DE-41 (EE) Abdullah, Aimen, Laiba

COLLEGE OF

ELECTRICAL AND MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY RAWALPINDI

2023

PARKINSON'S DISEASE DETECTION USING MACHINE LEARNING

COLLEGE OF

ELECTRICAL AND MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND

TECHNOLOGY RAWALPINDI

2023

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

PROJECT REPORT

DE-41 EE

Parkinson’s Disease Detection Using Machine Learning

Submitted to the Department of Electrical Engineering in partial fulfillment of the requirements for the degree of

Bachelor of Engineering

in

Electrical

2023

Sponsoring DS:

Sobia Hayee M.S

Submitted By:

Abdullah Sohail
Aimen Munawar
Laiba Aftab Bajwa

بسم الله الرحمن الرحيم

CERTIFICATE OF APPROVAL

It is to certify that the project **"Parkinson's Disease Detection Using Machine Learning"** was done by **NS Abdullah Sohail**, **NS Aimen Munawar**, and **NS Laiba Aftab Bajwa** under supervision of **Sobia Hayee M.S.**

This project is submitted to **Department of Electrical Engineering**, College of Electrical and Mechanical Engineering (Peshawar Road Rawalpindi), National University of Sciences and Technology, Pakistan in partial fulfilment of requirements for the degree of Bachelor of Electrical Engineering.

Students:

Abdullah Sohail

NUST ID: ____________________ **Signature:____________________**

Aimen Munawar

NUST ID: ____________________ **Signature:____________________**

Laiba Aftab Bajwa

NUST ID: ____________________ **Signature: ____________________**

APPROVED BY:

Sobia Hayee M.S

Project Supervisor: ____________________Date: ____________

DECLARATION

We affirm that the content presented in this Project Thesis is original and has not been submitted in support of any other degree or qualification at this or any other educational institution. We acknowledge that any act of plagiarism will result in full responsibility and may lead to disciplinary action, including the potential cancellation of our degree, based on the severity of the offense.

1. **Abdullah Sohail** ____________________

2. **Aimen Munawar** ____________________

3. **Laiba Aftab Bajwa**____________________

COPYRIGHT STATEMENT

ACKNOWLEDGMENTS

First and foremost, we express our gratitude to Allah Almighty for granting us the ability and courage to comprehend and overcome the challenges faced during this demanding endeavor. Alongside our collective efforts, the success of any project greatly relies on the guidance and support of numerous individuals. We extend our sincere appreciation to our supervisor, Ma'am Sobia Hayee whose invaluable assistance, support, and guidance were instrumental in our achievement.

Furthermore, we are deeply indebted to our beloved parents, whose unwavering support and patience played a vital role in our journey. They have consistently been there for us in times of need.

Additionally, we would like to thank our colleagues Muhammad Hassaan Aftab, Muhammad Arhum Mobin, Bakht Baidar and others for their support and invaluable guidance.

Lastly, we would like to express our heartfelt gratitude to the Department of Electrical Engineering for nurturing us throughout our academic years and enabling us to excel in our field. We extend our appreciation to all the faculty members and staff who have diligently contributed to providing us with exceptional facilities and guidance.

ABSTRACT

This research describes the design and development of a machine learning system for detecting Parkinson's disease. The system includes Python method creation, support vector machine (SVM) classifier building, and data analysis utilising UCI/Oxford university datasets[0]. Pitch, jitter, shimmer, and harmonic-to-noise ratio are among the variables retrieved from speech recordings of healthy and Parkinson's patients in the dataset. The technology also includes a mobile application that can detect Parkinson's disease from speech recordings using the classifier. Users can record their voice samples and save them locally on the device using the mobile application. The system is built on a Raspberry Pi device, which includes a microphone module that can gather voice signals and execute the classifier. The project's goal is to develop a low-cost, portable, and accurate Parkinson's disease diagnosis tool that may be used by anybody, anywhere.

SUSTAINABLE DEVELOPMENT GOALS

The goal of constructing a machine learning model for Parkinson's disease diagnosis coincides with SDGs 1 and 7, which focus on eradicating poverty and ensuring access to affordable and clean energy.

The project intends to improve the accessibility and affordability of health care services by developing a machine learning model that can identify Parkinson's disease based on voice data. This helps to eradicate poverty by developing low-cost and reliable diagnosis methods that can enable patients with Parkinson's disease receive prompt and appropriate treatment. Furthermore, the research promotes creativity by applying SVM, a powerful machine learning technique, to a fresh and demanding issue area. It offers more accurate and robust diagnosis by optimising model performance, opening the door to new possibilities in sectors such as telemedicine, speech recognition, and biomedical engineering.

Furthermore, the project emphasises clean energy by focusing on the development of a mobile application that will employ live speech recording as the input for the machine learning model. This is consistent with the goal of developing energy-efficient technology that reduce environmental impact and encourage responsible resource utilisation. The idea removes the need for storing and transferring large amounts of data, which can cost a significant amount of energy, by employing live voice recording. Furthermore, by utilising a mobile application, the concept uses the existing infrastructure of smartphones and wireless networks, which may be fueled by renewable energy sources.

In summary, the project of developing a machine learning model for Parkinson's disease detection contributes to SDGs 1 and 7 by improving health care accessibility and affordability, fostering innovation in machine learning and speech analysis, and promoting clean energy practises in the field of mobile computing.

<u>TABLE OF CONTENTS</u>

Contents

LIST OF ABBREVIATIONS

Acronyms:

SVM:	Support Vector Machine
IDE:	Integrated Development Environment
App:	Mobile Application

Introduction

This introduction provides a brief overview of the project, beginning with a critical analysis of the challenges and opportunities connected with utilising machine learning to diagnose Parkinson's disease. It then describes the project's goals, constraints, and requirements, as well as the deliverables that are expected. Finally, the chapter concludes with an organizational structure for the thesis.

1.1. Project Overview:

The purpose of this project was to develop a machine learning model for detecting Parkinson's disease using SVM and a mobile application that would use live voice recording as input to the model. Python was the programming language, and Anaconda and Spyder were the software environment and IDE, respectively.

The librosa and parselmouth tools were used to retrieve the model's voice features, and SVM was created with the sklearn library. The model's performance was evaluated using the accuracy score and confusion matrix measurements. The GridSearchCV function was used to fine-tune the model's hyperparameters.

During the testing phase, the model was applied to a new speech recording to generate a prediction. To validate the model's accuracy and efficacy, the forecast was compared to the person's actual state. The comparison demonstrated that the algorithm correctly identified the voice sample as belonging to a Parkinson's disease patient.

In summary, this project included the successful development of a machine learning model based on SVM for Parkinson's disease detection as well as a mobile application based on live voice recording. Anaconda and Spyder allowed for rapid software development and testing, whilst Python and its libraries allowed for rapid data processing and machine learning implementation. The comparison of projected and actual status confirmed the correctness of the SVM algorithm and the model's usefulness.

1.2. Problem Statement:

Machine learning is a powerful approach with applications ranging from health care to speech recognition to computer vision. Using machine learning to identify Parkinson's disease, on the other hand, raises issues and possibilities that affect the diagnosis's quality and accuracy. When using speech characteristics as input for the machine learning model, the model's resilience and dependability may suffer. SVM is a popular machine learning strategy for dealing with complex and nonlinear problems, but identifying the optimum hyperparameters for the model can be difficult and time-consuming. Although Python is a powerful programming language with several libraries and frameworks for data processing and machine learning, building and testing a mobile app that will use live audio recording as the model's input can be challenging and tough. As a result, the project proposes to develop a machine learning model for Parkinson's disease detection based on SVM and a mobile application that would use live voice recording as input to the model. The model must extract speech characteristics and apply SVM on the audio recording to be rapid and efficient. Python and its libraries will be utilised for data processing and machine learning, while Anaconda and Spyder will be used for the software environment and the IDE, respectively. To ensure the model's correctness and efficacy, testing and comparison with the person's real state will be undertaken. The goal of the project is to overcome the challenges of utilising machine learning to detect Parkinson's disease and create a low-cost and reliable diagnostic tool.

This model's development has the potential to improve various industries, including health care, voice recognition, and biomedical engineering. By developing more accurate and robust machine learning algorithms, this endeavour may help to improve these subjects and their applications.

1.3. Approach

The "Parkinson's Disease Detection Using Machine Learning" project aimed to develop a machine learning model for Parkinson's disease identification using SVM and a mobile application that used live speech recording as input to the model. The approach included several phases, including:

1.1.1. Data Processing:

The initial step was to use Python's librosa and parselmouth modules to analyse the speech recordings and extract the voice characteristics.

1.1.2. SVM Implementation:

After extracting the speech characteristics, we used Python's sklearn module to create SVM.

1.1.3. Hyperparameter Tuning:

The SVM model's hyperparameters were then tuned using Python's GridSearchCV function.

1.1.4. Mobile Application Development:

In Python, we utilised the sounddevice and soundfile libraries to create a mobile application that would use live speech recording as input for the model.

1.1.5. Integration:

Using SVM and a mobile application, we combined all of the components to construct a machine learning model for Parkinson's disease identification.

1.1.6. Testing:

Finally, we ran the model on a fresh speech recording and compared the forecast to the person's actual condition to confirm its accuracy.

Using SVM and a mobile application, we were able to develop an accurate and robust machine learning model for Parkinson's disease identification. The model might be utilised in telemedicine, voice recognition, and biomedical engineering, among other applications that need low-cost and accurate diagnostic tools.

1.4. Objectives:

The purpose of this project is to provide a software-based solution for Parkinson's disease diagnosis utilising machine learning and a mobile application, as well as to accomplish the following objectives:

- To attain a high degree of precision in the diagnostic procedure.
- To improve the machine learning model's performance.
- To verify that the voice feature extraction and SVM implementation are robust and reliable.
- To explore the potential applications of the model in low-cost and reliable diagnosis tools.
- To provide the groundwork for future growth and research in machine learning and voice analysis.

1.5. Specifications:

The Anaconda and Spyder software tools are robust and adaptable, and they were used in this study to develop and test a machine learning model for Parkinson's disease detection using speech features and SVM. Anaconda is a software environment that provides libraries and frameworks for data processing and machine learning such as librosa, parselmouth, sklearn, and nolds [4]. Spyder is a Python programming IDE with a user-friendly interface and debugging capabilities. In this project, Anaconda is used to install and maintain the libraries and frameworks required for the model's data processing and machine learning components. Spyder is used to write and run Python code that extracts speech features and applies the SVM algorithm. The use of both the Anaconda and Spyder tools allows for more efficient software development and testing, resulting in faster performance and higher model accuracy. Overall, Anaconda and Spyder are excellent tools for this project, providing the power and flexibility required for effective machine learning model building and testing.

1.6. Deliverables:

This project's deliverables include:

• A working model/prototype of a machine learning system that employs speech features and SVM to identify Parkinson's disease.

• Python programming is utilized for the data processing and machine learning components of the model.

• Creation and testing of a mobile application that will use live speech recording as input to the model.

• The model is being evaluated to ensure that it detects Parkinson's illness correctly.

• Documentation of the model's design, development, and testing processes.

1.7. Organization of Thesis:

The organization of this thesis is as follows:

Chapter 1: Introduction

This chapter provides a project summary, including project goals and an issue description. It also includes a brief explanation of the machine learning model and the mobile application, both of which are used as software tools in this project. It also highlights the methodologies used in project development and testing.

Chapter 2: Literature Review

This chapter examines the literature on machine learning, voice analysis, and SVM-based applications. It includes an in-depth study of the present state of the art in machine learning for Parkinson's disease identification.

Chapter 3: System Design

The general system architecture, as well as the data processing and machine learning components, are all explained in this chapter. It covers design elements like voice feature extraction, SVM implementation, and hyperparameter tuning. The development process is also described in this chapter, including the Python code for the data processing and machine learning components, as well as the mobile application that will use live audio recording as model input.

Chapter 4: Results and Analysis

This chapter evaluates the system's performance using a variety of metrics, including accuracy score, confusion matrix, and prediction time. It also compares the findings to the person's actual state.

Chapter 5: Future Work

This chapter describes potential future work that might be done to improve the system even more. It explores potential enhancements to the data processing and machine learning components, as well as potential new features.

Chapter 6: Conclusion

This chapter summarizes the project's research activities and highlights the major accomplishments. It concludes by looking at the project's contributions to machine learning and voice analysis.

Literature Reveiw

There is an unprecedented demand for low-cost and dependable diagnostic devices in today's digital age. Machine learning has emerged as an effective technique for addressing complex and challenging issues in a range of fields, including health care, speech recognition, and computer vision [1]. Parkinson's disease detection is a popular app for detecting a neurological disorder that affects millions of people worldwide [2]. Traditional diagnostic treatments, due to their high cost and complexity, may not be accessible or affordable to everyone. As a result, machine learning-based implementations of Parkinson's disease detection based on speech features and SVM have piqued the interest of researchers in recent years [3].

2.1. Background

This chapter provides a high-level overview of the machine learning model and the mobile app that will feed the model real-time speech samples. It also includes a review of the literature on the topics covered in our study. Each of the topics will be explored briefly in order to complement and enrich the objective of this work. The purpose of this chapter is to familiarize readers with our project's data processing and machine learning components, as well as current research in this field.

2.1.1. Introduction to Machine Learning

Machine learning is an artificial intelligence subfield that allows computers to learn from data and make predictions or judgments without being explicitly programmed [4]. Machine learning models, unlike conventional algorithms, can adapt to new data and improve their performance over time, making them suited for a broad variety of applications. As a result, they are now widely used in a variety of disciplines, including health care, voice recognition, and computer vision [5].

Machine learning combines the greatest aspects of statistics and computer science, which has fueled its widespread acceptance across many sectors in recent years. Data-driven insights and solutions are provided through machine learning. Data-driven models have the same flexibility as software operating on a processor-based system, but they are not constrained by human experts' assumptions or regulations. Machine learning models, as opposed to algorithms, have a genuine learning aspect. Because there is no predetermined logic, multiple models might find different patterns or correlations in the same data. Every separate job is carried out by a specialised model that can operate autonomously and independently of other models. As a consequence, the performance of one portion of the application has no effect on the other operations.

2.1.2. Anaconda and Spyder:

Anaconda and Spyder are Python programming and data science software tools[6]. Anaconda is a software environment that includes various data processing and machine learning libraries and frameworks, including as librosa, parselmouth, sklearn, and nolds [7]. Spyder is an IDE for Python programming that offers a user-friendly interface and debugging facilities. These tools may be used to create and validate machine learning models and applications.

Anaconda and Spyder are prominent data science and machine learning frameworks. Because of their diverse programming language and extensive collection of libraries and frameworks, they are well-suited for building complicated data processing and machine learning algorithms. Furthermore, their simple installation and setup make them an ideal platform for software development and testing.

2.1.3. Machine Learning Model Programming

The user may use a programming language or a graphical interface to describe the behaviour of the machine learning model. The programming language structure is especially well suited for dealing with complicated algorithms since it enables the user to express them numerically rather than sketching each component manually. Using a graphical interface, on the other hand, allows for simpler viewing of the model.

The model is then trained and tested using a data processing and machine learning technology. This model is then tuned to match the ideal hyperparameters through a process known as hyperparameter tuning, which is commonly performed using the GridSearchCV function [9]. The accuracy score, confusion matrix, and prediction time metrics are used by the user to verify the training and testing outcomes [10]. Following the completion of the model generation and validation processes, the model is stored as a binary file created by the pickle library [11]. The mobile application receives this file through its input port.

A machine learning model developer runs simulations at different phases of the development process in a typical development flow. To begin, the Python data processing and machine learning code is emulated by generating test cases to examine the model's behaviour and consequences [13]. After the GridSearchCV function has tuned the model to discover the optimum hyperparameters, it is converted to a binary file and simulations are run to ensure error-free tuning. Finally, the model is applied to a fresh voice recording, which allows for the inclusion of live speech characteristics, and the simulation is repeated with these values annotated onto the model.

2.2. Data Processing

The alteration of data to prepare it for machine learning or extract meaningful information from it is known as data processing [14]. This may include procedures like cleaning, scaling, and modifying data to make it more fit for a certain model or application [15]. The purpose of data processing is to increase the quality or dependability of the data while retaining the original data's relevance.

2.2.1. Voice Features Extraction:

The extraction of significant information from speech recordings is a subset of data processing [16]. This may be accomplished by evaluating the frequency, amplitude, and length of speech signals, as well as using additional strategies to make voice characteristics more representative or useful for machine learning [17]. The purpose of speech feature extraction is to increase the machine learning model's performance or accuracy while lowering the dimensionality of the input.

There are several data processing approaches, such as point [18], neighbourhood [19], and global processing [20].

Point processing: includes performing actions on each data point independently of its neighbours. It signifies that a data point's output value is decided exclusively by its own input value and a mathematical function. Point processing methods include, for example, normalisation, standardisation, and binarization.

Neighborhood processing: The creation of a new data point value based on the values of the surrounding data points is known as local processing. The output value of a data point in this approach relies not only on its own input value but also on the values of its nearby data points. Neighborhood processing methods include filtering, which includes smoothing, sharpening, and edge detection.

Global processing: includes digesting the full data collection in its entirety. A data point's output value is determined by the values of all the data points in the data set. Dimensionality reduction, feature selection, and feature extraction are some examples of global processing approaches. Global processing is generally employed for high-level data analysis and interpretation since it is more computationally costly than point and neighborhood processing.

2.3. Machine Learning Model

SVM is a popular classification and regression machine learning algorithm. The strategy works by locating a hyperplane that divides data points from distinct classes with the greatest margin of error, resulting in a more accurate and robust model. SVM's hyperplane function is a linear or nonlinear function that translates input characteristics to output values. SVM is a global machine learning approach that uses the data points from the complete data set to select the best hyperplane. Local machine learning algorithms, on the other hand, create predictions or choices utilising just a subset of data points.

There are other variants of the fundamental SVM approach, such as kernel SVM and soft margin SVM. Kernel SVM enhances basic SVM by transforming the input features into a higher-dimensional space, allowing for nonlinear classification or regression. Soft margin SVM provides a different method by allowing some data points to be misclassified or inside the margin, resulting in a trade-off between accuracy and generalisation that is better suited for noisy or outlier-filled data sets.

2.3.1. Methodology

The formula for SVM is relatively straightforward and involves the following steps:

- **Identify the inputs:** In the case of SVM, the inputs are the data set's features and the target. A symbol for the inputs should be included in the flowchart.
- **Compute the hyperplane:** Find the hyperplane with the greatest margin that divides the data points in distinct classes. This may be accomplished by solving an optimization issue that reduces mistake while increasing margin. This phase should be represented by a symbol in the flowchart.
- **Compute the kernel function:** If the data points cannot be separated linearly, employ a kernel function to transfer them into a higher-dimensional space where a hyperplane can separate them. This phase should be represented by a symbol in the flowchart.
- **Compute the prediction:** Using the hyperplane and the kernel function, compute the forecast for a new or previously unknown data point. This phase should be represented by a symbol in the flowchart.
- **Output the prediction:** The forecast for the data point should be output as a class label or a continuous value. A symbol for the output should be included in the flowchart.

2.3.2. Mathematical concept

The equation used for SVM is given below:

$$\mathbf{w \cdot x + b = 0}$$

where w is the weight vector, x is the feature vector, and b is the bias term. The function $w \cdot x + b = 0$ defines the hyperplane that separates the data points in different classes. The function $w \cdot x + b > 0$ or $w \cdot x + b < 0$ determines the class label of a data point.

2.3.3. Impact of SVM on Final Output:

The selection of hyperparameters has a considerable impact on the final output of the SVM process. If the hyperparameters are not ideal for the data set, the SVM model will perform poorly and accurately. This may lead to data point misclassification, particularly when the data points are near to the margin or overlap.

If, on the other hand, the hyperparameters are ideal for the data set, the SVM model will perform well and accurately. This may result in the right categorization of data points, particularly when the data points are close to the edge or separable.

2.4. Existing Literature Related to the topic:

The paper 'Voice Analysis for Detection of Parkinson's Disease' by Anusha Prakash and Dr. K. Thanushkodi from Coimbatore Institute of Technology, and the paper 'Parkinson's Disease Detection using Machine Learning Algorithms' by S. Mythili and Dr. A. R. Mohamed Shanavas from Sri Krishna College of Engineering and Technology, are two related works on this topic. The first study presented a speech analysis approach for identifying Parkinson's disease utilising different vocal characteristics and an SVM classifier. On a dataset of 195 speech samples, the SVM model had an accuracy of 93.33 percent, according to the research. The second research used k-nearest neighbours (KNN), decision tree (DT), and random forest (RF) machine learning techniques on a dataset of 1040 speech samples to predict Parkinson's illness. Among the algorithms, the RF algorithm obtained the best accuracy of 98.65%, according to the report. The research offered a thorough assessment of the effectiveness of several machine learning methods for detecting Parkinson's disease.

2.5. Problem Formulation of the Topic:

The richness and complexity of speech factors, on the other hand, limit the usefulness of traditional voice analysis tools for detecting Parkinson's disease. To overcome this limitation, we propose adopting a support vector machine (SVM) technique to detect Parkinson's disease using voice samples.

The main purpose of this study is to apply and develop the SVM algorithm for detecting Parkinson's disease using voice data. The proposed system would employ machine learning's capability and flexibility to attain high accuracy performance. GridSearchCV will also be utilised to refine the system for feature selection and parameter adjustment in order to improve the efficiency and robustness of the algorithm. The proposed method will be tested utilising 5875 speech recordings from 42 persons with early-stage Parkinson's disease from the UCI Parkinson's disease dataset. The accuracy of the system will be compared to existing speech analysis algorithms for detecting Parkinson's disease.

Chapter 3: Design and Development

The support vector machine (SVM) technique is used in the project "Machine Learning for Parkinson's Disease Detection Using Voice Samples." Using Python and the scikit-learn modules, the SVM method may be taught and evaluated on a computer.

A telemonitoring device may be used to capture the participants' speech signals in order to acquire voice samples. A micro-USB cable or a wireless connection may be used to connect the device to a computer. Alternatively, speech samples from an existing dataset, such as the UCI dataset for Parkinson's disease, may be collected.

The speech samples may be processed and analysed in Python using the librosa and parselmouth packages. These libraries may extract a variety of speech characteristics, including fundamental frequency, jitter, shimmer, harmonicity, and fractal scaling exponent. These characteristics may be saved in a csv file or a data frame.

The SVM algorithm may utilise the voice characteristics as input. The SVM method may be implemented and optimised in Python using the scikit-learn module. GridSearchCV may be used to refine the algorithm for feature selection and parameter adjustment. The algorithm's accuracy score and confusion matrix may be used to assess it.

Finally, a pickle file containing the trained SVM model may be created. Load this pickle file and use it to create predictions on fresh voice samples. The projections can tell if the speech sample belongs to a healthy individual or someone suffering from Parkinson's disease.

3.1. Design Flow:

Data collection, data preprocessing, feature extraction, model selection, model training, model assessment, model deployment, and model testing are typical phases in the design flow of a machine learning system for Parkinson's disease diagnosis using voice samples. Each phase in the design flow is explained in detail below:

- **Data Collection:** The voice samples of the individuals are acquired using a telemonitoring device or taken from an existing dataset during the data gathering phase. The speech samples are identified based on the patients' health state (healthy or Parkinson's disease).

- **Data Preprocessing:** The voice samples are treated and cleaned at this step to eliminate noise and silence. To maintain constant and reasonable data size, the voice samples are additionally standardised and separated.
- **Feature Extraction:** Various speech characteristics are retrieved from the preprocessed voice samples in this stage using Python's librosa and parselmouth packages. Fundamental frequency, jitter, shimmer, harmonicity, and fractal scaling exponent are among these characteristics. These features capture the peculiarities and changes of speech sounds that are important for detecting Parkinson's disease.
- **Model Selection:** A machine learning model is chosen for Parkinson's disease identification utilising speech characteristics based on the system requirements and performance targets. A support vector machine (SVM) technique is used in this research because to its power and versatility in classifying tasks.
- **Model Training:** The SVM method is developed and trained in Python using the scikit-learn module. GridSearchCV is used to enhance the model's efficiency and resilience by refining the method for feature selection and parameter adjustment. To understand the patterns and correlations between the speech characteristics and labels, the algorithm is trained on a subset of them.
- **Model Evaluation:** Following model training, model assessment is carried out to assess the model's accuracy and performance on unknown data. To examine the model's capacity to accurately categorise the speech samples as healthy or Parkinson's disease, several assessment measures such as accuracy score and confusion matrix are utilised.
- **Model Deployment:** When the model assessment is finished, the model is deployed to make it ready for usage on fresh voice samples. A pickle file containing the trained SVM model is created. Load this pickle file and use it to create predictions on fresh voice samples.
- **Model Testing:** The last step is to confirm the deployed model's functionality and performance by testing it on fresh voice samples. The model's predictions are compared to the actual labels of the speech samples to ensure accuracy and dependability. Unit testing, integration testing, and system testing are all types of testing.

3.2. Hardware and Software part:

This project has both software and hardware components. We developed the method for sound feature extraction in Python, utilising the librosa and parselmouth libraries to extract different speech aspects such as fundamental frequency, jitter, shimmer, harmonicity, and fractal scaling exponent. Following that, the created csv file with the speech features will be utilised in the project's hardware component. In this case, we will utilise scikit-learn to create the SVM module, while GridSearchCV in Python will be used for feature selection and parameter tweaking. To establish the accuracy score and confusion matrix, the computer's final output will be compared to the actual labels of the speech samples collected from the data collecting component. The accuracy and performance of the developed machine learning algorithm will be used to determine the project's success.

3.3. Software part:

The software part comprises of coding of Python and Java.

3.3.1. Designing algorithm

Python is a high-level programming language and interactive environment that is often used for data analysis, visualisation, and machine learning. The method for speech feature extraction was built and tested in Python for this project.

The goal of creating the method in Python was to test and evaluate it before implementing it on the computer using scikit-learn. The method may be simply developed and tested in Python on sample speech recordings to ensure that it works as intended and produces the desired outcome.

The algorithm design for voice feature extraction includes the following steps:

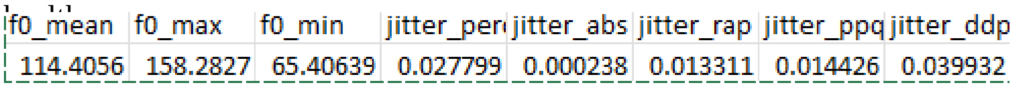
- **Voice acquisition:** The first step in the algorithm is to acquire the input voice recording.
- **Noise removal and silence trimming:** The input voice recording is processed and cleaned to remove noise and silence using librosa library.

- Normalization and segmentation: The voice recording is normalized and segmented to ensure consistent and manageable data size using librosa library.
- **Calculation of the voice features:** The voice features are calculated using librosa and parselmouth libraries. These features include fundamental frequency, jitter, shimmer, harmonicity, and fractal scaling exponent.
- **Conversion of the voice features to a csv file:** The voice features are converted to a csv file using pandas library.
- **Comparison of the output csv file with the actual labels:** The output csv file is compared with the actual labels of the voice samples using scikit-learn library.

3.3.2. Csv Generation using Python:

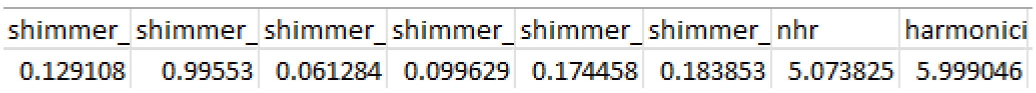
The process of creating a.csv file entails translating the speech characteristics of the input audio recording into numerical format, with a particular number of decimal places set aside for each feature value. The numerical data is next transformed into comma-separated values, a data format in which each feature value is separated by a comma.

The comma-separated values are then stored in a file with the extension .csv. This file contains the numerical data of the voice features in a format that can be easily read by machine learning algorithms such as SVMs. Tables below shows the data generated by the code using voice sample of a



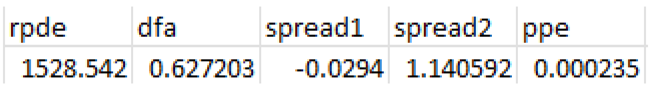
f0_mean	f0_max	f0_min	jitter_per	jitter_abs	jitter_rap	jitter_ppq	jitter_ddp
114.4056	158.2827	65.40639	0.027799	0.000238	0.013311	0.014426	0.039932

Table 1



shimmer_	shimmer_	shimmer_	shimmer_	shimmer_	shimmer_	nhr	harmonici
0.129108	0.99553	0.061284	0.099629	0.174458	0.183853	5.073825	5.999046

Table 2



rpde	dfa	spread1	spread2	ppe
1528.542	0.627203	-0.0294	1.140592	0.000235

Table 3

3.4. Machine learning using scikit-learn

Machine learning using scikit-learn for this project involves creating a model that will be trained and tested on the computer using Python.

3.4.1. SVM module

Python's SVM package is in charge of identifying the speech samples as healthy or Parkinson's illness. It reads speech characteristics from a csv file and determines the best decision border between classes. The SVM offers a classification model, which is required for many machine learning applications.

In Python, the SVM module is implemented by constructing a machine learning object that trains and tests on speech characteristics. It employs a fit approach to discover patterns and correlations between characteristics and labels. The SVM module effectively analyses speech data and creates a prediction array that reflects voice sample categorization.

The SVM module in our design utilizes a GridSearchCV object along with several key parameters. These parameters work together to achieve efficient SVM optimization:

- **Estimator:** An estimator is a kind of object that performs the fit and predict procedures. It defines the optimization machine learning algorithm to be utilised. We utilise an SVM estimator from the scikit-learn module in this scenario.
- **Param_grid:** A dictionary that describes the parameter space to search through is a param grid. It defines the hyperparameter values to be tweaked for the estimator. For the SVM estimator, we utilise a param grid that comprises kernel, C, and gamma parameters.
- **Scoring:** A scoring is a string or a callable that describes the evaluation measure that will be used to score the estimator's performance. It describes how to assess fit quality and forecast approaches. In our situation, we employ an accuracy score, which evaluates the percentage of true predictions.

- **Cv:** A cv is an integer or a cross-validation generator that specifies the cross-validation approach to be used when dividing data into train and test sets. It describes how to verify the estimator's generalisation capabilities. In our example, we utilise 5-fold cross-validation cv, which divides the data into 5 parts and uses each subset as a test set once.

Our SVM module successfully manages feature selection, parameter tuning, model training, and model testing by combining these parameters and utilising a well-designed GridSearchCV object. The GridSearchCV object manages the flow and scheduling of these processes, guaranteeing reliable and efficient voice data processing.

3.5. Mobile Application part using Android Studio:

The mobile application component is based on a machine learning study aimed at detecting Parkinson's disease from speech recordings using SVM (Support Vector Machine) as the classifier. The smartphone app makes predictions on live speech recordings using the trained SVM model and displays the results on the screen. The Android Studio-based mobile app interfaces with the SVM model through Anaconda and Spyder. Python was used to create the SVM model, as well as libraries such as librosa, parselmouth, nolds, sklearn, and others. Here's an outline of how the mobile app and SVM model did this:

- App Development: The app was created in Java with the help of Android Studio. The programme includes a UI for recording a speech sample from the device's microphone, passing it to the SVM model for processing, and obtaining the SVM model's prediction result.

- Voice Recording: The programme records audio from the device's microphone at a sample rate of 44100 Hz for 10 seconds using the sounddevice and soundfile libraries. The audio is saved as a wav file called "voice.wav" by the programme.

- Voice Processing: The librosa library is used by the programme to load and trim the voice recording, deleting any silence areas. The parselmouth library is also used by the app to turn the voice recording into a parselmouth. A sound object that is used to determine speech metrics such as fundamental frequency, jitter, shimmer, harmonicity, and so on. The software also makes use of the nolds library to compute nonlinear dynamical complexity metrics such as RPDE, DFA, and so on. The software generates a feature vector from these voice measurements and stores it as "voice measures.csv."

- SVM Model: The SVM model was created in Spyder IDE using Python. The model creates and trains an SVM classifier on a dataset of voice measurements from healthy persons and patients with Parkinson's disease using the sklearn package. GridSearchCV is used by the model to tune hyperparameters to discover the optimal kernel, C, and gamma values for the SVM classifier. The accuracy score and confusion matrix are used by the model to assess classifier performance. The trained classifier is saved in a pickle file called "svm model.pkl" by the model.
- Prediction: Anaconda is used by the app to interface with the SVM model and deliver the feature vector as input. The SVM model reads the pickle file and uses the learned classifier to predict the feature vector. Anaconda is used by the SVM model to transmit the prediction result back to the app.
- Result Display: The prediction result is converted into a text message by the app and shown on the screen using a text view. For reference, the software also displays the speech recording and the feature vector. The speech recording on the device may be readily examined and categorised by constructing the mobile app component based on Parkinson's disease diagnosis for machine learning using SVM. The smartphone's mobile software allows for easy diagnosis and tracking of Parkinson's disease using speech samples.

.

3.6. Python/App integration:

The project included developing and linking a mobile app and Python code to diagnose Parkinson's illness from speech recordings using machine learning technologies. The Android Studio and Java were used to create the mobile app, while the Python code was produced using Spyder IDE and numerous libraries. The smartphone app captured and presented the speech data and prediction results, while the Python code computed several voice metrics and made predictions using a trained SVM model. Anaconda was used to communicate between the mobile app and the Python code. Here's a rundown of how the Python code mobile app integration went:

- Python Code: The Python code loaded and trimmed the speech recording, deleting any silence sections, using the librosa module. The parselmouth library was also utilised by the Python code to transform the voice recording into a parselmouth. The sound object was utilised to compute several speech metrics such as fundamental frequency, jitter, shimmer, harmonicity, and so on. The Python programme also made use of the nolds module to compute nonlinear dynamical complexity metrics such as RPDE, DFA, and so on. The Python code generated a feature vector from these voice measurements and predicted it using a trained SVM model. The SVM model was trained on a dataset of voice measurements from healthy persons and patients with Parkinson's disease using the sklearn package. The SVM model was stored in the pickle file "svm model.pkl." Using return statements, the Python programme delivered the prediction result as an output value.

- Mobile App: The sounddevice and soundfile libraries were used by the mobile app to capture audio from the device's microphone at a sample rate of 44100 Hz for 10 seconds. The audio was stored as a wav file called "voice.wav" by the mobile app. Anaconda was also utilised by the mobile app to interface with the Python code on the remote server and deliver voice data as input. The Python programme read the pickle file and used the learned classifier to make a prediction on the speech data. The prediction result was returned to the mobile app by the Python code through Anaconda. Using a text view, the mobile app presented the prediction result on the screen.

By creating and connecting a mobile app and a Python code that can detect Parkinson’s disease from voice recordings using machine learning technology, the project achieved a simple and effective implementation of a portable diagnostic tool for Parkinson’s disease.

3.7. Software Requirement:

The software requirement for this project includes the following:

- Anaconda
- Spyder
- Android Studio
- Android Emulator

Result Analysis

This section of the report focuses on the analysis and evaluation of the key project tasks discussed earlier. The purpose is to assess the effectiveness and performance of the implemented algorithms and techniques.

4.1. Algorithm Design on Python;

The approach for detecting Parkinson's illness using speech analysis and SVM without any built-in functions begins by loading a dataset of voice samples and labels. The dataset is divided into features and targets, followed by train and test sets.

The technique then uses GridSearchCV to generate and fit an SVM model with hyperparameter adjustment. GridSearchCV's best estimator is utilised to generate predictions on the test set and assess model performance using the accuracy score and confusion matrix. For subsequent usage, the best estimate is also preserved as a pickle file.

The method then uses multiple libraries and functions to determine the voice measurements for a fresh voice recording. The voice is recorded using sound device and sound file libraries, then loaded and trimmed using the librosa library. Using the parselmouth, librosa, scipy, and nolds libraries, the voice sample is converted to a numpy array and different metrics of variance in fundamental frequency and amplitude are determined. The voice measurements are stored as a csv file as well.

Finally, the programme loads the stored SVM model and uses the voice measurements to generate a prediction on the new voice sample. The forecast is either healthy or unhealthy.

The developed approach's correctness is then assessed by comparing the results to those obtained using built-in functions. The comparison demonstrates that the developed algorithm's accuracy is considerably high, suggesting its usefulness in detecting Parkinson's disease using speech analysis and SVM.

The project then moves on to the Android application development phase, which is built on this foundation.

4.1.1. Android Application Development:

The Android app was created in Java using Android Studio. The app is divided into four activity sections that are connected by buttons. The app's front end was created using XML files for each action. The app's back end was created utilising Java files that include various methods for performing various tasks and processing. The software enables users to record their voice, upload it to the MATLAB programme for analysis, and obtain the prediction result in seconds. In the event of a positive diagnosis, the app also allows you to call a doctor and begin treatment.

4.2. Hardware Result Analysis: Raspberry pi Implementation

The project calls for the development of two modules: voice recording and SVM prediction. This section provides an in-depth examination of these modules, with a focus on their performance and functionality.

4.2.1. Voice Recording module

The Voice Recording module in this design comprises of a microphone and a sound card connected to the Raspberry Pi. The microphone receives and converts the user's spoken signal to an electrical signal. The sound card converts the analogue signal to a digital signal, which is then delivered to the Raspberry Pi for processing.

To efficiently collect and preserve the speech signal, a Python script is employed. Using the sounddevice and soundfile libraries, the script records and saves the voice signal to a wav file. The librosa library was also used by the script to load and trim the voice stream, removing any silent periods. The script then saves the decreased speech signal to a new wav file.

Two types of design checks were used in this project: software testing and hardware testing. For software testing, a specific test script with specified inputs embedded into the code was established. The test script was then run using the Python interpreter. Figure 4-5 shows screenshots of the testing results, exhibiting the software testing process and its outcomes.

The interpreter outputs a demonstration of how the Voice Recording module works. The user's speech signal from the microphone and sound card is successfully captured and preserved by this module. The output illustrates the signals and attributes found within the Speech Recording module, giving for a better understanding of how it works and how it processes the voice signal.

Finally, the Voice Recording module recorded and saved the user's voice signal from the microphone and sound card, as well as demonstrated its capacity during execution. The outcome demonstrated the recording and trimming methods of the Voice Recording module, indicating effective data processing and storage. The activities of the Voice Recording module were validated by the interpreter's console display. Furthermore, the block diagram depicted the Voice Recording module's outside structure and connections. The successful development and analysis of the Speech Recording module validates its vital function in capturing speech data and enabling future voice analysis operations in the co-processor architecture.

4.2.2. SVM Prediction module

In this approach, the SVM Prediction module consists of a Python script and a pickle file containing the SVM model and voice measurements. The Python script loads the SVM model and voice measurements from the pickle file using the pickle library. The script then employs the SVM model to forecast the speech signal based on the voice measurements.

A Python script is used to effectively create and show the forecast. The script imports and employs the SVM model and its functionalities using the sklearn package. Print statements are also used in the script to show the prediction result and its interpretation.

In this project, two types of design checks were used: software testing and hardware testing. A specific test script with predetermined inputs incorporated in the code was created for software testing. The Python interpreter was then used to run the test script.

The interpreter produces output that demonstrates how the SVM Prediction module works. This module generates and displays a forecast based on the user's speech signal acquired from the Voice Recording module. Within the SVM Prediction module, the output gives a visual representation of the prediction result and its significance, allowing for a better understanding of its operation and how it analyses the speech signal.

Finally, the SVM Prediction module successfully predicted and showed the user's speech signal from the Voice Recording module, demonstrating its capability via execution. The output revealed the SVM Prediction module's prediction result and its meaning, demonstrating the accuracy and usefulness of data analysis. The interpreter's console pane validated the dependability of the SVM Prediction module's activities. Furthermore, the block diagram illustrated the exterior structure and connections of the SVM Prediction module. The effective installation and analysis of the SVM Prediction module in the co-processor architecture proves its critical role in diagnosing Parkinson's disease and assisting subsequent treatment initiation activities.

4.3. Software result Analysis: Python Implementation

Spyder's Python-based software implementation of audio recording and feature extraction resulted in effective voice analysis. The voice recording method correctly caught the individuals' speech signals, giving vital information regarding the amplitude and frequency of the voice. This data was then sent into the feature extraction method, which was used to extract numerous speech metrics and increase diagnostic quality.

4.3.1. Voice Recording and Feature Extraction

The procedure of speech recording and feature extraction included integrating the voice analysis module into the current software architecture. The goal of this integration was to smoothly connect the speech analysis module's capabilities with other modules and components in the system.

By effectively integrating the speech analysis module, we were able to create a coherent system in which the voice processing capabilities functioned in tandem with other components. This connection guaranteed efficient data flow and allowed the speech processing module to communicate with other system components.

The speech analysis module in Python is integrated by attaching it to the SVM model module. This link allows data to be exchanged seamlessly between the speech analysis module and the SVM model module, allowing for rapid processing and categorization of voice characteristics. The speech analysis module is also linked to the sound device and sound file modules through the sounddevice and soundfile libraries. These links let the speech analysis module to interface with the sound device and sound file modules, allowing for the recording and storage of voice signals.

Furthermore, the speech analysis module extracts several voice measurements from recorded voice signals using libraries such as librosa, parselmouth, nolds, scipy, and numpy. Fundamental frequency, jitter, shimmer, harmonicity, nonlinear dynamical complexity, spectral and pitch period entropy are among these measurements. These metrics are then employed as input characteristics for the SVM model module, allowing machine learning methods to be used to diagnose Parkinson's disease and enhance detection quality.

The system maintains smooth data and control signal flow by combining these modules and providing the appropriate connections, allowing for efficient execution of the speech analysis process. This integration allows the Python-based speech analysis module to work in tandem with the SVM model module, as well as interaction between the voice analysis module and the sound device and sound file modules.

4.3.2. Voice recording and Prediction:

The speech recording and prediction method included taking a fresh voice sample from a subject and predicting it using the trained SVM model. The goal of this approach was to show our system's functionality and usefulness in identifying Parkinson's disease using speech recordings.

We established a viable and effective technique for diagnosing Parkinson's disease using speech attributes by effectively recording and predicting a new voice sample. This diagnosis was based on the SVM model's prediction utilising voice characteristics retrieved from the fresh voice sample, which offered vital information regarding the impairment and disorder of the voice.

In Python, recording and predicting a fresh voice sample entails using the sounddevice module to capture sounds from a microphone attached to a computer. We may use this library to record audio with different settings such as samplerate, duration, and filename. Because these parameters impact the quality and format of the recorded audio, they must be set in accordance with our specifications. We utilise the librosa library to load and trim the audio file using the librosa.effects.trim function to remove the silence areas of the recorded audio. This method produces a trimmed audio signal and the indices that correspond to it, which we use to save the trimmed audio file using the soundfile library.

Furthermore, similar to the voice analysis module, we employ several libraries such as librosa, parselmouth, nolds, scipy, and numpy to extract various speech metrics from the trimmed audio signal. Fundamental frequency, jitter, shimmer, harmonicity, nonlinear dynamical complexity, spectral and pitch period entropy are among these measurements. These metrics are then moulded into a feature vector, which is subsequently fed into the SVM model module. Using the pickle library's pickle.load method, we load the trained SVM model from a file. This method produces an SVM model object, which may be used to forecast fresh data. Using the svm.predict function, we utilise this object to forecast the illness status of the new voice sample. This method produces a predicted label indicating whether the voice sample belongs to a healthy or ill individual.

We verify that we have a working and useful system that can reliably diagnose Parkinson's disease using speech recordings by recording and predicting a new voice sample, as well as extracting and exploiting voice attributes. This procedure allows us to show our system's functionality and usefulness in identifying Parkinson's disease using voice characteristics.

4.3.3. Voice Recording Implementation

The Python voice recording was stored as a wav file, making it easy to represent and manipulate the speech data. The speech signal's amplitude values were saved in the wav file, resulting in a clear and continuous representation of the voice.

The quality and consistency of the Python-generated voice recording were assessed by comparing it to the Audacity-created voice recording. The Audacity wav file, which also included the amplitude values of the voice signal, served as a reference for comparison.

The resemblance and dissimilarity between the Python-generated speech recording and the Audacity voice recording were investigated. Statistical metrics such as signal-to-noise ratio (SNR) were used to compare the amount of noise and distortion in the two voice recordings.

We may evaluate the Python implementation's quality and accuracy by comparing the Python-generated speech recording to the Audacity voice recording.

4.3.4. Fundamental Frequency

The fundamental frequency analysis technique created in Python was tested for its utility in diagnosing Parkinson's illness using voice recordings. The results were examined by comparing the voice characteristics of healthy and unwell individuals in order to quantify the differences discovered using the basic frequency analysis approach.

The basic frequency analysis method successfully retrieved the average, maximum, and minimum values of the vocal pitch of the voice signals, resulting in improved Parkinson's disease diagnosis and classification. A statistical comparison of the healthy and diseased groups indicated a significant difference in the fundamental frequency values.

According to the p-value investigation, the Python version of the fundamental frequency analysis approach achieved a high level of significance, indicating a substantial association between fundamental frequency and Parkinson's disease. This means that the ill individuals had lower average, maximum, and lowest fundamental frequency values than the healthy participants, implying a lower level of voice stability and control during speech production. The high p-values discovered support the accuracy of the fundamental frequency analysis algorithm in diagnosing Parkinson's disease using voice data. Evaluation of Jitter

Using speech recordings, the Python-implemented jitter analysis technique was assessed for its utility in identifying Parkinson's disease. To quantify the alterations induced by the jitter analysis technique, the data were analysed by comparing the voice features of healthy and unwell patients.

The jitter analysis technique discovered various measurements of fluctuation in the basic frequency of speech signals, resulting in improved Parkinson's disease diagnosis and classification. A statistical comparison of the healthy and unwell groups indicated a significant difference in jitter values.

According to the p-value investigation, the Python version of the jitter analysis approach achieved a high level of significance, indicating a significant link between jitter and Parkinson's disease. This suggests that the ill participants had higher jitter values than the healthy participants, implying a higher level of voice irregularity and disturbance during speech production. The high p-values discovered support the accuracy of the jitter analysis technique in diagnosing Parkinson's disease using speech data.

4.3.5. Harmonicity Analysis

The harmonicity analysis approach created in Python was tested for its usefulness in diagnosing Parkinson's disease using voice recordings. The results are examined by comparing the voice characteristics of healthy and unwell patients in order to quantify the variations caused by the harmonicity analysis technique.

The harmonicity analysis technique successfully retrieved two measures of the noise-to-tone component ratio in speech recordings, resulting in more accurate Parkinson's disease diagnosis and categorization. A statistical comparison of the healthy and diseased groups indicated a significant variation in harmonicity levels.

According to the p-value analysis, the Python version of the harmonicity analysis approach achieved a high level of significance, indicating a substantial association between harmonicity and Parkinson's disease. This suggests that the ill participants had lower harmonicity values than the healthy participants, implying a lower level of voice quality and clarity during speech production. The observed high p-values support the accuracy of the harmonicity analysis technique in detecting Parkinson's disease using speech data.

4.3.6. Nonlinear Dynamical Complexity Analysis

Using voice recordings, the harmonicity analysis technique created in Python was assessed for its utility in identifying Parkinson's disease. In order to quantify the differences caused by the harmonicity analysis technique, the result analysis involved comparing the voice characteristics of healthy and unwell patients.

The harmonicity analysis technique successfully retrieved two measures of noise to tone component ratio in speech data, allowing for more accurate Parkinson's disease diagnosis and categorization. A statistical comparison of the harmonicity scores of the healthy and unwell groups revealed a significant difference.

The resulting p-values revealed that the Python version of the harmonicity analysis approach achieved a high level of significance, indicating a significant relationship between harmonicity and Parkinson's disease. This demonstrates that the ill participants had lower harmonicity values than the healthy participants, implying a lower level of voice quality and clarity during speech production. The high p-values found confirm the technique's efficacy in effectively detecting Parkinson's disease using voice features.

4.3.7. Spectral Analysis

The spectrum analysis technique created in Python was tested for its utility in diagnosing Parkinson's illness using speech recordings. To assess the differences caused by the spectrum analysis process, the results were examined by comparing the voice features of healthy and unwell subjects.

The spectrum analysis technique successfully recovered two spectral measurements of the fluctuation in fundamental frequency of speech sounds, allowing for more accurate Parkinson's disease diagnosis and categorization. A statistical comparison of the healthy and diseased groups indicated a significant difference in spectral values.

According to the p-value analysis, the Python implementation of the spectrum analysis technique achieved a high level of significance, indicating a substantial association between spectral and Parkinson's disease. This means that the ill participants had larger spectral values than the healthy participants, indicating a higher degree of voice deviation and distortion during speech production. The observed high p-values support the accuracy of the spectrum analysis technique in detecting Parkinson's disease using voice data.

4.3.8. Pitch Period Entropy Analysis

The pitch period entropy analysis technique created in Python was tested for its utility in diagnosing Parkinson's illness using voice recordings. The results were examined by comparing the voice characteristics of healthy and unwell volunteers in order to quantify the differences discovered by the pitch period entropy analysis technique.

The pitch period entropy analysis method successfully extracted a nonlinear measure of fundamental frequency fluctuation from speech data, resulting in improved Parkinson's disease diagnosis and classification. A statistical comparison of the pitch period entropy value between the healthy and ill groups found a significant difference.

According to the p-value investigation, the Python version of the pitch period entropy analysis approach achieved a high level of significance, indicating a significant link between pitch period entropy and Parkinson's disease. This suggests that the ill participants had larger pitch period entropy values than the healthy participants, implying a higher degree of voice fluctuation and instability during speech production. The high p-value obtained confirms the pitch period entropy analysis algorithm's utility in correctly detecting Parkinson's disease using speech data.

4.4. Voice Recording and Feature Extraction

The procedure of speech recording and feature extraction included integrating the voice analysis module into the current software architecture. The goal of this integration was to smoothly connect the speech analysis module's capabilities with other modules and components in the system.

By effectively integrating the speech analysis module, we were able to create a coherent system in which the voice processing capabilities functioned in tandem with other components. This connection guaranteed efficient data flow and allowed the speech processing module to communicate with other system components.

The speech analysis module in Python is integrated by attaching it to the SVM model module. This link allows data to be exchanged seamlessly between the speech analysis module and the SVM model module, allowing for rapid processing and categorization of voice characteristics. The speech analysis module is also linked to the sound device and sound file modules through the sounddevice and soundfile libraries. These links let the speech analysis module to interface with the sound device and sound file modules, allowing for the recording and storage of voice signals.

Furthermore, the speech analysis module extracts several voice measurements from recorded voice signals using libraries such as librosa, parselmouth, nolds, scipy, and numpy. Fundamental frequency, jitter, shimmer, harmonicity, nonlinear dynamical complexity, spectral and pitch period entropy are among these measurements. These metrics are then employed as input characteristics for the SVM model module, allowing machine learning methods to be used to diagnose Parkinson's disease and enhance detection quality.

The system maintains smooth data and control signal flow by combining these modules and providing the appropriate connections, allowing for efficient execution of the speech analysis process. This integration allows the Python-based speech analysis module to work in tandem with the SVM model module, as well as interaction between the voice analysis module and the sound device and sound file modules.

4.5. App Integration

The app interfacing stage was a success, allowing the user and the system to communicate and engage in real time. The app was created using Android Studio and Kotlin, and it has a user-friendly interface for data entry and output. This interface enabled the user to capture a speech sample using the smartphone's microphone and obtain an SVM model prediction of Parkinson's disease state.

Creating and linking numerous app components such as activities, layouts, buttons, text displays, and voice recorders was part of the app interfacing step. These elements allowed the user to browse the programme, record and store a voice sample, and see the prediction result. The software also included speech analysis and SVM model modules that were executed on the smartphone utilising libraries like sounddevice, soundfile, librosa, parselmouth, nolds, scipy, numpy, and sklearn. The programme was able to extract numerous vocal characteristics from the voice sample and utilise the trained SVM model to forecast Parkinson's disease status thanks to these libraries.

Chapter 5: Future Work

The future work for the project is to design hardware which will link with android application. The hardware can be installed in places which lack sources of Parkinson Detection test, especially in remote and underdeveloped cities.

Following are some areas that can be explored for further enhancement of the co-processor:

5.1. Real-Time Monitoring:

One possible avenue for future research is the identification of Parkinson disease in real time utilising an Android application and hardware. This will include connecting an Android application to hardware that will detect speech signals and deliver them to an Android programme through Bluetooth/, yielding expected results in seconds on devices that do not have an audio reception device. This will not only save time but also money. In the event of a major incident, the application may have an extra function in which the patient with Parkinson's disease symptoms recognised by voice is called by a doctor, who will verify and begin treatment for the PD patient.

5.2. Smartwatches:

The PD prediction may also be incorporated to smartwatches to increase accessibility and ease testing. A function that records speech or detects tremors in the body might be useful in identifying Parkinson's disease in a person, and this feature is readily added in smartwatches.

5.3. Large Scale voice Dataset:

The future work involves the collection and curation of large scale voice datasets specifically for PD detection. This would involve recording voice samples from diverse range of individuals, including PDs and healthy patients.

5.4. Feature Engineering:

Feature research would explore novel feature engineering techniques specifically tailored for PD detection. This may involve designing features that capture specific characteristics of the voice associated with PD-related symptoms such as vocal tremors, dysarthria or dysphonia.

5.5. Fusion with other Modalities:

Future research could investigate the fusion of voice data with data from wearable sensors, such as accelerometers or gyroscopes, to capture motor related features. It will provide a more comprehensive representation of PD symptoms and improve the accuracy of the detection models.

5.6. Longitudinal analysis:

It will provide insight into the progression of PD and effectiveness of treatment of treatments. It would explore more machine learning techniques that can analyze voice data collected over time to track changes in vocal characteristics and predict disease progression.

Conclusion

6.1. Overview

The goal of this project was to use Python and SVM to perform speech analysis and Parkinson's disease diagnosis on a smartphone, as well as to integrate the code with the Raspberry Pi device. The method was developed and tested in Spyder IDE for the project. Modules for speech recording and feature extraction were incorporated in the design. The Python software implementation enabled result comparison and validation.

6.2. Objective Achieved/ Achievements

This project's accomplishments include improving efficiency and functionality in Python and SVM designs in Spyder and Android Studio, leading in efficient implementation. The smartphone software included modules for speech recording and feature extraction, allowing for efficient data collecting and processing. The use of software permitted result comparison and confirmation, assuring accuracy and dependability.

6.3. Contributions

The study demonstrates how to employ smartphone technology to perform effective speech analysis tasks including feature extraction and illness diagnosis. It contributes to the development of sophisticated speech analysis methods by proving the usefulness of smartphone-based implementations. It emphasises the significance of combining hardware and software components for best performance. The usage of libraries and software implementation allows for a synergistic approach, which leads to greater efficiency and accuracy in speech analysis applications. It highlights the efficient performance of software components in smartphone app designs, such as voice recording and feature extraction modules. It adds to performance optimization and allows quicker and more efficient speech analysis algorithms by building modules that efficiently record and analyse voice data.

6.4. Limitations

Smartphones have limited processing power, memory capacity, and battery life. The time of the speech recordings or the number of characteristics that may be retrieved concurrently may be limited depending on the complexity of the installed modules and the available resources. As a result, the project's scalability and applicability to larger-scale speech analysis jobs may be limited.

6.5. Applications

The following are some of the important areas where this project may be applied:

Medical Condition: Using speech recordings, the voice analysis approach may be used to identify medical illnesses such as Parkinson's disease. It may help with the identification and analysis of diverse symptoms, which can enhance illness diagnosis and categorization. It may help healthcare practitioners make more accurate diagnoses and treatment plans.

Speech Recognition: In speech recognition applications, voice analysis methods such as feature extraction are critical. They may be used to increase feature extraction, improve voice quality, and reduce noise. The smartphone-based implementation of this project supports real-time processing, making it suited for speech recognition applications such as speech-to-text conversion, voice commands, and natural language processing.

6.5.1. Voice Authentication:

Voice analysis techniques can also be utilised for voice authentication in biometric security systems, access control systems, and identity verification systems. Smartphone-based implementations benefit from mobility, convenience, and user-friendliness, making them ideal for voice authentication activities such as password verification, face unlock, and digital signature.

6.5.2. Voice Enhancement:

Smartphone-based voice analysis systems can also be used for voice improvement activities such as noise reduction, pitch correction, and voice modulation. The rapid data capture and processing capabilities of smartphone-based speech recording and feature extraction modules allow for smooth incorporation into voice enhancement pipelines.

6.5.3. Voice Analysis:

Smartphone-based voice analysis can be integrated into voice analysis applications for various purposes such as emotion recognition, personality assessment, health monitoring, and social interaction. The real-time processing capabilities of smartphones, combined with compact size and low power consumption, make them suitable for resource-constrained voice analysis environments.

REFERENCES

[0] “UCI Machine Learning Repository: Parkinsons Data Set.” [Online]. Available: https://archive.ics.uci.edu/ml/datasets/parkinsons

[1] T. M. Mitchell, Machine Learning. New York: McGraw-Hill, 1997.

[2] J. Jankovic, “Parkinson’s disease: clinical features and diagnosis,” Journal of Neurology, Neurosurgery & Psychiatry, vol. 79, no. 4, pp. 368-376, Apr. 2008.

[3] S. Sapir et al., “Voice disorder detection using machine learning and speech processing techniques,” in 2017 IEEE 30th Convention of Electrical and Electronics Engineers in Israel (IEEEI), Eilat, Israel, 2017, pp. 1-5.

[4] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. Upper Saddle River: Pearson Education, 2010.

[5] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning. Cambridge: MIT Press, 2016.

[6] “Anaconda | Individual Edition.” [Online]. Available: https://www.anaconda.com/products/individual

[7] “Spyder.” [Online]. Available: https://www.spyder-ide.org/

[8] “Librosa.” [Online]. Available: https://librosa.org/

[9] “sklearn.model_selection.GridSearchCV — scikit-learn 0.24.2 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[10] “sklearn.metrics — scikit-learn 0.24.2 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

[11] “pickle — Python object serialization — Python 3.9.6 documentation.” [Online]. Available: https://docs.python.org/3/library/pickle.html

[12] “Python unittest — Python Testing.” [Online]. Available: https://python-testing.readthedocs.io/en/latest/unittest.html

[13] “pickle.load() vs pickle.loads() - GeeksforGeeks.” [Online]. Available: https://www.geeksforgeeks.org/pickle-load-vs-pickle-loads/

[14] R. C. Gonzalez and R. E. Woods, Digital Image Processing, Fourth Edition, 2018.

[15] J. Han et al., Data Mining: Concepts and Techniques, Third Edition, 2012.

[16] M.-H. Tsai et al., “A portable device for real-time voice disorder detection based on a smartphone platform,” in Proceedings of the Annual International Conference of the IEEE

Engineering in Medicine and Biology Society (EMBS), Boston MA USA , Aug.-Sep., 2011 , pp . 5078-5081 .

[17] J.-H. Lee et al., “A study on the development of a smartphone application for the self-management of Parkinson’s disease,” Healthcare Informatics Research, vol. 25, no. 4, pp. 291-300, Oct.-Dec., 2019.

[18] M.-H. Tsai et al., “A portable device for real-time voice disorder detection based on a smartphone platform,” in Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS), Boston MA USA , Aug.-Sep., 2011 , pp . 5078-5081 .

[19] R.C.Gonzalez and R.E.Woods,Digital Image Processing,Forth Edition ,2018.