



CAPSTONE PROJECT REPORT

(Project Term January-May 2023)

IoT and Machine Learning based Health Monitoring System

Submitted by

Name	Registration Number
Chandra Prakash Rai	11915260
Kamal Kant	11915735
Arpit Thanoch	11912146
Juaira Kanon Rumky	11900160

Project Group Number: KC324
Course Code: CSE-445

Under the guidance of

Dr. Varun Dogra

School of Computer Science & Engineering
Lovely Professional University

PAC FORM



TOPIC APPROVAL PERFORMA

School of Computer Science and Engineering (SCSE)

Program : P132::B.Tech. (Computer Science and Engineering)

COURSE CODE : CSE445

REGULAR/BACKLOG : Regular

GROUP NUMBER : CSESGC0324

Supervisor Name : Dr.Varun Dogra

UID : 12343

Designation : Associate Professor

Qualification : Ph.D.

Research Experience : 12 years

SR.NO.	NAME OF STUDENT	Prov. Regd. No.	BATCH	SECTION	CONTACT NUMBER
1	Kamal Kant	11915735	2019	K19RB	7589159497
2	Chandra Prakash Rai	11915260	2019	K19DE	6378066569
3	Chandan Gupta	11915251	2019	K19BH	6200891521
4	Arpit Thanoch	11912146	2019	K19PG	9463443504
5	Juaira Kanon Rumky	11900160	2019	K19JC	7973803399

SPECIALIZATION AREA : Programming-II

Supervisor Signature: *For Dr. Varun Dogra*

PROPOSED TOPIC : Health monitoring system

Qualitative Assessment of Proposed Topic by PAC		
Sr.No.	Parameter	Rating (out of 10)
1	Project Novelty: Potential of the project to create new knowledge	7.41
2	Project Feasibility: Project can be timely carried out in-house with low-cost and available resources in the University by the students.	7.41
3	Project Academic Inputs: Project topic is relevant and makes extensive use of academic inputs in UG program and serves as a culminating effort for core study area of the degree program.	7.41
4	Project Supervision: Project supervisor's is technically competent to guide students, resolve any issues, and impart necessary skills.	7.59
5	Social Applicability: Project work intends to solve a practical problem.	7.95
6	Future Scope: Project has potential to become basis of future research work, publication or patent.	7.45

PAC Committee Members		
PAC Member (HOD/Chairperson) Name: Raj Karan Singh	UID: 14307	Recommended (Y/N): Yes
PAC Member (Allied) Name: Vikas Verma	UID: 11361	Recommended (Y/N): Yes
PAC Member 3 Name: Dr. Makul Mahajan	UID: 14575	Recommended (Y/N): Yes

Final Topic Approved by PAC: Health monitoring system

Overall Remarks: Approved

PAC CHAIRPERSON Name: 25708::Dr.Rachit Garg

Approval Date: 04 Apr 2023

DECLARATION

DECLARATION

We hereby declare that the project work entitled "**Health Monitoring System**" is an authentic record of our own work carried out as requirements of the Capstone Project for the award of B.Tech degree in Computer Science & Engineering from Lovely Professional University, Phagwara, under the guidance of **Dr. Varun Dogra**, during January to May 2023. All the information furnished in this capstone project report is genuine and based on our own intensive work.

Project Group Number: KC324

Name of Student 1: Chandra Prakash Rai

Registration Number: 11915260

Name of Student 2: Kamal Kant

Registration Number: 11915735

Name of Student 3: Arpit Thanoch

Registration Number: 11912146

Name of Student 4: Juaira Kanon Rumky

Registration Number: 11900160

(Signature of Student 1) Chandra Prakash Rai
Date: 10/05/2023

(Signature of Student 2) Kamal
Date: 10/05/2023

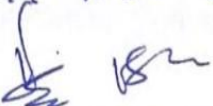
(Signature of Student 3) Arpit Thanoch
Date: 10/05/2023

(Signature of Student 4) Juaira
Date: 10/05/2023

CERTIFICATE

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort, and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the B.Tech degree in Computer Science & Engineering award from Lovely Professional University, Phagwara.

for Dr. Varun Dogra


Signature and Name of the Mentor

Designation Assistant Professor

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date: 10/5/23

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance. It would not have been possible to prepare this report in this form without the valuable help, cooperation, and guidance of our supervisor, friends, and team members. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

Our respect and thanks to our respected supervisor **Dr. Varun Dogra** who assist us on the project and gave us all support and guidance, which made us complete the project on “**Health Monitoring System**”. We owe our deep gratitude to our supervisor, who took a keen interest in our project work and guided us all along, till the completion of the project by providing all the necessary information and essential study material. We’re extremely thankful to him for his help.

We’re thankful for and fortunate enough to get constant encouragement, support, and guidance from all of them and my university **LPU** and my school ‘**School of Computer Science & Engineering**’ to successfully complete our project.

Chandra Prakash Rai
Kamal Kant
Arpit Thanoch
Juaira Kanon Rumky

Thank You!

Table of Contents

IOT AND MACHINE LEARNING BASED HEALTH MONITORING SYSTEM	1
PAC FORM.....	2
DECLARATION.....	3
CERTIFICATE	4
ACKNOWLEDGEMENT.....	5
1. INTRODUCTION.....	8
2. PROFILE OF THE PROBLEM	9
3. EXISTING SYSTEM.....	10
3.1. INTRODUCTION	10
3.2. EXISTING SOFTWARE	10
3.3. DFD FOR THE PRESENT ARRANGEMENT.....	12
3.4. WHAT'S NEW IN THE SYSTEM TO BE DEVELOPED.....	12
4. PROBLEM ANALYSIS	13
4.1. PRODUCT DEFINITION.....	13
4.2. FEASIBILITY ANALYSIS	14
4.2.1. Technical feasibility	14
4.2.2. Economic Feasibility	14
4.2.3. Operational feasibility	15
4.2.4. Legal and regulatory analysis.....	15
4.3. PROJECT PLAN.....	15
5. SOFTWARE REQUIREMENT ANALYSIS	16
5.1. INTRODUCTION	16
5.2. GENERAL DESCRIPTION	16
5.3. SPECIFIC REQUIREMENTS	17
6. DESIGN	18
6.1. SYSTEM DESIGN.....	18
6.2. DESIGN NOTATION.....	19
6.3. FLOWCHART	20
6.4. PSEUDO CODE.....	21
7. TESTING	22
7.1. FUNCTIONAL TESTING.....	22
7.2. STRUCTURAL TESTING	23
7.3. LEVELS OF TESTING	24

7.4. TESTING THE PROJECT	25
8. IMPLEMENTATION	26
8.1. IMPLEMENTATION OF THE PROJECT.....	26
8.2. CONVERSION PLAN	27
8.3. POST-IMPLEMENTATION & SW MAINTENACE.....	28
9. PROJECT LEGACY	28
9.1. CURRENT STATUS OF THE PROJECT.....	28
9.1.1. Real-Time Data Collection Using IoT	29
9.1.2. Applying Machine Learning Technique.....	29
9.2. REMAINING AREAS OF CONCERN	29
9.3. TECHNICAL AND MANAGERIAL LESSONS LEARNT.....	31
9.3.1 technical lessons	31
9.3.2 managerial lessons	31
10. USER GUIDE	32
10.1. INTRODUCTION	32
10.2. SYSTEM REQUIREMENTS	32
10.3. GETTING STARTED.....	32
10.4. USING THE SYSTEM	32
10.4.1. Diabetes prediction.....	32
10.4.2. Heart disease prediction	34
10.4.3. Parkinson’s disease prediction	36
10.5. REVIEW ALERTS	38
10.6. CUSTOMIZABLE SYSTEM	38
10.7. CONCLUSION	38
11. SOURCE CODE	39
11.1. DIABETES PREDICTION	39
11.2. PARKINSON’S DISEASE PREDICTION	44
11.3. HEART DISEASE PREDICTION.....	49
12. BIBLIOGRAPHY	53

1. INTRODUCTION

The Internet of Things (IoT) speaks for the connectivity of all distinct digital and physical items within the Internet buildings, which offers a range of services including smart farms, smart healthcare, etc. These little seeming things are capable of interacting both with the outside world and with one another. They can also react to what is happening in their immediate environment.

In the past, designated medical centers had a noisy sitting area where a hefty number of individuals were made to wait for a long time to see the medical professionals. Healthcare devices using IoT and Machine Learning technologies to collect and monitor physiological data from patients have been a major contributor to the healthcare industry. The wearable device continuously collects data and transfers it wirelessly to the cloud-based platform. Machine Learning algorithms analyze the collected data to enable remote patient monitoring, personalized treatment plans, predictive maintenance, and drug discovery. The combination of IoT and Machine Learning technologies promises to transform the healthcare industry by enabling healthcare professionals to deliver personalized, data-driven care that is more efficient, effective, and affordable.

Within IoT-based healthcare applications, the layer that senses gather data from people and sends it via communication technologies to the storage layer. Sensible decisions are made for healthcare applications using machine learning. This survey covers every area, from the use of Intelligence in the healthcare industry to micro-processing devices.

Every country has been impacted by the Internet's size, which has a profound impact on people's lives. Thanks to recent developments in micro-electrical sensing and flexible electronic devices, high-scale data processing, and the rapid rise of wireless technology, the IoT is now crucial to many application sectors. IoT and machine learning are two revolutionary technologies that are drastically changing the healthcare sector. The healthcare sector has undergone a revolution thanks to the integration of IoT and machine learning technology.

The efficiency of healthcare services needs to be improved; this is a major concern nowadays. With the development of IoT, it is now simple to gather a lot of data through sensors and wearable devices. As the number of such interconnected devices rises, a vast amount of data is produced, revealing numerous prospects amid many difficulties.

Ultra-fast, high-speed wireless communication makes it possible to transmit data from one end to another. A lot of people are currently interested in cloud servers since they can store and analyze a lot of data. Additionally, the contribution of machine learning has recently helped the Internet of Things to an exceptional success. Nowadays, data mining over IoT-generated data is used in medical services to carry out many difficult tasks. Thus, the improved efficacy and enhanced performance.

People now perceive traditional medical procedures differently Given to IoT-Machine Learning combo's significance on the healthcare industry. Everything has eventually started to change with assistance of IoT technology, wellness mobile applications, and the use of big data processing techniques to allow for the early detection of diseases and patient monitoring. Additionally, a lot of research has been done to determine how to reduce waiting times for patients with IoT-based health services.

IoT-based health services employ ML frameworks to boost productivity in the healthcare industry and the performance of medical experts. IoT device data is processed using ML algorithms to find hazardous surroundings and enhance quality of life. IoT, cloud, and ML methods working together offer a variety of applications, particularly in the medical field, and can therefore handle complicated jobs with high intelligence. Soon, healthcare professionals will be able to monitor patients and predict diseases with little to no human intervention thanks to advanced programs.

2. PROFILE OF THE PROBLEM

Health remains a crucial aspect of all technological development humans undertake. The significance of healthcare has been highlighted in recent times due to the outbreak of the coronavirus. The use of technology for remote monitoring of health is a viable alternative, particularly in regions where the epidemic is rapidly spreading. Hence, the present remedy is a healthcare surveillance mechanism that operates on the Internet of Things (IoT). This system, also called remote patient monitoring, allows medical professionals to monitor patients in nontraditional healthcare settings, like their homes, thereby increasing access to healthcare facilities at a reduced price.

The ability to remotely monitor health issues via wearable technology has been made possible by breakthroughs in wireless communication, and methods based on the accurate analysis

of data. These sensors and wearable gadgets can be found in a variety of gear, that includes but is not limited to, eyeglasses, fitness bands, as well as in gadgets like smartphones, headphones, wristwatches, fitness trackers, smart scales, and blood pressure monitors.

A health and wellness system using Internet of Things and machine algorithms is a system that uses devices having internet connectivity to collect information on a person's health and uses machine learning algorithms to analyze that data and provide insights and recommendations. The data collected by devices are sent to a cloud-based platform that employs Artificial Intelligence techniques for the analytical breakdown in turn to detect patterns, identify potential health concerns, and make personalized recommendations for maintaining or improving health. If the system notices any potential health problems, it can also send alerts to the user or to medical professionals. For a more complete picture of a person's medical history, the system can also be integrated with electronic health records (EHRs).

3. EXISTING SYSTEM

3.1. INTRODUCTION

Medical technology employs the Internet of Things (IoT) for a variety of purposes, such as remote monitoring, smart sensors, and the synchronization of medical devices. It improves the way medical practitioners treat their patients and keeps them secure in their health. The accuracy and sweetness of medical data are increased through the collection of various data by healthcare equipment from a large number of real-world situations.

A wearable device that continuously gathers and monitors numerous physiological features, such as heart rate, and blood pressure, constitutes the IoT and machine learning-based healthcare device. The device transmits the data it has gathered to a cloud-based platform utilizing sensors and wireless connectivity, where it is stored and exposed to Machine Learning algorithms for examination.

3.2. EXISTING SOFTWARE

Here are a few examples of how IoT and Machine Learning work in healthcare:

- **Data Collection:**

IoT and Machine Learning devices such as wearables, sensors, and smart medical devices can collect real-time health data from patients and distribute data to a platform that resides in the cloud so that it can be kept and studied.

- **Data Analysis:**

Artificial machine-based algorithms analyze the collected data to identify patterns and connections. This analysis can be used to develop and modify treatment plans, predict potential health risks, and identify new drug candidates. ML algorithms can also identify variations or outliers in the data, allowing healthcare professionals to detect potential health risks early.

- **Remote Patient Monitoring:**

Healthcare professionals can monitor patient health remotely using the data collected by IoT devices using trained Machine Learning models. This enables timely actions and reduces the need for patients to visit a healthcare center, making healthcare more accessible and simpler for patients.

- **Predictive Maintenance:**

IoT-enabled medical devices can be monitored using Machine Learning algorithms to detect potential failures before they occur. This can help prevent medical device interruption, reduce maintenance costs, and ensure that patients receive timely and accurate care.

- **Personalized Treatment Plans:**

The analysis of data collected by IoT devices enables healthcare professionals to develop personalized treatment plans designed for each patient's unique needs. Algorithms based on machine learning can anticipate which treatments are likely to be helpful for each patient by finding patterns and associations in the data, so minimizing the chance of treatment by trial and error.

- **Drug Discovery:**

Enormous amounts of medical data may be examined by machine learning algorithms to find potential new candidates willing to take the medication and forecast their efficacy. This could accelerate the discovery of new drugs, lower the price of medication development, and enhance patient satisfaction.

Overall, IoT and Machine Learning are transforming the healthcare industry by enabling healthcare professionals to deliver personalized, data-driven care that is more efficient, effective, and affordable.

3.3. DFD FOR THE PRESENT ARRANGEMENT

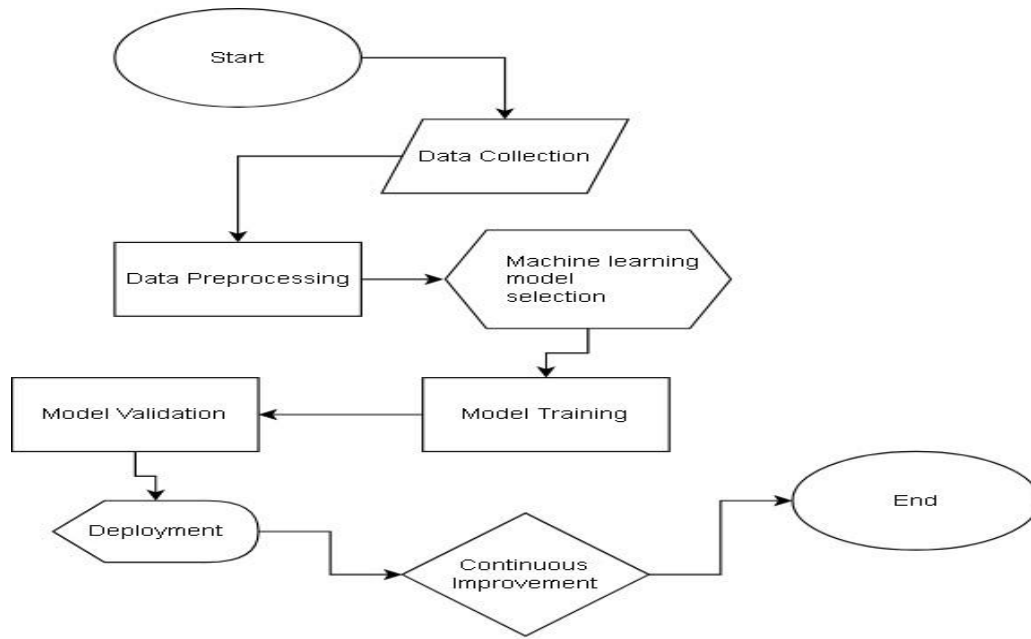


Figure 1: Data Flow Diagram of the system

3.4. WHAT'S NEW IN THE SYSTEM TO BE DEVELOPED

IoT-based healthcare monitoring systems can provide real-time data about patients' health and well-being, enabling doctors to monitor their condition remotely and provide personalized care. With the help of wearable devices, such as smartwatches and fitness trackers, patients can monitor their vital signs and receive alerts if any abnormalities are detected. This technology can be especially useful for individuals who need constant surveillance and care for chronic illnesses like diabetes and heart disease.

Massive amounts of patient data can be analyzed by Machine Learning (ML) healthcare monitoring systems, which may then provide discoveries that aid medical professionals in formulating plans for the appropriate treatment. Using machine learning algorithms, physicians may predict the start of specific disorders and take preventive actions by seeing patterns and trends in patient data. A patient's unique traits and medical background can be used to build specific treatment programs using this technology.

4. PROBLEM ANALYSIS

4.1. PRODUCT DEFINITION

During the selection of hardware components for an Artificial Intelligence coupled with an Internet of Things wellness surveillance system, it was indeed crucial and important to take the genus of health data to be collected, the system size, and how much power is needed into account. It is also vital to take into account if the hardware components will work with the software platform being used to manage and analyze the collected data.

The hardware used in our project includes:

- **Microcontroller**

Little computers called microcontrollers can be used to manage and process data from sensors. They can also be used to leverage Wi-Fi or Bluetooth to link sensors to the internet. IoT projects frequently use Arduino, Raspberry Pi, and ESP32 microcontrollers. In our use case, the ESP32 microcontroller has been utilized.

- **Sensors**

The sensors are the most important aspect of any health monitoring system. Heart rate, blood pressure and many more health-related data points., can be gathered using a variety of sensors. ECG, PPG, temperature, and humidity sensors are a few examples of common sensor types. This project uses a pulse meter sensor, which in the most used language is generally referred to as an oximeter.

- **Power**

Supply A power source is a crucial part of every Internet of Things project. Depending on the project's size and complexity, you might have to decide between using batteries, solar panels, or a power adaptor. In this instance, a USB cable is used to power the Iot-based wellness monitoring device. and can be connected to any USB Type-A adapter.

- **Jumper Wires**

DuPont wires, commonly referred to as jumper wires, are a kind of electrical cable used to link several parts of electronics projects. They are available in an array of lengths and colors and are typically composed of insulated copper wire.

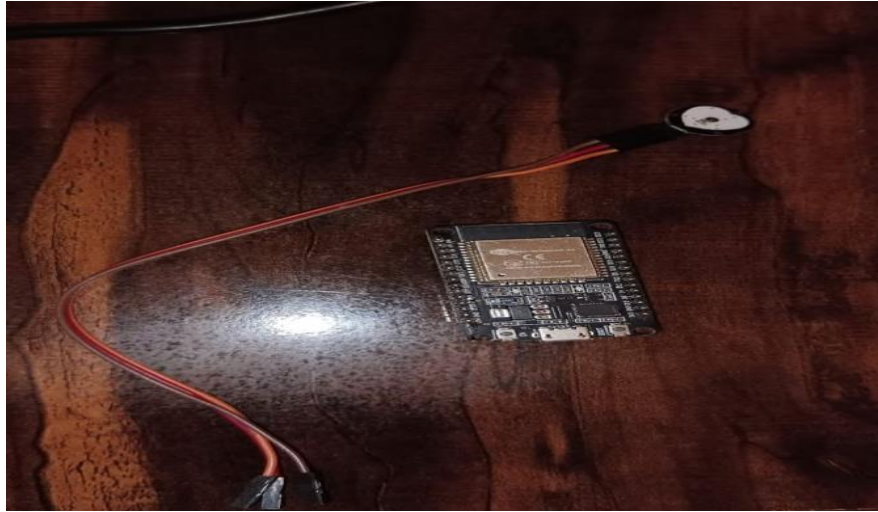


Figure 2: Outlook of hardware

On a breadboard, a testing board used to develop and test electronic circuits, jumper wires can be used to establish temporary connections between components. Both ends of the wires include connections that may be inserted into the breadboard's holes to connect various components. A total of three Female to Female jumper wires were used in the manifestation of the project.

4.2. FEASIBILITY ANALYSIS

4.2.1. Technical feasibility

Machine Learning algorithms are trained using historical data to detect anomalies in patient's data in partnership with the patient's vitals namely heart rate, blood pressure, and body heat. Wearables can collect this data and transmit it to the concerned model for the evaluation. The system then can generate the alerts whenever there is any anomaly in the data so that the doctors can treat them on time. There are many ML models that can be used to develop the system. For the purpose of identifying any anomalies, we have employed Regression analysis alongside support vector machines, and Random Forest in this instance.

4.2.2. Economic Feasibility

The cost of implementing such a system can vary depending on the scale and complexity of the project. However, as the cost of IoT devices has decreased in recent years, it is becoming increasingly affordable to deploy wearable sensors and other IoT devices. We have used streamlit to integrate ML and IoT on one platform. The return on investment (ROI) of such a system can be significant, as early detection and intervention can reduce healthcare costs associated with prolonged hospital stays and chronic illnesses.

4.2.3. Operational feasibility

The healthcare monitoring system should be easy to use and integrate into existing healthcare workflows. Healthcare professionals should be able to access the patient data easily and quickly, and the system should generate alerts in a timely manner. It is important to ensure that the system is reliable and available 24/7, as healthcare monitoring is a critical function. The system should also be scalable to accommodate increasing numbers of patients as the system expands.

4.2.4. Legal and regulatory analysis

Multiple rules and regulations regarding to patient privacy, data security, and medical device restrictions must be complied with by a healthcare monitoring system. Making sure, that the system is built to protect patient data and that it does indeed complies with all applicable laws, taking into account the Health Insurance Portability and Accountability Act (HIPAA) and the GDPR, of the European Union.

4.3. PROJECT PLAN

The project plan for developing a Healthcare monitoring system using Machine Learning and IoT is as follows:

1. Define project scope and objectives:

- Define the purpose and scope of the healthcare monitoring system, including the types of data to be collected, the patient population to be served, and the expected outcomes.
- Choose the key performance indicators (KPIs) that will be employed in evaluating how well the system is working.

2. Design the system architecture:

- Determine the hardware and software components needed to build the system, such as wearable sensors, IoT gateways, cloud infrastructure, and ML algorithms.
- Design the system architecture, including data flow, processing, and storage.

3. Develop the ML algorithms:

- Identify the ML algorithms that will be used to analyze the patient data, such as regression, classification, and clustering.
- Develop and train the ML algorithms using historical patient data.

4. Build the IoT infrastructure:

- Choose the appropriate IoT devices, such as wearables, and configure them to collect patient data.

- Build and configure the IoT gateway and connectivity infrastructure.

5. Integrate ML algorithms with IoT and cloud infrastructure:

- Integrate the ML algorithms with the IoT to process and analyze patient data.
- Configure the system to generate alerts when anomalies or patterns are detected.

5. SOFTWARE REQUIREMENT ANALYSIS

5.1. INTRODUCTION

For the very objective, of creating a successful healthcare monitoring system using ML and IoT, it is important to perform a software requirements analysis. The analysis helps to define the functional and non-functional requirements of the system and ensures that the system is developed in accordance with the desired specifications. Functional requirements refer to the features and functionalities of the system, such as data collection, processing, alert generation, user management, and reporting. Non-functional requirements refer to the system's performance, security, scalability, reliability, and usability.

In addition to functional and non-functional requirements, technical requirements are also important. Technical requirements refer to the hardware and software components needed to build the system, such as IoT devices, APIs, programming frameworks, ML techniques, and cloud infrastructure. The creation and operation of the healthcare monitoring system are heavily dependent on the technical requirements.

5.2. GENERAL DESCRIPTION

General description of the Software Requirement Analysis is as follows:

1. Functional Requirements:

- **Data Collection:** From IoT devices like wearables, the system should be able to gather patient data including heartbeat/second, body heat temperature.
- **Data Processing:** The system should be able to process the collected data and perform real-time analysis to detect patterns or anomalies that may indicate health issues.
- **Alert Generation:** The system should be able to generate alerts for healthcare professionals when abnormal data patterns are detected.
- **User Management:** The system should provide role-based access control to guarantee that patient data can only be accessed by authorized users.

- **Reporting:** The system should be able to generate reports and visualizations of patient data to help healthcare professionals monitor patient health and make informed decisions.

2. Non-Functional Requirements:

- **Security:** The system should be created to safeguard patient data and follow applicable laws like HIPAA and GDPR.
- **Reliability:** The system should be reliable and available 24/7 to ensure that patient data is monitored continuously.
- **Performance:** To provide swift identification and action, continuous live data processing and analyzing should be done by the system.
- **Scalability:** The system should be scalable to accommodate increasing numbers of patients as the system expands.
- **Usability:** The system should be easy to use and integrate into existing healthcare workflows.

3. Technical Requirements

- **IoT devices:** The system should support various IoT devices such as wearables, sensors, and other medical devices.
- **Cloud Infrastructure:** The system should use a cloud infrastructure to store and process patient data and perform ML analysis.
- **ML Algorithms:** The system should use various ML algorithms such as regression, classification, and clustering to analyze patient data.
- **Programming languages and frameworks:** Programming languages like Python and frameworks like TensorFlow, Scikit-learn, and Keras should be used to create the system.

5.3. SPECIFIC REQUIREMENTS

1. Python:

Python is an extensible language for programming that may be used for a wide range of tasks, including machine learning, data analysis, web development, and scientific computing. A developer community that actively contributes to its growth and manages a huge library of third-party packages that increase its functionality.

2. Google Colab:

A cloud-based development environment called Google Colab (short for Google Collaboratory) is provided by Google to create cloud-based Jupyter notebooks. It provides free access to a virtual machine that has pre-installed tools such as Python, Jupyter notebooks, and TensorFlow, as well as access to GPU and TPU resources for running deep learning models. Users can access Colab through a web browser and can collaborate with others in real time, making it a useful tool for teaching, research, and prototyping.

3. Scikit-Learn:

Scikit-learn offers a number of machine learning tools and methods for tasks including grouping and creating meaningful cluster maps.

4. Stream-lit:

Stream-lit is a powerful and easy-to-use framework for building data-driven web applications in Python, making it a popular choice for data scientists, developers, and researchers.

6. DESIGN

6.1. SYSTEM DESIGN

Data collection from IoT devices, AI data processing, an online overview for data visualization, and research and interpretation are commonly included in the system design of a healthcare monitoring system integrating ML and IoT. Here is a high-level overview of the system design:

- **Data Collection:**

Data collection from wearables, sensors, and other connected IoT devices is the first phase in system design. This data is typically collected using an IoT platform that supports secure and reliable data transmission. The data can include vital signs, activity levels, medication adherence, and other relevant health data.

- **Data Processing and Analysis:**

Once the data is collected, it is preprocessed and cleaned to ensure it is ready for analysis. This involves removing any noisy or irrelevant data points, and converting the data into a format suitable for machine learning algorithms. The data is then analyzed using ML algorithms such as classification, clustering, or regression to identify patterns and insights in the data.

- **Model Deployment:**

For real-time analysis of new information, the ML models are deployed to a cloud-based environment. The deployment may be carried out utilizing serverless computing platforms like AWS Lambda or Google Cloud Functions, or containerization tools like Docker.

- **Web-Based Dashboard:**

A web-based dashboard is developed to display the results of the data analysis and provide insights to healthcare providers and patients. The dashboard is typically built using web development frameworks such as Flask or Django, and data visualization libraries such as Plotly or Bokeh. The dashboard can be customized to display various types of information, including graphs, charts, and tables.

- **Alerting and Notifications:**

The system may be designed to provide alerts and notifications to healthcare providers and patients based on certain thresholds or events. These alerts can be sent via email, SMS, or push notifications to mobile devices.

- **Security and Privacy:**

Finally, the system must be designed in such a sophisticated manner that allows for the complete security and privacy of the individual's data. This involves implementing appropriate access controls, data encryption, and other security measures to protect patient data from unauthorized access or disclosure.

6.2. DESIGN NOTATION

A medical surveillance system using ML and IoT can be represented using a variety of design notations. The Unified Modeling Language (UML), an internationally recognized notation used to describe software systems, is one popular notation.

1. **Use Case Diagram:**

The system for monitoring healthcare can be described in detail using a use case diagram. The monitoring of vital signs, data analysis, and alarm generation may all be use cases for the wellness monitoring system.

2. **Class Diagram:**

The data models and connections between the various parts of the healthcare monitoring system can be portrayed using a class diagram.

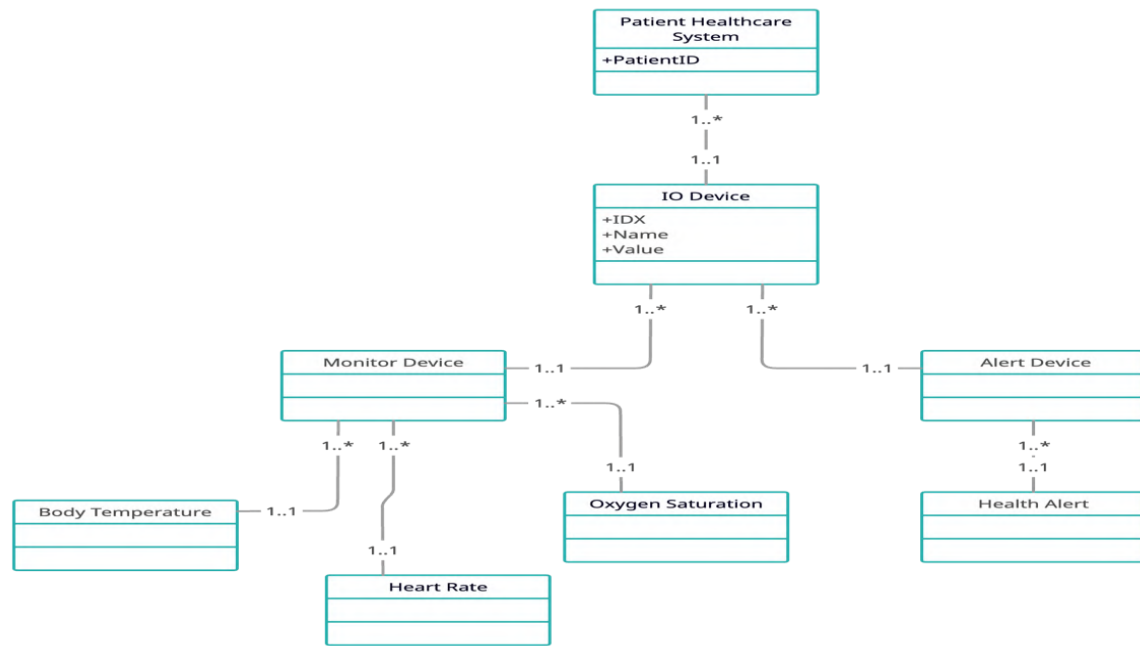


Figure 3: Class diagram of the system

3. Sequence Diagram:

A sequence diagram can be used to represent the interactions between the different components of the healthcare monitoring system. The sequence diagram may show how data is collected from the sensors, processed by the data analytics component, and used to generate alerts.

4. Component Diagram:

A component diagram can be used to represent the physical and logical components of the healthcare monitoring system, including the IoT devices, the data analytics component, and the alert system.

5. Deployment Diagram:

A deployment diagram can be used to represent the physical deployment of the healthcare monitoring system, including the IoT devices, the data analytics component, and the cloud infrastructure used for storage and processing.

6.3. FLOWCHART

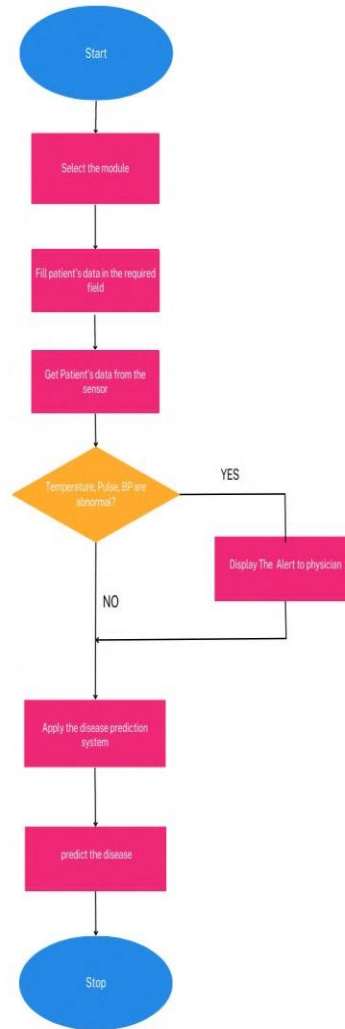


Figure 4: Flow Chart of the system

6.4. PSEUDO CODE

Basic pseudo code for the proposed system is as follows:

- Set up Internet of Things (IoT) devices to start gathering the patient's vital indicators (heart rate, blood pressure, and temperature).
- Set up data acquisition component to collect and store data from IoT devices in a database.
- Implement data preprocessing component to clean and transform the data (remove missing values, normalize, etc.).
- Train a machine learning model using preprocessed data to predict patient's health status.
- Set up an alert system to trigger alarms when abnormal vital sign readings are detected by the ML model.

- In order to show the patient's vital signs and alerts in real time, provide a user interface (dashboard).
- Deploy the model on cloud infrastructure for scalability and reliability.
- Continuously monitor the system and collect feedback to improve its accuracy and performance.

7. TESTING

7.1. FUNCTIONAL TESTING

Functional assessment of a healthcare monitoring system involves testing the system's core functionalities to ensure they work as expected:

1. Data Collection Testing:

- Verifying that the IoT devices are able to collect patient's vital signs accurately.
- Testing the data acquisition component to ensure it collects and stores data in the database correctly.

2. Data Pre-processing Testing:

- Validating that the data pre-processing component cleans and transforms data correctly (e.g., removes missing values, normalizes data, etc.).
- Verifying that the pre-processed data is accurate and suitable for training the ML model.

3. Machine Learning Testing:

- Testing the trained ML model to ensure it predicts patient's health status accurately.
- Verifying that the ML model produces consistent and reliable results.

4. Alert System Testing:

- Verifying that the alert system triggers alarms when abnormal vital sign readings are detected by the ML model.
- Testing the alert system's response time to ensure it is timely and effective.

5. User Interface Testing:

- Checking that the dashboard's user interface (which shows the patient's vital signs and alarms in real time) is working as anticipated.
- Evaluate the usability and responsiveness of the user interface.

6. Cloud Infrastructure Testing:

- Testing the system's scalability and reliability on cloud infrastructure.
 - Ensuring the system is able to handle hefty amounts of queries from multiple users.
7. Security and Privacy Testing:
- Testing the system's security features to ensure patient's data is protected from unauthorized access.
 - Verifying that the system complies with privacy regulations and standards

7.2. STRUCTURAL TESTING

Structural testing of a healthcare monitoring system involves testing the system's internal components and logic:

1. Unit Testing:
 - Testing each software component (e.g., IoT devices, data pre-processing, ML model, alert system, user interface) in isolation to ensure they function correctly.
 - Using test cases to cover all possible scenarios and inputs.
 - Validating that each component meets its specific requirements and specifications.
2. Integration Testing:
 - Checking the integration of various software parts to make sure they function as intended.
 - Verifying that the communication and data exchange between components is accurate and reliable.
 - Validating that the integration meets the system's overall requirements and specifications.
3. System Testing:
 - Assessing the system as a whole to make sure it adheres to general standards and criteria.
 - Using test cases to cover all possible user scenarios and inputs.
 - Validating that the system's functionalities and features work together seamlessly.
4. Performance Testing:
 - Evaluating the system's performance under various loads to make sure it can manage the anticipated data volume and user traffic.

- Verifying that the system's response time and resource utilization are within acceptable limits.
5. Security and Compliance Testing:
- Testing the system's security features to ensure patient's data is protected from unauthorized access.
 - Verifying that the system complies with privacy regulations and standards (e.g., HIPAA).
 - Validating that the system's authentication and access control mechanisms are effective.

7.3. LEVELS OF TESTING

The Four crucial four levels of testing

1. Unit Testing:
 - Individual software components are tested such as data pre-processing, machine learning algorithms, and the alert system.
 - Verifying that each component functions correctly and produces the expected results.
2. Integration Testing:
 - Testing how different software components integrate and work together.
 - Verifying that the system is functioning correctly as a whole.
3. System Testing:
 - Testing the entire system and verify that it meets the specified requirements.
 - Verifying that the system functions correctly with external systems, such as databases and cloud infrastructure.
4. Acceptance Testing:
 - Running the system through real-world tests to make sure it satisfies end-user requirements.
 - Checking to see if the system complies with the needs and requirements outlined in the project plan.

7.4. TESTING THE PROJECT

Individuals who wore a wearable device for a week in order to continuously collect data on their vital signs were asked to provide their vitals in order to evaluate the IoT-based health monitoring system. SVM, Random Forest Classifier, Logistic Regression.

The findings indicate that the system is precise and has an average precision of 90%. This implies that it can make an almost accurate forecast about a patient's health condition by analyzing their vitals. Besides, the system allows real-time tracking and examination of vital signs, which assists medical professionals in identifying and addressing any variations in a patient's health condition promptly.

Table 1. The score of evaluation metrics for machine learning models

Disease	ML Algorithm Used	Accuracy	Precision	Recall	F1-Score
Heart Disease	Logistic Regression	0.859	0.878	0.828	0.852
	SVM	0.803	0.848	0.8	0.823
	RF Classifier	0.819	0.818	0.843	0.83
Parkinson's Disease	Logistic Regression	0.897	0.966	0.906	0.935
	SVM	0.92	0.967	0.937	0.952
	RF Classifier	0.897	0.937	0.937	0.937

Diabetes	Logistic Regression	0.759	0.518	0.717	0.602
	SVM	0.803	0.848	0.8	0.823
	RF Classifier	0.746	0.518	0.682	0.589

For predictive systems, we utilized Support Vector Machine SVM, Random Forest Classifier, and Logistical Regression. The results pointed out that the top-quality performance for the Cardiac Disease Detection model was when Logistic Regression was used, as well as the best performance for the Parkinson's Disease Detection model and the best performance for the Diabetes Detection model when SVM was used.

8. IMPLEMENTATION

8.1. IMPLEMENTATION OF THE PROJECT

Prominent algorithms for machine learning including SVM, Logistical Analysis and Regression, and technique of Random Forest are used by the proposed IoT-based wellness tracking system for analysis. The method involved the following steps:

- The patient's vital signs data were gathered using sensors at regular intervals and exported to a Internet-based platform for keeping the record.
- The acquired vitals were cleaned up to remove any unnecessary or inconsistent data points before using the machine learning algorithms. Thus, filtering up the data sets, to be free of any kind of noise and outliers to ensure the correctness of the system.
- Training the Machine Learning algorithms: The cleaned and filtered data was used to train the SVM (Support Vector Machine), Logistic Regression, and Random Forest ML algorithms.
- Based upon the information gathered, the algorithms were programmed to forecast the patient's state of health.

- Usage of trained algorithms on a cloud-based platform enables real-time health monitoring and analysis. The platform collects data and provides insights into the patient's health status. No information has been omitted from the original text.
- Alert system: The system was designed to alert healthcare professionals in case of any abnormal readings. The alert system was based on predefined thresholds, and healthcare professionals were notified if the patient's vital signs crossed these thresholds.

The suggested technique is an affordable, trustworthy, and user-friendly answer for monitoring and assessing health. Patients can use the system at their residences, which reduces their visits to medical centers. The algorithms based on machine learning employed in the system can offer precise forecasts about a patient's health situation. This helps medical experts to take necessary measures for the prevention or treatment of health issues.

8.2. CONVERSION PLAN

A conversion plan for a healthcare monitoring system involves transitioning from the current system to the new system while minimizing disruption to operations. The steps involved in the conversion plan is as follows:

1. Planning Stage:
 - Assessing the current system and identify its strengths and weaknesses.
 - Defining the goals and objectives for the new system.
 - Developing a timeline for the transition process.
2. Communication Stage:
 - Communicating the conversion plan to all stakeholders, including patients, healthcare professionals, and technical staff.
 - Providing training and support to users of the new system.
3. Testing Stage:
 - Verifying the new system passes the requirements and specifications provided in the project plan by testing it in a confined environment.
 - Conducting a pilot test with a small group of patients and healthcare professionals to identify any issues.
4. Rollout Stage:
 - Implementing the new system in phases to minimize disruption to operations.

- Monitoring the system during the rollout to identify and address any issues.
5. Evaluation Stage:
- Evaluating the effectiveness of the new system by comparing it with the previous system.
 - Soliciting feedback from users to identify areas for improvement.

8.3. POST-IMPLEMENTATION & SW MAINTENANCE

Software maintenance and post-implementation are essential stages of the healthcare monitoring system. These phases involve ensuring that the system is functioning correctly, meeting the requirements of the users, and addressing any issues that arise. Here are some key activities involved in post-implementation and software maintenance:

1. Monitoring and Evaluation:
 - Monitor the system to ensure that it is functioning correctly.
 - Evaluate the system's performance against the defined goals and objectives.
2. Bug Fixing and Issue Resolution:
 - Identify and address any issues that arise with the system.
 - Prioritize and fix any bugs that impact the system's functionality or user experience.
3. Upgrades and Enhancements:
 - Identify opportunities to improve the system and implement upgrades and enhancements.
 - Plan and implement upgrades and enhancements in a structured and controlled manner.
4. Backup and Recovery:
 - A well-planned backup recovery plan, ensures the safety of the data in the worst-case scenario of a system breakdown.
 - Monitor it frequently to ensure the correctness of the failsafe program put in place.

9. PROJECT LEGACY

9.1. CURRENT STATUS OF THE PROJECT

A wearable device, an access point device, and a cloud-based platform make up the three primary parts of the proposed system. Patients' real-time vital sign data, such as blood pressure, is

collected using the wearable device. A microcontroller is used by the sensor-equipped device to process and communicate the vital sign data to the gateway device.

9.1.1. Real-Time Data Collection Using IoT

The gateway device is responsible for receiving the vital sign data transmitted by the wearable device and sending it to the cloud-based platform for analysis [11]. The gateway device is equipped with a Wi-Fi module or a cellular module to enable data transfer. The gateway device also includes a microcontroller to process the data before transmitting it to the cloud-based platform.

9.1.2. Applying Machine Learning Technique

The cloud-based platform is responsible for hosting the machine learning algorithm that processes and analyses the vital sign data collected by the wearable device. The platform can be hosted on a cloud server, which provides scalable and secure storage and computation capabilities. The platform is made to take vital sign data sent by the gateway device and process it using algorithms for adaptive algorithmic learning like SVM, Logistical Analysis and Regression, and art of Random Forest in real-time.

For developing a machine learning algorithm, there are two essential components: data preprocessing and data analysis. The first phase, data pre-processing, includes three crucial steps: data cleaning, feature extraction, and normalization. The data cleaning process involves checking for and correcting any missing or corrupted values. Finding important features from the raw data, such as heart-beats per minute, measure of body heat temperature, is known as feature extraction. Normalization ensures that each feature contributes equally to the analysis by scaling the features to a common range.

During the data analysis phase, pre-processed data goes through machine-based learning algorithms like, SVM, random forest Technique and logistic regression. These algorithms, trained on historical data, detect patterns and correlations between the patient's health status and vital sign data. Following training, the algorithms can predict the patient's health status using the information gathered. A real-time dashboard shows the monitoring and analysis of the patient's health state after the analysis outcomes are provided to the gateway device.

9.2. REMAINING AREAS OF CONCERN

The medical field has seen outstanding technological advancement in recent years, as well as in its use to address healthcare-related real-world problems. Due to this, healthcare services have

substantially improved and are now available at the tip of the finger and most importantly from the comfort of our homes. However, it still has some areas of concern which is discussed below:

- **Servicing & Maintenance Costs-** High maintenance & servicing, continuous upgradation costs.
- **Power Consumption-** High-power cells power the majority of IoT devices. The battery is almost infeasible to be able to be actually replaced after the complete installation of a sensor. However, research is still ongoing to find an efficient yet cheap alternative to match the demands of the current scenario.
- **Data Privacy and Security-** The notion of continuous surveillance has changed as a result of cloud computing integration. However, this has also increased the fragility of healthcare networks to cyberattacks. IoT security services may be compromised if there is a problem by fusing nimble and safe algorithms and cryptographies into a more secure environment, this issue might be tackled efficiently.
- **Scalability-** Manifesting such a device which has unparalleled scalability factor, is indeed necessary.
- **Identification-** Doctors and nurses simultaneously manage a number of patients as well as caregivers. Thus, to shorten the delay in the delivery of effective medical attention, it is essential that within a single treatment process, the identities of the patient, caregiver, and doctors are exchanged.
- **Self-Configuration-** IoT devices need to provide users more control by incorporating features like customized setup. The users will be able to adjust the system parameters in response to application demands and changing surroundings thanks to this.
- **Continuous Monitoring-** In the case of live monitoring situations, the IoT device must be capable of doing real-time monitoring.
- **Exploration of New Diseases-** A steadily increasing magnitude of healthcare applications are being developed every day as a result of the swift improvement of mobile technology.
- **Environmental Impact-** The establishment of an IoT system needs the union of numerous biological sensors with semiconductor-heavy hardware. Therefore, more study must be done to develop sensors made of biodegradable material, such that the materials used do not cause any harm to the user as well as the planet.

9.3. TECHNICAL AND MANAGERIAL LESSONS LEARNT

Presented here are technical and management lessons learned throughout the project:

9.3.1 technical lessons

1. **Proper Data Management:** For machine learning (ML) models to be trained, healthcare monitoring systems integrating IoT and ML need a lot of data. In order to ensure that the data is accurate, full, and safe, good data management is crucial.
2. **Integration Challenges:** Integration of various hardware and software components can pose significant technical challenges. It is important to ensure that all components work together seamlessly to ensure the system's reliability.
3. **Choosing Appropriate ML Algorithms:** Choosing appropriate ML algorithms for healthcare monitoring systems can be challenging. Therefore, careful evaluation and selection of ML algorithms are necessary to ensure the accuracy of the system's predictions.
4. **Performance Optimization:** Healthcare monitoring systems are often time-sensitive, and real-time predictions are essential. Therefore, performance optimization is critical to ensure that the system is fast enough to provide predictions in real-time.

9.3.2 managerial lessons

1. **Team Collaboration:** It is vital for team members to work together to make the project a success. Collaboration and effective communication can guarantee that the project is finished on schedule and to the satisfaction of all parties involved.
2. **Risk Management:** To make sure that possible dangers are recognized and reduced before they become challenges, risk management is crucial. The project can stay on track and problems can be found and fixed right away with the help of good risk management.
3. **Clear Project Goals:** To keep the project on schedule and satisfy the needs of the stakeholders, it is crucial to have clear project goals and objectives. Setting clear objectives is one way to keep the project on track and guarantee that everyone on the team is pursuing the same objective.
4. **User Involvement:** Involving users in the project can help to ensure that the final product meets their needs. User feedback can help to identify potential issues and areas for improvement.

10. USER GUIDE

10.1. INTRODUCTION

The healthcare monitoring system using IoT and ML is a powerful tool for monitoring patients' health conditions and predicting potential health issues. This system combines IoT sensors and ML algorithms to provide real-time health monitoring and personalized healthcare services. In this user guide, we will walk you through the process of using the healthcare monitoring system and how to take advantage of its features.

10.2. SYSTEM REQUIREMENTS

To use the healthcare monitoring system, you will need the following requirements:

- A computer or smartphone with internet connectivity.
- Usage of a web browser, such as Safari, Mozilla Firefox, or Google Chrome.
- A user account on the healthcare monitoring system platform.

10.3. GETTING STARTED

To get started with the healthcare monitoring system, follow these steps:

1. Open your web browser and go to the healthcare monitoring system platform.
2. Log in to your user account using your credentials.
3. Once you are logged in, you will be directed to the dashboard.
4. On the dashboard, you can view all the available features and data related to your health.

10.4. USING THE SYSTEM

After logging in to the healthcare monitoring system, we side bar from which we can choose the disease for which we need to predict.

We have 3 type of disease prediction:

1. Diabetes Prediction
2. Heart Disease Prediction
3. Parkinson's Disease Prediction

10.4.1. Diabetes prediction

To monitor your health for diabetes, you need to fill the data asked in the dashboard or your physician will fill the data based on previous tests.

1. **Age of the person:** Fill in the age of the patient.
2. **Number of pregnancies:** If the patient has history of pregnancy, then fill in the value else fill 0.
3. **Glucose Level:** The quantification of measure of glucose, a form of sugar, in the blood at any point in time. Enter the blood glucose level (in mg/dL).
4. **Blood Pressure:** In mmHg, enter the blood pressure reading.
5. **Skin Thickness Value:** The term "skin thickness value" relates to the skin's thickness, which is often expressed in millimeters. Skin thickness is typically measured by taking a fold of skin and subcutaneous fat at a specific site on the body, such as the back of the upper arm or the abdomen and measuring the thickness of this fold with calipers. A higher reading can point to more subcutaneous fat, which has been linked to type 2 diabetes risk and insulin resistance. Input the Skin Thickness (in mm) field.
6. **Insulin Level:** Medical personnel may check both blood glucose and insulin levels when diagnosing diabetes. In the event a person has type 1 diabetes, their insulin levels have a tendency to be low because their body does not produce enough insulin to match the threshold criteria. However, with time, blood glucose levels could increase and insulin production could decrease. Enter the amount of insulin (in mcU/mL).
7. **BMI Value:** According to research, potential of type 2 diabetes surges by being overweight or obese, and their BMI can serve as a gauge of that risk. A BMI of 25 or more is usually considered as overweight, and a BMI of 30 or more is regarded as obese. These BMI scores had been attributed to a higher risk of type 2 diabetes.
8. **Diabetes Pedigree Function Value:** A score known as the diabetes pedigree function (DPF) is used to calculate an individual's risk for having type 2 diabetes based on their family's history of diseases. Each member of the family who has diabetes is given a point value, which is then added up to determine the DPF score.

The DPF score can range from 0 to 2.5 or higher, with a higher score indicating a greater likelihood of developing type 2 diabetes. A score of 0 indicates no family history of diabetes, while a score of 2.5 or higher indicates a strong family history of diabetes.

After filling all the required fields, click on the Diabetes Test Result to check if the patient is diabetic or not.

10.4.2. Heart disease prediction

To monitor your health for heart disease, you need to fill the data asked in the dashboard or your physician will fill the data based on previous tests.

1. **Age:** Fill in the age of the patient.
2. **Sex:** Fill in the sex of the patient (male/female).
3. **Chest Pain Types:** Chest pain is a common symptom of many types of heart disease, and the type of chest pain can sometimes provide clues about the underlying condition. Angina is the most typical form of chest pain linked to heart disease, which is often described as a squeezing, pressure, or tightness in the chest that can radiate to the neck, jaw, shoulders, or arms. Angina is typically triggered by physical activity or emotional stress, and can be relieved by rest or medication.

Other types of chest pain associated with heart disease may include:

- Heart attack pain:

This type of pain is usually more severe and intense than angina, and may be described as a crushing, heavy, or constricting feeling in the chest that may radiate to the back, neck, jaw, or arms. Other symptoms including shortness of breath, nausea, or dizziness may also be present.

- Pericarditis pain:

Inflammation of the membrane surrounding the heart known as pericarditis, can cause piercing chest discomfort that may be alleviated by sitting up straight and bending forward. Coughing or hard breathing may make the pain worse.

- Aortic dissection pain:

An exceptionally risky of a disturbance known as aortic dissection occurs when the inner layer of the aorta breaks, and fills up the inner layer of the aortic membrane with blood. This can cause severe, tearing chest pain that may radiate to the back, neck, or abdomen.

4. **Resting Blood Pressure:** Among the numerous factors that may be taken into account in determining the risk of developing heart disease is resting blood pressure, which is the measurement of blood pressure measured while a person is at rest.

Resting blood pressure is one of the primary indicators of hypertension, and individuals with consistently high blood pressure readings may be at increased risk for developing heart disease.

5. **Serum Cholesterol:** Serum cholesterol is one of the many factors that may be considered in predicting the risk of developing heart disease. A heightened chance of developing heart disease has been associated with high blood cholesterol.
A blood test can measure serum cholesterol levels, including LDL cholesterol, high-density lipoprotein (HDL) cholesterol, and total cholesterol. The American Heart Association recommends that adults have their cholesterol levels checked regularly, starting at age 20. Measure your serum cholesterol level (in mg/dl).
6. **Fasting Blood Sugar:** Fasting plasma glucose is another name for fasting blood sugar, is a measurement of the amount of glucose (sugar) in the blood after a period of fasting, usually for at least 8 hours. Elevated fasting blood sugar levels can be an indicator of developing type 2 diabetes. Research studies have found that individuals with higher fasting blood sugar levels may be at a greater risk for having atherosclerosis. Elevated fasting blood sugar levels have also been linked to other risk factors for heart disease, such as hypertension, dyslipidemia, and inflammation.
7. **Exercise Induced Angina:** Exercise-induced angina is a symptom of coronary artery disease (CAD) that occurs when the heart doesn't get enough oxygen-rich blood during physical activity, leading to chest pain or discomfort.
Then it comes to predicting the risk of heart disease, exercise-induced angina can be a significant indicator. A history of exercise-induced angina suggests that an individual may have underlying CAD or other heart-related conditions that increase their risk of developing cardiovascular disease. Exercise stress testing, which involves monitoring the heart during exercise, is a standard technique for diagnosing CAD, especially exercise-induced angina, and determining its existence and severity.
8. **ST depression induced by exercise:** ST depression induced by exercise is a common finding on an electrocardiogram (ECG) test performed during an exercise stress test. During an exercise stress test, the patient is asked to perform physical exercise while the ECG is being monitored. ST depression occurs when the ST segment on the ECG tracing is lower than the baseline, indicating reduced blood flow to the heart muscle during physical activity.
Exercise-induced ST depression is general in nature, which means that it can be brought on by a number of different illnesses, including heart disease as well as other issues like electrolyte imbalances, drug side effects, and respiratory issues. However, in the context of other clinical

and diagnostic findings, ST depression induced by exercise can be a useful tool in predicting the risk of heart disease.

9. **Fluoroscopy:** Fluoroscopy can be used to guide certain heart procedures, such as cardiac catheterization, angioplasty, or stent placement. By providing real-time images of the heart and blood vessels, fluoroscopy can help doctors visualize the procedure and ensure that the instruments are properly positioned.

After filling all the required fields, click on the Heart Disease Test Result to check if the patient has heart disease or not.

10.4.3. Parkinson's disease prediction

To monitor your health for Parkinson's Disease, you need to fill in the data asked in the dashboard or your physician will fill the data based on previous tests.

1. **Age:** Fill in the age of the patient.
2. **Sex:** Fill in the sex of the patient (male/female).
3. **MDVP:** MDVP stands for Multidimensional Voice Program, which is a tool that can be used to analyze various aspects of a person's voice, including pitch, tone, and variability. It has been discovered that some MDVP measurements help predict the existence or severity of Parkinson's disease.

A disease that is purely a neurological condition and one that worsens over time and can cause impaired speech and movement, voice changes, such as a harsh or monotonous tone are common in Parkinson's disease and can be an early symptom of the condition.

One specific measure that has been studied in relation to Parkinson's disease is the MDVP jitter measure, which reflects the variability in the timing of vocal fold vibrations during speech. In some studies, increased jitter has been associated with a higher likelihood of having Parkinson's disease or a greater severity of symptoms.

4. **Shimmer:** Shimmer is a measure of the variability in vocal fold amplitude during speech, and it has been studied as a potential biomarker for Parkinson's disease. This disease is a neurodegenerative disorder that impairs movement, and early diagnosis and treatment can be important for managing symptoms and improving quality of life.

In Parkinson's disease, changes in the neural control of speech and voice can lead to variations in vocal fold movement and amplitude, which can be quantified using measures such as shimmer. Some studies have suggested that higher levels of shimmer may be associated with

greater severity of Parkinson's disease symptoms, although this relationship is still being studied and is not yet fully understood.

5. **NHR:** NHR stands for Nigrosome-to-pons ratio, which is a relatively new imaging biomarker that has been studied as a potential tool for predicting and diagnosing Parkinson's disease.

The substantia nigra, the most important sphere of the brain for the functionality of the body is affected in Parkinson's disease, contains structures called nigrosomes that can be imaged using high-resolution MRI. The Nigrosome-to-pons ratio is the ratio of the signal intensity of the nigrosome to the signal intensity of a nearby structure called the pons, which can be used to assess the health of the nigrosome.

Research has suggested that a lower Nigrosome-to-pons ratio may be associated with an increased risk of Parkinson's disease. In addition, changes in the Nigrosome-to-pons ratio over time may also be used to track disease progression and the effects of treatment.

6. **HNR:** HNR stands for Harmonic-to-Noise Ratio, and it is a measure used in the analysis of voice and speech signals. In Parkinson's disease, changes in the voice and speech are common and can include changes in pitch, loudness, and speech rate, as well as changes in voice quality such as hoarseness or breathiness.

Studies have shown that the Harmonic-to-Noise Ratio (HNR) can be used as a quantitative measure of voice quality in people with Parkinson's disease. In particular, lower HNR values may be associated with more severe dysphonia, or difficulty speaking. Therefore, measuring the HNR can be a useful tool in assessing the severity of voice and speech changes in people with Parkinson's disease.

7. **RPDE:** RPDE stands for Recurrence Period Density Entropy, which is a feature of signal processing that can be used to analyze patterns in time-series data, such as those obtained from recordings of brain activity. In the context of Parkinson's disease prediction, RPDE has been explored as a potential biomarker of disease progression and severity.

Research studies have suggested that RPDE may be altered in individuals with Parkinson's disease, with higher values indicating greater disruption in the regularity of patterns in brain activity. Specifically, higher RPDE values have been associated with more advanced disease stages and greater motor impairment.

8. **DFA:** DFA stands for De-trended Fluctuation Analysis, which is a mathematical method used to analyze the time series data of signals such as brain waves or physiological signals like heart rate

variability. DFA has received attention as a possible technique for Parkinson's disease diagnosis in recent years.

The degeneration of brain cells that produce dopamine is an indicator of Parkinson's disease, a neurodegenerative condition that impairs mobility. Studies have shown that DFA analysis of signals such as gait and tremor data may be able to identify specific patterns that are associated with Parkinson's disease.

After filling all the required fields, click on the Parkinson's Test Result to check if the patient has Parkinson's disease or not.

10.5. REVIEW ALERTS

When an alert is generated, healthcare professionals should take immediate action to address the issue. This may involve contacting the patient directly, adjusting their medication, or scheduling an in-person visit.

10.6. CUSTOMIZABLE SYSTEM

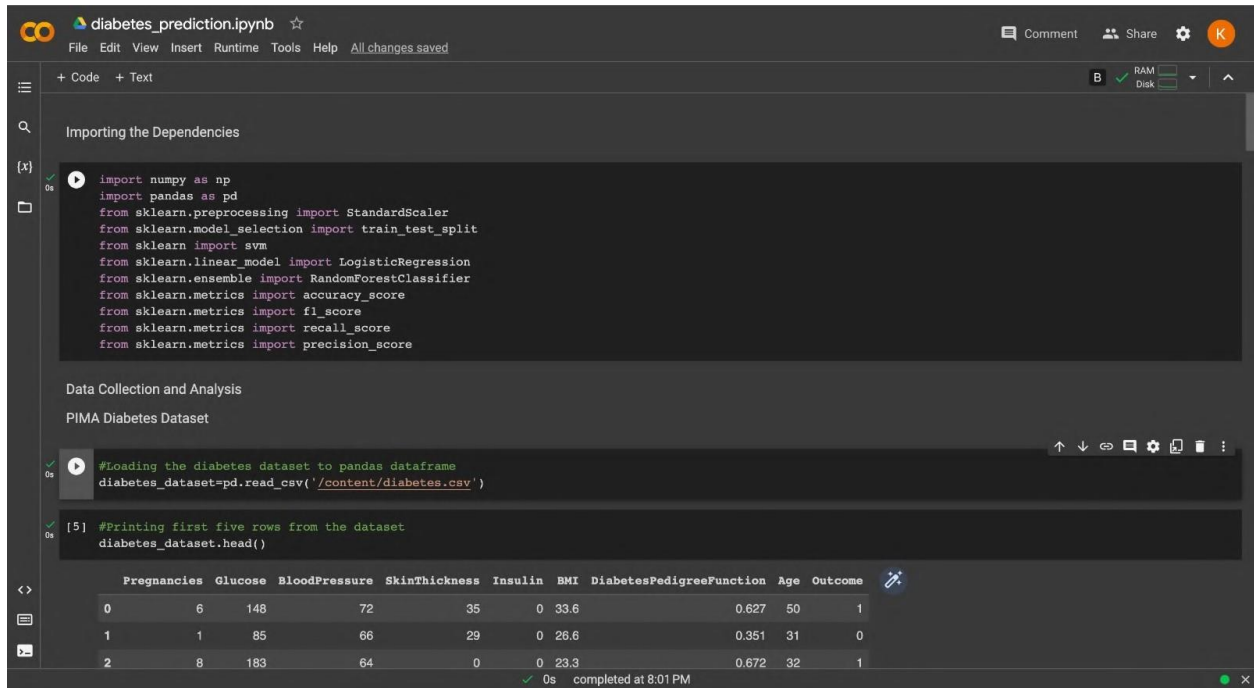
The healthcare monitoring system using IoT and ML is customizable to meet the unique needs of healthcare professionals and patients. The system can be configured to monitor specific patient conditions, generate alerts at different thresholds, and provide customized reports.

10.7. CONCLUSION

The healthcare monitoring system using IoT and ML is a powerful tool for monitoring your health and predicting potential health issues. One could modify their healthcare services to suit their specific needs and desires and also make the most of the system's capabilities by adhering to this well-planned user manual. Remember to regularly check your health status and predictive analysis to stay ahead of potential health issues

11. SOURCE CODE

11.1. DIABETES PREDICTION



```
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.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
```

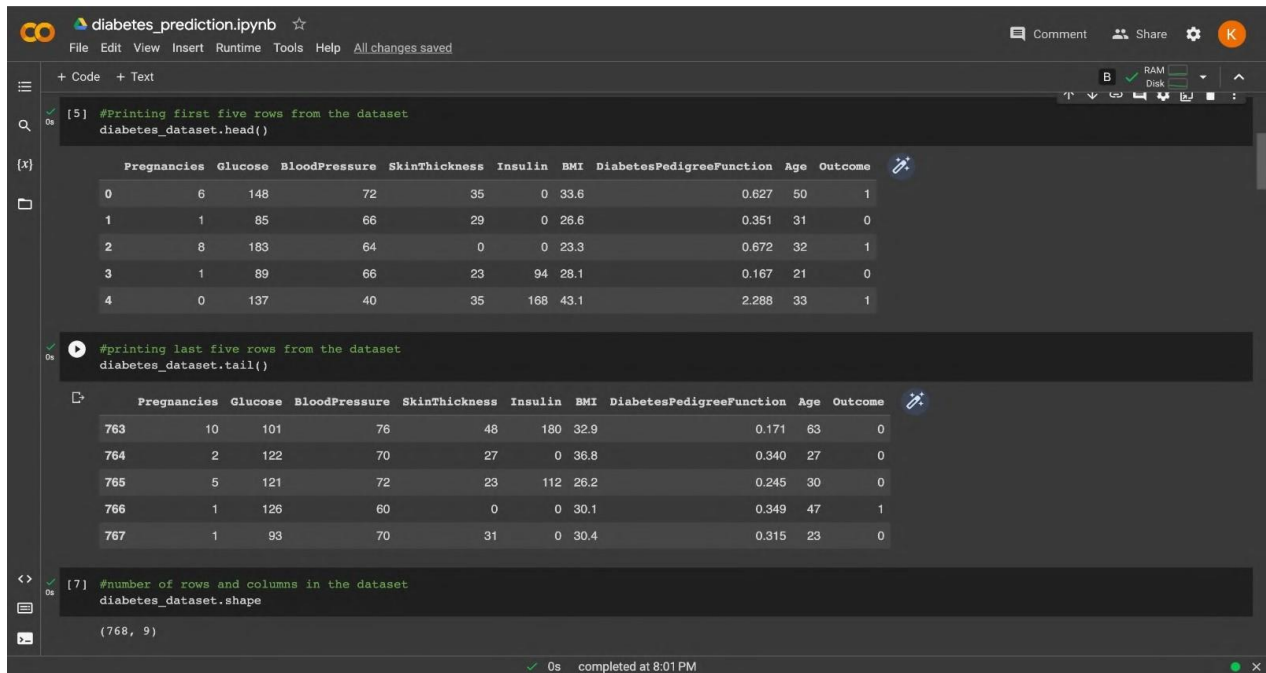
Data Collection and Analysis

PIMA Diabetes Dataset

```
#Loading the diabetes dataset to pandas dataframe
diabetes_dataset=pd.read_csv('/content/diabetes.csv')
```

```
[5] #Printing first five rows from the dataset
diabetes_dataset.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1



```
#printing last five rows from the dataset
diabetes_dataset.tail()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
[7] #number of rows and columns in the dataset
diabetes_dataset.shape
```

(768, 9)

```
diabetes_prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
B RAM Disk

#getting the statistical measure of the dataset
diabetes_dataset.describe

<bound method NDFrame.describe of
0      6      148      72      35      0  33.6
1      1      85      66      29      0  26.6
2      8     183      64      0      0  23.3
3      1      89      66      23     94  28.1
4      0     137      40      35    168  43.1
..     ...     ...     ...     ...     ...
763    10     101      76      48    180  32.9
764     2     122      70      27     0  36.8
765     5     121      72      23    112  26.2
766     1     126      60      0      0  30.1
767     1      93      70      31      0  30.4

DiabetesPedigreeFunction  Age  Outcome
0      0.627      50      1
1      0.351      31      0
2      0.672      32      1
3      0.167      21      0
4      2.288      33      1
..     ...     ...     ...
763    0.171      63      0
764    0.340      27      0
765    0.245      30      0
766    0.349      47      1
767    0.315      23      0

[768 rows x 9 columns]>

[9] diabetes_dataset['Outcome'].value_counts()

0      500
1      268
Name: Outcome, dtype: int64

0s completed at 8:01 PM
```

```
diabetes_prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
B RAM Disk

0 -> Non-Diabetic
1 -> Diabetic

[10] diabetes_dataset.groupby('Outcome').mean()

Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age
Outcome
0      3.298000  109.980000      68.184000      19.664000      68.792000  30.304200      0.429734  31.190000
1      4.865672  141.257463      70.824627      22.164179     100.335821  35.142537      0.550500  37.067164

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

print(X)

Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  \
0      6      148      72      35      0  33.6
1      1      85      66      29      0  26.6
2      8     183      64      0      0  23.3
3      1      89      66      23     94  28.1
4      0     137      40      35    168  43.1
..     ...     ...     ...     ...     ...
763    10     101      76      48    180  32.9
764     2     122      70      27     0  36.8
765     5     121      72      23    112  26.2
766     1     126      60      0      0  30.1
767     1      93      70      31      0  30.4

0s completed at 8:01 PM
```

```
diabetes_prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
0s [12] DiabetesPedigreeFunction Age
0 0.627 50
1 0.351 31
2 0.672 32
3 0.167 21
4 2.288 33
.. ..
763 0.171 63
764 0.340 27
765 0.245 30
766 0.349 47
767 0.315 23

[768 rows x 8 columns]

0s print(Y)
0 1
1 0
2 1
3 0
4 1
..
763 0
764 0
765 0
766 1
767 0
Name: Outcome, Length: 768, dtype: int64

Data Standardization

0s [14] scaler = StandardScaler()
0s completed at 8:01 PM
```

```
diabetes_prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
0s [15] scaler.fit(X)
StandardScaler
StandardScaler()

0s [16] standardized_data = scaler.transform(X)

0s print(standardized_data)
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
  1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
 -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
 -0.10558415]
 ...
 [ 0.3429808  0.00330087  0.14964075 ... -0.73518964 -0.68519336
 -0.27575966]
 [-0.84488505  0.1597866 -0.47073225 ... -0.24020459 -0.37110101
  1.17073215]
 [-0.84488505 -0.8730192  0.04624525 ... -0.20212881 -0.47378505
 -0.87137393]]

0s [18] X = standardized_data
Y = diabetes_dataset['Outcome']

0s [19] print(X)
print(Y)
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
  1.4259954 ]
0s completed at 8:01 PM
```

```
diabetes_prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
-0.19067191]
[ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
-0.10558415]
...
[ 0.3429808  0.00330087  0.14964075 ... -0.73518964 -0.68519336
-0.27575966]
[-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
 1.17073215]
[-0.84488505 -0.8730192  0.04624525 ... -0.20212881 -0.47378505
-0.87137393]]
0 1
1 0
2 1
3 0
4 1
..
763 0
764 0
765 0
766 1
767 0
Name: Outcome, Length: 768, dtype: int64

Train Test Split

[20] X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)

[21] print(X.shape, X_train.shape, X_test.shape)

(768, 8) (614, 8) (154, 8)

Training The Model
```

```
diabetes_prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[22] classifier = RandomForestClassifier()

[23] #training the support vector machine
classifier.fit(X_train, Y_train)

Model Evaluation

Accuracy Score

#Accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

[25] print('Accuracy of the training data : ',training_data_accuracy)

Accuracy of the training data :  0.998371335504886

[26] #Accuracy score on the testing data
X_test_prediction = classifier.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction, Y_test)

[27] print('Accuracy of the testing data : ',testing_data_accuracy)

Accuracy of the testing data :  0.7272727272727273
```

diabetes_prediction.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

0s [27] print('Accuracy of the testing data : ',testing_data_accuracy)

Accuracy of the testing data : 0.7272727272727273

Precision

0s [28] X_test_prediction = classifier.predict(X_test)
testing_data_accuracy = precision_score(X_test_prediction, Y_test)
print('Accuracy of the testing data : ',testing_data_accuracy)

Accuracy of the testing data : 0.48148148148148145

Recall

0s [29] X_test_prediction = classifier.predict(X_test)
testing_data_accuracy = recall_score(X_test_prediction, Y_test)
print('Accuracy of the testing data : ',testing_data_accuracy)

Accuracy of the testing data : 0.65

F1-Score

0s [30] X_test_prediction = classifier.predict(X_test)
testing_data_accuracy = f1_score(X_test_prediction, Y_test)
print('Accuracy of the testing data : ',testing_data_accuracy)

Accuracy of the testing data : 0.553191489361702

Making a predictive system

0s completed at 8:01 PM

diabetes_prediction.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

0s Accuracy of the testing data : 0.65

F1-Score

0s [30] X_test_prediction = classifier.predict(X_test)
testing_data_accuracy = f1_score(X_test_prediction, Y_test)
print('Accuracy of the testing data : ',testing_data_accuracy)

Accuracy of the testing data : 0.553191489361702

Making a predictive system

0s [31] input_data = (0,137,40,35,168,43.1,2.288,33)

#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_resaped = input_data_as_numpy_array.reshape(1,-1)

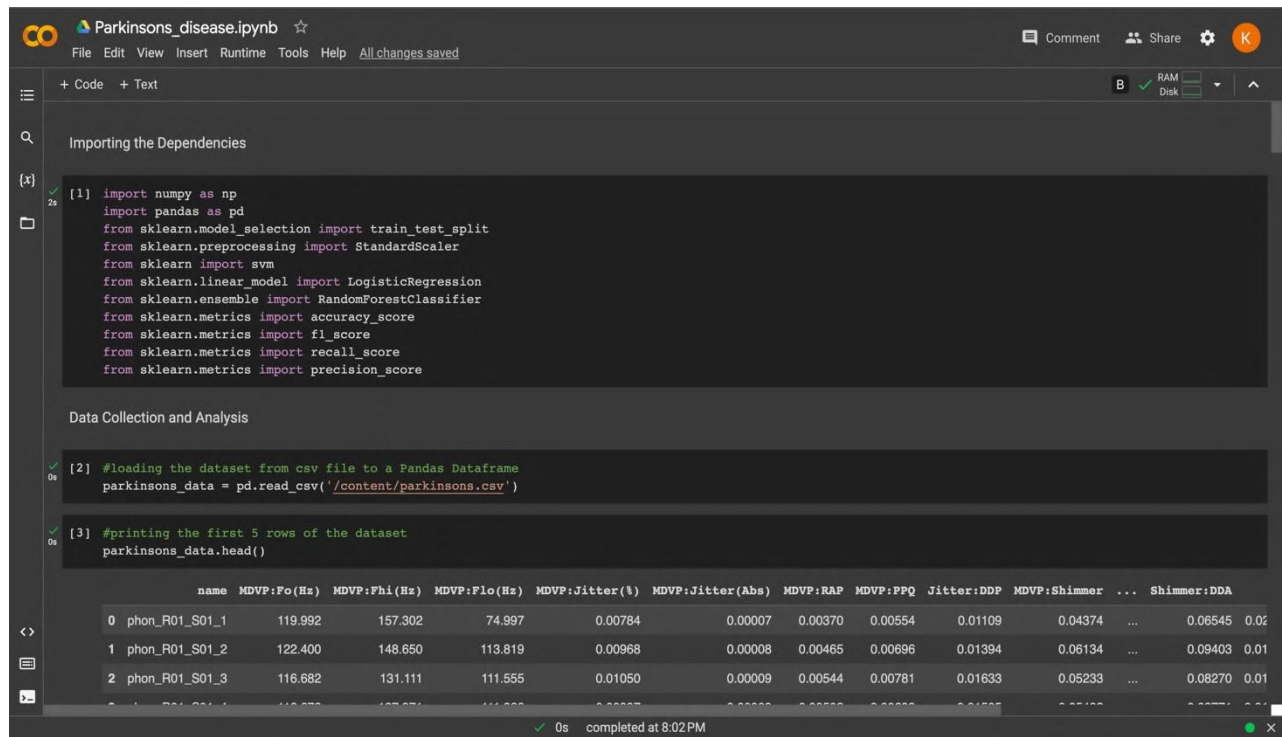
#standardize the input data
std_data = scaler.transform(input_data_resaped)

prediction = classifier.predict(std_data)

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

0s completed at 8:01 PM

11.2. PARKINSON'S DISEASE PREDICTION



Importing the Dependencies

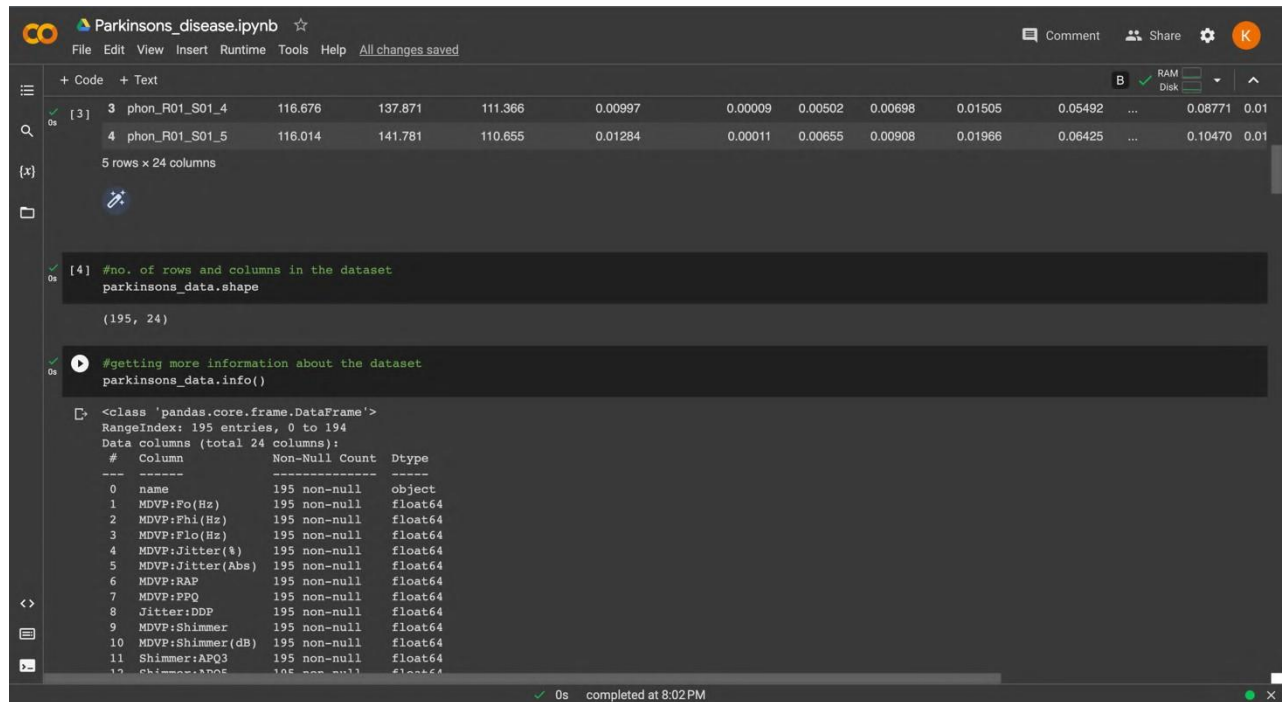
```
[1] import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
```

Data Collection and Analysis

```
[2] #loading the dataset from csv file to a Pandas Dataframe
parkinsons_data = pd.read_csv('/content/parkinsons.csv')
```

```
[3] #printing the first 5 rows of the dataset
parkinsons_data.head()
```

	name	MDVP:Fo(Hz)	MDVP:Phi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270



```
[3] 3 phon_R01_S01_4 116.676 137.871 111.366 0.00997 0.00009 0.00502 0.00698 0.01505 0.05492 ... 0.08771 0.01
4 phon_R01_S01_5 116.014 141.781 110.655 0.01284 0.00011 0.00655 0.00908 0.01966 0.06425 ... 0.10470 0.01
5 rows x 24 columns
```

```
[4] #no. of rows and columns in the dataset
parkinsons_data.shape
```

```
(195, 24)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
#   Column                Non-Null Count  Dtype
---  -
0   name                  195 non-null    object
1   MDVP:Fo(Hz)           195 non-null    float64
2   MDVP:Phi(Hz)          195 non-null    float64
3   MDVP:Flo(Hz)          195 non-null    float64
4   MDVP:Jitter(%)        195 non-null    float64
5   MDVP:Jitter(Abs)      195 non-null    float64
6   MDVP:RAP              195 non-null    float64
7   MDVP:PPQ              195 non-null    float64
8   Jitter:DDP            195 non-null    float64
9   MDVP:Shimmer          195 non-null    float64
10  MDVP:Shimmer(dB)      195 non-null    float64
11  Shimmer:APQ3          195 non-null    float64
12  Shimmer:APQ5          195 non-null    float64
```


Parkinsons_disease.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

0s [5] 20 spread1 195 non-null float64
21 spread2 195 non-null float64
22 D2 195 non-null float64
23 PPE 195 non-null float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB

0s #checking for missing value in each column
parkinsons_data.isnull().sum()

```
name      0
MDVP:F0(Hz)    0
MDVP:F1(Hz)    0
MDVP:Flo(Hz)   0
MDVP:Jitter(%) 0
MDVP:Jitter(Abs) 0
MDVP:RAP       0
MDVP:RPQ       0
Jitter:DDP     0
MDVP:Shimmer   0
MDVP:Shimmer(dB) 0
Shimmer:APQ3   0
Shimmer:APQ5   0
MDVP:APQ       0
Shimmer:DDA    0
NHR            0
HNR            0
status        0
RPDE          0
DFA           0
spread1       0
spread2       0
D2            0
PPE           0
dtype: int64
```

0s completed at 8:02 PM

Parkinsons_disease.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

0s #getting some statistics measures about the data
parkinsons_data.describe()

	MDVP:F0(Hz)	MDVP:F1(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:RPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(dB)	...	Shimmer:
count	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	...	195.000000
mean	154.228641	197.104918	116.324631	0.006220	0.000044	0.003306	0.003446	0.009920	0.029709	0.282251	...	0.046
std	41.390065	91.491548	43.521413	0.004848	0.000035	0.002968	0.002759	0.008903	0.018857	0.194877	...	0.030
min	88.333000	102.145000	65.476000	0.001680	0.000007	0.000680	0.000920	0.002040	0.009540	0.085000	...	0.018
25%	117.572000	134.862500	84.291000	0.003460	0.000020	0.001660	0.001860	0.004985	0.016505	0.148500	...	0.024
50%	148.790000	175.829000	104.315000	0.004940	0.000030	0.002500	0.002690	0.007490	0.022970	0.221000	...	0.038
75%	182.769000	224.205500	140.018500	0.007365	0.000060	0.003835	0.003955	0.011505	0.037885	0.350000	...	0.060
max	260.105000	592.030000	239.170000	0.033160	0.000260	0.021440	0.019580	0.064330	0.119080	1.302000	...	0.168

8 rows x 23 columns

0s [8] #distribution of a target variable
parkinsons_data['status'].value_counts()

```
1    147
0     48
Name: status, dtype: int64
```

1 -> Parkinson's Positive

0s completed at 8:02 PM

Parkinsons_disease.ipynb

File Edit View Insert Runtime Tools Help All changes saved

0 -> Healthy

```
[9] #grouping the data based on the target variable
parkinsons_data.groupby('status').mean()
```

<ipython-input-9-75067c7b398c>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only

```
parkinsons_data.groupby('status').mean()
      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)  ...  MDVP:APQ
```

status	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)	...	MDVP:APQ
0	181.937771	223.636750	145.207292	0.003866	0.000023	0.001925	0.002056	0.005776	0.017615	0.162958	...	0.013305
1	145.180762	188.441463	106.893558	0.006989	0.000051	0.003757	0.003900	0.011273	0.033658	0.321204	...	0.027600

2 rows x 22 columns

Data Preprocessing

Seperating the features and Target

```
[10] X = parkinsons_data.drop(columns=['name','status'], axis=1)
      Y = parkinsons_data['status']
```

```
[11] print(X)
```

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	
0	119.992	157.302	74.997	0.00784	

0s completed at 8:02 PM

Parkinsons_disease.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
193      0.241 ... 0.01588 0.03794 0.07223 19.020 0.451221
194      0.190 ... 0.01373 0.03078 0.04398 21.209 0.462803
```

	DFA	spread1	spread2	D2	PPE
0	0.815285	-4.813031	0.266482	2.301442	0.284654
1	0.819521	-4.075192	0.335590	2.486855	0.368674
2	0.825288	-4.443179	0.311173	2.342259	0.332634
3	0.819235	-4.117501	0.334147	2.405554	0.368975
4	0.823484	-3.747787	0.234513	2.332180	0.410335
..
190	0.657899	-6.538586	0.121952	2.657476	0.133050
191	0.683244	-6.195325	0.129303	2.784312	0.168895
192	0.655683	-6.787197	0.158453	2.679772	0.131728
193	0.643956	-6.744577	0.207454	2.138608	0.123306
194	0.664357	-5.724056	0.190667	2.555477	0.148569

[195 rows x 22 columns]

```
[12] print(Y)
```

0	1
1	1
2	1
3	1
4	1
..	...
190	0
191	0
192	0
193	0
194	0

Name: status, Length: 195, dtype: int64

Splitting the data into train and test data

0s completed at 8:02 PM

```

Parkinsons_disease.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
print(X)
0 MDVP:F0(Hz) MDVP:F1(Hz) MDVP:F2(Hz) MDVP:F3(Hz) MDVP:F4(Hz) \
1 119.992 157.302 74.997 0.00784
2 122.400 148.650 113.819 0.00968
3 116.682 131.111 111.555 0.01050
4 116.676 137.871 111.366 0.00997
.. ..
190 174.188 230.978 94.261 0.00459
191 209.516 253.017 89.488 0.00564
192 174.688 240.005 74.287 0.01360
193 198.764 396.961 74.904 0.00740
194 214.289 260.277 77.973 0.00567

MDVP:Jitter(Abs) MDVP:RAP MDVP:PPQ Jitter:DDP MDVP:Shimmer \
0 0.00007 0.00370 0.00554 0.01109 0.04374
1 0.00008 0.00465 0.00696 0.01394 0.06134
2 0.00009 0.00544 0.00781 0.01633 0.05233
3 0.00009 0.00502 0.00698 0.01505 0.05492
4 0.00011 0.00655 0.00908 0.01966 0.06425
.. ..
190 0.00003 0.00263 0.00259 0.00790 0.04087
191 0.00003 0.00331 0.00292 0.00994 0.02751
192 0.00008 0.00624 0.00564 0.01873 0.02308
193 0.00004 0.00370 0.00390 0.01109 0.02296
194 0.00003 0.00295 0.00317 0.00885 0.01884

MDVP:Shimmer(dB) ... MDVP:APQ Shimmer:DDA NHR HNR RPDE \
0 0.426 ... 0.02971 0.06545 0.02211 21.033 0.414783
1 0.626 ... 0.04368 0.09403 0.01929 19.085 0.458359
2 0.482 ... 0.03590 0.08270 0.01309 20.651 0.429895
3 0.517 ... 0.03772 0.08771 0.01353 20.644 0.434969
4 0.584 ... 0.04465 0.10470 0.01767 19.649 0.417356
.. ..
190 0.405 ... 0.02745 0.07008 0.02764 19.517 0.448439
191 0.263 ... 0.01879 0.04812 0.01810 19.147 0.431674
192 0.263 ... 0.01879 0.04812 0.01810 19.147 0.431674
193 0.263 ... 0.01879 0.04812 0.01810 19.147 0.431674
194 0.263 ... 0.01879 0.04812 0.01810 19.147 0.431674

0s completed at 8:02 PM

```

```

Parkinsons_disease.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[13] X_train,X_test,Y_train,Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)

[14] print(X.shape,X_train.shape,X_test.shape)

(195, 22) (156, 22) (39, 22)

Data Standardization

[15] scaler = StandardScaler()

[16] scaler.fit(X_train)

StandardScaler
StandardScaler()

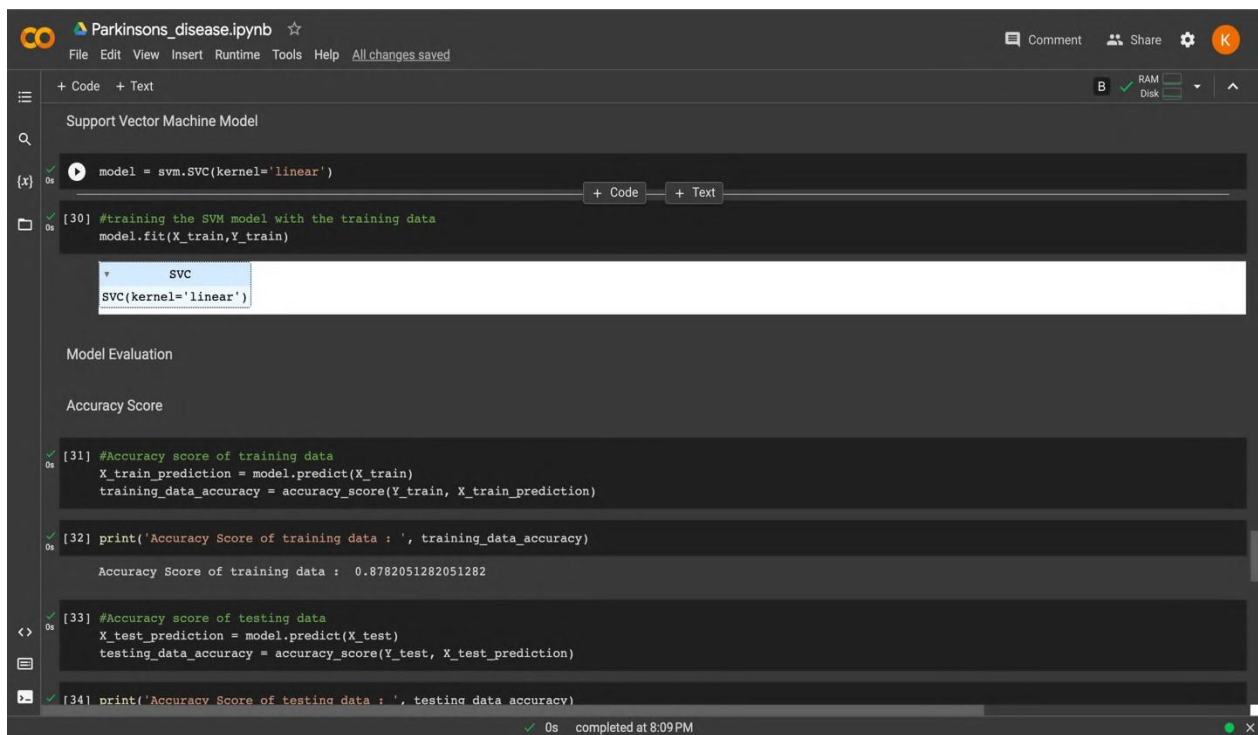
[17] X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

[18] print(X_train)

[[-0.14219924 -0.44087433  0.44559911 ...  0.2013613  1.24374363
 -0.14140947]
 [-0.91887281 -0.73124637 -0.13666403 ...  0.10797451  0.15775053
 -0.23176004]
 [ 0.55407102  0.26929249 -0.73342257 ...  0.68885116  0.72892978
  0.34947004]
 ...
 [-0.81919408 -0.80817681 -0.25304279 ... -0.08414744  0.38093939
 -0.39918566]]

0s completed at 8:02 PM

```



Parkinsons_disease.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Support Vector Machine Model

```
model = svm.SVC(kernel='linear')
```

[30] #training the SVM model with the training data

```
model.fit(X_train,Y_train)
```

SVC

```
SVC(kernel='linear')
```

Model Evaluation

Accuracy Score

```
[31] #Accuracy score of training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

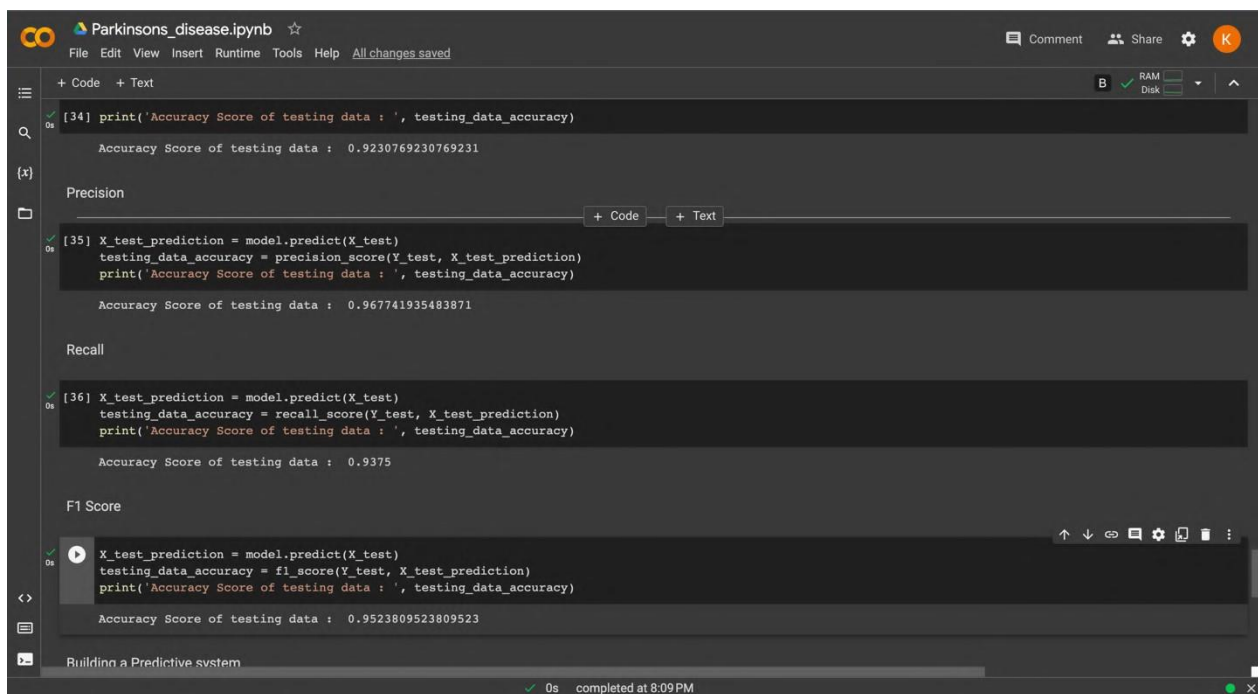
```
[32] print('Accuracy Score of training data : ', training_data_accuracy)
```

Accuracy Score of training data : 0.8782051282051282

```
[33] #Accuracy score of testing data
X_test_prediction = model.predict(X_test)
testing_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

```
[34] print('Accuracy Score of testing data : ', testing_data_accuracy)
```

0s completed at 8:09 PM



Parkinsons_disease.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[34] print('Accuracy Score of testing data : ', testing_data_accuracy)
```

Accuracy Score of testing data : 0.9230769230769231

Precision

```
[35] X_test_prediction = model.predict(X_test)
testing_data_accuracy = precision_score(Y_test, X_test_prediction)
print('Accuracy Score of testing data : ', testing_data_accuracy)
```

Accuracy Score of testing data : 0.967741935483871

Recall

```
[36] X_test_prediction = model.predict(X_test)
testing_data_accuracy = recall_score(Y_test, X_test_prediction)
print('Accuracy Score of testing data : ', testing_data_accuracy)
```

Accuracy Score of testing data : 0.9375

F1 Score

```
X_test_prediction = model.predict(X_test)
testing_data_accuracy = f1_score(Y_test, X_test_prediction)
print('Accuracy Score of testing data : ', testing_data_accuracy)
```

Accuracy Score of testing data : 0.9523809523809523

Building a Predictive system

0s completed at 8:09 PM

The screenshot shows a Jupyter Notebook interface with the following content:

```

[36] Accuracy Score of testing data : 0.9375

F1 Score

X_test_prediction = model.predict(X_test)
testing_data_accuracy = f1_score(Y_test, X_test_prediction)
print('Accuracy Score of testing data : ', testing_data_accuracy)

Accuracy Score of testing data : 0.9523809523809523

Building a Predictive system

[28] input_data = (122.40000,148.65000,113.81900,0.00968,0.00008,0.00465,0.00696,0.01394,0.06134,0.62600,0.03134,0.04518,0.04368,0.09403,0.01929,19.08500,0.45835)

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

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

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

prediction = model.predict(std_data)

if(prediction[0] == 0):
    print('The person does not have Parkinsons Disease')
else:
    print('The person has Parkinsons Disease')

```

The notebook status bar at the bottom indicates "0s completed at 8:09 PM".

11.3. HEART DISEASE PREDICTION

The screenshot shows a Jupyter Notebook interface with the following content:

```

Importing The Dependencies

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score

Data Collection and Processing

[2] #Loading CSV data to pandas dataframe
heart_data = pd.read_csv('/content/heart_disease_data.csv')

[3] #Print first 5 rows of dataset
heart_data.head()

```

The output of the `heart_data.head()` command is displayed as a table:

	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1

The notebook status bar at the bottom indicates "0s completed at 8:03 PM".

Heart_Disease.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

0s [3] 3 56 1 1 120 236 0 1 178 0 0.8 2 0 2 1
4 57 0 0 120 354 0 1 163 1 0.6 2 0 2 1

0s #Print last 5 rows of dataset
heart_data.tail(10)

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
293	67	1	2	152	212	0	0	150	0	0.8	1	0	3	0
294	44	1	0	120	169	0	1	144	1	2.8	0	0	1	0
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3	0
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2	0
297	59	1	0	164	176	1	0	90	0	1.0	1	2	1	0
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

0s [5] #number of rows and columns in the dataset
heart_data.shape

(303, 14)

0s [6] #checking for missing values
heart_data.isnull().sum()

0s completed at 8:03 PM

Heart_Disease.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

0s age 0
sex 0
cp 0
trestbps 0
chol 0
fbs 0
restecg 0
thalach 0
exang 0
oldpeak 0
slope 0
ca 0
thal 0
target 0
dtype: int64

0s [7] # statistical measure of the data
heart_data.describe()

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

0s completed at 8:03 PM


```
Heart_Disease.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[8] # target variable distribution
heart_data['target'].value_counts()

1    165
0    138
Name: target, dtype: int64

1 -> Defective Heart
0 -> Healthy Heart

Splitting Features and target

[9] X = heart_data.drop(column='target', axis=1)
    Y = heart_data['target']

[10] print(X)

   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0   63   1   3    145    233   1      0    150     0     2.3
1   37   1   2    130    250   0      1    187     0     3.5
2   41   0   1    130    204   0      0    172     0     1.4
3   56   1   1    120    236   0      1    178     0     0.8
4   57   0   0    120    354   0      1    163     1     0.6
..  ...  ...  ..    ...    ...  ...    ...    ...    ...    ...
298  57   0   0    140    241   0      1    123     1     0.2
299  45   1   3    110    264   0      1    132     0     1.2
300  68   1   0    144    193   1      1    141     0     3.4
301  57   1   0    130    131   0      1    115     1     1.2
302  57   0   1    130    236   0      0    174     0     0.0
   slope  ca  thal
0      0   0    1
..      ..  ..  ..
298     1   0    3
299     1   0    3
300     1   2    3
301     1   1    3
302     1   1    2

[303 rows x 13 columns]
```

```
Heart_Disease.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[11] print(Y)

0    1
1    1
2    1
3    1
4    1
..
298  0
299  0
300  0
301  0
302  0
Name: target, Length: 303, dtype: int64

Splitting into training and testing data

[12] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=1)
```

```
[24] model = LogisticRegression()

[25] #train the logistic regression model with training data
model.fit(X_train,Y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
  LogisticRegression
  LogisticRegression())

Model Evaluation

Accuracy Score

[26] # accuracy on training data
x_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(x_train_prediction, Y_train)

[27] print('Accuracy on training data : ', training_data_accuracy)

Accuracy on training data : 0.859504132231405

[28] # accuracy on testing data
```

```
[28] # accuracy on testing data
x_test_prediction = model.predict(X_test)
testing_data_accuracy = accuracy_score(x_test_prediction, Y_test)

[29] print('Accuracy on testing data : ', testing_data_accuracy)

Accuracy on testing data : 0.8360655737704918

Precision Score

[30] x_train_prediction = model.predict(X_test)
training_data_accuracy = precision_score(x_train_prediction, Y_test)
print('precision on testing data : ', training_data_accuracy)

precision on testing data : 0.8787878787878788

Recall

[31] x_train_prediction = model.predict(X_test)
training_data_accuracy = recall_score(x_train_prediction, Y_test)
print('precision on testing data : ', training_data_accuracy)

precision on testing data : 0.8285714285714286

F1-Score

[32] x_train_prediction = model.predict(X_test)
training_data_accuracy = f1_score(x_train_prediction, Y_test)
print('precision on testing data : ', training_data_accuracy)
```


The screenshot shows a Jupyter Notebook interface with the following content:

```

File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
F1-Score

[32] x_train_prediction = model.predict(X_test)
training_data_accuracy = f1_score(x_train_prediction, Y_test)
print('precision on testing data : ', training_data_accuracy)

precision on testing data : 0.8529411764705883

Prediction System

input_data = (64,1,2,125,309,0,1,131,1,1.8,1,0,3)

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

# reshape the numpy array as we are predicting for only on instance
input_data_resaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_resaped)

if (prediction[0]== 0):
    print('The Person does not have a Heart Disease')
else:
    print('The Person has Heart Disease')

The Person does not have a Heart Disease
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature
warnings.warn(

```

12. BIBLIOGRAPHY

1. Ghosh, A.M., Halder, D., and Hossain, S.A. IoT-based remote health monitoring system. The 5th International Conference on Informatics, Electronics, and Vision (ICIEV) was held in 2016. (pp. 921-926). IEEE, May 2016.
2. P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poullos, et al., "Enabling smart cities through a cognitive management framework for the internet of things" in Communications Magazine, IEEE, vol. 51, no. 6, pp. 102-111, 2013.
3. S. Sasidharan, A. Somov, A. R. Biswas and R. Giaffreda, "Cognitive management framework for Internet of Things: A prototype implementation" in InInternet of Things (WF-IoT), IEEE, pp. 538-543, March 2014.
4. Hashim, H.; Salihudin, S.F.B.; Saad, P.S.M. Development of IoT Based Healthcare Monitoring System. In Proceedings of the 2022 IEEE International Conference in Power Engineering Application (ICPEA), Selangor, Malaysia, 7–8 March 2022; pp. 1–5.
5. Kaur, P.; Kumar, R.; Kumar, M. A healthcare monitoring system using random forest and internet of things (IoT). Multimed. Tools Appl. 2019, 78, 19905–19916
6. Jenifer, M.; Rinesh, S.; Thamaraiselvi, K. Internet of Things (IOT) based Patient health care Monitoring System using electronic gadget. In Proceedings of the 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 25–27 May 2022; pp. 487–490.
7. Jeong, S.; Shen, J.-H.; Ahn, B. A Study on Smart Healthcare Monitoring Using IoT Based on Blockchain. Wirel. Commun. Mob. Comput. 2021, 2021, 9932091.
8. Souri, A.; Ghafour, M.Y.; Ahmed, A.M.; Safara, F.; Yamini, A.; Hoseyninezhad, M. A new machine learning-based healthcare monitoring model for student's condition diagnosis in Internet of Things environment. Soft Comput. 2020, 24, 17111–17121.

9. Manoj, A.S.; Hussain, M.A.; Teja, P.S. Patient health monitoring system using IoT. Mater. Today Proc. 2021, 2214–7853.
10. Raj, J.S. A novel information processing in IoT based real time health care monitoring system. J. Electron. 2020, 2, 188–196
11. https://www.researchgate.net/publication/275328861_Software_Design_Framework_for_Healthcare_Systems
12. <https://www.geeksforgeeks.org/structural-software-testing/>
13. https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm
14. <https://www.google.com/>
15. <https://www.wikipedia.org/>