability of long-range correlations in stride interval fluctuations," *Journal of applied physiology*, vol. 80, no. 5, pp. 1448–1457, 1996. DOI: 10.1152/jappl.1996.80.5.1448.

[2] D. A. Winter, "Biomechanical Motor Patterns in Normal Walking," *Journal of Motor Behavior*, vol. 15, no. 4, pp. 302–330, 1983. DOI: 10.1080/00222895.1983.10735302.

[3] K. Flint, K. Kennedy, S. V. Arnold, J. A. Dodson, S. Cresci, and K. P. Alexander, "Slow gait speed and cardiac rehabilitation participation in older adults after acute myocardial infarction," *Journal of the American Heart Association*, vol. 7, no. 5, 2018. DOI: 10.1161/JAHA.117.008296.

[4] N. Khouri and E. Desailly, "Rectus femoris transfer in multilevel surgery: Technical details and gait outcome assessment in cerebral palsy patients," *Orthopaedics and Traumatology: Surgery and Research*, vol. 99, no. 3, pp. 333–340, 2013. DOI: 10.1016/j.otsr.2012.10.017.

[5] H. Yu, J. Riskowski, R. Brower, and T. Sarkodie-Gyan, "Gait variability while walking with three different speeds," *2009 IEEE International Conference on Rehabilitation Robotics, ICORR 2009*, pp. 823–827, 2009. DOI: 10.1109/ICORR.2009.5209486.

[6] J. D. Schaafsma, N. Giladi, Y. Balash, A. L. Bartels, T. Gurevich, and J. M. Hausdorff, "Gait dynamics in Parkinson's disease: relationship to Parkinsonian features, falls and response to levodopa.," *Journal of the neurological sciences*, vol. 212, no. 1-2, pp. 47–53, 2003. DOI: 10.1016/S0022-510X(03)00104-7.

[7] J. M. Hausdorff, D. A. Rios, and H. K. Edelberg, "Gait variability and fall risk in community-living older adults: A 1-year prospective study," *Archives of Physical Medicine and Rehabilitation*, vol. 82, no. 8, pp. 1050–1056, 2001. DOI: 10.1053/apmr.2001.24893.

[8] J. Perry and J. M. Burnfield, "Gait analysis. Normal and pathological function 2nd ed," *California: Slack*, 2010. DOI: 10.1001/jama.2010.1210.

[9] A. Muro-de-la-Herran, B. García-Zapirain, and A. Méndez-Zorrilla, "Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications," *Sensors (Switzerland)*, vol. 14, no. 2, pp. 3362–3394, 2014. DOI: 10.3390/s140203362.

[10] M. Gabel, R. Gilad-Bachrach, E. Renshaw, and A. Schuster, "Full body gait analysis with Kinect," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2012, pp. 1964–1967. DOI: 10.1109/EMBC.2012.6346340.

[11] C. Kirtley, "Clinical Gait Analysis," *Clinical Gait Analysis*, no. September, pp. 35–40, 2006. DOI: 10.1016/B978-0-443-10009-3.X5001-2.

[12] D. H. Sutherland, "The evolution of clinical gait analysis: Part II kinematics," *Gait and Posture*, vol. 16, no. 2, pp. 159–179, 2002. DOI: 10.1016/S0966-6362(02)00004-8.

[13] L. Wang, T. Tan, H. Ning, and W. Hu, "Silhouette Analysis-Based Gait Recognition for Human Identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1505–1518, 2003. DOI: 10.1109/TPAMI.2003.1251144.

[14] J. Suutala and J. Röning, "Towards the adaptive identification of walkers: Automated feature selection of footsteps using distinction-sensitive lvq," *Proceedings of International Workshop on Processing Sensory Information for Proactive Systems (PSIPS 2004)*, pp. 1–7, 2004. DOI: 10.1.1.122.12.

[15] P. Fernandez-Lopez, J. Liu-Jimenez, C. Sanchez-Redondo, and R. Sanchez-Reillo, "Gait recognition using smartphone," *Proceedings - International Carnahan Conference on Security Technology*, vol. 0, 2016. DOI: 10.1109/CCST.2016.7815698.

[16] P. Bours and A. Evensen, "The Shakespeare experiment: Preliminary results for the recognition of a person based on the sound of walking," *Proceedings - International Carnahan Conference on Security Technology*, vol. 2017-Octob, pp. 1–6, 2017. DOI: 10.1109/CCST.2017.8167839.

[17] K. O'Donovan, R. Kamnik, D. O'Keeffe, and G. ÓLaighin, "An inertial and magnetic sensor based technique for joint angle measurement," *Journal of biomechanics*, vol. 40, pp. 2604–2611, 2007. DOI: 10.1016/j.jbiomech.2006.12.010.

[18] J. Tomaro and R. G. Burdett, "The effects of foot orthotics on the EMG activity of selected leg muscles during gait," *Journal of Orthopaedic and Sports Physical Therapy*, vol. 18, no. 4, pp. 532–536, 1993. DOI: 10.2519/jospt.1993.18.4.532.

[19] D. C. Kerrigan, M. K. Todd, U. Della Croce, L. A. Lipsitz, and J. J. Collins, "Biomechanical gait alterations independent of speed in the healthy elderly: evidence for specific limiting impairments.," eng, *Archives of physical medicine and rehabilitation*, vol. 79, no. 3, pp. 317–322, Mar. 1998. DOI: 10.1016/s0003-9993(98)90013-2.

[20] H. Stolze *et al.*, "Typical features of cerebellar ataxic gait," *Journal of Neurology Neurosurgery and Psychiatry*, vol. 73, no. 3, pp. 310–312, 2002. DOI: 10.1136/jnnp.73.3.310.

[21] P. Friedman, D. Richmond, and J. Baskett, "A prospective trial of serial gait speed as a measure of rehabilitation in the elderly," *Age and Ageing*, vol. 17, no. 4, pp. 227–235, 1988. DOI: 10.1093/ageing/17.4.227.

[22] R. Dickstein, A. Dunsky, and E. Marcovitz, "Motor Imagery for Gait Rehabilitation in Post-Stroke Hemiparesis," *Physical Therapy*, vol. 84, no. 12, pp. 1167–1177, 2004. DOI: 10.1093/ptj/84.12.1167.

[23] F. Ferrarello *et al.*, "Tools for Observational Gait Analysis in Patients With Stroke: A Systematic Review," *Physical Therapy*, vol. 93, no. 12, pp. 1673–1685, 2013. DOI: 10.2522/ptj.20120344.

[24] N. M. Fisher, S. C. White, H. J. Yack, R. J. Smolinski, and D. R. Pendergast, "Muscle function and gait in patients with knee osteoarthritis before and after muscle rehabilitation," *Disability and Rehabilitation*, vol. 19, no. 2, pp. 47–55, 1997. DOI: 10.3109/09638289709166827.

[25] J. Liang, S. Lang, Y. Zheng, Y. Wang, and H. Chen, "The effect of anti-gravity treadmill training for knee osteoarthritis rehabilitation on joint pain, gait, and EMG," *Medicine*, vol. 18, no. December 2018, 2019. DOI: 10.1097/MD.0000000000015386.

[26] K. Aminian *et al.*, "Evaluation of an ambulatory system for gait analysis in hip osteoarthritis and after total hip replacement," *Gait & Posture*, vol. 20, no. 1, pp. 102–107, 2004. DOI: https://doi.org/10.1016/S0966-6362(03)00093-6.

[27] A. Esquenazi, "Gait analysis in lower-limb amputation and prosthetic rehabilitation," *Physical medicine and rehabilitation clinics of North America*, vol. 25, no. 1, pp. 153–167, 2014. DOI: 10.1016/j.pmr.2013.09.006.

[28] W. Zeng, F. Liu, Q. Wang, Y. Wang, L. Ma, and Y. Zhang, "Parkinson's disease classification using gait analysis via deterministic learning," *Neuroscience Letters*, vol. 633, pp. 268–278, 2016. DOI: https://doi.org/10.1016/j.neulet.2016.09.043.

[29] Y. Cedervall, K. Halvorsen, and A. C. Åberg, "A longitudinal study of gait function and characteristics of gait disturbance in individuals with Alzheimer's disease," *Gait & Posture*, vol. 39, no. 4, pp. 1022–1027, 2014. DOI: https://doi.org/10.1016/j.gaitpost.2013.12.026.

[30] M. L. Callisaya, L. Blizzard, M. D. Schmidt, J. L. McGinley, and V. K. Srikanth, "Ageing and gait variability—a population-based study of older people," *Age and Ageing*, vol. 39, no. 2, pp. 191–197, 2010. DOI: 10.1093/ageing/afp250.

[31] H. Stolze, J. P. Kuhtz-Buschbeck, H. Drücke, K. Jöhnk, M. Illert, and G. Deuschl, "Comparative analysis of the gait disorder of normal pressure hydrocephalus and Parkinson's disease.," eng, *Journal of neurology, neurosurgery, and psychiatry*, vol. 70, no. 3, pp. 289–297, Mar. 2001. DOI: 10.1136/jnnp.70.3.289.

[32] V. Cimolin and M. Galli, "Summary measures for clinical gait analysis: A literature review," *Gait and Posture*, vol. 39, no. 4, pp. 1005–1010, 2014. DOI: 10.1016/j.gaitpost.2014.02.001.

[33] C. Senanayake and S. M. Senanayake, "Human assisted tools for gait analysis and intelligent gait phase detection," *2009 Innovative Technologies in Intelligent Systems and Industrial Applications, CITISIA 2009*, no. July, pp. 230–235, 2009. DOI: 10.1109/CITISIA.2009.5224208.

[34] C. M. Senanayake and S. M. Senanayake, "Computational intelligent gait-phase detection system to identify pathological gait," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 5, pp. 1173–1179, 2010. DOI: 10.1109/TITB.2010.2058813.

[35] H. Yu, M. Alaqtash, E. Spier, and T. Sarkodie-Gyan, "Analysis of muscle activity during gait cycle using fuzzy rule-based reasoning," *Measurement*, vol. 43, no. 9, pp. 1106–1114, 2010. DOI: https://doi.org/10.1016/j.measurement.2010.04.010.

[36] L. M. Schutte, U. Narayanan, J. L. Stout, P. Selber, J. R. Gage, and M. H. Schwartz, "An index for quantifying deviations from normal gait," *Gait & Posture*, vol. 11, no. 1, pp. 25–31, 2000. DOI: https://doi.org/10.1016/S0966-6362(99)00047-8.

[37] M. H. Schwartz and A. Rozumalski, "The gait deviation index: A new comprehensive index of gait pathology," *Gait & Posture*, vol. 28, no. 3, pp. 351–357, 2008. DOI: https://doi.org/10.1016/j.gaitpost.2008.05.001.

[38] S. Bei, Z. Zhen, Z. Xing, L. Taocheng, and L. Qin, "Movement Disorder Detection via Adaptively Fused Gait Analysis Based on Kinect Sensors," *IEEE Sensors Journal*, vol. 18, no. 17, pp. 7305–7314, 2018. DOI: 10.1109/JSEN.2018.2839732.

[39] G. Guo, K. Guffey, W. Chen, and P. Pergami, "Classification of Normal and Pathological Gait in Young Children Based on Foot Pressure Data," *Neuroinformatics*, vol. 15, no. 1, pp. 13–24, 2017. DOI: 10.1007/s12021-016-9313-x.

[40] E. Dolatabadi, B. Taati, and A. Mihailidis, "An automated classification of pathological gait using unobtrusive sensing technology," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 12, pp. 2336–2346, 2017. DOI: 10.1109/TNSRE.2017.2736939.

[41] A. K. Seifert, A. M. Zoubir, and M. G. Amin, "Detection of gait asymmetry using indoor doppler radar," *2019 IEEE Radar Conference, RadarConf 2019*, 2019. DOI: 10.1109/RADAR.2019.8835611. arXiv: 1902.09977.

[42] W. Teufl *et al.*, "Automated detection and explainability of pathological gait patterns using a one-class support vector machine trained on inertial measurement unit based gait data," *Clinical Biomechanics*, vol. 89, no. August, p. 105 452, 2021. DOI: 10.1016/j.clinbiomech.2021.105452.

[43] M. Khokhlova, C. Migniot, A. Morozov, O. Sushkova, and A. Dipanda, "Normal and pathological gait classification LSTM model," *Artificial Intelligence in Medicine*, vol. 94, no. December 2018, pp. 54–66, 2019. DOI: 10.1016/j.artmed.2018.12.007.

[44] S. Jain and A. Nandy, "Human Gait Abnormality Detection Using Low Cost Sensor Technology," S. K. Singh, P. Roy, B. Raman, and P. Nagabhushan, Eds., pp. 330–340, 2021.

[45] J. Gao, P. Gu, Q. Ren, J. Zhang, and X. Song, "Abnormal Gait Recognition Algorithm Based on LSTM-CNN Fusion Network," *IEEE Access*, vol. 7, pp. 163 180–163 190, 2019. DOI: 10.1109/ACCESS.2019.2950254.

[46] S. Yin, C. Chen, H. Zhu, X. Wang, and W. Chen, "Neural networks for pathological gait classification using wearable motion sensors," *BioCAS 2019 - Biomedical Circuits and Systems Conference, Proceedings*, pp. 2019–2022, 2019. DOI: 10.1109/BIOCAS.2019.8919096.

[47] S. Potluri, S. Ravuri, C. DIedrich, and L. Schega, "Deep Learning based Gait Abnormality Detection using Wearable Sensor System," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pp. 3613–3619, 2019. DOI: 10.1109/EMBC.2019.8856454.

[48] P. Albuquerque, T. T. Verlekar, P. L. Correia, and L. D. Soares, "A Spatiotemporal Deep Learning Approach for Automatic Pathological Gait Classification," *Sensors*, vol. 21, no. 18, 2021. DOI: 10.3390/s21186202.

[49] C. Gomez, *What is machine learning?* 2020. [Online]. Available: https://medium.com/@1154_75881/what-is-machine-learning-fb3821129ff0 (visited on 09/30/2021).

[50] M. Shouman, T. Turner, and R. Stocker, "Using decision tree for diagnosing heart disease patients," *Conferences in Research and Practice in Information Technology Series*, vol. 121, pp. 23–30, 2010. DOI: 10.5555/2483628.2483633.

[51] S. Sahoo, A. Subudhi, M. Dash, and S. Sabut, "Automatic Classification of Cardiac Arrhythmias Based on Hybrid Features and Decision Tree Algorithm," *International Journal of Automation and Computing*, vol. 17, no. 4, pp. 551–561, 2020. DOI: 10.1007/s11633-019-1219-2.

[52] H. Lu and X. Ma, "Hybrid decision tree-based machine learning models for short-term water quality prediction," *Chemosphere*, vol. 249, p. 126 169, 2020. DOI: 10.1016/j.chemosphere.2020.126169.

[53] C. Wang, A. Wang, J. Xu, Q. Wang, and F. Zhou, "Outsourced privacy-preserving decision tree classification service over encrypted data," *Journal of Information Security and Applications*, vol. 53, p. 102 517, 2020. DOI: 10.1016/j.jisa.2020.102517.

[54] J. Yan, Z. Zhang, K. Lin, F. Yang, and X. Luo, "A hybrid scheme-based one-vs-all decision trees for multi-class classification tasks," *Knowledge-Based Systems*, vol. 198, p. 105 922, 2020. DOI: 10.1016/j.knosys.2020.105922.

[55] M. Dagdoug, C. Goga, and D. Haziza, "Imputation procedures in surveys using nonparametric and machine learning methods: an empirical comparison," *arXiv*, no. 1, pp. 1–52, 2020. arXiv: 2007.06298.

[56] S. Zhang, "Cost-sensitive KNN classification," *Neurocomputing*, vol. 391, pp. 234–242, 2020. DOI: 10.1016/j.neucom.2018.11.101.

[57] A. Ajanki, *Example of k-nearest neighbour classificationnb*. [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#/media/File:KnnClassification.svg (visited on 02/25/2021).

[58] E. F. Jr and J. . L. . Hodges, "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties," *International Statistical Review / Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989. DOI: 10.2307/1403797.

[59] S. A. Dudani, "The Distance-Weighted k-Nearest-Neighbor Rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. Pp. 325–327, 1976. DOI: 10.1109/TSMC.1976.5408784.

[60] S. S. Skiena, *The data science design manual*. 2017, pp. 1–445.

[61] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, 2020. DOI: 10.1016/j.neucom.2019.10.118.

[62] B. Stecanella, *An Introduction to Support Vector Machines (SVM)*, 2017. [Online]. Available: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/ (visited on 02/25/2021).

[63] A. Zendehboudi, M. A. Baseer, and R. Saidur, "Application of support vector machine models for forecasting solar and wind energy resources: A review," *Journal of Cleaner Production*, vol. 199, pp. 272–285, 2018. DOI: 10.1016/j.jclepro.2018.07.164.

[64] M. T. B. Richhariya, *EEG signal classification using universum support vector machine*. Elsevier Ltd, 2018, vol. 106, pp. 169–182. DOI: 10.1016/j.eswa.2018.03.053.

[65] H. Al-Hadeethi, S. Abdulla, M. Diykh, R. C. Deo, and J. H. Green, "Adaptive boost LS-SVM classification approach for time-series signal classification in epileptic seizure diagnosis applications," *Expert Systems with Applications*, vol. 161, p. 113 676, 2020. DOI: 10.1016/j.eswa.2020.113676.

[66] R. E. Neapolitan and R. E. Neapolitan, *Neural Networks and Deep Learning*. 2018, pp. 389–411. DOI: 10.1201/b22400-15.

[67] W. De Mulder, S. Bethard, and M. F. Moens, "A survey on the application of recurrent neural networks to statistical language modeling," *Computer Speech and Language*, vol. 30, no. 1, pp. 61–98, 2015. DOI: 10.1016/j.csl.2014.09.005.

[68] Michael Phi, *Illustrated Guide to LSTM's and GRU's: A step by step explanation*, 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (visited on).

[69] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," pp. 1–9, 2014. arXiv: 1412.3555.

[70] P. Ratan, *What is the Convolutional Neural Network Architecture?* 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/ (visited on).

[71] J. Stalfort, *Hyperparameter tuning using Grid Search and Random Search: A Conceptual Guide*, 2019. [Online]. Available: https://medium.com/@jackstalfort/hyperparameter-tuning-using-grid-search-and-random-search-f8750a464b35 (visited on).

[72] P. Fernandez-Lopez, J. Sanchez-Casanova, P. Tirado-Martin, and J. Liu-Jimenez, "Optimizing resources on smartphone gait recognition," *IEEE International Joint Conference on Biometrics, IJCB 2017*, vol. 2018-Janua, pp. 1–6, 2018. DOI: 10.1109/BTAS.2017.8272679.

[73] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," *Bulletin of the Technical Committee on Data Engineering*, vol. 24, no. 4, pp. 1–56, 2001. DOI: 10.1.1.98.8661.

[74] D. Cousineau and S. Chartier, "Outliers detection and treatment: a review.," *International Journal of Psychological Research*, vol. 3, no. 1, pp. 58–67, 2010. DOI: 10.21500/20112084.844.

[75] P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "An Anomaly-Based Intrusion Detection System for the Smart Grid Based on CART Decision Tree," *2018 Global Information Infrastructure and Networking Symposium, GIIS 2018*, 2019. DOI: 10.1109/GIIS.2018.8635743.

[76] C. L. Liu, C. H. Lee, and P. M. Lin, "A fall detection system using k-nearest neighbor classifier," *Expert Systems with Applications*, vol. 37, no. 10, pp. 7174–7181, 2010. DOI: 10.1016/j.eswa.2010.04.014.

[77] R. Palaniappan, K. Sundaraj, S. Sundaraj, N. Huliraj, and S. S. Revadi, "A telemedicine tool to detect pulmonary pathology using computerized pulmonary acoustic signal analysis," *Applied Soft Computing Journal*, vol. 37, pp. 952–959, 2015. DOI: 10.1016/j.asoc.2015.05.031.

[78] N. Sáenz-Lechón, J. I. Godino-Llorente, V. Osma-Ruiz, and P. Gómez-Vilda, "Methodological issues in the development of automatic systems for voice pathology detection," *Biomedical Signal Processing and Control*, vol. 1, no. 2, pp. 120–128, 2006. DOI: 10.1016/j.bspc.2006.06.003.

[79] Y. Chen *et al.*, "A Feature-Free 30-Disease Pathological Brain Detection System by Linear Regression Classifier," *CNS & Neurological Disorders - Drug Targets*, vol. 16, no. 1, pp. 5–10, 2017. DOI: 10.2174/1871527314666161124115531.

[80] X. Zhou *et al.*, "Detection of Pathological Brain in MRI Scanning Based on Wavelet-Entropy and Naive Bayes Classifier," F. Ortuño and I. Rojas, Eds., pp. 201–209, 2015. DOI: 10.1007/978-3-319-16483-0_20.

[81] Y. Unal and H. E. Kocer, "Diagnosis of pathology on the vertebral column with backpropagation and Naive Bayes classifier," *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering, TAEECE 2013*, pp. 276–279, 2013. DOI: 10.1109/TAEECE.2013.6557285.

[82] F. T. Al-Dhief *et al.*, "Voice Pathology Detection Using Machine Learning Technique," *2020 IEEE 5th International Symposium on Telecommunication Technologies, ISTT 2020 - Proceedings*, pp. 99–104, 2020. DOI: 10.1109/ISTT50966.2020.9279346.

[83] M. Markaki and Y. Stylianou, "Voice pathology detection and discrimination based on modulation spectral features," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 7, pp. 1938–1948, 2011. DOI: 10.1109/TASL.2010.2104141.

[84] P.-h. Lai, M. Trayer, S. Ramakrishna, and Y. Li, "Database Establishment for Machine Learning in NILM," *1st International Workshop on Non-Intrusive Load Monitoring*, no. May, pp. 12–14, 2012. [Online]. Available: http://nilmworkshop.org/2012/abstracts/lai_Samsung_NILM2012_abstract.pdf.

[85] "Dynamic time warping," in *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 69–84. DOI: 10.1007/978-3-540-74048-3_4.

[86] Technaid, *Technaid S.L.* [Online]. Available: https://www.technaid.com/.

[87] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, pp. 35–45, 1960. DOI: 10.1115/1.3662552.

[88] H. B. Menz *et al.*, "Foot pain and mobility limitations in older adults: The framingham foot study," *Journals of Gerontology - Series A Biological Sciences and Medical Sciences*, vol. 68, no. 10, pp. 1281–1285, 2013. DOI: 10.1093/gerona/glt048.

[89] J. L. Taylor, L. J. Parker, S. L. Szanton, and R. J. Thorpe, "The association of pain, race and slow gait speed in older adults," *Geriatric Nursing*, vol. 39, no. 5, pp. 580–583, 2018. DOI: 10.1016/j.gerinurse.2018.04.004.

[90] S. Homes, "Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes," pp. 1–26, 2019. DOI: 10.3390/s19092047.

[91] O. Karan, C. Bayraktar, H. Gümüşkaya, and B. Karlik, "Diagnosing diabetes using neural networks on small mobile devices," *Expert Systems with Applications*, vol. 39, no. 1, pp. 54–60, 2012. DOI: 10.1016/j.eswa.2011.06.046.

[92] Z. H. Zhou, Y. Jiang, Y. B. Yang, and S. F. Chen, "Lung cancer cell identification based on artificial neural network ensembles," *Artificial Intelligence in Medicine*, vol. 24, no. 1, pp. 25–36, 2002. DOI: 10.1016/S0933-3657(01)00094-X.

[93] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0.

[94] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[95] F. Chollet, *Keras*, https://keras.io, 2015.

[96] C. Zhang, G. Wang, J. Zhao, P. Gao, J. Lin, and H. Yang, "Patient Specific Ecg Classification Based on Recurrent Neural Networks and Clustering Technique," *2017 13th IASTED International Conference on Biomedical Engineering (BioMed)*, no. 7, pp. 63–67, 2017. DOI: 10.2316/P.2017.852-029.

[97] M. Cheng, W. J. Sori, F. Jiang, A. Khan, and S. Liu, "Recurrent Neural Network Based Classification of ECG Signal Features for Obstruction of Sleep Apnea Detection," *Proceedings - 2017 IEEE International Conference on Computational Science and Engineering and IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, CSE and EUC 2017*, vol. 2, pp. 199–202, 2017. DOI: 10.1109/CSE-EUC.2017.220.

[98] P. D. Welch, "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms," *J. Am. Chem. Soc.*, vol. 113, no. ii, pp. 1047–1049, 1991. DOI: 10.1109/TAU.1967.1161901.

[99] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-Augu, pp. 785–794, 2016. DOI: 10.1145/2939672.2939785. arXiv: 1603.02754.

[100] L. E. Raileanu and K. Stoffel, "Theoretical comparison between the Gini Index and Information Gain criteria," *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, 2004. DOI: 10.1023/B:AMAI.0000018580.96245.c6.

[101] R. G. Mantovani, R. Cerri, J. Vanschoren, T. Horváth, S. B. Junior, and A. C. de Carvalho, "An empirical study on hyperparameter tuning of decision trees," *arXiv*, 2018. arXiv: 1812.02207.

[102] S. Yıldırım, *Hyperparameter Tuning for Support Vector Machines — C and Gamma Parameters*, 2020. [Online]. Available: https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167 (visited on 03/27/2021).

[103] A. Janecek, W. N. W. Gansterer, M. Demel, and G. Ecker, "On the Relationship Between Feature Selection and Classification Accuracy.," *Fsdm*, vol. 4, pp. 90–105, 2008. DOI: 10.5555/3053814.3053821.

[104] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," *Proceedings of 2014 Science and Information Conference, SAI 2014*, pp. 372–378, 2014. DOI: 10.1109/SAI.2014.6918213.

[105] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research 3*, vol. 3, pp. 1157–1182, 2003. DOI: 10.1016/j.aca.2011.07.027.

[106] Y. Lin, "Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns," *Technometrics*, vol. 44, no. 2, pp. 196–197, 2002. DOI: 10.1198/tech.2002.s721.

[107] A. Tharwat, "Principal component analysis - a tutorial," *International Journal of Applied Pattern Recognition*, vol. 3, no. 3, p. 197, 2016. DOI: 10.1504/ijapr.2016.079733.

[108] J. M. Hancock, M. J. Zvelebil, and N. Cristianini, "Fisher Discriminant Analysis (Linear Discriminant Analysis)," *Dictionary of Bioinformatics and Computational Biology*, pp. 1–6, 2004. DOI: 10.1002/9780471650126.dob0238.pub2.

[109] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," *AI Communications*, vol. 30, no. 2, pp. 169–190, 2017. DOI: 10.3233/AIC-170729.

[110] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers and Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014. DOI: 10.1016/j.compeleceng.2013.11.024.

[111] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," vol. 58, no. 1, pp. 267–288, 2016. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[112] M. Kraus, S. Feuerriegel, and A. Oztekin, "Deep learning in business analytics and operations research: Models, applications and managerial implications," *European Journal of Operational Research*, vol. 281, Sep. 2019. DOI: 10.1016/j.ejor.2019.09.018.