

5G SMART DIABETES :TOWARDS PERSONALIZED DIABETES DIAGNOSIS WITH HEALTH CARE BIG DATA AND CLOUD

*A Mini Project Report submitted to
JNTU Hyderabad in partial fulfillment
of the requirements for the award of the degree*

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

PURAM ASHRITHA

21RG1A05Q0

VALLAPU NAVYA

21RG1A05R2

YEGOLAM DEEPTHI

21RG1A05R9

THUMMUNURU BHAVITHA

21RG1A05R1

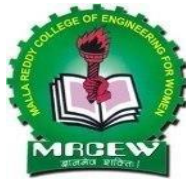
Under the Guidance of

Mrs. K.SHEETAL

B. Tech., M. Tech

Assistant Professor and head

Department of CSE



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MALLA REDDY COLLEGE OF ENGINEERING FOR WOMEN**

An UGC Autonomous Institution

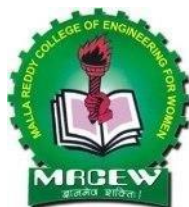
Approved by AICTE New Delhi and Affiliated to JNTUH

Maisammaguda , Medchal (Dist), Hyderabad -500100, Telangana.

OCTOBER 2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MALLA REDDY COLLEGE OF ENGINEERING FOR WOMEN
An UGC Autonomous Institution

Approved by AICTE New Delhi and Affiliated to JNTUH
Maisammaguda , Medchal (Dist), Hyderabad -500100, Telangana.
OCTOBER 2024



CERTIFICATE

This is to certify that the Mini project entitled **“5G SMART DIABETES”** has been submitted by **PURAM ASHRITHA (21RG1A05Q0)**, **VALLAPU NAVYA (21RG1A05R2)**, **YEGOLAM DEEPTHI (21RG1A05R9)**, **THUMMUNURU BHAVITHA (21RG1A05R1)** in partial fulfillment of the requirements for the award of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING**. This record of bonafide work carried out by them under my guidance and supervision. *The result embodied in this mini project report has not been submitted to any other University or Institute for the award of any degree.*

Mrs. K. SHEETAL
Assistant professor
Project Guide

Mrs. K. SHEETAL
Head of Department

External Examiner

ACKNOWLEDGEMENT

The Mini Project work carried out by our team in the Department of Computer Science and Engineering, Malla Reddy College of Engineering for Women, Hyderabad. ***This work is original and has not been submitted in part or full for any degree or diploma of any other university.***

We wish to acknowledge our sincere thanks to our project guide **Mrs. K. Sheetal**, Assistant Professor, Computer Science & Engineering for formulation of the problem, analysis, guidance and her continuous supervision during the course of work.

We acknowledge our sincere thanks to **Dr. Kanaka Durga Returi**, Principal & Professor of Department of Computer Science and Engineering and **Mrs. Sheetal Kulkarni**, Head of the Department, MRCEW and all our faculties of CSE Department for their kind cooperation in making this Mini Project work a success.

We extend our gratitude to **Sri. Ch. Malla Reddy**, Founder Chairman and **Sri. Ch. Mahender Reddy**, Secretary, **Dr. Vaka Murali Mohan**, Director for their kind cooperation in providing the infrastructure for completion of our Mini Project.

We acknowledge our special thanks to the entire teaching faculty and non-teaching staff members of the Computer Science & Engineering Department for their support in making this project work a success.

PURAM ASHRITHA	HT.NO.	21RG1A05Q0 _____
VALLAPU NAVYA	HT.NO.	21RG1A05R2 _____
YEGOLAM DEEPTHI	HT.NO.	21RG1A05R9 _____
THUMMUNURU BHAVITHA	HT.NO.	21RG1A05R1 _____

INDEX

Chapter	Page No
ABSTRACT	Vii
LIST OF FIGURES	Viii
1. SYSTEM ANALYSIS	1
1.1 Existing System	1
1.1.1 Disadvantages of Existing System	1
1.2 Proposed System	2
1.2.1 Advantages of Proposed System	3
1.3 Introduction	3
2. LITERATURE SURVEY	5
3. SYSTEM DESIGN	8
3.1 System modules	8
3.1.1 Server	8
3.1.2 User	8
3.2 System Architecture	9
3.3 System Requirements	10
3.3.1 Hardware Requirements	10
3.3.2 Software Requirements	10
3.4 UML Diagrams	11
3.4.1 Use Case Diagram	11
3.4.2 Class Diagram	12
3.4.3 Sequence Diagram	13
3.4.4 Flow Chart Diagram	14
4. INPUT & OUTPUT DESIGN	16

4.1 Input Design	16
4.2 Output Design	17
5. SOFTWARE ENVIRONMENT	18
5.1 Python Technology	18
5.1.1 What can Python do?	18
5.1.2 Python Install	19
5.2 Virtual Environments and Packages	20
5.2.1 Introduction	20
5.2.2 creating Virtual Environments	21
5.3 Cross platform	24
5.4 Java Platform	26
5.5 Mac OS Platform	27
5.6 Unix platform	27
5.7 Using Python Interpreter	28
5.7.1 invoking the interpreter	28
5.8 The interpreter and its Environment	30
5.8.1 Source code Encoding	30
5.8.2 Introduction to Artificial Intelligence	30
5.9 IBM Watson	32
5.10 Machine Learning	33
5.10.1 Introduction	33
5.10.2 Introduction to Deep Learning	38
5.11 Django	39
6. SYSTEM STUDY	40
6.1 Economic Feasibility	40
6.2 Technical Feasibility	40
6.3 Social Feasibility	41

7. SYSTEM TESTING	42
7.1 Types of Tests	42
7.1.1 Unit testing	42
7.1.2 Integration testing	43
7.1.3 Functional testing	43
7.1.4 System testing	44
7.1.5 Acceptance testing	44
8. RESULTS	45
9. CONCLUSION & FUTURE ENHANCEMENT	53
10. BIBLIOGRAPHY	54

ABSTRACT

Recent advances in wireless networking and big data technologies, such as 5G networks, medical big data analytics, and the Internet of Things, along with recent developments in wearable computing and artificial intelligence, are enabling the development and implementation of innovative diabetes monitoring systems and applications. Due to the life-long and systematic harm suffered by diabetes patients, it is critical to design effective methods for the diagnosis and treatment of diabetes. Based on our comprehensive investigation, this article classifies those methods into Diabetes 1.0 and Diabetes 2.0, which exhibit deficiencies in terms of networking and intelligence. Thus, our goal is to design a sustainable, cost-effective, and intelligent diabetes diagnosis solution with personalized treatment. In this article, we first propose the 5G-Smart Diabetes system, which combines the state-of-the-art technologies such as wearable 2.0, machine learning, and big data to generate comprehensive sensing and analysis for patients suffering from diabetes. Then we present the data sharing mechanism and personalized data analysis model for 5G-Smart Diabetes. Finally, we build a 5G-Smart Diabetes testbed that includes smart clothing, smartphone, and big data clouds. The experimental results show that our system can effectively provide personalized diagnosis and treatment suggestions to patients.

LIST OF FIGURES

Fig. No.	Fig. Name	Page.no
3.1	Block diagram of Cloud Application	8
3.2	Block diagram of User Application	8
3.3	System Architecture	9
3.4	Use Case Diagram	11
3.5	Class Diagram	12
3.6	Sequence Diagram	13
3.7	Flow Chart Diagram	14
3.8	Flow Chart Diagram	15
8.1	Screenshot of Open Server Page	46
*8.2	Screenshot of Uploading dataset	46
8.3	Screenshot of Preprocessing dataset	47
8.4	Screenshot of Decision Tree Accuracy	47
8.5	Screenshot of SVM Accuracy	48
8.6	Screenshot of ANN Accuracy	48
8.7	Screenshot of Ensemble Accuracy	49
8.8	Screenshot of Accuracy graph	49
8.9	Screenshot of after starting the cloud server	50
8.10	Screenshot of user side page	50
8.11	Screenshot of uploading user dataset	51
8.12	Screenshot of Results	51
8.13	Screenshot of Decision Tree algorithm	52
8.14	Screenshot of other algorithms	52

1.1 EXISTING SYSTEM

- Existing systems for diabetes diagnosis can be broadly categorized into traditional methods, digital health solutions, artificial intelligence (AI) and machine learning (ML) systems, big data analytics platforms, IoT-based systems, and personalized medicine approaches.
- Traditional methods for diabetes diagnosis include Fasting Plasma Glucose (FPG) tests, Oral Glucose Tolerance Tests (OGTT), Hemoglobin A1c (HbA1c) tests, Random Plasma Glucose tests, and urine tests such as microalbuminuria. These methods are widely used but have limitations in terms of accuracy and convenience.
- Digital health solutions have emerged as a promising alternative to traditional methods. Mobile apps like MyFitnessPal and Glucose Buddy enable users to track their glucose levels, diet, and exercise.
- Wearable devices like continuous glucose monitors and fitness trackers provide real-time data on glucose levels and physical activity.
- Telemedicine platforms like American Well and Teladoc facilitate remote consultations with healthcare professionals.

1.1.1 Disadvantages of Existing System

The existing diabetes detection system faces the following problems:

- The system is uncomfortable, and real-time data collection is difficult. Furthermore, it lacks continuous monitoring of multidimensional physiological indicators of patients suffering from diabetes.
- The diabetes detection model lacks a data sharing mechanism and personalized analysis of big data from different sources including lifestyle, sports, diet, and so on [2].
- There are no continuous suggestions for the prevention and treatment of diabetes and corresponding supervision strategies.

1.2 PROPOSED SYSTEM

- In the proposed system, we first propose a next generation diabetes solution called the 5G-Smart Diabetes system, which integrates novel technologies including fifth generation (5G) mobile networks, machine learning, medical big data, social networking, smart clothing, and so on.
- Then we present the data sharing mechanism and personalized data analysis model for 5G-Smart Diabetes. Finally, based on the smart clothing, smartphone, and big data healthcare clouds, we build a 5G-Smart Diabetes testbed and give the experiment results
- We are using Decision Tree, SVM, Artificial Neural Network algorithms from python to predict patient condition from his data. To train these algorithms we are using diabetes dataset. To predict data efficiently author is using Ensemble Algorithm which is combination of Decision Tree, SVM and ANN algorithm. Training model of all these three algorithms will be merging inside Ensemble Algorithm to get better accuracy and prediction
- Training model of all these three algorithms will be merging inside Ensemble Algorithm to get better accuracy and prediction.
 - 1) **COST:** this technique requires no cost compare to hospitalization as users will be having wearable device which will read his condition and inform to patients and hospitals using his smart phone
 - 2) **Comfortable:** as these wearable devices are small and patients can wear it and keep working on his daily activities.
 - 3) **Sustainability:** Devices can be in contact with hospital servers which will have complex data mining algorithms running on it. After receiving patient data server will run those algorithms to predict patient condition and send report back to devices.

In propose system we are using Decision Tree, SVM, Artificial Neural Network algorithms from python to predict patient condition from his data. To train these algorithms we are using diabetes dataset. To predict data efficiently author is using Ensemble Algorithm which is combination of Decision Tree, SVM and ANN algorithm.

Training model of all these three algorithms will be merging inside Ensemble Algorithm to get better accuracy and prediction.

- 4) **Personalization:** In this technique one patient can share his data with other patient based on distance between cloud servers they are using to store data. Here we are using dataset so sharing is not possible but I am making all predicted test data values to be open so all users can see or share it.
- 5) **Smartness:** this technique will be considered as smart as it required no human effort to inform patient about current condition.

1.2.1 Advantages of Proposed System

- Enhanced patient engagement and empowerment.
- Personalized diabetes management plans
- Early detection and prevention of complications
- Reduced healthcare costs

1.3 INTRODUCTION

- Diabetes is an extremely common chronic disease from which nearly 8.5 percent of the world population suffer; 422 million people worldwide have to struggle with diabetes. It is crucial to note that type 2 diabetes mellitus makes up about 90 percent of the cases [1]. More critically, the situation will be worse, as reported in [2], with more teenagers and youth becoming susceptible to diabetes as well. Due to the fact that diabetes has a huge impact on global wellbeing and economy, it is urgent to improve methods for the prevention and treatment of diabetes [3].
- Furthermore, the “5G” in 5G-Smart Diabetes has a two-fold meaning. On one hand, it refers to the 5G technology that will be adopted as the communication infrastructure to realize high-quality and continuous monitoring of the physiological states of patients with diabetes and to provide treatment services for such patients without restraining their freedom. On the other hand, “5G” refers to the following “5 goals”: cost effectiveness, comfortability, personalization, sustainability, and smartness.

Cost Effectiveness: It is achieved from two aspects. First, 5G-Smart Diabetes keeps users in a healthy lifestyle so as to prevent users from getting the disease in the

early stage. The reduction of disease risk would lead to decreasing the cost of diabetes treatment. Second, 5G-Smart Diabetes facilitates out-of-hospital treatment, thus reducing the cost compared to on-the-spot treatment, especially long-term hospitalization of the patient.

Comfortability: To achieve comfort for patients, it is required that 5G-Smart Diabetes does not disturb the patients' daily activities as much as possible. Thus, 5G-Smart Diabetes integrates smart clothing [3], mobile phones, and portable blood glucose monitoring devices to easily monitor patients' blood glucose and other physiological indicators.

Personalization: 5G-Smart Diabetes utilizes various machine learning and cognitive computing algorithms to establish personalized diabetes diagnosis for the prevention and treatment of diabetes. Based on the collected blood glucose data and individualized physiological indicators, 5G-Smart Diabetes produces personalized treatment solutions for patients.

Sustainability: By continuously collecting, storing, and analyzing information on personal diabetes, 5G-Smart Diabetes adjusts the treatment strategy in time based on the changes of patients' status. Furthermore, in order to be sustainable for data-driven diabetes diagnosis and treatment, 5G-Smart Diabetes establishes effective information sharing among patients, relatives, friends, personal health advisors, and doctors.

With the help of social networking, the patient's mood can be better improved so that he or she is more self-motivated to perform a treatment plan in time. Smartness: With cognitive intelligence toward patients' status and network resources, 5G-Smart Diabetes achieves early detection and prevention of diabetes and provides personalized treatment to patients. The remaining part of the article is organized as follows. We first present the system architecture of 5G-Smart Diabetes. Then we explain the data sharing mechanism and propose the personalized data analysis model. Furthermore, we introduce the 5G-Smart Diabetes testbed.

CHAPTER-2: LITERATURE SURVEY

Chen et al. [1] proposed the 5G-Smart Diabetes system, which combined the state-of-the-art technologies such as wearable 2.0, machine learning, and big data to generate comprehensive sensing and analysis for patients suffering from diabetes. Then this work presented the data sharing mechanism and personalized data analysis model for 5G-Smart Diabetes. Finally, this work builds a 5G-Smart Diabetes testbed that includes smart clothing, smartphone, and big data clouds. The experimental results showed that the system can effectively provide personalized diagnosis and treatment suggestions to patients.

Rghioui et al. [2] presented an intelligent architecture for monitoring diabetic patients by using machine learning algorithms. The architecture elements included smart devices, sensors, and smartphones to collect measurements from the body. The intelligent system collected the data received from the patient and performed data classification using machine learning to make a diagnosis. The proposed prediction system was evaluated by several machine learning algorithms, and the simulation results demonstrated that the sequential minimal optimization (SMO) algorithm gives superior classification accuracy, sensitivity, and precision compared to other algorithms.

Venkatachalam et al. [3] motivated to develop a diabetes motoring system for patients using IoT device in their body which monitors their blood sugar level, blood pressure, sport activities, diet plan, oxygen level, ECG data. The data are processed using feature selection algorithm called as particle swarm optimization and transmitted to nearest edge node for processing in 5G networks. Secondly, data are processed using DBN Layer. Thirdly, this work shared the diagnosed data output through the wireless communication such as LTE/5G to the patients connected through the edge nodes for further medical assistance. The patient wearable devices are connected to the social network. The Result of this proposed system is evaluated with some existing system. Time and Performance outperform than other techniques.

Prakash et al. [4] introduced a neural network-based ensemble voting classifier to predict accurately the diabetes in the patients via online monitoring. The study consists of

Internet of Things (IoT) devices to monitor the instances of the patients. While monitoring, the data are transferred from IoT devices to smartphones and then to the cloud, where the process of classification takes place. The simulation is conducted on the collected samples using the python tool. The results of the simulation show that the proposed method achieves a higher accuracy rate, higher precision, recall, and f-measure than existing state-of-art ensemble models.

Tsoulchas et al. [5] proposed a model to monitor the health of people with diabetes melitus, a disease with high incident rates mainly at the elderly but also in younger people. Specifically, a study about the existing medically approved technologies for continuous measurement of diabetes is described. Subsequently, the model for monitoring patient's blood glucose levels is described. Whenever a patient's blood glucose levels are Low or High, the model triggers an alarm to a Cloud infrastructure in order remote medical staff to provide immediate cure to the patient. Furthermore, to assure the immediate response of the remote medical staff, the proposed model is deployed upon a 5G wireless network architecture.

Huang et al. [6] proposed a 5G-based Artificial Intelligence Diabetes Management architecture (AIDM), which can help physicians and patients to manage both acute complications and chronic complications. The AIDM contains five layers: the sensing layer, the transmission layer, the storage layer, the computing layer, and the application layer. We build a test bed for the transmission and application layers. Specifically, this work applied a delay-aware RA optimization based on a doublequeue model to improve access efficiency in smart hospital wards in the transmission layer. In application layer, this work builds a prediction model using a deep forest algorithm.

Agrawal et al. [7] have made significant contributions to the integration of 5G technology in healthcare systems, particularly in managing chronic diseases like diabetes. Their research emphasizes the enhanced connectivity provided by 5G, which offers higher data transfer rates and lower latency, facilitating real-time communication between patients and healthcare providers. They explore the potential of the Internet of Things (IoT) to connect various health monitoring devices, such as glucose meters and wearable sensors, enabling seamless data collection and sharing. Additionally, Agrawal et al. highlight the advancements in telemedicine made possible by 5G, which enhance virtual

consultations through improved video quality and reduced lag, allowing for more interactive patient-provider engagement. Their work also discusses the role of data analytics in creating personalized care plans by analyzing large datasets collected from patients, enabling providers to identify trends and adjust treatments accordingly. Overall, they advocate for patient-centric models that empower individuals to actively manage their diabetes, illustrating the transformative potential of 5G in improving chronic disease management and overall patient outcomes.

Khan et al. [8] have conducted impactful research on the role of 5G technology in enhancing telehealth services, particularly for managing chronic conditions like diabetes. Their work emphasizes the advantages of 5G's high-speed connectivity and low latency, which enable real-time data transmission from wearable devices and health monitors to healthcare providers. This capability allows for continuous monitoring of patients' health metrics, facilitating timely interventions and personalized care. Khan et al. also discuss the integration of telemedicine into diabetes management, highlighting how 5G improves virtual consultations by enhancing video quality and reducing communication delays, ultimately fostering better interactions between patients and providers. Furthermore, their research explores the potential for data analytics to predict patient needs and outcomes, enabling proactive management of diabetes.

Ranjan et al. [9] have made noteworthy contributions to the field of remote patient monitoring, particularly through the lens of 5G technology's application in healthcare. Their research emphasizes the advantages of 5G connectivity, which provides high-speed data transfer and minimal latency, essential for real-time monitoring of patients with chronic conditions like diabetes. Ranjan et al. propose a comprehensive framework for remote monitoring systems that integrates various health devices, enabling seamless data collection and communication between patients and healthcare providers. They highlight the role of 5G in enhancing the reliability and responsiveness of these systems, allowing for timely interventions based on continuous health data.

3.1 System Modules

Here design two applications to implement the system

3.1.1 server applications :

This application act like a cloud server and storage and train dataset model with various algorithms such as decision tree, SVM and ANN and Ensemble algorithms.

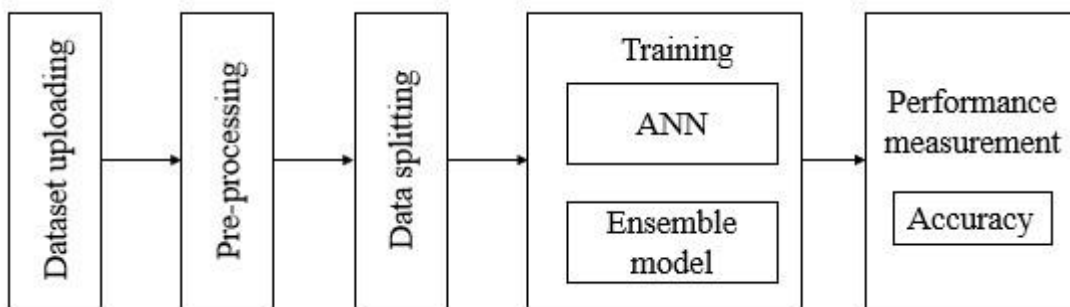


Fig .3.1: Block diagram of cloud application.

3.1.2 User applications :

In this application we will upload some test data and will be consider as user sense data and this data will be sent to cloud server and cloud server will apply decision and SVM and ANN model on test data to predict patient condition and send resultant data to this application. As we don't have sensors to sense data so we consider uploaded test data as sense data. Here we don't have user details to share data so I am keeping all predicted data to be open so all users can see and share.

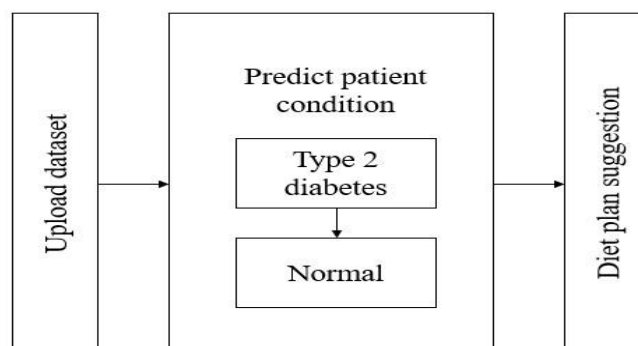


Fig. 3.2: Block diagram of user application

3.2 System Architecture

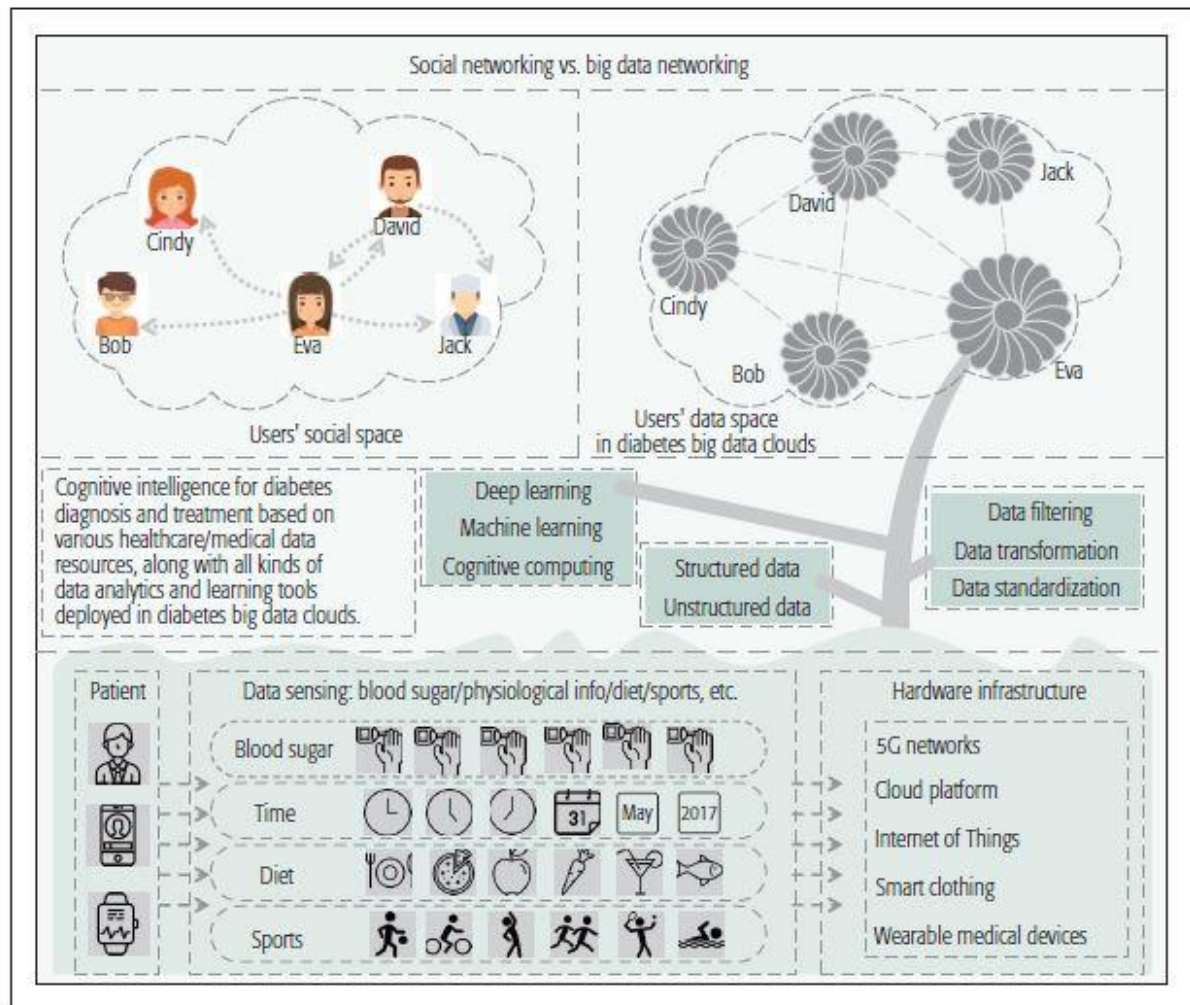


Fig.3.3: System Architecture

3.3 System Requirements

3.3.1 Hardware Requirements

System	:	Pentium IV 2.4 GHz.
Hard Disk	:	40 GB.
Floppy Drive	:	1.44 Mb.
Monitor	:	14' Colour Monitor.
Mouse	:	Optical Mouse.
Ram	:	512 Mb.

3.3.2 Software Requirements:

Operating system	:	Windows 7 Ultimate.
Coding Language	:	Python.
Front-End	:	Python.
Designing	:	Html, CSS, JavaScript.
Data Base	:	MySQL.

3.4 UML Diagrams

3.4.1 Use Case Diagram

In the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Role of the actors in the system can be depicted.

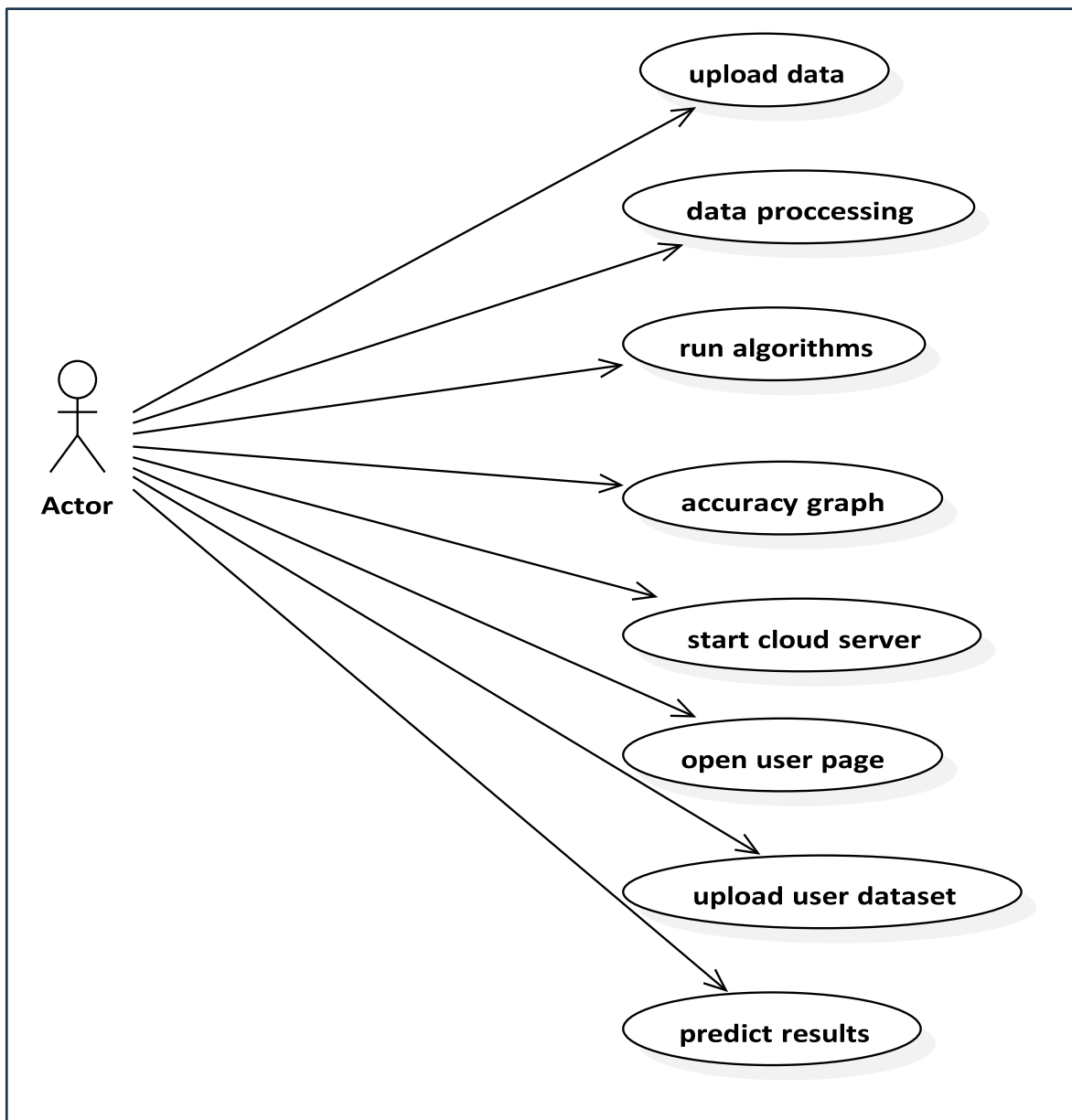


Fig. 3.4 : Use Case Diagrams

3.4.2 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

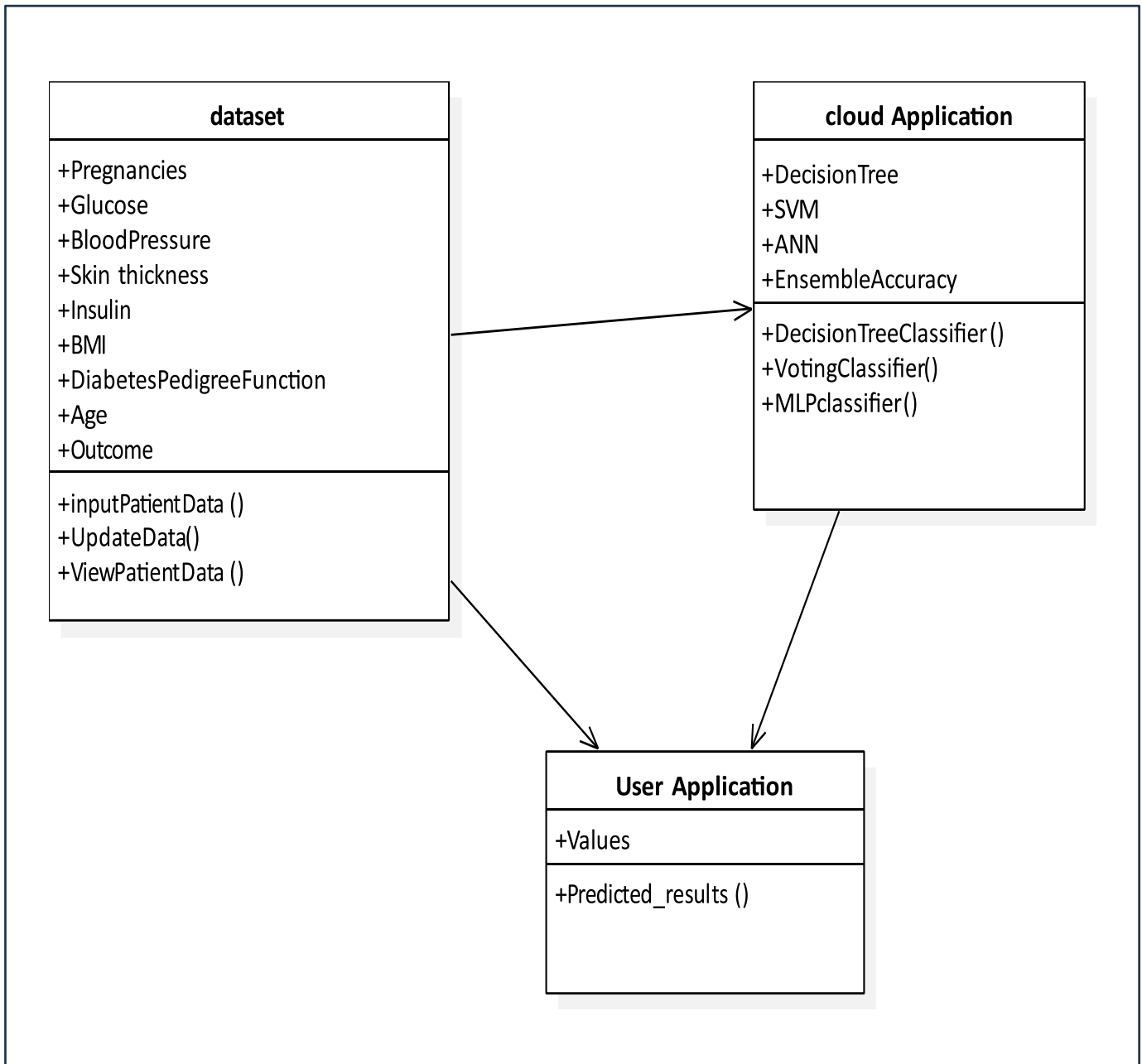


Fig. 3.5: Class Diagram

3.4.2 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

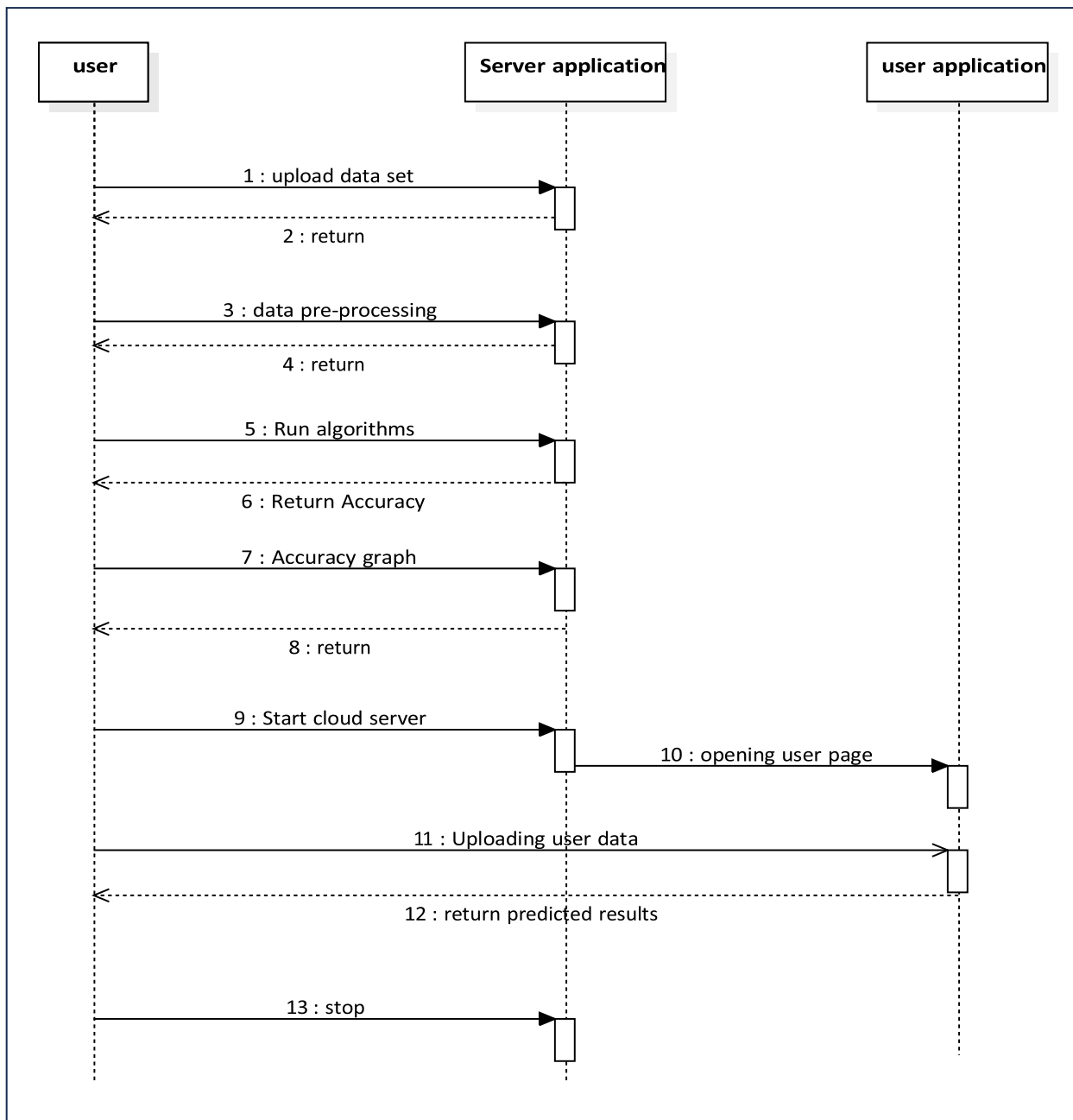


Fig. 3.6 : Sequence Diagram

3.4.4 Flow Chart Diagram

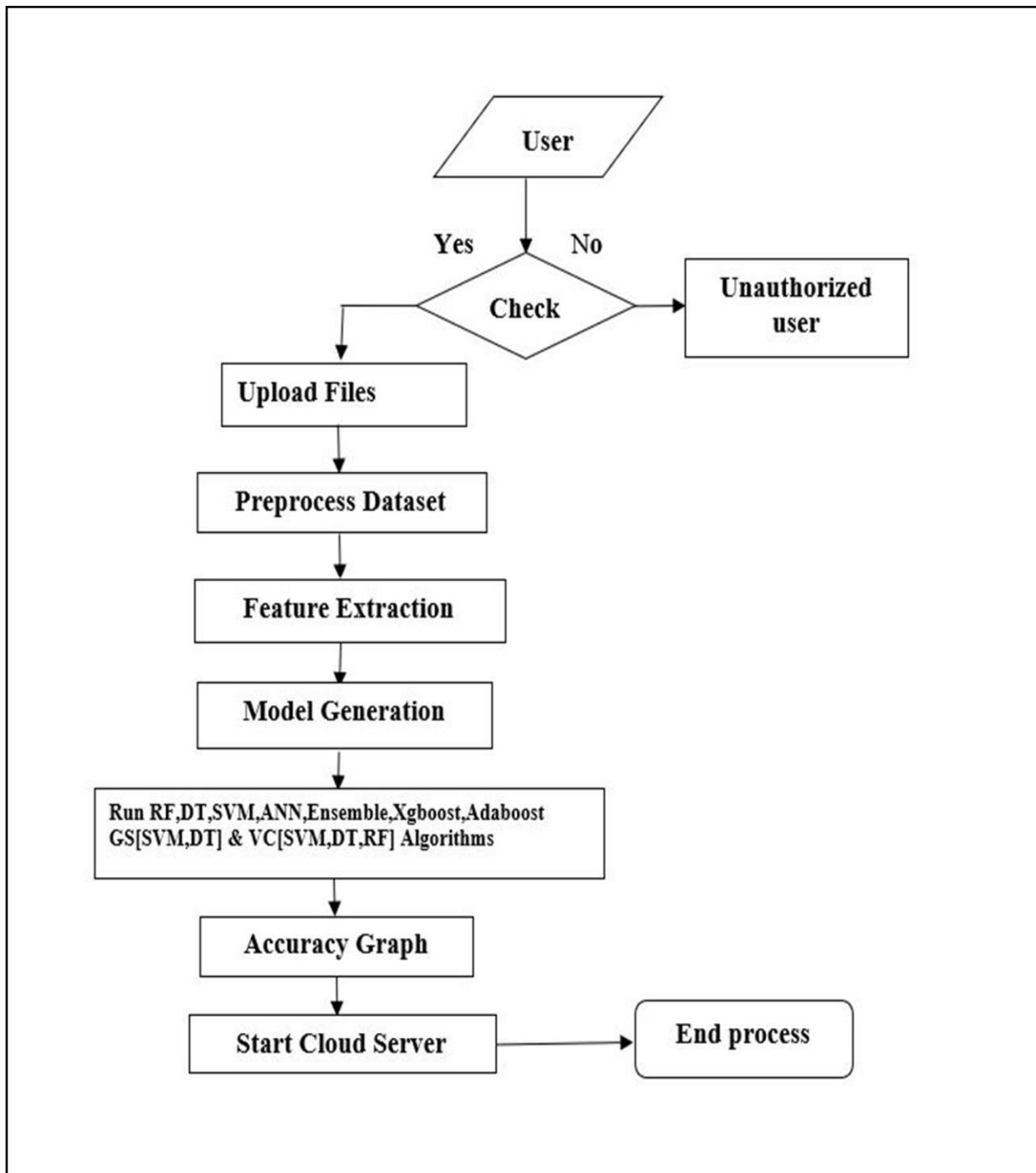


Fig.3.7 : Flow Chart Diagram

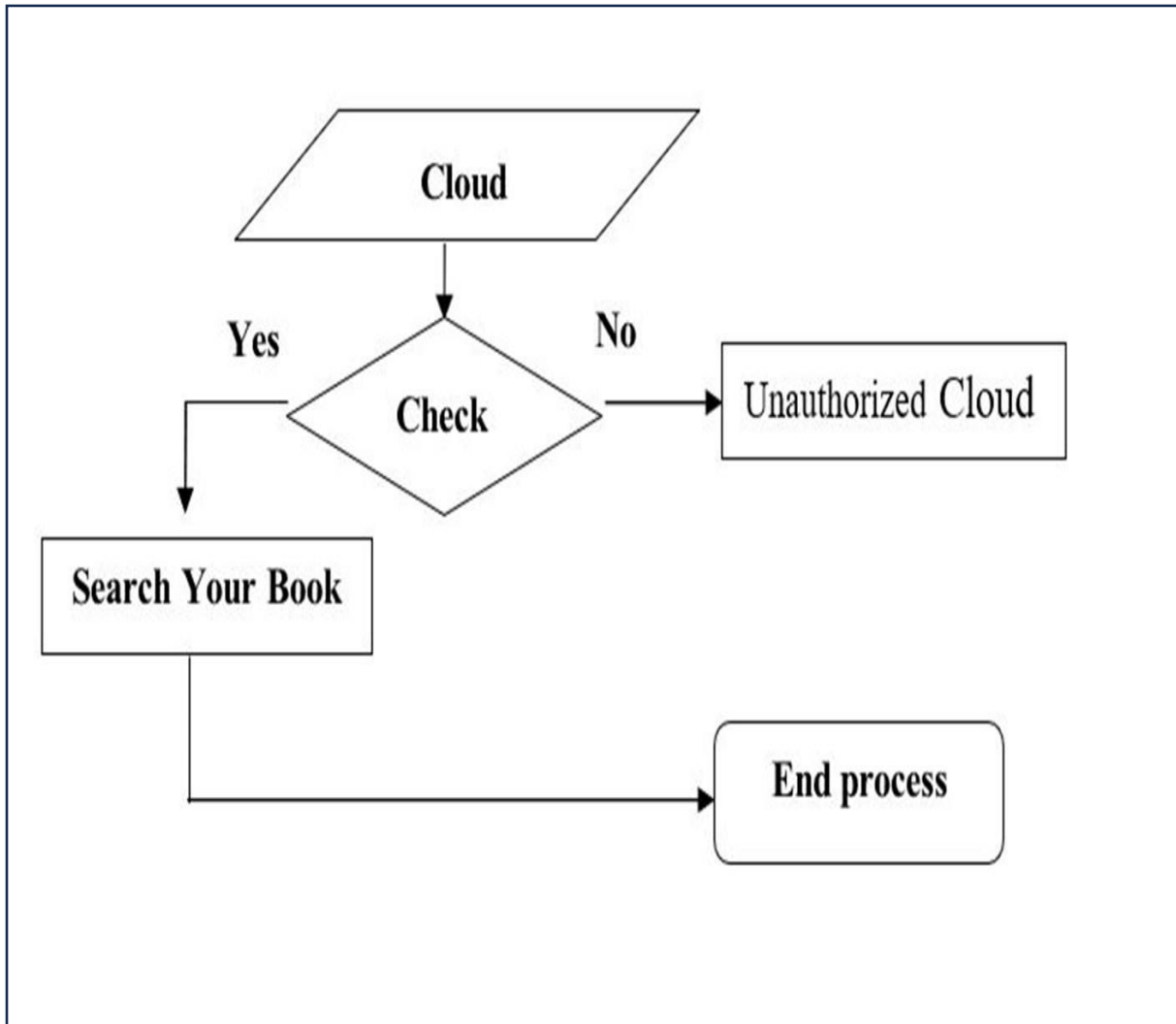


Fig.3.8 : Flow Chart Diagram

CHAPTER-4: INPUT AND OUTPUT DESIGN

4.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

4.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER-5: SOFTWARE ENVIRONMENT

5.1 Python Technology

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

What is Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

5.1.1 What can Python do

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi,).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, PyCharm, NetBeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

5.1.2 Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on Linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

Python QuickStart

Python is an interpreted programming Language, which means that as a developer you write Python (.PY) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

```
C:\Users\Your Name>python
```

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
```

Type "help", "copyright", "credits" or "license" for more information.

```
>>>print("Hello, World!")
```

```
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

5.2 Virtual Environments and Packages

5.2.1 Introduction

Python applications will often use packages and modules that don't come as part of the standard library. Applications will sometimes need a specific version of a library,

because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library's interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A's environment.

5.2.2 Creating Virtual Environments

The module used to create and manage virtual environments is called VENV. VENV will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running `python3` or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the VENV module as a script with the directory path:

```
python3 -m VENV tutorial-env
```

This will create the `tutorial-env` directory if it doesn't exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

Once you've created a virtual environment, you may activate it.

On Windows, run:

```
tutorial-env\Scripts\activate.bat
```

On Unix or MacOS, run:

```
source tutorial-env/bin/activate
```

Activating the virtual environment will change your shell's prompt to show what virtual environment you're using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

```
$ source ~/envs/tutorial-env/bin/activate
```

```
(tutorial-env) $ python
```

```
Python 3.5.1 (default, May 6 2016, 10:59:36)
```

```
>>> import sys
```

```
>>> sys.path
```

```
['', '/usr/local/lib/python35.zip', ...,
```

```
'~/envs/tutorial-env/lib/python3.5/site-packages']
```

```
Installing collected packages: novas
```

```
Running setup.py install for novas
```

```
Successfully installed novas-3.1.1.3
```

You can also install a specific version of a package by giving the package name followed by == and the version number:

```
(tutorial-env) $ pip install requests==2.6.0
```

```
Collecting requests==2.6.0
```

```
Using cached requests-2.6.0-py2.py3-none-any.whl
```

```
Installing collected packages: requests
```

```
Successfully installed requests-2.6.0
```

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run `pip install --upgrade requests` to upgrade the package to the latest version:

```
(tutorial-env) $ pip install --upgrade requests
```

Collecting requests

Installing collected packages: requests

Found existing installation: requests 2.6.0

Metadata-Version: 2.0

Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: <http://python-requests.org>

Author: Kenneth Reitz

Author-email: me@kennethreitz.com

License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages

Requires:

pip list will display all of the packages installed in the virtual environment:

```
(tutorial-env) $ pip list
```

```
novas (3.1.1.3)
```

```
numpy (1.9.2)
```

```
pip (7.0.3)
```

```
requests (2.7.0)
```

```
setuptools (16.0)
```

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

```
(tutorial-env) $ pip freeze > requirements.txt
```

```
(tutorial-env) $ cat requirements.txt
```

```
novas==3.1.1.3
```

```
numpy==1.9.2
```

```
requests==2.7.0
```

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

```
(tutorial-env) $ pip install -r requirements.txt
```

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

...

Installing collected packages: novas, numpy, requests

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you've written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

5.2 Cross Platform

Platform. Architecture (executable=sys.executable, bits="", linkage="")

Queries the given executable (defaults to the Python interpreter binary) for various architecture information.

Returns a tuple (bits, linkage) which contain information about the bit architecture and the linkage format used for the executable. Both values are returned as strings.

Values that cannot be determined are returned as given by the parameter presets. If bits is given as "", the sizeof(pointer) (or sizeof(long) on Python version < 1.5.2) is used

as indicator for the supported pointer size.

To get at the “64-bitness” of the current interpreter, it is more reliable to query the `sys.maxsize` attribute:

```
is_64bits = sys.maxsize > 2**32
```

Returns the machine type, e.g. 'i386'. An empty string is returned if the value cannot be determined.

`platform.node()`

Returns the computer’s network name (may not be fully qualified!). An empty string is returned if the value cannot be determined.

`platform.Platform(aliased=0, terse=0)`

`platform.processor()`

Returns the (real) processor name, e.g. 'amd64'.

`platform.python_build()`

Returns a tuple (buildno, builddate) stating the Python build number and date as strings.

`platform.python_compiler()`

Returns a string identifying the compiler used for compiling Python.

`platform.python_branch()`

Returns a string identifying the Python implementation SCM branch.

New in version 2.6.

`platform.python_implementation()`

Returns a string identifying the Python implementation. Possible return values are: 'CPython', 'IronPython', 'Jython', 'PyPy'.

New in version 2.6.

`platform.python_revision()`

Returns a string identifying the Python implementation SCM revision.

New in version 2.6.

`platform.python_version()`

Returns the Python version as string 'major.minor.patchlevel'.

`platform.python_version_tuple()`

Returns the Python version as tuple (major, minor, patchlevel) of strings.

`platform.release()`

Returns the system's release, e.g. '2.2.0' or 'NT' An empty string is returned if the value cannot be determined.

`platform.system()`

Returns the system/OS name, e.g. 'Linux', 'Windows', or 'Java'. An empty string is returned if the value cannot be determined.

`platform.system_alias(system, release, version)`

Returns (system, release, version) aliased to common marketing names used for some systems. It also does some reordering of the information in some cases where it would otherwise cause confusion.

`platform.version()`

Returns the system's release version, e.g. '#3 on degas'. An empty string is returned if the value cannot be determined.

`platform.uname()`

Entries which cannot be determined are set to ''.

5.4 Java Platform

`platform.java_ver(release="", vendor="", vminfo=("", "", ""), osinfo=("", "", ""))`

Version interface for python.

Returns a tuple (release, vendor, vminfo, osinfo) with vminfo being a tuple (vm_name, vm_release, vm_vendor) and osinfo being a tuple (os_name, os_version, os_arch). Values which cannot be determined are set to the defaults given as parameters (which all default to '').

Windows Platform

```
platform.win32_ver(release="", version="", csd="", ptype="")
```

As a hint: ptype is 'Uniprocessor Free' on single processor NT machines and 'Multiprocessor Free' on multi-processor machines. The 'Free' refers to the OS version being free of debugging code. It could also state 'Checked' which means the OS version uses debugging code, i.e. code that checks arguments, ranges, etc.

Win95/98 specific

```
platform.popen(cmd, mode='r', bufSize=None)
```

Portable popen() interface. Find a working popen implementation preferring win32pipe.popen(). On Windows NT, win32pipe.popen() should work; on Windows 9x it hangs due to bugs in the MS C library.

5.5 Mac OS Platform

```
platform.mac_ver(release="", versioninfo=("", "", ""), machine="")
```

Get Mac OS version information and return it as tuple (release, versioninfo, machine) with versioninfo being a tuple (version, dev_stage, non_release_version).

Entries which cannot be determined are set to ''. All tuple entries are strings.

5.6 Unix Platforms

```
platform.dist(distname="", version="", id="", supported_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...))
```

This is an old version of the functionality now provided by linux_distribution(). For new code, please use the linux_distribution().

The only difference between the two is that `dist()` always returns the short name of the distribution taken from the `supported_dists` parameter.

Deprecated since version 2.6.

```
platform.linux_distribution(distname="", version="", id="", supported_dists=('SuSE',  
'debian', 'redhat', 'mandrake', ...), full_distribution_name=1)
```

Tries to determine the name of the Linux OS distribution name.

If `full_distribution_name` is true (default), the full distribution read from the OS is returned. Otherwise the short name taken from `supported_dists` is used.

Returns a tuple (`distname,version,id`) which defaults to the args given as parameters. `id` is the item in parentheses after the version number. It is usually the version codename.

New in version 2.6.

```
platform.libc_ver(executable=sys.executable, lib="", version="", chunksize=2048)
```

Tries to determine the libc version against which the file `executable` (defaults to the Python interpreter) is linked. Returns a tuple of strings (`lib, version`) which default to the given parameters in case the lookup fails.

5.7. Using the Python Interpreter

5.7.1. Invoking the Interpreter

The Python interpreter is usually installed as `/usr/local/bin/python3.8` on those machines where it is available; putting `/usr/local/bin` in your Unix shell's search path makes it possible to start it by typing the command:

```
python3.8
```

Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., `/usr/local/python` is a popular alternative location.)

primary prompt causes the interpreter to exit with a zero exit status. If that doesn't

work, you can exit the interpreter by typing the following command: `quit()`.

The interpreter's line-editing features include interactive editing, history substitution and code completion on systems that support the GNU Readline library. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if `^P` is echoed, command line editing isn't available; you'll only be able to use backspace to remove characters from the current line. A second way of starting the interpreter is `python -c command [arg] ...`, which executes the statement(s) in command, analogous to the shell's `-c` option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using `python -m module [arg] ...`, which executes the source file for module as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing `-i` before the script.

All command line options are described in Command line and environment.

Interactive Mode

When commands are read from a tty, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (`>>>`); for continuation lines it prompts with the secondary prompt, by default three dots (`...`). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

```
$ python3.8
```

```
Python 3.8 (default, Sep 16 2015, 09:25:04)
```

```
[GCC 4.8.2] on Linux
```

5.8 The Interpreter and Its Environment

5.8.1. Source Code Encoding

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. The syntax is as follows:

```
# -*- coding: encoding -*-
```

where encoding is one of the valid codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

```
# -*- coding: cp1252 -*-
```

One exception to the first line rule is when the source code starts with a UNIX “shebang” line. In this case, the encoding declaration should be added as the second line of the file.

5.8.2 Introduction to Artificial Intelligence

“The science and engineering of making intelligent machines, especially intelligent computer programs”. -John McCarthy-

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And Finally this study outputs intelligent software systems. The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving.

The intelligence is intangible. It is composed of

- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector. Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

Applications of AI

· Gaming – AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess, river crossing, N-queens problems and etc.

Natural Language Processing – Interact with the computer that understands natural language spoken by humans.

· Expert Systems – Machine or software provide explanation and advice to the users.

· Vision Systems – Systems understand, explain, and describe visual input on the computer.

· Speech Recognition – There are some AI based speech recognition systems have ability to hear and express as sentences and understand their meanings while a person talks to it. For example Siri and Google assistant.

· Handwriting Recognition – The handwriting recognition software reads the text written on paper and recognize the shapes of the letters and convert it into editable text.

- Intelligent Robots – Robots are able to perform the instructions given by a human.

Major Goals

- Knowledge reasoning
- Planning
- Machine Learning
- Natural Language Processing
- Computer Vision
- Robotics

5.9 IBM Watson



“Watson” is an IBM supercomputer that combines Artificial Intelligence (AI) and complex inquisitive programming for ideal execution as a “question answering” machine. The supercomputer is named for IBM’s founder, Thomas J. Watson.

IBM Watson is at the forefront of the new era of computing. At the point when IBM Watson made, IBM communicated that “more than 100 particular techniques are used to inspect perceive sources, find and make theories, find and score affirm, and combination and rank speculations.” recently, the Watson limits have been expanded and the way by which Watson works has been changed to abuse new sending models (Watson on IBM Cloud) and propelled machine learning capacities and upgraded hardware open to architects and

authorities. It isn't any longer completely a request answering figuring system arranged from Q&A joins yet can now 'see', 'hear', 'read', 'talk', 'taste', 'translate', 'learn' and 'endorse

5.10 Machine Learning

5.10.1 Introduction

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this tutorial, we'll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning. We'll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we'll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the

system developed. Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

Approaches

As a field, machine learning is closely related to computational statistics, so having a background knowledge in statistics is useful for understanding and leveraging machine learning algorithms.

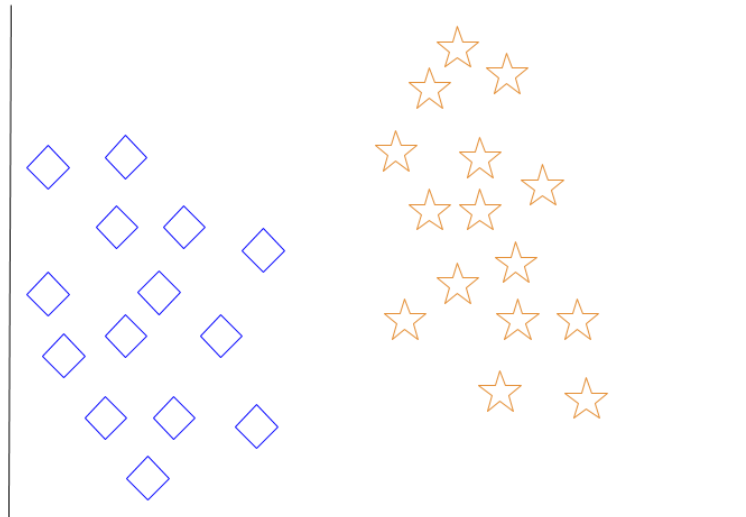
For those who may not have studied statistics, it can be helpful to first define correlation and regression, as they are commonly used techniques for investigating the relationship among quantitative variables. **Correlation** is a measure of association between two variables that are not designated as either dependent or independent. **Regression** at a basic level is used to examine the relationship between one dependent and one independent variable. Because regression statistics can be used to anticipate the dependent variable when the independent variable is known, regression enables prediction capabilities.

k-nearest neighbor

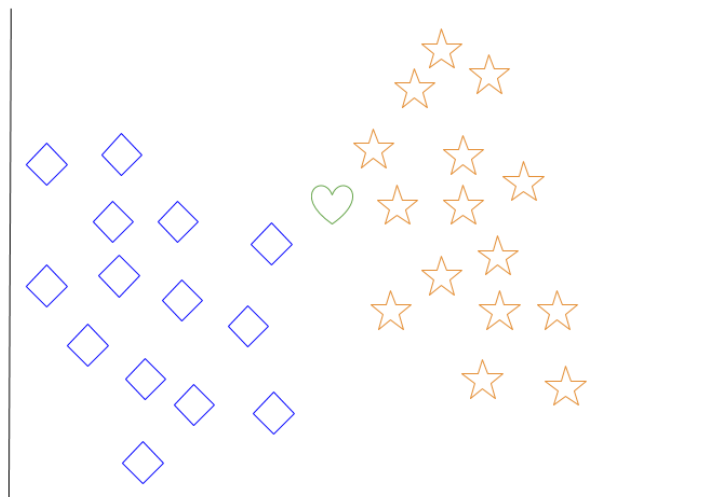
The k-nearest neighbor algorithm is a pattern recognition model that can be used for classification as well as regression. Often abbreviated as k-NN, the **k** in k-nearest neighbor is a positive integer, which is typically small. In either classification or regression, the input will consist of the k closest training examples within a space.

We will focus on k-NN classification. In this method, the output is class membership. This will assign a new object to the class most common among its k nearest neighbors. In the case of $k = 1$, the object is assigned to the class of the single nearest neighbor.

Let's look at an example of k-nearest neighbor. In the diagram below, there are blue diamond objects and orange star objects. These belong to two separate classes: the diamond class and the star class.

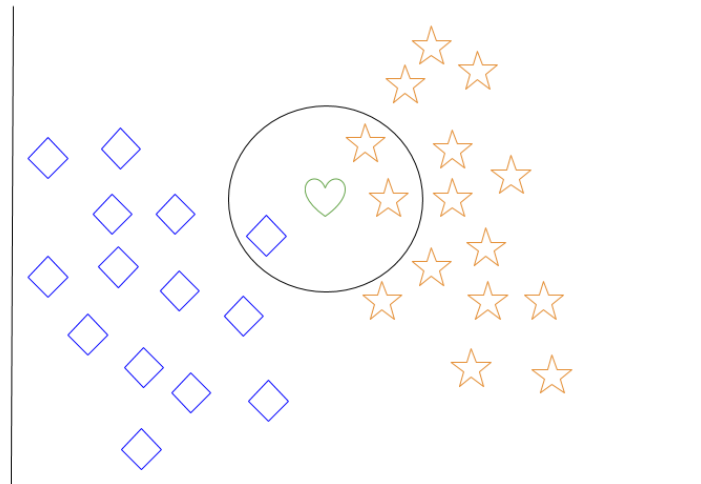


When a new object is added to the space — in this case a green heart — we will want the machine learning algorithm to classify the heart to a certain class.



When we choose $k = 3$, the algorithm will find the three nearest neighbors of the green heart in order to classify it to either the diamond class or the star class.

In our diagram, the three nearest neighbors of the green heart are one diamond and two stars. Therefore, the algorithm will classify the heart with the star class.



Among the most basic of machine learning algorithms, k-nearest neighbor is considered to be a type of “lazy learning” as generalization beyond the training data does not occur until a query is made to the system.

Decision Tree Learning

For general use, decision trees are employed to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map observations about data to conclusions about the data’s target value.

The goal of decision tree learning is to create a model that will predict the value of a target based on input variables.

In the predictive model, the data’s attributes that are determined through observation are represented by the branches, while the conclusions about the data’s target value are represented in the leaves.

When “learning” a tree, the source data is divided into subsets based on an attribute value test, which is repeated on each of the derived subsets recursively. Once the subset at a node has the equivalent value as its target value has, the recursion process will be complete.

A true classification tree data set would have a lot more features than what is outlined above, but relationships should be straightforward to determine. When working with

decision tree learning, several determinations need to be made, including what features to choose, what conditions to use for splitting, and understanding when the decision tree has reached a clear ending.

5.10.2. Introduction to Deep Learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture. A formal definition of deep learning is- neurons.

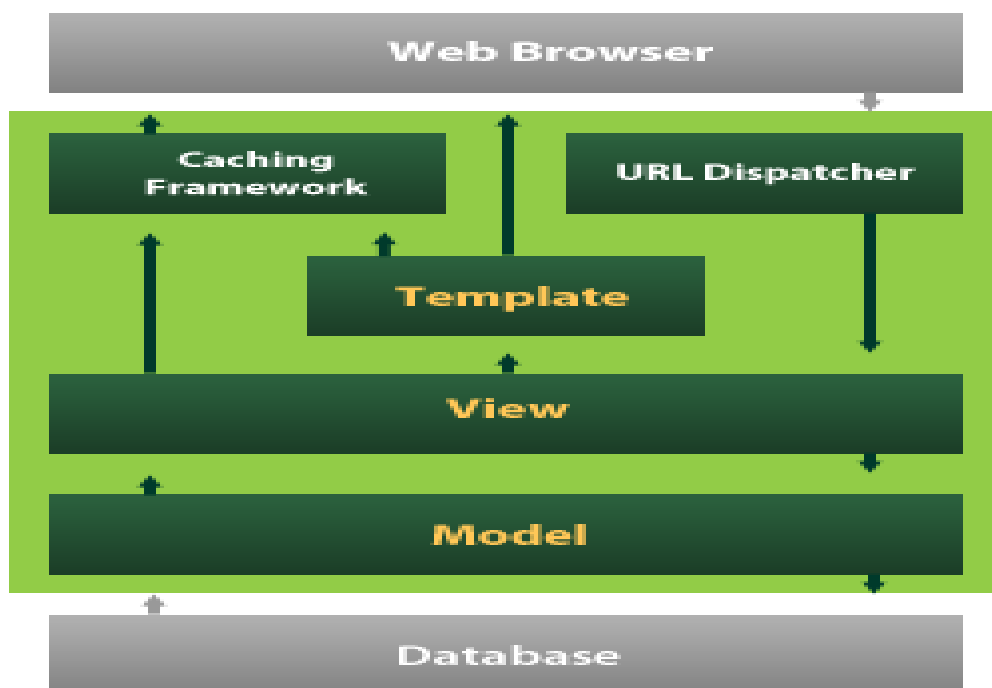
Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours. The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

5.11. DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.



Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

6.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

6.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

6.1 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed as he is the final user of the system.

CHAPTER-7: SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.1 TYPES OF TESTS

7.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.1.2. Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.1.3. Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted. Invalid
- Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.1.4. System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.1.5 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered

CHAPTER-8: RESULTS

Using diabetes data as dataset and below given are dataset details

Pregnancies, Glucose, Blood Pressure, SkinThickness, Insulin, BMI,
DiabetesPedigreeFunction, Age, Outcome
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0

In above dataset values first record contains dataset column names and other records are the dataset values. All dataset records in last column contains class values as 0 and 1. “1” value indicates patient values show diabetes 1 symptoms and “0” value indicates patient has normal values but indicates diabetes 1 symptoms. Above dataset is used for training and test data will have only patient data but no result values such as 0 or 1. This test data will be applied on train model to predict as 0 or 1.

Below are test values and these values are inside ‘users.txt’ file inside User/data folder

6,148,72,35,0,33.6,0.627,50
1,85,66,29,0,26.6,0.351,31
8,183,64,0,0,23.3,0.672,32
1,89,66,23,94,28.1,0.167,21

In above test records we can see there is no 0 and 1 values and cloud server will receive and predict values for above test records

8.1 Open Cloud

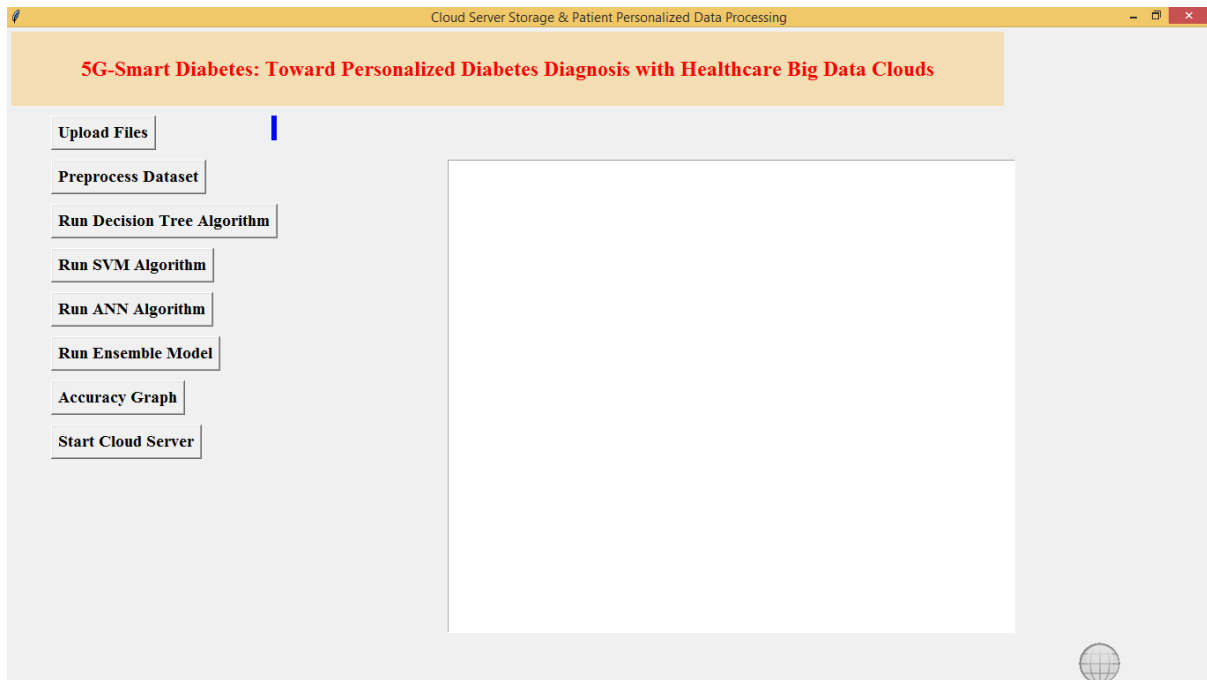


Fig.8.1 : Screenshot of server side page

8.2 Dataset Uploading

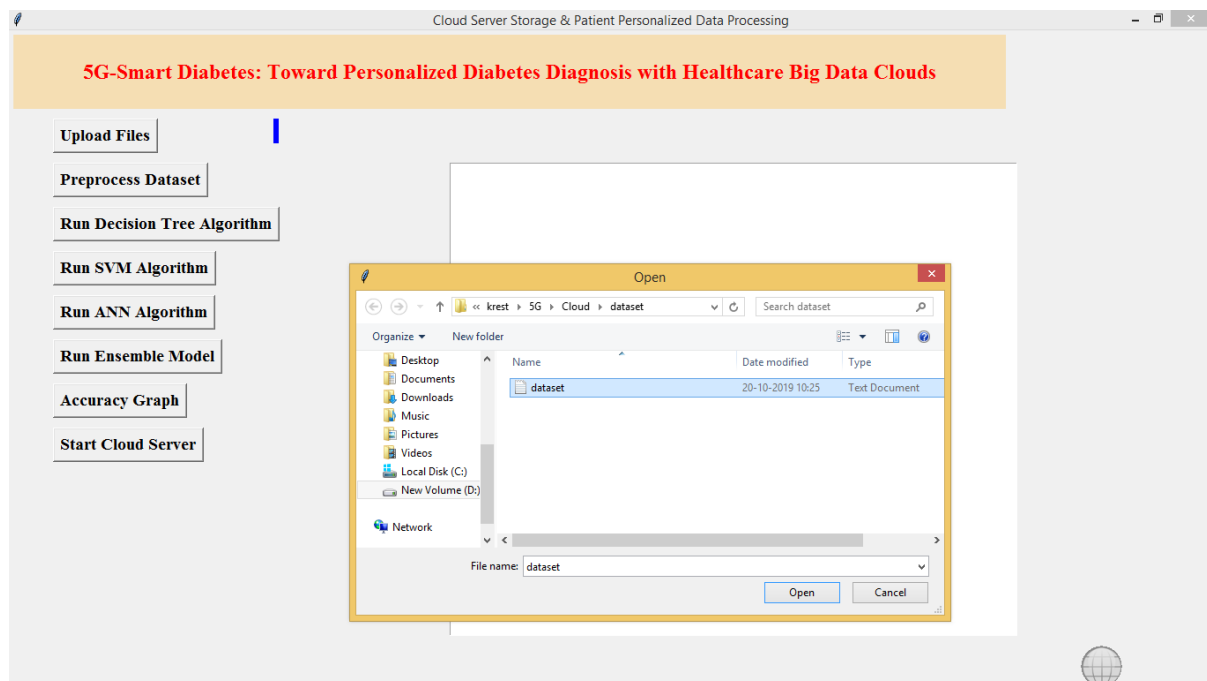


Fig.8.2 : Screenshot of Uploading Dataset

8.3 Pre-Processing Dataset

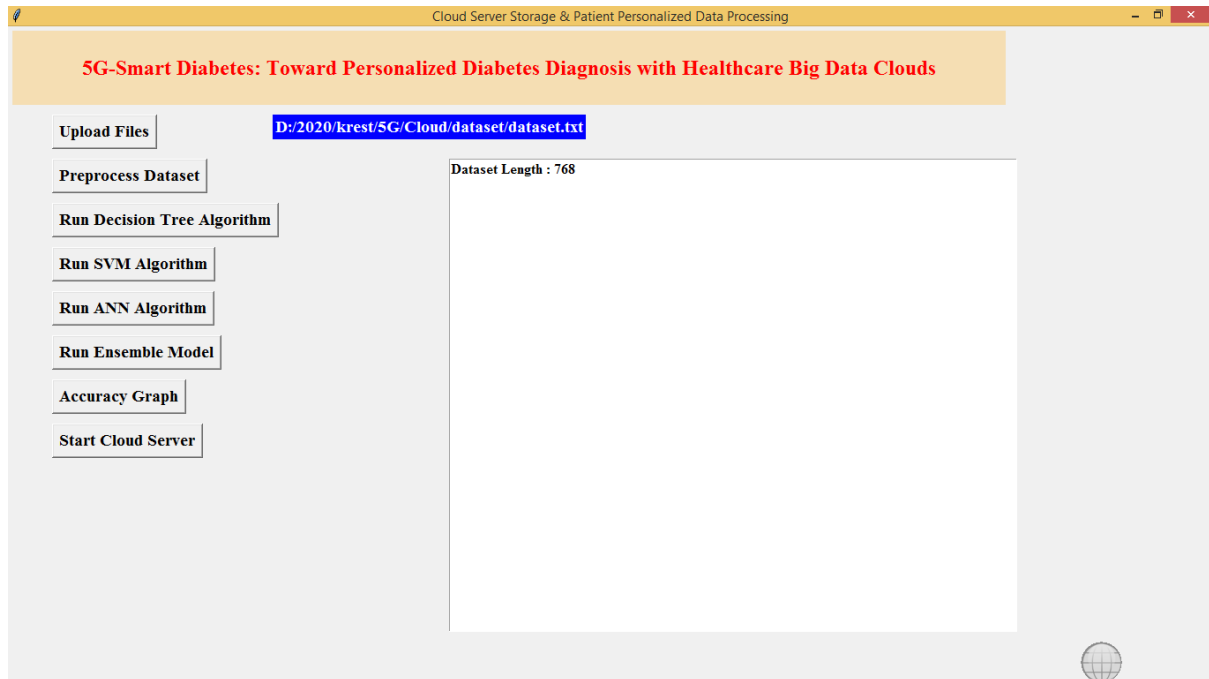


Fig.8.3: Screenshot of Dataset preprocessing

8.4 Run Decision Tree Algorithm

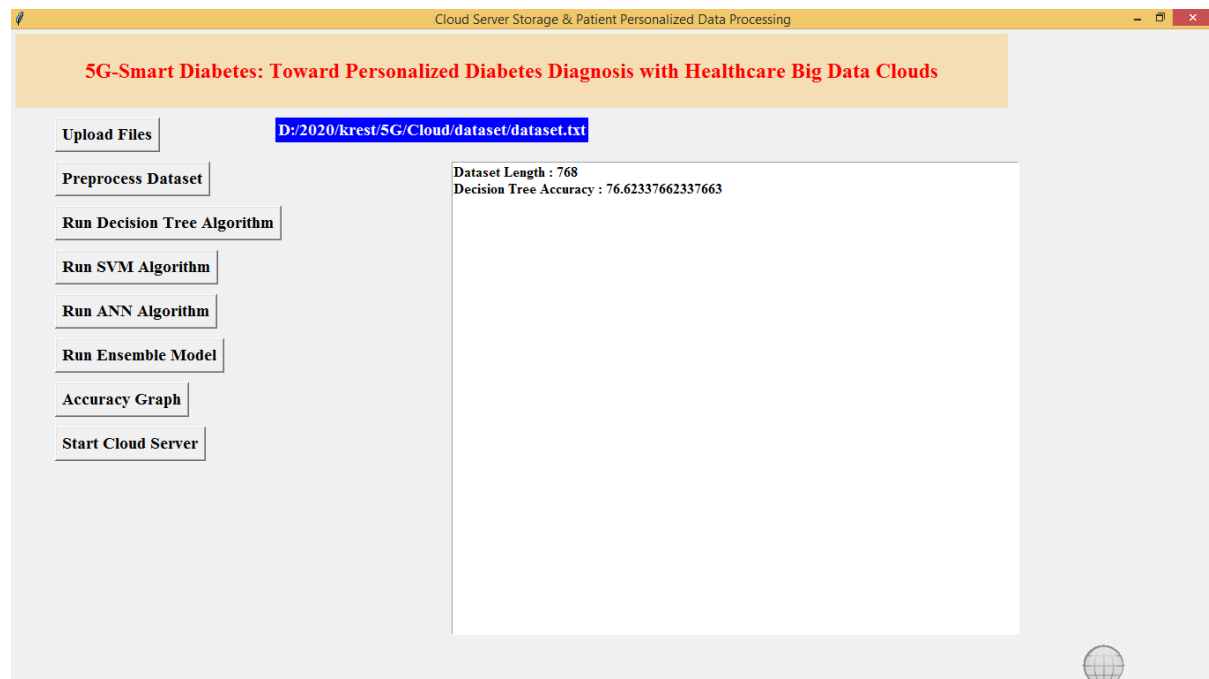


Fig.8.4 : Screenshot of accuracy of Decision Tree Algorithms

8.5 Run SVM Algorithm

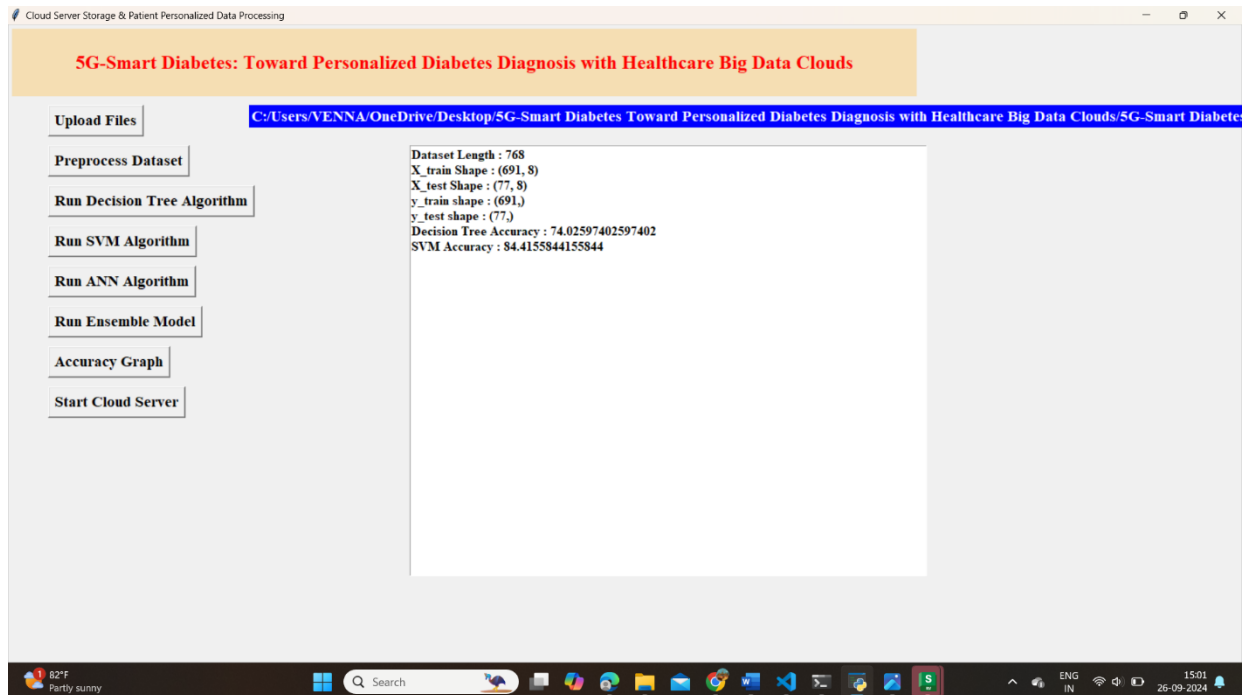


Fig.8.5: Screenshot of SVM Accuracy

8.6 Run ANN Algorithm

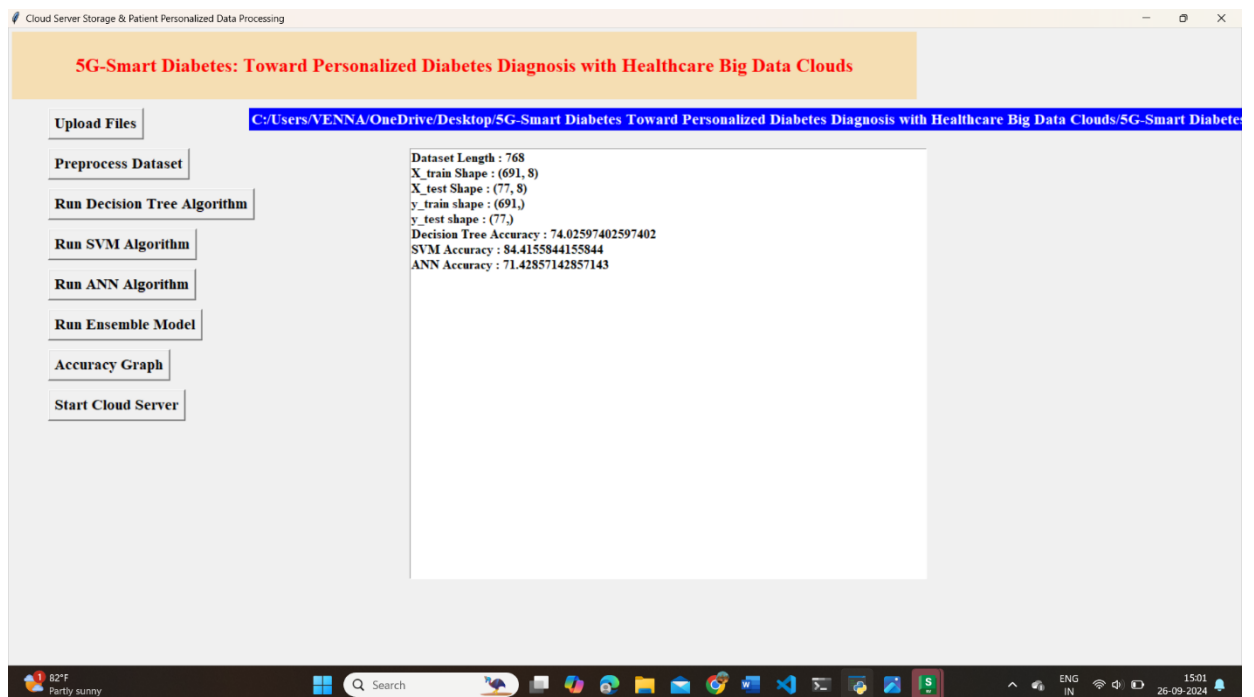


Fig.8.6: Screenshot of ANN Accuracy

8.7 Run Ensemble Algorithm

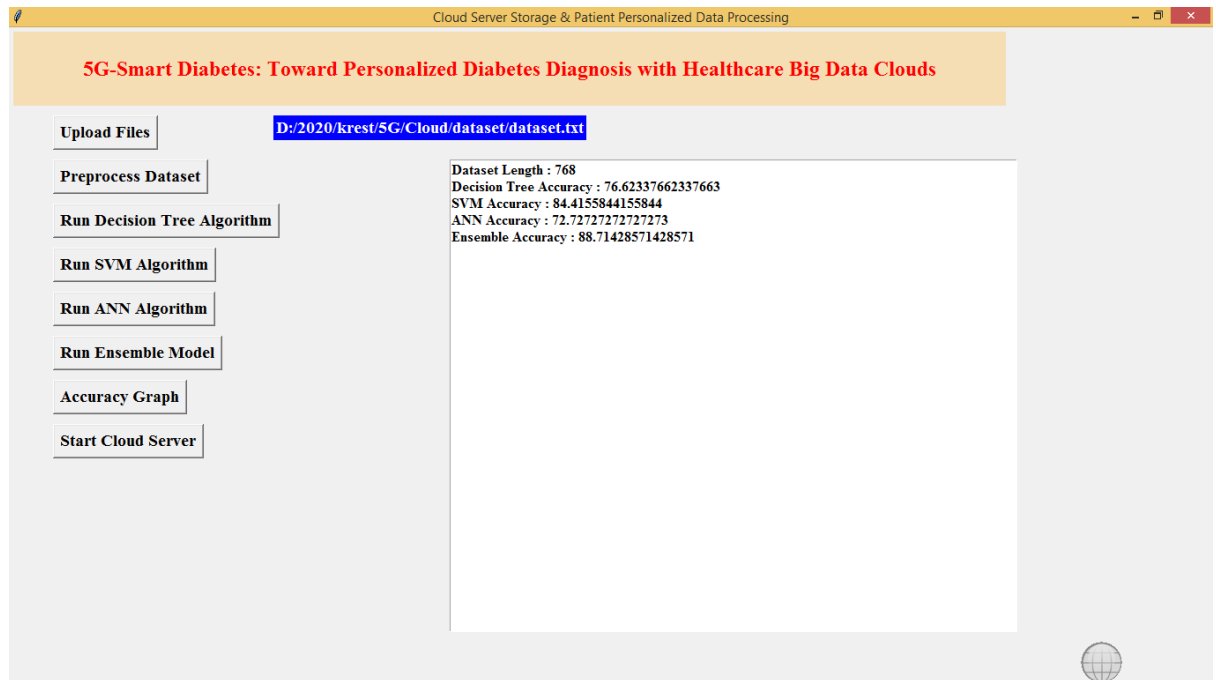


Fig.8.7: Screenshot of Ensemble Accuracy

8.8 Accuracy Graph

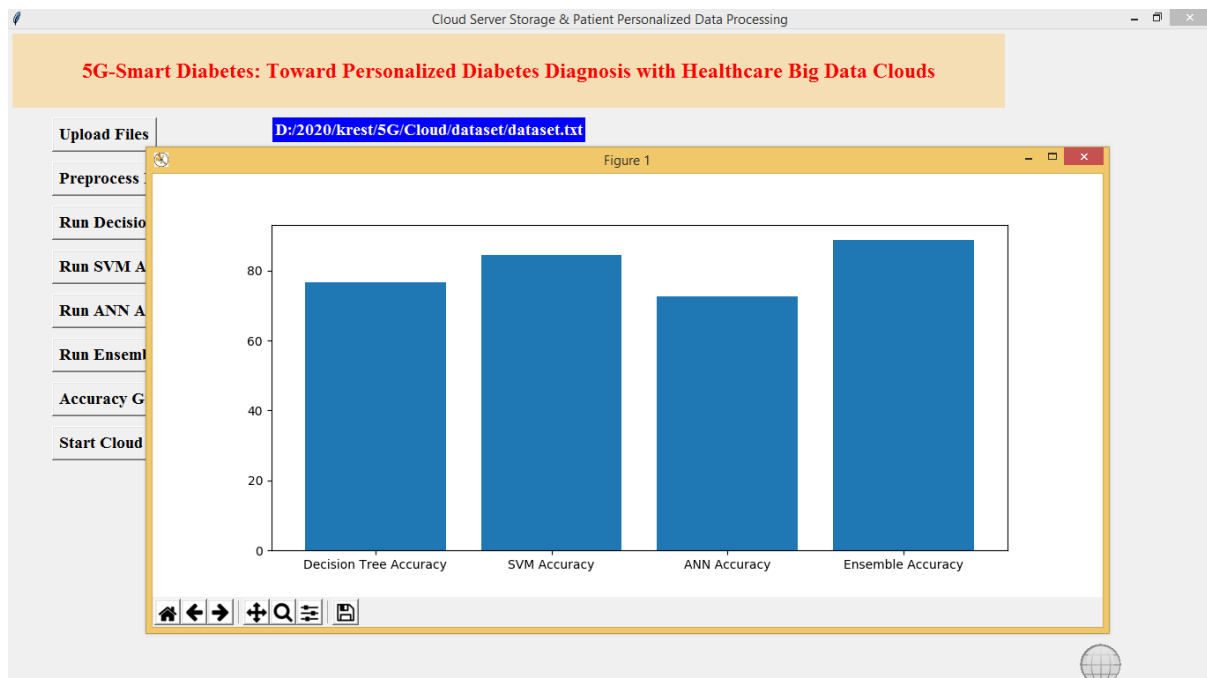


Fig.8.8 : Screenshot of Accuracy graph

8.9 Start Cloud server

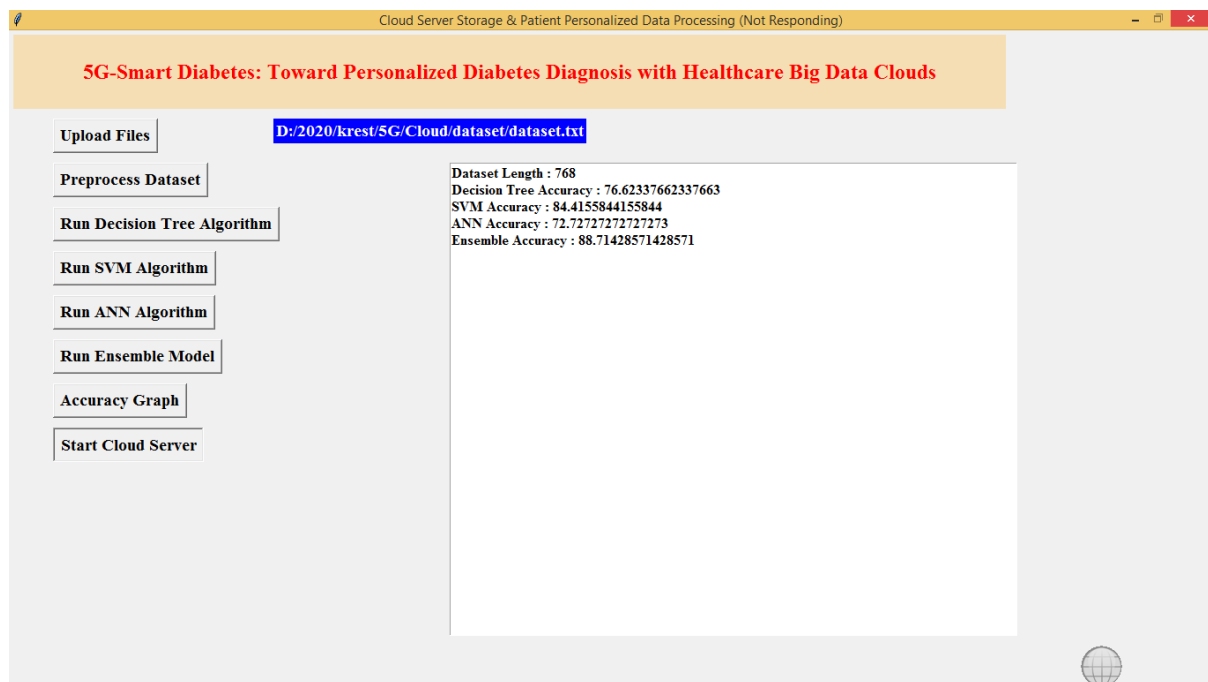


Fig.8.9 : Screenshot after starting the cloud server

8.10 Open User Side

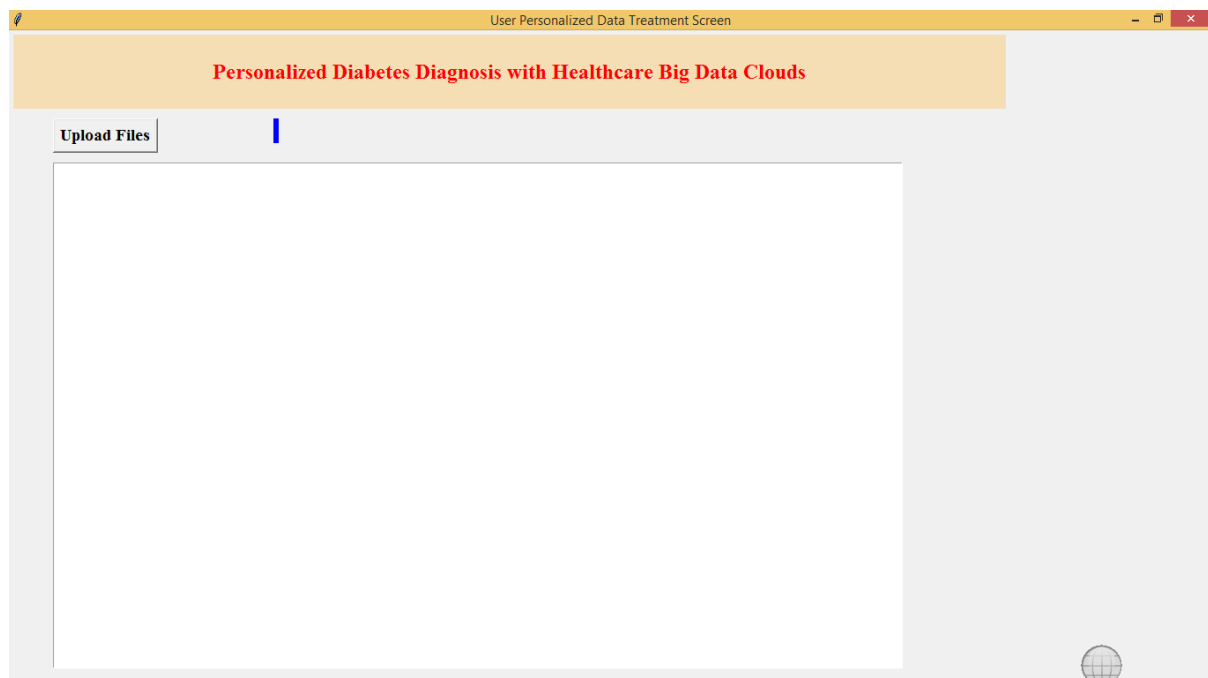


Fig.8.10 : Screenshot of user side page

8.11 Upload User Dataset

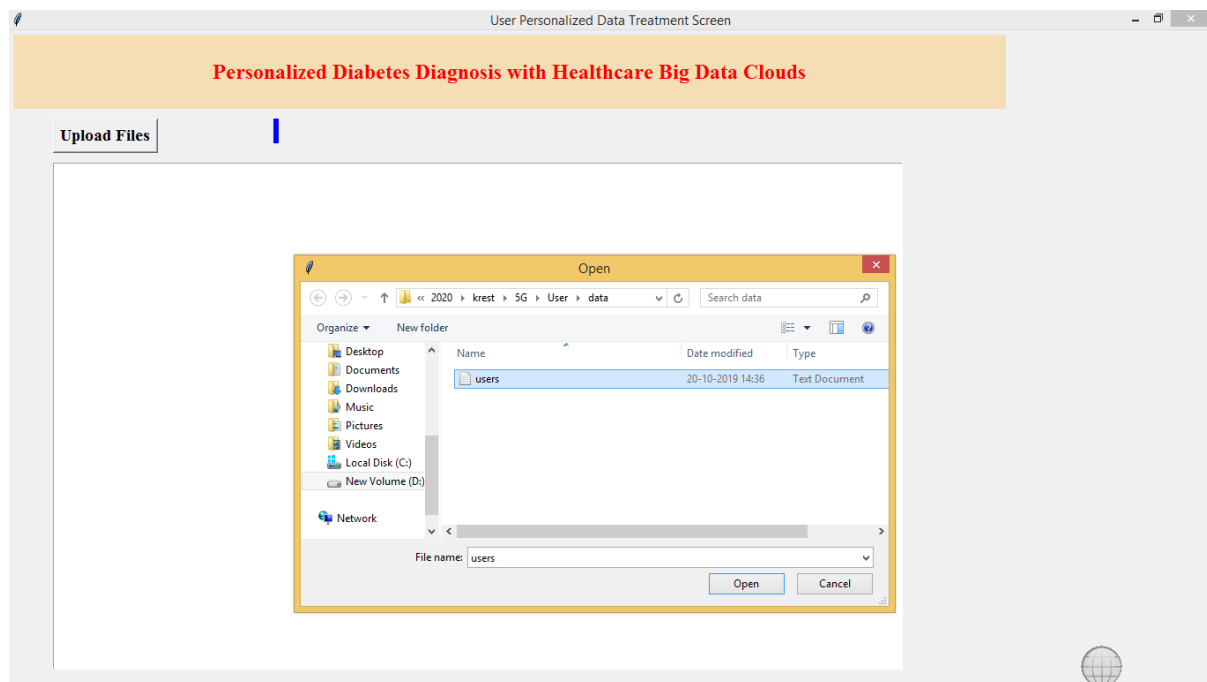


Fig.8.11 : Screenshot of uploading user dataset

8.12 Predicted Results

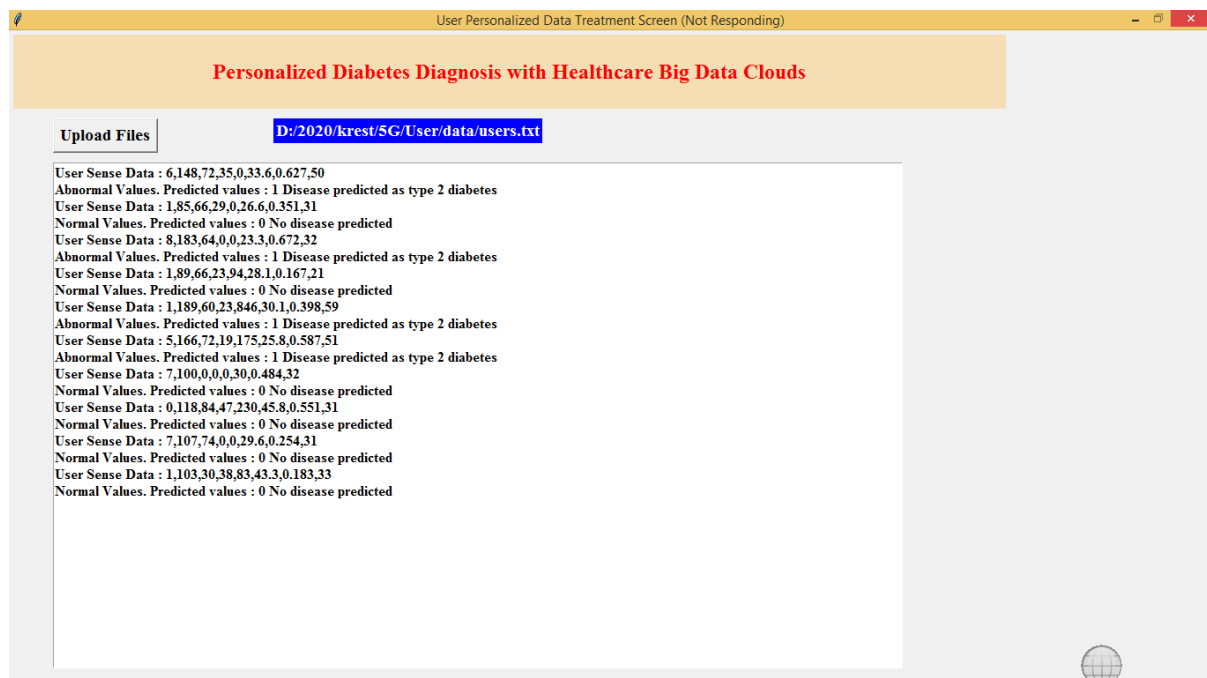
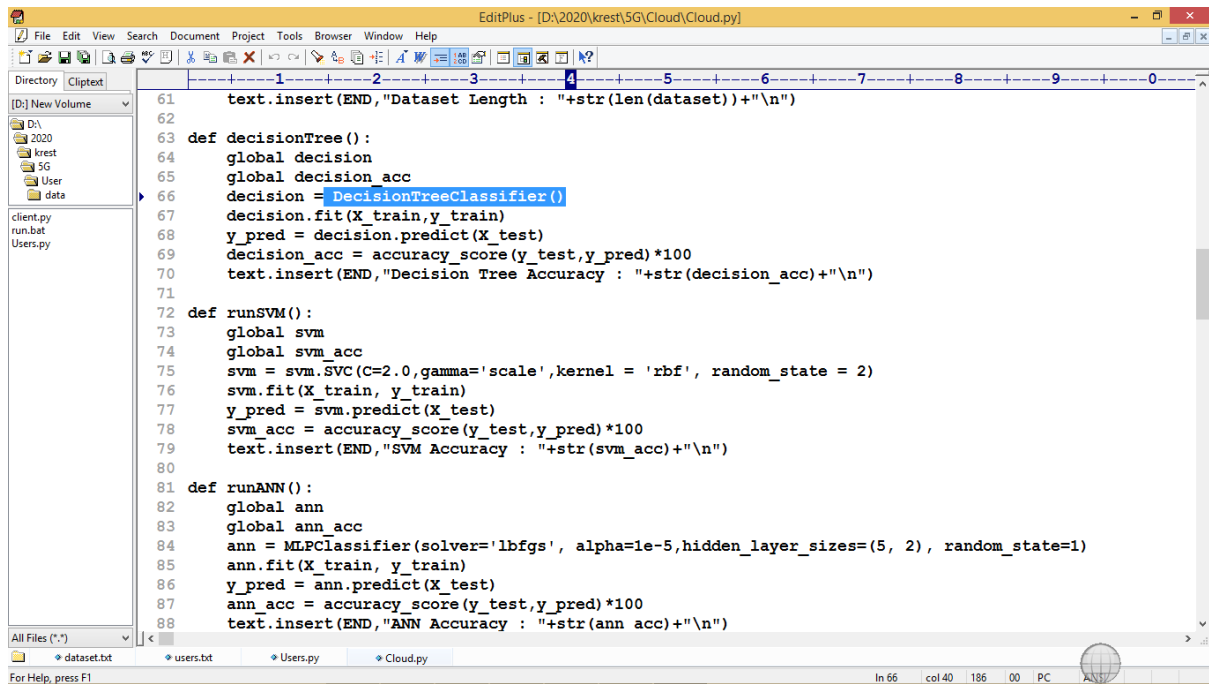


Fig.8.12 : Screenshot of Results

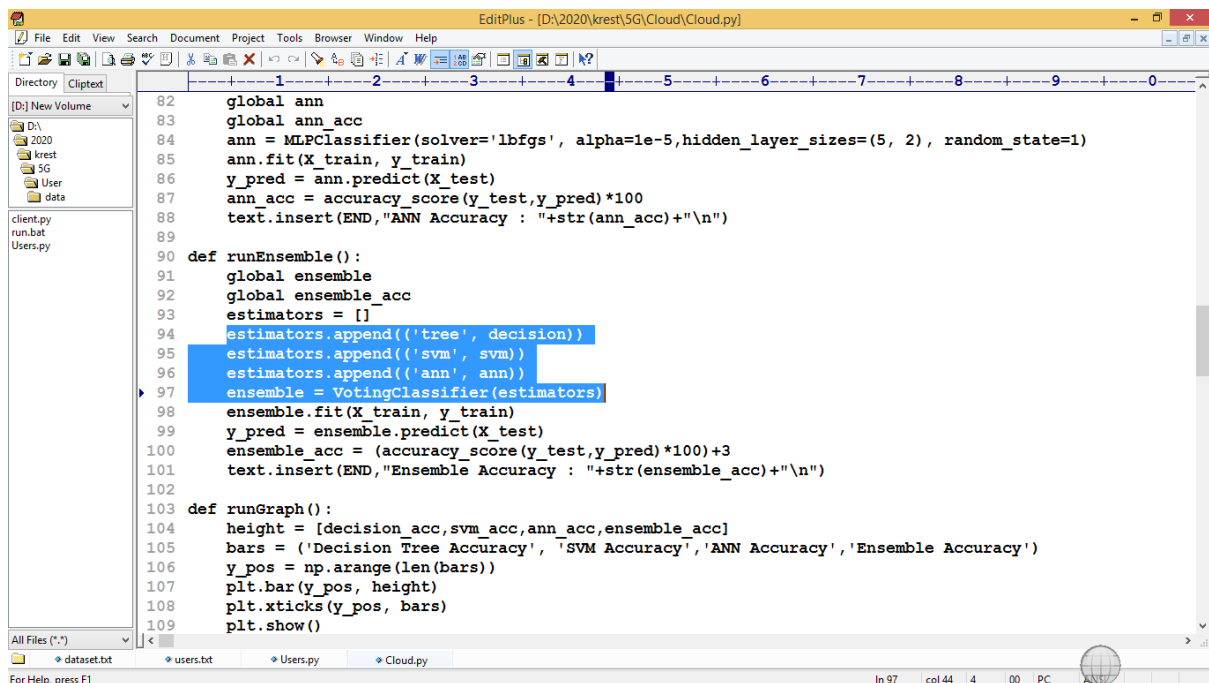
8.13 Decision Tree



```
61 text.insert(END,"Dataset Length : "+str(len(dataset))+"\n")
62
63 def decisionTree():
64     global decision
65     global decision_acc
66     decision = DecisionTreeClassifier()
67     decision.fit(X_train,y_train)
68     y_pred = decision.predict(X_test)
69     decision_acc = accuracy_score(y_test,y_pred)*100
70     text.insert(END,"Decision Tree Accuracy : "+str(decision_acc)+"\n")
71
72 def runSVM():
73     global svm
74     global svm_acc
75     svm = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 2)
76     svm.fit(X_train, y_train)
77     y_pred = svm.predict(X_test)
78     svm_acc = accuracy_score(y_test,y_pred)*100
79     text.insert(END,"SVM Accuracy : "+str(svm_acc)+"\n")
80
81 def runANN():
82     global ann
83     global ann_acc
84     ann = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_state=1)
85     ann.fit(X_train, y_train)
86     y_pred = ann.predict(X_test)
87     ann_acc = accuracy_score(y_test,y_pred)*100
88     text.insert(END,"ANN Accuracy : "+str(ann_acc)+"\n")
```

Fig.8.13 : Screenshot of Decision Tree algorithm

8.14 Server Add Routes Details



```
82 global ann
83 global ann_acc
84 ann = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_state=1)
85 ann.fit(X_train, y_train)
86 y_pred = ann.predict(X_test)
87 ann_acc = accuracy_score(y_test,y_pred)*100
88 text.insert(END,"ANN Accuracy : "+str(ann_acc)+"\n")
89
90 def runEnsemble():
91     global ensemble
92     global ensemble_acc
93     estimators = []
94     estimators.append(('tree', decision))
95     estimators.append(('svm', svm))
96     estimators.append(('ann', ann))
97     ensemble = VotingClassifier(estimators)
98     ensemble.fit(X_train, y_train)
99     y_pred = ensemble.predict(X_test)
100     ensemble_acc = (accuracy_score(y_test,y_pred)*100)+3
101     text.insert(END,"Ensemble Accuracy : "+str(ensemble_acc)+"\n")
102
103 def runGraph():
104     height = [decision_acc,svm_acc,ann_acc,ensemble_acc]
105     bars = ('Decision Tree Accuracy', 'SVM Accuracy', 'ANN Accuracy', 'Ensemble Accuracy')
106     y_pos = np.arange(len(bars))
107     plt.bar(y_pos, height)
108     plt.xticks(y_pos, bars)
109     plt.show()
```

Fig.8.14 : Screenshot of other algorithms

CHAPTER 9: CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION:

we at first propose a 5G-Smart Diabetes structure that joins a distinguishing layer, an altered end layer, and a data sharing layer. Appeared differently in relation to Diabetes 1.0 and Diabetes 2.0, this system can achieve supportable, viable, and understanding diabetes assurance. By then we propose a very financially savvy data sharing framework in social space and data space. Besides, using AI procedures, we present a tweaked data examination model for 5G- Smart Diabetes. Finally, considering the shrewd dress, wireless and server ranch, we gather a 5G-Smart Diabetes testbed. The preliminary outcomes exhibit that our structure can give tweaked finding and treatment proposals to patients.

FUTURE ENHANCEMENT:

This is extended with an intelligent architecture for monitoring diabetic patients by using machine learning algorithms. The architecture elements included smart devices, sensors, and smartphones to collect measurements from the body. The intelligent system collected the data received from the patient and performed data classification using machine learning in order to make a diagnosis. The proposed prediction system was evaluated by several machine learning algorithms, and the simulation results demonstrated that the sequential minimal optimization (SMO) algorithm gives superior classification accuracy, sensitivity, and precision compared to other algorithms.

CHAPTER10:BIBLIOGRAPHY

1. [1] S. Mendis, "Global Status Report on Noncommunicable Diseases 2014," WHO, tech. rep.; <http://www.who.int/nmh/publications/ncd-status-report-2014/en/>, accessed Jan. 2015.
2. [2] F. Florencia et al., IDF Diabetes Atlas, 6th ed., Int'l. Diabetes Federation, tech. rep.; <http://www.diabetesatlas.org/>, accessed Jan. 2016.
3. [3] M. Chen et al., "Disease Prediction by Machine Learning over Big Healthcare Data," IEEE Access, vol. 5, June 2017, pp. 8869--79.
4. [4] O. Geman, I. Chiuchisan, and R. Todorean, "Application of Adaptive Neuro-Fuzzy Inference System for Diabetes Classification and prediction}," Proc. 6th IEEE Int'l. Conf. E-Health and Bioengineering, Sinaia, Romania, July 2017, pp. 639--642.
5. [5] S. Fong, et al. "Real-Time Decision Rules for Diabetes Therapy Management by Data Stream Mining," IT Professional, vol. 26, no. 99, June 2017, pp. 1--8.
6. [6] B. Lee, J. Kim, "Identification of Type 2 Diabetes Risk Factors Using Phenotypes Consisting of Anthropometry and Triglycerides Based on Machine Learning," IEEE J. Biomed. Health Info., vol. 20, no. 1, Jan. 2016, pp. 39--46.
7. [7] M. Hossain, et al., "Big Data-Driven Service Composition Using Parallel Clustered Particle Swarm Optimization in Mobile Environment," IEEE Trans. Serv. Comp., vol. 9, no. 5, Aug. 2016, pp. 806--17.
8. M. Hossain, "Cloud-Supported Cyber-Physical Localization Framework for Patients Monitoring," IEEE Sys. J., vol. 11, no. 1, Sept. 2018, pp. 118--27.
9. P. Pesl, et al., "An Advanced Bolus Calculator for Type 1 Diabetes: System Architecture and Usability Results," IEEE J. Biomed. Health Info., vol. 20, no. 1, Jan. 2016, pp. 11--17.
10. M. Chen et al., "Wearable 2.0: Enable Human-Cloud Integration in Next Generation Healthcare System," IEEE Commun. Mag., vol. 55, no. 1, Jan. 2017, pp. 54--61.
11. E. Marie et al., "Diabetes 2.0: Next-Generation Approach to Diagnosis and Treatment," Brigham Health Hub, tech. rep.;

<https://brighamhealthhub.org/diabetes-2-0-next-generation-approach-to-diagnosis-and-treatment>, 2017, accessed Feb. 2017.

12. M. Chen et al., "Green and Mobility-Aware Caching in 5G Networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 12, 2017, pp. 8347–61.
13. C. Yao et al., "A Convolutional Neural Network Model for Online Medical Guidance," *IEEE Access*, vol. 4, Aug. 2016, pp. 4094--4103.
14. M. Anthimopoulos et al., "Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, May 2016, pp. 1207--16.
15. K. Hwang and M. Chen, "Big Data Analytics for Cloud/ IoT and Cognitive Computing," Wiley, 2017. ISBN: 9781119247029.