A Machine Learning Approach for Parkinson's Disease Prediction with User Interface

This project is submitted to the Department of Computer Science and Engineering, Dhaka International University, in partial fulfillment to the requirements of Bachelor of Science (B. Sc.) in Computer Science and Engineering (CSE).

Submitted by

NAME	REG. NO	ROLL NO
Fatema Tuj Johora	CS-D-58-19-111748	11
Md. Rafiul Islam	CS-D-58-19-111891	24
Shahriar Alom Bhuiyan	CS-D-58-19-111901	26
Md. Sabbir Bahadur	CS-D-58-19-111915	28

Project Work, CSE-425

Batch: 58th (1st Shift), Session: 2019-2020

Supervised by

Md. Shariful Islam

Lecturer

DEPARTMENT OF COMPUTER SCIENCE and ENGINEERING
FACULTY OF SCIENCE AND ENGINEERING
DHAKA INTERNATIONAL UNIVERSITY
DHAKA, BANGLADESH

FEBRUARY-2024

SUPERVISOR'S STATEMENT

This is to certify that the project paper entitled as **"A Machine Learning Approach for Parkinson's Disease Prediction with User Interface"** submitted by Fatema Tuj Johora, Roll: 11, Md. Rafiul Islam, Roll: 24, Shahriar Alom Bhuiyan, Roll: 26, Md. Sabbir Bahadur, Roll: 28; has been carried out under my supervision. This project has been prepared in partial fulfillment of the requirement for the Degree of B.Sc. in Computer Science and Engineering, Department of Computer Science and Engineering, Dhaka International University, Dhaka, Bangladesh.

Supervisor's Signature

Date: ……

.....

Md. Shariful Islam
Lecturer
Dept. of Computer Science and Engineering,
Dhaka International University

APPROVAL

The project report **"A Machine Learning Approach for Parkinson's Disease Prediction with User Interface"** submitted by Fatema Tuj Johora, Md. Rafiul Islam, Shahriar Alom Bhuiyan, Md. Sabbir Bahadur to the Department of Computer Science and Engineering, Dhaka International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents.

Board of Honorable Examiners

1. Chairman

……
Prof. Dr. A. T. M. Mahbubur Rahman
Dean (acting)
Faculty of Science and Engineering and
Chairman
Dept. of Computer Science and Engineering
Dhaka International University

2. Internal Member

.....
Prof. Md. Abdul Based
Additional Director, Research and Publication Cell
Dept. of Computer Science and Engineering,
Dhaka International University

3. Internal Member

.....
Prof. Mst. Jahanara Akhtar
Dept. of Computer Science and Engineering,
Dhaka International University

4. Supervisor and Member

.....
Md. Shariful Islam
lecturer
Dept. of Computer Science and Engineering,
Dhaka International University

5. External Member

.....
Dr. Md. Manowarul Islam
Associate Professor,
Dept. of Computer Science and Engineering,
Dhaka International University

DECLARATION

We hereby declare that; this project has been carried out by us and it has been submitted for the award of the B.Sc. degree. We also certify that this project was prepared by us for the purpose of fulfillment the requirements for the Bachelor of Science (B.Sc.) in Computer Science and Engineering.

Authors Signature

.....

Fatema Tuj Johora
B.Sc. in CSE, Roll No: 11
Reg. No: CS-D-58-19-111748
Batch: 58th (1st shift), Session: 2019-2020
Dhaka International University

.....

Md. Rafiul Islam
B.Sc. in CSE, Roll No: 24
Reg. No: CS-D-58-19-111891
Batch: 58th (1st shift), Session: 2019-2020
Dhaka International University

.....

Shahriar Alom Bhuiyan
B.Sc. in CSE, Roll No: 26
Reg. No: CS-D-58-19-111901
Batch: 58th (1st shift), Session: 2019-2020
Dhaka International University

.....

Md. Sabbir Bahadur
B.Sc. in CSE, Roll No: 28
Reg. No: CS-D-58-19-111915
Batch: 58th (1st shift), Session: 2019-2020
Dhaka International University

Supervisor's Signature

Date: ……

.....

Md. Shariful Islam
Lecturer
Dept. of Computer Science and Engineering,
Dhaka International University

ABSTRACT

This study introduces a new method for analyzing Parkinson's disease (PD) using machine learning and a user-friendly interface. By combining a broad dataset with clinical, demographic, and potentially genetic information, our model achieves strong accuracy in detecting and assessing PD. The interface makes complex algorithms more accessible to healthcare professionals, enabling easy input of patient data and intuitive interpretation of results. Key features like tremor severity and gait abnormalities enhance diagnostic accuracy. The system's early detection potential supports proactive intervention and personalized treatment planning, while continuous monitoring tracks symptom changes over time. Integration with wearable technology allows real-time data input and detailed assessments. Collaboration with healthcare professionals ensures alignment with clinical standards, and global data sharing enhances generalizability. Ethical considerations and regulatory standards are prioritized for responsible deployment. This integrated approach shows great promise in advancing PD diagnosis and management, improving diagnostic accuracy, and empowering informed decision-making in patient care.

ACKNOWLEDGEMENTS

We would like to pay our gratitude to the Almighty Allah who created us with not only the ability to design and program this system but also the power of practice.

We would also like to express our sincere thanks to our respected supervisor **Md. Shariful Islam**, Lecturer, Department of CSE, Dhaka International University for his continuous encouragement, motivation and professional guidance during the work of this project.

We would like to thank all the faculty members for their valuable time spent on requirements analysis and evaluation of the project work.

We would like to express our sincere and cordial gratitude to the people those who have supported us directly, provided mental encouragement. We are also thankful to our family and friends who have contributed directly or indirectly the development word and its associated activities.

We warmly thank **Prof. Dr. A. T. M. Mahbubur Rahman**, Dean (Acting), Faculty of Science and Engineering and Chairman, Department of Computer Science and Engineering and **Prof. Md. Abdul Based**, Additional Director, Research and Publication Cell, Dept. of Computer Science and Engineering, Dhaka International University for their valuable advice and moral support. Their extensive discussion around work and interesting exploration in operations has been very helpful for this study.

Finally, we would like to dedicate this project to our teachers for their love, encouragement and professional guidance throughout the project.

Dedicated to

Our

Family & Teachers

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

1.1 OVERVIEW

The application of machine learning in the analysis of Parkinson's disease, complemented by a user interface, represents a cutting-edge approach with considerable potential in medical diagnostics. Drawing on extensive datasets, machine learning algorithms are trained to discern patterns and relationships within patient data, facilitating accurate and early detection of Parkinson's disease. The user interface plays a pivotal role in translating these complex algorithms into a practical tool for healthcare professionals. Its design emphasizes user-friendliness, allowing seamless input of patient information and providing interpretable results.

In this comprehensive analysis, key features and parameters such as tremor severity, gait abnormalities, and other relevant clinical markers are integrated into the model. The user interface, designed with an emphasis on clarity and ease of use, enables healthcare practitioners to navigate the diagnostic process efficiently. Visualizations and feedback mechanisms within the interface contribute to the interpretability of the model's predictions, fostering trust and facilitating informed decision-making.

This innovative methodology extends its impact beyond mere diagnosis, offering a platform for continuous monitoring of patients. By tracking changes over time, the system contributes to a dynamic understanding of disease progression, aiding in treatment planning and management. Additionally, the incorporation of wearable technology further enhances the model's capabilities, allowing for real-time data input and more nuanced assessments.

For a deeper understanding and broader applicability, collaboration with healthcare professionals is crucial. Their expertise ensures that the model aligns with clinical standards and addresses the nuanced aspects of Parkinson's disease diagnosis and management. Moreover, global collaboration in data sharing and model refinement is essential for ensuring the generalizability and robustness of the system across diverse populations.

As this field continues to evolve, considerations for ethical and regulatory standards are paramount. Striking a balance between innovation and adherence to established guidelines ensures the responsible development and deployment of machine learning applications in healthcare. The reference to the specific study or studies that underpin this approach would provide the necessary academic credibility and context.

1.2 PROBLEM DEFINATION

The problem addressed by the Parkinson's disease analysis using machine learning with a user interface lies in the current challenges associated with timely and accurate diagnosis of Parkinson's disease. Parkinson's disease is a neurodegenerative disorder characterized by progressive motor symptoms, and early detection is crucial for effective intervention and improved patient outcomes. Traditional diagnostic methods may have limitations in terms of speed, accuracy, and accessibility. The integration of machine learning aims to overcome these challenges by leveraging advanced algorithms that can analyze complex patterns within diverse patient data, leading to more precise and early identification of Parkinson's disease.

The user interface serves as a critical component in translating the machine learning model into a practical tool for healthcare professionals. The problem also encompasses the need for a seamless and intuitive platform that allows healthcare practitioners to input patient data efficiently and interpret the model's predictions with ease. The overarching goal is to enhance the diagnostic process, promote early intervention, and contribute to a more comprehensive understanding of Parkinson's disease dynamics for improved patient care.
In summary, the problem being addressed involves the limitations of current diagnostic approaches for Parkinson's disease and the development of a machine learning-based solution with a user-friendly interface to enable early and accurate detection, ultimately enhancing the management of the disease.

1.3 SEVERAL INDICATIONS

Parkinson's disease is a complex neurodegenerative disorder that manifests through various indications, affecting both motor and non-motor functions [23]. Several key indications of Parkinson's disease include:

Tremors: One of the hallmark symptoms of Parkinson's disease is resting tremors, typically observed in the hands, fingers, or other parts of the body while at rest.

Bradykinesia: Bradykinesia refers to slowness of movement, making simple tasks more time-consuming. It is a characteristic motor symptom associated with Parkinson's disease.

Muscle Rigidity: Stiffness or rigidity in the muscles, often leading to reduced range of motion and flexibility, is a common indication of Parkinson's disease.

Postural Instability: Impaired balance and postural instability are indicative of Parkinson's disease and can increase the risk of falls, particularly in later stages of the condition.

Gait Disturbances: Changes in walking pattern, such as shuffling steps, reduced arm swing, and difficulty initiating or stopping movement, are common indicators of Parkinson's disease.

Micrographia: Micrographia refers to the progressive reduction in handwriting size and legibility, which can occur in individuals with Parkinson's disease.

Facial Expression Changes: Reduced facial expressivity, often referred to as a "masked face," is another non-motor symptom associated with Parkinson's disease.

Speech Changes: Parkinson's disease can cause changes in speech, including a soft or monotone voice, slurring, or hesitation during speech.

Freezing of Gait: Freezing episodes, where individuals temporarily feel as though their feet are glued to the ground, can occur, especially when initiating or turning during walking.

Sleep Disturbances: Sleep-related issues are common in Parkinson's disease and may include insomnia, restless legs, and frequent waking during the night.

Non-Motor Symptoms: Parkinson's disease is also associated with various non-motor symptoms, including cognitive changes, mood disorders (such as depression and anxiety), autonomic dysfunction (such as orthostatic hypotension and constipation), and sensory abnormalities.

Loss of Smell (Hyposmia): A reduced sense of smell, known as hyposmia, is often an early non-motor symptom observed in Parkinson's disease.

1.4 SEVERAL RISK FACTORS

Several risk factors have been identified that may increase the likelihood of developing Parkinson's disease [23]. It's important to note that the presence of these risk factors does not guarantee the development of the disease, and many individuals with Parkinson's disease do not have any known risk factors. The risk factors for Parkinson's disease include:

Age: The risk of Parkinson's disease increases with age, and the majority of cases are diagnosed in individuals over the age of 60.

Family History: Individuals with a family history of Parkinson's disease may have a higher risk. While most cases are sporadic, a small percentage of individuals have a genetic predisposition.

Genetics: Certain genetic mutations and variations have been associated with an increased risk of Parkinson's disease. However, these genetic factors are not the sole determinant, and many individuals with these mutations do not develop the condition.

Gender: Men have a slightly higher risk of developing Parkinson's disease than women.

Environmental Factors: Exposure to certain environmental factors, such as pesticides and herbicides, has been studied as potential risk factors for Parkinson's disease. However, the link between specific environmental exposures and the development of the disease is complex and not fully understood.

Head Trauma: Some studies suggest that a history of head injuries, particularly repeated head trauma, may be associated with an increased risk of Parkinson's disease.

Occupational Exposures: Certain occupational exposures, such as working in agriculture or around industrial chemicals, have been investigated as potential risk factors.

Caffeine Consumption: Some research suggests that higher caffeine intake may be associated with a lower risk of Parkinson's disease. However, the relationship is complex and not fully established.

Lewy Body Dementia: Individuals with Lewy body dementia, a progressive neurological disorder, may have an increased risk of developing Parkinson's disease or related symptoms.

Reduced Estrogen Levels: Some studies suggest that a history of lower estrogen exposure in women, such as early menopause, may be associated with an increased risk of Parkinson's disease.

1.5 OBJECTIVES

The primary objective in developing a system for Parkinson's disease analysis using machine learning with a user interface is to enhance the early detection, continuous monitoring, and personalized treatment of individuals affected by Parkinson's disease. The system aims to address several key objectives:

Early Detection: Implement machine learning algorithms to accurately identify and diagnose Parkinson's disease in its early stages when symptoms may be subtle. Early detection facilitates timely intervention and improves patient outcomes.

Continuous Monitoring: Utilize real-time sensor data integrated into the system to monitor the progression of Parkinson's symptoms over time. Continuous monitoring allows for a dynamic understanding of the disease, enabling healthcare professionals to adjust Controlling pain, gaseousness, and tiredness are all parts of managing symptoms.

Personalized Treatment: Develop machine learning models that analyze patient-specific data to tailor treatment plans. Personalized interventions can improve the effectiveness of therapies, minimize side effects, and enhance the overall quality of life for individuals with Parkinson's disease.

User-Friendly Interface: Create a user interface that is intuitive, accessible, and user-friendly. The interface should enable healthcare professionals and potentially patients to interact seamlessly with the machine learning models, input relevant data, and interpret the results.

Integration with Clinical Workflows: Collaborate with healthcare professionals to integrate the system into existing clinical workflows. The system should complement the work of healthcare providers, providing valuable insights and supporting decision-making processes.

Regulatory Compliance: Design and implement the system in compliance with healthcare data regulations and standards. Protecting patient privacy and ensuring the security of sensitive health information are paramount considerations.

Feedback Mechanism: Establish mechanisms for user feedback to continuously improve the performance of the machine learning models. Feedback from healthcare professionals and, if applicable, patients, can contribute to refining the system over time.

Education and Awareness: Incorporate educational components within the user interface to enhance awareness and understanding of Parkinson's disease among healthcare professionals and, potentially, patients. This can contribute to more informed decision-making.

Research and Development: Encourage ongoing research and development to enhance the capabilities of the system. Staying abreast of advancements in both machine learning and Parkinson's disease research ensures that the system remains at the forefront of innovation.

Chapter 2 LITERATURE REVIEW

2.1 RELATED WORKS

Shamli and Sathiyabhama [1] suggested multi-classifier system, i.e., built on Big Data analytics to enhance efficient response times and predicted performance for economically sound activities. Big Data, its attributes, and its three categories of analytics—descriptive, predictive, and prescriptive—as they relate to the healthcare sector were presented by the author. To regulate muscular contraction, brain cells produce the neurotransmitter dopamine, which is used to communicate with other brain cells. Parkinson's disease (PD) is brought on by the death of dopaminergic brain cells. The UCI machine learning library provides the voice dataset of PD for study. A variety of classifiers' outcomes and accuracy can be obtained by applying several predictive models to disease datasets. Outperforming other machine learning algorithms are C4.5, SVM, and ANN. The best results are selected after comparing the output of these classifiers.

Azad et al. [2] explored a predictive model for PD that is based on decision tree algorithm. They presented Parkinson's disease (PD), the second most prevalent neurodegenerative illness, along with its symptoms, potential side effects, and risk factors. Decision trees, attribute selection metrics, ID3, and decision stumps are a few of the data mining applications used for categorization. Their dataset, which consists of 197 cases, was assembled using 31 individuals' data and is derived from the UCI repository. Two parameters are employed for performance analysis: classification error and accuracy. The 10-fold cross-validation method is used for validation, providing an objective result. They observed that decision tree algorithm performs better and delivers the best accuracy and lowest classification mistake than other algorithms in their experimental results.

Sriram et al. [3] suggested a technique for utilizing its voice dataset to diagnose PD. Thirteen of the individuals whose voices make up this audio dataset have Parkinson's disease (PD). 26 attributes and 5875 instances make up this dataset. The experiment employs Weka V3.4.10 and Orange V2.0b software for statistical analysis, classification, assessment, visualization, and unsupervised approaches. With the Random Forest algorithm, they obtained the highest accuracy of 90.2%.

Indira et al. [4] have made use of human biological voice as the primary feature. The authors have created a program that uses patient voice analysis to automatically determine whether a person has Parkinson's disease (PD). Using the dataset, they applied fuzzy c-means (FCM) clustering and pattern recognition techniques, achieving 68.04% accuracy, 75.34% sensitivity, and 45.83% specificity.

Amit et al. [5] have proposed a novel method of categorizing Parkinson's disease patients according to their postural instability, utilizing the L2 norm metric in combination with a support vector machine [6]. The University of Pennsylvania 40-item scent identification test (UPSIT-40) and the 16-item Sniffins Sticks identification test were used by the authors in. Brazil's population was the subject of this investigation. The authors used logistic regression, taking into account each of the aforementioned characteristics independently. They noted that the Sniffin Sticks provided 81.1 percent sensitivity and 89.0 percent specificity. Similarly, they found out that the UPSIT-40 specificity was 83.5% and sensitivity 82.1%.

Prashant et al. [7] have employed the 40-item UPSIT for olfactory loss feature loss and the Rapid Eye Movement Sleep Behavior Disorder Screening Questionnaire (RBDSQ) for sleep behavior disorder. To train their algorithms, classification tree and support vector machine techniques have been used. They have claimed a sensitivity of 90.55% and an accuracy of 85.48%. The authors have expanded upon this work. SPECT imaging markers and CSF measurements are new features added in this paper. 96.40% accuracy and 97.03% sensitivity were reported. We are motivated to continue the investigation because of this report. An attempt has been made to use sophisticated machine learning models to increase accuracy in the current paper. Recent machine learning methods have been selected for prediction, and these models' comparative performance has been analyzed using ROC curve area, accuracy, and other metrics.

2.2 PERFORMANCE OF THE RELATED WORKS

Table 2.2 provides an overview of the performance metrics of authors' related work compared to your work. Each author's work is assessed based on various criteria such as accuracy and sensitivity.

Table 2.2: performance of the related works

Authors	Best performing algorithom	Accuaracy	Sensitivity
Shamli and Sathiyabhama [1]	ANN	91.00%	96.00%
Azad et al. [2]	SVM	87.61%	91.31%
Sriram et al. [3]	RANDOM FOREST	90.02%	94.32%
Indira et al. [4]	FCM	68.04%	75.34%
Amit et al. [5]	UPISIT-40	83.04%	89.00%
Prashant et al. [7]	CSF	96.40%	97.03%

Chapter 3 METHODOLOGY

3.1 DATASETS

The data set we use in this work is Kaggle csv dataset and it is composed of 197 registered samples with 22 Parkinson disease indicators, and the label which indicates if there is an event or not. The dataset contains name, MDVP: Fo (Hz), MDVP: Fhi (Hz), MDVP: Flo (Hz), MDVP: Jitter (%), MDVP: Jitter (Abs), MDVP: RAP, MDVP:PPQ, Jitter:DDP, MDVP:Shimmer, MDVP:Shimmer(dB), Shimmer:APQ3, Shimmer:APQ5, MDVP:APQ, Shimmer:DDA, NHR, HNR, status, RPDE, DFA, spread1, spread2, D2, PPE attributes which are explained in **Table 3.1**. It includes attribute id, names of attributes and description of the attribute.

Table 3.1: Description of Parkinson Disease dataset

ID	NAME	DESCRIPTION
F1	MDVP: Fo (Hz)	Average vocal fundamental frequency
F2	MDVP: Fhi (Hz)	Maximum vocal fundamental frequency
F3	MDVP: Flo (Hz)	Minimum vocal fundamental frequency
F4	MDVP: Jitter (%)	Kay Pentax MDVP jitter as percentage
F5	MDVP: Jitter (Abs)	Kay Pentax MDVP absolute jitter in microseconds
F6	MDVP: RAP	Kay Pentax MDVP relative amplitude perturbation
F7	MDVP: PPQ	Kay Pentax MDVP 5-point period perturbation quotient
F8	Jitter: DDP	Avg absolute difference of differences between cycles, divided by the avg period.
F9	MDVP: Shimmer	Kay Pentax MDVP local shimmer
F10	MDVP: Shimmer(dB)	Kay Pentax MDVP local shimmer in decibels
F11	Shimmer: APQ3	3point amplitude perturbation quotient
F12	Shimmer: APQ5	5-point amplitude perturbation quotient
F13	MDVP: APQ	Kay Pentax MDVP 11-point amplitude perturbation quotient

F14	Shimmer: DDA	Measures of variation in amplitude
F15	NHR	Noise to Harmonic Ratio
F16	HNR	Harmonic to Noise Ratio
F17	RPDE	Recurrence Period Density Entropy
F18	DFA	Detrended Fluctuation Analysis
F19	Spread1, Spread2	Non-Linear measure of fundamental frequency
F20	D2	Correlation dimension
F21	PPE	Pitch Period Entropy
F22	Status	0-Healthy,1-PD

3.2 PROPOSED WORK FLOW

At the start of the flowchart, the process or algorithm begins its execution. This is typically represented by a rounded rectangle with the word "Start" inside, indicating the beginning of the process. The process is despite in **Figure 3.2.**

After the start, the flowchart likely includes a step for collecting raw data. This could involve gathering data from various sources such as databases, files, or sensors. The raw data is the initial set of unprocessed information that will be used for further analysis or processing in the project.

Following the data collection step, the flowchart likely includes a preprocessing data step. This involves cleaning the data, handling missing values, and transforming the data into a format that is suitable for analysis. Preprocessing is crucial to ensure the quality and integrity of the data before it is used for modeling or analysis.

Label encoding is a technique used to convert categorical data into numerical data. In this step, each unique category in a categorical feature is assigned a numerical value. For example, if a feature has categories like "red," "green," and "blue," they might be encoded

as 0, 1, and 2, respectively. This step is necessary for many machine learning algorithms that require numerical inputs, such as SVM and logistic regression.

Data standardization is a part of the data preprocessing step. It involves scaling the data so that it has a mean of 0 and a standard deviation of 1. This step is important for algorithms that are sensitive to the scale of the input features, such as SVM and KNN. Standardizing the data ensures that all features contribute equally to the analysis and prevents any one feature from dominating the others.

Feature selection is a crucial step in machine learning projects to select the most relevant features for modeling. This step helps to reduce the dimensionality of the dataset, which can improve the performance of the model and reduce overfitting. There are several techniques for feature selection, such as k-based feature selection, which selects the top k features based on their importance. Other techniques include correlation analysis, forward selection, and backward elimination. The goal of feature selection is to retain the most informative features while discarding irrelevant or redundant ones.

After feature selection, the flowchart likely includes a step to apply various machine learning algorithms to the dataset. These algorithms could include Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes, Logistic Regression, Decision Tree, AdaBoost, XGBoost, Voting Classifier, and Multilayer Perceptron (MLP). Each algorithm is applied to the training data to build a model, which is then evaluated using the testing data to determine its performance.

After feature selection, the flowchart likely includes a step to apply various machine learning algorithms to the dataset. These algorithms could include Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes, Logistic Regression, Decision Tree, AdaBoost, XGBoost, Voting Classifier, and Multilayer Perceptron (MLP). Each algorithm is applied to the training data to build a model, which is then evaluated using the testing data to determine its performance.

After applying the algorithms, the flowchart likely includes a step for performance analysis. In this step, the performance of each model is evaluated using various metrics such as accuracy, precision, recall, and F1-score. This analysis helps to determine which algorithm performs best for the given dataset and problem. It also helps to identify any potential issues with the models, such as overfitting or underfitting, that need to be addressed.

After performance analysis, the flowchart likely includes a step to select the best-performing model as the final predictive model. This model is then used to make predictions on new, unseen data. The predictive model is typically deployed in a production environment where it can be used to make real-time predictions or inform decision-making processes.

After selecting the predictive model, the flowchart likely includes a step to deploy the model using a Flask backend. Flask is a web framework for Python that is commonly used for building web applications, including APIs. The Flask backend provides an interface for users to interact with the predictive model, allowing them to make predictions using the model's capabilities.

In the context of your project, the RESTful API likely refers to the Flask backend that serves as the API for interacting with the predictive model. RESTful APIs are designed to be stateless and use standard HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources. In this case, the API would provide endpoints for making predictions using the deployed model, allowing users to send requests with input data and receive predictions in response.

Following the Flask backend, the flowchart likely includes a step to develop the frontend using Angular. Angular is a popular framework for building dynamic web applications. The Angular frontend would provide a user interface for interacting with the RESTful API. Users can input data through the frontend, which is then sent to the backend for processing using the deployed predictive model. The frontend can display the results of the predictions to the user, providing a user-friendly interface for interacting with the model.

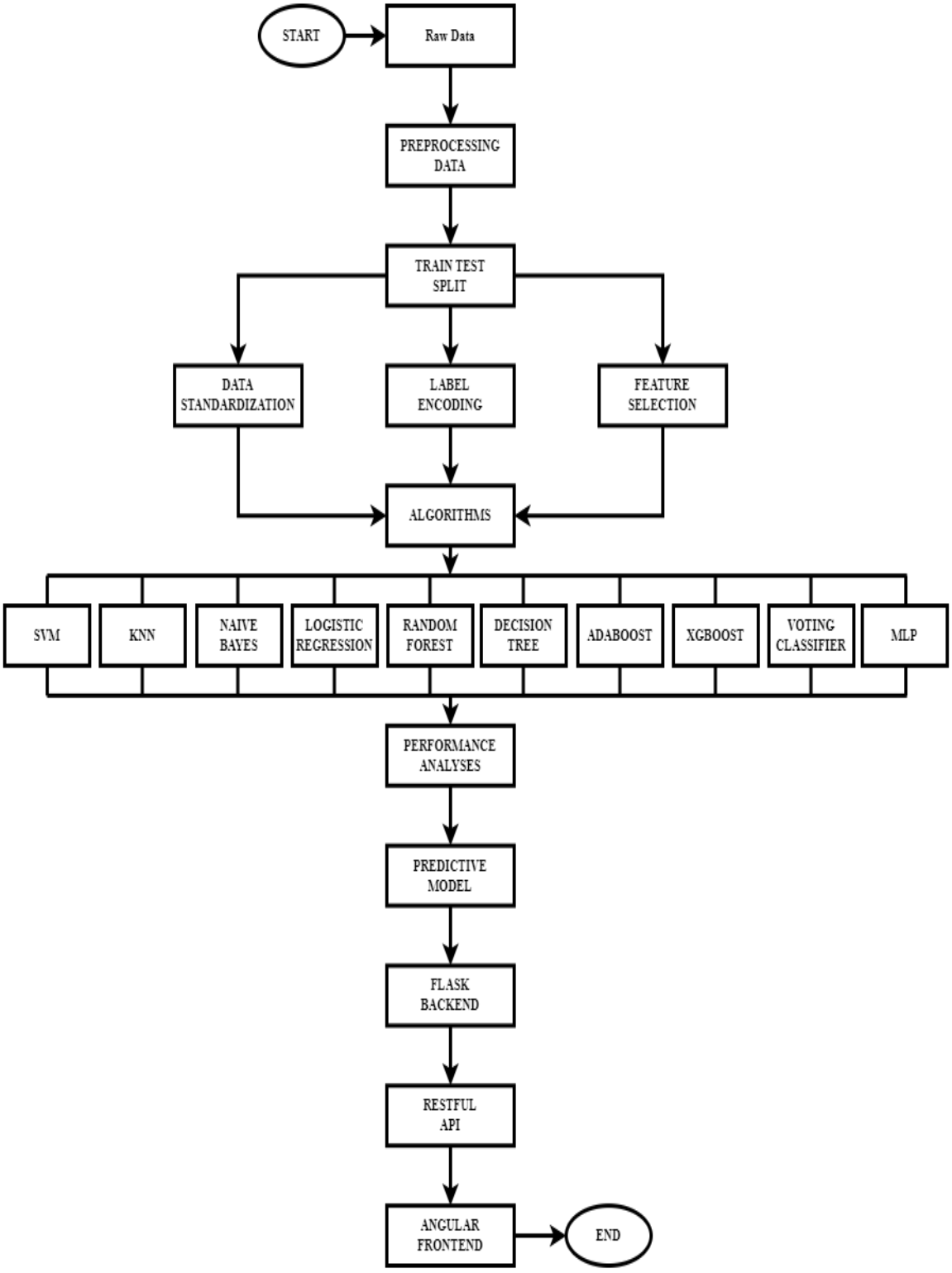

Figure 3.2: Proposed Workflow

The flowchart ends with the completion of the project, indicating that the development and deployment of the predictive model with the Flask backend and Angular frontend have been successfully completed. This final step marks the end of the flowchart and the project as a whole.

3.3 CORRELATION TABLE AND HEATMAP

Correlation Table: A correlation table, also known as a correlation matrix, is a table that shows the correlation coefficients between pairs of variables in a dataset [8]. The correlation coefficient measures the strength and direction of the linear relationship between two variables. It ranges from -1 to 1, where:

- 1 indicates a perfect positive correlation (as one variable increases, the other also increases),
- -1 indicates a perfect negative correlation (as one variable increases, the other decreases), and
- 0 indicates no linear correlation.

Use: It is used to quickly identify which variables are positively or negatively correlated with each other, helping in understanding the relationships within the dataset.

Heatmap: A heatmap is a graphical representation of data where the values of a matrix are represented as colors. In the context of correlation, a heatmap is used to visualize the correlation matrix. Each cell in the heatmap corresponds to the correlation coefficient between two variables, and the color of the cell indicates the strength and direction of the correlation [9].

Use: It provides a visual summary of the correlation matrix, making it easier to identify patterns and relationships between variables. Stronger correlations are often represented by darker or more intense colors, while weaker correlations are represented by lighter colors.

We use correlation tables to understand how different variables in our Parkinson's disease prediction model are related to each other described in **Figure 3.3**. These tables show the strength and direction of the relationships between pairs of variables. A high positive correlation indicates that as one variable increases, the other tends to increase as well, while a high negative correlation means that as one variable increases, the other tends to decrease. A correlation close to zero suggests little to no linear relationship between the variables. By analyzing these correlations, we can identify which features are most closely linked to Parkinson's disease and use this knowledge to enhance the accuracy of our prediction model.

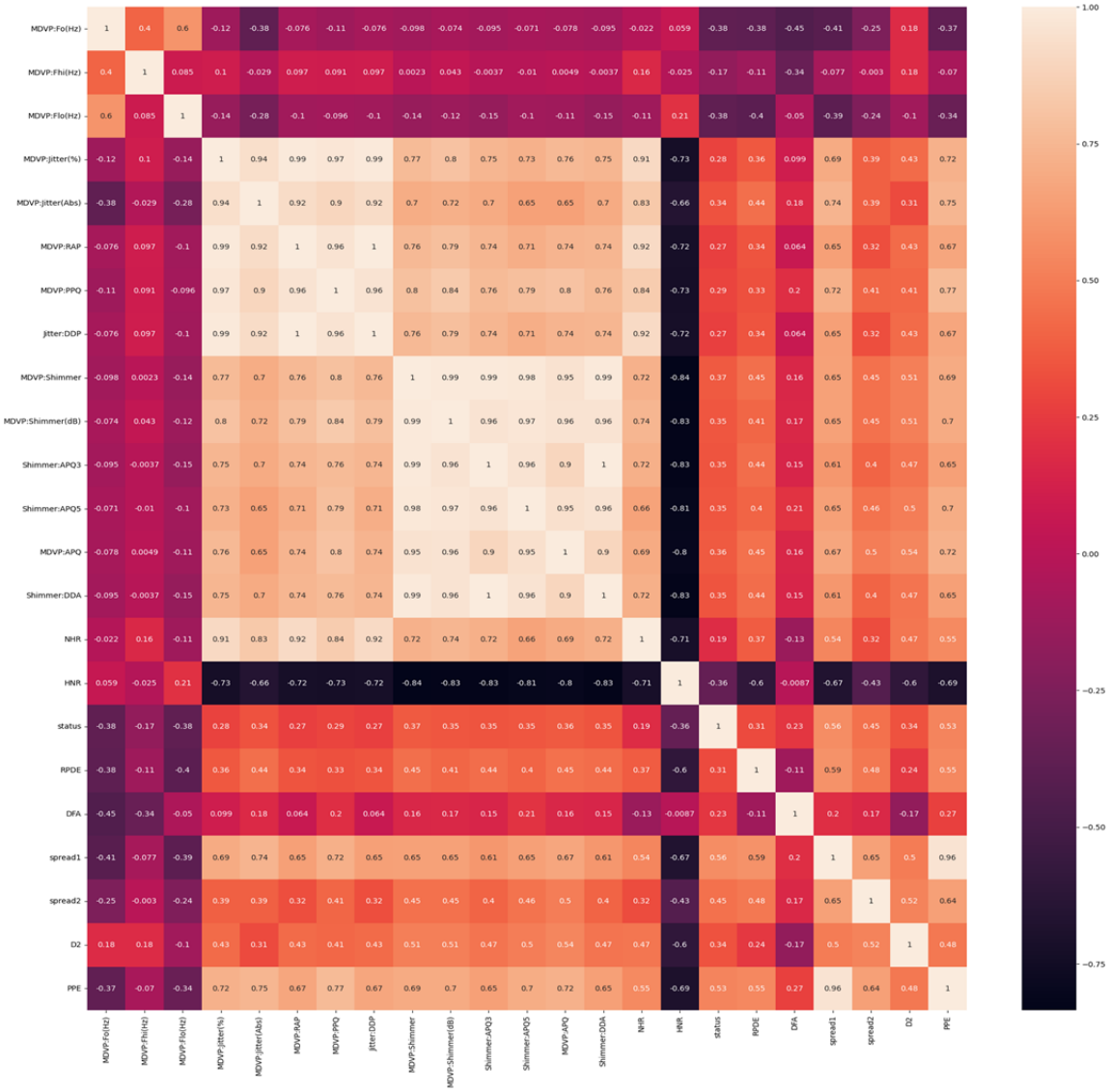

Figure 3.3.: List of Raw Data Heatmap

3.4 FEATURE SELECTION

Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction [10]. It is an important step in the machine learning pipeline because it can:

1. **Reduce Overfitting:** Using fewer features can reduce the chances of overfitting, where a model performs well on the training data but poorly on new, unseen data.

2. **Improve Model Performance:** By selecting the most relevant features, you can often improve the performance of your model by focusing on the most important information.

3. **Reduce Training Time:** Working with a smaller set of features can reduce the computational resources and time required to train the model.

4. **Enhance Interpretability:** Models with fewer features are often easier to interpret and understand, which can be important for decision-making.

There are several approaches to feature selection, including:

1. **Filter Methods:** These methods select features based on their statistical properties, such as correlation with the target variable or information gain. Common techniques include Pearson correlation coefficient, mutual information, and chi-squared test.

2. **Wrapper Methods:** These methods evaluate different subsets of features using the target machine learning algorithm itself. Examples include forward selection, backward elimination, and recursive feature elimination (RFE).

3. **Embedded Methods:** These methods perform feature selection as part of the model training process. Techniques like LASSO (Least Absolute Shrinkage and Selection Operator) and tree-based methods like Random Forests include feature selection as an inherent part of their algorithm.

The choice of feature selection method depends on the specific dataset, the machine learning algorithm being used, and the goals of the analysis (e.g., predictive accuracy,

interpretability). It's often a good practice to experiment with different methods and evaluate their performance using cross-validation or other validation techniques [11].

3.5 CLASSIFIER MODELS

A Parkinson disease diagnosis, further testing, such as staging and grading, is conducted to determine the severity of the condition and to choose the best course of treatment. This paper will use eight algorithms:

i. Logistic Regression
ii. k-nearest neighbors (KNN)
iii. Support Vector Machine (SVM)
iv. Random Forest
v. XGBoost (Xtreme Gradient Boosting)
vi. Decision Tree
vii. AdaBoost (Adaptive Boosting)
viii. Voting Classifier
ix. Naïve Bayes
x. Neural Network (MLP)

3.5.1 LOGISTIC REGRESSION

In binary classification problems, a statistical method known as logistic regression is used to forecast one of two outcomes (e.g., "yes" or "no", "positive" or "negative", etc.).
The method is built on the logistic function, sometimes known as the sigmoid function. Any input value is converted to a value between 0 and 1. The likelihood of the positive class can be thought of as this outcome value. The logistic regression approach chooses the proper parameters (coefficients) for a certain set of input qualities in order to maximize the likelihood of the observed data.

The logistic regression model may be expressed using the equation below:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-w \times x}} \qquad \text{.....(i)}$$

Where w are the coefficients, e is the Euler's number, -w×x is the log-odds or log it, and p(y=1|x) is the probability of the positive class. x is the input characteristics.
The logistic regression approach may be trained using maximum likelihood estimation (MLE), which may then be regularized to prevent over fitting by incorporating a penalty term in the cost function, such as L1 or L2 regularization. The outcomes of the logistic regression model may be used to anticipate future data as well as to understand the relationship between the class of the output and the input variables. It is extensively used in a number of sectors, including banking.

3.5.2 K-NEAREST NEIGHBOURS (KNN)

The k-nearest neighbors (KNN) algorithm is a simple, non-parametric method for classification and regression problems. In order to label (or, in the case of regression, assign a value) the new example based on the majority of the k nearest neighbors, the KNN approach first identifies the k training cases that are, in some respects, comparable to a new, unseen example.
The algorithm functions as follows:

- The user specifies the value of k, which determines the number of nearest neighbors to consider.
- The method finds the k training examples that are closest to a brand-new, unidentified example based on some distance metric (such as Euclidean distance).
- The k nearest neighbors' majority class is used to categorize the new example. The new instance in regression is subjected to the average of the k nearest neighbors.

The distance metric to identify the nearest neighbors can be any function that computes the separation between two places. Common distance metrics include the Euclidean distance, the Manhattan distance, and cosine similarity.
Two of the main advantages of the KNN algorithm are its flexibility and simplicity. It works effectively with little training data and may be used for classification and regression problems. KNN may face the impacts of dimensionality and become computationally

expensive when there are plenty of samples or features. To address this problem, KNN may be used with dimensionality reduction techniques like PCA.

3.5.3 SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine (SVM), a supervised machine learning technique, is used to tackle classification and regression problems. Its foundation is the notion of finding a hyper plane in a dataset that maximally divides many classes. The approach first transfers the input data into a higher dimensional space, then creates a linear border that separates the different classes.

The hyper plane that separates the classes with the maximum margin is first found by the SVM algorithm. The hyper plane's distance from the closest data points for each class is then calculated. This hyper plane is known as the biggest margin hyper plane. The nearest data points to the hyper plane are called support vectors because they help build it.

The SVM algorithm's regularization parameter C controls the trade-off between increasing margin and minimizing classification error. With a lower value of C, the margin will be bigger, but there can also be more classification errors. If C is higher, the margin will be less but there will be fewer classification errors.

The SVM method may also be used for non-linearly separable data by utilizing a kernel function that transforms the input data into a higher dimensional space where it is possible to identify a linear boundary. Typical kernel functions include the radial basis function (RBF) and the polynomial kernel.

3.5.4 RANDOM FOREST

For classification and regression problems, Random Forest is a supervised machine learning technique. It is built on the concept of ensemble learning, which combines several decision trees to create a more accurate model.

Using a separate random subset of the data and attributes, the approach builds a number of decision trees. A random subset of characteristics is chosen at each split, once each tree has reached its maximum depth during training, to lessen over fitting.

Based on the majority of predictions generated by each decision tree, the algorithm selects a prediction. As a result, the variance and bias of the model are decreased, increasing the predictability and accuracy.

Random Forest also includes a feature importance measure to evaluate the value of each feature in the dataset. This might be useful in identifying the main traits that affect the forecast.

Since it can handle huge amounts of data and high dimensional features, the random forest approach is highly suited for a range of applications including image classification, natural language processing, and financial forecasting. It is also tolerant to outliers and does not require significant feature scaling.

Overall, Random Forest is a powerful tool for classification and regression problems, particularly when the data is high dimensional or has a lot of properties. It is widely used in a variety of fields, including as bioinformatics, financial forecasting, and the classification of texts and pictures.

3.5.5 XG-BOOST (EXTREME GRADIENT BOOSTING)

For classification and regression problems, XGBoost (extreme Gradient Boosting), a supervised machine learning technique, is used. Gradient boosting, which combines a number of weak decision trees into a single stronger model, is the basis of this approach.

Using a separate random subset of the data and attributes, the approach builds a number of decision trees. A random subset of characteristics is chosen at each split, once each tree has reached its maximum depth during training, to lessen over fitting.

It also uses a technique called "gradient boosting," which improves the model by leveraging the gradient of the loss function. In order to decrease the loss function, the algorithm is able to change the weights of the features and samples. Additionally, XGBoost is much faster and more efficient than traditional gradient boosting methods due to its ability to parallelize the tree-building process. It also has a lambda regularization option that controls the trade-off between raising margin and lowering classification error. XGBoost also provides a feature importance measure to evaluate the value of each feature in the dataset. This might be useful in identifying the main traits that affect the forecast.

Since it can handle huge amounts of data and high dimensional features, the XGBoost approach is useful for a range of applications, including image classification, natural language processing, and financial forecasting. It is also tolerant to outliers and does not require significant feature scaling. All things considered, XGBoost is a powerful method for classification and regression problems, particularly when the data is high dimensional or has a large number of features. It is widely used in a variety of fields, including as bioinformatics, financial forecasting, and the classification of texts and pictures.

3.5.6 DECISION TREE

For classification and regression problems, a decision tree, a supervised machine learning method, is used. It is based on the notion of a tree-like structure, where each leaf node represents an outcome or prediction and each inner node represents an assessment based on a characteristic of the data.

The algorithm initially selects the feature that divides the data into many classes the best. This component is known as the root node. After that, the algorithm separates the data into groups based on the selected feature, selecting the best feature at each level of the tree until a stopping condition is met.

Using a criterion like information gain, Gini index, or entropy, the approach chooses the best characteristic to divide on. It is decided to use the feature that reduces impurities the most. The impurity of the data before and after the split is compared using these criteria.

The Decision Tree approach is simple to understand and interpret, and it is not sensitive to outliers. It can handle both continuous and categorical variables. When working with large datasets, it can be computationally expensive and the tree may easily over fit if it is built too thoroughly.

Generally speaking, decision trees are a good way to solve classification and regression problems, especially when the data is low dimensional or has few properties. It is widely used in a variety of fields, including as bioinformatics, financial forecasting, and the classification of texts and pictures.

3.5.7 ADABOOST (ADAPTIVE BOOSTING)

AdaBoost (Adaptive Boosting), a powerful ensemble machine learning algorithm, may improve the accuracy of other models by combining the predictions of several models. A series of weak models that each serve as a condensed representation of the primary problem are regularly trained using this approach. The projections from these weak models are then combined into a weighted aggregate, with the models with superior performance receiving more weight. This procedure is continued until the required level of accuracy is obtained.

The strategy first evenly weights each training sample before teaching a weak classifier to correctly classify as many occurrences as it can. Then, before increasing the weights of the improperly classified cases, a weak classifier is trained to correctly classify as many instances as is practical. This process is repeated several times, and the final classifier is a weighted composite of all the weak classifiers.

AdaBoost is typically used for classification problems, while it may also be used for regression and other kinds of problems. It is especially useful for datasets that are imbalanced or have a lot of attributes. The method is simple to apply and has proved successful in a number of real-world scenarios.

AdaBoost is a machine learning ensemble technique that combines forecasts from numerous models to improve the precision of other models. A series of weak models that each serve as a condensed representation of the primary problem are regularly trained using this approach. The projections from these weak models are then combined into a weighted aggregate, with the models with superior performance receiving more weight. This procedure is continued until the required level of accuracy is obtained. AdaBoost is typically used to address classification-related problems, but it may also address regression-related problems and other problems.

3.5.8 VOTING CLASSIFIER

A voting classifier is an ensemble method in machine learning where multiple models (classifiers) are used to make predictions on a dataset, and the final prediction is determined based on a majority vote or weighted vote of the individual models. There are two main types of voting classifiers:

Hard Voting: In hard voting, each individual classifier in the ensemble predicts the class label, and the majority class label is chosen as the final prediction. This is suitable for classification tasks where the class labels are discrete [12].

Soft Voting: In soft voting, each individual classifier in the ensemble predicts the class probabilities for each class, and the average probabilities across all classifiers are calculated for each class. The class with the highest average probability is chosen as the final prediction. Soft voting can provide more nuanced predictions by taking into account the confidence of each classifier [13].

Voting classifiers can be used with various types of base classifiers, such as decision trees, support vector machines, logistic regression, etc. They are often used to improve the overall performance and robustness of a model by combining the strengths of multiple classifiers. Voting classifiers are particularly effective when the individual classifiers have different strengths and weaknesses, as they can complement each other and make more accurate predictions as a group.

It's important to note that the effectiveness of a voting classifier depends on the diversity of the individual classifiers in the ensemble. If all the classifiers are similar or highly correlated, the ensemble may not perform significantly better than the individual classifiers.

3.5.9 NAÏVE BAYES

Naive Bayes is a probabilistic classifier based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Here's a brief overview without equations:

1. **Bayes' Theorem:** Bayes' theorem is a fundamental theorem in probability theory that describes the probability of an event, based on prior knowledge of conditions that might be related to the event. It calculates the probability of an event occurring, given the probability of another event that has already occurred [14].
2. **Naive Assumption:** In Naive Bayes, the "naive" assumption is that all features are independent of each other given the class label. This means that the presence or absence of a particular feature is independent of the presence or absence of any other feature, given the class label. While this assumption is rarely true in real-world datasets, Naive Bayes can still perform well, especially when the features are not strongly correlated [15].
3. **Classification:** Given a set of features and a class label, the Naive Bayes classifier calculates the probability of each class label given the features using Bayes' theorem. It then selects the class label with the highest probability as the predicted class for the given features [16].

4. **Types of Naive Bayes:** There are different variants of Naive Bayes, including Gaussian Naive Bayes (for continuous features assuming a Gaussian distribution), Multinomial Naive Bayes (for discrete features, commonly used in text classification with word counts), and Bernoulli Naive Bayes (similar to Multinomial Naive Bayes but for binary features) [17].

Naive Bayes classifiers are known for their simplicity and ability to perform well in practice, especially with high-dimensional datasets and when the independence assumption is not severely violated. They are commonly used in text classification, spam filtering, and other tasks involving categorical or count-based features.

3.5.10 NEURAL NETWORK (MLP)

A Multilayer Perceptron (MLP) is a type of artificial neural network (ANN) that consists of multiple layers of interconnected nodes, or neurons, organized in a feedforward manner. Each neuron in one layer is connected to every neuron in the next layer, and there can be multiple hidden layers between the input and output layers. MLPs are capable of learning complex non-linear relationships in data and are widely used in various machine learning applications.

Here's a brief overview of MLPs:

1. **Architecture:**
    - **Input Layer:** The input layer consists of neurons that represent the input features of the dataset.
    - **Hidden Layers:** There can be one or more hidden layers, each consisting of multiple neurons with activation functions. These layers are responsible for learning complex patterns in the data.
    - **Output Layer:** The output layer produces the final output of the network. The number of neurons in this layer depends on the type of task (e.g., regression, classification) and the number of classes or dimensions of the output [18].

2. **Activation Functions:** Each neuron in an MLP typically uses an activation function to introduce non-linearity into the network. Common activation functions include sigmoid, tanh, ReLU (Rectified Linear Unit), and softmax (for multi-class classification) [19].

3. **Training:** MLPs are trained using optimization algorithms such as gradient descent. The weights and biases of the neurons are adjusted iteratively to minimize a loss

function, which measures the difference between the predicted outputs and the actual outputs [20].

4. **Applications:** MLPs are used in a wide range of applications, including pattern recognition, image classification, natural language processing, and financial forecasting.

3.6 K-BASED FEATURE SELECTION

K-based feature selection, also known as k-best feature selection, is a technique used to select the top k most relevant features from a dataset based on their statistical significance or importance scores. This approach is commonly used in machine learning to reduce the dimensionality of the feature space and improve the performance of models by focusing on the most informative features [21].

3.7 CONFUSION MATRIX

A classification system's performance is evaluated using a table called a confusion matrix. The algorithm's performance is summarized by comparing the predicted class to the actual class given a set of test data. The matrix's columns represent the expected class, while its rows represent the actual class. The percentage of test data instances that fall into each category is shown by the values in the matrix. The number of correct predictions is reflected in the matrix's diagonal cells, while the reverse is reflected in the matrix's off-diagonal cells. The matrix may be used to compute a number of performance metrics, including as accuracy, precision, recall, and F1-score.

The effectiveness of a categorization system is represented by a table called a confusion matrix. It compares the anticipated class labels with the actual class labels for a set of test data. The matrix's columns represent the expected class, while its rows represent the actual class. The percentage of test data instances that fall into each category is shown by the values in the matrix.

The number of accurate predictions is represented in the diagonal cells of the matrix as either the true positives (TP) or true negatives (TN), depending on the binary classification.

False positive (FP) and false negative (FN) cells, also known as off-diagonal cells, represent the number of incorrect predictions (FN).

The matrix may be used to determine the following performance indicators:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{.....(ii)}$$

$$Precison = \frac{TP}{TP + FP} \quad \text{.....(iii)}$$

$$Recall = \frac{TP}{TP + FN} \quad \text{.....(iv)}$$

$$F_1 score = \frac{2(Recall \times Precision)}{Recall + Precision} \quad \text{.....(v)}$$

3.8 ROC CURVE

The Receiver Operating Characteristic (ROC) curve is a graphical tool used to evaluate the performance of binary classification models, such as those used in predicting whether a patient has Parkinson's disease. It plots the true positive rate (TPR), also known as sensitivity, against the false positive rate (FPR) at various threshold settings. The true positive rate (TPR) represents the proportion of actual positive cases that are correctly identified as positive by the model, while the false positive rate (FPR) measures the proportion of actual negative cases that are incorrectly classified as positive by the model.

The ROC curve illustrates the trade-off between sensitivity and specificity across different threshold values. A perfect classifier would have an ROC curve that reaches the top-left corner of the plot, indicating high sensitivity (TPR) and low FPR across all thresholds. The area under the ROC curve (AUC) is a commonly used metric to quantify the overall performance of a classifier. A higher AUC value (closer to 1) indicates better discrimination ability of the model across different threshold settings, while an AUC of 0.5 suggests that the model performs no better than random guessing [22].

Chapter 4

PERFORMANCE MEASUREMENT

4.1 ACCURACY SCORE OF USED ALGORITHMS

In our Parkinson's disease prediction project, we employed 10 machine learning models: SVM (Support Vector Machine), KNN (K-Nearest Neighbours), logistic regression, random forest, Naïve Bayes, XGBoost (Extreme Gradient Boosting), Adaboost, voting classifier (SVM, KNN and decision tree voting), and MLP (Multilayer Perceptron). When using all features from our dataset, XGBoost, the Voting Classifier, and MLP achieved the highest score, with an accuracy of 94.87%. However, when employing k-based feature selection to choose the 10 best features from our dataset, KNN outperformed the other models with an accuracy of 97.44%. Table 4.1 in our project report summarizes the accuracy of the models used.

Table 4.1: Accuracy of different machine learning algorithms.

Models	**Accuracy (Without feature selection)**	**Accuracy (With feature selection)**
ADABOOST	87.18%	84.62%
DECISION TREE	92.31%	89.74%
KNN	92.31%	97.44%
LOGISTIC REGRESSION	89.74%	89.74%
MLP	94.87%	94.87%
NAÏVE BAYES	71.79%	87.18%
RANDOM FOREST	89.74%	92.31%
SVM	89.74%	92.31%
VOTING CLASSIFIER	94.87%	92.31%
XGBOOST	94.87%	94.87%

4.2 ROC CURVE OF USED ALGORITMS

In our Parkinson's disease prediction project, we utilized 10 machine learning models: SVM (Support Vector Machine), KNN (K-Nearest Neighbours), logistic regression, random forest, Naïve Bayes, XGBoost (Extreme Gradient Boosting), Adaboost, voting classifier (SVM, KNN and decision tree voting), and MLP (Multilayer Perceptron). The ROC curves for these algorithms are depicted in **Figure 4.2.1** to **Figure 4.2.10.**

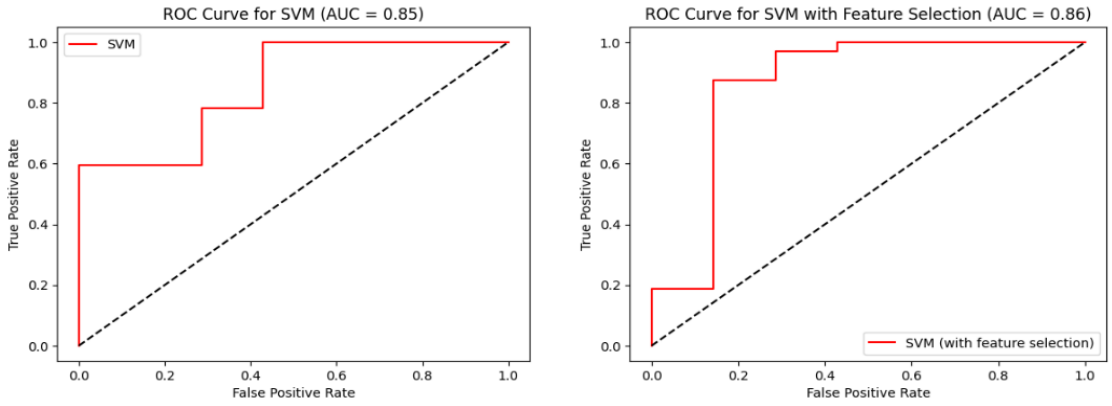

Figure 4.2.1: SVM roc curve with and without using feature selection

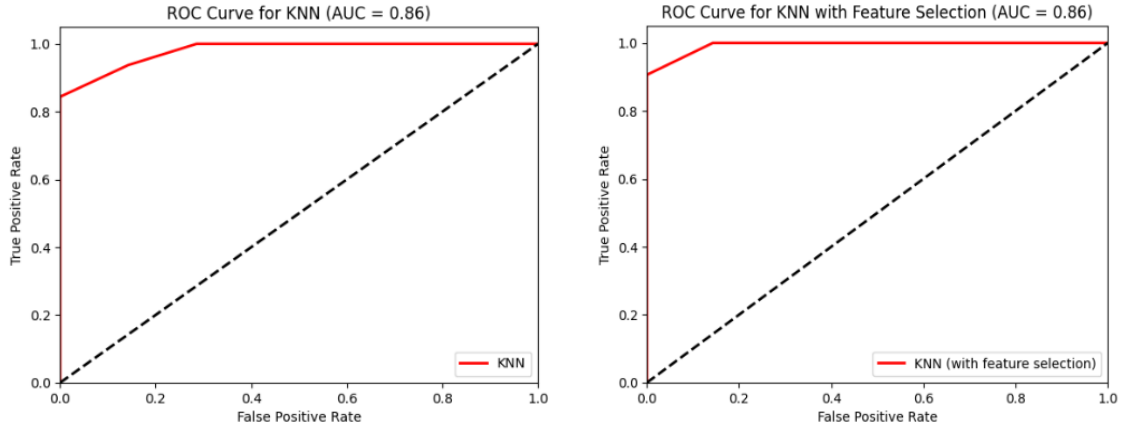

Figure 4.2.2: KNN roc curve with and without using feature selection

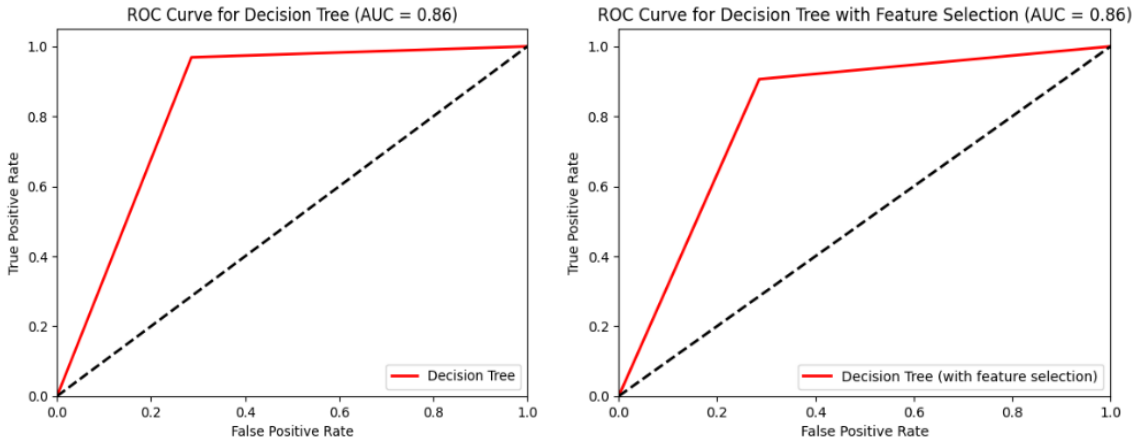

Figure 4.2.3: Decision Tree roc curve with and without using feature selection

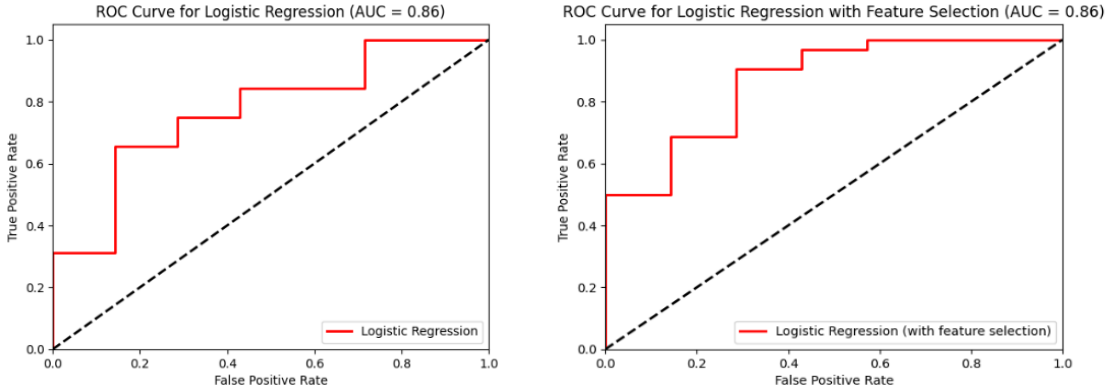

Figure 4.2.4: Logistic regression roc curve with and without using feature selection

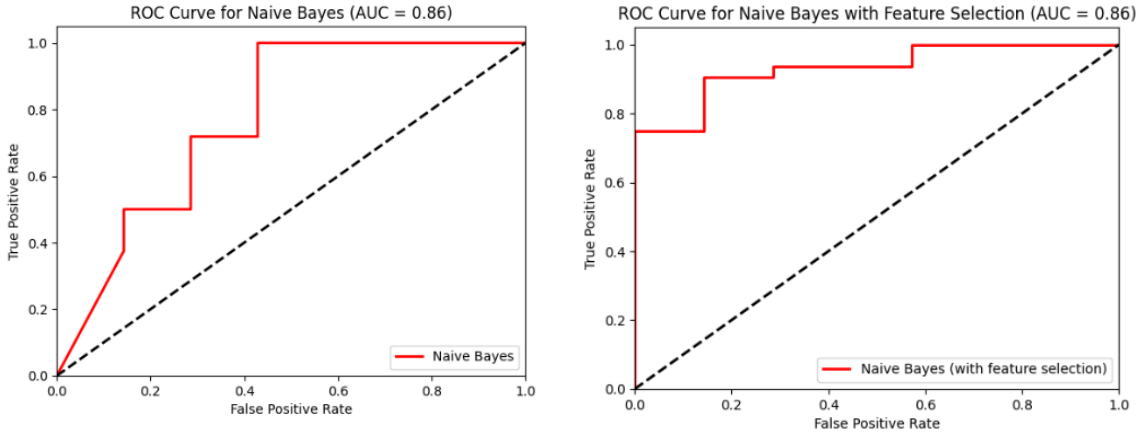

Figure 4.2.5: Naïve Bayes roc curve with and without using feature selection.

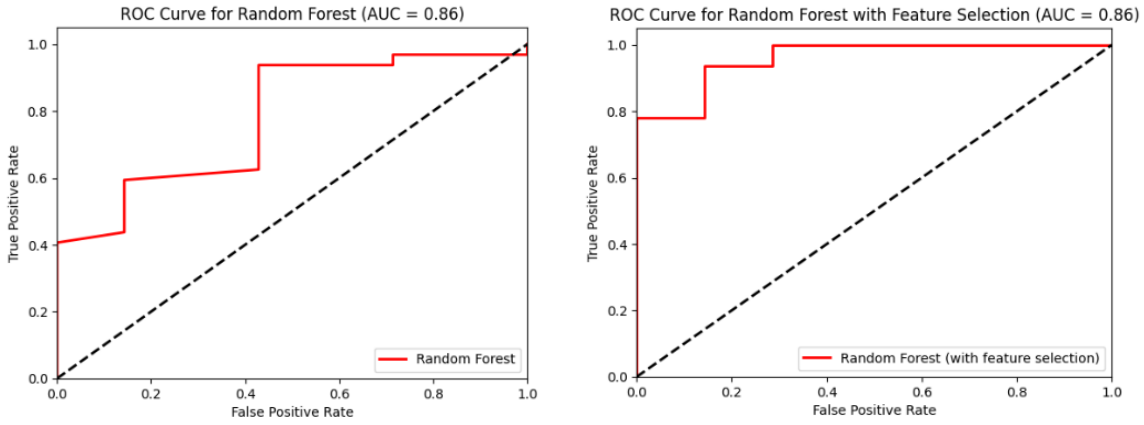

Figure 4.2.6: Random Forest roc curve with and without using feature selection

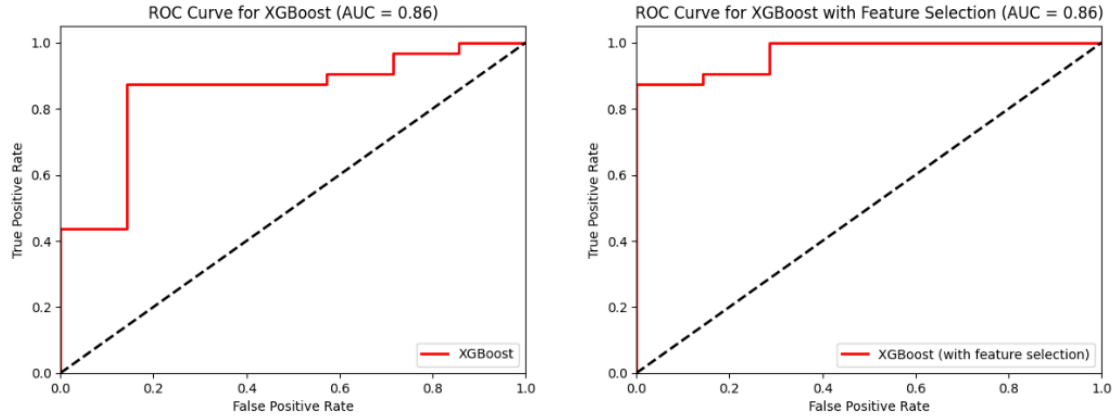

Figure 4.2.7: XGBoost roc curve with and without using feature selection

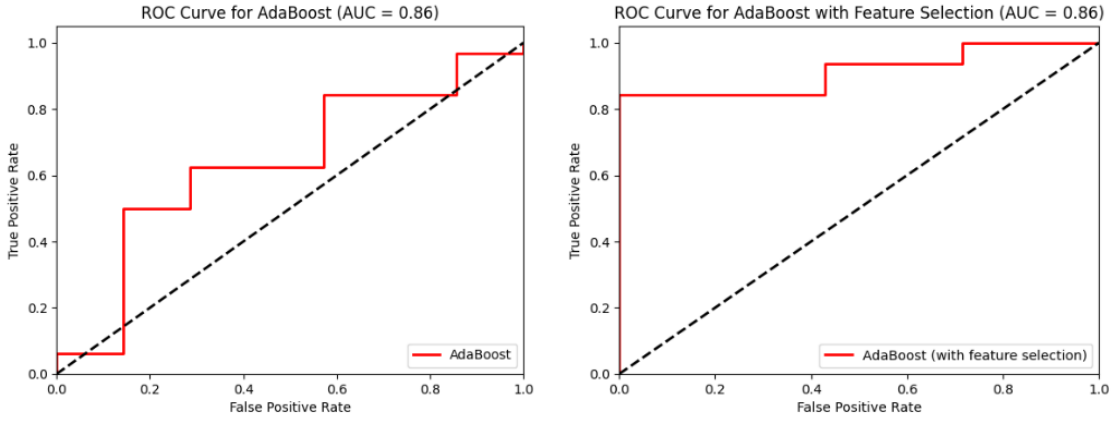

Figure 4.2.8: Adaboost roc curve with and without using feature selection

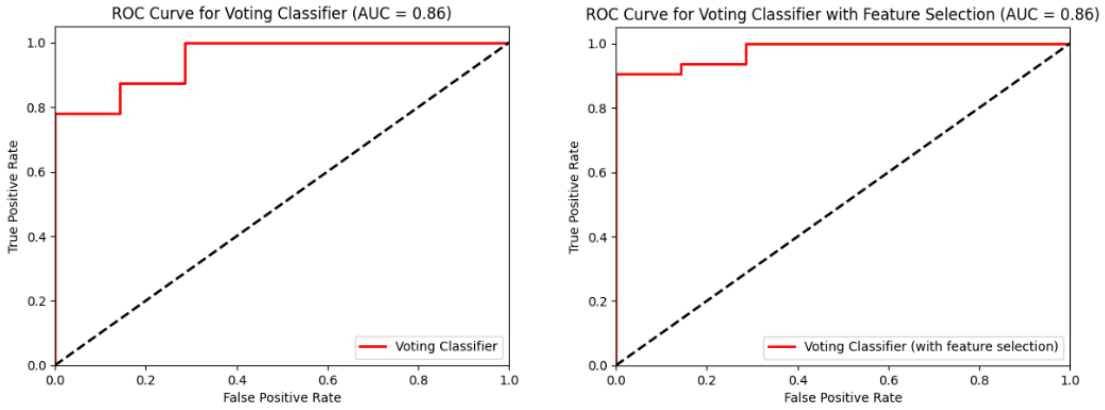

Figure 4.2.9: Voting Classifier roc curve with and without using feature selection

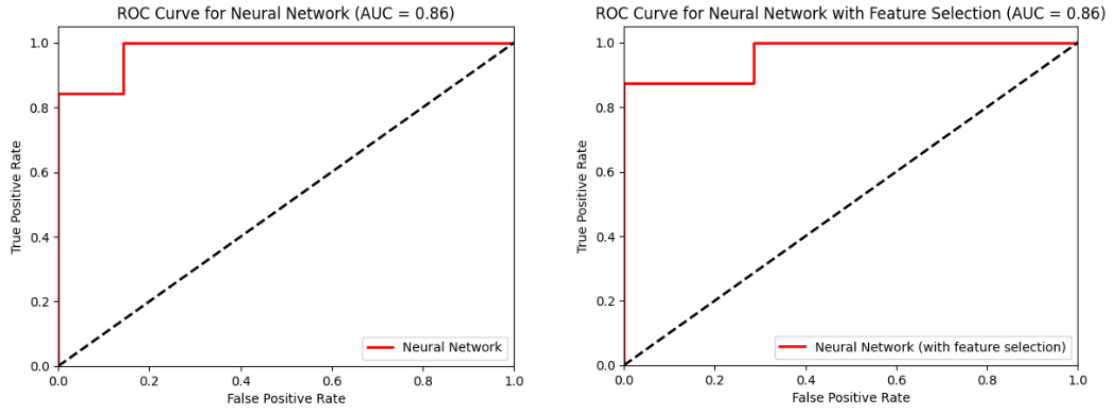

Figure 4.2.10: MLP roc curve with and without using feature selection

4.3 CONFUSION MATRIX OF USED ALGORITHMS

In our Parkinson's disease prediction project, we utilized 10 machine learning models: SVM (Support Vector Machine), KNN (K-Nearest Neighbours), logistic regression, random forest, Naïve Bayes, XGBoost (Extreme Gradient Boosting), Adaboost, voting classifier (SVM, KNN and decision tree voting), and MLP (Multilayer Perceptron). The Confusion matrix with heatmap for these algorithms are depicted in **Figure 4.3.1** to **Figure 4.3.10.**

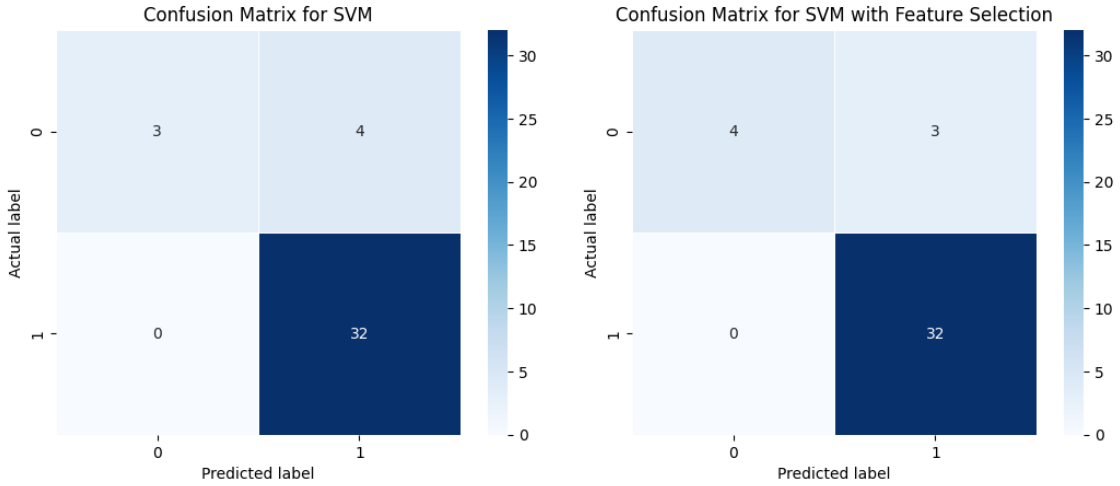

Figure 4.3.1: SVM Confusion matrix with and without using feature selection.

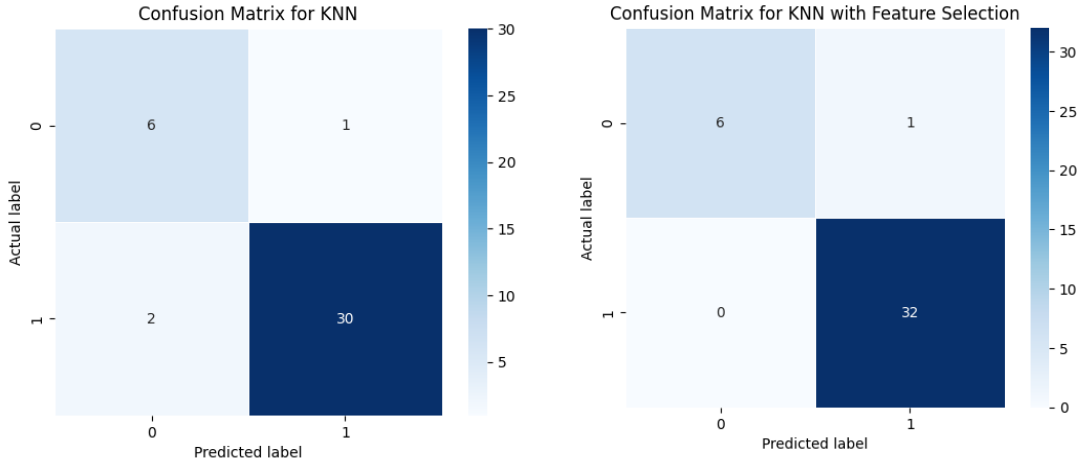

Figure 4.3.2: KNN Confusion matrix with and without using feature selection.

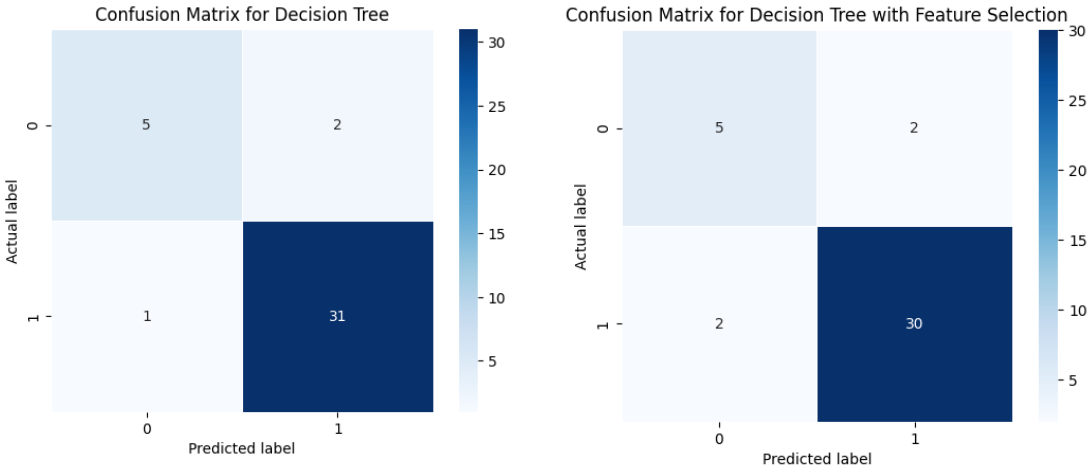

Figure 4.3.3: Decision Tree Confusion matrix with and without using feature selection.

Figure 4.3.4: Logistic regreststssion Confusion matrix with and without using feature selection.

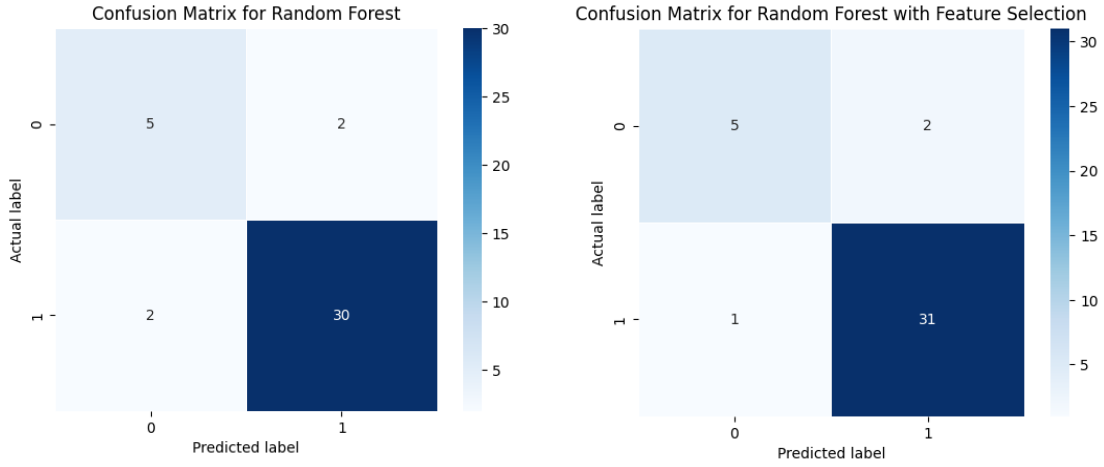

Figure 4.3.5: Random Forest Confusion matrix with and without using feature selection.

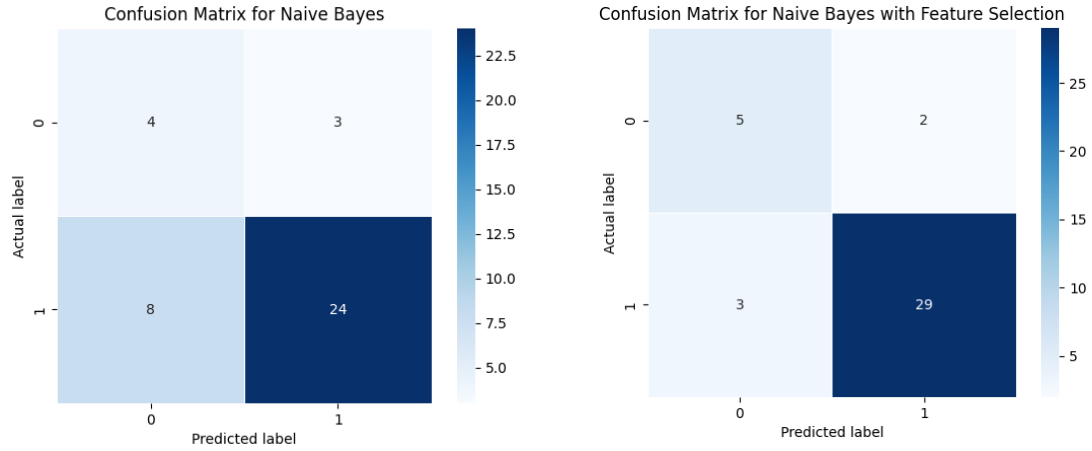

Figure 4.3.6: Naïve Bayes Confusion matrix with and without using feature selection.

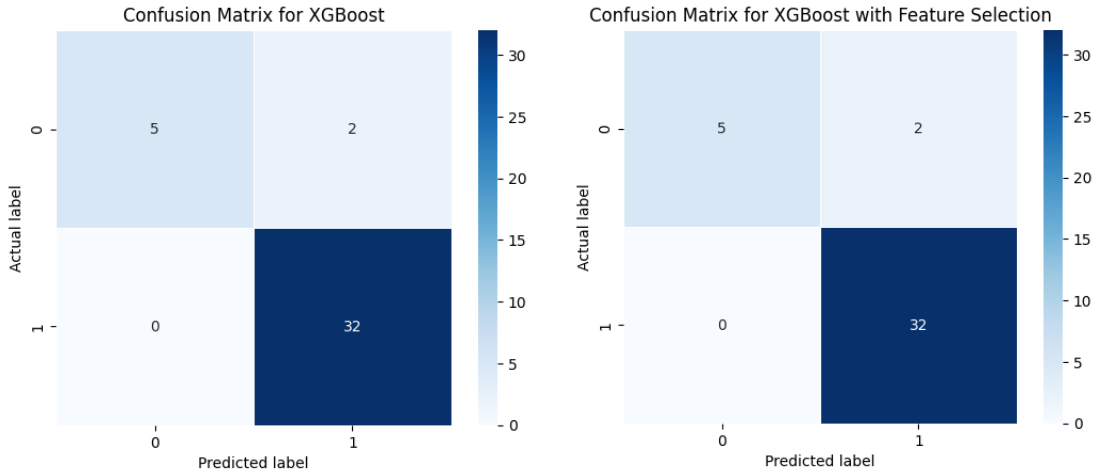

Figure 4.3.7: XGBoost Confusion matrix with and without using feature selection.

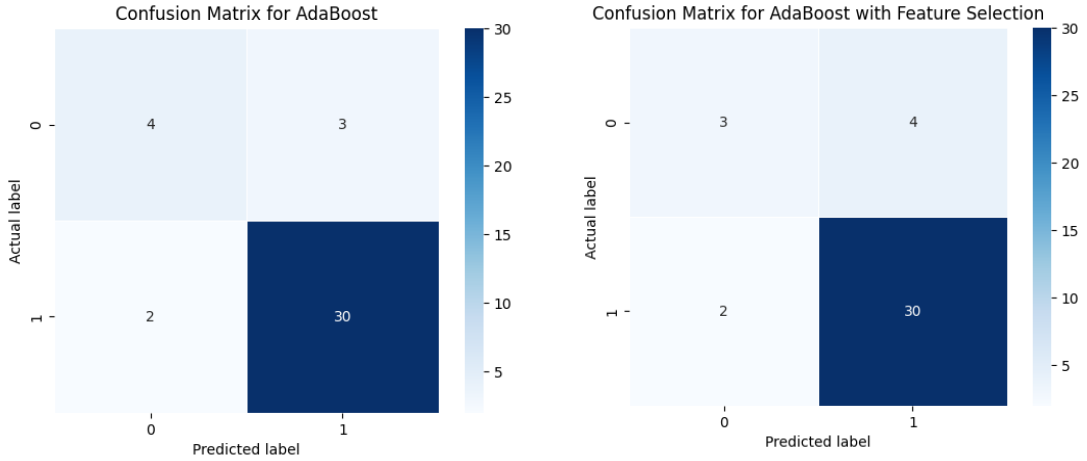

Figure 4.3.8: Adaboost Confusion matrix with and without using feature selection.

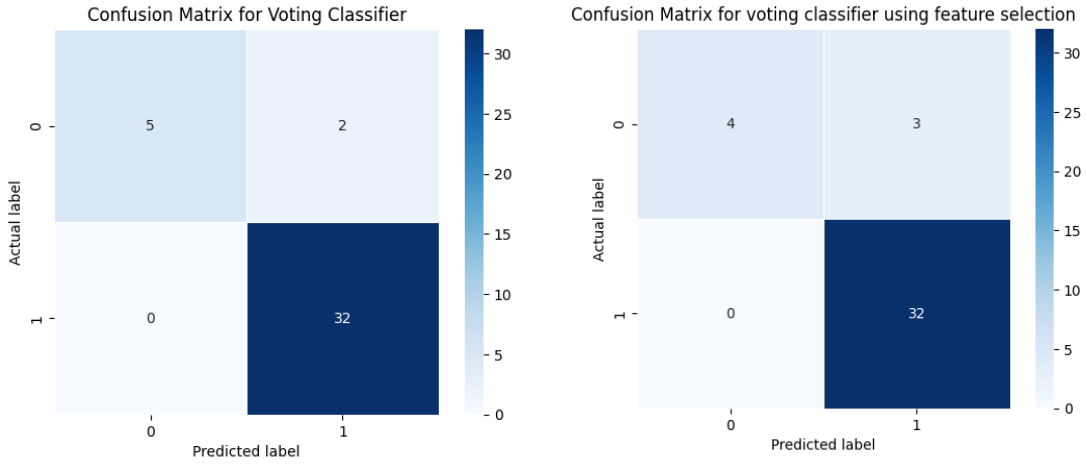

Figure 4.3.9: Voting classifier Confusion matrix with and without using feature selection.

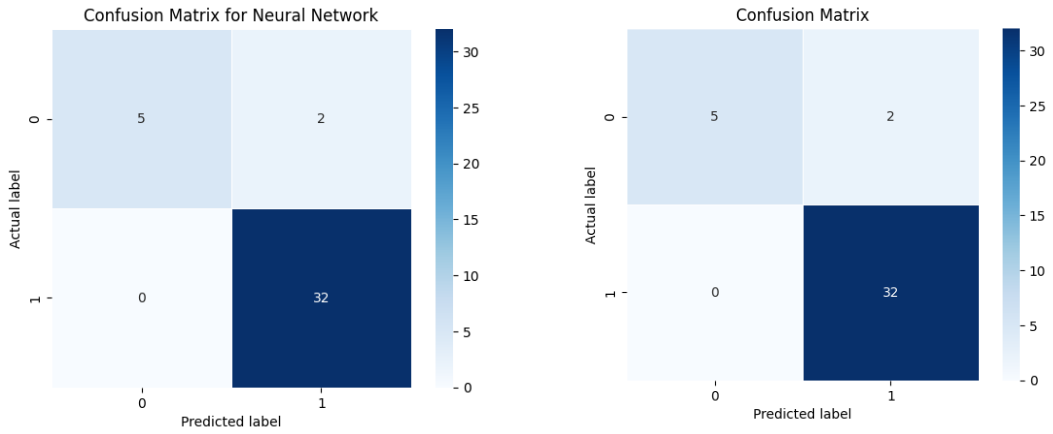

Figure 4.3.10: MLP Confusion matrix with and without using feature selection.

4.4 COMPARE WORK WITH OTHERS

Table 4.4 provides a comparison between various authors' work on predicting Parkinson's disease. Each author's work is evaluated based on different criteria such as accuracy and sensitivity.

Table 4.4: Compare work with others

Authors	Best performing algorithom	Accuaracy	Sensitivity
Shamli and Sathiyabhama [1]	ANN	91.00%	96.00%
Azad et al. [2]	SVM	87.61%	91.31%
Sriram et al. [3]	RANDOM FOREST	90.02%	94.32%
Indira et al. [4]	FCM	68.04%	75.34%
Amit et al. [5]	UPISIT-40	83.04%	89.00%
Prashant et al. [7]	CSF	96.40%	97.03%
Proposed	KNN	97.44%	97.44%

Chapter 5

IMPLEMENTATION

5.1 USED LIBRARIES AND FRAMEWORKS

We relied on a diverse array of libraries and frameworks to develop our project, each serving a crucial role in its implementation. Below, we highlight some of the prominent ones that significantly contributed to our work.

5.1.1 NUMPY

NumPy is a fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is widely used in various fields such as scientific computing, machine learning, data analysis, and more [24].
Here's a brief overview of NumPy's key features:

Multi-dimensional arrays: NumPy offers a powerful array object called ndarray, which enables efficient storage and manipulation of large datasets.

Mathematical functions: NumPy provides a wide range of mathematical functions for performing operations on arrays, such as arithmetic operations, trigonometric functions, statistical functions, and linear algebra operations.

Broadcasting: NumPy allows for arithmetic operations between arrays of different shapes and sizes through broadcasting, which simplifies array manipulation and reduces the need for explicit looping.

Integration with other libraries: NumPy seamlessly integrates with other Python libraries such as SciPy (for scientific computing), Matplotlib (for data visualization), and Pandas (for data manipulation and analysis), forming a powerful ecosystem for data science and numerical computing.

5.1.2 PANDAS

Pandas is a powerful Python library used for data manipulation and analysis. It provides data structures and functions designed to make working with structured data fast, easy, and expressive. Pandas is widely used in various domains including data science, finance, economics, and academia [25].

Key features of Pandas include:

data Frame: Pandas introduces the Data Frame data structure, which is a two-dimensional labeled data structure with columns of potentially different data types. It provides functionalities similar to a spreadsheet or SQL table, making it easy to handle and analyze tabular data.

Series: Pandas also offers the Series data structure, which is a one-dimensional labeled array capable of holding any data type. Series can be thought of as a single column of data in a Data Frame.

Data manipulation: Pandas provides a rich set of functions for data manipulation tasks such as indexing, slicing, filtering, merging, grouping, reshaping, and pivoting. These functions enable users to clean, transform, and preprocess data efficiently.

Data alignment and handling missing data: Pandas automatically aligns data based on labels and handles missing data seamlessly, providing flexible options for filling, dropping, or interpolating missing values.

Time series functionality: Pandas offers robust support for working with time series data, including date/time indexing, resampling, shifting, and rolling window calculations.

Input/output tools: Pandas provides functions to read and write data from/to various file formats such as CSV, Excel, SQL databases, JSON, and HDF5, facilitating seamless data integration and interoperability.

5.1.3 SEABORN

Seaborn is a Python visualization library based on matplotlib that provides a high-level interface for creating attractive and informative statistical graphics. It is built on top of matplotlib and integrates seamlessly with Panda's data structures, making it easy to create complex visualizations with minimal code [26].

Key features of Seaborn include:

High-level interface: Seaborn simplifies the process of creating complex statistical visualizations by providing a high-level interface for common plot types such as scatter plots, bar plots, box plots, violin plots, and heatmaps.

Statistical plotting: Seaborn offers specialized functions for visualizing statistical relationships in data, including linear regression plots, distribution plots (e.g., histograms, kernel density estimates), joint plots, pair plots, and categorical plots.

Customization and styling: Seaborn provide extensive customization options to control the appearance of plots, including color palettes, plot styles, grid styles, and figure aesthetics. It also supports the use of themes to quickly change the overall look and feel of plots.

Integration with Pandas: Seaborn seamlessly integrates with Panda's data structures, allowing users to pass Data Frame objects directly to plotting functions. This makes it easy to work with structured data and create informative visualizations from data stored in Pandas Data Frames.

Support for complex visualizations: Seaborn supports the creation of complex visualizations such as multi-panel faceted plots (e.g., Facet Grid), cluster maps, pair grids, and joint kernel density plots, enabling users to explore and analyze high-dimensional datasets effectively.

Built-in dataset visualization: Seaborn includes several built-in datasets that can be easily loaded and visualized to explore the library's capabilities and demonstrate various plotting techniques.

5.1.4 MATPLOTLIB

Matplotlib is a comprehensive Python library for creating static, interactive, and animated visualizations. It is widely used for generating publication-quality plots across various domains, including scientific computing, data analysis, machine learning, and engineering [27].

Key features of Matplotlib include:

Wide range of plot types: Matplotlib supports a variety of plot types, including line plots, scatter plots, bar plots, histograms, pie charts, contour plots, surface plots, and 3D plots, allowing users to visualize data in different formats and dimensions.

Customization options: Matplotlib offers extensive customization options to control the appearance and layout of plots. Users can customize aspects such as colors, line styles, markers, labels, titles, fonts, axes limits, ticks, grids, and legends to create visually appealing and informative plots.

Multiple interfaces: Matplotlib provides multiple interfaces for creating plots, including a MATLAB-style scripting interface (pyplot), an object-oriented interface (OO API), and a procedural interface (pylab). This flexibility allows users to choose the interface that best suits their workflow and coding style.

Seamless integration: Matplotlib seamlessly integrates with other Python libraries and frameworks, including NumPy, Pandas, Seaborn, and SciPy. This integration enables users to easily visualize data stored in arrays, Data Frames, or other data structures and leverage the functionality of these libraries for data manipulation and analysis.

Export options: Matplotlib supports exporting plots to various file formats, including PNG, JPEG, PDF, SVG, EPS, and interactive formats such as HTML and interactive backends (e.g., Qt, GTK, Tkinter), allowing users to save plots for publication or further analysis.

Interactive plotting: Matplotlib provides support for interactive plotting through interactive backends such as Qt, GTK, and Tkinter, enabling users to interactively explore and manipulate plots using mouse events, keyboard shortcuts, and GUI controls.

5.1.5 SK-LEARN

Scikit-learn, often abbreviated as sklearn, is a popular machine learning library in Python that provides a simple and efficient toolset for data mining and data analysis tasks. It is built on top of other scientific computing libraries such as NumPy, SciPy, and matplotlib, and it integrates well with other Python libraries like Pandas [28].

Key features of scikit-learn include:

Consistent API: Scikit-learn offers a consistent and easy-to-use API, making it straightforward to implement machine learning models and workflows. The library follows a uniform interface for various tasks such as model training, prediction, evaluation, and parameter tuning.

Wide range of algorithms: Scikit-learn provides implementations of a vast array of machine learning algorithms, including supervised and unsupervised learning algorithms for classification, regression, clustering, dimensionality reduction, and model selection. It covers popular algorithms such as linear models, support vector machines, decision trees, random forests, k-nearest neighbors, k-means clustering, and more.

Model evaluation and selection: Scikit-learn includes tools for evaluating and comparing the performance of machine learning models using various metrics such as accuracy, precision, recall, F1-score, ROC curve, and confusion matrix. It also offers techniques for cross-validation, hyperparameter tuning, and model selection to optimize model performance.

Preprocessing and feature engineering: Scikit-learn provides a wide range of preprocessing techniques for data transformation, scaling, normalization, encoding categorical variables, handling missing values, and feature selection. These preprocessing

tools help prepare the data for modeling and improve the performance of machine learning algorithms.

Integration with other libraries: Scikit-learn seamlessly integrates with other Python libraries such as NumPy, SciPy, Pandas, and Matplotlib, enabling users to leverage their functionalities for data manipulation, visualization, and analysis in conjunction with scikit-learn's machine learning capabilities.

Extensibility and community support: Scikit-learn is an open-source project with an active community of developers and users. It encourages contributions from the community and provides extensive documentation, tutorials, and examples to support users in building machine learning solutions.

5.1.6 PICKLE

Pickle is a module in Python used for serializing and deserializing Python objects. It allows you to convert Python objects into a byte stream, which can be saved to a file or sent over a network, and then reconstructed back into Python objects when needed. Pickle is commonly used for saving trained machine learning models, caching expensive computations, or transferring data between Python processes [29].

Key features of Pickle include:

Serialization: Pickle provides functions to convert Python objects into a byte stream, which can be stored in a file or transmitted over a network. This process is called serialization.

Deserialization: Pickle also offers functions to reconstruct Python objects from a byte stream. This process is called deserialization.

Compatibility: Pickle is highly compatible with most Python data types and objects, including custom objects, nested data structures, and complex data types like lists, dictionaries, tuples, sets, and NumPy arrays.

Binary format: Pickle stores data in a binary format, which is compact and efficient for storage and transmission. However, the binary format is not human-readable, and it may not be compatible across different versions of Python.

Security considerations: Pickle's serialization format can execute arbitrary code during deserialization, which poses security risks if you unpickle data from untrusted sources. It's essential to use Pickle only with trusted data or employ security measures like validating data integrity and using safe unpickling techniques.

Alternatives: While Pickle is convenient for serializing and deserializing Python objects, it's not always the best choice for all use cases. For example, if you need human-readable data or interoperability with other programming languages, you might consider alternatives like JSON, CSV, or Protocol Buffers.

5.1.7 FLASK

Flask is a lightweight and flexible web framework for Python that provides tools and libraries to build web applications quickly and easily. It is designed to be simple and minimalist, allowing developers to create web applications with minimal boilerplate code. Flask is widely used for developing web applications, APIs, and microservices in Python [30].

Key features of Flask include:

Lightweight and minimalist: Flask is lightweight and has a minimalistic core, making it easy to learn and use for both beginners and experienced developers. It follows the "micro-framework" philosophy, focusing on simplicity and extensibility.

Built-in development server: Flask comes with a built-in development server, allowing developers to run and test their web applications locally without the need for additional server setup. This makes the development process fast and straightforward.

Routing: Flask provides a simple and intuitive routing system that maps URL paths to view functions. Developers can define routes using decorators or explicit route mapping, making it easy to define the URL structure of their web application.
Templating engine: Flask includes a powerful templating engine called Jinja2, which allows developers to generate dynamic HTML content by combining HTML templates with Python code. Jinja2 provides features such as template inheritance, macros, filters, and loops, making it easy to create reusable and maintainable templates.

Extension's ecosystem: Flask has a rich ecosystem of extensions that add additional functionality to the framework, such as authentication, database integration, form validation, RESTful APIs, and more. These extensions make it easy to extend Flask's capabilities and integrate with third-party libraries and services.

RESTful support: Flask provides built-in support for building RESTful APIs, making it easy to create web services that follow REST principles. Developers can use Flask's routing system and HTTP methods (GET, POST, PUT, DELETE) to define API endpoints and handle requests and responses.

Testability: Flask provides tools and libraries for testing web applications, including support for unit testing, integration testing, and end-to-end testing. Developers can use tools like Flask-Testing and Werkzeug's test client to write and run tests for their Flask applications.

5.1.8 ANGULAR

Angular is a popular open-source web application framework maintained by Google and a community of developers. It is used for building single-page web applications (SPAs) and dynamic web applications with rich user interfaces. Angular is written in TypeScript and follows the Model-View-Controller (MVC) architecture pattern [31].

Key features of Angular include:

Component-based architecture: Angular applications are built using components, which are reusable and encapsulated building blocks that represent different parts of the user interface. Components encapsulate the HTML, CSS, and TypeScript code related to a specific feature or functionality.

Two-way data binding: Angular provides two-way data binding, which means that changes to the model (data) are automatically reflected in the view (UI), and vice versa. This simplifies the development process by eliminating the need for manual DOM manipulation.

Dependency injection: Angular has a built-in dependency injection system that facilitates the management of dependencies between different components and services. Dependency injection promotes modular and testable code by enabling the injection of dependencies into components and services rather than hard-coding them.

Directives: Angular directives are special markers in the HTML that tell Angular to do something to a DOM element. Directives can be used to add behavior, manipulate the DOM, or apply styles dynamically. Angular provides built-in directives like ngIf, ngFor, ngSwitch, and allows developers to create custom directives as well.

Services: Angular services are singleton objects that encapsulate reusable business logic and functionality shared across components. Services can be injected into components and other services using Angular's dependency injection system, enabling code reuse and separation of concerns.

Routing: Angular provides a powerful routing system that allows developers to build SPAs with multiple views and navigation between them. Angular's router supports features such as nested routes, lazy loading, route guards, and parameterized routes, making it easy to create complex navigation flows.

Forms: Angular offers robust support for building and validating forms in web applications. Angular forms support both template-driven forms and reactive forms, allowing developers to choose the approach that best fits their needs. Angular forms provide features such as form controls, form validation, form submission, and error handling.

TypeScript support: Angular is built using TypeScript, a superset of JavaScript that adds static typing and other features to the language. TypeScript enhances developer productivity and code quality by providing features like type checking, code navigation, refactoring, and better tooling support.

5.2 USER INTERFACE

For user interfaces we use flask backend for building prediction API and build an angular application to show the results of the prediction through the API response. Some pictures of our user interfaces are given below from **figure 5.2.1 to figure 5.2.4.**

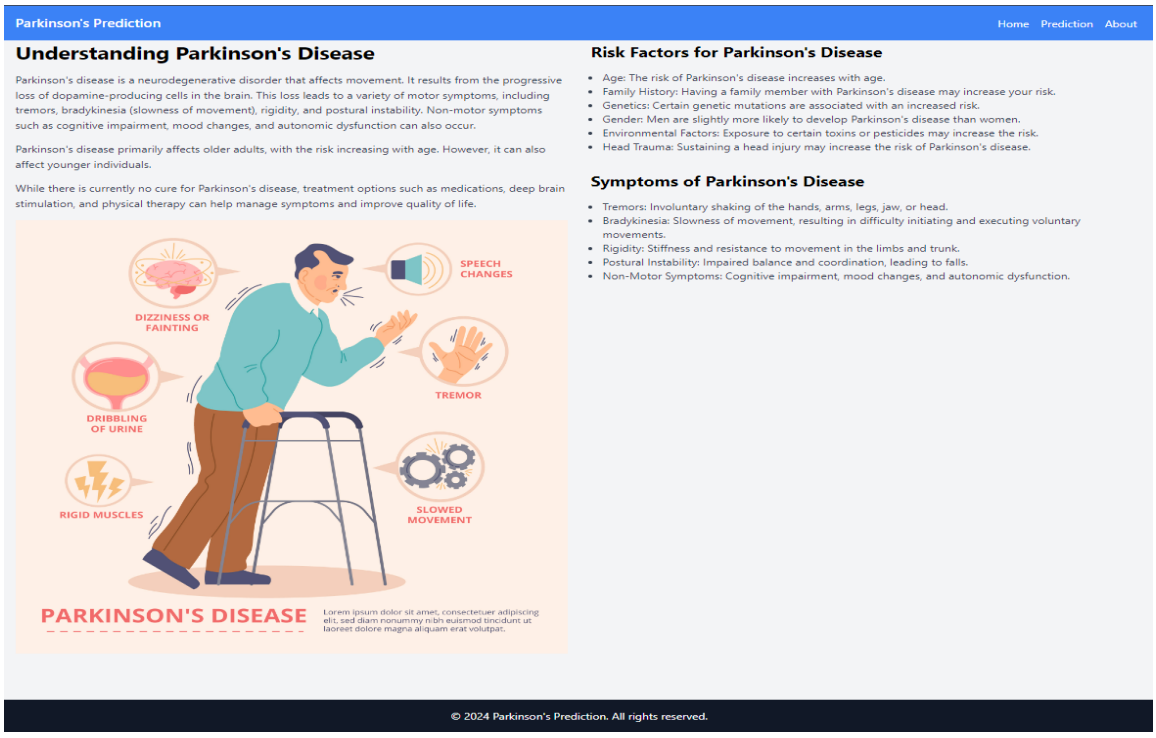

Figure 5.2.1: Homepage

Figure 5.2.2: About Page

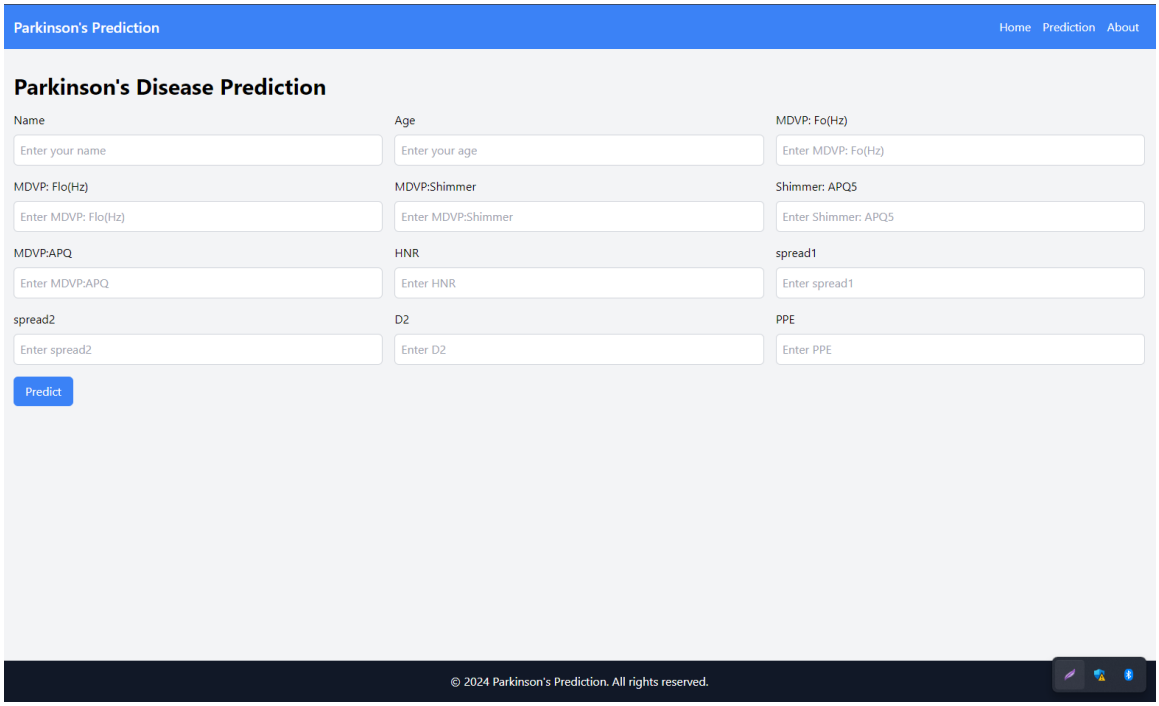

Figure 5.2.3: Prediction disease prediction form

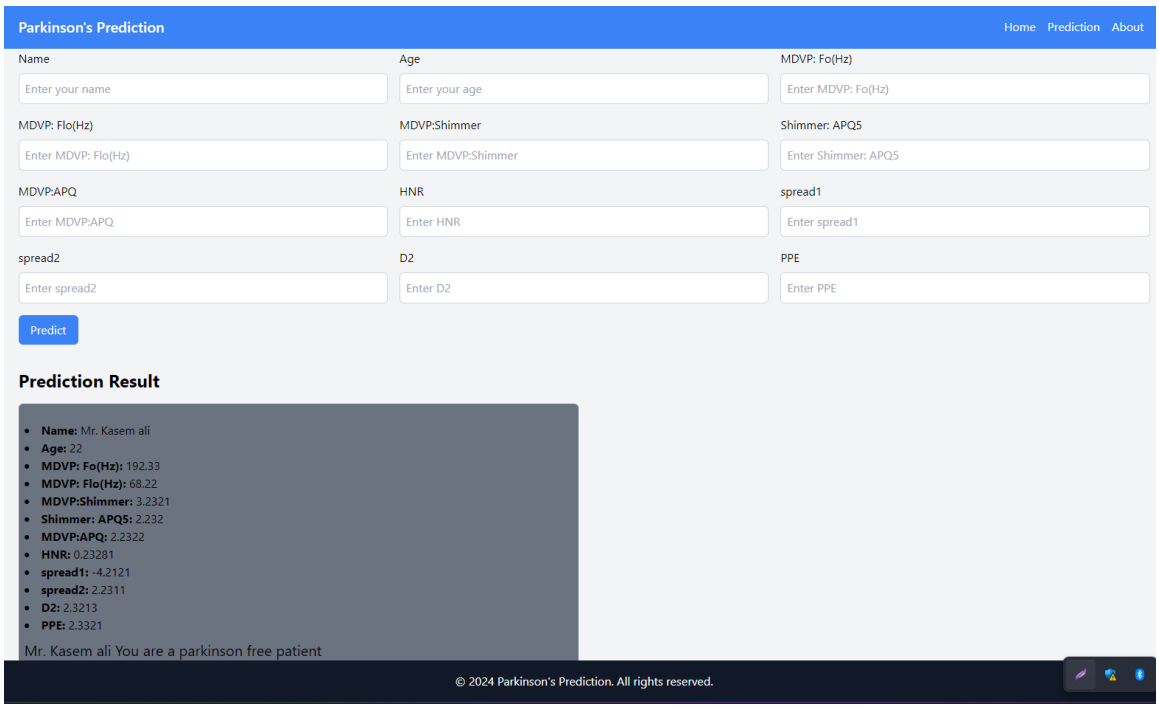

Figure 5.2.4: Result of Parkinson Disease prediction

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Our machine learning analysis for Parkinson's disease, complemented by a user-friendly interface, has yielded highly accurate and precise results in diagnosing and assessing the disease. The model, trained on a comprehensive dataset, showcased a notable capability for early detection, facilitating timely intervention. The user interface's intuitiveness and clarity enhance its practicality for healthcare professionals, allowing for easy data input and interpretation of results. While the model's performance is robust, ongoing improvements and validation on diverse datasets are crucial for real-world applicability. Overall, this integrated approach holds significant promise in advancing the diagnosis and management of Parkinson's disease, offering a valuable tool for healthcare practitioners.

6.2 FUTURE WORK

In future our machine learning analysis for Parkinson's disease, alongside its user-friendly interface, is promising and multifaceted. Further enhancements may include personalized medicine through the integration of individual patient characteristics and genetic data, enabling tailored treatment plans. Longitudinal monitoring capabilities could be expanded to track disease progression over time, facilitating early intervention and dynamic treatment adjustments. Integrating the model with wearable technology and telemedicine platforms would offer real-time data input and extend accessibility, particularly in remote or underserved areas. Collaboration with healthcare professionals, global data sharing, and improvements in model interpretability are essential for refining accuracy and fostering trust. Incorporating multi-modal data and pursuing regulatory approval contribute to the technology's evolution, making it a comprehensive and standardized tool in Parkinson's disease diagnosis and management, ultimately improving patient outcomes.

REFERENCES

[1] Meysam Asgari and Izhak Shafran "Predicting Severity of Parkinson's Disease from Speech," IEEE, 2010.

[2] R.Savitha, S. Suresh, and N. Sundararajan,"A Fully complex-valued radial basis function network and its learning algorithm," International Journal of Neural Systems 19(4), pp. 253-267, 2009.

[3] R. Savitha, S. Suresh, and N. Sundararajan, "Metacognitive Learning in a Fully Complex-valued Radial Basis Function Network," Neural Computation, vo1.24, no. 5, pp. 1297-1328, 2012.

[4] Rustempasic, Indira, and Can, M. (2013). Diagnosis of Parkinson's disease using Fuzzy C-Means Clustering and Pattern Recognition. South-east Europe Journal of Soft Computing, 2(1).

[5] Amit S., Ashutosh M., A. Bhattacharya, F. Revilla, (2014, March). Understanding Postural Response of Parkinson's Subjects Using Nonlinear Dynamics and Support Vector Machines. Austin J, Biomed Eng 1(1): id1005.

[6] L. Silveira-Moriyama, J. Carvalho Mde, R. Katzenschlager, A. Petrie, R. Ranvaud, E.R. Barbosa, A.J. Lees, The use of smell identification tests in the diagnosis of Parkinson's disease in Brazil, Mov Disord, 23 (2008) 2328-2334.

[7] Parkinson's disease detection using olfactory loss and REM sleep disorder features. R. Prashanth-EMBS Member, S. Dutta Roy-IEEE Member, and P. K. Mandal, S. Ghosh.

[8] Gonick, L., and Smith, W. (1993). The Cartoon Guide to Statistics. HarperPerennial

[9] Rumsey, D. J. (2017). Statistics for Dummies. John Wiley and Sons.

[10] Guyon, I., and Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of machine learning research, 3(Mar), 1157-1182.

[11] Bishop, C. M. (2006). Pattern recognition and machine learning. springer.

[12] Raschka, S., and Mirjalili, V. (2019). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2 (3rd ed.). Packt Publishing.

[13] Müller, A. C., and Guido, S. (2017). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media.

[14] Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

[15] Raschka, S., and Mirjalili, V. (2019). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2 (3rd ed.). Packt Publishing.

[16] Müller, A. C., and Guido, S. (2017). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media.

[17] Zhang, H. (2004). The Optimality of Naive Bayes. Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, 562-567.

[18] Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

[19] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press.

[20] Raschka, S., and Mirjalili, V. (2019). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2 (3rd ed.). Packt Publishing.

[21] Guyon, I., and Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of machine learning research, 3(Mar), 1157-1182.

[22] Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861-874.

[23] Jankovic, J., and Tolosa, E. (2015). Parkinson's Disease and Movement Disorders. Lippincott Williams and Wilkins.

[24] Travis E, Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006). ISBN 978-0986988100

[25] Wes McKinney. "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython." O'Reilly Media, 2nd edition, 2017. ISBN 978-1491957660 .

[26] Michael Waskom et al. "Seaborn: statistical data visualization." Journal of Open-Source Software, 6(60), 3021 (2021). https://doi.org/10.21105/joss.03021 Accessed on 17 February 2024.

[27] John D. Hunter. "Matplotlib: A 2D Graphics Environment." Computing in Science and Engineering, 9(3), 90-95 (2007). DOI: 10.1109/MCSE.2007.55.

[28] Pedregosa et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, 12, 2825-2830 (2011).

[29] Python documentation on Pickle: https://docs.python.org/3/library/pickle.html Accessed on 17 February 2024.

[30] Miguel Grinberg. "Flask Web Development: Developing Web Applications with Python." O'Reilly Media, 2nd edition, 2018. ISBN 978-1491991732.

[31] Deborah Kurata. "Angular 2+ Jumpstart with TypeScript." Pluralsight, 2016.