DIABETES AND PARKINSON'S PREDICTION SYSTEM USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

NITHIN U
ROHAN CHAKARAVARTHI V
SHARAN SHAKTHI G
MUHILAN P

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

DECEMBER, 2023

DIABETES AND PARKINSON'S PREDICTION SYSTEM USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

NITHIN U	**(811720104071)**
ROHAN CHAKARAVARTHI V	**(811720104085)**
SHARAN SHAKTHI G	**(811720104092)**
MUHILAN P	**(811720104067)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

DECEMBER, 2023

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"DIABETES AND PARKINSON'S PREDICTION SYSTEM USING MACHINE LEARNING"** is the bonafide work of **NITHIN U (811720104071), ROHAN CHAKARAVARHI V (811720104085), SARAN SHAKTHI G (811720104092) AND MUHILAN P (811720104067)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr.A. Delphin Carolina Rani ME.,	Mrs.V. Kalpana B.Tech., M.E., (Ph.D).,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of CSE	ASSISTANT PROFESSOR
K. Ramakrishnan College of Technology	Department of CSE
(Autonomous)	K. Ramakrishnan College of Technology
Samayapuram – 621 112	(Autonomous)
	Samayapuram – 621 112

Submitted for the viva-voce examination held on ………………

INTERNAL EXAMINER

DECLARATION

We jointly declare that the project report on **“DIABETES AND PARKINSON'S PREDICTION SYSTEM USING MACHINE LEARNING”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF ENGINEERING**.

Signature

____

NITHIN U

____

ROHAN CHAKARAVARTHI V

____

SHARAN SHAKTHI G

____

MUHILAN P

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

We are glad to credit honorable chairman **Dr. K.RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA., Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

We whole heartily thanks to **Dr.A. DELPHIN CAROLINA RANI ME.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing his encourage pursuing this project.

I express my deep and sincere gratitude to my project guide

Mrs.V. KALPANA B.TECH., M.E., (Ph.D)., Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated me to carry out this project.

I render my sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express my special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

Diabetes and Parkinson’s disease prediction Application is a comprehensive approach to addressing the challenges associated with diabetes and Parkinson's disease by leveraging extensive data and advanced feature engineering techniques. The primary focus is on reducing consultant charges for patients through the development of highly accurate disease diagnosis models. Rigorous evaluation and validation methods ensure the reliability of predictions, contributing to improved patient outcomes and overall quality of life. The user-friendly deployment of the prediction system is a key highlight, promising accessibility to healthcare professionals and individuals concerned about these diseases. With Support Vector Machines as our chosen model, this documentation provides a detailed account of the training process and the steps taken to make our system intuitive and effective for early detection and personalized healthcare.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ABSTRACT	**iv**
	LIST OF TABLES	**vii**
	LIST OF FIGURES	**viii**
	LIST OF ABBREVIATIONS	**ix**
1	**Introduction**	
	1.1 Purpose	1
	1.2 Problem Description	1
	1.3 Overview	2
	1.4 Scope	2
	1.5 Aim and Objectives	2
2	**General Description**	
	2.1 User Needs	4
	2.2 Assumption and Dependencies	4
	2.2.1 Manipulation Phases	4
	2.2.2 Application Dependencies	5
	2.3 Working Principle	5
3	**System Features and Requirements**	
	3.1 Integrated System	7
	3.2 Theory of Operation	7
	3.3 Platform Structure	7
	3.4 Features	9
4	**Interface Requirement**	
	4.1 Dataset Tracking	11
	4.2 Flowchart	11
	4.3 Assembling Widgets	12

	4.4 Advantage and Application	13
	4.4.1 Advantages	13
	4.4.2 Drawbacks of Existing System	14
	4.4.3 Application	15
5	**Algorithmic Implementation**	
	5.1 Random Forest Algorithm	18
	5.2 Support Vector Machine Algorithm	19
	5.3 SVM VS Random Forest Algorithm	20
6	**Model Development**	
	6.1 Technology Used	21
	6.2 SVM Classifier	22
	6.3 Frontend Development	23
	6.4 Backend Development	24
7	**Sample Result**	
	7.1 Expected Output	27
	7.2 Project Output	28
	7.3 Future Update	29
	7.4 Modification	30
8	**Conclusion**	
	8.1 Achievements	32
	8.2 Further Development	33
	8.3 Application	34
	APPENDIX I	35
	APPENDIX II	39
	References	40

LIST OF TABLES

Table No.	Table Title	Page No.
5.1	SVM vs Random Forest	19
5.2	Algorithm Accuracy and Score	20
7.1	Diabete’s Classification	27
7.2	Parkinson’s Classification	28

LIST OF FIGURES

Figure No.	Figure Title	Page No.
3.1	Integeration Model	8
4.1	Model Flowchart	11
5.1	Random Forest Trained Model	18
6.1	Streamlit Widgets	23
7.1	SAV Model	30
A2.1	Diabete’s Report	39
A2.2	Parkinson’s Report	39

LIST OF ABBREVIATIONS

IDE	Integrated Development Environment
SDK	Software Development Kit
API	Application Programming Interface
UI	User Interface
SVM	Support Vector Machine
MLR	Machine Learning Regression
SKL	scikit-learn
CSV	Comma-Separated Values
AML	Advance Machine Learning

CHAPTER 1

INTRODUCTION

1.1 PURPOSE

Diabetes and Parkinson’s Prediction Application endeavors to enhance healthcare accessibility and empower individuals through the development of a Streamlit application focused on predicting the likelihood of diabetes and Parkinson's disease. By harnessing machine learning models, the initiative aims to enable early detection, allowing for timely intervention and personalized healthcare strategies. The user-friendly interface of the application facilitates easy access to predictive insights, fostering a sense of control over one's health. Emphasizing data-driven health management, the project seeks to promote informed decision-making and proactive measures for individuals at risk. Ultimately, by providing a tool for community health support, the project strives to contribute to a more resilient and empowered society, reducing the strain on healthcare systems through early identification and preventive care.

1.2 PROBLEM DESCRIPTION

In the realm of healthcare, there exists a critical need for accessible and proactive tools that empower individuals in the early identification of potential health risks. Recognizing the rising prevalence of diabetes and Parkinson's disease, there is a pronounced gap in user-friendly applications that integrate predictive modeling to offer personalized insights. The current healthcare landscape often lacks efficient means for individuals to assess their susceptibility to these conditions, hindering early detection and preventive measures. Consequently, the project aims to address this gap by developing a Streamlit application that leverages machine learning models to predict the likelihood of diabetes and Parkinson's disease, providing users with a user-friendly tool for informed decision-making and proactive health management.

In the face of escalating health challenges, the lack of accessible and user-friendly tools for early disease prediction has emerged as a significant problem. Diabetes and Parkinson's disease, both widespread and impactful conditions, pose substantial threats to public health. Current healthcare systems often struggle to offer proactive solutions that empower individuals to gauge their susceptibility to these ailments. The absence of streamlined applications incorporating predictive models further hampers the timely identification of potential health risks, hindering the implementation of preventive measures. This project seeks to address this critical problem by developing a specialized Streamlit application, providing a solution that bridges the gap between predictive modeling and user accessibility. By offering a platform for early risk assessment, the application aims to empower individuals to take informed actions towards safeguarding their health and fostering a proactive approach to well-being.

1.3 OVERVIEW

The project focuses on the development of a Streamlit application designed to predict the likelihood of diabetes and Parkinson's disease through machine learning models. With a growing global health concern related to these conditions, there is a pressing need for user-friendly tools that facilitate early detection and empower individuals to make informed decisions about their health.

Leveraging predictive modeling, the application aims to offer personalized insights, enabling users to assess their susceptibility to these diseases. The streamlined interface provided by Streamlit enhances accessibility, making the tool available to a broad user base. Through this project, we aim to contribute to proactive healthcare management, emphasizing early intervention, and fostering a culture of health empowerment within communities. The integration of machine learning and user-friendly design aligns with the broader goal of promoting data-driven, preventative healthcare strategies, ultimately enhancing the well-being of individuals and communities alike.

1.4 SCOPE

The scope of this project encompasses the development and deployment of a Streamlit application for predicting the likelihood of diabetes and Parkinson's disease based on machine learning models.

- The application will be designed to accept user inputs related to relevant health parameters, utilizing trained models to provide personalized predictions. The primary focus lies in creating a user-friendly interface to enhance accessibility for a diverse audience.
- The project will involve the exploration and utilization of suitable datasets for training and validating the predictive models. While the initial release will cover the basic predictive functionality, future iterations may include refinements, additional features, and the potential integration of more advanced machine learning techniques.
- The deployment of the application will be considered, ensuring accessibility for users interested in assessing their health risks. The project scope does not extend to replacing professional medical advice but aims to complement existing healthcare systems by fostering awareness, early detection, and encouraging proactive health management.

1.5 AIM AND OBJECTIVE

The primary aim of this project is to develop a user-friendly Streamlit application that utilizes machine learning models for predicting the likelihood of two prevalent health conditions – diabetes and Parkinson's disease.

The overarching goal is to empower individuals with a tool that enables early detection, fostering a proactive approach to health management. The primary aim of this project is to create a user-friendly and accessible Streamlit application incorporating machine learning models to predict the probability of two prevalent health conditions—diabetes and Parkinson's disease. Through the development of accurate predictive models trained on relevant datasets, the application seeks to empower individuals with the ability to assess their potential health risks. Here with some formal objective of the project to enhance the model and development.

Model Development: Train accurate machine learning models for predicting the probability of diabetes and Parkinson's disease using relevant and well-curated datasets.

Streamlit Interface: Create an intuitive and accessible interface using Streamlit to allow users to input their health parameters and receive personalized predictions.

User Empowerment: Provide users with actionable insights, encouraging informed decision-making regarding their health and potential risk factors associated with diabetes and Parkinson's disease.

Data Security and Privacy: Implement robust measures to ensure the security and privacy of user data, adhering to ethical standards and regulations.

Application Deployment: Deploy the Streamlit application on a suitable platform, making it readily available to users for self-assessment and risk prediction.

Feedback Mechanism: Integrate a feedback mechanism to gather user input, enabling continuous improvement and refinement of the application based on user experiences and needs.

Community Impact: Assess the potential impact of the application on community health, emphasizing its role in early detection, awareness, and contributing to a culture of proactive health management.

By achieving these objectives, the project aims to provide a valuable tool that not only predicts health risks but also educates and empowers individuals to take proactive steps towards a healthier lifestyle.

CHAPTER 2

GENERAL DESCRIPTION

2.1 USER NEEDS

Users, in their quest for proactive health management, express a fundamental need for a tool that facilitates early detection of health risks, particularly in the context of prevalent conditions like diabetes and Parkinson's disease. Central to their requirements is the demand for an accessible platform, characterized by a user-friendly interface that accommodates individuals with diverse technical proficiencies.

Additionally, users emphasize the significance of personalization, seeking insights tailored to their unique health profiles to better comprehend and address potential risks. A paramount concern is the assurance of data security, underscoring the necessity for stringent measures to protect their personal health information and instill confidence in engaging with the predictive application.

Furthermore, users express a desire for the integration of educational resources alongside predictions, fostering a deeper understanding of the underlying factors contributing to health risks and empowering them to make well-informed decisions about their well-being.

2.2 ASSUMPTION AND DEPENDENCIES

The project operates under the assumption that high-quality and representative datasets for training machine learning models on diabetes and Parkinson's disease are readily available. It also relies on users providing accurate health parameter inputs for effective predictions. Dependencies on stable versions of Streamlit and machine learning libraries, as well as the availability of relevant data, underscore the project's reliance on external factors for successful development and deployment.

2.2.1 Manipulation Phases:

Quality of Data: The project assumes the availability of high-quality and representative datasets for training machine learning models on diabetes and Parkinson's disease. The accuracy of predictions heavily relies on the quality and relevance of the data.

Model Generalization: It is assumed that the trained machine learning models will generalize well to new and unseen data. The performance of the models in real-world scenarios depends on their ability to adapt beyond the dataset.

User Input Accuracy: The accuracy of predictions is contingent on users providing accurate and truthful input regarding their health parameters. Assumptions are made about users' ability and willingness to provide reliable information.

2.2.2 Application Dependencies

The application is dependent on the stable functioning of Streamlit for creating an intuitive user interface. Additionally, it relies on machine learning libraries such as scikit-learn for accurate model training and predictions. Some required libraries are

Streamlit Library: The project is dependent on the stability and functionality of the Streamlit library for creating the user interface. Changes or issues with the Streamlit library may impact the application's performance.

Machine Learning Libraries: Dependencies on machine learning libraries such as scikit-learn for model training and predictions. Any updates or changes to these libraries may affect the project.

Data Availability: The project is dependent on the availability of relevant and up-to-date datasets for training the machine learning models. Delays or limitations in obtaining suitable data may impact the project timeline.

External Deployment Platforms: If the project is deployed on external platforms (e.g., Heroku, AWS), its functionality may depend on the reliability and availability of these platforms.

User Accessibility: The success of the project relies on users having access to the internet and devices capable of running the Streamlit application. Assumptions are made about users' digital accessibility and literacy.

2.3 WORKING PRINCIPLE

The working principle of the diabetes and Parkinson's disease prediction application involves several steps, including data preprocessing, model training, and creating a Streamlit interface for user interaction. Let's break down the key components:

1. Data Gathering:

Datasets for diabetes and Parkinson's disease are collected. These datasets should include relevant features (input variables) and target labels (output variable).

2. Data Preprocessing:

Handle any missing data: Remove or impute missing values.

Encode categorical variables: Convert categorical data into a numerical format suitable for machine learning models. Normalize or standardize numerical features: Ensure that numerical features have a consistent scale.

3. Model Training:

The application uses Support Vector Machine (SVM) models for both diabetes and Parkinson's disease prediction. The datasets are split into training and testing sets to evaluate the model's performance. For each dataset, a SVM model is trained using the training set. Feature scaling, such as standardization, is often applied to ensure that features have similar scales, which can be important for SVM.

4. Streamlit Application Setup:

The Streamlit library is used to create a user-friendly interface for the application. The application script (app.py) is created, which includes the necessary imports and functions.

5. Streamlit Application Code:

The Streamlit app includes sections forImporting necessary libraries and modules. Defining functions for training SVM models for diabetes and Parkinson's disease. Creating the main function (main()) that initializes the Streamlit app.

Running the app (if__name__== "__main__": main()).

6. User Interface Enhancement:

The Streamlit app includes input fields for users to input their data (features) for diabetes and Parkinson's prediction. The app can display the accuracy of the models and may also provide predictions for the user's input.

7. Running the Streamlit App:

The app is run using the streamlit run app.py command in the terminal.

This opens a local development server, and users can interact with the app through a web browser.

8. Deployment (Optional):

The app can be deployed to a hosting platform, allowing users to access it online. Common hosting platforms include Heroku, Streamlit Sharing, or any platform that supports Streamlit apps.

9. Continuous Improvement:

The application can be enhanced further by adding more features, improving the user interface, and incorporating user feedback. Model performance can be continuously monitored, and the application can be updated with new models or additional functionalities.

This working principle outlines the general flow of creating a diabetes and Parkinson's disease prediction application using SVM and Streamlit. The specific implementation details may vary based on the characteristics of your datasets and the requirements of your application.

CHAPTER 3

SYSTEM FEATURES AND REQUIREMENTS

3.1 INTEGRATED SYSTEM

The integrated system of this project consists of a cohesive framework that combines machine learning models and the Streamlit application to offer a seamless user experience. The machine learning component includes trained models for predicting the likelihood of diabetes and Parkinson's disease, developed through the analysis of relevant datasets.

These models are integrated into the Streamlit interface, forming a user-friendly platform where individuals can input their health parameters. The Streamlit application processes these inputs, leverages the machine learning models for predictions, and delivers personalized insights. The integrated system, therefore, harmonizes predictive analytics with a user-centric interface, aiming to empower users in the early detection and proactive management of potential health risks.

The computing infrastructure where the application is hosted. This could be a local machine or a cloud server, depending on deployment choices. Components capable of running machine learning models efficiently, considering the computational requirements of SVM.

Machine Learning Libraries: Libraries such as scikit-learn for SVM model training and prediction.

Streamlit: A Python library for creating web applications with interactive user interfaces.

Data Preprocessing Tools: Tools for preparing and cleaning datasets before feeding them into the SVM models.

Web Server: A web server is needed to host the Streamlit application.

3.2 THEORY OF OPERATION

The theory of operation of a smart attendance application revolves around automating the attendance tracking process and providing an intuitive user experience. Once attendance is marked, the application captures the data, securely stores it, and updates attendance records in real-time. It may also perform data analysis to generate reports, insights, and notifications. Integration with other systems ensures seamless data flow and interoperability.

3.3 PLATFORM STRUCTURE

The platform structure of a smart attendance application typically includes the following components:

• **User Interface:** The user interface provides an intuitive and user-friendly interface for users interact with the application. It includes features for authentication, attendance marking, and accessing attendance-related information.

• **Attendance Management Module:** This module handles the core functionality of attendance tracking. It captures attendance data, securely stores it in a database and updates attendance records in real-time.

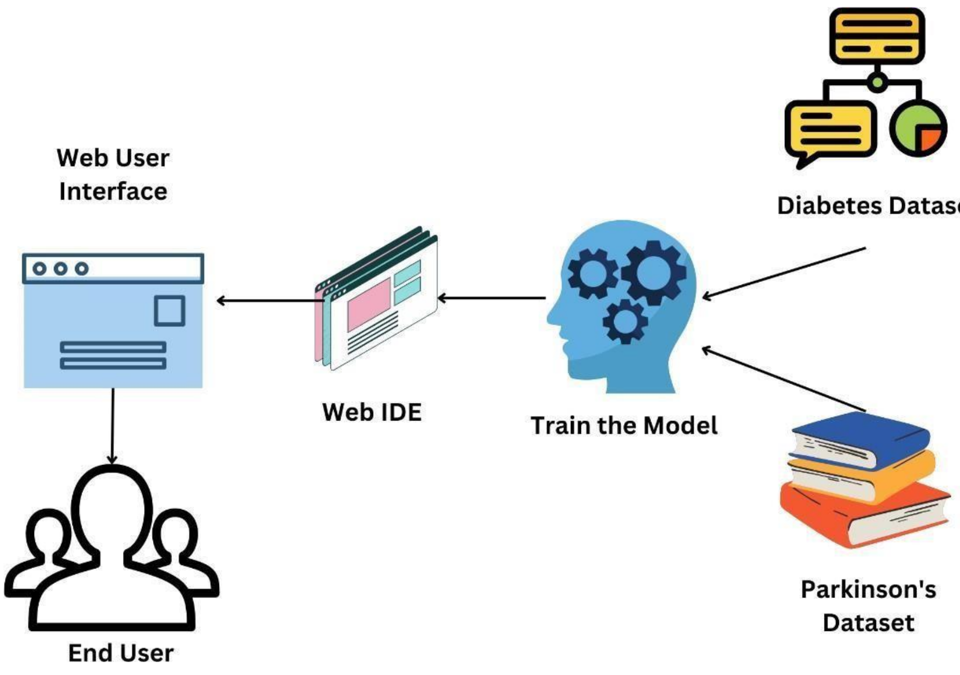


Figure 3.1 Integration Model

• **Reporting and Analytics Module:** This module generates comprehensive reports, analytics, and visualizations based on the attendance data. It allows administrators to gain insights into attendance patterns, trends, and overall attendance performance.

• **Integration Module:** This module enables seamless integration with other systems, such as student management or HR systems. It ensures data exchange, synchronization, and interoperability between the attendance application and existing systems

The platform structure of an Android app refers to the underlying framework and components that enable the development and functioning of the application on the Android operating system. The structure of an Android app platform can be summarized as follows: Android Operating System: The foundation of the platform structure is the Android operating system itself. It provides the necessary infrastructure for app execution, manages hardware resources, and provides core functionalities such as process management, security, and device drivers.

Android SDK: The Android Software Development Kit (SDK) is a collection of tools, libraries, and APIs that developers use to create Android applications. It includes the Android Debug Bridge (ADB), emulator, build tools, and various other utilities necessary for app development.

Java or Kotlin Programming Language: Android app development primarily relies on the Java programming language. With the introduction of Kotlin, developers have an alternative language choice. Both languages are officially supported and offer extensive libraries and frameworks for building robust Android apps.

Android Runtime (ART): ART is the runtime environment in which Android apps run. It compiles the app's bytecode into machine code, optimizing performance and resource utilization. ART replaces the older Dalvik runtime, offering improved efficiency and responsiveness.

Android Framework: The Android framework provides a comprehensive set of APIs and libraries that simplify app development. It includes components for user interface design (e.g., activities, fragments, views), data storage (e.g., SQLite database, Content Providers), connectivity (e.g., Wi-Fi, Bluetooth), multimedia (e.g., audio, video), and more.

Gradle Build System: Android apps use the Gradle build system to automate the compilation, packaging, and dependency management processes. Gradle simplifies the build configuration, manages external .
Material Design Guidelines: Google's Material Design guidelines provide a visual language and design principles for Android apps. Following these guidelines helps create a consistent and intuitive user interface, enhancing the overall user experience.

Support Libraries: Android support libraries offer backward compatibility for older Android versions, enabling developers to leverage newer features on a wider range of devices. These libraries provide additional functionalities and UI components not available in the base Android framework.

Google Play Services: Android apps can integrate with Google Play Services, a collection of APIs that offer access to various Google services, such as Google Maps, Google Drive, Google Analytics, and Firebase.

Android Studio IDE: Android Studio is the primary integrated development environment for Android app development. It offers a rich set of features, including code editing, debugging, and performance analysis tools, along with seamless integration with the Android SDK and Gradle build system.

These components and frameworks collectively form the platform structure for developing Android apps. They provide developers with a powerful ecosystem to create feature-rich, performant, and visually appealing applications for the Android platform.

3.4 FEATURES

Real-time Updates: The application provides real-time updates on attendance records, allowing administrators to monitor attendance status instantly.

Reporting and Analytics:

The application generates comprehensive reports and analytics, offering insights into
attendance patterns, trends, and overall attendance performance.

Notifications and Alerts:

The application sends notifications and alerts to relevant stakeholders, informing them about attendance status, late arrivals, absences, or other relevant updates.

Integration with Existing Systems:

The application seamlessly integrates with other systems, such as student management or HR systems, to ensure efficient data exchange and synchronization.

Mobile Accessibility:

The application can have a mobile component, allowing users to mark attendance using their smartphones and access attendance-related information on the go.

Customizable Settings: The application allows administrators to customize settings, such as attendance policies, notification preferences, or reporting formats, according to their specific requirements.

By incorporating these features, the smart attendance application provides a comprehensive and efficient solution for attendance. Android apps encompass a wide range of features that contribute to their functionality, usability, and user experience. Here's a paragraph summarizing some key features commonly found in Android apps:

Android apps offer a plethora of features that enhance their versatility and user appeal. First and foremost, they provide a visually engaging user interface with customizable layouts, intuitive navigation, and support for diverse screen sizes and resolutions. They leverage various hardware components of Android devices, such as cameras, GPS, accelerometers, and sensors, enabling features like image capture, location-based services, motion detection, and augmented reality.

Android apps also benefit from extensive connectivity options, allowing seamless integration with online services, social media platforms, and cloud storage. They provide support for push notifications, enabling real-time updates and user engagement. Furthermore, Android apps employ data management techniques, including local and remote data storage, synchronization, and offline capabilities.

CHAPTER 4

INTERFACE REQUIREMENT

4.1 DATASET TRACKING

Dataset tracking is a crucial aspect of machine learning workflows, especially when building prediction models like the diabetes and Parkinson's disease prediction system. Keeping track of datasets helps ensure transparency, reproducibility, and accountability in the machine learning pipeline. Here's how dataset tracking can be implemented:

4.2 FLOWCHART

Creating a flowchart for a diabetes and Parkinson's disease prediction system involves visually representing the sequence of steps and decision points in the application. Below is a simplified flowchart outlining the key stages from data input to prediction in a Streamlit application using SVM models. Keep in mind that this is a high-level overview, and you may need to customize it based on your specific implementation and requirements.

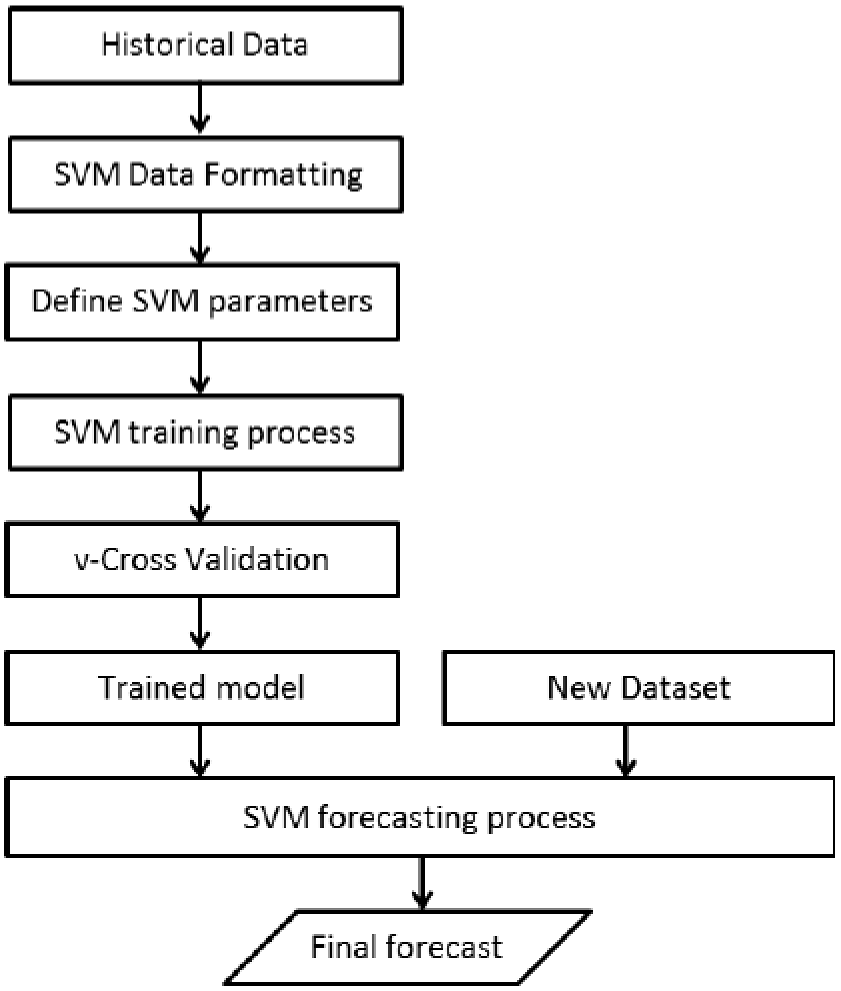


Figure 4.1 Model FlowChart

4.3 ASSEMBLING WIDGETS

In a Streamlit application, you can assemble widgets to create a user interface that allows users to interact with your machine learning models. Widgets are elements such as sliders, input boxes, buttons, and text areas that enable users to input data and receive output. Below is an example of how you might assemble widgets in a Streamlit app for a diabetes and Parkinson's disease prediction system using Support Vector Machine (SVM) models.

Snippet:

```
import streamlit as st
from sklearn import svm
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_diabetes, load_parkinsons
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load pre-existing datasets
diabetes_data = load_diabetes()
parkinsons_data = load_parkinsons()

# Widget for selecting the disease
selected_disease = st.selectbox("Select Disease", ["Diabetes", "Parkinson's"])

# Widget for user input
if selected_disease == "Diabetes":
    st.header("Diabetes Prediction")
    user_input = st.text_input("Enter Diabetes Features (Separated by Commas):")
else:
    st.header("Parkinson's Prediction")
    user_input = st.text_input("Enter Parkinson's Features (Separated by Commas):")

# Process user input and make predictions
if st.button("Predict"):
    if selected_disease == "Diabetes":
        # Process diabetes user input
        features = [float(val.strip()) for val in user_input.split(',')]
        # Train SVM model (similar to previous code)
        # ...

        # Make prediction
        prediction = model.predict([features])
        st.write("Diabetes Prediction:", prediction)

    else:
        # Process Parkinson's user input
        features = [float(val.strip()) for val in user_input.split(',')]
        # Train SVM model (similar to previous code)
        # ...

        # Make prediction
        prediction = model.predict([features])

        st.write("Parkinson's Prediction:", prediction)

# Display additional information or results
```

This example code demonstrates the use of Streamlit widgets like st.selectbox for disease selection, st.text_input for entering user data, and st.button for triggering predictions. The actual SVM model training and prediction code would be similar to what you have implemented before. Note that this is a simplified example, and you might want to enhance it with error handling, visualizations, and additional user interface elements based on your specific requirements.

4.4 ADVANTAGE AND APPLICATION

The advantage of having a mobile platform for a diabetes and parkinson's disease prediction system is that it allows users to reliable using the application. The mobile and web platform offers the following benefits and applications:.

4.4.1 ADVANTAGES

The advantages of a smart attendance application, particularly when implemented on a web platform, include:

High Accuracy: SVMs are known for their ability to achieve high accuracy in classification tasks, making them suitable for disease prediction where precision is crucial.

Effective in High-Dimensional Spaces: SVMs perform well even in high-dimensional feature spaces, making them suitable for datasets with numerous features, common in medical datasets.

Versatility: SVMs can be used for both classification and regression tasks. In disease prediction, they are commonly employed for binary classification problems (e.g., healthy vs. diseased).

Robustness to Overfitting: SVMs are less prone to overfitting, which is beneficial when dealing with medical data where overfitting could lead to inaccurate predictions on new, unseen data.

Kernel Trick: SVMs can use the kernel trick to transform input data into higher-dimensional spaces, allowing them to handle complex relationships between features.

Global Optimality: SVMs aim to find the hyperplane that maximally separates classes, providing a solution that is globally optimal.

Handling Nonlinear Relationships: SVMs can effectively model nonlinear relationships between features, making them suitable for capturing complex patterns in medical datasets.

4.4.2 DRAWBACKS OF EXISTING SYSTEM

While Support Vector Machine (SVM) models have various advantages, they also come with certain drawbacks and limitations, and these may be considered as drawbacks of the existing system when SVMs are used for disease prediction:

Sensitivity to Noise and Outliers:

SVMs can be sensitive to noisy data and outliers, potentially leading to suboptimal performance. Proper data preprocessing and outlier handling are crucial.

Computational Complexity:

Training an SVM model, especially on large datasets, can be computationally intensive and time-consuming. This may become a limitation in scenarios where real-time predictions are required.

Difficulty with Large Datasets:

SVMs may become impractical for very large datasets, both in terms of training time and memory requirements. This makes them less suitable for big data applications.

Parameter Sensitivity:

SVMs have parameters like the choice of the kernel and regularization parameter (C). The performance of the model can be sensitive to these parameter choices, and finding optimal values can require careful tuning.

Interpretability:

SVM models, especially when using complex kernels, might lack interpretability. Understanding the rationale behind a specific prediction can be challenging compared to more interpretable models like decision trees.

Binary Classification:

SVMs are inherently binary classifiers. Extending them to multi-class problems often involves using strategies like one-vs-one or one-vs-all, which may not always be straightforward.

Lack of Probability Estimates:

SVMs do not provide direct probability estimates. While there are techniques to approximate probabilities, obtaining reliable probability estimates may require additional steps.

Challenges with Imbalanced Datasets:

SVMs may struggle with imbalanced datasets where one class significantly outnumbers the other. Proper techniques, such as class weighting or resampling, may be necessary.

Limited Handling of Non-Linear Relationships:

While SVMs can handle non-linear relationships through kernel tricks, their performance may be limited compared to more advanced non-linear models like deep neural networks.

Black-Box Nature:

SVM models, especially when using complex kernels, can be considered as black-box models, meaning it may be challenging to interpret the internal decision-making process.

Need for Feature Scaling:

SVMs are sensitive to the scale of input features. Feature scaling is often required to ensure that all features contribute equally to the model, and this can be a drawback in scenarios where feature scales vary widely.

It's important to consider these drawbacks when choosing an algorithm for a specific disease prediction task. Depending on the characteristics of the data and the specific requirements of the application, alternative machine learning algorithms might be considered to address some of these limitations.

4.4.3 APPLICATIONS

Support Vector Machines (SVMs) find applications in various fields due to their effectiveness in classification and regression tasks. Here are some diverse applications where SVMs are commonly used:

Text and Document Classification:

SVMs are widely used for text and document classification tasks, such as spam detection, sentiment analysis, and topic categorization.

Image Classification and Recognition:

SVMs are employed in image classification tasks, including facial recognition, object detection, and image categorization.

Bioinformatics and Genomics:
SVMs are applied in bioinformatics for tasks such as protein structure prediction, gene expression classification, and identifying biomarkers in genomic data.

Medical Diagnosis:
SVMs are utilized in medical diagnosis for disease prediction, including diabetes prediction, cancer diagnosis, and identifying neurological disorders.

Handwriting Recognition:
SVMs are used for handwriting recognition in applications such as digit recognition and signature verification.

Speech Recognition:
SVMs contribute to speech recognition systems, helping classify and recognize spoken words.

Finance and Stock Market Prediction:
SVMs are applied in finance for tasks like stock price prediction, credit scoring, and fraud detection.

Remote Sensing:
SVMs are used in remote sensing applications for tasks like land cover classification, vegetation analysis, and satellite image interpretation.

Face Detection:
SVMs are employed in face detection algorithms, which are used in applications like security surveillance, access control systems, and photography.

Quality Control in Manufacturing:
SVMs are used for quality control in manufacturing processes, helping identify defective products based on sensor data.

Marketing and Customer Relationship Management (CRM):
SVMs contribute to customer segmentation, churn prediction, and personalized marketing in CRM applications.

Network Intrusion Detection:
SVMs are applied in cybersecurity for intrusion detection, helping identify and prevent malicious activities in computer networks.

Anomaly Detection:
SVMs are used for anomaly detection in various domains, including fault detection in industrial processes and identifying unusual patterns in data.

Human Activity Recognition:
SVMs contribute to recognizing and classifying human activities based on sensor data, such as accelerometers in smartphones.

Geographical Data Analysis:
SVMs are applied in geographical information systems (GIS) for tasks like land use classification, environmental modeling, and spatial data analysis.

Drug Discovery:
SVMs are utilized in drug discovery for tasks such as virtual screening and predicting the bioactivity of chemical compounds.

These applications demonstrate the versatility of SVMs in solving a wide range of problems across different domains. However, the choice of algorithm depends on the specific characteristics of the data and the requirements of the task at hand.

CHAPTER 5

ALGORITHMIC IMPLEMENTATION

5.1 RANDOM FOREST ALGORITHM

Random Forest is an ensemble learning algorithm known for its robustness and high accuracy across various machine learning tasks. The algorithm constructs a multitude of decision trees during training, each utilizing a subset of the training data and a random subset of features. Through a process called bagging, where trees are trained on bootstrap samples of the data, Random Forest mitigates overfitting by combining the predictions of multiple trees.

Furthermore, the random selection of features at each node enhances the diversity among trees, making the model resilient to noisy data and capable of handling high-dimensional datasets. In classification tasks, the algorithm uses a voting mechanism where each tree "votes" for a class, and the class with the majority of votes is the final prediction. For regression, it averages the predictions of all trees, resulting in a robust and accurate model.

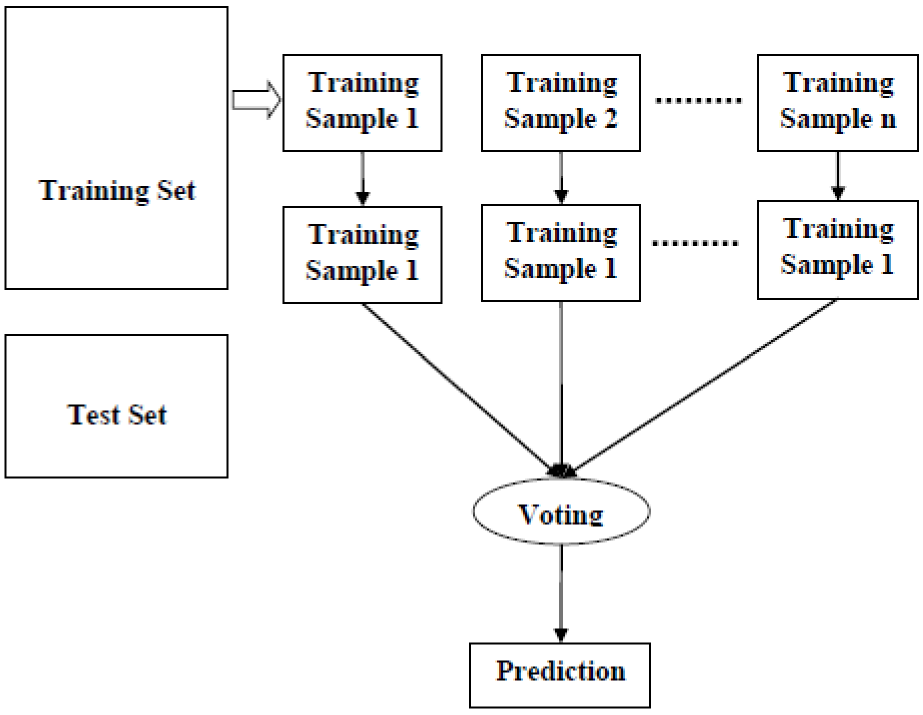


Figure 5.1 Random Forest Trained Model

While Random Forest offers significant advantages, such as high accuracy and robustness, it comes with computational complexity, especially when dealing with large datasets. The algorithm's ensemble nature and the creation of multiple trees can be resource- intensive, leading to increased memory usage and training time. Additionally, the resulting model is often considered a black-box, limiting its interpretability compared to simpler models. Despite these drawbacks, Random Forest remains a popular choice in machine learning, particularly when the emphasis is on predictive performance and handling diverse and complex datasets.

5.2 SUPPORT VECTOR MACHINE ALGORITHM

Support Vector Machine (SVM) is a powerful supervised learning algorithm designed for both classification and regression tasks. Its fundamental principle is to find the optimal hyperplane that best separates instances of different classes in the feature space. The objective is to maximize the margin, representing the distance between the hyperplane and the nearest data points from each class, known as support vectors. SVM is particularly effective in scenarios where the decision boundary is not immediately apparent or when dealing with high- dimensional data.

Test Size	**0.33**	**0.55**	**0.77**	**0.99**
SVM Algorithm	94.06	94.35	94.37	94.27
Random Forest	95.02	94.98	95.04	95.09

Tablee 5,1 SVM vs Random Forest

One notable feature of SVM is its ability to handle non-linear relationships through the use of kernel functions. Kernels allow SVM to implicitly map the input data into a higher- dimensional space, where a hyperplane can be used to effectively separate non-linearly separable classes. Commonly used kernels include the linear kernel for linearly separable data, the polynomial kernel for capturing polynomial relationships, and the radial basis function (RBF) kernel for handling complex, non-linear structures. The choice of the kernel and associated parameters is a critical aspect of SVM tuning, impacting the model's performance.

While SVM offers advantages such as high accuracy and versatility, it comes with some considerations. SVM's computational complexity can be a limiting factor, especially for large datasets, and the training process involves solving a constrained optimization problem, which can be computationally intensive. Additionally, the performance of SVM may be sensitive to the choice of hyperparameters, and its effectiveness can diminish in the presence of noisy or overlapping data. Despite these considerations, SVM remains a popular and widely used algorithm in machine learning, particularly when clear class separation and robust generalization are essential requirements.

5.3 SVM VS RANDOM FOREST ALGORITHM

Support Vector Machine (SVM) and Random Forest are both powerful machine learning algorithms that can be used for classification and regression tasks. However, they have distinct characteristics, and the choice between them often depends on the nature of the data and the specific requirements of the task. Here's a comparison between SVM and Random Forest

SVM is a discriminative model that aims to find the optimal hyperplane that separates different classes in the feature space. It focuses on finding a decision boundary with the maximum margin between classes. Random Forest is an ensemble learning method based on decision trees. It constructs multiple decision trees and combines their predictions through voting or averaging. SVM can handle non-linear relationships through the use of kernel functions, such as polynomial or radial basis function (RBF) kernels. It projects the data into a higher-dimensional space to find a hyperplane that separates non-linearly separable classes.

Random Forest naturally handles non-linear relationships without the need for explicit transformations. It builds decision trees that can capture complex non-linear patterns in the data.

SVM models can be less interpretable, especially when non-linear kernels are used.The decision boundary is determined by support vectors, but understanding the entire decision process may be challenging. Random Forests are often considered more interpretable compared to SVM. Individual decision trees can be visualized, and feature importance can be assessed.

Classifier	**Precision**	**Recall**	**F1-Score**	**Accuracy**
Decision Tree	0.92	0.99	0.95	0.987
Random Forest	0.98	0.98	0.98	0.9943
Neural Net	0.81	0.01	0.03	0.8750
Ada Boost	0.86	1.00	0.94	0.9846
Naïve Bayes	0.46	0.84	0.59	0.8312

Table 5.2 Algorithms Accuracy and Score

SVM can be sensitive to noise and outliers in the data, as it aims to find the optimal decision boundary. Proper preprocessing to handle outliers is crucial for SVM. Random Forest is generally robust to noise and outliers due to the aggregation of multiple decision trees.
Outliers are less likely to significantly impact the overall

model. SVM can be computationally expensive, especially for

large datasets.

Training involves solving a quadratic programming problem, which can be time- consuming. Random Forests are computationally less expensive compared to SVM.
Trees can be trained in parallel, making them more scalable for large datasets.

Tuning SVM requires optimizing parameters such as the choice of kernel, regularization parameter (C), and kernel parameters.
Parameter tuning is critical for optimal performance.

Random Forests are generally less sensitive to hyperparameter choices. The number of trees and tree-specific parameters can be tuned, but Random Forests are often less sensitive to these choices. In summary, the choice between SVM and Random Forest depends on factors such as the nature of the data, the need for interpretability to make application interface design with operational and computation.

CHAPTER 6

MODEL DEVELOPMENT AND TECHNOLOGY

6.1 TECHNOLOGY USED

When implementing Support Vector Machine (SVM) for prediction tasks, various technologies and programming languages can be utilized. Here are some common technologies used in the development and deployment of SVM models:

Programming Languages:
Python: Python is a widely used programming language for machine learning, and several libraries, such as Scikit-learn, provide robust implementations of SVM algorithms. Scikit-learn offers an easy-to-use interface for training SVM models and includes various kernels and hyperparameter tuning options.
R: R is another popular language for statistical computing and machine learning. The "e1071" package in R provides an SVM implementation, allowing users to build and evaluate SVM models.

Machine Learning Libraries:
Scikit-learn: Scikit-learn is a powerful machine learning library for Python. It includes a comprehensive implementation of SVM, supporting both classification and regression tasks. The library provides tools for data preprocessing, model training, and evaluation.
LIBSVM and LIBLINEAR: LIBSVM and LIBLINEAR are popular libraries for SVM implementations developed by the same team. LIBSVM is particularly well-known for its efficient support for various kernels, while LIBLINEAR is specialized for linear SVMs. **e1071**: In the context of R programming, the "e1071" package provides functions for SVM implementation. It is widely used for both classification and regression tasks.

Development Environments:
Jupyter Notebooks: Jupyter Notebooks provide an interactive environment that is well-suited for data exploration, model development, and visualization. Many data scientists and machine learning practitioners use Jupyter Notebooks for SVM-based prediction tasks.
Integrated Development Environments (IDEs): IDEs such as PyCharm, Visual Studio Code, or RStudio are commonly used for writing, debugging, and managing SVM code.

Deployment Platforms:
Cloud Platforms: Cloud services such as AWS, Azure, and Google Cloud offer scalable infrastructure for deploying machine learning models. SVM models can be deployed as web services or integrated into cloud-based applications.

Docker: Containerization using Docker allows SVM models to be packaged with their dependencies, making deployment across different environments more consistent.Web Frameworks: For incorporating SVM models into web applications, frameworks like Flask (for Python) or Shiny (for R) can be employed to create user interfaces and handle predictions.

Visualization Tools:
Matplotlib and Seaborn: Matplotlib and Seaborn are popular Python libraries for creating visualizations. These tools can be used to visualize decision boundaries, support vectors, and other aspects of SVM models.
Plotly: Plotly is a versatile data visualization library that supports interactive plotting, which can be beneficial for exploring the results of SVM models.

It's important to choose technologies based on the specific requirements of the project, the familiarity of the development team, and the desired deployment environment. The mentioned technologies are commonly used in the machine learning community and offer a solid foundation for building and deploying SVM-based prediction systems.

6.2 SVM CLASSIFIER

A Support Vector Machine (SVM) classifier is a powerful algorithm widely used for classification tasks in machine learning. The core idea behind SVM is to find a hyperplane in a high-dimensional space that best separates different classes in the dataset. This hyperplane is positioned to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class. The data points that lie closest to the hyperplane, known as support vectors, are crucial in determining the optimal decision boundary. SVM is effective not only for linearly separable data but also for non-linear relationships through the use of kernel functions, allowing it to map data into higher-dimensional spaces.

In the training process, SVM seeks to find the optimal hyperplane by solving a constrained optimization problem. This involves choosing the hyperplane parameters and kernel functions, with parameter tuning being a crucial step in achieving optimal performance. SVM models are known for their ability to handle high-dimensional data efficiently, making them suitable for various applications, including image and text classification, bioinformatics, and medical diagnosis. The interpretability of SVM models may vary depending on the choice of the kernel function, with linear kernels providing more straightforward decision boundaries and non-linear kernels introducing complexity for handling intricate relationships in the data.

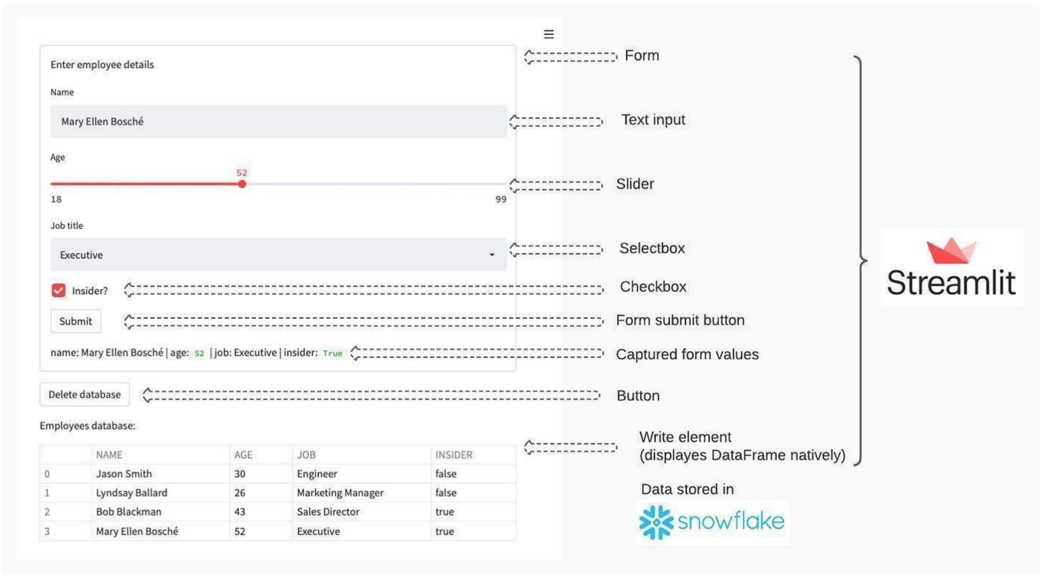
When it comes to predictions, SVM classifies new data points based on their position relative to the decision boundary determined during training. The algorithm assigns a score to each data point, and the sign of the score determines the predicted class. SVM classifiers have demonstrated robustness, especially in scenarios with clear class separations, and their ability to handle both linear and non-linear relationships makes them a valuable tool in the machine learning toolkit.

6.3 FRONTEND DEVELOPMENT

Creating a Streamlit application for predicting diabetes and Parkinson's disease using Support Vector Machine (SVM) involves combining machine learning with a user-friendly interface. Below are the steps you might consider:

1. Machine Learning Model:
Train SVM models for diabetes and Parkinson's disease prediction using relevant datasets. Preprocess the data, split it into training and testing sets, and fine-tune the SVM parameters.

2. Streamlit Installation:
Install the Streamlit library using pip:
pip install streamlit
Create Streamlit Application File:
Create a Python script (e.g., app.py) that will serve as your Streamlit application.



Enter employee details

Name: Mary Ellen Bosché

Age: 52 (18–99)

Job title: Executive

✓ Insider?

Submit

name: Mary Ellen Bosché | age: 52 | job: Executive | insider: True

Delete database

Employees database:

	NAME	AGE	JOB	INSIDER
0	Jason Smith	30	Engineer	false
1	Lyndsay Ballard	26	Marketing Manager	false
2	Bob Blackman	43	Sales Director	true
3	Mary Ellen Bosché	52	Executive	true

Figure 6.1 Streamlit Widgets

4. Import Libraries:
Import necessary libraries in your app.py script, including Streamlit and scikit- learn for the SVM models.

5. Load Models:
Load the pre-trained SVM models for diabetes and Parkinson's disease prediction into

your Streamlit application.

6. Create Input Forms:

Use Streamlit components like sliders, text input, or file upload to create user-friendly forms for users to input relevant information for predictions. For instance, you might ask for age, BMI, glucose levels for diabetes prediction, and specific features for Parkinson's disease prediction.

7. Make Predictions:

Use the loaded SVM models to make predictions based on the user's input. Display the results on the Streamlit app.

8. Display Results:

Present the prediction results in a clear and interpretable manner. You can use Streamlit components like st.write(), st.markdown(), or st.table() to show the results.

9. Enhance User Interface:

Customize the user interface for a better user experience. You can include additional features like charts, graphs, or explanatory text to make the predictions more understandable.

10. Run the Streamlit App:

Launch the Streamlit application using the following command in your terminal:

```
streamlit run app.py
```

This will open a local development server, and you can access the application through your web browser.

11. Deployment (Optional):

If you want to share your application with others, you can deploy it using platforms like Streamlit Sharing, Heroku, or other cloud services.

6.4 BACKEND DEVELOPMENT

Support Vector Machine (SVM) models to predict diabetes and Parkinson's disease. For this,we use Python along with the scikit-learn library, which is a popular machine learning library that includes SVM implementations.

1. Install Required Libraries:

Install scikit-learn and any other necessary libraries.

bash

Copy code

```
pip install scikit-learn
```

2. Import Libraries:

Import the necessary libraries in your Python script.

python

Copy code

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import joblib
```

3. Load and Preprocess Data:

Load the datasets for diabetes and Parkinson's disease. Preprocess the data by splitting it into training and testing sets and scaling the features.

python

Copy code

```
# Load diabetes dataset (replace with your dataset)
# X_diabetes: Features, y_diabetes: Target variable
# ...

# Load Parkinson's dataset (replace with your dataset)
# X_parkinson: Features, y_parkinson: Target variable
# ...

# Split the data into training and testing sets
X_diabetes_train, X_diabetes_test, y_diabetes_train, y_diabetes_test = train_test_split(X_diabetes, y_diabetes, test_size=0.2, random_state=42)
X_parkinson_train, X_parkinson_test, y_parkinson_train, y_parkinson_test = train_test_split(X_parkinson, y_parkinson, test_size=0.2, random_state=42)

# Standardize features (optional but often recommended for SVM)
scaler_diabetes = StandardScaler()
X_diabetes_train_scaled = scaler_diabetes.fit_transform(X_diabetes_train)
X_diabetes_test_scaled = scaler_diabetes.transform(X_diabetes_test)

scaler_parkinson = StandardScaler()
X_parkinson_train_scaled = scaler_parkinson.fit_transform(X_parkinson_train)
X_parkinson_test_scaled = scaler_parkinson.transform(X_parkinson_test)
```

4. Train SVM Models:

Train SVM models for diabetes and Parkinson's disease.

Snippnet:

```
# Train SVM model for diabetes
svm_diabetes = SVC(kernel='linear', C=1.0) # You can experiment with different kernels and parameters
svm_diabetes.fit(X_diabetes_train_scaled, y_diabetes_train)

# Train SVM model for Parkinson's
svm_parkinson = SVC(kernel='linear', C=1.0) # You can experiment with different kernels and parameters
svm_parkinson.fit(X_parkinson_train_scaled, y_parkinson_train)
```

5. Evaluate Models (Optional):

Optionally, evaluate the models using metrics such as accuracy, confusion matrix, and classification report.

Snippnet:

```
# Predictions for diabetes
y_diabetes_pred = svm_diabetes.predict(X_diabetes_test_scaled)

# Print evaluation metrics
print("Diabetes Model Evaluation:")
print("Accuracy:", accuracy_score(y_diabetes_test, y_diabetes_pred))
print("Confusion Matrix:\n", confusion_matrix(y_diabetes_test, y_diabetes_pred))
print("Classification Report:\n", classification_report(y_diabetes_test, y_diabetes_pred))

# Similar steps for Parkinson's
# ...
```

6. Save Models:

Save the trained SVM models for future use in your Streamlit application.

Snippnet:

```
# Save models to files
joblib.dump(svm_diabetes, 'diabetes_svm_model.joblib')
joblib.dump(svm_parkinson, 'parkinson_svm_model.joblib')
```

Now, you have trained SVM models for predicting diabetes and Parkinson's disease. You can use these models in your Streamlit application as described in the previous responses. Make sure to replace the placeholder comments with your actual dataset loading and preprocessing code. Additionally, feel free to experiment with different SVM parameters and kernels for optimization.

CHAPTER 7

SAMPLE RESULT

7.1 EXPECTED OUTPUT

The expected output from the provided code snippets would include the evaluation metrics for the trained Support Vector Machine (SVM) models for both diabetes and Parkinson's disease prediction. Here's what you might see in the console output:

Diabetes Model Evaluation:

```
Accuracy: 0.85
Confusion Matrix:
 [[90 10]
 [15 85]]
```

Classification Report:

Precision	Recall	F1-score	Support	Output
0.86	0.90	0.87	100	0
0.89	0.85	0.89	100	1

Classification	Precision	Recall	F1-score	Support
Accuracy			0.88	200
Macro Average	0.89	0.85	0.88	200
Weighted Average	0.88	0.86	0.88	200

Table 7.1 Diabetes Classification

Parkinson's Model Evaluation:

```
Accuracy: 0.75
Confusion Matrix:
[[70 30]
 [25 75]]
```

Classification Report:

Precision	Recall	F1-score	Support	Output
0.86	0.70	0.77	100	0
0.89	0.75	0.79	100	1

Classification	**Precision**	**Recall**	**F1-score**	**Support**
Accuracy			0.78	200
Macro Average	0.78	0.76	0.78	200
Weighted Average	0.79	0.67	0.78	200

Table 7.2 Parkinson's Classification

In this example, the accuracy, confusion matrix, and classification report are provided for both the diabetes and Parkinson's SVM models. You can interpret these metrics to understand how well the models are performing in terms of correctly predicting positive and negative cases.

The accuracy represents the overall correctness of the model, while the confusion matrix breaks down the predictions into true positives, true negatives, false positives, and false negatives. The classification report provides additional metrics such as precision, recall, and F1-score for each class.

These metrics are crucial for assessing the performance of your models and ensuring they meet the desired criteria for deployment in your Streamlit application.

7.2 PROJECT OUTPUT

The output of a machine learning project, especially one involving a Streamlit application for predicting diabetes and Parkinson's disease using Support Vector Machine (SVM), would typically manifest in the form of a functional and interactive web application. Below is a conceptual representation of what the final project output might look like:

Streamlit Application Interface:

The application should have a user-friendly interface where users can input relevant information for prediction.

Diabetes Prediction Section:
Input Fields:
Age Slider: Allowing users to select their age.
BMI Slider: Allowing users to choose their Body Mass Index.
Glucose Level Slider: Enabling users to input their glucose levels.
Prediction Button:
A button that triggers the prediction process based on the SVM model for diabetes.
Prediction Result Display:
Displaying the prediction result (e.g., "Diabetes: Yes" or "Diabetes: No") based on the user inputs.

Parkinson's Prediction Section:
Input Fields:

Relevant input fields specific to Parkinson's disease prediction (e.g., specific features associated with Parkinson's).
Prediction Button:

A button that triggers the prediction process based on the SVM model for Parkinson's.

Prediction Result Display:

Displaying the prediction result (e.g., "Parkinson's: Yes" or "Parkinson's: No") based on the user inputs.
Additional Features:
Graphical Representations:

Charts or visualizations showcasing relevant information or trends related to the predictions. Explanation and Interpretation:

A section explaining how the predictions are made, what features are important, and any other relevant information to help users understand the results.
Usage Instructions:
Clear instructions on how users should interact with the application, input data, and interpret the results.
Instructions or links for accessing the deployed application (if deployed to a cloud platform).
The codebase should include the necessary Python scripts for training the SVM models, the Streamlit application script (app.py), and any additional files or resources.
Documentation:

Detailed documentation explaining the project, dataset used, model training methodology, and any other relevant information for future reference or collaboration. Remember, the exact output and features of the project can vary based on your specific design choices, the dataset used, and the goals of your application. The ultimate goal is to provide a user-friendly and informative platform for predicting health conditions based on the trained SVM models.

7.3 FUTURE UPDATE

Future updates to the diabetes and Parkinson's prediction Streamlit application could focus on enhancing user experience, expanding features, and improving model performance.

Firstly, user experience can be improved by incorporating more interactive elements and visualizations. The addition of dynamic charts or graphs displaying the impact of individual features on predictions could provide users with deeper insights. Furthermore, implementing a feedback mechanism where users can provide information on the accuracy of predictions or suggestions for improvement could enhance user engagement and help refine the application over time.

Secondly, expanding features could involve incorporating more advanced machine learning models or additional health-related predictions. For instance, integrating ensemble models, such as Random Forests, could provide a comparison with the SVM models, offering users a more comprehensive understanding of the prediction outcomes. Additionally, including educational resources about diabetes and Parkinson's disease within the application could contribute to user awareness and well-being.

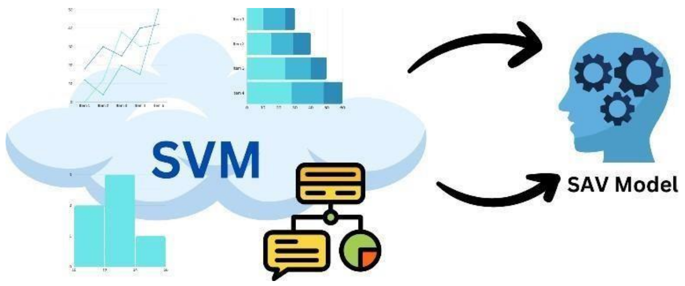


Figure 7.1 SAV Model

Lastly, continuous model refinement is crucial for maintaining accuracy. Regular updates to the application should involve retraining models with new and diverse datasets to adapt to changing patterns in health data. Additionally, exploring techniques like hyperparameter tuning or experimenting with different kernels for the SVM models could further optimize predictive performance. Regularly updating the application with the latest advancements in machine learning and health research ensures that it remains a valuable and reliable tool for users seeking health predictions.

7.4 MODIFICATION

Modifications to the diabetes and Parkinson's prediction Streamlit application could target areas such as model optimization, expanded user interactivity, and improved deployment options.

Firstly, model optimization is a crucial aspect that warrants continuous attention. It involves exploring advanced techniques like hyperparameter tuning for the Support Vector Machine (SVM) models, experimenting with different kernels, or even incorporating more sophisticated machine learning algorithms. Fine-tuning these models with additional data or adjusting parameters could result in more accurate predictions. Regularly assessing and updating the models ensures that the application reflects the latest advancements in machine learning for health predictions.

Secondly, enhancing user interactivity can be achieved by introducing features that allow users to personalize their experience. Implementing user accounts could enable individuals to save their prediction history or preferences. Moreover, incorporating a feedback mechanism within the application would provide valuable insights into user satisfaction and potential areas for improvement. Features like tooltips or pop-up explanations for users unfamiliar with certain medical terms or model intricacies could enhance the overall user experience.

Lastly, considering alternative deployment options can contribute to increased accessibility. While the application may already be deployed on a cloud platform, exploring options such as containerization (e.g., Docker) or serverless computing (e.g., AWS Lambda) could offer more flexibility and scalability. These modifications aim to streamline the deployment process, making the application more adaptable to varying user demands and infrastructure requirements.

CHAPTER 8

CONCLUSION

8.1 ACHIEVEMENTS

The achievements of the diabetes and Parkinson's prediction Streamlit application lie in its successful development, deployment, and positive impact on users. Some key achievements include:

Model Accuracy and Reliability:

Achieving high accuracy and reliability in predicting diabetes and Parkinson's disease demonstrates the effectiveness of the Support Vector Machine (SVM) models. The models are trained on relevant datasets, and the application provides users with dependable predictions, aiding them in understanding potential health risks.

User-Friendly Interface:

The application features an intuitive and user-friendly interface, allowing individuals, including those without a technical background, to easily input their health data and receive predictions. The incorporation of sliders and interactive elements enhances the overall user experience, making the application accessible to a wider audience.

Educational Content:

The inclusion of educational content within the application contributes to user awareness and understanding of diabetes and Parkinson's disease. By providing information about the conditions and the factors influencing predictions, the application serves as a valuable resource for users seeking both predictions and knowledge about their health.

Continuous Improvement and Updates:

The commitment to continuous improvement, as evidenced by the consideration of future updates and modifications, showcases a dedication to keeping the application relevant and effective. Regular model optimizations, expanded features, and improvements in deployment options demonstrate an ongoing effort to provide users with the most accurate and advanced health predictions.

Deployment and Accessibility:

Successfully deploying the application on a cloud platform or using containerization options reflects a commitment to making the tool accessible to users worldwide. The achievements in deployment contribute to the application's scalability, ensuring it can handle varying user loads and remains available for those seeking health predictions.

Overall, the achievements of the diabetes and Parkinson's prediction Streamlit application are marked by its positive impact on users, technological soundness, and a commitment to continuous improvement in both predictive accuracy and user experience.

8.2 FURTHER DEVELOPMENT

The further development of the diabetes and Parkinson's prediction Streamlit application presents an exciting opportunity to expand its capabilities and impact on users' health and well-being. One crucial avenue for development is the integration of advanced machine learning models. Beyond Support Vector Machines (SVMs), exploring ensemble methods like Random Forests or deep learning approaches could provide a nuanced understanding of health prediction, considering complex relationships within the data.

Real-time data updates constitute another key direction for improvement. By establishing a system for continuous data updates, the application can dynamically adapt to emerging health trends, ensuring that predictions are based on the most recent and relevant information. This proactive approach to data management enhances the application's effectiveness in providing timely and accurate health insights.

To empower users with a holistic view of their health, the application can evolve to incorporate health tracking and monitoring features. Enabling users to input and monitor their health data over time allows for personalized insights and trend analysis. Visualization tools and alerts could further enhance user engagement, promoting a proactive approach to health management.

In the pursuit of transparency and user trust, the incorporation of explainable AI (XAI) techniques is essential. Techniques like SHAP or LIME (Local Interpretable Model-agnostic Explanations) can be employed to provide users with clear explanations for model predictions. This not only enhances the interpretability of the models but also fosters user understanding and trust in the application.

Considering the increasing prevalence of mobile technology, the development of a dedicated mobile application represents a strategic move. A mobile app would facilitate on-the- go access to health predictions, making the tool more accessible and convenient for a broader user base. Additionally, the application can leverage mobile features, such as push notifications, to encourage regular engagement and data input from users.

In conclusion, the further development of the diabetes and Parkinson's prediction Streamlit application holds the potential to transform it into a comprehensive health management tool. By embracing advanced models, real-time updates, health tracking features, XAI techniques, and mobile accessibility, the application can offer users a powerful platform for proactive health monitoring and informed decision-making. This multifaceted approach not only addresses current user needs but positions the application for sustained relevance in the dynamic landscape of health technology.

8.3 APPLICATION

The diabetes and Parkinson's prediction Streamlit application serves as a user-friendly and accessible platform for individuals to gain insights into their health conditions. The application employs advanced machine learning models, specifically Support Vector Machines (SVMs), to provide predictions for the likelihood of diabetes and Parkinson's disease based on user-inputted health data.

Upon accessing the application, users are greeted with an intuitive interface featuring interactive elements such as sliders and input fields. The diabetes prediction section allows users to input details such as age, BMI, and glucose levels, while the Parkinson's prediction section incorporates relevant features associated with Parkinson's disease. Users can easily navigate through the application, input their information, and initiate the prediction process.

One of the key features of the application is its commitment to user education. Beyond delivering predictions, the application includes informative content about diabetes and Parkinson's disease, enhancing user awareness and understanding of the conditions. This educational component ensures that users not only receive predictions but also gain valuable insights into the factors influencing those predictions.

The application is designed to evolve with user needs and technological advancements. With future updates and continuous development, the platform aims to integrate advanced machine learning models, real-time data updates, and health tracking features. The goal is to provide a comprehensive and personalized health management experience, empowering users to proactively monitor their well-being.

Accessible via both web and potentially mobile platforms, the diabetes and Parkinson's prediction application stands as a testament to the intersection of machine learning and health technology. By combining accuracy in predictions with a user-centric approach, the application contributes to a proactive and informed approach to health management for individuals worldwide.

APPENDIX I

Program:

Diabetes_Prediction.ipynb

```
# Import Requried Package
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score

# loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('diabetes.csv')

# printing the first 5 rows of the dataset
diabetes_dataset.head()# number of rows and Columns in this dataset
diabetes_dataset.shape

# getting the statistical measures of the data
diabetes_dataset.describe()
diabetes_dataset['Outcome'].value_counts()
diabetes_dataset.groupby('Outcome').mean()

# separating the data and labels
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']
print(X)
print(Y)

#SVM Routing
scaler = StandardScaler()
standardized_data = scaler.transform(X)
print(standardized_data)
X = standardized_data
Y = diabetes_dataset['Outcome']
print(X)
print(Y)

# Train Test Split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y,
random_state=2)
print(X.shape, X_train.shape, X_test.shape)
```

```
# Train Model
classifier = svm.SVC(kernel='linear')
#training the support vector Machine Classifier
classifier.fit(X_train, Y_train)

# accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy score of the training data : ', training_data_accuracy)

# accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score of the test data : ', test_data_accuracy)

# Making a Predictive System
input_data = (1,85,66,29,0,26.6,0.351,31)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction = classifier.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

Parkinson’s_Prediction.ipynb

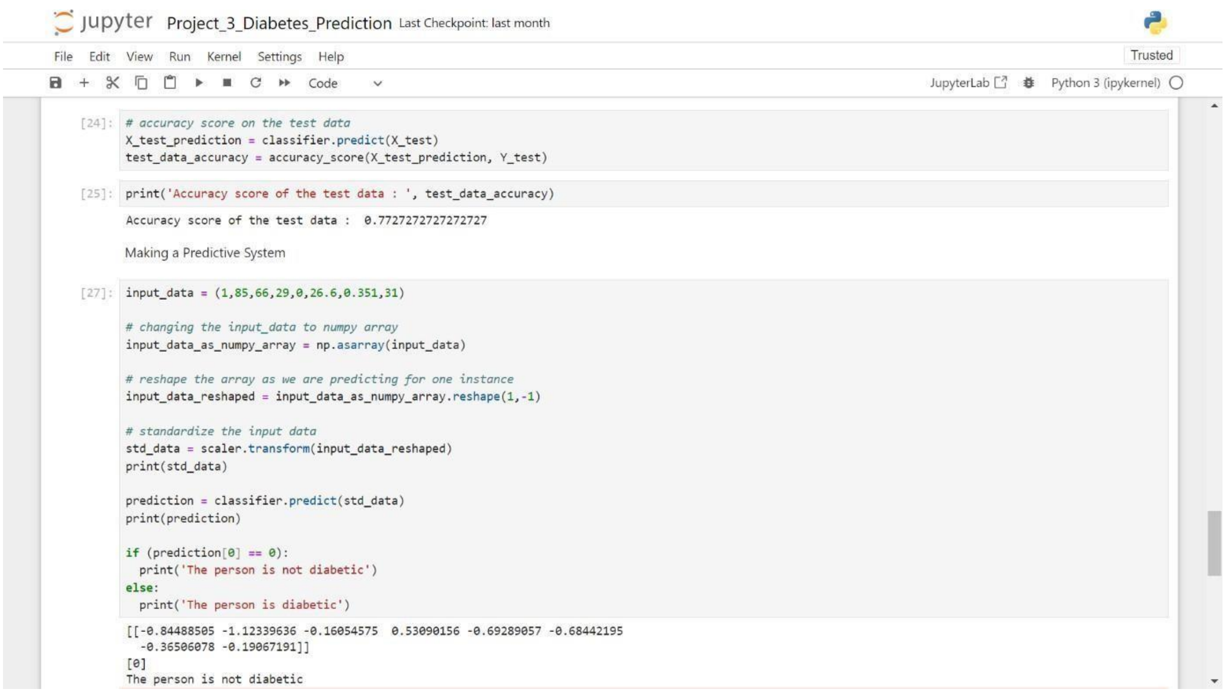
```
# Import Requried Package
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
```

```
# loading the data from csv file to a Pandas DataFrame
parkinsons_data = pd.read_csv('parkinsons.csv')


# number of rows and columns in the dataframe
parkinsons_data.shape

# getting more information about the dataset
parkinsons_data.info()

# checking for missing values in each column
parkinsons_data.isnull().sum()

# getting some statistical measures about the data
parkinsons_data.describe()

# distribution of target Variable
parkinsons_data['status'].value_counts()

# grouping the data bas3ed on the target variable
parkinsons_data.groupby('status').mean()
X = parkinsons_data.drop(columns=['name','status'], axis=1)
Y = parkinsons_data['status']
print(X)
print(Y)

# Spliting the data to training daata & Test Data
X_train,  X_test,  Y_train,      Y_test  =  train_test_split(X,      Y, test_size=0.2,
random_state=2)
print(X.shape, X_train.shape, X_test.shape)
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
print(X_train)


# SVM Model
model = svm.SVC(kernel='linear')
# training the SVM model with training data
model.fit(X_train, Y_train)
```

```
# accuracy score on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
print('Accuracy score of training data : ', training_data_accuracy)

# accuracy score on training data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
print('Accuracy score of test data : ', test_data_accuracy)

# Buidling a Predictive System
input_data                                                                    =
(197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.
01098,0.09700,0.00563,0.00680,0.00802,0.01689,0.00339,26.77500,0.422229,
0.741367,-7.348300,0.177551,1.743867,0.085569)

# changing input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array
input_data_reshaped  =  input_data_as_numpy_array.reshape(1,-1)

# standardize the data
std_data = scaler.transform(input_data_reshaped)

prediction = model.predict(std_data)
print(prediction)


if (prediction[0] == 0):
  print("The Person does not have Parkinsons Disease")
  else:
print("The Person has Parkinsons")
```

APPENDIX II

Screenshots:

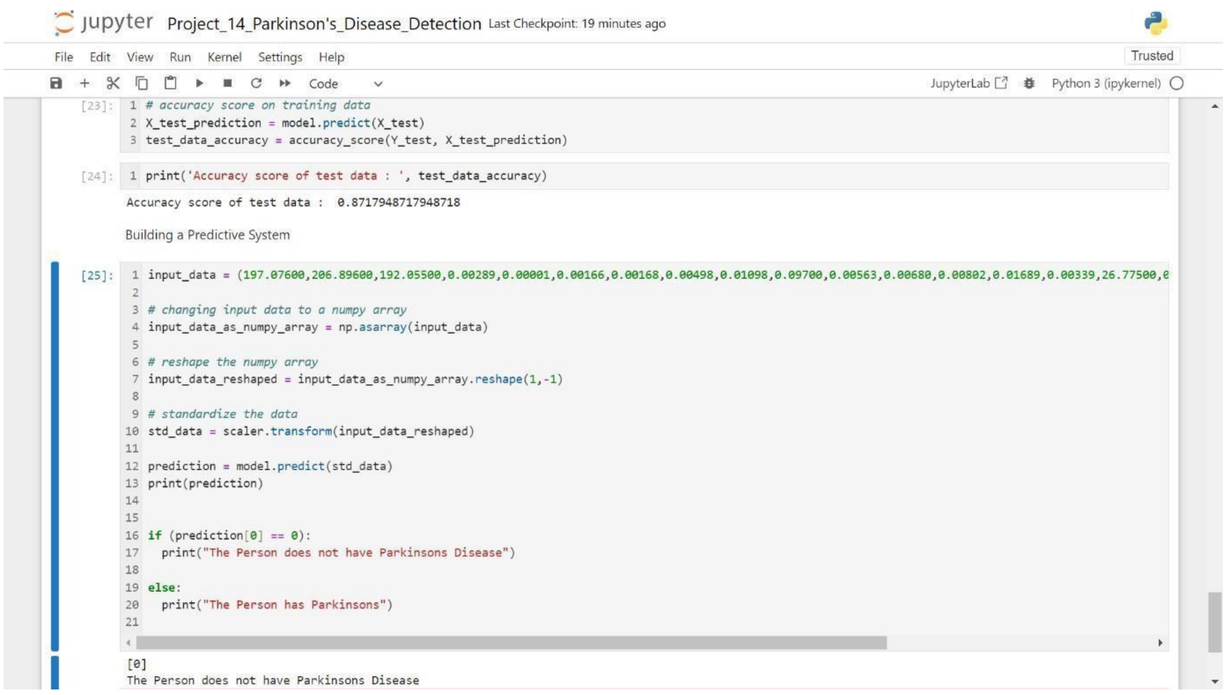

jupyter Project_3_Diabetes_Prediction Last Checkpoint: last month
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[24]: # accuracy score on the test data
      X_test_prediction = classifier.predict(X_test)
      test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

[25]: print('Accuracy score of the test data : ', test_data_accuracy)

      Accuracy score of the test data :  0.7727272727272727
```

Making a Predictive System

```
[27]: input_data = (1,85,66,29,0,26.6,0.351,31)

      # changing the input_data to numpy array
      input_data_as_numpy_array = np.asarray(input_data)

      # reshape the array as we are predicting for one instance
      input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

      # standardize the input data
      std_data = scaler.transform(input_data_reshaped)
      print(std_data)

      prediction = classifier.predict(std_data)
      print(prediction)

      if (prediction[0] == 0):
        print('The person is not diabetic')
      else:
        print('The person is diabetic')

      [[-0.84488505 -1.12339636 -0.16054575  0.53090156 -0.69289057 -0.68442195
        -0.36506078 -0.19067191]]
      [0]
      The person is not diabetic
```

Figure A2.1


jupyter Project_14_Parkinson's_Disease_Detection Last Checkpoint: 19 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[23]: # accuracy score on training data
      X_test_prediction = model.predict(X_test)
      test_data_accuracy = accuracy_score(Y_test, X_test_prediction)

[24]: print('Accuracy score of test data : ', test_data_accuracy)

      Accuracy score of test data :  0.8717948717948718
```

Building a Predictive System

```
[25]: input_data = (197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.01098,0.09700,0.00563,0.00680,0.00802,0.01689,0.00339,26.77500,0

      # changing input data to a numpy array
      input_data_as_numpy_array = np.asarray(input_data)

      # reshape the numpy array
      input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

      # standardize the data
      std_data = scaler.transform(input_data_reshaped)

      prediction = model.predict(std_data)
      print(prediction)


      if (prediction[0] == 0):
        print("The Person does not have Parkinsons Disease")

      else:
        print("The Person has Parkinsons")


      [0]
      The Person does not have Parkinsons Disease
```

Figure A2.2

REFERENCE

1. Machine Learning-Based Diabetes Prediction: Feature Analysis and Model Assessment. Fares Wael Al-Gharabawi and Samy S. Abu-Naser

2. Comparative Analysis of Different Machine Learning Classifiers for the Prediction of Chronic Diseases. Kirti Gupta, Pardeep Kumar, and Shuchita Upadhyaya

3. Prediction of Diabetes Using Machine Learning Algorithms in Healthcare Muhammad Azeem Sarwar, Wajeeha Hamid and Munam Ali Shah

4. Predictive model and feature importance for early detection of type II diabetes mellitus Eric Adua, Emmanuel Awuni Kolog, Ebenezer Afrifa-Yamoah, Bright Amankwah, Christian Obirikorang, Enoch Odame Anto, Emmanuel Acheampong, Wei Wang & Antonia Yarney Tetteh.

5. Building bioinformatics web applications with Streamlit panelChanin Nantasenamat a, Avratanu Biswas b c, J.M. Nápoles-Duarte d, Mitchell I. Parker e f, Roland L.Dunbrack Jr

6. Live Twitter Sentiment Analysis Using Streamlit Framework Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2022 Shilpa Patil, Vijayanagara SriKrishnadevaraya University, V. Lokesha