الجـــــــــــــــــــمـهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــعـليـم العـــــــالي و البـــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــامعة أبي بكـــر بـلـقـايد – تـلمســـــــــــــان –

Université Aboubakr Belkaïd – Tlemcen –
Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Automatique

Spécialité :Automatique et Informatique Industrielle

Par : BOUCHETA Mohamed Anes
BOUGUETAIB Zakaria

Sujet

Classification of kinematic data from a robotic upper limb rehabilitation orthosis

Soutenu publiquement, le 22 /09 /2024 , devant le jury composé de :

M.YAGOUBI Boumediene	MAA	Université de Tlemcen	Président
Mme.SAIDI Farah	MCB	Université de Tlemcen	Examinateur
M.Hadj Amine Abdelkader	Prof	Université de Tlemcen	Encadreur
M.MOULAY Khatir Ahmed Nassim	MCB	Université de Tlemcen	Co-Encadreur

Année universitaire : 2024/2023

Acknowledgements

I am deeply grateful to express my sincere appreciation to my advisor, Dr Amine, for his invaluable support and mentorship that played a pivotal role in guiding and shaping our research endeavours. I want to extend my heartfelt thanks to each member of the jury for their thoughtful and constructive feedback, which significantly enriched the quality of our work.

I would also like to thank everyone who generously provided the essential data that enabled us to conduct our experiments and meticulously develop our algorithms. Moreover, I am immensely thankful to my friends and family whose unwavering encouragement and belief in our work provided essential emotional and moral support throughout this journey.

Epigraph

"Knowledge is the most powerful tool to overcome ignorance and uncertainty." — *Al-Khwarizmi*

The Non-Plagiarism Charter

Plagiarism clarification

We "BOUCHETA Mohammed Anes" AND "Bouguetaib Zakaria" students of Abou Bekr Belkaid University

We hereby declare that this thesis is the result of our work and research. we have neither counterfeited, falsified, nor copied, in whole or in part, the work of others to present it as my own. All sources of information used (including paper, audiovisual, and digital media) and author citations have been properly acknowledged according to current standards.

We are fully aware that failing to cite a source or failing to cite it clearly and completely, constitutes plagiarism. We understand that plagiarism is considered a serious offence within the University and may be subject to severe penalties.

Fait à le
Signature

Summary

CHAPTER 1

Introduction

The human being is favoured by his ability to do a bunch of tasks that other species cannot do such as typing, writing and operating, and that's because of his upper limb functionality. However, he might lose it whenever something harmful occurs such as injury, stroke or accident leaving him handicapped or unable to enjoy this blessing [12], here is where rehabilitation comes in touch to regain it back, Robotic orthosis is one of the tools used in rehabilitation due to its potential to provide help, support, better guidance and training. Moreover, we collect kinematic data from these orthoses for us to obtain better results after analysing and classifying them in a proper way, preparing them to be studied medically, thus it will empower therapists to handle all different situations better. This thesis aims to classify kinematic data gathered from a robotic upper limb orthosis to create a path of the patient's progress according to their interaction with it during his rehabilitation. after finishing it, this thesis will have a clinical impact thus every patient will have a unique therapy, not a universal one for everyone of the same handicapped category.

By analysing the patient's patterns of movement precisely we will be able to create interventions that resonate with the challenges they need to overcome to achieve the maximum potential of recovering. This research is one step towards more advanced robotic orthosis with algorithms capable of real-time analysis, to be the base stone of adjusting the assistance levels, providing better feedback in a fraction of a second making it more personalized for each patient and transforming them into indispensable devices for physical therapy. In this study, we will dive into the complex net of the upper limb bio-mechanics help us to solve this riddle into better knowledge [23]. The first step in our journey will be "DATA collection" in which we will gather data from several participants performing various exercises using the robot and the result will be translated into the digital score to be deciphered, after that, harvested data will be analysed cleaned and filtered to a meaningful set of numbers, as a third step we will have to select a classifying algorithm after deep consideration of its powers and In this study, we will dive into the machine learning world to train these models using the results from the last step as a guide, next, we will have to evaluate the models according to the accuracy of their results to be im-

plemented later into real-time experience known as clinical integration either by using them to create patterns or study the progress aiming to help therapists in their job. This research was accomplished to make upper limb rehabilitation robotics smarter waiting for a fast and accurate classification system that translates patients' movements into real-time data for therapists to stand on making therapy sessions more valuable and efficient [16].

1 Arm Rehabilitation

Arm disability represents a significant challenge that threatens individuals' quality of life, impeding their ability to engage in hobbies, work activities, and daily tasks, and sometimes contributing to mental health issues. Despite advancements in medical treatments and technologies, this form of disability persists due to the high incidence of accidents, strokes, obesity, and mental disorders, which can vary depending on individual circumstances [12].

Researchers and therapists are diligently working to restore functionality and hope to those affected by arm disabilities, investing considerable time and effort into studying rehabilitation methods. Their goal is to improve outcomes and transform outcomes from improbable to promising. The history of arm rehabilitation dates back to the early 1800s, characterized by conventional practices such as massage, herbal remedies, and manipulations influenced by cultural beliefs of the time. The pivotal moment in limb rehabilitation occurred in the early 19Th century, coinciding with rapid technological and scientific advancements that enhanced our understanding of physiology, anatomy, and arm functionalities. This period also saw the emergence of specialized physiotherapy chairs designed to focus attention on rehabilitation efforts.

The 20th century, marked by global conflicts resulting in millions of injuries and disabilities, propelled the field of therapy into the spotlight, garnering both public and governmental support due to its critical role in post-war recovery efforts. This heightened awareness spurred advancements in prosthetic and the development of neurological rehabilitation techniques aimed at improving motor control and sensory deficits in daily activities and exercises.

Recent advancements have seen the integration of interactive therapies using augmented reality (AR) and virtual reality (VR), alongside robotic systems and personalized medicine tailored to individual genetics and conditions. Future efforts are increasingly focused on implementing next-generation technologies such as artificial intelligence (AI) to predict patient outcomes accurately. These technologies enable therapists to create personalized challenges using recent bio materials, aimed at preventing upper limb injuries and enhancing rehabilitation strategies [16].

Effective rehabilitation for stroke patients begins immediately after impairment is identified, focusing on achieving full range of motion in every joint of the limb. Therapeutic sessions commence cautiously, often starting with bedside exercises and progressing to various techniques tailored to individual needs.

2 Classic Arm Rehabilitation

Definition

Classical arm rehabilitation is applying traditional techniques to help patients obtain flexibility and arm functionality of their arm back, this term has held its meaning since the 19Th century before robotics, AR and VR existed. It consists of using a group of therapeutic methods that we can count some of them:

Classic rehabilitation methods

Physical therapy

Physical therapy plays a pivotal role in the rehabilitation process, encompassing traditional exercises and neuro-physiological approaches. This therapeutic method focuses on enhancing arm mobility and function through structured exercises.

Range of Motion Exercises: Range of motion exercises involve gradual movements designed to improve flexibility and mobility of the arm joints. These exercises aim to restore and maintain the full range of motion in the affected limb.

Strength Training: Strength training is essential for building muscle strength and endurance in specific muscles of the arm. These exercises are tailored to the individual's needs and progress, targeting muscles that may have weakened due to injury or disuse.

Stretching: Stretching exercises are employed to elongate and relax arm muscles and tendons. This helps prevent stiffness, improves flexibility, and reduces the risk of contractures.

Traditional exercises programs

Traditional exercise programs are crucial in preventing complications associated with immobilization and promoting recovery. These programs are typically categorized into two types:

Passive Exercises: Carried out by a caregiver or therapist, passive exercises involve moving the patient's upper limb joints through their full range of motion. This helps maintain joint mobility and prevent stiffness.

Active Exercises: In active exercises, the patient independently engages their muscles to move their joints. Initially starting with smaller movements, the intensity and range of these exercises gradually increase over time. Progression from simple to complex exercises is essential to challenge and strengthen the muscles effectively.

Effective rehabilitation programs integrate both passive and active exercises, tailored to the individual's rehabilitation goals and capabilities. Consistent and progressive engagement in these exercises is key to achieving optimal recovery and functional independence.

Neuro-physiological approaches

Neuro-physiological therapy is the use of neuro-physiology principles in the rehabilitation of people with neurological conditions affecting their abilities to move, this approach is based on knowledge about the nervous system functions and it aims to improve recovery and enhance limb functionality for individuals with neurological impairments. This approach itself uses several treatment methods such as stretching, cryo-therapy, vibration and joint compression.

Functional electrical stimulation (FES)

This method is based on the electrical stimulation of the nerve using some electrodes attached to his limb as we can see in figure 1.1, causing it to contract although it's painful and causes muscle fatigue, it helps improve muscle strength, control movements and the maintenance of the posture. It may also reduce time spent by therapists and it's considered to be much better than normal orthotic and assistive devices, in this method the patient uses his limb without an assist.

Electromyographic biofeedback

EMG biofeedback is a method in which we use myoelectrical signals taken from the muscle converted into visual and auditory signals as shown in figure 1.2. Using electrodes attached to the limb to generate a new feedback system that the user can identify and react in parallel with.

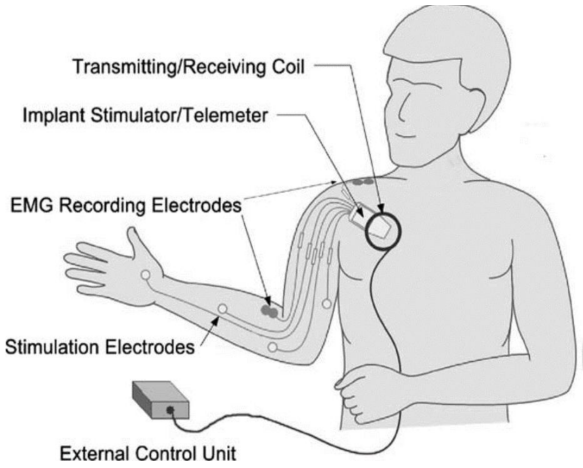


Fig. 1.1 Schematic representation of the CWRU/VA 12-channel implanted upper extremity neuroprosthesis with myoelectric control capability (P. Hunter Peckham and Knutson 2005)

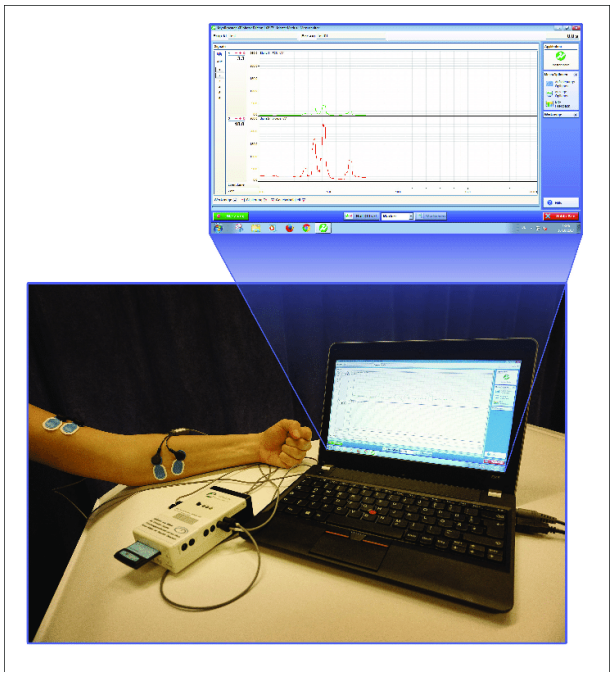


Fig. 1.2 Surface EMG biofeedback set-up with the TeleMyo system (Noraxon, United States) and a screenshot of the TeleMyo-Software simultaneously recording two EMG signals, represented by colour-coded graphs.

BCI

Brain-computer interfaces or "BCI" are a direct communication pathway between the brain's electrical activity and an external device enables the patients BCI to enable patients to control prosthetic devices or robotic orthosis directly through brain signals as shown in figure 1.3, thus improving motor function and independence.

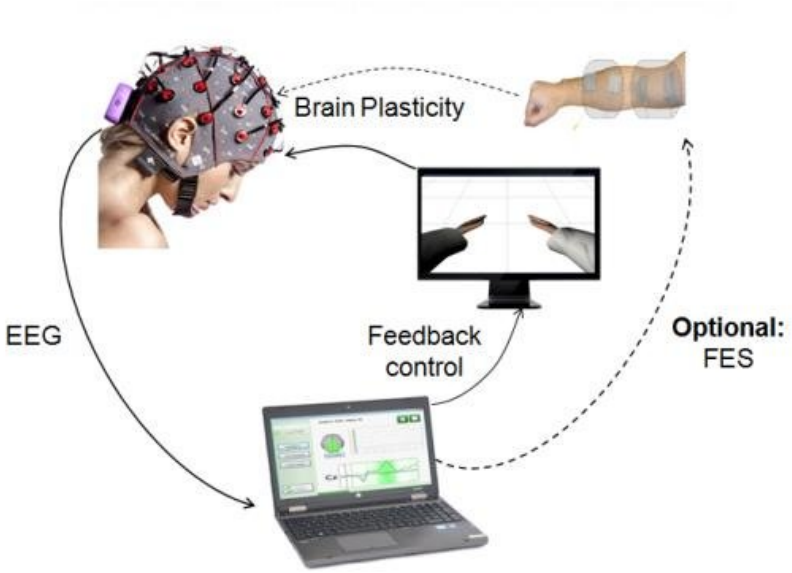


Fig. 1.3 Oversimplified BCI system [20]

Cell therapy

Biotherapy or cell therapy is the process in which we implant viable cells to achieve a medicinal effect, as shown in Figure 1.4.

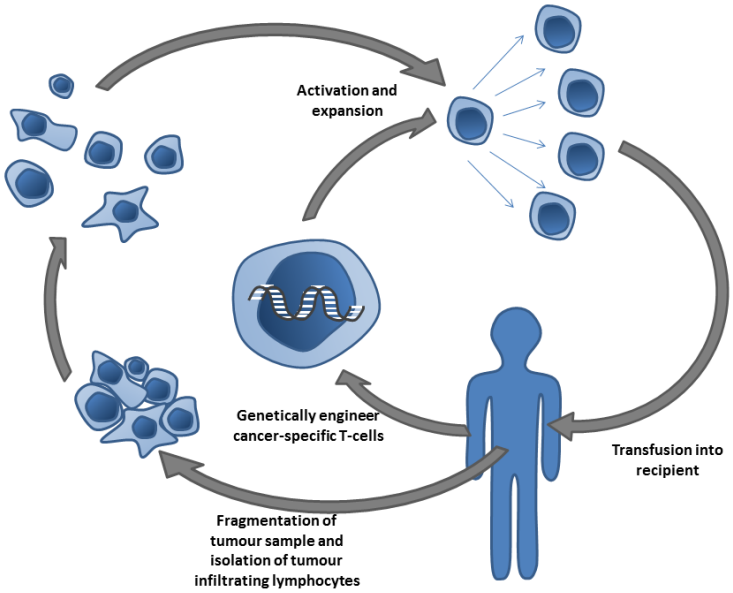


Fig. 1.4 Adoptive T-cell therapy. Cancer-specific T-cells can be obtained by fragmentation and isolation of tumour-infiltrating lymphocytes, or by genetically engineering cells from peripheral blood. The cells are activated and grown before transfusion into the recipient (tumour bearer).

2.1 Advantages of Classic Arm Rehabilitation

As experts nowadays have two choices to make — whether to choose classic or enhanced (technological) methods for their patients' rehabilitation journey — there is a growing preference for classic methods due to the following advantages:

Accessibility

Most of these techniques do not require complex machinery, making them accessible. They are affordable with low-cost maintenance and have a low rate of breakdown.

Proven Effectiveness

The adage "if it ain't broke, don't fix it" holds true. Classical rehabilitation therapy has been utilized since the late 19th century and continues to be employed because it consistently demonstrates effectiveness at minimal costs.

Versatility

Standard methods can be adapted to accommodate a wide range of disabilities, tailoring treatments to meet individual patient needs regardless of the cause of their impairment.

Cost Effectiveness

Compared to robot-assisted rehabilitation, classical methods are more cost-effective. The complex technology and skilled manpower required for robot-assisted therapy contribute to its higher expenses, whether for acquisition, maintenance, or operational costs.

Therapist-Patient Interaction

A crucial aspect of successful rehabilitation is the interaction between therapist and patient. Therapists not only apply their techniques but also provide motivation and emotional support, creating a comfortable and encouraging environment for recovery. This human touch is often lacking in robot-assisted rehabilitation, where interactions primarily involve mechanical movements without emotional engagement.

2.2 Shortcomings of Classic Arm Rehabilitation

Although classic rehabilitation therapy is often relied upon, it has several disadvantages and shortcomings that prevent it from being universally ideal.

Time Consuming

In today's fast-paced society, time is a precious resource, and classical rehabilitation methods are often perceived as time-consuming compared to modern techniques that may offer faster results.

Limited Utilization

While effective in many cases, classical rehabilitation techniques may not be suitable for all conditions. Injuries such as Traumatic Brain Injury (TBI) or Spinal Cord Injury (SCI) require specialized approaches that classic methods may not adequately address.

Dependency on Therapist Skills

The success of classical rehabilitation heavily relies on the expertise of the therapist, including their choice of devices and tools, as well as their proficiency in their application. Variability in therapist skill levels can impact treatment outcomes.

Stagnation

Patients may experience a plateau in their progress, commonly referred to as "hitting the wall" This stagnation occurs when further improvement becomes challenging or progresses at a slower rate, despite ongoing rehabilitation efforts.

Inexact Assessment

Classical rehabilitation often relies on subjective assessments rather than objective measurements. Assessing progress based solely on visual observations or patient-reported feelings can lead to inaccurate evaluations of rehabilitation outcomes. In a data-driven world, precise measurement through objective metrics is crucial for accurately tracking patient progress and adjusting treatment plans accordingly.

3 Robot-assisted Arm Rehabilitation

With the advancement of technology and the increasing demand for efficient solutions, robotics has made significant strides in various domains, including rehabilitation. Robotics offers a promising avenue for enhancing therapeutic outcomes by providing precise and consistent assistance that complements human efforts.

In 1989, the field of robot-assisted rehabilitation saw a significant milestone with the introduction of the first commercial rehabilitation robot developed by MIT-MAUS. This marked the beginning of integrating robotics into clinical settings, where it was first clinically tested in 1994 [**?**]. Rehabilitation robots are specialized devices designed to assist therapists during therapy sessions, offering both physical support and valuable data through integrated sensors. These sensors capture kinematic and time-based data, enabling therapists to assess patients' progress objectively and tailor treatments accordingly.

Several biomedical companies, such as "HACOMA" and "LIFEaWARD," have developed and marketed robot-based devices for clinical use globally. These devices vary in their applications based on the type of injuries they address and the specific limb targeted. For the purpose of this thesis, which focuses on upper limb rehabilitation, particular attention is given to the "ARMEO" series developed by HACOMA.

3.1 ARMEO Robotic Series

The ARMEO series, developed by HACOMA, represents a revolutionary approach to arm rehabilitation for individuals recovering from neurological and musculoskeletal disorders. It in-

tegrates virtual reality environments with robotic assistance to enhance patient strength and mobility significantly. The ARMEO robot consists of an exoskeleton equipped with artificial joints and sensors worn by the patient, which interface with sophisticated computer software.

These sensors within the exoskeleton precisely track the patient's movements, translating them into real-time data that therapists can analyze and use to assess progress accurately. This data-driven approach not only provides quantitative insights into the patient's rehabilitation journey but also allows therapists to make informed decisions based on objective metrics.

The ARMEO series is distinguished by its ability to offer personalized rehabilitation experiences tailored to each patient's needs. By combining robotic precision with virtual reality immersion, it creates an engaging and effective therapeutic environment aimed at optimizing recovery outcomes.

Armeo series:

Armeo series is a combination of different robot orthoses under the same name but different structures, including:
ArmeoSpring: As its name shows and as shown in Figure 2.3. "Armeo spring" is an arm and hand adjustable skeleton with joints connected through springs to facilitate the rehabilitation manoeuvre and assist the patient's arm and hand, in combination with virtual reality, this robotic arm uses task-specific and self-initiated skills to improve the early upper limb recovery[6].
ArmeoPower: Armeo power device is a robot technology combined with a virtual reality integration used to repair lost gestural functions. It's a combination of joints motorized by electric motors to assist the patient. There are installed sensors which detect if the person can do the requested gestural function and it organizes the needed exercise according to the needs of patients with the developed technology. While the lost functions are being enhanced, the support given by the device is reduced. In this way, a successful rehab process occurs.
ArmeoSenso: What makes this device special is that it can be used without a therapist's assistance, it's not made of joints or a skeleton, it's just a bunch of sensors that you should attach to special spots on your arm, it focuses on sensory retraining and motor control, incorporating haptic feedback and virtual reality.

3.2 Kinematic data

3.2.1 Definition

Kinematics studies motion and the forces required to produce it. This includes the different forces at work during the movement of a single part of the body and more complex movements such as running and climbing but in our case is rehabilitation, we can categorize our data obtained from robotic orthosis into kinematic measurements such as:

Joints angels and movements

One of the kinematic measurements we obtain during the use of robotic-assisted rehabilitation is the number or interval of angles that the patient's joint could attend which determines his flexibility and range of motion.

Trajectories

By following the patient's arm we can obtain coordinates that we can use to draw a trajectory map, which aids in determining the accuracy, smoothness of the movements and the time of completion of the task.

Forces and torque

Regaining strength will require exerting some effort in the exercises presented to the patients, that effort can be a force or torque picked up by the sensors embedded in the robot.

Progress tracking

Collecting data from each session helps us build a model that can predict the patients' overall performances of the patients, and providing patients with evidence of their improvements will have a beneficial moral effect that helps them be comfortable and build up enthusiasm for their next sessions.

3.3 Advantages of Robot-Assisted Rehabilitation

Robotics play a pivotal role in rehabilitation by offering a wide array of benefits that significantly enhance therapy sessions and patient outcomes. Some of the key advantages include:

Enhanced Feedback

Robot-assisted rehabilitation provides instantaneous and precise feedback to patients during therapy sessions. Unlike human therapists, robots can consistently monitor and analyze movements, offering real-time data on performance and progress.

Precision and Control

Robots are characterized by their exceptional precision and control, which is crucial for exercises targeting specific muscle groups and movements. This precision ensures that therapeutic interventions are accurately administered and adjusted according to patient needs.

Consistency and Repetitiveness

Robots can maintain consistent therapy routines and repetitions without fatigue, ensuring reliable and predictable outcomes over extended periods. This consistency is beneficial for achieving therapeutic goals effectively and efficiently.

Remote Rehabilitation

Advancements in robotic technology have made it feasible to deliver rehabilitation remotely. This capability is particularly advantageous for patients who face geographical or logistical challenges in accessing in-person therapy sessions. Remote rehabilitation reduces barriers to care and facilitates continuous therapy without the need for frequent travel.

Data Gathering and Analysis

Rehabilitation robots generate comprehensive data sets that capture detailed metrics of patient performance. These data are invaluable for analyzing movement patterns, tracking progress, and optimizing therapy protocols. Furthermore, the data can contribute to advancing our understanding of human biomechanics and inform future developments in rehabilitation science.

3.4 Shortcomings of Robot-Assisted Rehabilitation

While robot-assisted rehabilitation offers significant advantages, it also faces several challenges that need to be addressed:

Complexity

Rehabilitation robots are intricate systems comprising numerous components such as sensors, motors, and precise electro-mechanical parts. Operating these devices at full capacity requires well-trained technicians and therapists. Moreover, their reliance on electricity introduces reliability concerns, as power interruptions can disrupt therapy sessions.

High Cost

The initial acquisition and ongoing maintenance costs of rehabilitation robots are substantial. As a nascent technology, robotics equipment tends to be expensive, and the availability of skilled personnel capable of repairing or producing these devices in high volumes is limited. This contributes to the overall high cost of ownership, making it a barrier for widespread adoption in clinical settings.

Dependency on Technology

Robot-assisted rehabilitation heavily relies on integrated hardware and software systems. Repair and maintenance require specialized engineering expertise rather than conventional technical skills. This dependency complicates troubleshooting and repair processes, potentially leading to extended downtimes if technical issues arise.

Lack of Personalization

While robots offer structured or pre-loaded exercise programs, they often lack the flexibility to tailor therapies precisely to individual patient needs. Personalized rehabilitation requires dynamic adjustments based on real-time patient responses, which current robotic systems may struggle to accommodate effectively.

CHU TLEMCEN: the armeo spring orthosis

Presentation

CHU Tlemcen is a hospitalization centre situated in the department of Tlemcen, in Algeria and it has a rehabilitation service created in 1999 specialising in physical medicine and neurological rehabilitation. This service aims at people with pediatric pathology and congenital motor disability and also orthopaedic and rheumatic illness, The patient in this service have access to plenty of equipment and devices under the control of multiple competent therapists and doctors

to help them acquire their well-being such as the armeo spring robot, VR headset and equipment in additionally to classic rehabilitation equipment like a pool, weights, resistance bands and also fes device, it's mainly used to perform several important tasks to regain the patient functionalities such as using the VR headset in parallel with computer software and the armeo spring robot mentioned earlier and shown in figure 1.3 to simulate daily life exercises, and saving data obtained from such tasks to provide helpful feedback to standardize exercises for each patient. The Armeo Spring orthosis is an upper limb exercise orthotic exoskeleton device used in upper limb rehabilitation assessment at Tlemcen's Physical Rehabilitation Center (PRC), the device is passive which means that it's designed to support the weight of the patient's arm, allowing them to perform exercises with reduced gravity. Results taken from this combination are data obtained from the end effector while being mobilized to the effector X-Y coordinates in the real world, movements are accomplished by motivating the user using mini-games/tasks.

Current rehab protocol

Rehabilitation protocol in this service contains various activities to diagnose and proceed in the process that includes in engaging order:

Traditional physical therapy

Traditional physical therapy as practised in CHU Tlemcen or any other therapeutic service is the act of aiming certain symptoms of disabilities or injuries as a standardized protocol through stretching, therapeutic exercises and modalities(heat and cold therapy, electrical stimulation, ultrasound and laser therapy). Traditional physical therapy is based on basic biomechanic principles that are made to improve flexibility, relieve pain, and restore proper alignment.

Vertical assessment task

This step is important to reclassify goals grade the process and place scores according to the rehabilitation journey development. and if to jump to the following step.

Serious game sessions

-this task includes practising some useful games to stimulate the patient's instincts into doing certain useful therapeutic movements without getting bored or tired. These games could be using VR technology to make them real or maybe mini computer games, the patient could play these games using cameras such as (Wii or Kinect) figure1.5, which skeletons the patient sticks like ARMEO or robots to use as effectors in the game figure 1.6, Robots may be passive such as ARMEO SPRING figure,or active such as ARMEO POWER as shown in figure1.7.

4 Data set

4.1 Introduction

After covering the theoretical aspects of rehabilitation processes and therapy methods, we now transition to the practical phase. In this phase, we utilized the ARMEO SPRING ROBOT available at CHU Tlemcen to collect a comprehensive dataset from patients undergoing upper limb rehabilitation. This dataset was gathered using sensors attached to the skeleton's joints, capturing valuable data points and coordinates essential for our primary objective: developing

Fig. 1.5 Patient of Martha-marry Care Centre using a computer mini-game with Kinect

Fig. 1.6 Armeo spring on duty with an injured kid

Fig. 1.7 Group of people testing ARMEO POWER

an algorithm for patient classification based on the obtained data. The collected numerical data were meticulously organized into Excel files, where each row represents a patient and each column represents a specific aspect of upper limb functionality. Additionally, a separate file was maintained to document the progress achieved by patients during their rehabilitation sessions.

4.2 Types of Gathered Data

The gathered data primarily represent the mobility and functional aspects of the upper limb based on various angles and measurements. Some key types of data collected include:

4.2.1 Inner Shoulder Angle

The inner shoulder angle measures the degree of movement when the arms are at the sides, with elbows bent at 90 degrees and forearms facing the body. This angle, crucial for assessing shoulder mobility, typically averages around 70° in healthy individuals [2].

4.2.2 Outer Shoulder Angle

Also known as the critical shoulder angle (CSA), this measurement is defined as the angle formed between the plane of the glenoid and the line connecting the most lateral border of the acromion process. The CSA is important for evaluating shoulder pathology and functionality in rehabilitation scenarios [2]. Note that these are not the only angles obtained ,other angles will be discussed later

CHAPTER 2

Data classification

1 Introduction to machine learning

Machine learning is one of the computer science fields where the machine is capable of learning and growing capabilities without being programmed or interrupted by external hands, it's more like a student with great self-thought capabilities and better recognition of solving problems and patterns just more and more only with practice, machine learning is divided into two major sections :

Supervised learning:
In this approach, machines use labelled data as data inputs so any information we give to a machine with this type should be labelled with its meaning,f.g(inputs are animal pictures, cats should be labelled as cats and dogs should be labelled as dogs)

unsupervised learning: this approach on the other hand uses unlabelled data in which the machine itself tries to find distinct patterns.

The choice between supervised and unsupervised learning depends on the data you're provided with and also the goal you're aiming for, whether you're making it to predict something or just look for new patterns through differences, if the programmer wants both there's a method that fuses both of them supervised and unsupervised in one approach called semi-supervised learning, this method combines both of the approaches and it uses labelled data just as initial training model but it can use a bigger amount of unlabeled data for predictions.

Machine learning and everything else related to computer science uses maths as the stone, in our case maths is translated into different algorithms to differentiate between data given to it in what we call"classification algorithms", as the name shows these algorithms classify data accordingly to a reference input"training model", data as the figure shows might be pictures, numbers, emails, products or even random texts or signals as shown in figure 2.1.

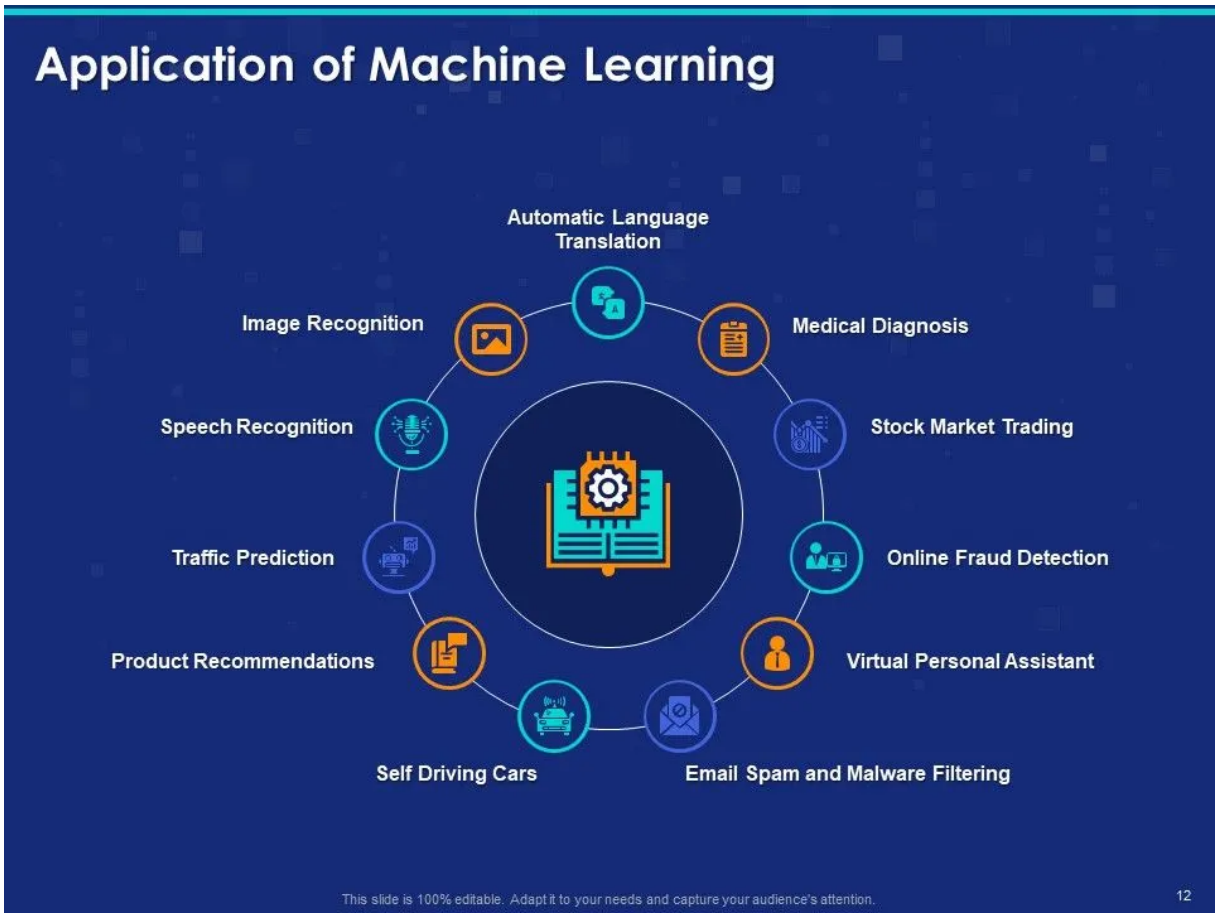


Fig. 2.1 AI applications [7]

1.1 Machine learning applications in biomedical engineering

This thesis aims to use machine learning in upper limb rehabilitation, hence in this section we will have a brief look into a bunch of applications that are already made up in biomedical engineering that serve different services:

Parkinson's Disease Detection Using Machine Learning

Parkinson's disease (PD), as studied in this article [5], is a neurodegenerative disorder that attacks the central nervous system, leading to movement malfunctions, tremors, stiffness, and difficulty walking. The exact cause of PD is unknown, but it is believed to be related to genetics and risk factors such as age, family history, and environment. Traditional detection methods involve medical history and physical exams, but there is no cure, only treatments to manage symptoms, including medication, surgery, deep brain stimulation, and therapy.

In this study, patients provided information before and after taking medication or treatments. Researchers collected multiple voice samples from patients, saved as CSV files, and processed them with Voicebox to remove background noise and PyAudioAnalysis to extract features like frequency, spectrograms, and chromatograms. This analysis resulted in eleven features. Techniques such as 2013 audio-visual emotion recognition and minimum redundancy

maximum relevance (mRMR) were used to prioritize informative features. These features were fed into models like random forest and SVM. Models were tested with different feature set sizes to determine optimal accuracy, concluding that using 1200 features yielded the best results. The Geneva Minimalistic Acoustic Parameter Set (GEMaps) was also used to extract essential sound characteristics.

Various classifiers were explored to achieve the highest accuracy in predicting PD, including decision trees, SVM, and deep learning networks. Stratified cross-validation was used to split the data into training and testing sets while maintaining proportional PD and healthy samples. Model performance was evaluated using metrics such as precision, F1 score, and area under the curve (AUC). The results showed that models using AVEC method features performed better than those using GEMaps features, supported by ROC curve analysis.

Prediction of Epileptic Seizures Using EEG Signals and Machine Learning

Epilepsy is a neurological disease classified as a chronic disorder characterized by recurrent seizures that occur suddenly due to abnormal electrical activity in the brain. This activity can cause several symptoms, such as loss of consciousness, jerking movements of limbs and muscles, sensory disturbances, altered mental state, and emotional changes. The disease may vary from one person to another, with different types including focal seizures and generalized seizures. This disorder can result from various circumstances such as head injury, stroke, brain tumour, genetic predisposition, and infections. Doctors diagnose epilepsy using medical history, physical exams, EEG to analyze brain activity, and imaging tests. Unfortunately, there is no cure for this disorder, only symptom management methods such as medication, surgery, and vagal nerve stimulation. EEG is particularly effective in diagnosing epilepsy and has been broadly used to predict appropriate medication for patients with this disorder. Recently, many studies have aimed to implement EEG with ML, and the objective of this paper is to further advance these research efforts in applying ML for epilepsy prediction.

The article [19] starts with a brief introduction to neuroscience and then explains the EEG signal, a non-invasive method to measure brain electrical activity using electrodes placed on the scalp, invented in 1923. This signal is analyzed based on its frequency interval up to 100 Hz and ranges from 10 to 100 microvolts. Like other signals, EEG undergoes various processing steps starting with feature extraction to transform them for training ML models for tasks like ours, passing through spectral analysis by applying mathematical theories. To record this signal, doctors or therapists place electrodes on active scalp regions or one on that region and the other on an inactive region. There are numerous ways to study this signal, including time domain methods, frequency domain methods, time-frequency domain methods, and non-linear methods. An example of seizure prediction is "my seizure gauge."

To begin with, the model, the EEG signal taken from the patient should be pre-processed for better seizure prediction. Pre-processing starts with reducing noise caused by power lines, muscle movements, or faulty electrode recordings using common filtration techniques such as bandpass filters, wavelet filters, finite impulse response, and adaptive filters. It should then be normalized and prepared for comparison and analysis. Features must be extracted and selected based on two aspects (mathematics and channels). Finally, the most important step is to distinguish pre-ictal and interictal (between seizure) patterns in EEG data obtained from the previous steps using several algorithms such as ANNs, K-means clustering, decision trees, SVM, and fuzzy logic.

Behavioral Modeling for Mental Health Using Machine Learning Algorithms

Just like physical health, mental health is crucial for overall well-being. It encompasses the ability to make relationships, communicate, socialize, and feel good about oneself. Psychological illnesses can sometimes be fatal for some individuals. The World Health Organization (WHO) predicted in 2011 that by 2030, depression would be the leading cause of global disease burden. Due to its importance, mental health has become attractive to scientists who are developing new tools to detect such problems, and significant progress has been made. For example, phone apps and wearable devices have been created to collect data, which can then be analyzed by computer programs and passed through ML models and complex algorithms to find patterns that might be related to mental health challenges. This process is called "predictive analytics in mental health." These tools are designed to assist therapists, not replace them.

In this paper, the authors aim to create a model that can identify individuals who are mentally distressed in the target population. Inputs for this model are the population's responses to various questions created by psychologists. The next step is clustering, where researchers try to find similarities and group them into useful sets, helping to understand data and identify potential classes for further processing. Unsupervised ML, also known as clustering, is used for this purpose. The most commonly used clustering algorithms are: K-Means clustering, Hierarchical clustering, Density clustering, K-medoids, and their variants. To confirm if the number of clusters obtained by applying these algorithms is correct, cluster validation is performed [24].

Machine Learning Methodologies in Brain-Computer Interfaces

This article discusses the application of machine learning in brain-computer interface (BCI) systems, focusing on the P300 Speller BCI paradigm. BCI systems enable communication through brain signals without motor functions. The role of machine learning algorithms is to differentiate between brain activity patterns. The algorithm used in this study is the SVM classifier, which showed high accuracy in categorizing EEG (Electroencephalography) patterns. The P300 Speller BCI helps disabled subjects spell words using brain signals. Gathering data and classifying it using SVM is crucial for enhancing BCI systems' performance.

Application of Real-Time Machine Learning to Myoelectric Prosthesis Control

The article [21] discusses the application of machine learning in myoelectric prosthesis control, focusing on real-time machine learning methods. The study highlights the use of adaptive switching, which significantly reduces the time and cognitive load needed for joint control compared to non-adaptive methods. This improves efficiency and user experience, underscoring the potential of adaptive switching to enhance prosthetic control autonomously, offering better functionality and user interaction across different tasks.

Epidemic Model Guided Machine Learning for COVID-19 Forecasts in the United States

The article [31] focuses on developing an epidemic model using machine learning to forecast COVID-19 spread in the United States. The study focuses on the SuEIR model, which incorporates parameters like the incubation period and underreported cases to predict COVID-19 dynamics. It highlights the use of machine learning algorithms to train and predict the spread of the virus accurately. This research emphasizes the importance of accurate initialization and validation sets for model training.

1.2 Conclusion

In all of the articles and projects studied and summarized, the models developed aimed at facilitating the medical diagnosis of various prevalent and dangerous diseases that are increasingly affecting the population. It is important to highlight that these models were not created to replace medical staff, but rather to support and enhance their work to achieve faster and more accurate results. This can help in the early detection and treatment of diseases, ultimately leading to a better prognosis for patients.

Upon analyzing the articles, it was found that the first step in the development of these models was to gather proper datasets from as many patients as possible. The datasets needed to be clear and accurate to ensure the efficiency of the models. The data obtained was then subjected to a preprocessing phase, which involved the use of standard techniques, albeit with different libraries.

The preprocessing phase included opening the data file in its original format, removing null rows and columns, and eliminating duplicates. The removal of null rows and columns was necessary as they do not provide any useful information and can result in errors in the model's predictions. The elimination of duplicates was necessary to avoid the replication of data, which could skew the results and lead to erroneous diagnoses and slower results.

The models aimed to analyze the preprocessed data to identify patterns and correlations that could aid in disease diagnosis and prediction. After preprocessing, the data was subjected to feature extraction and selection to ensure that only the most relevant information was used for model training. Various machine learning algorithms were then applied, including neural networks, support vector machines, and clustering algorithms, depending on the specific requirements of each study.

Overall, these models demonstrated significant potential in improving medical diagnosis processes, providing valuable insights that can assist healthcare professionals in making informed decisions. Future work in this area will likely focus on further refining these models, improving their accuracy, and integrating them more seamlessly into clinical practice to maximize their impact on patient care.

2 Introduction to classification algorithms

Classification algorithms are complex algorithms used in computer science precisely in data science to classify data or classes based on specific criteria, most popular classification algorithms use Logistic Regression, Naive Bayes, K-Nearest Neighbors, Decision Trees, and Support Vector Machines. These algorithms are so important in supervised learning (mentioned before) by predicting characterised values for future observations and aiding in various real-time tasks such as face detection, document classification, handwriting recognition, and speech recognition. The choice of classification algorithm is based on the aim of the user. For instance, Logistic Regression is used to predict binary results for example Yes/No or Pass/Fail, while Naive Bayes is based on probability calculations. Decision Trees are effective for document classification, and Support Vector Machines draw hyper-planes to classify data with polarity degrees beyond simple X/Y predictions‘ [1]. Different tools for different jobs In machine learning, there are many classification algorithms, and each one works best for certain tasks. Picking the right one is important for getting accurate results and making the most of the data. By understanding these algorithms, researchers can unlock hidden patterns in information, leading to better decisions. In the next sections, we'll step more into these methods.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a straightforward machine learning algorithm that classifies new data based on similarity to existing data points. In KNN, the value of 'k' represents the number of nearest neighbors used in the classification process.

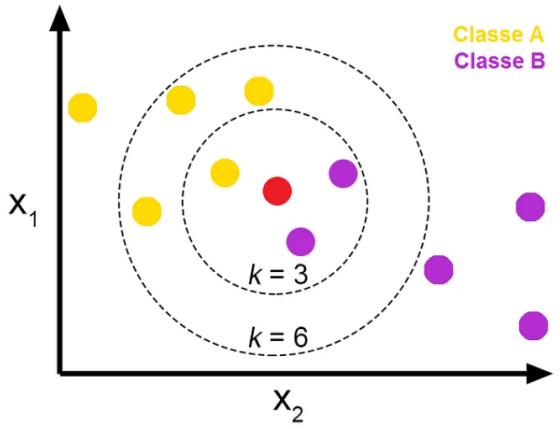


Fig. 2.2 The parameter 'k' in KNN

The algorithm works by selecting the 'k' nearest neighbors based on a distance metric like Euclidean distance. By comparing the categories of these neighbors, KNN assigns the new data point to the most common category among its neighbors.

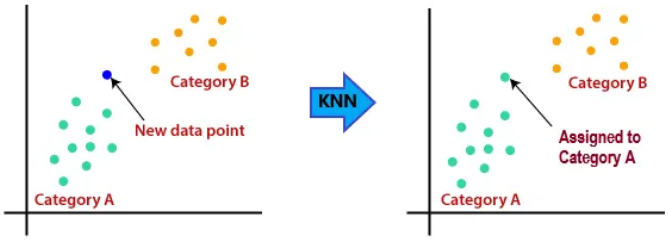


Fig. 2.3 Simplified illustration of the KNN algorithm

KNN is appreciated for its simplicity in implementation and its capability to handle multi-class cases. However, choosing the optimal 'k' value and computing distances for each query instance can be computationally expensive [30].

In rehabilitation, KNN utilizes assessment data to predict patient outcomes, aiding therapists in making informed decisions based on comparisons between training data and actual patient data [25, 30].

2.1 Support Vector Machines (SVM)

Support Vector Machines (SVM) are powerful supervised learning models used for classification tasks. SVMs find the optimal hyperplane that separates data into classes with the maximum possible margin. They are versatile and capable of handling linearly separable and non-linearly separable data through kernel functions.

SVMs are widely used for binary classification tasks and excel in processing high-dimensional data efficiently. Their strong theoretical foundations and flexibility make them suitable for various applications including rehabilitation.

- SVMs predict functional recovery outcomes post-stroke, aiding clinical decision-making processes [9, 11, 30].
- They enhance precision in robot-assisted interventions by classifying gait patterns indicative of knee osteoarthritis and predicting post-surgery gait improvement [10, 13, 15].
- SVM-based classifiers analyze and validate finger kinematics in stroke patients undergoing rehabilitation, showcasing their role in evaluating hand movement post-stroke [13].

2.2 Linear Regression

Linear regression is a fundamental supervised learning algorithm that establishes a linear relationship between input variables (predictors) and an output variable (target). The model predicts the target value by minimizing the error between predicted and actual values, typically using techniques like gradient descent to update coefficients and minimize Mean Squared Error (MSE).

- Linear regression is widely used for prediction tasks, time series analysis, and understanding cause-effect relationships [6, 17, 28].
- It assumes a linear relationship between dependent and independent variables, making it suitable for forecasting trends and understanding correlations in data [17].

3 Classification algorithms

3.1 Introduction

Classification algorithms are complex algorithms used in computer science precisely in data science to classify data or classes based on specific criteria, most popular classification algorithms use Logistic Regression, Naive Bayes, K-Nearest Neighbors, Decision Trees, and Support Vector Machines. These algorithms are so important in supervised learning (mentioned before) by predicting characterised values for future observations and aiding in various real-time tasks such as face detection, document classification, handwriting recognition, and speech recognition. The choice of classification algorithm is based on the aim of the user. For instance, Logistic Regression is used to predict binary results for example Yes/No or Pass/Fail, while Naive Bayes is based on probability calculations. Decision Trees are effective for document classification, and Support Vector Machines draw hyper-planes to classify data with polarity degrees beyond simple X/Y predictions‘ [1]. Different tools for different jobs In machine learning, there are many classification algorithms, and each one works best for certain tasks. Picking the right one is important for getting accurate results and making the most of the data. By understanding these algorithms, researchers can unlock hidden patterns in information, leading to better decisions. In the next sections, we'll step more into these methods.

3.2 How to use the classifiers

Before creating the model we need to classify data we should first pass through various steps:

Step 1: Collecting Data

The first step in developing a machine learning model for upper limb rehabilitation is to collect comprehensive and labeled training data representing each patient's rehabilitation session. This data should include detailed information such as the types of exercises performed, session duration, intensity levels, and physiological measurements. Cleaning the data is crucial to ensure accuracy and reliability, which involves removing outliers, addressing missing values, and ensuring the data is as precise as possible. The quality and quantity of features directly impact the model's output. It is also essential to determine whether the data is labeled to decide if supervised or unsupervised machine learning techniques will be employed.

Step 2: Data Preparation

The raw data collected cannot be used directly in the model. It needs to undergo a rigorous cleaning process to remove duplicates, biases, and errors. One of the filtering algorithms that can be used is the Kalman filter, as employed in this thesis. Data preparation also involves transforming and fitting the data into the appropriate format for the algorithm. Visualizing the data can be helpful to identify patterns and relationships, which can inform further processing steps.

Step 3: Feature Labeling and Representation

Identifying and representing the various features in the dataset is crucial for the model. These features include the types of exercises (e.g., stretching, resistance training, and range of motion exercises), the duration and intensity of each exercise (such as resistance levels and number of repetitions), physiological measurements (including muscle activity and joint angles), and patient demographic information (such as age, gender, and medical history). Selecting informative and relevant features is essential for effectively distinguishing between different types of rehabilitation sessions.

Step 4: Selecting the Correct Kernel Algorithm

When training the model, selecting an appropriate kernel function is vital to accurately capture the relationships between features and rehabilitation outcomes. The Radial Basis Function (RBF) kernel is often used due to its capability to model complex relationships. Building the Support Vector Machine (SVM) model involves using labeled training data to learn the underlying patterns and relationships between rehabilitation sessions and their outcomes. Fine-tuning the model's parameters throughout the training process enhances its ability to differentiate between various types of rehabilitation sessions while minimizing classification errors.

Step 5: Evaluation and Testing

Evaluating the SVM model's performance involves assessing metrics such as accuracy, precision, recall, and speed. Techniques like cross-validation are applied to ensure the model's ability to generalize and perform well on new, unseen data. Evaluating the model's performance helps identify any weaknesses or areas for improvement. It is crucial to examine how well the model predicts outcomes and its robustness in handling different data variations.

Step 6: Recollecting New Data for Implementation

After training and assessing the model, it can be applied to classify new rehabilitation sessions for patients. The SVM algorithm uses the characteristics of a new session to predict the most likely outcome or classification (e.g., session effectiveness or patient progress). These predictions can then inform decision-making in patient care, allowing for adjustments to rehabilitation programs based on the anticipated outcomes. Continuous recollection of new data ensures the model remains up-to-date and improves over time, providing more accurate predictions and supporting better patient outcomes.

3.2.1 Dynamic Time Warping Algorithm

Dynamic Time Warping (DTW) is an algorithm used in time series analysis for measuring the similarity between two temporal sequences that may vary in speed. The primary goal of DTW is to align these sequences optimally by minimizing the distance between them. This technique is particularly useful when the sequences are of different lengths or have temporal distortions. The DTW algorithm achieves this by creating a warping path through a cost matrix, which helps in finding the minimal cumulative distance between the sequences.

The DTW algorithm can be described as follows:

Given two sequences:

$$X = (x_1, x_2, \ldots, x_n)$$
$$Y = (y_1, y_2, \ldots, y_m)$$

We define a $n \times m$ cost matrix D, where each element $D(i,j)$ represents the cost (or distance) between the elements x_i and y_j. The distance is typically computed using an absolute difference:

$$d(x_i, y_j) = |x_i - y_j|$$

The cumulative cost $D(i,j)$ is computed as:

$$D(i,j) = d(x_i, y_j) + \min(D(i-1,j), D(i,j-1), D(i-1,j-1))$$

The boundary conditions are:

$$D(1,1) = d(x_1, y_1)$$
$$D(i,0) = \infty \quad \text{for} \quad i > 1$$
$$D(0,j) = \infty \quad \text{for} \quad j > 1$$

The optimal warping path $W = (w_1, w_2, \ldots, w_k)$ is found by tracing back from $D(n,m)$ to $D(1,1)$. The warping path minimizes the total cost of alignment between the two sequences.

To measure the similarity between the two sequences, the DTW distance is defined as the cumulative cost along the optimal path:

$$DTW(X,Y) = \frac{1}{K} \sum_{k=1}^{K} D(w_k)$$

where K is the length of the warping path, and $D(w_k)$ is the cost associated with the k-th element of the warping path.

DTW is widely used in various AI and machine learning tasks, including:

- **Speech Recognition**: Aligning spoken words or phonemes, despite variations in speaking speed or pronunciation.
- **Handwriting and Signature Recognition**: Comparing handwritten signatures or characters that may have different writing speeds or styles.
- **Gesture Recognition**: Matching sequences of gestures or movements, which may vary in duration.
- **Time Series Classification and Clustering**: Identifying patterns and similarities in financial, biological, or any other temporal data.

The flexibility of DTW in handling time series data with different lengths and temporal distortions makes it a powerful tool in these applications.

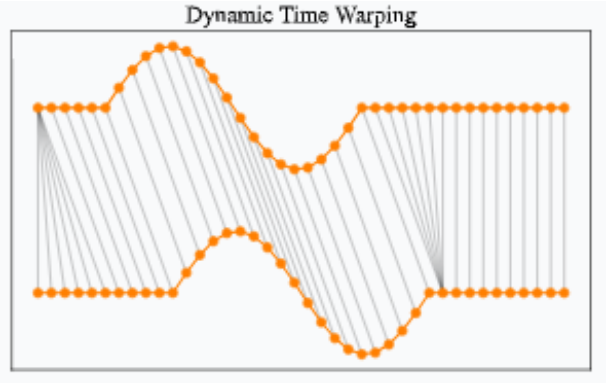


Fig. 2.4 Dynamic Time Warping path visualization. The path aligns the sequences X and Y by minimizing the cumulative cost [26].

4 KNN

Introduction

K-Nearest Neighbors (KNN) is a simple yet effective machine learning algorithm used for classification tasks based on similarity metrics. In KNN, the parameter 'k' determines the number of nearest neighbors considered during classification (Figure 2.5). The algorithm computes the distance between the new data point $\mathbf{x}_{\text{new}}$ and each training data point $\mathbf{x}_i$ using a distance metric such as the Euclidean distance:

$$d(\mathbf{x}_{\text{new}}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{n} (x_{\text{new},j} - x_{i,j})^2}$$

Where n is the number of features. It then selects the 'k' closest neighbors based on these distances. The classification decision for $\mathbf{x}_{\text{new}}$ is determined by a majority vote among its nearest neighbors, given by:

$$\hat{y}_{\text{new}} = \arg\max_{y} \sum_{i \in \text{NearestNeighbors}} I(y_i = y)$$

Where $\hat{y}_{\text{new}}$ is the predicted class for $\mathbf{x}_{\text{new}}$, y_i are the classes of the neighbors, and $I(\cdot)$ is the indicator function. Figure 2.6 illustrates the process of assigning a new data point to a class using KNN.

KNN is favoured for its simplicity and ability to handle multi-class classification problems and noisy data. However, choosing an appropriate 'k' value is crucial and can impact the algorithm's performance significantly [30]. In the context of rehabilitation, KNN leverages patient assessment data to predict the potential benefits of rehabilitation, providing therapists with valuable insights for patient management [25, 30].

Implementing KNN

We can categorize the KNN method into 5 essential steps:

Step 1: Data Preparation

Preparing our data, whether for training or testing, involves filtering, arranging, and plotting it to gain a comprehensive understanding of its structure. This step is crucial for identifying

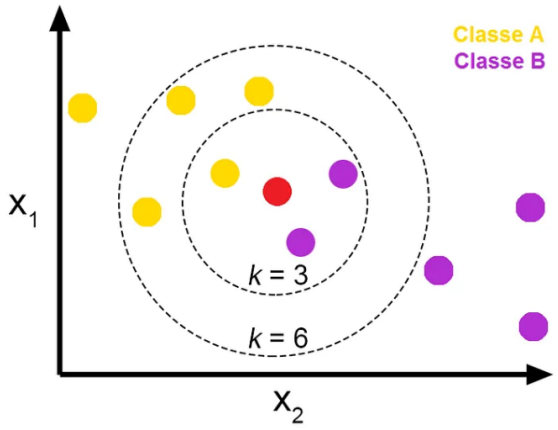


Fig. 2.5 The "K" parameter in KNN [14].

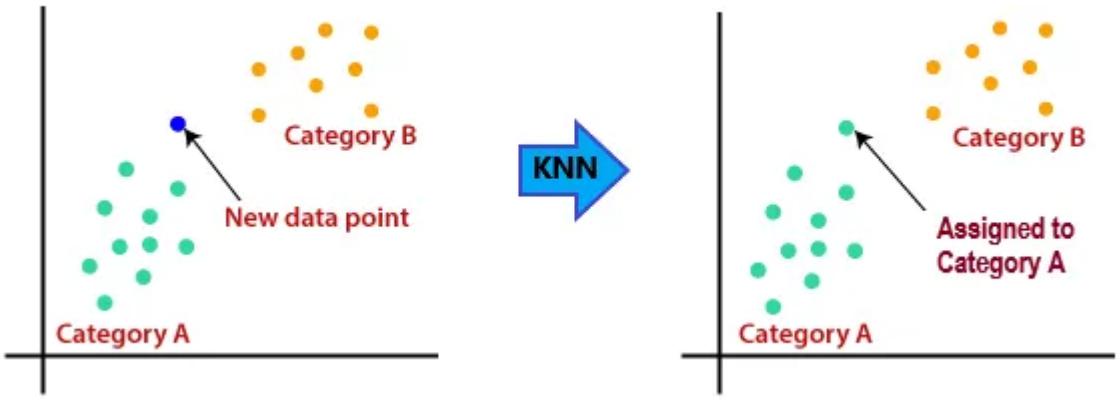


Fig. 2.6 Simplified illustration of the KNN algorithm Adapted from [27].

patterns and anomalies, which can significantly impact the performance of the Dynamic Time Warping (DTW) algorithm. The DTW algorithm is particularly effective for measuring similarity between two temporal sequences, which may vary in speed. This preparatory phase includes the following sub-steps:

- **Filtering:** Removing noise and irrelevant data to ensure the dataset's integrity.
- **Arranging:** Structuring the data in a consistent format suitable for analysis.
- **Plotting:** Visualizing the data to identify trends, patterns, and potential outliers.

Step 2: Distance Calculation

We measure the distance between each patient's data point and the data points in the training set. Although there are various distance calculation methods (e.g., Euclidean distance, Manhattan distance, and Minkowski distance), we opted for the Euclidean method due to its compatibility with our dataset and ease of use. The Euclidean Distance between two points p and q in n-dimensional space is defined as:

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Step 3: Identifying the Nearest Neighbors

After calculating the distances between each data point, the algorithm sorts them and selects the smallest distances between the patient data points and the training data points. This involves:

- **Sorting:** Arranging the distances in ascending order.
- **Selection:** Choosing the k smallest distances, where k is a predetermined number of neighbors.

Step 4: Aggregation of the Nearest Neighbors

In classification using the K-Nearest Neighbors (KNN) algorithm, we gather the class labels of the 'k' nearest neighbors to make a prediction about the class of the current data point. This is achieved by identifying the mode (most frequent class) among the 'k' neighbors. Mathematically, this is represented as:

$$\text{Classification KNN} = C_q = \text{mode}\{C_{n_1}, C_{n_2}, \ldots, C_{n_k}\}$$

Where C_{n_i} represents the class label of the i-th nearest neighbor.

Step 5: Outcome Prediction

KNN makes predictions about the class for classification tasks. Unlike other algorithms, KNN does not require a specific model upfront. Instead, it utilizes the dataset and calculates distances between points to make predictions. This non-parametric approach allows for flexibility in handling various data distributions.

Step 6: Error Rate Calculation

The error rate of the model on the testing set is calculated to evaluate its performance. The error rate is defined as the ratio of the number of incorrectly classified data points (M) to the total number of data points (N) in the testing set. The error rate E can be computed as:

$$E = \frac{M}{N}$$

where:

- N is the total number of data points in the testing set.
- M is the number of incorrectly classified data points.

Step 7: Parameter Fine-Tuning

To achieve the desired accuracy and execution speed, the algorithm's parameters need to be fine-tuned. This includes adjusting the "k" parameter, which represents the number of nearest neighbors considered during classification. Fine-tuning may involve:

- **Cross-Validation:** Using techniques such as k-fold cross-validation to assess the performance of different parameter values.
- **Grid Search:** Systematically searching through a range of parameter values to identify the optimal configuration.
- **Regularization:** Applying methods to prevent underfitting and improve generalization.

Fine-tuning these parameters helps in reducing the error rate and enhancing the algorithm's performance.

5 SVM Support Vector Machines

Introduction

Support Vector Machines (SVM) are supervised machine learning algorithms used for linear or non-linear classification, regression, and even outlier detection tasks. The main objective of this algorithm is to find the optimal hyperplane that separates data into classes with the largest possible margin in an N-dimensional space. This hyperplane separates the data points into different classes in the feature space, as shown in Figure 2.7. The hyperplane aims to maximize the margin between different classes, and the dimension of the hyperplane is dependent on the number of features.

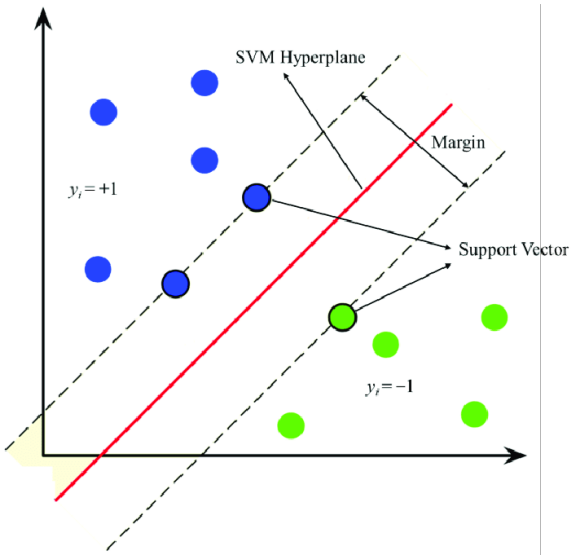


Fig. 2.7 Illustration of a linear SVM classifier separating the two classes [29].

Hyperplanes

Hyperplanes are decision boundaries that separate different sets of data points in an N-dimensional space. For a 2D space, a hyperplane is a line, and for a 3D space, it is a plane. The purpose of the hyperplane is to classify data points into different categories. The farther the data points are from the hyperplane, the more confident we are in their classification.

A key concept in SVM is the **margin**, which is the distance between the hyperplane and the nearest data point from either class. The goal of SVM is to maximize this margin. However, in many real-world scenarios, a clear linear separation may not be possible due to overlapping data points, known as a linearly non-separable dataset. In such cases, SVM uses a technique called **kernelling** to handle non-linear separations.

Using kernel functions, SVMs can handle both linearly separable and non-linearly separable data. The term "kernel" refers to a method used to transform input data into a higher-dimensional space. Some widely used kernel functions include:

5.0.1 Linear Kernel

The linear kernel calculates the dot product of the input vectors in the original feature space, making it ideal for data that can be separated linearly. The equations governing the linear SVM are:

$$\begin{aligned} w \cdot x - b &\geq 1 && \text{if } y_i = 1 \\ w \cdot x - b &\leq -1 && \text{if } y_i = -1 \end{aligned}$$

$$y_i(w \cdot x - b) \geq 1 \quad \text{for all } y_i = \{-1, 1\}$$

where:

- y_i represents the class label.
- x_i represents the input feature vector.
- w is the weight vector perpendicular to the hyperplane.
- b is the bias term.

5.0.2 Polynomial Kernel

This kernel calculates the polynomial expansion of the input vectors' dot product, making it beneficial for non-linear data and enabling the creation of more intricate decision boundaries, as shown in Figure 2.8.

5.0.3 Radial Basis Function (RBF) Kernel

Also known as the Gaussian kernel, it is a popular choice for mapping data points into a high-dimensional space using a Gaussian similarity function. It is widely used due to its ability to effectively capture complex decision boundaries, as shown in Figure 2.9.

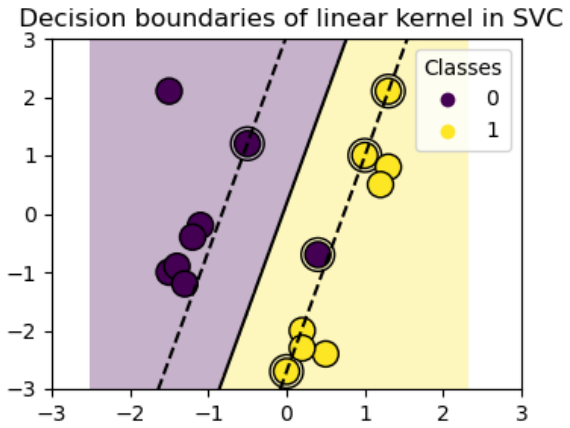


Fig. 2.8 Polynomial kernel [3].

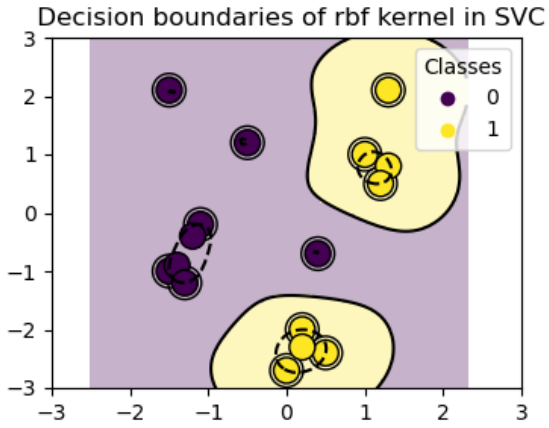


Fig. 2.9 Gaussian kernel [3].

5.0.4 Sigmoid Kernel

Derived from the hyperbolic tangent function, the Sigmoid Kernel is ideal for handling data that cannot be easily separated in its original feature space, as shown in Figure 2.10.

SVMs boast strong theoretical foundations, flexibility, and straightforward usage. Unlike the KNN algorithm, SVMs are memory-efficient due to using a subset of training points in the decision function, making them widely used in high-dimensional spaces.

Although conceptualized in the 1960s, their introduction in the 1990s sparked widespread adoption across applications. In rehabilitation contexts, SVMs can predict functional recovery outcomes post-stroke, assist clinical decision-making processes, and enhance precision in robot-assisted interventions [30] [9] [11].

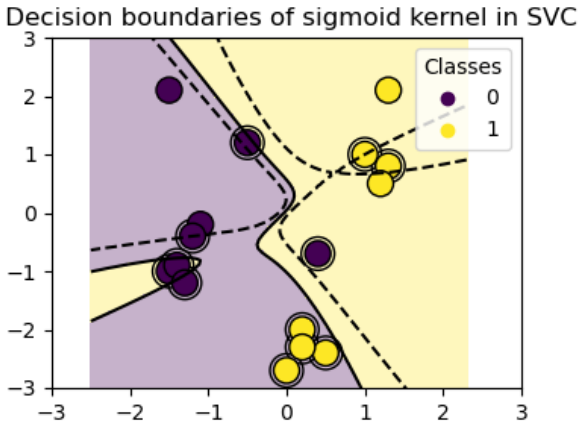


Fig. 2.10 The Sigmoid Kernel [3].

5.1 Implementing SVM

Step 1: Choosing the Appropriate Libraries

Before starting to code, we must first import the appropriate libraries from the programming environment into our code. For SVM, the essential libraries include:

scikit-learn: This Python library is known for its ease of use and wealth of classes that can be used in SVM, such as `SVC`, `NuSVC`, `LinearSVC`, `SVR`, and `OneClassSVM`. It also provides a variety of other features such as:

- Support for different kernel functions (linear, polynomial, RBF, sigmoid)
- Automatic scaling of features for better performance
- Cross-validation for hyper-parameter tuning
- Probability estimates for classification

LIBSVM: A C/C++ library known for its speed when working with large datasets and high efficiency with results. It provides fine control over SVM parameters and supports various kernel functions.

PyTorch: Developed by META AI lab, PyTorch offers dynamic graph computation and seamless integration with Python. It provides strong GPU acceleration, making it efficient for large-scale training.

NumPy and Pandas: These libraries are indispensable for data processing and manipulation, forming a fundamental structure for data handling and transformation.

Keras: Built on top of TensorFlow, Keras provides a high-level API for building neural networks. It allows for rapid prototyping and experimentation with deep learning models and supports both convolutional neural networks (CNNs) and recurrent neural networks (RNNs), among other architectures.

Step 2: Define the SVM Model

In this step, we should choose the type of SVM model based on the problem we want to solve or study. The choices include:

- **Classification (binary or multi-class):** `SVC` (scikit-learn), `C-SVC` (LIBSVM)
- **Regression (linear or non-linear):** `SVR` (scikit-learn), `nu-SVR` (LIBSVM)
- **Outlier detection:** `OneClassSVM` (scikit-learn)

Step 3: Train the SVM Model

To train the SVM using the training data, we use the library to find the optimal hyperplane or decision function based on the training data. The process involves:

- Selecting the appropriate kernel function
- Setting the hyper-parameters
- Feeding the training data into the algorithm

The SVM then learns to classify the data points by finding the optimal separating hyperplane that maximizes the margin between the two classes. This hyperplane is represented by a decision boundary that separates the two classes in the feature space. Mathematically, the optimization problem for a linear SVM can be formulated as:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i (w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \end{aligned}$$

where:

- w is the weight vector.
- b is the bias term.
- ξ_i are the slack variables.
- C is the regularization parameter.

Step 4: Make Predictions

Once the model is trained, the next step is predicting class labels or continuous values for unseen data points in the testing session. This involves feeding the trained model with new data points that have not been used during the training phase. The model will make predictions based on its learned decision boundary. The prediction function for a linear SVM is:

$$f(x) = \text{sign}(w \cdot x + b)$$

For non-linear SVMs with kernel functions, the decision function becomes:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \right)$$

where:

- α_i are the Lagrange multipliers.
- $K(x_i, x)$ is the kernel function.

Step 5: Model Evaluation

Evaluating a machine learning model's performance involves using relevant metrics to assess its accuracy. The choice of metrics depends on the type of problem being solved:

- For classification tasks, **accuracy**, **precision**, **recall**, and the **F1-score** are common metrics.
- For regression tasks, **mean squared error (MSE)** and **mean absolute error (MAE)** are typically used.

Accuracy is calculated as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Mean squared error is calculated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where:

- y_i are the actual values.
- $\hat{y}_i$ are the predicted values.
- n is the number of data points.

We can also visualize the output using plots to compare actual values and predicted values, providing important insights into the model's accuracy and effectiveness in real-world scenarios.

6 Linear regression

6.1 Introduction

Linear regression has been in use for over two centuries and remains essential in fields like economics, epidemiology, social sciences, and machine learning. It is valued for its simple interpretation and strong mathematical basis [18]. The basic idea is to use historical data to spot patterns and predict future trends, which helps in decision-making and understanding cause-and-effect relationships. In today's world, linear regression is thriving in the era of big data brought on by the digital age. It is being used for tasks such as market forecasting, risk assessment, medical diagnostics, and environmental modelling [18]. With advancements in computing power and sophisticated algorithms, linear regression continues to evolve and integrate with modern techniques to improve predictions and accuracy [18].

The appeal of linear regression lies in its concept of linearity, where changes in predictor variables lead to proportional shifts in the outcome variable. This simplifies interpretation and helps to identify the key factors behind observed phenomena [22]. Linear regression goes beyond just prediction; it helps distil complex data into useful insights for making policies, business strategies, and scientific discoveries [4].

Mathematically, linear regression is as simple as the linear equation:

$$y = ax + b$$

where it defines the relationship between:

- y = the dependent variable
- x = the independent variable
- a = the scale factor, coefficient, or the intercept of the regression
- b = the bias coefficient

The bias coefficient is used as an extra degree to the model used where it sits directly in front of the independent variables, these coefficients could be found through different methods, there is various regression coefficients it's not always as simple as it cold tells you the value of (Y) if all if the independent values were zeros, Differently than the previous equation other equations could be more complex ,The size and direction of the relationship between the dependent variable (also known as the response variable) and a specific independent variable (also known as a predictor variable) are estimated by each regression coefficient.to look more into these equations we will use three types of equations

6.1.1 Simple linear regression

this is simply

$$y = ax + b$$

in this case the relationship between x and y is only approximated by a straight line and the coefficient or (a) will give the slope of that line

6.1.2 Multiple linear regression

linear regression can have multiple independent variables so more than one regression coefficient called "multiple linear regression" in the form of The regression equation in its general form is:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon$$

where:

- y = the predicted value of the dependent variable
- β_0 = the y-intercept (value of y when all other parameters are set to 0)
- $\beta_1 X_1$ = the regression coefficient (β_1) of the first independent variable (X_1) (a.k.a. the effect that increasing the value of the independent variable has on the predicted y value)
- $\cdots$ = do the same for however many independent variables you are testing
- $\beta_n X_n$ = the regression coefficient of the last independent variable
- ϵ = model error (a.k.a. how much variation there is in our estimate of y)

This model is used when there are multiple independent variables. For example, suppose you have data on a person's height, age, and sex (which are independent variables) and you want to predict their weight (the dependent variable). In that case, multiple linear regression can be used to understand how each of these factors influences weight.

6.2 coefficients calculation

these coefficients earlier discussed are still unknown and need to be found for this method to be used to find these coefficients we use two approaches OLS (ordinary least mean square method) and the Gradient descent.

6.3 Ordinary least mean square

The multiple linear regression model is given by:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon$$

matrecial of :

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where:

- $\mathbf{y}$ is the vector of dependent variable values.
- $\mathbf{X}$ is the matrix of independent variables (with a column of ones for the intercept).
- $\boldsymbol{\beta}$ is the vector of coefficients.
- $\boldsymbol{\epsilon}$ is the vector of errors.

6.3.1 Preparing the Data

- **Matrix of Independent Variables ($\mathbf{X}$)**: This matrix includes a column of ones for the intercept term and columns for each of the independent variables.
- **Vector of Dependent Variable ($\mathbf{y}$)**: This vector includes the observed values of the dependent variable.

For example, if you have n observations and p independent variables, $\mathbf{X}$ will be an $n \times (p+1)$ matrix (including the intercept), and $\mathbf{y}$ will be an $n \times 1$ vector.

6.3.2 Computing the Coefficients

To find the coefficients $\boldsymbol{\beta}$ that minimize the residual sum of squares, we use the following formula:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

1. **Compute $\mathbf{X}^T\mathbf{X}$**: This is the matrix product of the transpose of $\mathbf{X}$ and $\mathbf{X}$.
2. **Compute $\mathbf{X}^T\mathbf{y}$**: This is the matrix product of the transpose of $\mathbf{X}$ and $\mathbf{y}$.
3. **Invert $\mathbf{X}^T\mathbf{X}$**: Find the inverse of the matrix $\mathbf{X}^T\mathbf{X}$.
4. **Compute $\hat{\boldsymbol{\beta}}$**: Multiply the inverse of $\mathbf{X}^T\mathbf{X}$ by $\mathbf{X}^T\mathbf{y}$ to obtain the estimated coefficients.

6.3.3 example of implementation

Consider a random dataset with three independent variables X_1, X_2, and X_3, and a dependent variable y. The dataset is:

y	X_1	X_2	X_3
4	2	3	1
6	3	5	2
5	4	4	3
8	5	6	2
7	6	5	4

6.3.4 Formulating the Model

The multiple linear regression model is given by:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

In matrix form, it is:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 1 & 3 & 5 & 2 \\ 1 & 4 & 4 & 3 \\ 1 & 5 & 6 & 2 \\ 1 & 6 & 5 & 4 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 4 \\ 6 \\ 5 \\ 8 \\ 7 \end{bmatrix}$$

6.3.5 Computing the Coefficients

To find the coefficients $\boldsymbol{\beta}$ that minimize the residual sum of squares, we use the formula:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

1. **Compute $\mathbf{X}^T\mathbf{X}$**:

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 5 & 20 & 23 & 12 \\ 20 & 90 & 97 & 54 \\ 23 & 97 & 111 & 57 \\ 12 & 54 & 57 & 34 \end{bmatrix}$$

2. **Compute $\mathbf{X}^T\mathbf{y}$**:

$$\mathbf{X}^T\mathbf{y} = \begin{bmatrix} 30 \\ 128 \\ 145 \\ 75 \end{bmatrix}$$

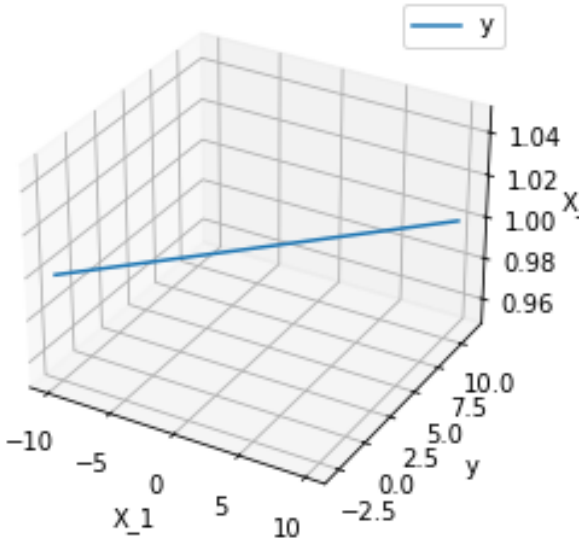


Fig. 2.11 plot of the y1 equation

3. **Invert $\mathbf{X}^T\mathbf{X}$**:

$$(\mathbf{X}^T\mathbf{X})^{-1} = \begin{bmatrix} 5.5556 & 0.9722 & -1.4444 & -1.0833 \\ 0.9722 & 0.8264 & -0.5278 & -0.7708 \\ -1.4444 & -0.5278 & 0.5556 & 0.4167 \\ -1.0833 & -0.7708 & 0.4167 & 0.9375 \end{bmatrix}$$

4. **Compute $\hat{\beta}$**:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \begin{bmatrix} 0.4167 \\ 0.6042 \\ 0.9167 \\ -0.4375 \end{bmatrix}$$

6.3.6 Regression Equation

The estimated regression equation is:

$$y1 = 0.4167 + 0.6042X_1 + 0.9167X_2 + -0.4375X_3$$

with a plot that looks like [**?**]

This example demonstrates the application of the Ordinary Least Squares (OLS) method to a multiple linear regression problem with three independent variables.

6.4 Gradient Descent for Linear Regression

Gradient Descent is an iterative optimization algorithm used to find the parameters (coefficients) of a model that minimize the cost function. In linear regression, the goal is to find the coefficients that minimize the Mean Squared Error (MSE) between the observed values and the predicted values.

6.4.1 Formulating the Cost Function

The cost function $J(\boldsymbol{\beta})$ used in Gradient Descent is the same as Mean Squared Error (MSE) function:

$$J(\boldsymbol{\beta}) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

where:

- m = number of training examples
- y_i = observed value of the i-th training example
- $\hat{y}_i$ = predicted value of the i-th training example

The predicted value $\hat{y}_i$ is given by:

$$\hat{y}_i = \mathbf{x}_i^T \boldsymbol{\beta}$$

where $\mathbf{x}_i$ is the vector of features for the i-th example, and $\boldsymbol{\beta}$ is the vector of coefficients.

6.4.2 Gradient Descent Algorithm

The Gradient Descent algorithm iteratively adjusts the coefficients to minimize the cost function. The update rule for each coefficient β_j is:

$$\beta_j := \beta_j - \alpha \frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j}$$

where:

- α = learning rate (controls the size of the steps taken towards the minimum)
- $\frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j}$ = partial derivative of the cost function with respect to β_j

6.4.3 Computing the Gradient

The partial derivative of the cost function concerning β_j is:

$$\frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j} = -\frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)\, x_{ij}$$

where x_{ij} is the j-th feature of the i-th training example.

6.4.4 Gradient Descent Implementation Example

To make it clearer and to test this method by comparing it to the previous one we will Consider a dataset with three independent variables X_1, X_2, and X_3, and a dependent variable y. The dataset is:

y	X_1	X_2	X_3
4	2	3	1
6	3	5	2
5	4	4	3
8	5	6	2
7	6	5	4

6.4.5 Formulating the Model

The multiple linear regression model is given by:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

The predicted value $\hat{y}$ is given by:

$$\hat{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$$

6.4.6 Gradient Descent Algorithm

Gradient Descent iteratively updates the coefficients $\boldsymbol{\beta}$ to minimize the cost function. The cost function is the Mean Squared Error (MSE):

$$J(\boldsymbol{\beta}) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

where m is the number of training examples.
The update rule for each coefficient β_j is:

$$\beta_j := \beta_j - \alpha \frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j}$$

where α is the learning rate.

6.4.7 Computing the Gradient

The partial derivative of the cost function with respect to β_j is:

$$\frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j} = -\frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)\, x_{ij}$$

where x_{ij} is the j-th feature of the i-th training example.

6.4.8 Example Implementation

now assuming he previous example used for OLs , Assume the following initial coefficients:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Set the learning rate α:

$$\alpha = 0.01$$

1. **Initialize Parameters:**

$$\boldsymbol{\beta} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2. **Iterate Until Convergence:**
For each iteration, compute the gradients and update the coefficients:

$$\frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j} = -\frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)\, x_{ij}$$

Update the coefficients:

$$\beta_j := \beta_j - \alpha \frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j}$$

After several iterations, assume the coefficients converge to:

$$\boldsymbol{\beta} = \begin{bmatrix} 2.1 \\ 0.65 \\ 0.95 \\ 0.55 \end{bmatrix}$$

3. **Regression Equation**
The estimated regression equation using Gradient Descent is:

$$y2 = 2.1 + 0.65X_1 + 0.95X_2 + 0.55X_3$$

with a plot that looks like 2.12

6.4.9 Analysis

A slight difference was noticed after diving into both ways of finding coefficients and trying each away on the same example. The reason is that the OLs find the exact values of the coefficients that minimize the residual sum of squares so it finds it using calculus. Still, the gradient descent in the other hand finds it through iterating to optimize the algorithm used to minimize the cost function by updating the coefficients in the direction of the steepest descent.

6.4.10 Conclusion

Despite the increased accuracy of the Ordinary Least Squares (OLS) method stemming from its calculation nature, its complexity escalates with the addition of independent variables due to heightened matrix operations. Under such circumstances, the Gradient Descent method presents superior adaptability with multidimensional matrix calculus, thereby serving as a potentially preferable alternative.

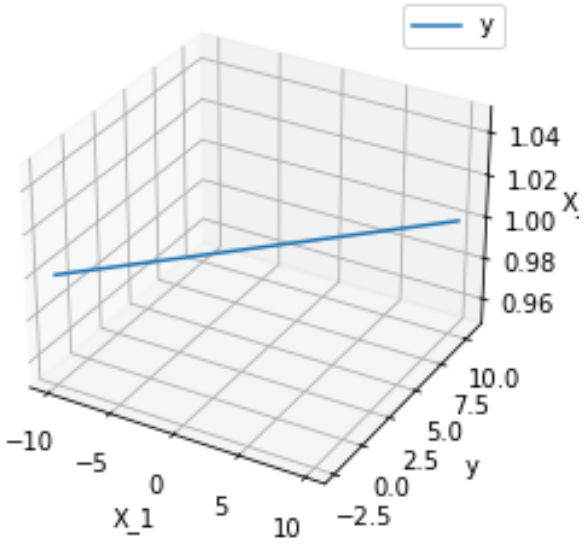


Fig. 2.12 plot of the y2 equation

6.5 Linear regression implementation

There isn't much to add regarding the previous algorithms to implement linear regression on any dataset, because all the steps remain the same from preprocessing to visualization.

CHAPTER 3

Applying classifiers on our data

1 Introduction

In this chapter, we will carry out the practical implementation of the various classification methods previously studied. Our objective is to construct multiple models using the three fundamental machine learning algorithms - Support Vector Machine (SVM), Linear Regression, and K-Nearest Neighbors (KNN). We will meticulously analyze the performance of these models by applying them to the preprocessed data that we already obtained from the previous study. Furthermore, we will apply these machine learning algorithms to predict outcomes based on a comprehensive range of parameters and features each time differently including range of motion, joint angles, session duration, trajectories, and paths. Through a systematic comparison of the outcomes and results generated by each algorithm, by evaluating, analyzing, and interpreting the results to determine the most effective approach among them. This comprehensive exploration will provide valuable comprehension of the real-world applicability of these machine-learning techniques and their potential to enhance our understanding of the data. By applying these models to the specific dataset, we will aim to uncover patterns and relationships that could have a potential contribution to advancing our research in this field. The rigorous evaluation of the performance of each algorithm will enable us to make informed decisions regarding the most suitable approach for our predictive modelling tasks. Through this detailed examination, we aspire to contribute to the ongoing discourse in the field of machine learning and its practical applications.

2 Data processing

One important step of any machine learning is to process the data, and in our case, we noticed that not all the patients have the same amount of sessions which resulted in our data being filled with zeroes or NAN values and also the problem of the mismatched length of our columns in the data frame, we will face this problem later on this chapter and to prevent it we transformed

our data frame from its original shape to a direct and easy to understand data frame and that was done by calculating the duration for all the patient and storing them into a separate file containing 2 columns one for the patient's names and other for the duration, the next step was using the DTW "Dynamic Time Warping" algorithm to handle the missing values for the different patients by extrapolation, in the end, we have a clean ready to use data frame as shown in 3.1

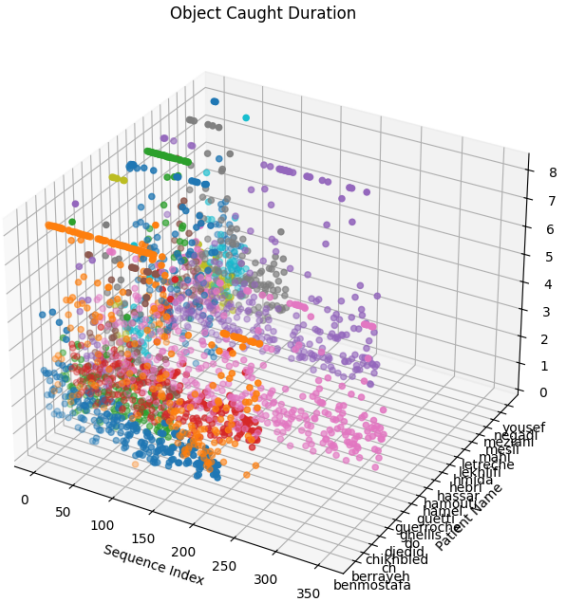


Fig. 3.1 Data frame plotted on 3 axis " duration patients names sequence index"

3 Implementation to our data

In this section, we will discuss the results we could obtain using the parameters we have from the datasets we were provided with. The data we have include angles from different joints (Outer Shoulder Angle, Upper Arm Angle, Elbow Angle Flexion, Elbow Angle Extension, Forearm Angle Pronation, Forearm Angle Supination), grip, sessions duration, path ratio, trajectories on 2D and 3D, duration of each sequence of the serious games practised by patients during the rehabilitation sessions.

The data we obtained after processing them will permit us to predict:

1. The range of motion for each angle (one and multiple patients): the ROM is the space in which the angle could move. It is affected by many parameters, including the bones' configuration within the joint, the joint capsule, ligaments, tendons, and the muscles acting on the joint. There are two main types of ROM, PROM or passive range of motion in which the therapist applies external force, and AROM or active range of motion which means that the patient or the person moves his limb or muscles by himself.

2. The range of motion of the upper limb for each patient: this is an outcome of using the previous range of motion for each angle and predicting the range of motion for the entire limb.

3. The healthiness of each angle for each patient: through the ROM of each angle we can determine whether that joint is healthy or not, and also the degree of healthiness and unhealthiness as a percentage.

4. The general healthiness of the upper limb for each patient: using the previous data processed from the model of the range of motion of each patient we can make an algorithm that predicts the healthiness of the upper limb.

5. The effectiveness of the physical therapy of each patient: also using both the duration and the upper limb range of motion we can predict both the effectiveness of the physical therapy for each patient and the effectiveness of physical therapy on each angle.

6. The number of sessions left for each patient to be healed: This model will predict approximately the number of sessions left for each patient to be healed and also how much time in each session and how much time is left to stay under therapy.

4 Implementing SVM

To successfully implement SVM, we need to carefully follow all the steps we discussed in the correct order.

4.1 Preprocessing files

To effectively utilize the Support Vector Machine (SVM) algorithm with our data, it is crucial to preprocess the data in a specific manner to ensure that each algorithm produces accurate and efficient results. The preprocessing process involves meticulously filtering, sorting, and analyzing the datasets to address empty rows, eliminate unwanted rows, and handle any duplicate entries.

The preprocessing phase for each model will involve the following steps: First, the algorithm will navigate through all the folders in the directory containing the files to identify the appropriate files containing the same exercises for each patient. Subsequently, it will transfer these files to a new directory, replicating the original folder structure and housing the relevant CSV files of the required exercises. Additionally, a filtering process will be applied to remove unwanted data and address any NaN (Not a Number) values by replacing them with real values or imputing them. The Kalman filter is our chosen way of reducing noise caused by sensors and other factors. This filter operates in two main steps. First, it uses a system model to predict the current state of the system based on the previous state estimate, thus it uses past data to anticipate the present. It also calculates the predicted covariance, which indicates the uncertainty associated with the prediction. Next, it incorporates new sensor measurements into the state estimates, later it calculates a Kalman gain to determine the weight given to the prediction and the measurement based on their respective uncertainties. The final state estimate is a weighted average of the predicted and measured states, while the covariance is updated to reflect reduced uncertainty after the measurement is incorporated.
Deleting NaN prblem will be done either by removing them in some models through a function called the" dropna()" which is an embedded function in the pandas library that allows s to remove all NaNs, or through replacing them in other models with real values using the function "fillna()" which allows us to impute them using the replacement method with constants or variables or more developed methods Mean/Median/Mode.

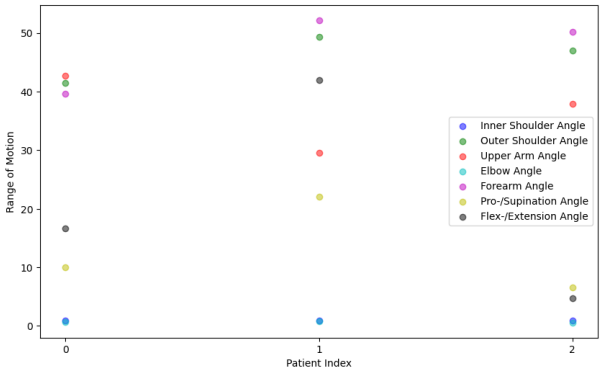


Fig. 3.2 Scatter plot of the range of motion for each available angle in the dataset for the same patient during three rehabilitation sessions

4.2 Using SVM on the RANGE OF MOTION

In this part of the project, we will focus on the first machine-learning algorithm, which is the Support Vector Machine (SVM). Our objective is to thoroughly analyze, observe, and interpret the outcomes produced by SVM. The purpose of this analysis is to compare these outcomes with those of other algorithms in the subsequent sections.

4.2.1 calculating the Range of motion for each angle

The process involves extracting the specific joint angles from the provided files. These joint angles are then utilized to forecast the range of motion for each angle. This is achieved by subtracting the minimum value from the maximum value of each angle, effectively determining the range of motion. This calculated range of motion serves as a fundamental component for subsequent prediction models. Once the range of motion is computed, it will be visualized through a scatter plot for further analysis., shown in 3.2 , this plot represents the range of motion for one patient during three different sessions the first session is the earliest and the other two were during the same day with 10 in between them, from the first sight to this plot we can see the difference that occurred and that the range of motion became less, analytically that means that the therapy wasn't that effective, later on, we will use the SVM to predict whether it is or it's not.

4.2.2 Calculating the Range of motion for the entire Upper limb

Now after we got the range of motion for each angle, we can use them up to get the range of motion for the upper limb, of course, the bigger it gets the more it's healthier,by having such information we created a model that predicts the ROM for the entire upper limb and it does that by, first taking the file data that contained the angles, calculates the range of motion then use the range of motion results that we got as parameters, it basically sums up all the ranges of motion that we got to get a variable or a comprehensive measure that represents the upper limb mobility, it uses these data as actual ROM of the upper limb and later it trains itself using it so later we have a model that can do the total range of motion without calculating the ROM for each angle which makes things faster less complex and more efficient for clinical use because therapist overall are interested in the general results rather than the angles themselves, the result as shown in 3.3, observing the Regression outcome plot of the model we can see how

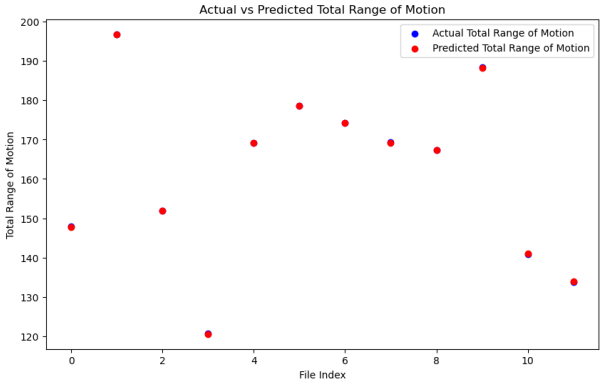


Fig. 3.3 Scatter plot of the total range of motion for a single patient during eleven different random sessions

identical are the actual total ROM with predicted ones which shows that the accuracy is pretty high,since the model is basic no accuracy test is needed because it's all obvious in the plot.

4.2.3 The healthiness of each angle for each patient

Now things will start to get more complex, using the range of motion for reach angle we will create a model that determines whether that joint is healthy or not, so we will have four classes (excellent, good, bad, severe), and the output should be binary, and to do this task we should define an appropriate threshold because it is a key parameter and it has a major effect on the accuracy of the model because it's considered as a reference for the classifiers, after searching and examining several articles that we will refer to later, and since there is no fixed or constant threshold for our task the threshold was taken as : and to interpret the results we will use the

Angle	**Threshold (degrees)**
Inner Shoulder Angle	45
Outer Shoulder Angle	75
Upper Arm Angle	150
Elbow Angle Flexion	120
Elbow Angle Extension	-10
Forearm Angle Pronation	80
Forearm Angle Supination	70

Tab. 3.1 Thresholds for Joint Angles

normal plot, a bar plot for each patient that contains how healthy is each angle. After processing the code for the first time and while using the normal SVM, it prompts that there is one single class concerning one of the angles for a patient's folder that contains 11 files which is false of course thus we need to change the threshold to something more accurate precise and meaningful, or skip that angle because the exercise that the patient is performing doesn't make that angle mobile, after doing that and changing the code to first skip the same or unique classes without touching the threshold we found the results shown in 3.4 where it clearly states that almost all of them were skipped as shown also in the confusion matrix 3.7, the only case of difference was in the pro/supination angle as shown in 3.5
The other ones were all alike 3.6

```
Unique classes for Inner Shoulder Angle: {'Bad'}
Unique classes for Outer Shoulder Angle: {'Bad'}
Unique classes for Upper Arm Angle: {'Bad'}
Unique classes for Elbow Angle: {'Bad'}
Unique classes for Forearm Angle: {'Bad'}
Unique classes for Pro-/Supination Angle: {'Bad', 'Good'}
Unique classes for Flex-/Extension Angle: {'Bad'}
Skipping training for Inner Shoulder Angle due to insufficient class diversity.
Skipping training for Outer Shoulder Angle due to insufficient class diversity.
Skipping training for Upper Arm Angle due to insufficient class diversity.
Skipping training for Elbow Angle due to insufficient class diversity.
Skipping training for Forearm Angle due to insufficient class diversity.
Skipping training for Flex-/Extension Angle due to insufficient class diversity.
```

Fig. 3.4 The output prompt that shows the skipped angles due to the unicity of classes

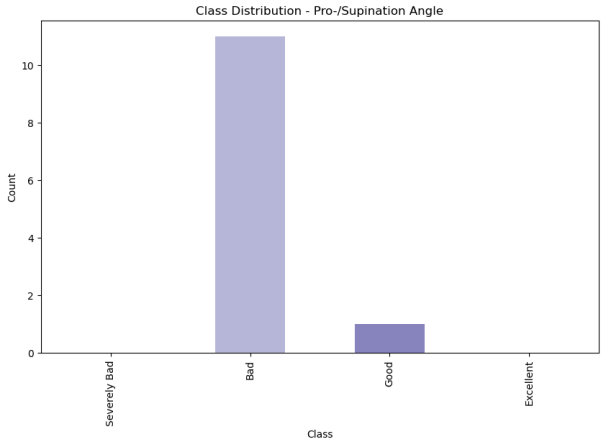


Fig. 3.5 The result of predicting the healthiness of the pro/supination angle for one patient but different sessions using the first model

After trying several other patients we have got the same result because of the insufficiency of class diversity, thus we must now change the threshold to something more specific, and to acquire that we will use another way of defining the threshold that will make our data more meaningful

The method used now to calculate the threshold after giving up on the information we got about it is based on Calculating the Mean and Standard Deviation, so For each angle we will calculate the mean and standard deviation and use them to define the threshold.

Classification	**Criteria**
Severely Bad	$\text{ROM} < \text{Mean} - 2 \times \text{Std Dev}$
Bad	$\text{Mean} - 2 \times \text{Std Dev} \leq \text{ROM} \leq \text{Mean} - \text{Std Dev}$
Good	$\text{Mean} - \text{Std Dev} < \text{ROM} < \text{Mean} + \text{Std Dev}$
Excellent	$\text{ROM} \geq \text{Mean} + \text{Std Dev}$

Tab. 3.2 Classification Criteria for Range of Motion

where mean is

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i i$$

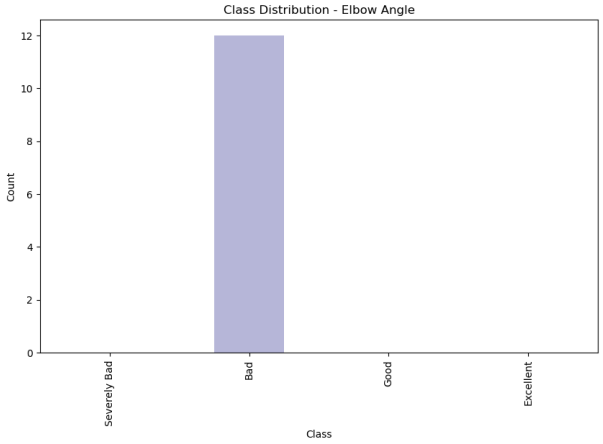


Fig. 3.6 Class distribution of the range of motion for the current model

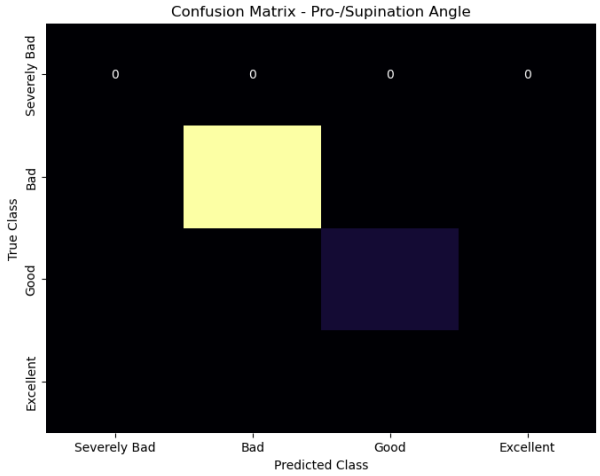


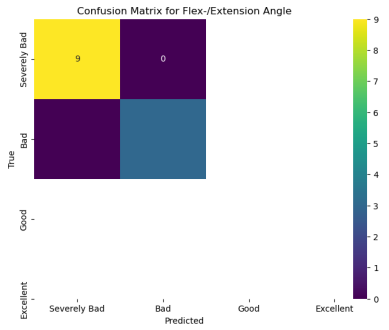
Fig. 3.7 Resulting confusion matrix of the skipped angles from the model that predicts the healthiness of each angle

and the dev is

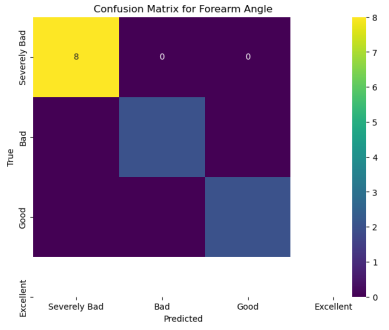
$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

This will ensure that the threshold won't be just constant but will the data flow, after implementing this method we've got something more interesting because the new outcomes were more meaningful and we have had some diversity except for the flex and extension, pro and supination angle because the exercises exclude them from the mobility, we can observe that in the confusion matrix 3.8

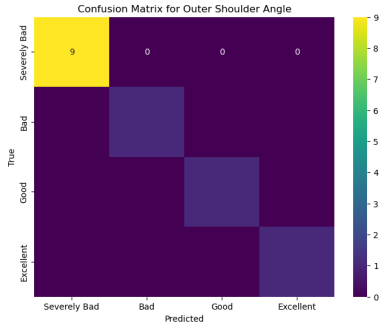
Inner shoulder angle range of motion



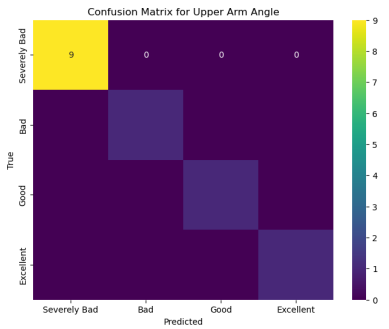
Inner shoulder angle range of motion



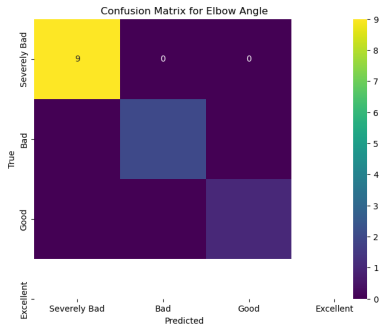
upper arm angle range of motion



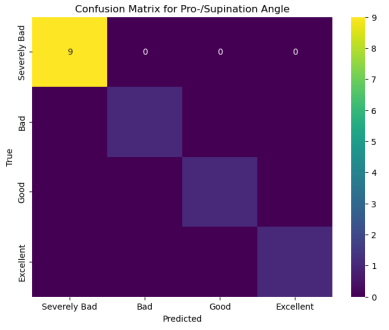
Elbow angle range of motion



Forearm range of motion



pro-supination angle range of motion

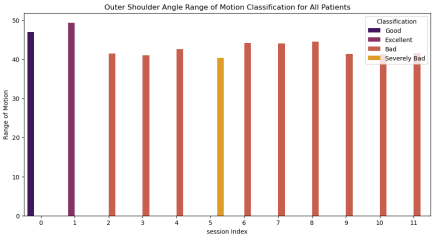


flex-extension range of motion

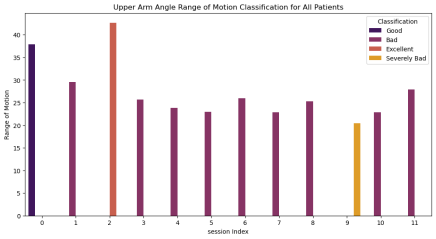
Fig. 3.8 Plots of the confusion matrix resulting from the new model that calculates the healthiness of each angle using the dynamic threshold

The analysis of the confusion matrix revealed that employing the dynamic threshold method yielded superior results compared to the static approach. This is attributed to the dynamic method's utilization of the entire dataset to derive a more precise threshold. However, it is essential to note that further evaluation is required by examining the bar plots generated by this model before deeming it an appropriate threshold.

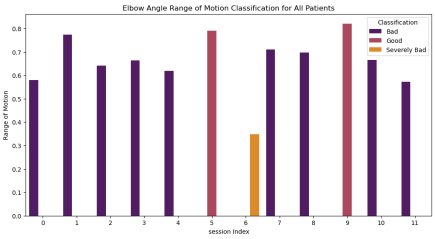
inner shoulder angle range of motion



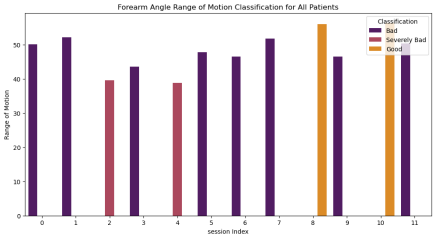
outer shoulder angle range of motion



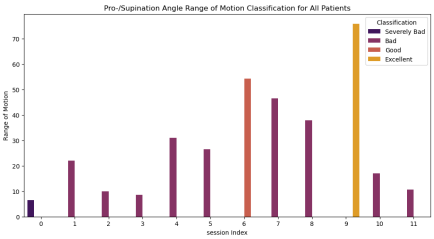
upper arm angle range of motion



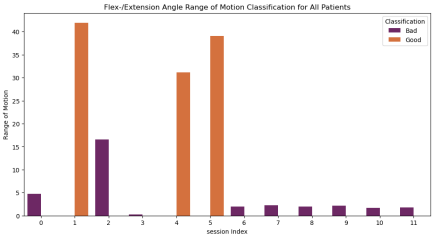
elbow angle range of motion



forearm range of motion



pro-supination angle range of motion



flex-8extension range of motion

Fig. 3.9 Bar plots resulting of the new model that calculate the healthiness of each angle using the dynamic threshold

Although it's the same exercise but different sessions for one person with an accuracy shown

in table 3.3

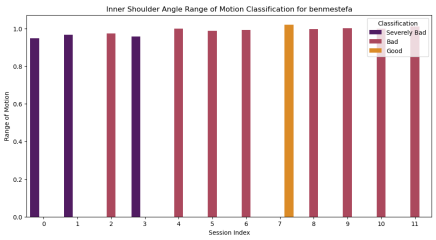
Angle	Accuracy
Inner Shoulder Angle	0.3333
Outer Shoulder Angle	0.6667
Upper Arm Angle	0.3333
Elbow Angle	0.6667
Forearm Angle	1.0000
Pro-/Supination Angle	0.6667
Flex-/Extension Angle	1.0000

Tab. 3.3 Range of Motion Accuracy for Each Angle

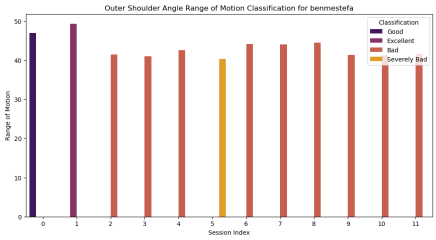
Even though there is a clear difference in the range of motion between therapy sessions, with older sessions showing different ranges compared to newer ones, this demonstrates the variability in the effectiveness of therapy for each individual. It's important to note that the variability in range of motion can be observed even when therapy sessions are ordered from oldest to newest.

To further evaluate the model for multiple patients, we plan to create an additional model that will iterate through all patient files, extract the relevant data, and use it to predict the range of motion for each angle of each patient. The aim is to classify the predicted data based on previous classes, providing a comprehensive understanding of the effectiveness of therapy across multiple patients.

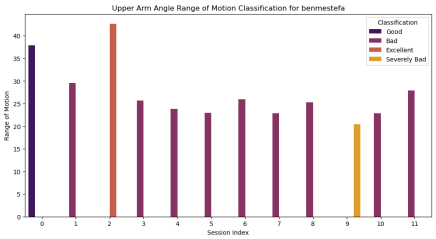
After completing the development of the new model, we successfully tested it on one patient. However when we attempted to apply it to the third patient, we encountered a setback due to insufficient data. Our specific error was "ValueError: The number of classes has to be greater than one; got 1 class." To address this issue, we had to modify the code to exclude the files containing only one class, ensuring the model's robustness and accuracy across all patient data. The results we got were overwhelming and strangely different from using the model for one patient, for example subject 1 3.10.



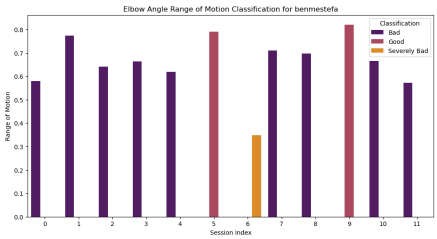
Inner shoulder angle range of motion



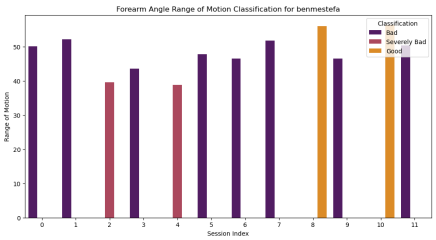
outer shoulder angle range of motion



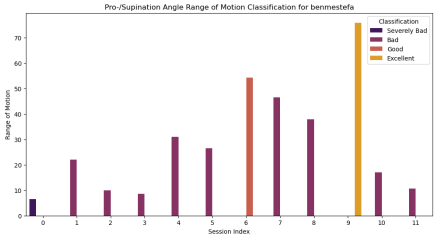
Upper arm angle range of motion



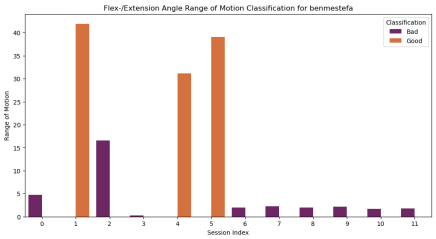
Elbow angle range of motion



Forearm range of motion



Pro-/supination angle range of mo- tion



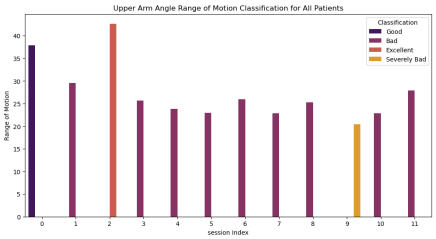
FLEX-extension angle

Fig. 3.10 bar plot results of different angles for subject 1

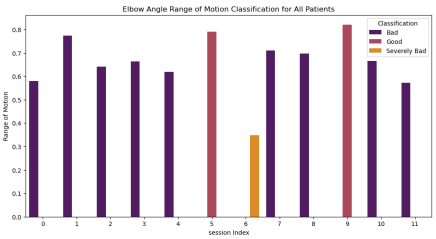
What's strange is that it has the same accuracy as before, and the bar plots were the same as shown in figure 3.13.

Inner shoulder angle range of motion

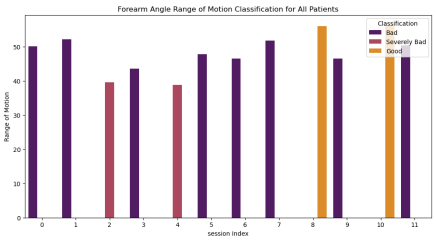
Outer shoulder angle range of motion



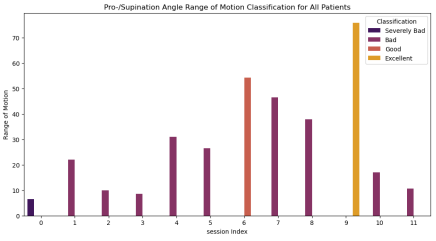
Upper arm angle range of motion



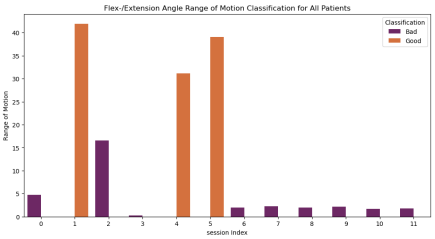
Elbow angle range of motion



Forearm range of motion



Pro-supination angle range of motion



Flex-extension range of motion

Fig. 3.11 Bar plot for the healthiness of each angle of subject1 using the model for every patient

4.2.4 Incremental learnigng

Up to this point, we have primarily relied on a standard basic SVM (Support Vector Machine) model. However, in our pursuit of higher accuracy, we are planning to adopt a new approach known as "Incremental Learning." This methodology involves the progressive enhancement of knowledge without the loss of previously acquired information. Essentially, it mirrors human learning patterns by assimilating new information over time while retaining and expanding

upon existing knowledge. This approach is particularly valuable in scenarios where the learning system needs to adapt to evolving data streams and where the retention of historical knowledge is essential for ongoing performance improvements.

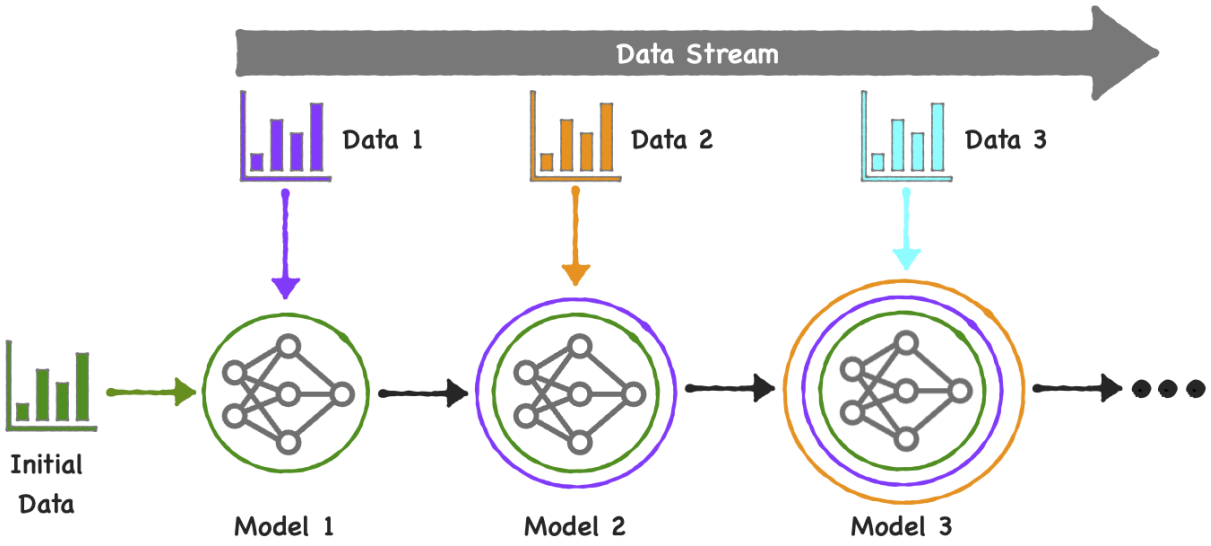
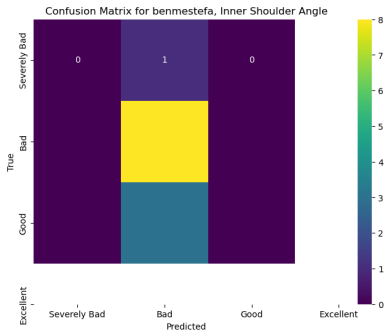


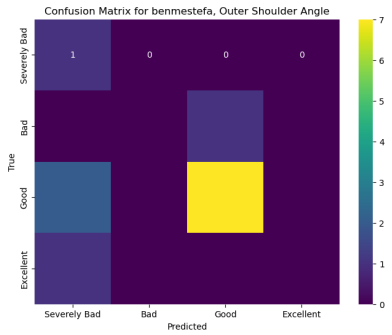
Fig. 3.12 Figure taken from MathWorks blog

4.2.5 Implementing incremental learning on our data

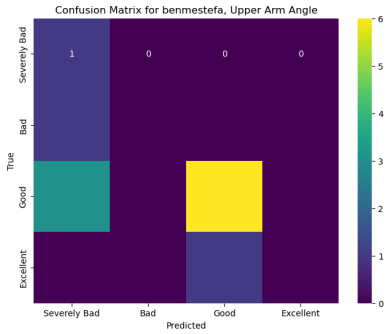
After applying this methodology on our model for the same patient using "SGDClassifier" which is an estimator that uses regularized linear models with stochastic gradient descent (SGD) learning, the results for subject 1 were, for the confusion matrix :



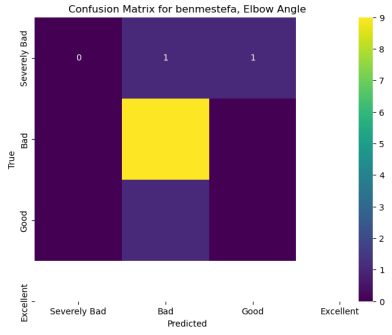
Inner shoulder angle range of motion



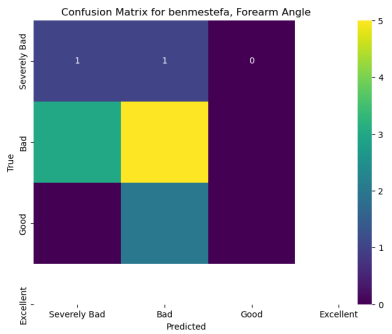
Outer shoulder angle range of motion



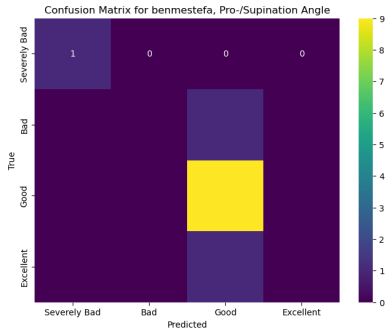
Upper arm angle range of motion



Elbow angle range of motion



Forearm range of motion



Pro-/supination angle range of motion

Flex-/extension range of motion

Fig. 3.13 Confusion matrices for the healthiness of each angle using SVM with incremental learning

as we can see in these upcoming confusion matrices 3.13 and after comparing them with the plots we had earlier from the previous models it looks radically different and more and more specific even the accuracy now has changed

Angle	Accuracy
Inner Shoulder Angle	0.6667
Outer Shoulder Angle	0.6667
Upper Arm Angle	0.5833
Elbow Angle	0.7500
Forearm Angle	0.5000
Pro-/Supination Angle	0.8333
Flex-/Extension Angle	1.0000

Tab. 3.4 Range of Motion Accuracy for Each Angle

For the inner shoulder angle, there was an increase, as well as for the upper arm angle and elbow angle, and also the pro/sup angle. The only angle that decreased was the forearm. But since it's called incremental learning, we have to observe it for the next patient's data, taking "subject 2" as an example. The old accuracy values using the old model before the incremental learning were: Accuracy for subject 2,

Tab. 3.5 Comparison of Old and New Accuracy Values of subject 2

Angle	Old Accuracy	New Accuracy
Inner Shoulder Angle	0.333	0.769
Outer Shoulder Angle	0.667	0.923
Upper Arm Angle	0.0	0.538
Elbow Angle	1.0	1.0
Forearm Angle	0.667	0.923
Pro-/Supination Angle	0.333	0.692
Flex-/Extension Angle	0.667	0.923

that's a big advancement in accuracy and the accuracy has changed a lot over patients which shows the success of using such a methodology in our model

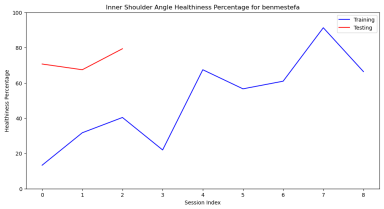
Our latest model has proved its efficiency in classifying the patient's joints ROM according to the classes we provided and also the threshold we used, the outcomes were binary so four classes (excellent, good, bad, and severely bad) but what about trying to get the results "healthiness" as a percentage In this section, we will start by choosing the proper threshold for each of the parameters that we'll be using in our models, the threshold is a key parameter and it has a major effect on the accuracy of the model because it's considered a reference for the classifiers,

Linear regression in machine learning is a fundamental algorithm that establishes a linear relationship between input variables and output variables.
It aims to predict the target value by minimizing the error between the predicted and true values.
Linear regression is widely used for prediction tasks and understanding the impact of input variables on the output.
The model calculates the best-fit line by updating coefficients through techniques like gradient

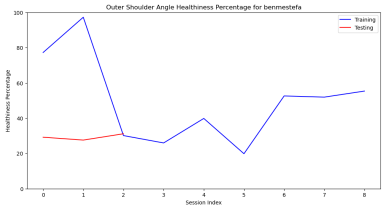
descent to minimize the Mean Squared Error (MSE).
This process ensures that the linear regression line accurately represents the relationship in the data.
Linear regression is a supervised learning technique beneficial for forecasting, time series analysis, and understanding cause-effect relationships. Companies need to predict trends, relationships like salary with experience, or associations like reckless driving and road injuries. Linear regression assumes linearity between dependent and independent variables, making it a powerful tool for predictive modeling in machine learning [17] [28] [6].

4.2.6 Healthiness percentage of each angle

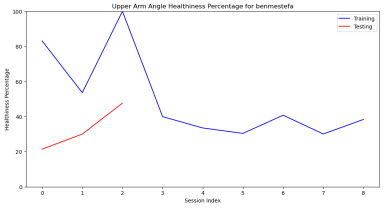
After employing the Support Vector Machine (SVM) model to ascertain binary outcomes from our dataset and observing incremental learning accuracy, we have decided to embark on creating a new model. This model will be designed to predict the healthiness level of each angle as a percentage, enabling us to monitor the profitability of each session. It's important to note that we will be retaining the dynamic threshold in our new model where the results where shown in Figure 3.14



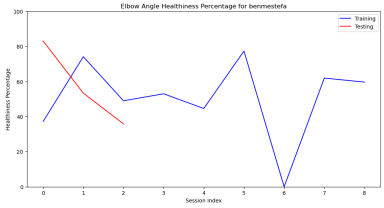
Inner shoulder angle range of motion



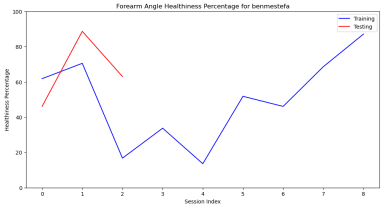
outer shoulder angle range of motion



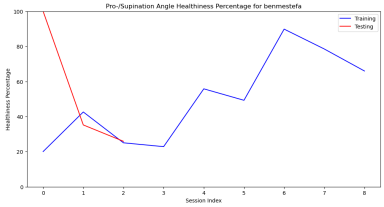
Upper arm angle range of motion



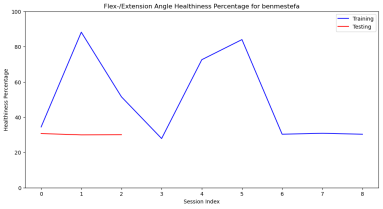
Elbow angle range of motion



Forearm range of motion



Pro-/supination angle range of mo- tion



FLEX-extension angle

Fig. 3.14 A line graph showing the training vs testing results for the healthiness of each angle of the upper limb for subject1 using SVM

After reviewing the plotted data, we observed that most of the angles' ranges of motion (ROMs) are improving. Sometimes there is a slight decrease or even spikes, which can be attributed to external factors affecting the patient. At times, lack of progress is due to the patient not working at that specific angle. However, overall, the results from the plots provide promising prospects.

To validate our observations, we referred to the accuracy table in 3.6. The table highlights variations in accuracies, with some angles having an accuracy of 1, while others have 0.3 or 0. This diversity is a result of class imbalance caused by dynamic thresholds.The other tests shown in table 3.8

Angle	Accuracy
Inner Shoulder Angle	0.3333
Outer Shoulder Angle	0.0
Upper Arm Angle	1.0
Elbow Angle	0.3333
Forearm Angle	0.6667
Pro-/Supination Angle	1.0
Flex-/Extension Angle	1.0

Tab. 3.6 Accuracies for different angles for subject1

Tab. 3.7 Metrics measurment for subject1

Angle	MAE	MSE	R^2
Inner Shoulder Angle	0.3333	0.3333	0.0
Outer Shoulder Angle	0.3333	0.3333	0.0
Upper Arm Angle	0.0	0.0	1.0
Elbow Angle	0.6667	0.6667	-2.0
Forearm Angle	0.3333	0.3333	-0.5
Pro-/Supination Angle	0.0	0.0	1.0
Flex-/Extension Angle	0.0	0.0	1.0

Tab. 3.8 MSE,MAE,R squared tests for the healthiness percentage model of subject1

4.3 Path ratio

The path ratio is defined as the efficiency of the path taken by the patient's hands in this case relative to the optimal path, in our case we created 2 classes healthy and unhealthy with the patient that has the closest path ratio value to 1 is the healthiest with a threshold of 0.4, it can be defined as follows:

$$\text{Path Ratio (PR)} = \frac{\text{Actual Path Length}}{\text{Optimal Path Length}} \tag{3.1}$$

We plotted the individual path or the trajectory of the sequences 3.15, and we can clearly see the position of the different objects appearing position, in our case the path ratio was already a given data, so there was no need to calculate it.

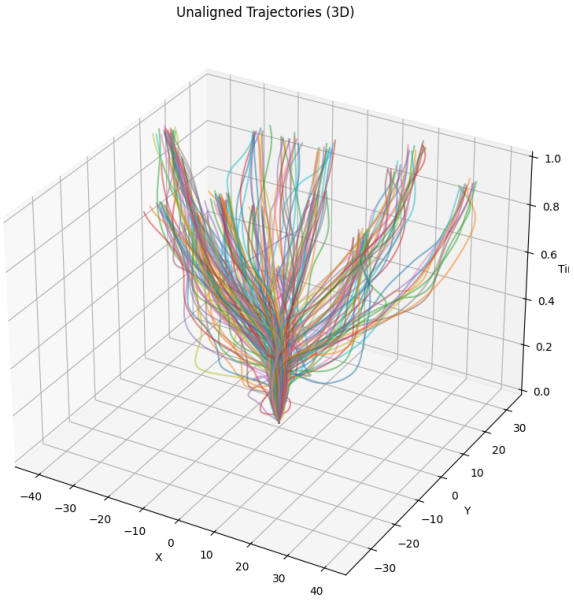


Fig. 3.15 Trajectory plot for 1 exercise using time, x and y coordinates for 1 patient.

As for the data processing, we followed the same procedure as the duration to simplify the task, here is the confusion matrix that shows the distribution of the predicted sequences into 2 classes healthy and unhealthy of the SVM model 3.16

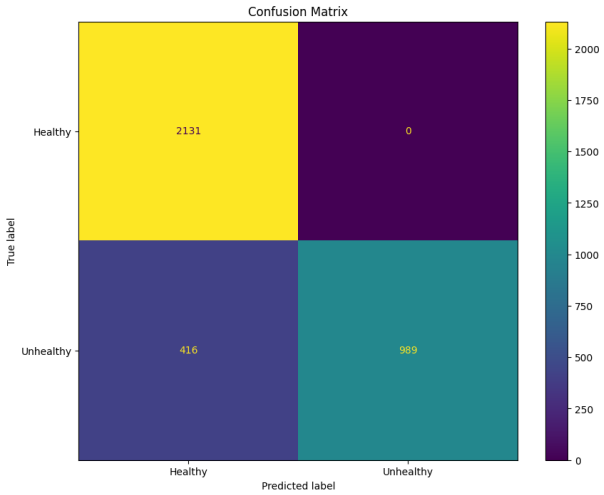


Fig. 3.16 confusion matrix for the SVM model with a linear kernel

as for the SVM model shown in 3.17

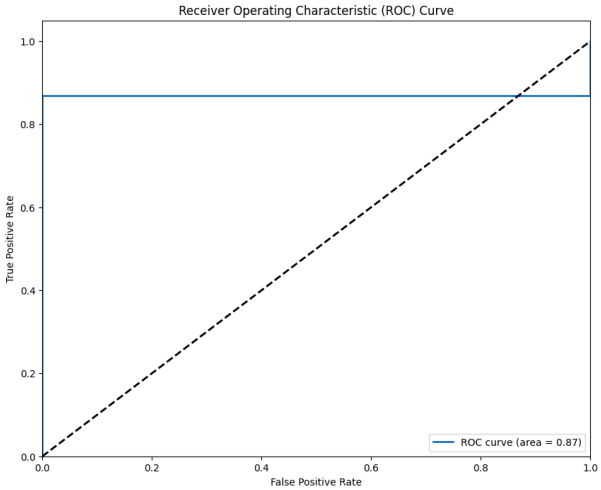


Fig. 3.17 ROC curve for the SVM model with a linear kernel

4.4 Using SVM on Duration

In our case, each patient has multiple files containing the complete data of a single session sometimes some are missing but they were rectified using specific algorithms. Each file consists of approximately 18 to 20 sequences. It's important to note that a sequence represents an action taken to complete a task in a given exercise. For example, it could involve an object appearing on the screen and the patient moving their arm to capture it. To calculate the duration of a sequence, we subtract the time of the object's appearance from the time of its disappearance. This time represents the time taken to complete a sequence. For our classes, we have chosen specific thresholds.

- <2 seconds: Excellent duration.
- $2 \leq$ duration <4 seconds: Good duration.
- $4 \leq$ duration <6 seconds: Acceptable duration.
- $6 \leq$ duration <8 seconds: Bad duration.

4.4.1 Kernelling

In our model training,this time and same four classes as before we divided the data from each patient into two halves - one for training the model and the other for testing it. We experimented with different kernels, including linear and RBF. For visualization, we used a confusion matrix to display the class distribution as shown in . 3.18

Fig. 3.18 Confusion matrix for the linear kernel of the SVM model shows the distribution of the predicted sequences for different patients into 4 different classes

Another model evaluation method commonly used is the Mean Squared Error or (MSE) , previously explained this is a common metric used for evaluating the model accuracy, the error here is the difference between the actual and predicted values, the lower the MSE the more accurate the model and for our model:

- **the Mean Squared Error:** 0.267
- **the model accuracy was :** 0.732

Now we will jump to The ROC curve or Receiver Operating Characteristic curve which is mainly adapted for accessing the model accuracy by plotting the the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings,as shown in 3.19.
Area Under the Curve (AUC)
The AUC quantifies the overall ability of the model to discriminate between positive and negative classes.
AUC Values:

- 0.5: No discrimination (random classifier).
- $0.5 < \textbf{AUC} < 0.7$: Poor performance.
- $0.7 \leq \textbf{AUC} < 0.8$: Fair performance.
- $0.8 \leq \textbf{AUC} < 0.9$: Good performance.
- $0.9 \leq \textbf{AUC} \leq 1$: Excellent performance.

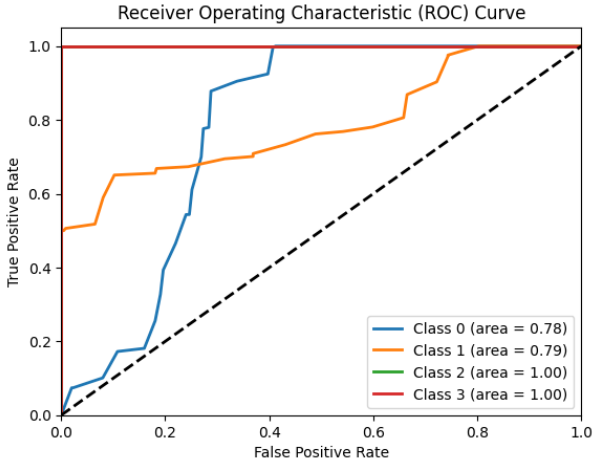


Fig. 3.19 ROC curve of a single parameter " duration" for the 4 classes

the decision boundary layer plot is a visualisation of our data being separated to 4 classes, depending on the kernel used it can differ, as shown in 3.20, and for the RBF kernel as shown in 3.21

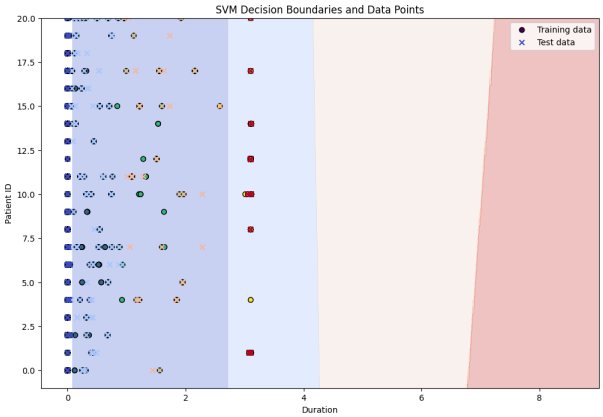


Fig. 3.20 Boundary layer for the linear kernel

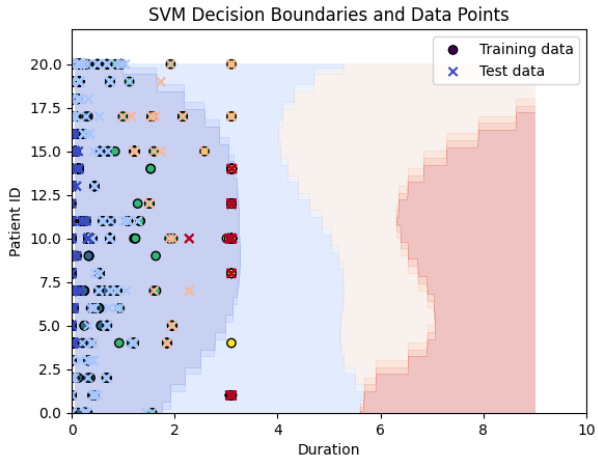


Fig. 3.21 Boundary layer for the RBF kernel

We observed that when working with a single parameter, the linear kernel yielded the best results in terms of accuracy and execution time. On the other hand, the RBF kernel, while slightly less accurate, significantly increased the execution time.

5 Implementing KNN

Earlier, we talked about how applying KNN won't be much different from SVM when it comes to preprocessing. So, let's move straight to the point. The models created in KNN are the same as SVM in terms of the required outcomes and input parameters.

5.1 Duration

The KNN model followed the same steps as the SVM only this time there were no kernels, except for the distance metrics which both gave the same result, the confusion matrix as shown in 3.22

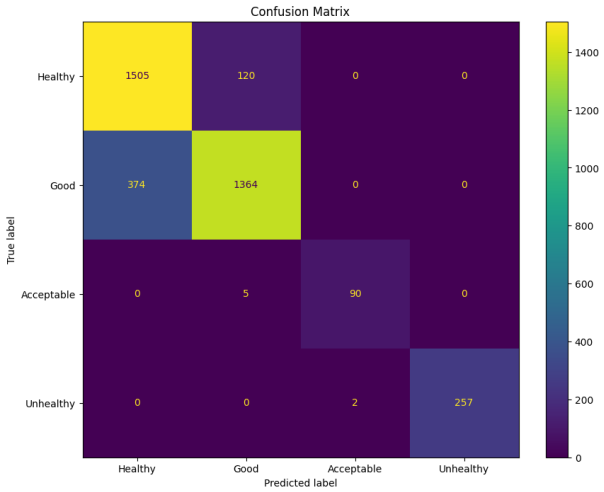


Fig. 3.22 Confusion matrix for the Duration model using KNN

- **the MSE :**0.13

- **the accuracy of the model reached :**0.865

as for ROC curve it in 3.23

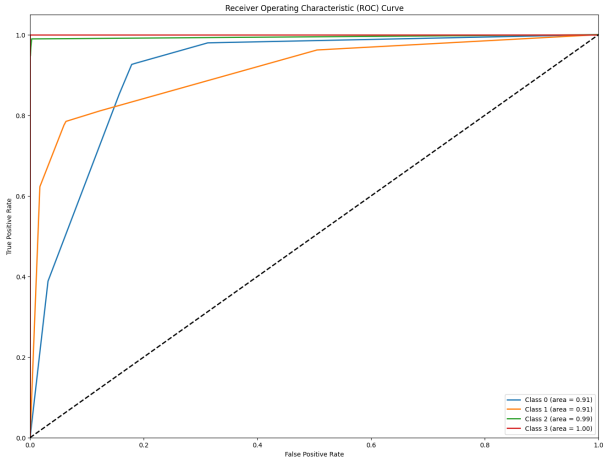


Fig. 3.23 ROC curve for the Duration model using KNN

and as for the boundary layer it in 3.24

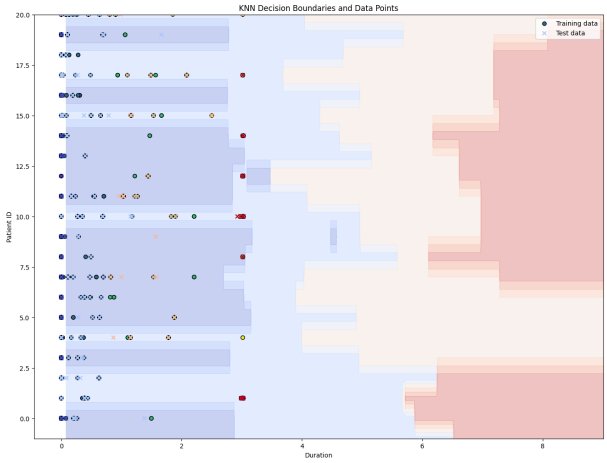


Fig. 3.24 boundary layer for the Duration model using KNN

5.2 Path Ratio

After studying the duration algorithm outcomes using SVM, we will perform the same analysis using the KNN model, this time with the same classes, to achieve the thesis goal. The results will be plotted as a confusion matrix. 3.25

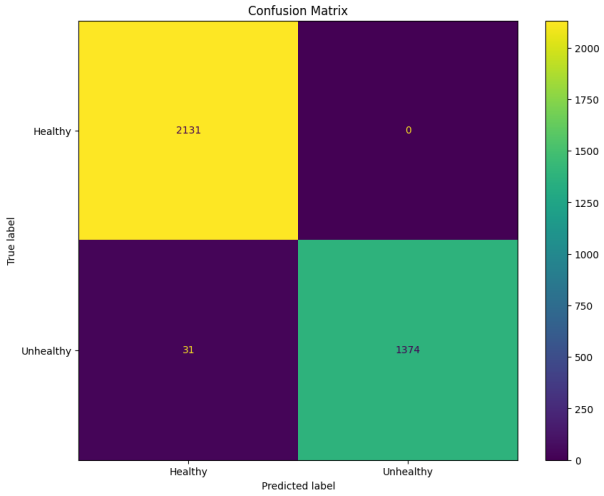


Fig. 3.25 confusion matrix for the KNN model

Also the ROC curve shown in 3.26

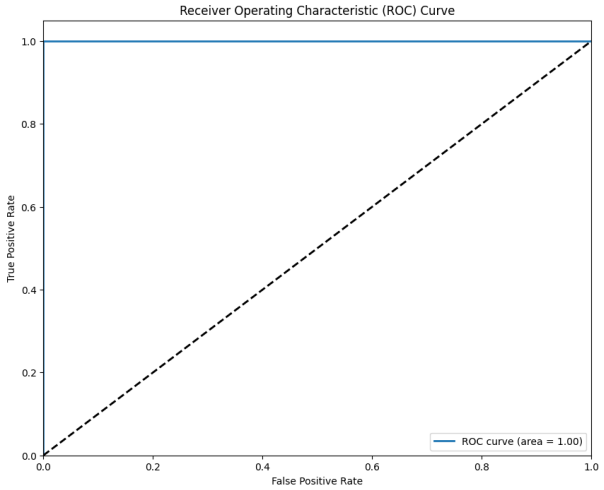


Fig. 3.26 ROC curve for the SVM model with a linear kernel

5.3 Using KNN on the RANGE OF MOTION

In the previous section, we discussed calculating the range of motion and the potential outcomes using the SVM algorithm. Now, we need to replicate the same process, but this time using the KNN algorithm. For a valid comparison, we should use the same code and generate the same plots as before.

5.3.1 Calculating the Range of motion for each angle

In calculating the range of motion, it is unnecessary to employ complex machine learning algorithms as the task can be accomplished using simple mathematical calculations. The range of motion for each patient can be computed individually and stored in a designated folder. This approach will enhance the efficiency and robustness of the algorithm. These stored files can

be utilized across various algorithms, eliminating the need for repetitive iteration over files for recalculations.

5.3.2 Calculating the Range of motion for the entire Upper limb

The results obtained using SVM were crucial in assessing each patient's health status and tracking their progress after each session. I wonder if using KNN instead would yield different results for predicting the overall health of the upper limb, or else affect the other outcomes. to answer that we will experiment with it on the same data as SVM.
After we coded a new algorithm and ran it on the same dataset as the SVM. As expected, the outcome was different from that of the SVM. However, the difference was much more drastic than anticipated, particularly in the predicted range of motion, as illustrated in the scatter plot of the KNN algorithm. 3.27

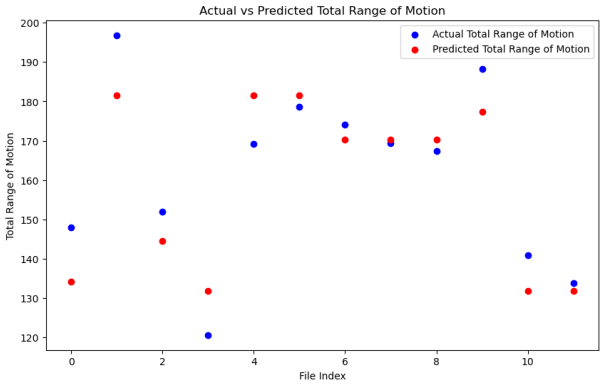


Fig. 3.27 scatter plot of the actual and predicted ROM of the entire upper limb using KNN

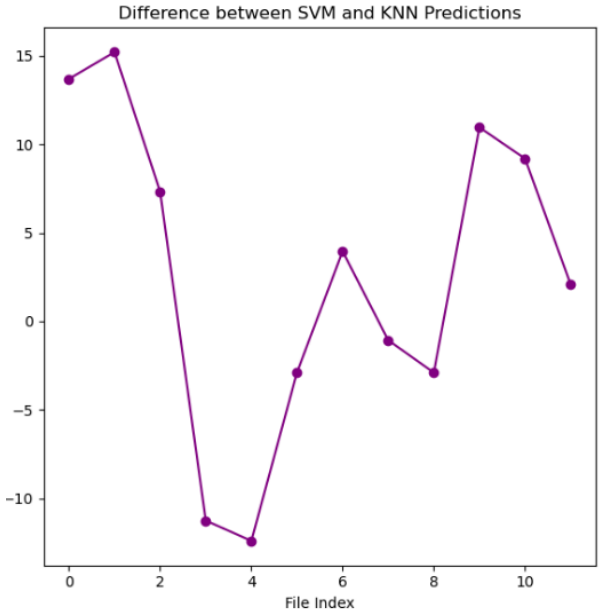
the difference is crystal clear, but to delve deeper, we will utilize another plot that represents the difference between the results as a scattered line plot graph.3.28 this figure clearly shows the big difference between both of the outcomes, even if we were restricted to coding using the same structure to avoid errors.

5.3.3 The healthiness of each angle for each patient

Let's utilize the range of motion that was calculated for each angle in the previous sections to assess the condition of each patient as either excellent, good, bad, or severe. We will apply the same threshold to every angle of each patient just as we did with the SVM using the same files. However, this time, we will omit the insufficient data, as it yields the same results, and the process for skipping or bypassing it remains unchanged.

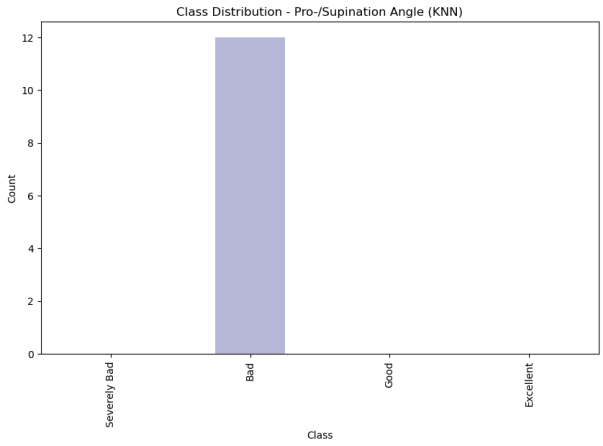
The only noticeable difference this time was in the pro/supination angle, where it consistently indicated that the range of motion was bad across all sessions, contrary to the results obtained from the SVM. 3.29 also for the confusion matrix of the skipped angles shown in 3.31

To verify if we could achieve similar results as the Support Vector Machine (SVM) using the same algorithm with different data, we conducted experiments on sessions from other patients. As a result, we found that the same algorithm produced consistent results for all patients, leading us to adopt the dynamic threshold employed in the SVM section.



H

Fig. 3.28 line plot of the difference between KNN outcome and SVM outcome full ROM



h

Fig. 3.29 The result of predicting the healthiness of the pro/supination angle for one patient but different sessions using KNN

After incorporating the dynamic threshold method into our algorithm, we achieved more meaningful and promising results. shown as confusion matrices in Figure 3.30

These confusion matrices represent a good measurement of the difference between both of the algorithms because they were way so different from each other,and to measure the accuracy we applied an accuracy test in which we found

Tab. 3.9 Accuracy of KNN Classifier for Different Angles

Angle	KNN Accuracy
Inner Shoulder Angle	0.75
Outer Shoulder Angle	0.75
Upper Arm Angle	0.75
Elbow Angle	0.75
Forearm Angle	0.67
Pro-/Supination Angle	0.75
Flex-/Extension Angle	1.00

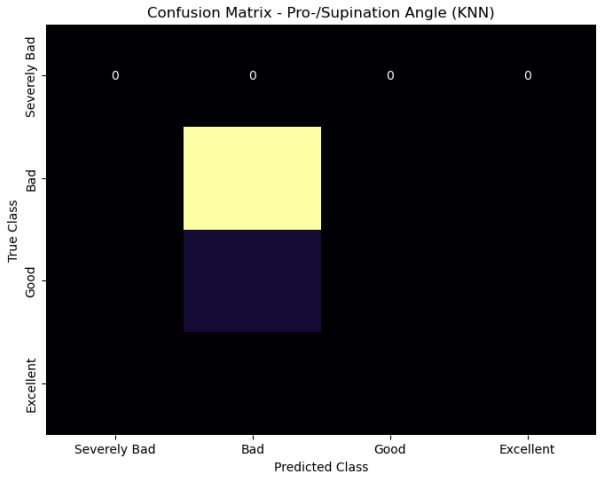


Fig. 3.31 Resulting confusion matrix of the skipped angles from the model that predicts the healthiness of each angle using knn

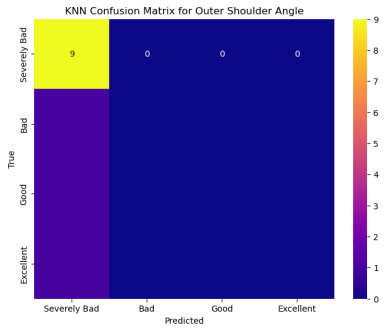
5.3.4 Incremental learnigng

Now, let's move on to one of the most important parts of coding, which is the implementation of incremental learning. This method is used to continuously feed the model with incoming data to improve its accuracy. However, in our case, KNN is not inherently incremental. Therefore, we will simulate incremental learning by retraining the classifier with accumulated data for each upcoming file.

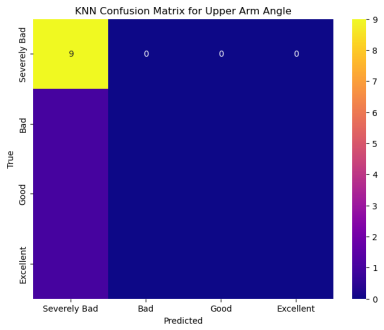
After simulating incremental learning on KNN, almost all results changed, including the accuracy and the confusion matrices for every angle. This is illustrated in the example in 3.34, where some of the changes resulted in more true positives example 3.33, while others only affected individual instances.example 3.32

H

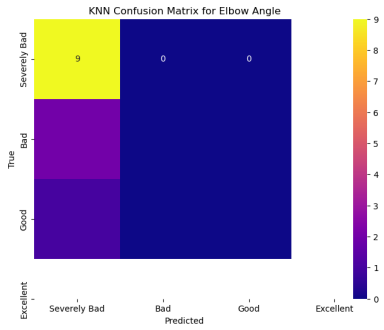
Inner shoulder angle range of motion



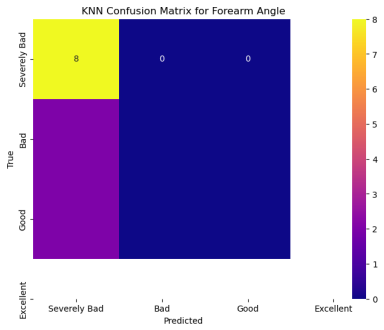
Outer shoulder angle range of motion



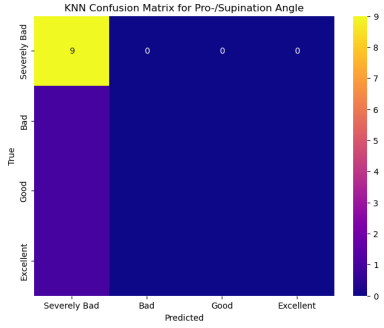
Upper arm angle range of motion



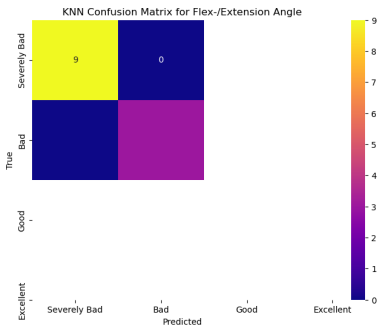
Forearm range of motion



Pro-/supination angle range of mo- tion



FLEX-extension angle



flex-8extension range of motion

Fig. 3.30 confusion matrices of the healthiness of each angle using knn with incremental learning

Inner shoulder angle range of motion

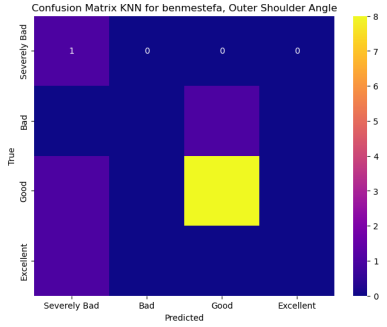
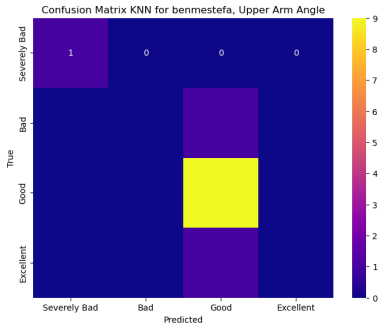


Fig. 3.32 outer shoulder angle range of motion



Upper arm angle range of motion

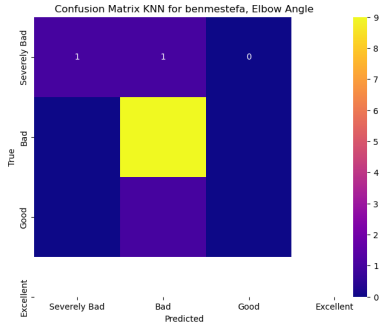
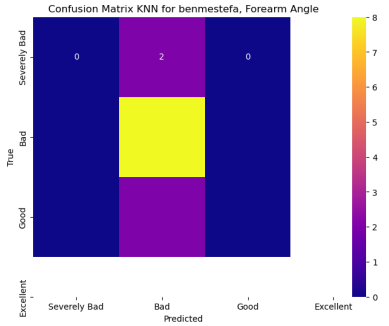
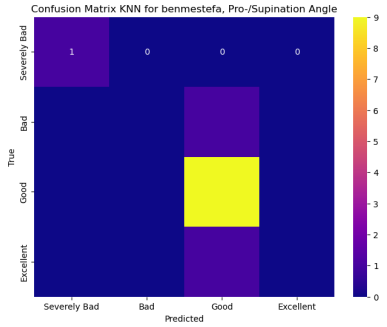


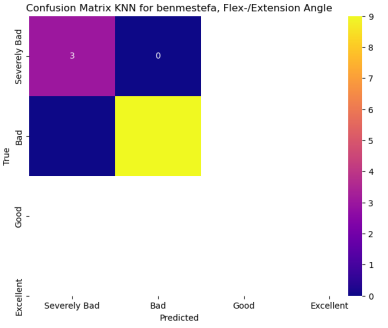
Fig. 3.33 Elbow angle range of motion



Forearm range of motion



Pro-/supination angle range of mo- tion



FLEX-extension angle

Fig. 3.34 Resulting confusion matrices of using KNN with incremental learning and dynamic threshold

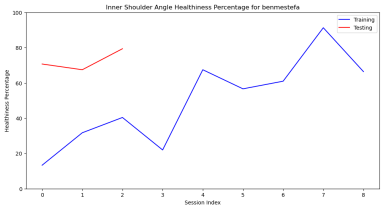
with the new accuracy being 3.10, which shows a slight improvement for most of the angles and a decrease in one which is probably due to the lack of data.

Angle	Accuracy
Inner Shoulder Angle	0.75
Outer Shoulder Angle	0.75
Upper Arm Angle	0.8333
Elbow Angle	0.8333
Forearm Angle	0.6667
Pro-/Supination Angle	0.8333
Flex-/Extension Angle	1.0

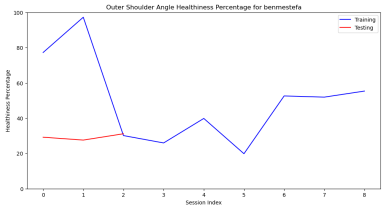
Tab. 3.10 Accuracies for different angles for subect1 after incremental learning simulation KNN

5.3.5 Healthiness percentage of each angle

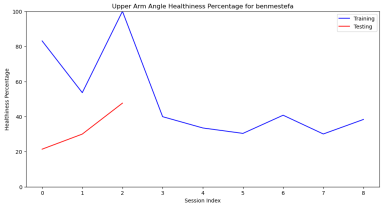
After going through the incremental learning simulation phase, we now step into our final model for kNN, which calculates the healthiness percentage for each angle. After finishing the coding phase and observing the outcomes, remarkably we can observe how well it works on our data in results shown in 3.35.



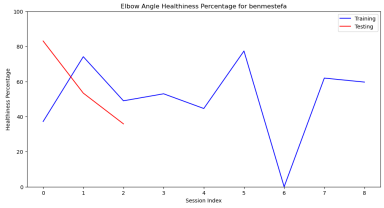
Inner shoulder angle range of motion



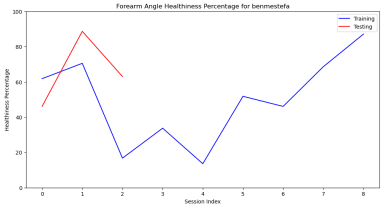
outer shoulder angle range of motion



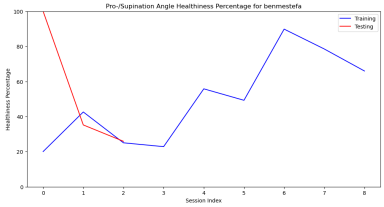
Upper arm angle range of motion



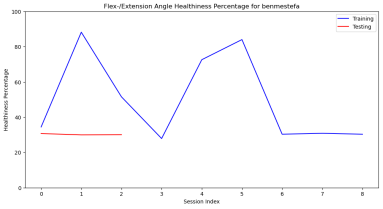
Elbow angle range of motion



Forearm range of motion



Pro-/supination angle range of mo- tion



FLEX-extension angle

Fig. 3.35 A line graph showing the training vs testing results for the healthiness of each angle of the upper limb for subject1 using KNN

To numerically test the reliability of our model, we conduct an accuracy test. The results are shown in3.11

The accuracy table above shows the prediction accuracy for each angle. Upon reviewing the numbers, it's evident that the accuracy rates vary; some angles have high accuracy while others have low accuracy. Overall, the accuracy could be considered medium. This variation in accuracy may be due to the in-variance of the data.

Angle	Accuracy
Inner Shoulder Angle	0.333
Outer Shoulder Angle	0.333
Upper Arm Angle	0.667
Elbow Angle	1.000
Forearm Angle	0.333
Pro-/Supination Angle	1.000
Flex-/Extension Angle	1.000

Tab. 3.11 Accuracy's for subject 1 on Different Angles in healthiness percentage model knn

But what's strange is that these results are identical to the ones of SVM thus we will do the additional tests ,MSE, MAE,and R squared shown in 3.12

Tab. 3.12 Metrics for Each Angle

Angle	Accuracy	MAE	MSE	R^2
Inner Shoulder Angle	0.333	0.667	0.667	0.000
Outer Shoulder Angle	0.333	0.667	0.667	0.000
Upper Arm Angle	0.667	0.333	0.333	0.000
Elbow Angle	1.000	0.000	0.000	1.000
Forearm Angle	0.333	0.667	0.667	-2.000
Pro-/Supination Angle	1.000	0.000	0.000	1.000
Flex-/Extension Angle	1.000	0.000	0.000	1.000

6 Implementing Linear regression

After covering KNN and SVM, we will now focus on studying linear regression models that we will be using on our data. This data is the same as before, and we will organize it in the same format as in the previous sections to ensure a fair comparison.

6.1 Duration and Path Ratio

For the linear regression model, we combined two parameters in one model where we used Duration and Path Ratio using logistic regression since, as we said, linear regression can't be used for classification.

First of all, we will discuss the Decision Boundary Plot. This plot shows the decision boundaries for the logistic regression classifier, with data points colored based on their actual classes. The x-axis represents the "Duration," while the y-axis represents the "Path Ratio."

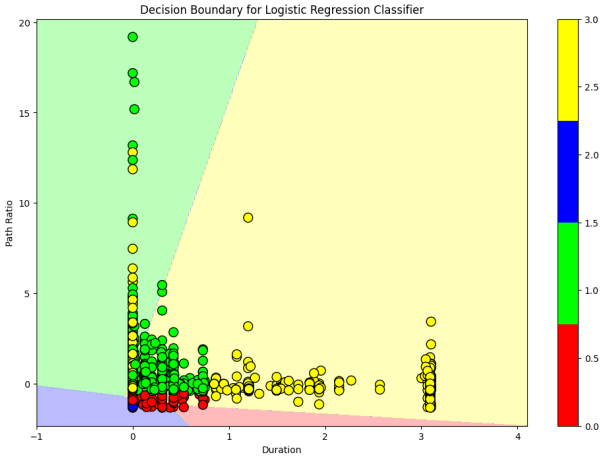


Fig. 3.36 The decision boundary for the logistic regression method.

The decision boundaries result seem to be linear, which is expected for logistic regression. However, there's a significant overlap between classes, especially around the origin (0,0). This might be causing the classifier to struggle with correctly separating the classes.

Later on, to measure the accuracy of our model, we had to run confusion matrix tests through plotting them. The confusion matrix shows how well the model is performing across the different classes. The true labels are on the y-axis, and the predicted labels are on the x-axis. The classes are labeled as "Fast Healthy," "Fast Unhealthy," "Slow Healthy," and "Slow Unhealthy."

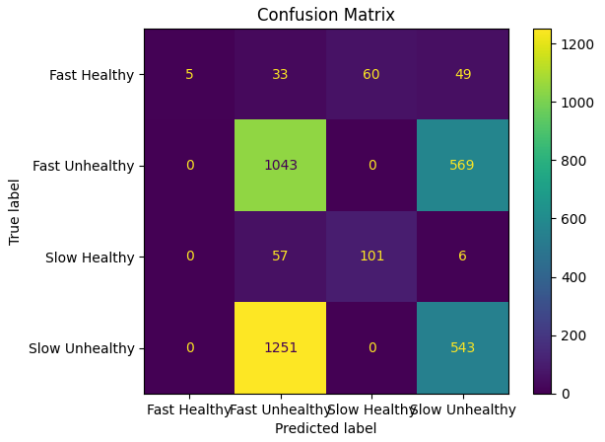


Fig. 3.37 Confusion matrix for the logistic regression model.

After doing the plot, we observe that the model has a high number of correct predictions for the "Fast Unhealthy" class (1,043) but struggles with others, especially "Slow Unhealthy," where it misclassifies a large number of samples (1,251). This indicates that the model is biased toward predicting certain classes over others. The problem is that it indicates possible class imbalance, especially with the misclassification of the "Slow Unhealthy" class, which may need to be addressed either by balancing the dataset or adjusting the model.

Lastly, and to confirm the previous results, we plotted an ROC curve. The Receiver Operating Characteristic (ROC) curve illustrates the true positive rate against the false positive

rate for each class. The area under the curve (AUC) provides insight into how well the model distinguishes between the classes.

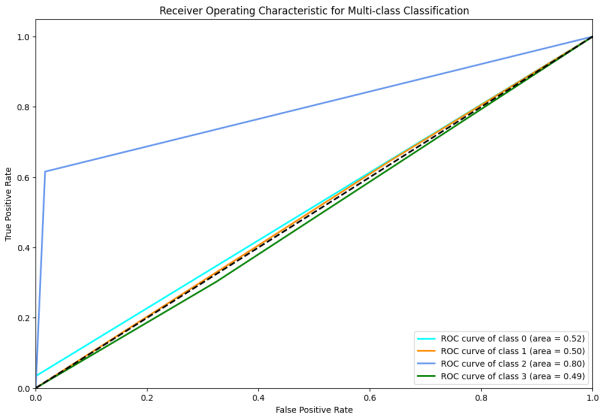


Fig. 3.38 ROC curve for the logistic regression model.

What we obtained is that the ROC curves show varying performance across classes, with the AUC ranging from 0.49 to 0.80. The model performs best for "Class 2" (Slow Healthy) with an AUC of 0.80, while it performs poorly for "Class 3" (Slow Unhealthy) with an AUC of 0.49. To sum up, the overall accuracy of the model was found to be 0.45, which is relatively low. This suggests that the model may not be well-suited to this particular classification task.

6.2 Using Linear regression on the RANGE OF MOTION

We will continue studying the potential results of the range of motion, but this time we will use a linear regression model instead of KNN or SVM. Calculating the range of motion for each angle is simple; it involves basic math. We have already covered and calculated this, so we will skip that part.

6.2.1 Calculating the Range of motion for the entire Upper limb

After creating a linear regression model for our data and running it, we obtained some strange results, as shown in Figure 3.39. Initially, we thought it might be a programming or compiling error because the blue scatter representing the actual range of motion was not visible. However, when we changed the code to draw a line between the scatters in both colors, we got even stranger results. The actual and predicted instances were almost identical, and sometimes completely identical, as shown in the table in 3.13. This phenomenon is known as underfitting, and it occurs when the model is too simple. It performs well on the training data, but poorly on test data. To confirm whether this is the case and not just a data-related problem, we tried the data of "subject 2", which unsurprisingly gave the same results identically, as shown in 3.41, so now we confirmed that it's underfitting, But maybe it's just for this model, thus we will continue with the other required models.

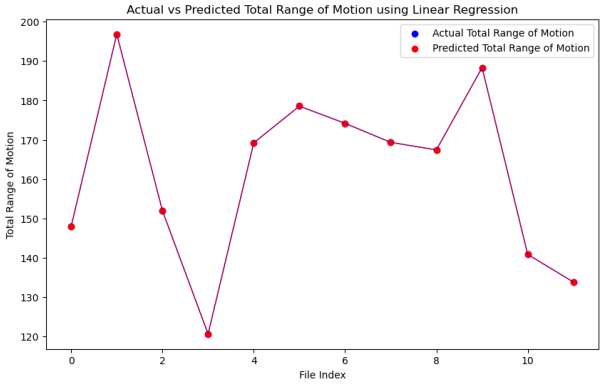


Fig. 3.40 scatter Plot with a line showing the actual and predicted total range of motion for one patient during multiple sessions

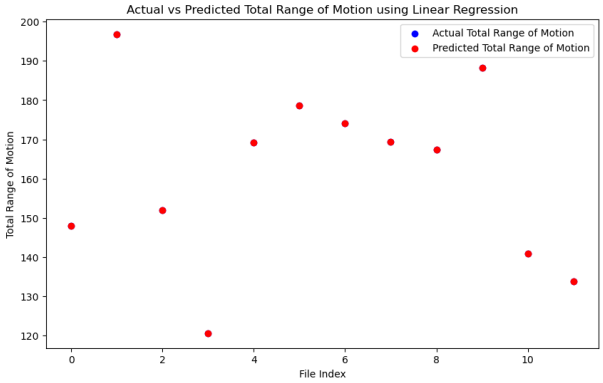


Fig. 3.39 Plot showing the actual and predicted total range of motion for one patient during multiple sessions using linear regression

Index	Actual Total Range of Motion	Predicted Total Range of Motion
1	147.919	147.919
2	196.819	196.819
3	151.988	151.988
4	120.653	120.653
5	169.213	169.213
6	178.544	178.544
7	174.168	174.168
8	169.340	169.340
9	167.428	167.428
10	188.326	188.326

Tab. 3.13 Actual vs Predicted Total Range of Motion

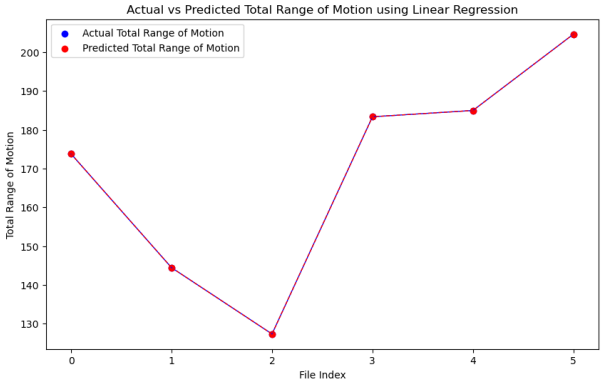


Fig. 3.41 Plot showing the actual and predicted total range of motion for one patient during multiple sessions using linear regression for "sbject2"

6.2.2 The healthiness of each angle for each patient

The linear regression model is suitable for predicting continuous variables but not categorical variables, so it's not appropriate for our case. Its mathematical properties make it unsuitable for binary predictions. To address this issue, we may need to consider using a different regression technique, such as logistic regression, for binary outcomes. The resulting confusion matrix is shown in Figure 3.44. The bar plots all displayed the same result, indicating that all sessions of subject 1 were bad, as shown in Figure 3.42, except for the pro/supination angle, which is depicted in Figure 3.43. The reason for this is the threshold, the same as in previous sections with KNN and SVM. We will proceed with the same solution, which is the dynamic threshold that gave us the results shown as confusion matrices in Figure **??**, where it clearly states the difference between the static threshold taken the first time and the dynamic threshold used now.

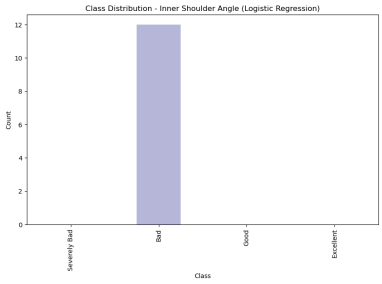


Fig. 3.42 bar plot of the healthiness of inner shoulder angle using logistic regression

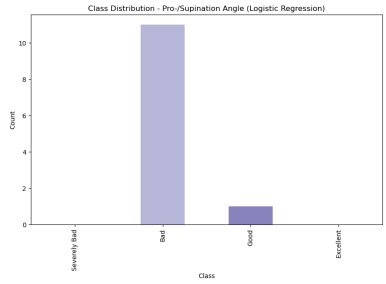


Fig. 3.43 bar plot of the healthiness of pro/supination angle using logistic regression

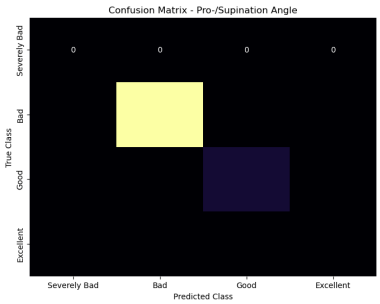


Fig. 3.44 confusion matrix of the healthiness of each angle using logistic regression

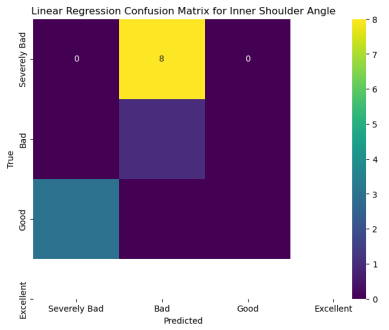
To further test our model we ran an accuracy test and we got the result shown in the tables 3.14

Angle	Accuracy
Inner Shoulder Angle	0.92
Outer Shoulder Angle	1.00
Upper Arm Angle	1.00
Elbow Angle	1.00
Forearm Angle	1.00
Pro-/Supination Angle	1.00
Flex-/Extension Angle	1.00

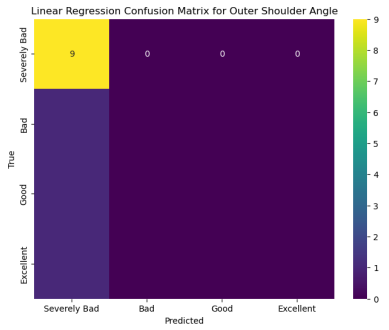
Tab. 3.14 Model Accuracy by Angle before incremental learning logistic regression

6.2.3 Incremental learnigng

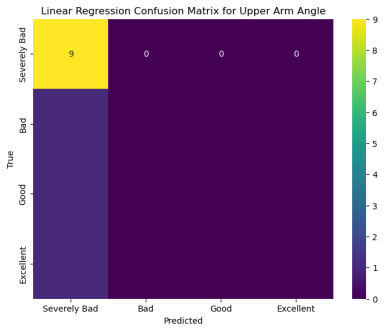
To delve more into the accuracy of our model and to compare it with the previous ones, we will use incremental learning. When we used it, we obtained the results below in the confusion matrices shown in Figure **??**.



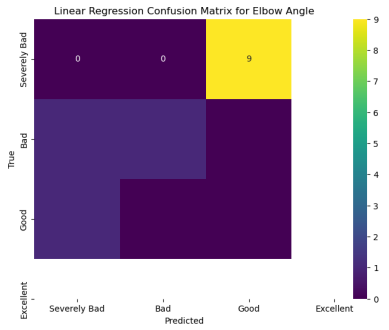
Inner shoulder angle range of motion



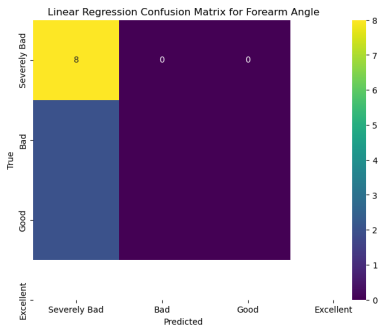
outer shoulder angle range of motion



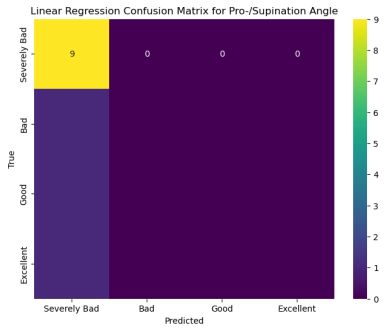
Upper arm angle range of motion



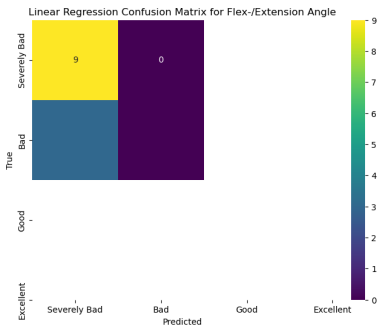
Elbow angle range of motion



Forearm range of motion

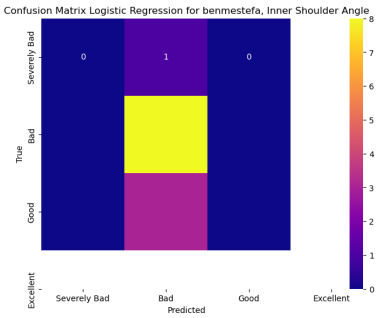


Pro-/supination angle range of mo- tion

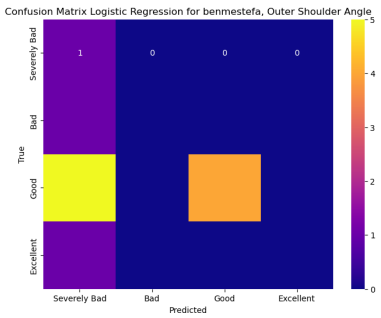


FLEX-extension angle

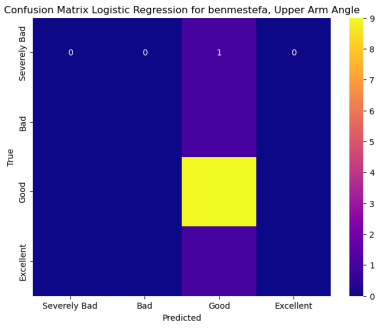
Fig. 3.45 confusion matrices of the healthiness of each angle using linear regression without incremental learning



Inner shoulder angle range of motion

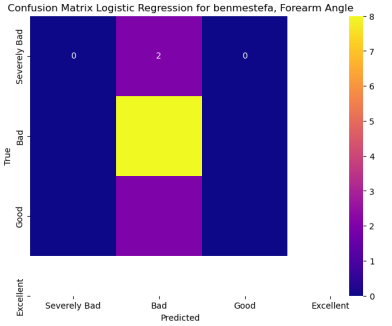


outer shoulder angle range of motion

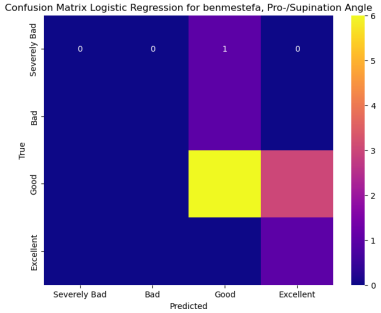


Upper arm angle range of motion

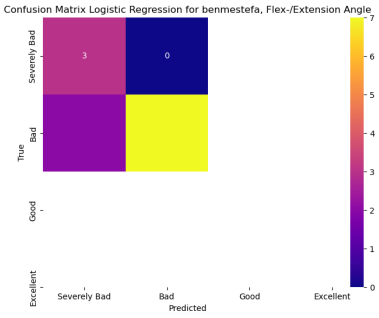
Elbow angle range of motion



Forearm range of motion



Pro-/supination angle range of mo- tion



FLEX-extension angle

Fig. 3.46 confusion matrices of the healthiness of each angle using linear regression with incremental learning

An accuracy test is shown in the table 3.15 We can observe a significant difference between

Angle	Accuracy
Inner Shoulder Angle	0.67
Outer Shoulder Angle	0.75
Upper Arm Angle	0.75
Elbow Angle	0.75
Forearm Angle	0.50
Pro-/Supination Angle	0.75
Flex-/Extension Angle	0.83

Tab. 3.15 Accuracy for Each Angle Using SGD Classifier

the outputs before and after applying incremental learning to both the confusion matrices and the accuracy test. This demonstrates the impact of incremental learning on our model. Although the test accuracy appears to decrease from one numerical value to another at first glance, it is not a sign of poor model performance. The initial accuracy was due to underfitting, which was rectified after applying incremental learning.

6.2.4 Healthiness percentage of each angle

To summarize, we will use linear regression for the last time on range of motion part and exclude logistic regression because we will work with continuous variables.

Moving on to the results depicted in the plot, which are exactly the same as the SVM and KNN models. This means that the plot for **??** is the same for this model, as well as for SVM. To understand why, we conducted some MAE and MSE tests all the way with r squared, and the results are shown in the table. 3.16 The test results were quite unexpected when compared

Tab. 3.16 Metrics for Different Angles

Angle	MAE	MSE	R^2
Inner Shoulder Angle	4.26×10^{-14}	2.76×10^{-27}	1.00
Outer Shoulder Angle	2.65×10^{-13}	7.76×10^{-26}	1.00
Upper Arm Angle	0.229	0.057	0.9995
Elbow Angle	8.136	73.10	0.808
Forearm Angle	7.34×10^{-14}	7.42×10^{-27}	1.00
Pro-/Supination Angle	7.134	152.68	0.859
Flex-/Extension Angle	1.18×10^{-13}	1.46×10^{-26}	1.00

to the plotted data.

7 Conclusion

This chapter is considered pivotal in this thesis, as it delves into the exploration and application of machine learning algorithms for the classification of upper limb rehabilitation data in addition to the impact of using some of the possible enhancements,the main reason for this chapter was aiming to determine the most effective approach, algorithm and model for our specific objectives. this last chapter involved an in-depth analysis of various models developed to perform classification tasks using data such as range of motion, path ratio, and duration, gathered from multiple real subject sessions during their rehabilitation. After rigorous data processing and feature extraction, they were fed into machine learning models to predict or classify binary outcomes (e.g., good, bad, severely bad) or continuous variables like healthiness percentage. Each model yielded unique visualizations, outcomes, accuracies, and test results. It is essential to note that our analysis did not solely rely on standard machine learning algorithms but as mentioned earlier we also integrated enhancements such as dynamic threshold and incremental learning to improve model performance. In additionally to the use of regular preprocessing techniques and methods,in this project we used a special technique called DTW in order to align these sequences, allowing you to compare The kinematic data collected from patients during rehabilitation exercises that varies in speed and timing, even when performing the same task accurately by stretching or compressing the time axis. For instance, if one patient performs a movement more slowly than another, DTW will adjust the time axis to match corresponding points in the movement, enabling a fair comparison, so if a patient performs an exercise slower than the other, DTW will adjust the time axis to match corresponding points in the movement, enabling a fair comparison. DTW technique was only used with path ratio and duration models since the other ones dontrequire time axis for clasification Initially, all models without the dynamic threshold failed to provide accurate predictions due to the adaptive nature of our data, which varies significantly over time. Additionally, the data sources we used do not conform to "normal" distributions typical of healthy populations, making standard comparisons ineffective. This variability introduced a challenge: the models often defaulted to classifying most instances into the "bad" category across all angles, as the data showed minimal variation and simplicity, particularly evident in situations where there was almost no change in the data. This phenomenon was most apparent before applying the dynamic threshold, where the model simply categorized everything into the "bad" class, as depicted in Figure 3.6. The slight difference observed in Figure 3.5 was due to a significant variation in one session for a single subject,and this problem didn't rely on one model or algorithm only but on the three of them, In generally, the initial approach where we used a static threshold for the ROM was unreliable for our case.

The implementation of the dynamic threshold was a game-changer for our models. By introducing adaptability into the model's decision-making process, the dynamic threshold allowed for more accurate and meaningful predictions, as demonstrated in the confusion matrices attached to each model. This method adjusted the decision boundary according to the specific characteristics of each subject's data, thus resolving the issue of the model's previous tendency to overgeneralize or under perform.

Another critical enhancement was the introduction of incremental learning, particularly in models such as SVM and linear regression. We could not apply true incremental learning to KNN due to its nature, but we simulated an approach to mitigate similar challenges. Incremental learning became necessary after facing persistent underfitting, even with the dynamic threshold in place. The underfitting problem was evident in the initial accuracy results, which were unrealistically high or low or abnormal numbers such as zeros and ones, as shown by the

difference in accuracy before and after applying incremental learning (compare **??** and 3.4 for SVM, and 3.14 and 3.15 for linear regression). Incremental learning adjusted the model to more realistically reflect the underlying patterns in the data, avoiding the pitfalls of underfitting even after using the dynamic threshold method.

Despite the improvements brought by dynamic thresholding, The underfitting problem was severe enough that it led to highly biased models, which produced tables filled with ones or zeros, indicating a lack of generalization. To combat this, we implemented incremental learning techniques for SVM and linear regression. Incremental learning allowed us to update the models progressively with new data without retraining on the entire dataset, which was crucial given the changing nature of our data.

For K-Nearest Neighbors (KNN), true incremental learning was not feasible due to the algorithm's reliance on the entire dataset during prediction. Instead, we simulated incremental learning by periodically updating the data and re-training the model, which helped somewhat mitigate underfitting. The effectiveness of this method appeared in linear regression. The results demonstrated a more balanced and realistic model performance, effectively addressing the underfitting issues.

Overall, the enhancements we implemented, including dynamic thresholding and incremental learning, were essential in improving the accuracy and reliability of our machine-learning models. These techniques addressed the unique challenges posed by our simple and adaptive "ROM" data, ultimately leading to better classification performance and more meaningful insights from the data.

Later on, Our choice for these models wasn't random because each one here serves a purpose and each one was tested, starting by the SVM before and after being chosen for its robustness with underfitting the thing proven with the accuracy tests we made comparing it to KNN and linear regression since they give mostly ones or zeros in their accuracy tests oppositely to it, also another advantage of it, is the effectiveness with both regression and classification, our models deal with both binary outcome (classification) in and non-binary outcomes (regression) while linear regression for example deals with the regression only and can't deal with the classification that's what made us choose another way which is the "logistic regression".

To accomplish the underlined goals we had to find a similar or close method to linear regression and the reason why we chose that is the mathematical approach where Linear Regression is The model based on a linear equation of the form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

And the Logistic Regression being The model that uses the logistic (sigmoid) function to map the linear combination of inputs to a probability with the upcoming equation:

$$P(y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$$

they both share a similar mathematical foundation, as both involve a linear combination of input features. However, their purposes and how they map with their outputs are different. Logistic regression was tested all the way with SVM and knn in "the healthiness of each angle, duration and path ratio,where it didn't perform so well comparing it to the other two algorithms according to the resultant accuracy tests performed, confusion matrices or numerical accuracies, first before the incremental learning all of the results were unrealistic, bad and that's because the logistic regression is so sensitive to underfitting and it's know for it's bad performance with it as a proof we have 3.14 for logistic regression and 3.9 for KNN, and **??** for SVM,even after

applying incremental learning to it we couldn't get rid of the underfitting so we can say that regression for these tasks where the "ROM" data is simple or almost unchangeable it doesn't do well. The race now became enclosed between knn and SVM when speaking about binary outcomes or classification.

Knn in the other hand is considered one of the best classification algorithms for its simplicity and speed comparing it to the other algorithms whats noticeable here because it needs no training phase but this advantage is limited to only small or medium datasets not all of them in large datasets it becomes slow for the computation of distances and when it comes to accuracy it performed pretty well comparing it to the SVM in the classification models for "the healthiness of each angle" for both before using the incremental learning and after simulating it and that's pretty evident in the confusion matrices **??** fr knn and **??** for svm ,even when we compare the accuracy results we find that knn did better than SVM,so for classification the KNN was better at its job

When it comes to evaluating the models on regression tasks we will have to include only SVM and KNN (after incremental learning for SVM and simulating it for KNN),and well study the tests one by one,MSE or mean squared error is he average of the squared differences between predicted values and actual target values so the lower it gets the better accuracy gets and for our models SVM have done better passing this test as it has Lower MSE than KNN so that's a point for SVM , moving to MAE,MAE is considered better than MSE because it generalizes it since MAE is the average absolute difference between actual and predicted values so for this one also the lower it gets the best model did , again in our tests the SVM performed better as shown in 3.12 comparing to 3.8, And for the last test , the R squared or (Coefficient of Determination) , we can't rely on this one that much because it has some errors with our models that could be caused by many things such as lack of variance or the models is under fitting ,but even for this test SVM did better than KNN. To support our findings that K-Nearest Neighbors (KNN) is both more accurate and faster than Support Vector Machine (SVM) for classifying arm rehabilitation data, we can refer to several supporting studies and articles.

In the context of classifying arm rehabilitation data, we found that K-Nearest Neighbors (KNN) outperformed Support Vector Machines (SVM) in terms of both accuracy and speed. KNN's straightforward approach seems to match the specific characteristics of this dataset, making it more effective. In contrast, SVM, while powerful, can be more complex and sensitive to parameter settings, which might have led to its lower accuracy and slower processing in our case. Given these factors, KNN stands out as a more efficient and precise choice for this particular application.

For example, a study from the Idaho Conference on Undergraduate Research indicated that KNN generally delivers higher precision in various classification tasks, often outperforming SVM in accuracy. While SVM is typically faster, KNN's better performance in our tests suggests that it can be more reliable for certain datasets, including those related to arm rehabilitation. [8]
The lacking performance of the Support Vector Machine (SVM) in our study seems to be mainly because we only had a limited number of parameters to work with—specifically, just the duration and path ratio. SVMs are powerful but need careful fine-tuning, like selecting the right kernel and adjusting regularization terms. With only these few parameters, it was hard for the SVM to effectively separate the classes in our arm rehabilitation dataset.

So overall, we can conclude that for our specific dataset, regression tasks perform better with SVM, whereas classification tasks yield better results with KNN.

To conclude this section of the chapter, it's important to note that all of the plots showed similar results when analyzing 'the healthiness of each angle,' despite making multiple changes

to the code and obtaining different accuracy results. This issue could have arisen due to various factors, such as over/underfitting because of the simplicity of our data, identical data preprocessing, and feature engineering to ensure a fair comparison. Therefore, it's difficult to pinpoint the exact problem we encountered, and we'll need to rely on the accuracy tests we conducted to evaluate the models.

8 General conclusion

This research was conducted to classify kinematic data from a robotic upper limb rehabilitation orthosis. Our primary goal was to select the most effective classification model by evaluating the performance of various algorithms in accurately classifying the data gathered from our dataset, which is crucial for assessing patient progress and modernizing rehabilitation protocols.

In this study, we focused on three machine learning models: k-Nearest Neighbors (k-NN), Linear Regression, and Support Vector Machines (SVM). Each model was applied to our kinematic data, and their performance was assessed based on classification accuracy, robustness to data imbalance, and adaptability to new data. SVM demonstrated its efficiency and robustness in both regression and classification tasks across all models, with fewer issues that required additional techniques to resolve. k-NN performed slightly better than SVM in terms of handling the data, especially since our dataset is relatively simple and tends to cause underfitting. As for Linear Regression, it was only applicable to one model that required regression, but its results were subpar compared to SVM and k-NN. Consequently, we switched to Logistic Regression for classification tasks, but it still did not perform as well as the other models.

A significant part of our research involved creating a model to assess the "healthiness percentage per each angle" and analyzing the results for each algorithm. Unexpectedly, the plots generated for each model were identical. This uniformity suggests that despite the differences in the models' methodologies, they may have converged on similar classifications due to the nature of the data and the thresholds applied. This observation raises the possibility that using multiple models might be redundant when the characteristics of the dataset lead to convergent outcomes. Further investigation, which we could not cover in this research, is necessary to understand the underlying reasons for this uniformity and to determine whether it is a result of the data, the thresholds, or inherent similarities in the models' decision boundaries.

Our findings have significant implications for the field of robotic rehabilitation. Classifying kinematic data using machine learning enables more precise monitoring of patient progress and status, potentially leading to more personalized and effective rehabilitation strategies. SVM's robustness and adaptability make it particularly well-suited for clinical settings, where data quality and quantity can vary. Overall, this research has demonstrated the power of machine learning in the rehabilitation process and the capabilities and limitations of each of the three machine learning models, while also highlighting the potential for new technologies to improve human lives.

Bibliography

[1] 5 Types of Classification Algorithms in Machine Learning — monkeylearn.com. [Accessed 03-04-2024].

[2] Critical Shoulder Angle and Its Clinical Correlation in Shoulder Pain — ncbi.nlm.nih.gov. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7494420/. [Accessed 26-09-2024].

[3] Plot classification boundaries with different SVM Kernels — scikit-learn.org. https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html. [Accessed 05-04-2024].

[4] Statistical models : theory and practice : Freedman, David, 1938-2008 : Free Download, Borrow, and Streaming : Internet Archive — archive.org. https://archive.org/details/statisticalmodel0000free. [Accessed 26-09-2024].

[5] Raya Alshammri, Ghaida Alharbi, Ebtisam Alharbi, and Ibrahim Almubark. Machine learning approaches to identify parkinson's disease using voice signal features. *Frontiers in Artificial Intelligence*, 6:1084001, 2023.

[6] Kyriakos Apostolidis, Christos Kokkotis, Serafeim Moustakidis, Evangelos Karakasis, Paraskevi Sakellari, Christina Koutra, Dimitrios Tsiptsios, Stella Karatzetzou, Konstantinos Vadikolias, and Nikolaos Aggelousis. Machine learning algorithms for the prediction of language and cognition rehabilitation outcomes of post-stroke patients: A scoping review. *Human-Centric Intelligent Systems*, December 2023.

[7] SwissCognitive's Featured Article. Applications of machine learning, Dec 2022.

[8] (Mentor) Barry Myers. KNN vs SVM: A Comparison of Algorithms — scholarworks.boisestate.edu. https://scholarworks.boisestate.edu/icur/2017/Poster_Session/139/. [Accessed 03-08-2024].

[9] Tanin Boka, Arshia Eskandari, S. Ali A. Moosavian, and Mahkame Sharbatdar. Using machine learning algorithms for grasp strength recognition in rehabilitation planning. *Results in Engineering*, 21:101660, 2024.

[10] Siqi Cai, Yan Chen, Shuangyuan Huang, Yan Wu, Haiqing Zheng, Xin Li, and Longhan Xie. Svm-based classification of semg signals for upper-limb self-rehabilitation training. *Frontiers in Neurorobotics*, 13, 2019.

[11] Victor C Espinoza Bernal, Shivayogi V Hiremath, Bethany Wolf, Brooke Riley, Rochelle J Mendonca, and Michelle J Johnson. Classifying and tracking rehabilitation interventions through machine-learning algorithms in individuals with stroke. *Journal of Rehabilitation and Assistive Technologies Engineering*, 8:205566832110446, January 2021.

[12] Fernanda Márcia Rodrigues Martins Ferreira, Guilherme de Paula Rúbio, Fabrício Henrique de Lisboa Brandão, Arthur Mazzini da Mata, Natália Batista Castilho de Avellar, João Paulo Fernandes Bonfim, Leandro Gonzaga Tonelli, Thales Gomes Silva, Rina Mariane Alves Dutra, Adriana Maria Valladão Novais Van Petten, and Claysson Bruno Santos Vimieiro. Robotic orthosis for upper limb rehabilitation. In *The 1st International Electronic Conference on Actuator Technology: Materials, Devices and Applications*, IeCAT 2020. MDPI, November 2020.

[13] Toyohiro Hamaguchi, Takeshi Saito, Makoto Suzuki, Toshiyuki Ishioka, Yamato Tomisawa, Naoki Nakaya, and Masahiro Abo. Support vector machine-based classifier for the assessment of finger movement of stroke patients undergoing rehabilitation. *Journal of Medical and Biological Engineering*, 40(1):91–100, September 2019.

[14] Italo José. KNN (K-Nearest Neighbors) 1 — towardsdatascience.com. [Accessed 03-04-2024].

[15] Pazit Levinger, Daniel T.H. Lai, Rezaul K. Begg, Kate E. Webster, and Julian A. Feller. The application of support vector machines for detecting recovery from knee replacement surgery using spatio-temporal gait parameters. *Gait amp; Posture*, 29(1):91–96, January 2009.

[16] Paweł Maciejasz, Jörg Eschweiler, Kurt Gerlach-Hahn, Arne Jansen-Troy, and Steffen Leonhardt. A survey on robotic devices for upper limb rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, 11(1), January 2014.

[17] Dastan Maulud and Adnan M. Abdulazeez. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, 1(2):140–147, December 2020.

[18] Selva Prabhakaran. Linear Regression in Machine Learning – Clearly Explained — machinelearningplus.com. `https://www.machinelearningplus.com/machine-learning/linear-regression-in-machine-learning/`. [Accessed 26-09-2024].

[19] Khansa Rasheed, Adnan Qayyum, Junaid Qadir, Shobi Sivathamboo, Patrick Kwan, Levin Kuhlmann, Terence O'Brien, and Adeel Razi. Machine learning for predicting epileptic seizures using eeg signals: A review. *IEEE reviews in biomedical engineering*, 14:139–155, 2020.

[20] Nikolaus Sabathiel, Danut Irimia, Brendan Allison, Christoph Guger, and Günter Edlinger. Paired associative stimulation with brain-computer interfaces: A new paradigm for stroke rehabilitation. volume 9743, pages 261–272, 07 2016.

[21] A. E. Selim, M. Abdel Wahed, and Y. M. Kadah. Machine learning methodologies in brain-computer interface systems. In *2008 Cairo International Biomedical Engineering Conference*. IEEE, December 2008.

[22] ShantanuDeo. Learn the Concept of linearity in Regression Models - DataScienceCentral.com — datasciencecentral.com. `https://www.datasciencecentral.com/learn-the-concept-of-linearity-in-regression-models/`. [Accessed 26-09-2024].

[23] Mariusz Sobiech, Wojciech Wolański, and Ilona Karpiel. *Brief Overview Upper Limb Rehabilitation Robots/Devices*, page 286–297. Springer International Publishing, 2022.

[24] M. Srividya, Mohanavalli Subramaniam, and Bhalaji Natarajan. Behavioral modeling for mental health using machine learning algorithms. *Journal of Medical Systems*, 42:88, 04 2018.

[25] Chaitanya Suryadevara. Issue 4 diabetes risk assessment using machine learning: A comparative study of classification algorithms. *International Journal of Applied Engineering Research and Development*, 8:1–10, 02 2022.

[26] Romain Tavenard. An introduction to Dynamic Time Warping — rtavenar.github.io. `https://rtavenar.github.io/blog/dtw.html`. [Accessed 21-04-2024].

[27] Tech-AI-Math. K-Nearest Neighbors (KNN) in Depth — ai.plainenglish.io. [Accessed 03-04-2024].

[28] Michael Tschuggnall, Vincent Grote, Michael Pirchl, Bernhard Holzner, Gerhard Rumpold, and Michael J. Fischer. Machine learning approaches to predict rehabilitation success based on clinical and patient-reported outcome measures. *Informatics in Medicine Unlocked*, 24:100598, 2021.

[29] I-Tung Yang and Handy Prayogo. Efficient reliability analysis of structures using symbiotic organisms search-based active learning support vector machine. *Buildings*, 12:455, 04 2022.

[30] Mu Zhu, Zhanyang Zhang, John P Hirdes, and Paul Stolee. Using machine learning algorithms to guide rehabilitation planning for home care clients. *BMC Medical Informatics and Decision Making*, 7(1), December 2007.

[31] Difan Zou, Lingxiao Wang, Pan Xu, Jinghui Chen, Weitong Zhang, and Quanquan Gu. Epidemic model guided machine learning for covid-19 forecasts in the united states. May 2020.

ملخص

في عالم اليوم، أصبح دور علوم الكمبيوتر بالغ الأهمية لدرجة أنه يمكن اعتباره إلزاميًا في مختلف جوانب الحياة. بمرور الوقت، اندمجت علوم الكمبيوتر بسلاسة في كل مجال تقريبًا، بدءًا من الهندسة وصولاً إلى علم الأحياء وحتى الطب. أمثلة على كيفية قيام علوم الكمبيوتر بإحداث ثورة في المجال الطبي تشمل استخدام الخوارزميات لتحسين وضوح الصور الشعاعية، تحسين جودة صور تخطيط القلب، ومساعدة الأطباء في تشخيص المرضى. هذه التطورات سمحت بتشخيصات أكثر دقة، اتخاذ قرارات أسرع، وتحسين الرعاية الصحية بشكل عام. لكن ماذا يحدث عندما ندمج القوة المشتركة للهندسة وعلوم الكمبيوتر؟ تتسع إمكانيات الابتكار وحل المشكلات بشكل كبير، مما يفتح مسارات جديدة لابتكار حلول متقدمة أو تحسين التقنيات الحالية، خاصة في مجالات مثل إعادة التأهيل .إعادة التأهيل، رغم أنها ليست مفهومًا جديدًا في الطب، ظلت متخلفة تاريخيًا بسبب القيود المادية ونقص المعرفة الشاملة. حتى وقت قريب، كانت التقدمات في إعادة التأهيل متواضعة في أحسن الأحوال. ولكن مع التقدم السريع في التكنولوجيا الطبية، تطورت أساليب إعادة التأهيل بشكل كبير. تقنيات جديدة مثل واجهات الدماغ-الكمبيوتر(BCIs) ، العلاج بالخلايا الجذعية، والتحفيز الكهربائي الوظيفي قد ظهرت كطرق واعدة لتعافي المرضى. على الرغم من هذه الابتكارات، لا تزال هناك تحديات كبيرة تواجه ممارسات إعادة التأهيل الحديثة. من بين أبرز المشكلات التي تواجهها هذه الممارسات هو الركود أو توقف التقدم خلال جلسات العلاج، والذي يتفاقم غالبًا بسبب الاعتماد الكبير على المعالجين البشر. مشكلة أخرى تتمثل في صعوبة التعرف على التحسن إلا بعد فترة طويلة من الوقت، مما يجعل من الصعب تتبع التقدم التدريجي .هذه التحديات المستمرة تؤكد الحاجة إلى حلول أكثر تقدمًا وموثوقية. إحدى الطرق الواعدة تتمثل في دمج علوم الكمبيوتر والهندسة لتطوير أدوات إعادة التأهيل الأكثر كفاءة وفعالية. يمكن تحقيق ذلك من خلال إنشاء أنظمة إعادة التأهيل الروبوتية، حيث تلعب الهندسة دورًا في تصميم وتطوير الجهاز الروبوتي، وتستخدم علوم الكمبيوتر، وخاصة من خلال التعلم الآلي، لتحليل البيانات التي يتم جمعها أثناء جلسات إعادة التأهيل. يمكن للخوارزميات التعلم الآلي استخراج ومعالجة كميات هائلة من البيانات الحركية والفسيولوجية من الجهاز الروبوتي، واستخدام هذه البيانات لمراقبة تقدم المرضى، التنبؤ بالنتائج، وتكييف بروتوكولات العلاج في الوقت الفعلي. يتيح هذا التكامل بين علوم الكمبيوتر والهندسة نهجًا أكثر تخصيصًا يستند إلى البيانات في إعادة التأهيل، مما يقلل من الاعتماد على المعالجين، ويقدم ملاحظات فورية حول تقدم المرضى، ويؤدي إلى نتائج تعافي أفضل على المدى الطويل .من خلال دمج هذه المجالات، يمكننا بناء مستقبل تكون فيه إعادة التأهيل أكثر سهولة وكفاءة، مع تحسينات وتعديلات في الوقت الفعلي بناءً على بيانات المرضى. يمثل ذلك قفزة هائلة عن الأساليب التقليدية، حيث نتجه نحو شكل أكثر استقلالية وذكاءً من الرعاية العلاجية، مما يؤدي في النهاية إلى تجارب تعافي أفضل للمرضى .الروبوت الذي نناقشه هنا يُسمى روبوت Armeo، وقد تم إنشاؤه بواسطة شركة .Hocoma يُستخدم هذا الروبوت خلال جلسات إعادة التأهيل للمرضى الذين يعانون من إصابات أو خلل في الوظائف الحركية للطرف العلوي. يساعد الروبوت المرضى على ممارسة التمارين من خلال ألعاب صغيرة على الكمبيوتر تشجعهم على تحريك أطرافهم العلوية. يتم توصيل مستشعرات بأطراف المرضى لتسجيل البيانات المتعلقة بزوايا الحركة، والتي يتم استخدامها فيما بعد في نماذج التعلم الآلي للتنبؤ بالنتائج .كجزء من بحثنا، سعينا إلى تحديد أي من الخوارزميات المعروفة (KNN, SVM,Linear regression) هي الأنسب لهذا الهدف. وجدنا أن Linear regression لا يمكن استخدامه للتصنيف، لذلك استخدمنا بدلاً منه .Logistic regressionخلال هدا البحث، تم إنشاء عدة نماذج وتم تقسيمها إلى مجموعتين.

المجموعة الأولى كانت تخص نماذج regression، حيث تم إنشاء نموذجين لكل من الخوارزميات الثلاث. قامت النماذج بحساب نطاق الحركة للطرف العلوي بالكامل باستخدام نطاق حركة كل زاوية وتنبأت بصحة كل زاوية كنسبة مئوية. في هذه المهمة، كان أداء خوارزمية SVM أفضل من الخوارزميات الأخرى وفقًا لاختبارات الدقة .

أما المجموعة الثانية، فكانت تخص نماذج التصنيفclassification، حيث تم إنشاء ثلاثة نماذج لكل خوارزمية. النموذج الأول يحتوي على أربع فئات (جيد، سيئ، ممتاز، سيئ للغاية) وصنف الطرف العلوي للمريض بناءً على نطاق الحركة الذي تم الحصول عليه من مهمة الانحدار. ومن بين الخوارزميات الثلاث، كانت KNN الأفضل وفقًا لنتائج اختبارات الدقة. ثم قمنا باختبار المسارات والفترات الزمنية لتصنيف الطرف إلى فئتين (صحي وغير صحي)، وكان أداء KNN جيدًا أيضًا في هذه المهمة .من الجدير بالذكر أننا استخدمنا تحسينات في الأكواد مثل التعلم التزايدي والعتبات الديناميكية لتحقيق أهدافنا. وكانت الاختلافات المكتشفة نتيجة لتعقيد البيانات، حيث كنا نعمل مع بيانات غير تقليدية تتطلب مزيدًا من التحسين للحصول على نتائج دقيقة .بشكل عام، قدمت هذه التجارب رؤى قيمة حول كيفية أداء خوارزميات التصنيف إذا تم دمجها سريريًا.

الكلمات المفتاحية :علوم الحاسوب، الهندسة، إعادة التأهيل، التعلم الآلي، إعادة التأهيل الروبوتي، البيانات الحركية، إصابات الأطراف العلوية، HACOMA، نطاق الحركة، k-NN، SVM ، الانحدار اللوجستي،linear regression نماذج التصنيف، الانحدار الخطي، التصنيف، الانحدار، التعلم التدريجي، العتبات الديناميكية، العتبات الثابتة، إعادة التأهيل البدني، إعادة التأهيل المخصص.

Résumé

De nos jours, le rôle de l'informatique est devenu si essentiel qu'il peut être considéré comme indispensable dans divers aspects de la vie. Au fil du temps, elle s'est intégrée de manière fluide dans presque tous les domaines, de l'ingénierie à la biologie et même à la médecine. L'utilisation d'algorithmes pour améliorer la clarté des images radiographiques, la qualité des échocardiogrammes et assister les médecins dans le diagnostic des patients ne sont que quelques exemples de la manière dont l'informatique a révolutionné le domaine médical. Ces avancées ont permis des diagnostics plus précis, une prise de décision plus rapide, et une amélioration globale des soins aux patients. Cependant, que se passe-t-il lorsque l'on combine la puissance de l'ingénierie et de l'informatique ? Les possibilités d'innovation et de résolution de problèmes s'élargissent considérablement, ouvrant de nouvelles voies pour créer des solutions de pointe ou améliorer les technologies existantes, en particulier dans des domaines tels que la rééducation.

Bien que La rééducation ne soit pas un concept nouveau en médecine elle a historiquement été sous-développée en raison des limitations matérielles et d'un manque de connaissances approfondies. Jusqu'à récemment, les progrès dans ce domaine étaient au mieux modestes. Cependant, avec les avancées rapides de la technologie médicale, les méthodes de rééducation ont considérablement évolué. De nouvelles techniques, telles que les interfaces cerveau-ordinateur (BCI), la thérapie par cellules souches et la stimulation électrique fonctionnelle, ont émergé comme des approches prometteuses pour la récupération des patients. Malgré ces innovations, des défis importants persistent dans les pratiques de rééducation modernes. L'un des problèmes principaux est la stagnation ou le plafonnement des progrès au cours des séances de thérapie, souvent aggravé par une forte dépendance aux thérapeutes humains. Un autre problème notable est la difficulté de reconnaître les améliorations avant une période de temps prolongée, ce qui complique le suivi des progrès incrémentaux.

Ces défis continus soulignent la nécessité de solutions plus avancées et fiables. Une approche prometteuse consiste à fusionner l'informatique et l'ingénierie pour développer des outils de rééducation plus efficaces et efficients. Cela peut être réalisé par la création de systèmes de rééducation robotisés, où l'ingénierie joue un rôle dans la conception et le développement du dispositif robotique, tandis que l'informatique, en particulier grâce à l'apprentissage automatique, est utilisée pour analyser les données collectées lors des séances de rééducation. Les algorithmes d'apprentissage automatique peuvent extraire et traiter d'énormes quantités de données cinématiques et physiologiques provenant du dispositif robotique en utilisant ces données pour surveiller les progrès des patients, prédire les résultats et adapter les protocoles de thérapie en temps réel. Cette intégration de l'informatique et de l'ingénierie offre une approche plus personnalisée et basée sur les données, réduisant potentiellement la dépendance aux thérapeutes, fournissant un retour d'information immédiat sur les progrès des patients et conduisant à de meilleurs résultats de récupération à long terme.

En fusionnant ces deux domaines, nous pourrons construire un avenir où la rééducation sera non seulement plus accessible, mais également beaucoup plus efficace, offrant des améliorations en temps réel et des ajustements basés sur les données des patients. Cela représente un saut considérable par rapport aux méthodes traditionnelles, en évoluant vers une forme de soins thérapeutiques plus autonome et intelligente, conduisant finalement à de meilleures expériences de récupération pour les patients.

Le robot dont nous discutons ici est appelé le robot Armeo, créé par l'entreprise Hocoma. Ce robot est utilisé lors des séances de rééducation pour les patients souffrant de blessures ou d'anomalies fonctionnelles du membre supérieur. Il aide les patients à faire de l'exercice en interagissant avec des mini-jeux sur un ordinateur qui les encouragent à bouger leur membre supérieur. Des capteurs attachés aux membres des patients enregistrent les données sur les

angles de mouvement, qui sont ensuite utilisées dans des modèles d'apprentissage automatique pour prédire les résultats.

Dans le cadre de notre recherche, nous avons cherché à déterminer quel algorithme parmi les plus connus (KNN, SVM, régression linéaire) est le mieux adapté à cet objectif. Nous avons constaté que la régression linéaire ne pouvait pas être utilisée pour la classification, nous avons donc utilisé à la place la régression logistique.

Au cours de la recherche, plusieurs modèles ont été créés et divisés en deux groupes. Le premier groupe concernait les modèles de régression, créant deux modèles pour chacun des trois algorithmes. Les modèles calculaient l'amplitude des mouvements pour l'ensemble du membre supérieur en utilisant l'amplitude de chaque angle et prédisaient la santé de chaque angle en pourcentage. Dans cette tâche, le SVM a mieux performé que les deux autres algorithmes selon les tests d'exactitude. Le deuxième groupe consistait en des modèles de classification, où trois modèles ont été créés pour chaque algorithme. Le premier modèle avait quatre classes (bon, mauvais, excellent, gravement mauvais) et classait le membre supérieur du patient en fonction de l'amplitude de mouvement obtenue à partir de la tâche de régression. Parmi les trois algorithmes, le KNN a mieux performé selon les résultats des tests d'exactitude. Nous avons ensuite testé les trajectoires et les durées pour classifier le membre en deux classes (sain et malsain), et le KNN a également bien performé dans cette tâche.

Il est important de noter que nous avons utilisé des améliorations de code, telles que l'apprentissage incrémentiel et les seuils dynamiques, pour atteindre nos objectifs. Les différences observées étaient dues à la complexité des données, car nous travaillions avec des données atypiques qui nécessitent encore un affinement pour obtenir des résultats plus précis.

Dans l'ensemble, ces expériences ont fourni des informations précieuses sur la manière dont les algorithmes de classification se comporteraient s'ils étaient intégrés cliniquement.

Mots-clés : Informatique, Ingénierie, Rééducation, Apprentissage automatique, Rééducation robotique, Données cinématiques, Lésions des membres supérieurs, HACOMA, Amplitude de mouvement, SVM (Machines à Vecteurs de Support), k-NN (k-plus Proches Voisins), Régression logistique, Modèles de classification, Régression linéaire, Classification, Régression, Apprentissage incrémental, Seuils dynamiques, Seuils statiques, Rééducation physique, Rééducation personnalisée.

Abstract

In today's world, the role of computer science has become so essential that it can be considered mandatory in various aspects of life. Over time, it has seamlessly integrated into nearly every domain, from engineering to biology and even medicine. there are many examples in this matter such as using algorithms to enhance the clarity of radiographic images, improve the quality of echo-cardiograms, and assist doctors in diagnosing patients. These advancements have allowed for more precise diagnoses, quicker decision-making, and overall better patient care. However, what happens when we combine the power of both engineering and computer science? The possibilities for innovation and problem-solving expand tremendously, offering new pathways for creating cutting-edge solutions or improving existing technologies, particularly in fields like rehabilitation.

Rehabilitation, while not a novel concept in medicine, has historically been underdeveloped due to limitations in materials and a lack of comprehensive knowledge. Until recently, advancements in rehabilitation had been modest at best. However, with the rapid progress in medical technology, rehabilitation methods have evolved significantly. New techniques, such as brain-computer interfaces (BCIs), stem cell therapy, and functional electrical stimulation, have emerged as promising approaches to patient recovery. Despite these innovations, significant challenges still remain in modern rehabilitation practices. One of the primary issues is the stagnation or plateauing of progress during therapy sessions, often compounded by a heavy dependence on human therapists. Another notable problem is the difficulty in recognizing improvement until much later in the rehabilitation process, making it hard to track incremental progress.

These ongoing challenges underscore the need for more advanced and reliable solutions. One promising approach is the fusion of computer science and engineering to develop more efficient and effective rehabilitation tools. This can be achieved through the creation of robotic rehabilitation systems, where engineering plays a role in the design and development of the robotic device, and computer science, particularly through machine learning, is employed to analyze data collected during rehabilitation sessions. Machine learning algorithms can extract and process vast amounts of kinematic and physiological data from the robotic device, using this data to monitor patient progress, predict outcomes, and adapt therapy protocols in real-time. This integration of computer science and engineering offers a more personalized, data-driven approach to rehabilitation, potentially minimizing reliance on therapists, providing more immediate feedback on patient progress, and leading to better long-term recovery outcomes.

By merging these fields, we can build a future where rehabilitation is not only more accessible but also far more efficient, offering real-time improvements and adjustments based on patient data. This represents a significant leap forward from traditional methods, moving toward a more autonomous and intelligent form of therapeutic care, ultimately leading to better recovery experiences for patients.

The robot we are discussing is called the Armeo robot and is created by the company HACOMA. This robot is used during rehabilitation sessions for patients with upper limb injuries or abnormalities. It helps patients exercise by engaging in mini-games on a computer that encourage movement of the upper limb. Sensors attached to the patients' limbs record data on the angles of movement, which is then used in machine learning models to predict outcomes.

As part of our research, we aimed to determine which of the well-known algorithms (KNN, SVM, Linear regression) is best suited for this goal. We found that linear regression could not be used for classification, so we used logistic regression instead.

During the research, several models were created and divided into two groups. The first group was regression models, creating two models for each of the three algorithms. The models

calculated the range of motion for the entire upper limb using the range of motion of each angle, and predicted the healthiness of each angle as a percentage. In this task, SVM performed better than the other two algorithms according to accuracy tests. The second group consisted of classification models, with three models created for each algorithm. The first model had four classes (good, bad, excellent, severely bad) and classified the patient's upper limb based on the range of motion obtained from the regression task. Of the three algorithms, KNN performed better according to accuracy test results. We further tested the trajectories and durations to classify the limb into two classes (healthy and unhealthy), with KNN performing well in this task too. It's worth noting that we employed code enhancements such as incremental learning and dynamic thresholds to achieve our goals. The differences found were attributed to the complexity of the data, as we were working with atypical data that still requires refinement for accurate results. Overall, these experiments provided insights into how the classification algorithms would perform if clinically integrated.

Keywords: Computer Science, Engineering, Rehabilitation, Machine Learning, Robotic Rehabilitation, Kinematic Data, Upper Limb Injuries, HACOMA, Range of Motion, SVM (Support Vector Machines), k-NN (k-Nearest Neighbors), Logistic Regression, Classification Models, linear Regression ,classification,regression ,Incremental Learning, Dynamic Thresholds,static threshold, ,physical rehabilitation, Personalized Rehabilitation.