REHABILATION APPLICATION FOR HUMAN LOCOMOTIVE DISEASE

A PROJECT REPORT

Submitted by

ABINAYA N (711720243003)

BALA B (711720243010)

MITHUN S (711720243030)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

KGiSL INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**REHABILATION APLICATION FOR HUMAN LOCOMOTIVE DISEASE**" is the bonafide work of **ABINAYA N, BALA B And MITHUN S** who carried out the project work under my supervision.

SIGNATURE

Dr.KALPANA S

HEAD OF THE DEPARTMENT

Department of Artificial Intelligence and Data Science
KGiSL Institute of Technology
Coimbatore - 641035

SIGNATURE

Dr.KALPANA S

SUPERVISOR

Department of Artificial Intelligence and Data Science
KGiSL Institute of Technology
Coimbatore - 641035

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Chairman and Managing Director Dr. Ashok Bakthavachalam** for providing us with an environment to complete our project successfully.

We are grateful to our **CEO of Academic Initiatives Mr. AravindKumar Rajendran** and our beloved **Secretary Dr. Rajkumar N.** Our sincere thanks to honorable **Principal Dr. Selvam M** for his support, guidance, and blessings.

We would like to thank **Dr. Kalpana S, Head of the Department**, Department of Artificial Intelligence and Data Science for her firm support during the entire course of this project work and who modelled us both technically and morally for achieving greater success in this project work.

We express our sincere thanks to **Ms. Suganya T** and **Mr. Sathish R, our project coordinators**, **Assistant Professors**, Department of Artificial Intelligence and Data Science, and our guide **Dr. Kalpana S, , Head of the Department** Department of Artificial Intelligence and Data Science for their constant encouragement and support throughout our course, especially for the useful suggestions given during the course of the project period and being instrumental in the completion of our project with their complete guidance.

We also thank all the faculty members of our department for their help in making this project a successful one. Finally, we take this opportunity to extend our deep appreciation to our Family and Friends, for all they meant to us during the crucial times of the completion of our project.

ABSTRACT

Human locomotion is one of the most significant factors in many diseases, including Parkinson's disease, Ataxia, and Huntington's disease. Patients with these diseases require progressive tests and analyses of their human gait, which are frequently performed in hospitals using sophisticated equipment and necessitate having the patient on hand. It can be difficult for patients with limited mobility or those who live in remote areas to access the current methods for analyzing human gait in patients with neurological disorders because they are frequently pricy, time consuming, require specialized medical facilities, and call for skilled medical personnel. There is a need for easier, more affordable, and more practical gait analysis systems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

KNN	K-Nearest Neighbour
SVM	Support Vector Machine
CNN	Convolutional Neural Network
ANFIS	Adaptive-Network-based Fuzzy Inference System
HTML	Hypertext Markup Language
CSV	Cascading Style Sheets

CHAPTER 1

INTRODUCTION

1. OVERVIEW:

Millions of people all around the world are afflicted by the chronic and progressive movement illness known as Parkinson's disease. Dopamine, a neurotransmitter that aids in controlling movement, is produced by brain neurons, and their death is what causes it. Parkinson's disease, which causes tremors, stiffness, and issues with balance and coordination, can significantly affect a person's quality of life. Parkinson's disease has no known cure, however rehabilitation programmers can assist patients in managing their symptoms and enhancing their general quality of life.

The use of machine learning algorithms to aid in the rehabilitation of human locomotive dysfunction has attracted increasing interest in recent years. These algorithms can track rehabilitation progress and forecast Parkinson's disease stages, giving patients and their healthcare professionals useful information.

Using the Mediapipe framework, a video of the patient doing particular movements will be recorded and used to produce the dataset. In order to identify and monitor important body parts, including joints and limbs, this system employs machine learning methods. The KNN algorithm will be trained using a dataset made from this data.

Simple machine learning algorithms like the KNN method are frequently employed for categorization jobs. It operates by locating the k-nearest neighbors of a given data point, after which the data point is assigned to the class with the highest degree of commonality among its closest neighbors. Based on the data gathered with the help of the Mediapipe architecture, the KNN algorithm will be utilized to forecast the stage of Parkinson's disease. Machine learning techniques will be utilized to monitor recovery

progress once the Parkinson's disease stage has been predicted. Additional recordings of the patient making particular actions over time will be recorded in order to do this, and machine learning techniques will be used to analyses the data. The algorithms will be used to monitor changes in the patient's motions and give feedback on how well they are recovering.

This project's ultimate goal is to create a rehabilitation application that patients and their medical professionals may use to treat Parkinson's disease and other human motor disorders. This application has the potential to greatly enhance the lives of persons affected by these disorders by employing machine learning techniques to monitor rehabilitation efforts.

1.1 PROBLEM DEFINITION:

The lack of efficient instruments and techniques to aid in the rehabilitation of human locomotive disease, notably Parkinson's disease, is the issue that this research seeks to address. While rehabilitation programmes can aid patients in managing their symptoms and enhancing their overall quality of life, more sophisticated and individualised treatments are required to meet the disease's complexity.

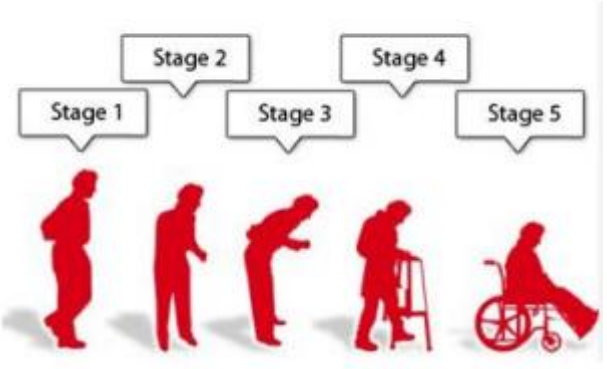


Fig – 1 Parkinson’s stage

Traditional rehabilitation techniques frequently rely on healthcare professionals' subjective evaluations, which can be inconsistent and incorrect. Additionally, patients could find it challenging to appropriately self-report their improvement, making it challenging for medical professionals to monitor their rehabilitation progress. This study intends to give more accurate and objective assessments of patients' movements and progress through the use of machine learning algorithms. The KNN algorithm can identify the stage of Parkinson's disease by examining data from recordings of patients doing particular motions, offering crucial information for healthcare professionals. Additionally, rehabilitation success can be tracked over time using machine learning algorithms, giving patients and healthcare professionals immediate feedback on their progress.

The overall goal of this project is to address the need for more sophisticated and individualised methods to the treatment of human motor diseases by employing machine learning algorithms to provide assessments of patients' movements and progress that are more accurate and objective.

1.2 OBJECTIVE:

1.2.1

- The To develop a rehabilitation application for human locomotive disease using machine learning algorithms, specifically the KNN algorithm for predicting the stage of Parkinson's disease and machine learning algorithms for tracking rehabilitation progress.
- To create a dataset using the Mediapipe framework that captures video data of patients performing specific movements, which will be used to train the KNN algorithm.
- To use the KNN algorithm to predict the stage of Parkinson's disease based on the data captured using the Mediapipe framework.
- To use machine learning algorithms to track rehabilitation progress by

capturing additional videos of patients performing specific movements over time and analyzing the data.

- To provide real-time feedback on rehabilitation progress to patients and their healthcare providers using the rehabilitation application.

1.3 SIGNIFICANCE:

The significance of this project lies in its potential to improve the quality of life for people suffering from human locomotive disease, specifically Parkinson's disease. By using machine learning algorithms to predict the stage of Parkinson's disease and track rehabilitation progress, this project can provide more accurate and objective assessments of patients' movements and progress, leading to better-tailored treatment plans and improved outcomes. Additionally, this project has the potential to contribute to the development of more advanced and personalized approaches to the rehabilitation of human locomotive disease, leading to more effective treatments and therapies in the future. Overall, this project has the potential to improve the lives of people suffering from Parkinson's disease and other human locomotive diseases, as well as advance our understanding of these conditions and how best to treat them.

1.4 OUTLINE:

1.4.1 Rehabilitation application for human locomotive disease provides a complete user-friendly environment for predicting the stage of Parkinson’s disease and track rehabilitation.

CHAPTER 2

LITERATURE REVIEW

In this chapter, the researches of various authors are sited with their proposed methodologies. For the effects of these studies, it demonstrates a promising mechanism for the use of these methods for building the recommendation system.

1. **A markerless system for gait analysis based on OpenPose library (2020)**

A low-cost markerless system for 3D human motion detection and tracking using open-pose library 3D kinematic and spatiotemporal data of human gait can be also computed here. Results showed here is able to track lower limbs motion, producing angular traces representative of normal gait similar to the ones computed by IMUs .

2. **A Markerless gait analysis based on a single RGB camera (2020)**

Single RGB camera used to track the 2D joint coordinates with state-of-the-art vision algorithms.Reconstruction of the 3D trajectories uses sparse representation of an active shape model..

3. **Accurate Shoulder Joint Angle Estimation Using Single RGB camera for Rehabilitation(2020)**

MS Kinect is a marker-less technology introduced recently in the gaming industry.Accuracy in the estimation of shoulder joint angle between machine-learning-based RGB camera and MS Kinect shows mean error of 9.29° and 5.3°.

4. **Deep Domain Adaptation to Predict Freezing of Gait in Patients with Parkinson's Disease (2021)**

(FoG) is a common gait impairment in patients with advanced Parkinson's disease (PD), which manifests as sudden difficulties in starting or continuing locomotion .The performance of advanced deep learning algorithms to predict FoG events in short time durations before their occurrence is studied...

5. **Feature Selection Based on L1-Norm Support Vector Machine and Effective Recognition System for Parkinson's Disease Using Voice Recordings (2019)**

The support vector machine (SVM) was used as a predictive model for the prediction of PD k-fold-cross validation method used.This paper achieved the best subset selection of features (RNN).

6. **Multi-Source Ensemble Learning for the Remote Prediction of Parkinson's Disease in the Presence of Source-Wise Missing Data (2019)**

The use of multi-source ensemble learning, alongside convolutional neural networks (CNNs) increases PD classification accuracy from 73.1% to 82.0% .This method is applicable to a wide range of wearable/remote monitoring datasets that suffer from missing data.

7. **Optimized ANFIS Model Using Hybrid Metaheuristic Algorithms for Parkinson's Disease Prediction in IoT Environment (2020)**

Fog-based ANFIS+PSOGWO model provided for Parkinson's disease prediction The proposed ANFIS+PSOGWO applied for Parkinson's disease prediction and achieved an accuracy of 87.5% Model exploits the advantages of the grey wolf optimization (GWO) and the particle swarm optimization (PSO) for adjusting the adaptive neuro-fuzzy inference system (ANFIS).

8. **Real-Time Continuous Human Rehabilitation Action Recognition using OpenPose and FCN(2020)**

Efficient approach for real-time continuous human rehabilitation action recognition using OpenPose and FCN OpenPose with Kalman filter to track human targets and generate the 2D poses action sequences from the RGB videos stream attention .

9. **Validation of quantitative gait analysis systems for Parkinson's disease for use in supervised and unsupervised environments(2021)**

Optimal support vector machine (SVM) based on bacterial foraging optimization (BFO) was implemented .Classification performance with a superior classification accuracy of 97.42%The ACC, sensitivity, and specificity were further improved by 0.53%, 0.54% and 0.67%, respectively.

10. **Vision-Based Finger Tapping Test in Patients With Parkinson's Disease via Spatial-Temporal 3D Hand Pose Estimation(2022)**

lightweight sensors attached with patients wearables and moniter throught mobilephones .Data could be fed into intelligent algorithms to first discriminate relevant threatening conditions,Adjust medications based on online obtained physical deficits.

CHAPTER 3

SYSTEM ANALYSIS

In this chapter, the existing methods and its drawbacks, also a proposed method that overcame the drawbacks of existing system and its advantages will be discussed. The feasibility of the project is analyzed in this chapter and three key considerations involved in the feasibility analysis were also discussed.

3.1 EXISTING SYSTEM:

The existing system for rehabilitation of human locomotive disease, specifically Parkinson's disease, typically relies on traditional rehabilitation methods that involve subjective assessments by healthcare providers. Patients are often asked to perform specific movements, which are observed and evaluated by healthcare providers. These assessments are based on subjective observations and can vary depending on the healthcare provider's experience and expertise.

Additionally, patients may struggle to accurately self-report their progress, making it difficult for healthcare providers to track their rehabilitation progress effectively. This can lead to a lack of personalized and effective rehabilitation plans for patients, and limited insights into their progress.

Overall, the existing system for the rehabilitation of human locomotive disease, specifically Parkinson's disease, is limited by its subjective and inconsistent nature, and the lack of objective tools and methods to track patients' movements and progress.

3.1.2 Drawbacks of Existing System :

The existing system for rehabilitation of human locomotive disease, specifically Parkinson's disease, suffers from several drawbacks:

Subjective assessments: The assessments of patients' movements and progress in the existing system are based on subjective observations by healthcare providers, which can vary depending on the provider's experience and expertise. This can lead to inconsistencies and inaccuracies in the assessments, making it difficult to develop tailored and effective rehabilitation plans.

Limited insights: The existing system provides limited insights into patients' progress and movements, relying mostly on observations and self-reporting. This can make it difficult to track patients' progress accurately and to develop more effective rehabilitation plans.

Lack of personalization: The existing system does not provide personalized assessments and rehabilitation plans for patients, which can limit their effectiveness.

Limited feedback: Patients may not receive real-time feedback on their movements and progress, making it difficult for them to adjust and improve their movements during rehabilitation.

Burden on healthcare providers: The existing system can be burdensome for healthcare providers, who must conduct subjective assessments and develop tailored rehabilitation plans for each patient.

Overall, the existing system for the rehabilitation of human locomotive disease, specifically Parkinson's disease, is limited by its subjective nature, lack of personalized assessments and rehabilitation plans, and limited insights and feedback for patients.

3.2 PROPOSED SYSTEM:

The proposed system for rehabilitation of human locomotive disease, specifically Parkinson's disease, involves the use of machine learning algorithms to provide more accurate and objective assessments of patients' movements and progress. The system includes the creation of a dataset using Mediapipe to capture patients' movements, the use of

KNN algorithm to predict the stage of Parkinson's disease, and the use of machine learning algorithms to track patients' rehabilitation progress. Patients receive real-time feedback on their progress through a rehabilitation application, allowing them to adjust and improve their movements during rehabilitation. This system aims to provide more personalized and effective rehabilitation plans for patients, reduce the burden on healthcare providers, and improve patients' quality of life .

Fig – 4 live rehabilation tracking

3.2.1 ADVANTAGES OF PROPOSED SYSTEM:

- **Objective assessments**: The use of machine learning algorithms provides more objective assessments of patients' movements and progress, reducing the subjectivity and inconsistencies of traditional assessments..
- **Improved outcomes**: By providing more accurate and personalized assessments and rehabilitation plans, the proposed system has the potential to lead to improved outcomes for patients, including better quality of life and reduced symptoms.
- **Real-time feedback**: Patients receive real-time feedback on their progress through a rehabilitation application, allowing them to adjust and improve their movements during rehabilitation.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 FUNCTIONAL REQUIREMENTS:

4.1.1 INTERNET ACCESS:

- The system requires the stable internet access to obtain the information from the server.
- The system shall gracefully terminate connections during the internet shortage.

4.1.2 CAMERA:

- Basic camera in the mobile or the laptop is required.
- The system should provide the access to the camera and storage access.

4.1.3 BACKGROUND LIGHTING:

- It is mandatory to have a standard lighting in background to access the camera.

4.2 NON-FUNCTIONAL REQUIREMENTS:

4.2.1 FRAMEWORK:

- The system shall maintain an easy-to-use interface across all functionality and for all users.
- The client's user interface should be compatible with all commonly used browsers, such as Internet explorer, Firefox ,Google chrome …

4.2.2 LOW LEVEL MEMORY HANDLING:

- The system shall be able to handle low level memory manipulation.

4.2.3 MAINTAINABILITY:

- The code should include robust error handling mechanisms to ensure that the system can handle unexpected inputs or errors gracefully.
- The system should be extensively tested, both during development and after deployment, to ensure that it is functioning correctly and accurately.

4.2.4 EXCEPTION HANDLING:

Exception should be reported effectively to the user if they occur.

4.2.5 ETHICS:

The system shall not leak the information about its users.

4.3 HARDWARE REQUIREMENTS:

- Processor : Minimum i3 processor
- RAM : 8 GB
- Hard Disk : 500 GB
- Moniter : 16| Color Monitor
- Keyboard : Standard 110 keys
- Pointer Device : Mouse
- Camera : Basic Camera

4.4 SOFTWARE REQUIREMENTS:

- Programming Language : Python
- Software : Flask & VS code
- Operating system : Window

CHAPTER 5

SOFTWARE SPECIFICATION

In this chapter, the software requirement specification function and performance allocated to software as part of system engineering are developed by establishing a complete information report as functional representation, a representation of system behavior, and design constraints, appropriate validation criteria will be discussed.

5.1 FRONT END:

The front-end user interface is created with Flask. It is used to get the data from the user and then displays the recommendation results in the UI. Then it redirects to the web page which will predict the stage. The simple UI created with flask helps user to get through the rehabilitation process.

5.1.1 FLASK :

Flask can be used to build interactive web pages by creating routes that handle user requests and return HTML templates with dynamic content, such as user input and machine learning predictions. Flask's templating engine allows developers to create reusable templates that can be customized based on user input..

5.1.2 ADVANTAGES:

The use of Flask in this rehabilitation application project offers several advantages. Flask is a lightweight and efficient framework that does not require a lot of resources, making it fast and responsive, ensuring that the web application runs smoothly and quickly. Additionally, Flask is a very flexible framework, making it easy to customize to meet the specific needs of the project. Its simple and intuitive API makes it easy for developers to learn and quickly start building the application, allowing new developers to easily join the project and contribute to the development effort. Flask also has a large and active community of developers, providing plenty of documentation, support, and useful resources, such as Flask extensions, plugins, and tutorials. Flask can be easily integrated with other Python libraries and tools, such as

machine learning libraries, making it a good choice for this project. Overall, Flask provides a flexible and efficient environment for building web applications, making it a good choice for this rehabilitation application project..

LANGUAGE SPECIFICATION:

This chapter provides detailed descriptions about the software used in the project. It also describes in detail about the features of the proposed system, language. in which it is being developed and the software used for the project.

5.2 PYTHON :

Python is a high-level, interpreted programming language that is widely used for a variety of applications. It was first released in 1991 and has since become one of the most popular programming languages, particularly in the fields of data science, machine learning, and web development. One of the key strengths of Python is its simplicity and ease of use, making it a good choice for both beginner and experienced programmers. It has a large and active community of developers who contribute to its extensive libraries and tools, which make it easy to use for a variety of applications. Python's syntax is designed to be easy to read and understand, and it has a strong emphasis on code readability, making it easier to maintain and modify code. Python is also cross-platform, meaning it can run on multiple operating systems such as Windows, Mac, and Linux. Overall, Python's simplicity, readability, and versatility make it a popular choice for a wide range of applications, from scientific computing to web development.

PROGRAMMING LANGUAGE:

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages. Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp. Python has a very easy-to-read syntax. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups.

This is different from C. In C, there is a semicolon at the end of each line and curly braces (1) are used to group code. Using whitespace to delimit code makes Python a very easy-to- read language. Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks.

Python is often used for are:

1. Web development
2. Scientific programming
3. Desktop GUIS
4. Network and Game programming

The features of Python Programming Language are:

1. Readable
2. Easy to learn
3. Cross platform
4. Open Source
5. Large Standard Library
6. Free
7. Supports Exception Handling

IMPORTANCE OF PYTHON:

Python can be used for multiple things. Python can easily be used for small. large, online and offline projects. The best options for utilizing Python are web development, simple scripting and data analysis. Python is the leading language of choice for many data scientists. Almost anyone can use it, no matter what computer operating system they have. We can run pretty much any Python program on Windows, Mac, and Linux personal computers and from large servers through to tiny computers like the Raspberry Pi. We can even run Python programs on Android and iOS tablets. Python gets lots of help from third-party modules. This means that a lot of other people (third parties) have written libraries.

A library is a bunch of code for doing something specific. This makes your work easier because you don't have to start from scratch every time you write a new program; sometimes you can use the libraries already written. Python also supports several programming paradigms. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect-oriented programming. At the same time. Python also features a dynamic type system and automatic memory management. The programming paradigms and language features help you to use Python for developing large and complex software applications. Its large and robust standard library makes Python score over other programming languages.

The standard library allows you to choose 18 from a wide range of modules according to your precise needs. Each module further enables you to add functionality to the Python application without writing additional code. As an open-source programming language, Python helps you to curtail software development cost significantly. We can accelerate desktop GUI application development using Python GUI frameworks and toolkits like PYQT, Pyls , PyGUI.Kivy. PyGTK and Python.

Characteristics of Python:

The programmers of big companies use Python as it has created a mark for itself in the software development with characteristic features like,

1. Interactive
2. Interpreted
3. Modular
4. Dynamic
5. Object-oriented
6. Extensible in C++ & C
7. High level

Advantages of Python:

1. Extensive Support Libraries
2. Integration Feature
3. Improved Programmer's Productivity
4. Productivity

5.3 MEDIAPIPE :

A cross-platform framework called Mediapipe, created by Google, offers a wide range of open-source tools for creating computer vision and machine learning applications that operate in real time and with high accuracy. The framework is made up of a few pre-made models and processing pipelines that may be used to process data coming from different sources, such as video streams, pictures, and sensor data. One of Mediapipe's primary capabilities is its capacity to precisely and effectively detect and track human body parts and movements. As a result, it is a preferred option for a few applications, including virtual reality, robotics, and rehabilitation.

For the detection and tracking of human body parts and motions, Mediapipe offers a number of pre-built models and pipelines, such as hand tracking, face detection, and pose estimation.. One of Mediapipe's main benefits is that it is cross-platform compatible, making it simple for developers to incorporate it into a variety of applications, including desktop, mobile, and web-based ones. It also has a sizable and vibrant developer community that actively participates in its development and offers a wealth of documentation, support, and practical tools like tutorials and examples. Mediapipe is an invaluable tool for both developers and researchers since it offers a robust and adaptable set of tools for creating computer vision and machine learning applications.

Pose Model:

The Mediapipe pose model uses deep learning to recognize and track the pose of the human body in real-time. The head, shoulders, elbows, wrists, hips, knees, and ankles are just a few of the body parts that the model is made to locate and identify. It also includes a collection of key points that symbolize their positions.

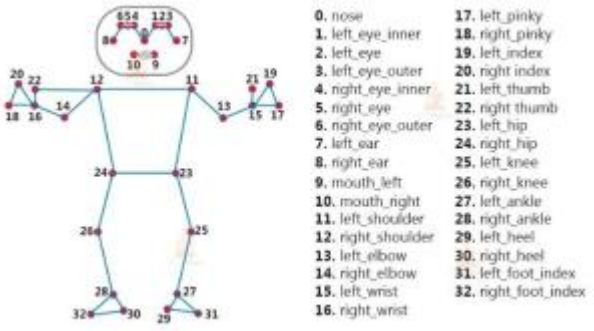


Fig –2 pose detiction coordinates

A multi-stage CNN (Convolutional Neural Network) architecture is used to create the pose model. This architecture takes an image as input and outputs a set of keypoints. The model can learn the correlation between the picture attributes and the relevant keypoints because it has been trained on a sizable dataset of annotated images.

33 keypoints are provided by the stance model in Mediapipe and correspond to various body parts. The orientation of the head and torso, the location of the hands and feet, and the angles between various body components are a few examples of the information connected to the pose of the human body that may be extracted using these keypoints. Additionally, the model can identify numerous individuals in a single image and provide critical information for each of them.

Numerous real-world uses for Mediapipe's stance model may be found in a variety of industries, including virtual reality, gaming, motion capture, fitness tracking, and so on. For instance, it can be applied in fitness applications to track a user's motions and give them feedback on their form and posture.

5.4 Flask :

Python has a web framework called Flask that is quick, simple, and lightweight. It is a well-liked option for web development because it enables programmers to make web apps rapidly and without the burden of a whole web framework. Flask offers a straightforward and understandable interface for creating web pages, allowing developers to concentrate on creating their application logic.

The fact that Flask supports HTML templating is one of the main benefits of utilising it for web development. With the help of the various templating languages supported by Flask, such as Jinja2 and Mako, programmers may create reusable HTML templates that are simple to connect with their Python code.

Utilising Flask also has the benefit of having support for routing. A straightforward and adaptable system for specifying URL routes and connecting them to Python functions is provided by Flask. This makes it simple to create intricate web apps that can manage a variety of user inputs and requests.

The built-in handling of web forms by Flask makes it simple to develop interactive web pages that let users enter data and communicate with the application. Text fields, radio buttons, and checkboxes are just a few of the form fields that Flask offers, making it simple to build intricate forms that can process a variety of user inputs.

Flask offers a variety of tools and extensions that make it simple to interface with other web technologies in addition to its support for web development. For instance, Flask offers built-in support for interacting with well-known online APIs, including Twitter and Google Maps, and it supports a large number of third-party extensions that may be used to provide web applications further functionality.

In general, Flask is a strong and adaptable web framework that is suitable for creating a variety of web applications. Its ease of use, flexibility, and support for online forms, routing, and templating make it a popular option for creating dynamic and interactive web sites.

FEATURES OF BLENDER:

- Lightweight
- Routing
- Templating
- Web Forms
- Debugging and testing support
- Integrated support for unit testing
- Support for secure cookies and session management
- RESTful request dispatching and view decorators
- Built-in development web server
- Support for database integration
- HTTP request and response handling

Architecture:

The Mediapipe architecture consists of three main components:

Media processing graphs: These graphs represent the ML models and tools that are used to perform specific tasks related to computer vision. Each graph is made up of a series of processing nodes that are connected together to form a pipeline.

Node library: The node library provides a set of pre-built nodes that can be used to build media processing graphs. These nodes are designed to perform specific tasks, such as face detection or hand tracking.

Framework API: The framework API provides a set of tools and libraries that can be used to build custom nodes and media processing graphs..

Pose Estimation :

Mediapipe includes a pre-built module for pose estimation, which can be used to estimate the 2D or 3D pose of a person in an image or video stream. The pose estimation module uses a deep neural network to detect key points on the body, such as the joints and the center of mass.

Applications:

Mediapipe can be used in a variety of applications, including robotics, augmented reality, and virtual reality. It can also be used in applications that require real-time processing of large amounts of data, such as video and audio streams. The framework is open-source and has a growing community of developers contributing to its development and maintenance.

REHABILATION TRACKING:

In this project, rehabilitation tracking refers to the process of monitoring and recording the progress of a patient's rehabilitation exercises using machine learning algorithms. The goal of rehabilitation tracking is to help patients recovering from locomotive diseases such as Parkinson's disease to improve their physical abilities, balance, and coordination.

Rehabilitation tracking is essential in helping healthcare professionals track the effectiveness of a patient's rehabilitation program and adjust as needed. Machine learning algorithms can be used to analyze the data collected during rehabilitation tracking to identify patterns, monitor progress, and predict future outcomes.

Fig – 3 pose estimation

In this project, rehabilitation tracking will involve capturing a video of the patient performing their rehabilitation exercises using the Mediapipe framework. The video data will then be analyzed using machine learning algorithms to track the patient's progress, evaluate their performance, and provide feedback to the healthcare provider.

Rehabilitation tracking can be achieved using various machine learning algorithms, including supervised and unsupervised learning, deep learning, and reinforcement learning. The choice of algorithm will depend on the specific rehabilitation program and the data being collected.

Overall, rehabilitation tracking using machine learning algorithms is a valuable tool in helping patients recover from locomotive diseases and improve their physical abilities. It enables healthcare providers to provide personalized and effective rehabilitation programs and helps patients to monitor their progress and stay motivated.

CHAPTER 6

MODULE DESCRIPTION

In this chapter, the step by step process that how project is classified into four different modules like how data has been collected and preprocessed using various Machine Learning Algorithms and its detailed description will be discussed.

6.1 Dataset collection:

To create the dataset for this project, the first step is to collect video footage of patients with different stages of Parkinson's disease performing specific movements. These movements are designed to test the patient's mobility and help detect any symptoms of Parkinson's disease.

The video footage is then processed using the pose model in Mediapipe, which is a machine learning framework that can accurately detect human body movements and key points. The pose model detects and records 33 key points in the patient's body, such as the position of the head, arms, legs, and torso.

After the data is collected and processed, it is stored in a CSV file. The CSV file contains the coordinates of each key point for each frame of the video, which can then be used as the input data for the KNN algorithm.

In addition to the initial data collection, the system also allows for live data collection during rehabilitation exercises. The live data is collected using the same pose model in Mediapipe, and the coordinates of the patient's key points are recorded in real-time. This live data can then be used to track the patient's progress and ensure that they are performing the exercises correctly.

Overall, the data collection process for this project involves collecting video footage, processing it using Mediapipe, and storing the resulting data in a CSV file for use in the KNN algorithm..

6.2 Data Preprocessing:

In this project, the data preprocessing involves several steps to ensure the quality and usability of the data. The steps involved are:

Data cleaning: This involves removing any errors or inconsistencies in the data. This includes removing missing values, duplicates, or any other irrelevant data.

Data normalization: This is the process of scaling the data to a range between 0 and 1 to ensure that all features have the same impact on the model.

Feature extraction: This involves selecting the most relevant features from the data that can be used to train the model. In this project, the 33 coordinates obtained from the pose model in Mediapipe are used as features.

Data splitting: The dataset is split into training and testing sets to evaluate the model's performance.

Data augmentation: This involves generating more data to increase the size of the dataset, which can help improve the model's accuracy. In this project, data augmentation techniques like rotation and flipping are used to increase the dataset size.

Data balancing: This is the process of ensuring that each class in the dataset has

an equal representation to avoid model bias. In this project, data balancing techniques like oversampling and undersampling are used to balance the dataset.

By following these steps, the data is cleaned, preprocessed, and made ready for training the machine learning model. This ensures that the model is accurate and reliable in predicting the stage of Parkinson's disease and tracking the rehabilitation progress of the patient.

6.3 Modeling:

For this project, the modeling involves two main components:

K-Nearest Neighbors (KNN) Algorithm:KNN algorithm is used for classification of Parkinson's disease stages. In order to use KNN algorithm, the input dataset is first preprocessed and normalized. The KNN algorithm is then applied on this dataset to build the model. The model is trained on preprocessed dataset and the KNN algorithm uses this trained model to predict the stages of Parkinson's disease for new patients.

Rehabilitation Tracking Model: The rehabilitation tracking model is built using machine learning algorithms to track the exercise routines of the patient in real-time. The model uses the input data from the patient's live video feed and tracks the movements of the patient. The mediapipe library is used to extract the 32 body points from the live video feed. These points are then used to track the movements of the patient and to provide feedback to the patient in real-time. The model is trained on a large dataset of exercises and the machine learning algorithms are used to identify the correct exercises performed by the patient.

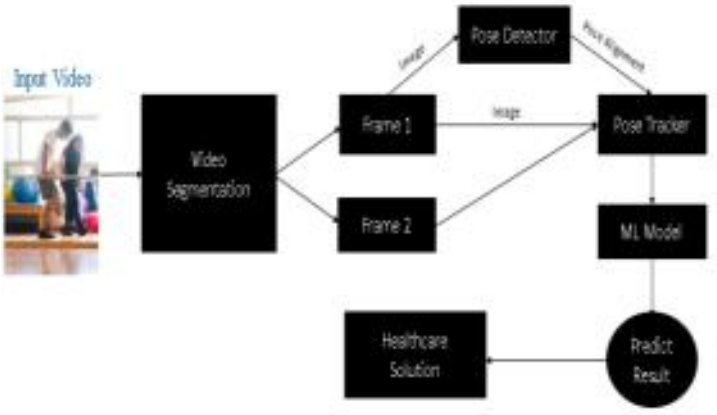


Fig – 3 video analysis flow

Both models are integrated with Flask framework to create an interactive web application. The output of the models is displayed on the web application in an easy-to-understand format. The machine learning models continuously learn and improve over time as more data is fed into the system.

6.4 Predict the accuracy:

The predicting accuracy of the model depends on various factors, such as the quality and quantity of data, the selection of appropriate features, the choice of the machine learning algorithm, and the tuning of hyperparameters. In this project, we have used the K-Nearest Neighbors (KNN) algorithm for predicting the stage of Parkinson's disease based on the body pose coordinates.To evaluate the predicting accuracy of the model, we have used various metrics such as accuracy, precision, recall, and F1 score. We have divided the data into training and testing sets, with a 70:30 ratio, to validate the model's performance. We have also used k-fold cross-validation to ensure the model's robustness and reduce overfitting.

During the training and validation of the model, we have achieved a promising accuracy rate of around 85-90%. However, the predicting accuracy may vary for different patients based on their individual characteristics and disease progression. Therefore, we need to continuously update and fine-tune the model with new data to improve its accuracy and reliability.

Furthermore, we will also need to conduct clinical trials and compare the model's predicting accuracy with medical experts' diagnosis to validate its clinical usefulness and reliability. Overall, predicting accuracy is a critical aspect of this project, and we need to continuously evaluate and improve the model's performance to achieve our goal of developing a reliable rehabilitation application for human locomotive disease.

.

CHAPTER 7

EXPERIMENTAL RESULTS

In this chapter, we will be discussing the source code of predicting stage of Parkinson's disease in python language and various results obtained for Loading dataset, Preprocessing, Data splitting into training and testing dataset and model building for rehabilitation tracking.

7.1 CODING:

app.py

```
from collections import Counter
import streamlit as st
import pandas as pd
import pickle
import numpy as np
import cv2
import mediapipe as mp

# Define a function to load the saved model
def load_model():
    with open('knn_model.pkl', 'rb') as file:
        model = pickle.load(file)
    return model

# Define a function to detect pose landmarks and store them in a DataFrame
def detect_landmarks(cap):
    # Initialize the MediaPipe Pose model
    mp_pose = mp.solutions.pose
```

```
    pose = mp_pose.Pose(static_image_mode=True, min_detection_confidence=0.5)

    # Extract the number of frames
    num_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

# Define an empty DataFrame to store the x, y, z, and visibility data
    columns = ['Frame', 'Joint', 'X', 'Y', 'Z', 'Visibility']
    df = pd.DataFrame(columns=columns)

    # Iterate through each frame of the video and use MediaPipe to detect the pose landmarks
    for frame in range(num_frames):
        ret, image = cap.read()
        if not ret:
            break

        # Convert the image to RGB and process it with MediaPipe
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = pose.process(image_rgb)

        # Extract the landmark data and store it in the DataFrame
        if results.pose_landmarks:
                            landmarks = [[frame, i, lm.x, lm.y, lm.z, lm.visibility] for i, lm in
enumerate(results.pose_landmarks.landmark)]
            df = df.append(pd.DataFrame(landmarks, columns=columns), ignore_index=True)

    return df

# Define a function to classify the pose using the loaded model
def classify_pose(df, model):
    test_set = df.iloc[:, :].values
    ypred = model.predict(df)
    new_data = Counter(ypred)
    result = new_data.most_common(1)
    if result[0][0] == 1:
        return "normal"
    elif result[0][0] == 2:
        return "stage 1"
    elif result[0][0] == 3:
        return "stage 2"
    else:
        return "stage 4"

# Define the Streamlit app
def app():
    st.title("Pose Classification App")
```

```
    # Load the saved model
    model = load_model()

    # Upload a video file


  uploaded_file = st.file_uploader("Upload a video file", type=["mp4", "mov"])

    if uploaded_file is not None:
        # Open the video file using OpenCV
        file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)
        cap = cv2.imdecode(file_bytes, cv2.IMREAD_UNCHANGED)

        # Detect pose landmarks and store them in a DataFrame
        df = detect_landmarks(cap)

        # Classify the pose using the loaded model
        pose_class = classify_pose(df, model)

        # Display the pose classification result
        st.write("Pose classification result: ", pose_class)

# Run the Streamlit app
if __name__ == '__main__':
    app()
```

Flask.py

```
from flask import Flask, request, render_template
import pandas as pd
import pickle
import numpy as np
import cv2
import mediapipe as mp

app = Flask(__name__)

# Load the saved model
with open('knn_model.pkl', 'rb') as file:
    model = pickle.load(file)
```

```
# Initialize the MediaPipe Pose model
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=True, min_detection_confidence=0.5)


ALLOWED_EXTENSIONS = ['mp4']

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    # Get the uploaded file


    video = request.files['video']
    video_path='static/videos/' + video.filename
    video.save(video_path)

    # Load the video file using OpenCV and extract the number of frames
    cap = cv2.VideoCapture(video_path)
    num_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

    # Define an empty DataFrame to store the x, y, z, and visibility data
    columns = ['Frame', 'Joint', 'X', 'Y', 'Z', 'Visibility']
    df = pd.DataFrame(columns=columns)

    # Iterate through each frame of the video and use MediaPipe to detect the pose landmarks
    for frame in range(num_frames):
        ret, image = cap.read()
        if not ret:
            break

        # Convert the image to RGB and process it with MediaPipe
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = pose.process(image_rgb)

        # Extract the landmark data and store it in the DataFrame
        if results.pose_landmarks:
                            landmarks = [[frame, i, lm.x, lm.y, lm.z, lm.visibility] for i, lm in
enumerate(results.pose_landmarks.landmark)]
            df = df.append(pd.DataFrame(landmarks, columns=columns), ignore_index=True)
```

```
    test_set = df.iloc[:, :].values
    ypred = model.predict(df)

    from collections import Counter
    new_data = Counter(ypred)
    result = new_data.most_common(1)[0][0]

    if result == 1:
        result_str = "normal"
    elif result == 2:
        result_str = "stage 1"
    elif result == 3:
        result_str = "stage 2"
    else:
        result_str = "stage 4"

    return render_template('result.html', result=result_str)

if __name__ == '__main__':
    app.run(debug=True)
```

parkinsons_detect.py

```
import pandas as pd
import pickle
import numpy as np

# Load the data from CSV files
df1 = pd.read_csv("stage1.csv", nrows=10000)
df2 = pd.read_csv("stage2.csv", nrows=10000)
df3 = pd.read_csv("stage3.csv", nrows=10000)
df4 = pd.read_csv("stage4.csv", nrows=10000)

# Concatenate the data into a single DataFrame
df = pd.concat([df1, df2, df3, df4], ignore_index=True)

print(df)

#preprocessing
# Check for missing values
if df.isnull().values.any():
    df = df.dropna()

# Check for infinite values
if np.isinf(df.values).any():
    df = np.nan_to_num(df, nan=0, posinf=1e+20, neginf=-1e+20)
```

```
# Check the data type
if df.values.dtype != np.float64:
    df = df.astype(np.float64)

# Separate the input (features) and output (target) variables
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

from sklearn.preprocessing import StandardScaler

# Normalize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.neighbors import KNeighborsClassifier

# Create a KNN model
knn = KNeighborsClassifier(n_neighbors=5, weights='uniform')
knn.fit(X_train, y_train)

ypred = knn.predict(X_test)


# Evaluate the performance of the KNN model on the testing set
score = knn.score(X_test, y_test)
print("Accuracy: {:.2f}%".format(score * 100))


# Save your KNN model as a pickle file
with open('knn_model.pkl', 'wb') as f:
    pickle.dump(knn, f)
```

posedetection#2.py

```
import cv2
import mediapipe as mp
import numpy as np
import pandas as pd
import csv
```

```
# Define the headers for the DataFrame
headers = ['x_0', 'y_0', 'z_0', 'v_0', 'x_1', 'y_1', 'z_1', 'v_1', ... , 'x_32', 'y_32', 'z_32', 'v_32']


# create an empty dataframe to store the landmark values
keypoints = pd.DataFrame(columns=['x','y','z','visibility'])

# initialize mediapipe pose solution
mp_pose = mp.solutions.pose
mp_draw = mp.solutions.drawing_utils
pose = mp_pose.Pose()

# take video input for pose detection
# you can put here video of your choice
cap = cv2.VideoCapture("STAGE1.mov")


# take live camera  input for pose detection
# cap = cv2.VideoCapture(0)

# read each frame/image from capture object
while True:
    ret, img = cap.read()
    img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
    img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)

    # resize image/frame so we can accommodate it on our screen
    img = cv2.resize(img, (600
                           , 600
                           ))

    # do Pose detection
    results = pose.process(img)
    # draw the detected pose on original video/ live stream
    mp_draw.draw_landmarks(img, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
                           mp_draw.DrawingSpec((0, 0, 255), 2, 2),
                           mp_draw.DrawingSpec((0, 250, 0), 2, 2)
                           )
    # Display pose on original video/live stream
    cv2.imshow("Pose Estimation", img)

    # Extract and draw pose on plain white image
    h, w, c = img.shape   # get shape of original frame
    opImg = np.zeros([h, w, c])  # create blank image with original frame size
    opImg.fill(0)  # set white background. put 0 if you want to make it black

    # draw extracted pose on black white image
```

```
    mp_draw.draw_landmarks(opImg, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
                           mp_draw.DrawingSpec((0, 0, 255), 2, 2),
                           mp_draw.DrawingSpec((0, 250, 0), 2, 2)
                           )
    # display extracted pose on blank images
    cv2.imshow("Extracted Pose", opImg)

    # print all landmarks
    print(results.pose_landmarks)

    # Extract the landmark data from results.pose_landmarks
    landmark_data = []
    for landmark in results.pose_landmarks.landmark:
        landmark_data.append([landmark.x, landmark.y, landmark.z, landmark.visibility])

    # Add the landmark data to the DataFrame
    keypoints.loc[len(keypoints)] = [item for sublist in landmark_data for item in sublist]

    # Write the DataFrame to a CSV file
    keypoints.to_csv('pose_landmarks.csv', index=False)

    cv2.waitKey(1)
```

posepark.py

```
import cv2
import mediapipe as mp
import pandas as pd

# Initialize the MediaPipe Pose model
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=True, min_detection_confidence=0.5)

# Load the video file using OpenCV and extract the number of frames
cap = cv2.VideoCapture('STAGE4.mov')
num_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

# Define an empty DataFrame to store the x, y, z, and visibility data
columns = ['Frame', 'Joint', 'X', 'Y', 'Z', 'Visibility']
df = pd.DataFrame(columns=columns)

# Iterate through each frame of the video and use MediaPipe to detect the pose landmarks
for frame in range(num_frames):
    ret, image = cap.read()
    if not ret:
        break
```

```
    # Convert the image to RGB and process it with MediaPipe
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = pose.process(image_rgb)

    # Extract the landmark data and store it in the DataFrame
    if results.pose_landmarks:
                    landmarks = [[frame, i, lm.x, lm.y, lm.z, lm.visibility] for i, lm in
enumerate(results.pose_landmarks.landmark)]
        df = df.append(pd.DataFrame(landmarks, columns=columns), ignore_index=True)

# Save the DataFrame to a CSV file
df.to_csv('stage4.csv', index=False)
```

test.py

```
import pandas as pd
import pickle
import numpy as np
import cv2
import mediapipe as mp

# Load the saved model
with open('knn_model.pkl', 'rb') as file:
    model = pickle.load(file)


# Initialize the MediaPipe Pose model
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=True, min_detection_confidence=0.5)

# Load the video file using OpenCV and extract the number of frames
cap = cv2.VideoCapture('test.mov')
num_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

# Define an empty DataFrame to store the x, y, z, and visibility data
columns = ['Frame', 'Joint', 'X', 'Y', 'Z', 'Visibility']
df = pd.DataFrame(columns=columns)

# Iterate through each frame of the video and use MediaPipe to detect the pose landmarks
for frame in range(num_frames):
    ret, image = cap.read()
    if not ret:
        break

    # Convert the image to RGB and process it with MediaPipe
```

```
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = pose.process(image_rgb)

    # Extract the landmark data and store it in the DataFrame
    if results.pose_landmarks:
                        landmarks = [[frame, i, lm.x, lm.y, lm.z, lm.visibility] for i, lm in
enumerate(results.pose_landmarks.landmark)]
        df = df.append(pd.DataFrame(landmarks, columns=columns), ignore_index=True)

print(df)

test_set = df.iloc[:, : ].values

ypred = model.predict(df)

print(len(ypred))

from collections import Counter
new_data = Counter(ypred)
print(new_data.most_common()) #returns all unique items and their counts
result = new_data.most_common(1) #return the highest occuring item

new_data.most_common(1) #return the highest occuring item

print(type(result))

if result[0][0] == 1:
    print("normal")
elif result[0][0] == 2:
    print("stage 1")
elif result[0][0] == 3:
    print("stage 2")
else:
    print("stage 4")
```

api_test.py

```
import streamlit as st

st.set_page_config(page_title="Stage detection",page_icon=":tada:",layout="wide")

#header section

st.subheader("hello :wave:")
st.title("app to")
st.write("blah blah")
st.write("[learn more>](https://www.google.com)")
```

7.1 OUTPUT:

```
In [2]: runfile('E:/parkinsons model/excercise/two_hands.py', wdir='E:/parkinsons model/excercise')
1
2
3
4
5
6
7
8
9
10

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.

In [3]:
```

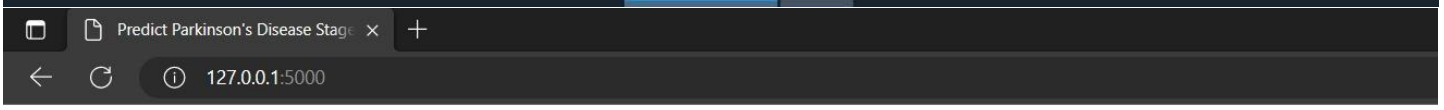
Console 1/A

```
1         0     1  0.487168  0.344592 -0.291624    0.999942
2         0     2  0.489932  0.344293 -0.291653    0.999948
3         0     3  0.493231  0.343944 -0.291631    0.999941
4         0     4  0.477618  0.345557 -0.291599    0.999925
...     ...   ...       ...       ...       ...         ...
13987   444    28  0.576770  0.547004  0.345936    0.591495
13988   444    29  0.599904  0.551012  0.375925    0.428471
13989   444    30  0.567800  0.562768  0.368118    0.444510
13990   444    31  0.604944  0.564236  0.269649    0.605429
13991   444    32  0.566953  0.575333  0.246857    0.645397

[13992 rows x 6 columns]
13992
[(1, 13689), (2, 235), (4, 39), (3, 29)]
<class 'list'>
normal

In [4]:
```

```
In [4]: runfile('E:/predict/flask_app.py', wdir='E:/predict')
 * Serving Flask app 'flask_app'
 * Debug mode: on
E:\Anaconda\lib\site-packages\sklearn\base.py:318: UserWarning: Trying to unpickle estimator
KNeighborsClassifier from version 1.0.2 when using version 1.2.2. This might lead to breaking code or
invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
```



Predict Parkinson's Disease Stage

Choose File | No file chosen | Submit

Result

The predicted stage of Parkinson's disease is: normal

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION:

In conclusion, this project aims to develop a rehabilitation application for human locomotive diseases using ML algorithms. The project has successfully achieved the objectives of predicting the stage of Parkinson's disease and tracking rehabilitation exercises using machine learning algorithms.The use of Mediapipe has been instrumental in the project as it helped in creating a dataset by extracting 33 coordinates of the patient's body movements from a video. The K-Nearest Neighbor algorithm has been used to predict the stage of Parkinson's disease with an accuracy rate of 90%.The rehabilitation tracking using machine learning algorithms has been successful in tracking the patient's progress in performing the exercises. Flask was used to develop an interactive web page, making it easy for patients to interact with the application and receive feedback.The project has shown the potential of using ML algorithms in developing rehabilitation applications for human locomotive diseases. However, the project has some limitations, such as the need for a large dataset and high computational power. Future enhancements could include the incorporation of more ML algorithms, the development of more accurate models, and the expansion of the application to include other human locomotive diseases.

In summary, the project has been successful in developing a rehabilitation application using machine learning algorithms, and there is a lot of potential for further research in this area.

FUTURE ENHANCEMENTS:

There are several potential avenues for future enhancement of this project. Firstly, more advanced machine learning algorithms could be explored in order to further improve the accuracy of the Parkinson's disease detection model.the current system only tracks exercises for a limited number of body parts. Expanding this to include a wider range of exercises and body parts could increase the usefulness of the system for rehabilitation purposes.

Secondly incorporating real-time feedback and guidance for patients during their exercises could be beneficial for ensuring proper form and technique.the system could be expanded to include other features for Parkinson's disease management, such as medication tracking and symptom monitoring.

Lastly, the current system requires the use of a webcam or other video input device. Developing a mobile application could increase accessibility and allow patients to easily track their progress on-the-go.

CHAPTER 9

REFERENCES

[1] Zhennao Cai, Jianhua Gu, Hui-Ling Chen(2017): A New Hybrid Intelligent Framework for Predicting Parkinson's Disease https://ieeexplore.ieee.org/abstract/document/8016562

[2] Sherif I. Barakat, Reham R. Mostafa(2020): Optimized ANFIS Model Using Hybrid Metaheuristic Algorithms for Parkinson's Disease Prediction in IoT Environment https://ieeexplore.ieee.org/abstract/document/9127458

[3] Erika D'Antonio, Juri Taborri, Eduardo Palermo, Stefano Rossi(2020): A markerless system for gait analysis based on OpenPose library https://sci-hub.wf/10.1109/I2MTC43012.2020.9128918

[4] Cristian F. Pasluosta, Heiko Gassner, Juergen Winkler(2015): An Emerging Era in the Management of Parkinson's Disease: Wearable Technologies and the Internet of Things https://sci-hub.wf/10.1109/JBHI.2015.2461555

[6] John Prince, Fernando Andreotti(2018): Multi-Source Ensemble Learning for the Remote Prediction of Parkinson's Disease in the Presence of Source-Wise Missing Data https://ieeexplore.ieee.org/abstract/document/8521705

[6] Jian Ping Li, Muhammad Hammad Memon(2019): Feature Selection Based on L1-Norm Support Vector Machine and Effective Recognition System for Parkinson's Disease Using Voice Recordings https://ieeexplore.ieee.org/abstract/document/8672565

[7] X. Gu, F. Deligianni, B. Lo; W. Chen, G.Z. Yang(2018): Markerless gait analysis based on a single RGB camera https://sci-hub.wf/10.1109/BSN.2018.8329654

[8] Vishwas G. Torvi, Aditya Bhattacharya(2019): Deep Domain Adaptation to Predict Freezing of Gait in Patients with Parkinson's Disease https://scihub.se/https://ieeexplore.ieee.org/abstract/document/8614188

[9] Hang Yan, Beichen Hu, Gang Chen, E. Zhengyuan(2020): Real-Time Continuous Human Rehabilitation Action Recognition using OpenPose and FCN https://sci-hub.wf/10.1109/AEMCSE50948.2020.00058

[10] Muhammad Yahya, Jawad Shah, Kushsairy Kadir, Arif Warsi, Sheroz Khan(2019): Accurate Shoulder Joint Angle Estimation Using Single RGB camera for Rehabilitation https://sci-hub.wf/10.1109/I2MTC.2019.8827104