

# REMOTE CARDIOVASCULAR HEALTH MONITORING SYSTEM WITH AUTO-DIAGNOSIS

## A PROJECT REPORT

*Submitted in partial fulfilment for the award of the degree of*

## Bachelor of Technology

in

## Electronics and Communication

*by*

Name	Registration Number
SAURAV MOHAPATRA	15BEC0002
SUNDERAM SAH	15BEC0004
ARKA PROVO MUKHOPADHYAY	15BEC0402

Under the guidance of  
**Dr. Budhaditya Bhattacharya**  
**SENSE (School of Electronics Engineering)**  
**VIT, Vellore**



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**APRIL 2019**

## **DECLARATION**

We hereby declare that the thesis entitled “**REMOTE CARDIOVASCULAR HEALTH MONITORING SYSTEM WITH AUTO-DIAGNOSIS**” submitted by us, for the award of the degree of *Bachelor of Technology in Electronics and Communication* to VIT is a record of bonafide work carried out by us under the supervision of Dr. Budhaditya Bhattacharya.

We declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

**Place:** Vellore

**Date:** 12/04/19

Saurav Mohapatra

Sunderam Sah

Arka Provo Mukhopadhyay

## **CERTIFICATE**

This is to certify that the thesis entitled “**REMOTE CARDIOVASCULAR HEALTH MONITORING SYSTEM WITH AUTO-DIAGNOSIS**” submitted by **Saurav Mohapatra (15BEC0002), Sunderam Sah (15BEC0004) and Arka Provo Mukhopadhyay (15BEC0402), School of Electronics Engineering (SENSE), VIT Vellore**, for the award of the degree of *Bachelor of Technology in Electronics and Communication*, is a record of bonafide work carried out by them under my supervision during the period, 01. 12. 2018 to 30.04.2019, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

**Place: Vellore**

**Date: 12/04/2019**

**Signature of the Guide**

**Internal Examiner**

**External Examiner**

**Head of the Department  
SENSE**

## **ACKNOWLEDGEMENTS**

At the outset, we shall present our sincere gratitude to our guide, Dr. Budhaditya Bhattacharya, whose invaluable support and assistance has been the primary reason for successful completion of this work. Without his constant guidance at every step, this endeavour would have remained a distant dream. We would also acknowledge the help of other faculty and students as a collective, who, albeit not directly involved, had set in forth a series of positive increments in our academic growth in college, which culminated in our pursuit of this project. Lastly, but no less importantly, we would thank the authors of the innumerable papers and books that we had to peruse in order to carry this thesis through to the end.

**Place:** Vellore

**Date:** 12/04/19

Saurav Mohapatra

Sunderam Sah

Arka Provo Mukhopadhyay

## **EXECUTIVE SUMMARY**

Cardio Vascular Diseases (CVD) led to almost one-third of deaths in the world which makes it most critical topic to study and provide efficient solutions. Considering that, a system is proposed to remotely monitor health of CVD patients using machine-to-machine (M2M) technology. The system contain various health sensors for measuring heart rate, Blood Pressure (BP), ECG, Body Mass Index (BMI), body temperature and general medical interview done by a chatbot. Using these sensors and chatbot, all information regarding current health condition of the patient is gathered by Raspberry Pi (processing unit), then using Machine Learning (ML) algorithm (in our case- Multi-layer Perceptron Neural Network), it automatically diagnoses the gathered data to suggest medication to the patient. Lastly, the final health report is sent to doctor's email for review purposes. We have referred to the famous UCI dataset for heart diseases for training our ML algorithm.

## **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>ACKNOWLEDGEMENTS</b>	i
	<b>EXECUTIVE SUMMARY</b>	ii
	<b>TABLE OF CONTENTS</b>	iii
	<b>LIST OF FIGURES</b>	iv
	<b>LIST OF TABLES</b>	v
	<b>LIST OF ABBREVIATIONS</b>	vi
	<b>SYMBOLS AND NOTATIONS</b>	vii
<b>1</b>	<b>INTRODUCTION AND LITERATURE REVIEW</b>	<b>1</b>
	1.1 OBJECTIVE	1
	1.2 MOTIVATION	1
	1.3 BACKGROUND	1
<b>2</b>	<b>PROJECT DESCRIPTION AND GOALS</b>	<b>4</b>
<b>3</b>	<b>TECHNICAL SPECIFICATIONS</b>	<b>6</b>
<b>4</b>	<b>DESIGN APPROACH AND DETAILS</b>	<b>10</b>
	4.1 MATERIALS & METHODS	10
	4.2 CODES & STANDARDS	34
	4.3 CONSTRAINTS, ALTERNATIVES & TRADE-OFF	36
<b>5</b>	<b>SCHEDULE, TASKS AND MILESTONES</b>	<b>37</b>
<b>6</b>	<b>PROJECT DEMONSTRATION</b>	<b>38</b>
<b>7</b>	<b>COST ANALYSIS</b>	<b>45</b>
<b>8</b>	<b>SUMMARY</b>	<b>46</b>
<b>9</b>	<b>REFERENCES</b>	<b>47</b>
	<b>Annexure I – PROGRAM CODES</b>	<b>49</b>
	<b>Annexure II- VITeCON’19 CONFERENCE</b>	<b>65</b>
	<b>Annexure III- Reboot’19 HACKATHON</b>	<b>67</b>

## LIST OF FIGURES

Figure No.	Title	Page No.
Fig 1.	Overall System Architecture	5
Fig 2.	Raspberry Pi 3	10
Fig 3.	Raspberry Pi UART	11
Fig 4.	DSB18B20	11
Fig 5.	Hardware interfacing of DSB18B20 with Raspberry Pi	12
Fig 6.	Blood Pressure Sensor	13
Fig 7.	MedXBot Agent Name	16
Fig 8.	Intents inside an Agent	17
Fig 9.	Training inside an intent	17
Fig 10.	Response inside intent	18
Fig 11.	Context inside intents	18
Fig 12.	MedXBot Questionnaire	19-20
Fig 13.	Scatter plot of dataset	23
Fig 14.	Comparison of ML algorithms	34
Fig 15.	IoT based Hardware System	38
Fig 16.	Real time data getting stored in databases	39
Fig 17.	Dialogflow console	40
Fig 18.	WebApp	40-41
Fig 19.	Signing in the WebApp	42
Fig 20.	Patient talking to MeXBot	42
Fig 21.	Virtual Nurse collecting all the data	43
Fig 22.	Data stored at Firebase	43-44
Fig 23.	Report generated and sent to the doctor	44

## LIST OF TABLES

<b>Table No.</b>	<b>Title</b>	<b>Page No.</b>
Table 1.	Pin details of DSB18B20	11
Table 2.	Classification of blood pressure in adults	12
Table 3.	Pin details of blood pressure sensor	13
Table 4.	Dataset description (UCI Heart Disease Dataset)	22
Table 5.	Budget of our model	45



## LIST OF ABBREVIATIONS

CVD	Cardio Vascular Disease
M2M	Machine-to-Machine
BP	Blood Pressure
ECG	Electrocardiogram
BMI	Additive White Gaussian Noise
ML	Machine Learning
UCI	University of California Irvine
WHO	World Health Organisation
CT	Computed Tomography
PPG	Photo Plethysmography
SVM	Support Vector Machine
CBR	Case Based Reasoning
AI	Artificial Intelligence
KNN	K-Nearest Neighbour
GPIO	General Purpose Input Output
SQL	Structured Query Language
HTML	HyperText Markup Language
CSS	Cascading Stylesheet
W3C	World Wide Web Consortium
JSON	Java Script Object Notation
NLP	Natural Language Processing
USB	Universal Serial Bus
HDMI	High Definition Multimedia Interface
ASCII	American Standard Code for Information Interchange

## SYMBOLS AND NOTATIONS

$F_s$	Sampling frequency
$N$	Total duration of signal
DSB18B20	Temperature sensor (digital)
AD8232	ECG sensor

# **1. INTRODUCTION & LITERATURE REVIEW**

## **1.1 OBJECTIVE**

Cardio Vascular Diseases (CVD) led to almost one-third of deaths in the world which makes it most critical topic to study and provide efficient solutions. Considering that, this paper proposes a system to remotely monitor health of CVD patients using machine-to-machine (M2M) technology.

## **1.2 MOTIVATION**

Millions of people die all over the world because of CVD (Cardio Vascular Disease) and every year the number is increasing so we need to find a cure for this. To mention some Indian statistics, in 2009, death due to CVD is 54% as compared to 1997 having deaths of 30% [1]. As per World Health Organization (WHO) report, a major share of 24% due to CVDs out of 53% of the total deaths in India in 2011 is analysed [2][3]. Thus, as per above statistics, it motivated us to find a solution to the problem.

## **1.3 BACKGROUND**

In the last decade, diabetes is also major disease suffered by many people. There are two types of diabetes namely type1 and type2. The type2 diabetes people are mostly affected by CVD [4]. Hence predicting the blood glucose level of the patient is also a major task to analyse CVD [5]. Hence people came with various methods to cure heart diseases such as chest X-Ray, computed tomography (CT), Blood Test etc. but these tests were costly. Therefore, a convenient low cost non-invasive method was developed known as Photo Plethysmography (PPG) [5]. PPG signals contain information like heart rate, blood pressure, blood saturation blood glucose. The PPG signal is measured using index finger of the patient which give information on heart rate and blood pressure.

There is wearable device manufactured which plots these waveforms and helps in prediction of which class of PPG signal is formed [6]. Generally, class1 PPG signal is seen in healthy people while class4 is found in old people and people who are likely to

suffer from CVD. After taking the PPG signal or waveform it is send to computer and heart rate is extracted by the following equation (1) [7]: -

$$\text{Heart Rate} = \frac{\text{No.of peaks} * Fs}{N * 60} \quad (1)$$

As mentioned in equation (1), N is total duration of signal, Fs is sampling frequency. Normally, heart rate is 60-100 bpm. Mostly blood sugar measurements are invasive thus causing pain and a chance of increasing infectious disease but this method is dependent on clinical parameters and computation of parameters creating a fixed length vector which is guaranteed by SVM (Support Vector Machine), thus finding blood glucose information [8]. Hence, PPG signal provides another diagnostic tool to analyse and research about cardiovascular system.

In another research [9], they have used CBR (Case Based Reasoning) technique and nearest neighbour method to provide medication. This system monitors patient's vital signs (heart rate, blood pressure and rhythm) then continuously report it to M2M server through the network. M2M server collects the data and analyses to recommend medication to the patient. This medication is verified by the doctor and will be notified to patient's mobile. Thus, he can consume the recommended and verified drug in his pill box. This work uses AI (Artificial Intelligence) approach with CBR to find out disease type. This technique has four steps [10]-

- 1) Retrieve- identify the issue and search out related problems from database
- 2) Reuse- information from similar previous tasks is used
- 3) Revise- information is evaluated and revised
- 4) Retain- combine and obtain new solution and save for the future use

Cardio Vascular Diseases (CVD) is one of the deadliest diseases and is a bigger problem in developing countries due to scarcity of physicians especially in rural areas. This problem can be solved to some extent with the introduction of telemedicine system based on Machine-to-Machine (M2M) technology [11]. M2M can be described as telecommunication system which use electronic devices restricting human intervention. It is designed using KNN (K-Nearest Neighbour) algorithm, which shows quite good accuracy for this application. Here they have taken the training data from

Harapan Kita hospital, Jakarta. Parameters used for this study are age, gender, systolic/diastolic pressures, pulse rate, ECG diagnosis and some more symptoms. The architectural design and implementation of the system consists of three sites [11]-

- 1) Patient site - At this site the heart condition of patient is measured with ECG sensor and Sphygmomanometer for the nurse to capture vital indications. She also orally interviews the patient to gather more data. Finally, M2M gateway collects the data and carry out KNN analysis then send the data to the server.
- 2) Server site – Acquires data from patient then stores data to web server and it is accountable for deploying of website and mobile application.
- 3) Doctor site- the doctor will verify resultant report, posting his advice which will be recorded and sent to the server and then to nurse's mobile application.

This has been implemented and trailed in rural areas the results of trials are measured in aspects of accuracy of prediction, KNN time for processing time and transfer duration. Overall prediction accuracy is about 74.67%. In this way, the research focusses to build a tele-medicine system dependent upon M2M technology for CVD patients especially in rural areas [12].

## 2. PROJECT DESCRIPTION AND GOALS

Let us discuss our system design which is modified version of previous proposed designs. It is modified in the sense that it overcomes the short-comings of previous design by-

- 1) Increasing the prediction accuracy from 74.67% [11] to 98.59% of the ML algorithm implemented in our system, thus making more reliable;
- 2) It also makes the system more interactive by providing it a user-friendly front-end design.

Our system starts by gathering various sensor data in Raspberry Pi. The sensors include temperature sensor (DSB18B20), blood pressure sensor (sphygmomanometer), heart rate sensor (pulse sensor) and ECG sensor (AD8232) which are interfaced with Raspberry Pi GPIO pins and coded using python. All the gathered data are sent to 'Thingspeak' server which is a NoSQL database server.

After all hardware data are collected and stored in a database server, the chatbot system is initiated (named as 'MedX' bot) which takes a general medical interview of the patient covering various aspects of his/her health. The chatbot is created using 'Dialogflow' system and the data gathered by the system is stored in 'Google Firebase' server which is again a NoSQL database server.

The data gathered till now are sent to a web page which acts as a front-end in our system. This page shows all the gathered data in various text fields and contains three buttons which are: -

- a) 'Train' button- It has a functionality of training the Machine Learning (ML) algorithm implemented using 'TensorFlow' onto the web page. After this, we are prompted to another page indicating a message that 'your data has been trained into the system'.
- b) 'Test and Generate' button- This button when pressed, implements a testing of the data over the trained ML algorithm using 'TensorFlow'. After this, we are prompted to another page which shows auto-diagnosed report with all necessary prescription of medicines and other medication tasks to be done by the patient. This page contains a

‘send’ button which when pressed, sends the report to the patient as well as the prescribed doctor for reviewing purposes.

- c) ‘Reset’ button- This button when pressed, clears all the data stored in various text fields.

Finally, doctor reviews the auto-diagnosed report (which is generated by the system trained with ML algorithm). If he/she feels to provide some feedback, then he/she can reply back to the system, this reply is sent to patient for his ease in using the prescribed medication. The following Figure 1 shows the complete system design of our model.

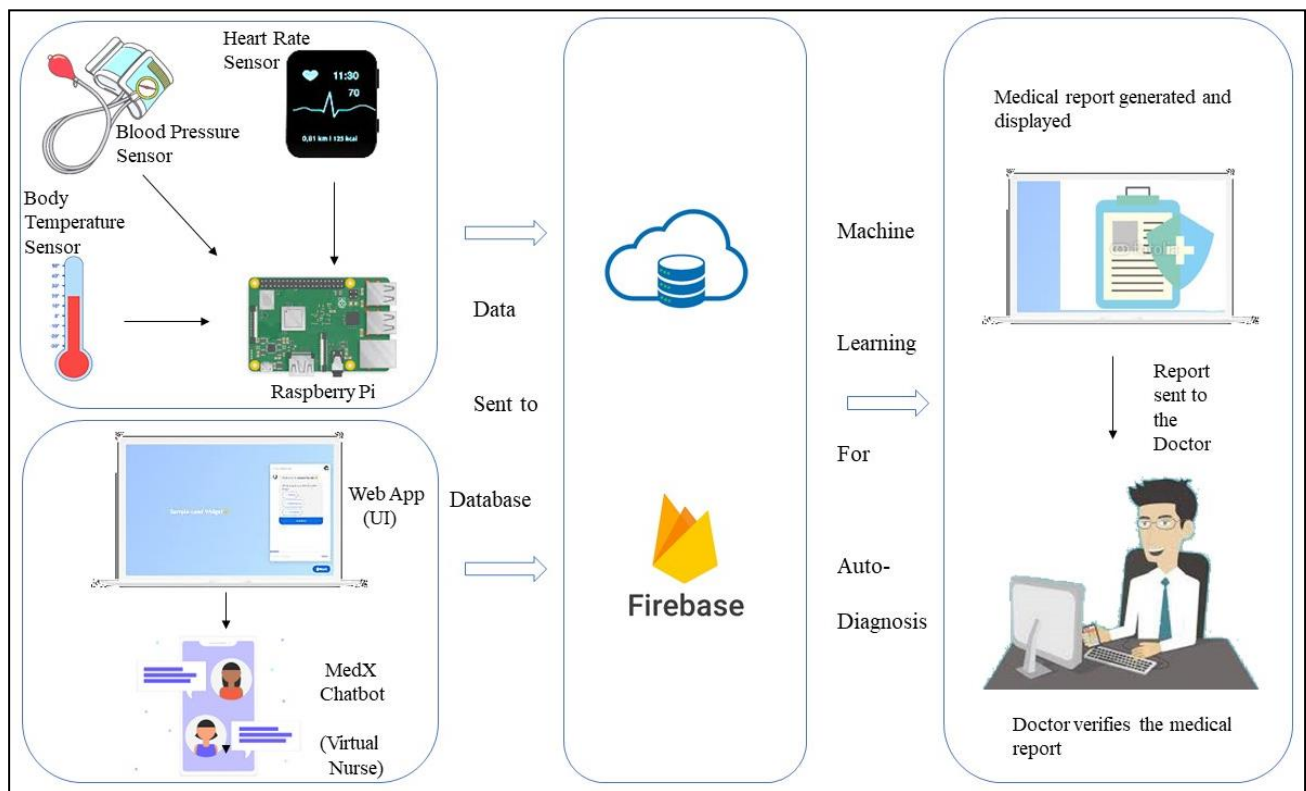


Fig 1. Overall System Architecture

### 3. TECHNICAL SPECIFICATIONS

#### Raspberry Pi Model 3b

- Manufactured by- Adafruit Industries
- Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board Computer running at 1.2GHz
- 1GB RAM
- BCM43143 WiFi on board
- Bluetooth Low Energy (BLE) on board
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source (now supports up to 2.4 Amps)
- Expected to have the same form factor has the Pi 2 Model B, however the LEDs will change position

#### Blood Pressure Sensor

- Manufactured by- Sunrom electronics
- Working Voltage- +5V, 200mA regulated
- Output Format- Serial Data at 9600 baud rate (8 bits data, No parity, and 1 stop bit). Outputs three parameters in ASCII.
- Sensing unit wire length is 2 meters
- Product does not require battery for operation. It can be powered from external PCB.

#### Digital Temperature Sensor

- Model name- DSB18B20
- Model type- Digital
- Programmable Digital Temperature Sensor



- Communicates using 1-Wire method
- Operating voltage- 3V to 5V
- Temperature Range- -55°C to +125°C
- Accuracy-  $\pm 0.5^{\circ}\text{C}$
- Output Resolution- 9-bit to 12-bit (programmable)
- Unique 64-bit address enables multiplexing
- Conversion time- 750ms at 12-bit
- Programmable alarm options
- Available as To-92, SOP and even as a waterproof sensor

### **Thingspeak Database Server [Hardware]**

- Designed by- MathWorks
- Repository- <https://github.com/iobridge/thingspeak>
- Written in- Ruby
- Operating System- Cross platform
- Type- API
- License- GPL version 3
- Website- <http://thingspeak.com/>

### **Dialogflow [chatbot]**

- Also called as API.AI
- Parent- Google
- Application- Natural Language Processing (NLP) user interface
- Products- Dialogflow, Assistant
- It supports more than 20+ platforms from Google home to Twitter
- It supports all the devices from wearables, to phones to devices.
- It supports more than 14+ languages worldwide & more support is coming.
- Website- <https://dialogflow.com/>

## **Firestore Database Server [Final]**

- Parent- Google
- Founder- James Tamplin, Andrew Lee
- Application- Mobile Application Development, Backend service
- Products- A/B Testing, App Indexing, Analytics, Authentication, Cloud Firestore, Cloud Functions, Cloud Messaging, Cloud Storage, Crashlytics, Dynamic Links, Hosting, In-App Messaging, ML Kit, Performance Monitoring, Predictions, Realtime Database, Remote Config, Test Lab
- Website- <https://firebase.google.com/>

## **HTML [UI development]**

- Also called as HyperText Markup Language
- It is open format type language with latest release 5.2
- Developed by- W3C (World Wide Web Consortium) & WHATWG
- File extension- .html or .htm
- Standards- ISO/IEC 15445, W3C HTML latest recommendation
- Website- <https://www.w3.org/html/>, <https://whatwg.org/>

## **CSS [UI development]**

- Also called as Cascading Style Sheets
- It is style sheet type formatting language
- Developed by- W3C
- File extension- .css
- Standards- <http://www.w3.org/TR/1999/REC-CSS1-19990111>
- Used for styling the HTML pages

### **JavaScript [UI development (scripting)]**

- It is scripting language to connect backend to HTML page
- It handles JSON object which contain data.
- Paradigm- event-driven, functional, imperative, object-oriented
- Developed by- Netscape Communications Corporation, Mozilla Foundation, Ecma International
- File extension- .js
- Website- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

### **easyPHP Dev Server & Web Server [send final report to doctor]**

- It installs a complete and ready-to-use development environment.
- It is portable, modular, fully configurable and easy to update and extend.
- It supports PHP, Apache, MySQL, Nginx, PhpMyAdmin, Xdebug, PostgreSQL, MongoDB, Python, Ruby.
- Webserver turns your computer into a ready-to-use personal web hosting server.

Hence, we have mentioned technical specifications of each and every device and technology used in our system.

## 4. DESIGN APPROACH & DETAILS

### 4.1 MATERIALS & METHODS

In this part, we briefly discuss about materials & methods used in our system. Firstly, let us discuss about each and every sensor which we have used here and then the ML algorithms used.

#### 4.1.1. IoT-based Machine-to-Machine system

##### 4.1.1.1 Raspberry Pi Model 3b

Raspberry Pi is an ARM based credit card sized SBC (Single Board Computer) created by Raspberry Pi Foundation. Raspberry Pi runs Debian based GNU/Linux operating system Raspbian and ports of many other OSes exist for this SBC.

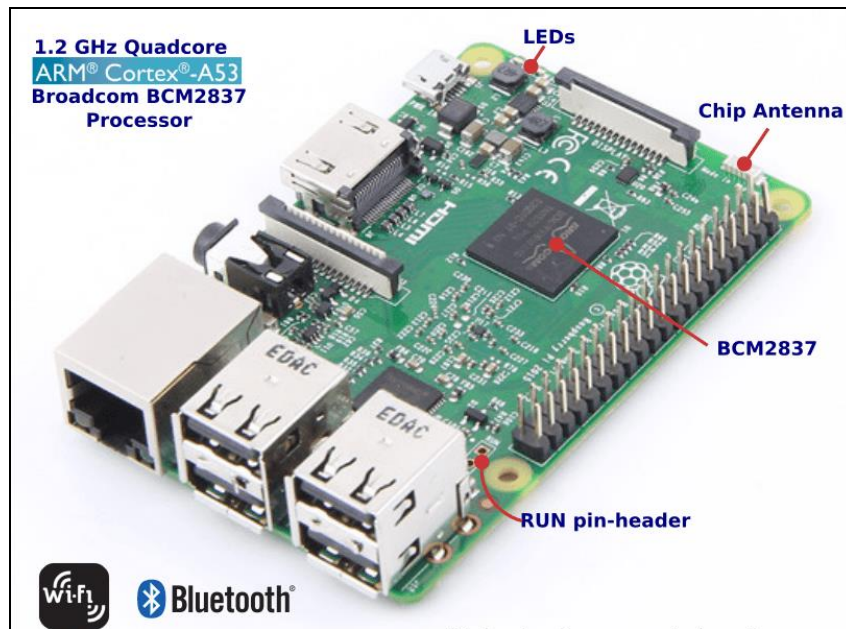


Fig 2. Raspberry Pi 3 [13]

The organisation behind the Raspberry Pi consists of two arms. The first two models were developed by the Raspberry Pi Foundation. After the Pi Model B was released, the Foundation set up Raspberry Pi Trading, with Eben Upton as CEO, to develop the third model, the B+. Raspberry Pi Trading is responsible for developing the technology while the Foundation is an educational charity to promote the teaching of basic computer science in schools and in developing countries.

The following figure shows the UART diagram of Raspberry Pi:-

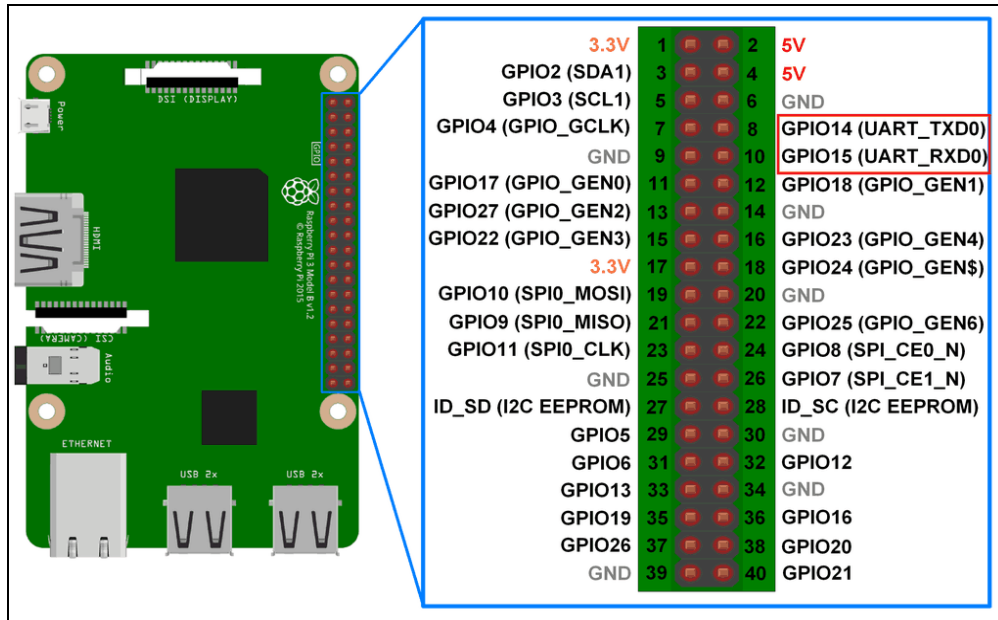


Fig 3. Raspberry Pi UART [13]

#### 4.1.1.2 Temperature sensor

We have used DS18B20 digital temperature sensor. It is the size of a transistor and transmits data to the microcontroller through only one wire but it needs three wires for operation i.e. VCC, GND and Data wires. They are made of silicon and have Analog to Digital convertor (ADC) memory to temporary store the data. It can sense temperature in the range of -55°C to 125°C.

Table 1. Pin details of DSB18B20

Pin	Description
GND	Connect to ground of circuit
VCC	To power up the sensor, 3.3 V or 5 V
Data	Output the temperature value which can be read using 1-wire protocol

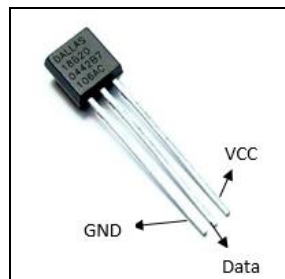


Fig 4. DSB18B20 [14]

(R1 is 4.7K Ohm or 10K Ohm)

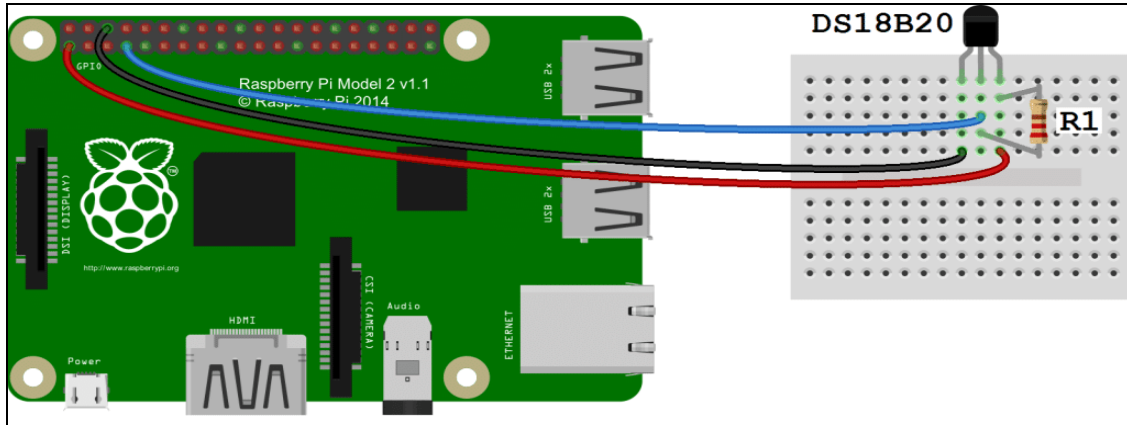


Fig 5. Hardware interfacing of DSB18B20 with Raspberry Pi [15]

#### 4.1.1.3 Blood Pressure Sensor

Blood pressure is the pressure of the blood in the arteries as it is pumped around the body by the heart. When your heart beats, it contracts and pushes blood through the arteries to the rest of your body. This force creates pressure on the arteries. Blood pressure is recorded as two numbers—the systolic pressure (as the heart beats) over the diastolic pressure (as the heart relaxes between beats). The unit which measures this is called Sphygmomanometer.

Monitoring blood pressure at home is important for many people, especially if you have high blood pressure. Blood pressure does not stay the same all the time. It changes to meet your body's needs. It is affected by various factors including body position, breathing or emotional state, exercise and sleep. It is best to measure blood pressure when you are relaxed and sitting or lying down.

Table 2. Classification of blood pressure in adults

Cardiac Condition	Systolic (mm Hg)	Diastolic (mm Hg)
Hypotension	< 90	< 60
Desired	90 - 119	60 - 79
Prehypertension	120 - 139	80 - 89
Stage 1 Hypertension	140 - 159	90 - 99
Stage 2 Hypertension	160 - 179	100 - 109
Hypertensive crisis	$\geq 180$	$\geq 110$

The features of this sensor are as follows: -

- Intelligent automatic compression and decompression
- Easy to operate, switching button to start measuring
- 60 store groups memory measurements
- Can read single or all measures
- 3 minutes automatic power saving device
- Intelligent device debugging, automatic power to detect
- Local tests for: wrist circumference as 135-195mm
- Large-scale digital liquid crystal display screen, Easy to Read Display
- Fully Automatic, Clinical Accuracy, High-accuracy
- Power by External +5V DC
- Serial output data for external circuit processing or display.

Table 3. Pin details of blood pressure sensor

Pin	Description
TX-OUT	Transmit output. Output serial data of 3V logic level, usually connected to RXD pin of microcontrollers.
+5V	Regulated 5V supply input
GND	Board Common Ground



Fig 6. Blood Pressure Sensor

Blood Pressure & Pulse reading are shown on display with serial out for external projects of embedded circuit processing and display. Shows Systolic, Diastolic and Pulse Readings. Compact design fits over your wrist like a watch. Easy to use wrist style eliminates pumping.

Each reading consists of 15 bytes at 9600 baud rate. The reading packet's last byte is always enter key character (0x0A in hex and 10 in decimal) so you can view each reading on new line. Also, this character can be used to sync in microcontrollers after reach readings. The output reading is 8 bit value in ASCII format fixed digits, from 000 to 255. Typical reading will be like below where the three values separated by comma and space- *Systolic, Diastolic, Pulse*.

#### **4.1.2. NLP-based chatbot system**

##### *4.1.2.1 Chatbot implementation using Dialogflow*

Artificial Intelligence (AI) is a field of study in intelligent agents and has been in existence since the Turing machine and much beyond. There have been great advancements in AI over the past few decades. One of the major areas of AI is concerned with the interaction between computers and human (natural) languages. Most natural language processing (NLP) systems were based on complex sets of hand-written rules up to the 1980s. Starting in the late 1980s, however, there was a revolution in NLP with introduction of machine learning (ML) algorithms for language processing. A chatbot (talkbot, chatterbot, bot or interactive agent) is a computer program which conducts a conversation via auditory or textual methods. Such programs are often designed to convincingly simulate how a human would behave as a conversational partner, thereby passing the Turing test. Most chatbots are either accessed via virtual assistants such as Google Assistant, Amazon Alexa, Apple Siri or via individual organizations' apps and websites. There are various platforms available to create a chatbot such as msg.ai, wit.ai, Dialogflow, Microsoft Language Understanding Intelligent Service (LUIS) etc. Dialogflow was chosen for this project because, the platform is very intuitive, the ease of creation of the chatbot was better when compared to other platforms, provides support for large set of languages and it provides



integrations to various third-party apps. But most of the platforms follow same methods and terminologies.

Dialogflow (Previously known as API.AI) is where the magic happens. It works on *natural language processing* and backed by *Machine Learning*. At Dialogflow the whole ‘conversation’ take place. Dialogflow is backed by Google and runs on Google infrastructure, which means you can scale to millions of users.

There are already chatbots available in areas like flight booking, hotel management, food ordering services, home automation etc. Healthcare is one of the areas where there is a lot of research and opportunities with respect to utilizing chatbots. Most of the chatbots available in this domain deal with hospital management like scheduling an appointment, getting to know the availability of the doctors, managing prescriptions, heart rate and blood pressure monitoring, reporting symptoms and illness etc. Our chatbot is named as ‘MedXBot’ which is a virtual doctor as it interviews about patient’s current health condition and suggests medication.

Agents- Agents are best described as NLU (Natural Language Understanding) modules. These can be included in the app, product, or service and transform natural user requests into actionable data. Agent is the name of the app that we are creating. The name of the agent is very important. In our case it is ‘MedXBot’.

Intents- An intent represents a mapping between what a user says and what action should be taken. Intent interfaces have the following sections:

Training Phrases- It has set of data which are trained into the intent to predict the type of data. Fig 12 shows training phase of one intent.

Action- It performs some action over data fed by user into the chatbot as per training phase. Fig 12 shows one screenshot of it.

Response- It prints the text as acknowledgement to the conversation in that intent. Fig 13 shows one example of it.

**Contexts-** Contexts represent the current context of a user's request. This is helpful for differentiating phrases which may be vague or have different meanings depending on the user's preferences, location or the topic of conversation. Contexts are designed for passing on information from previous conversations and can be used to manage the conversation flow. Each intent can have several input and output contexts. The output contexts from an intent will act as an input context to another intent. Fig 14 shows an example of branched context.

**Entity-** Entities are powerful tools used for extracting parameter values from natural language inputs. Any important data we want to get from a user's request, will have a corresponding entity. There are three types of entities: system (defined by Dialogflow. Example: given names, address, phone numbers, color, temperature etc.), developer (defined by a developer), and user (built for each individual end-user). The Fig 12 shows one such example of an entity inside an intent. One more thing to be noted is that an intent can contain multiple entities.

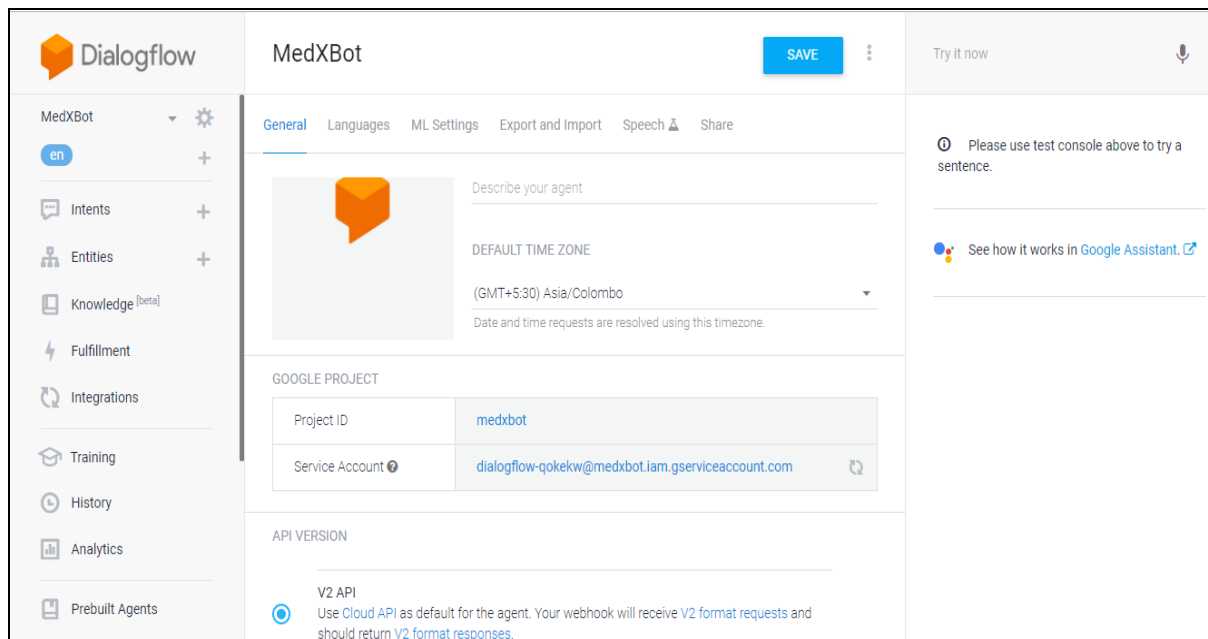


Fig 7. MedXBot Agent Name

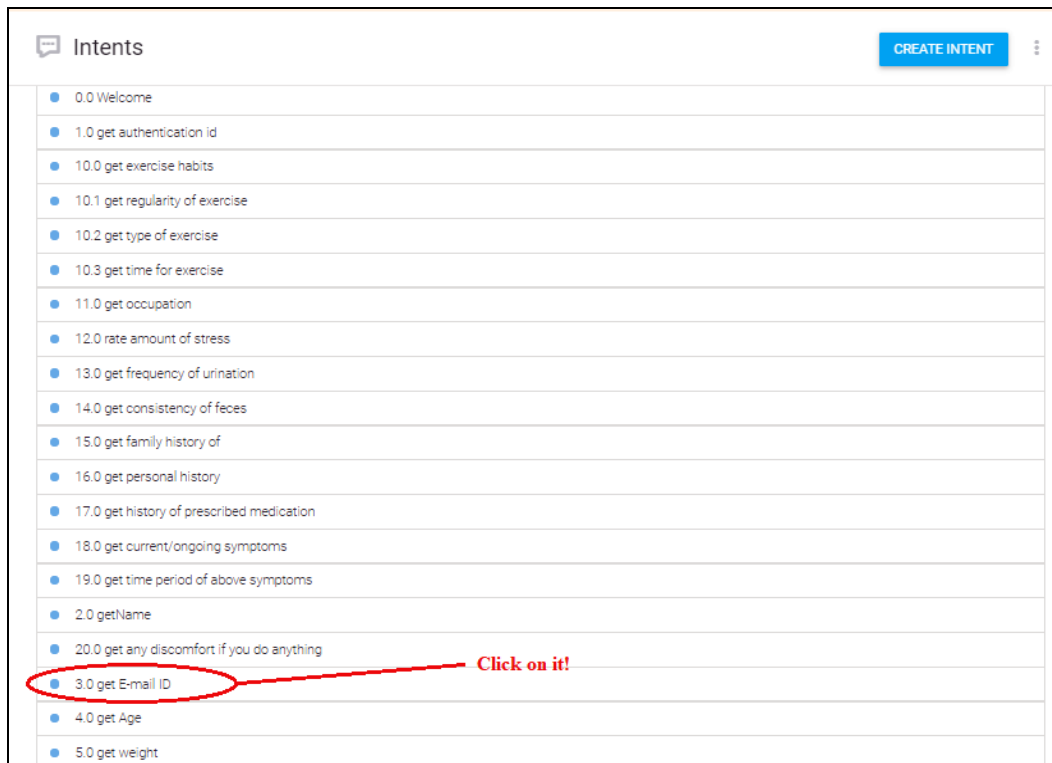


Fig 8. Intents inside an Agent

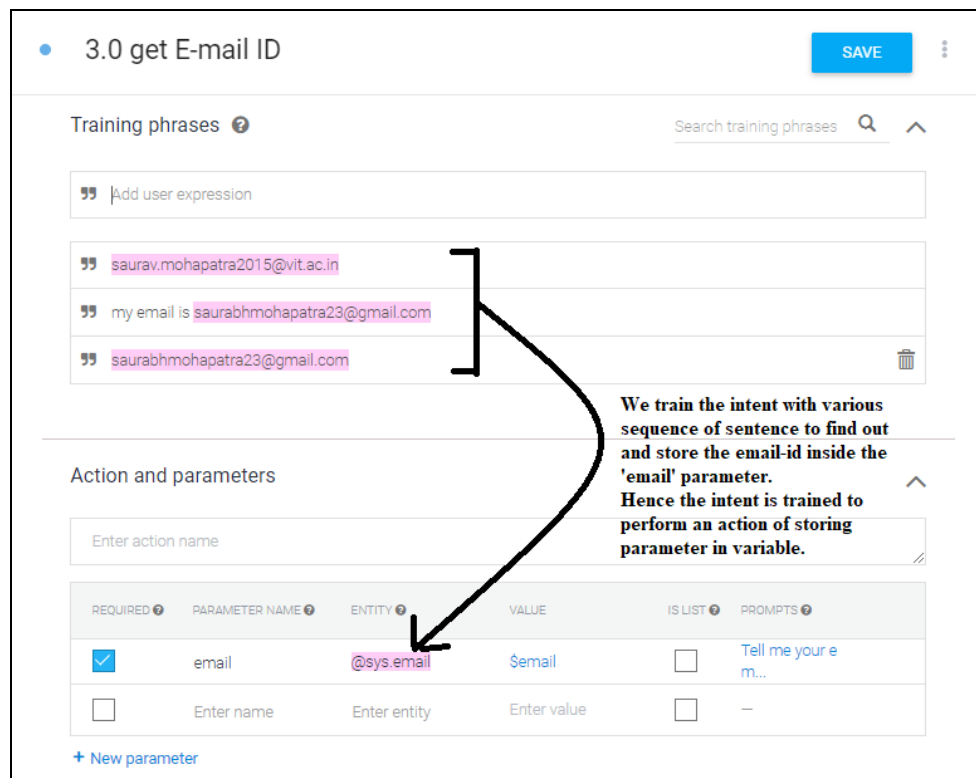


Fig 9. Training inside an intent

• 3.0 get E-mail ID SAVE

Responses ?

DEFAULT +

Text response

- I recorded your email id to be: \$email, we will proceed further by knowing your age (in years). What is your age?
- Enter a text response variant

ADD RESPONSES

☐ Set this intent as end of conversation ?

Fulfillment ?

☒ Enable webhook call for this intent

☐ Enable webhook call for slot filling

Response by the intent after the action is performed.

Enables the data gathered by the intent in its parameters to be sent to Firebase (cloud functions) or any other webhook service.

Fig 10. Response inside intent

• 7.0 get sex SAVE

Contexts ?

get\_sex Add input context

5 get\_female\_cycle 5 get\_lifestyle\_habit Add output context

Events ?

Training phrases ?

Search training phrases

Add user expression

others

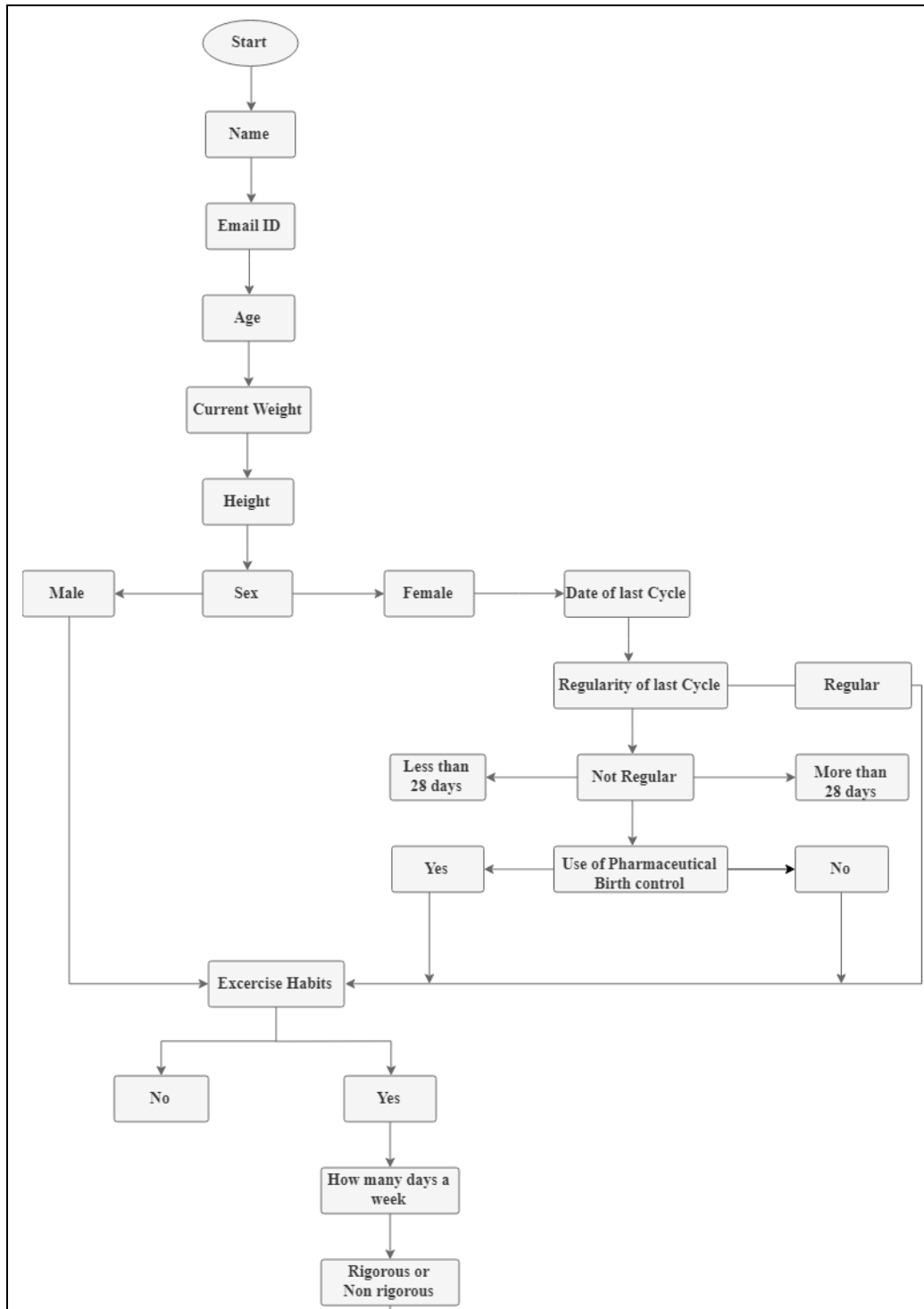
Can't specify

Female

Male

Fig 11. Context inside intents

Flowchart of MedXBot- Agents work based on decision trees and at each step of the conversation flow, it can have multiple paths to proceed and determines the best path based on the previous outcome and input.



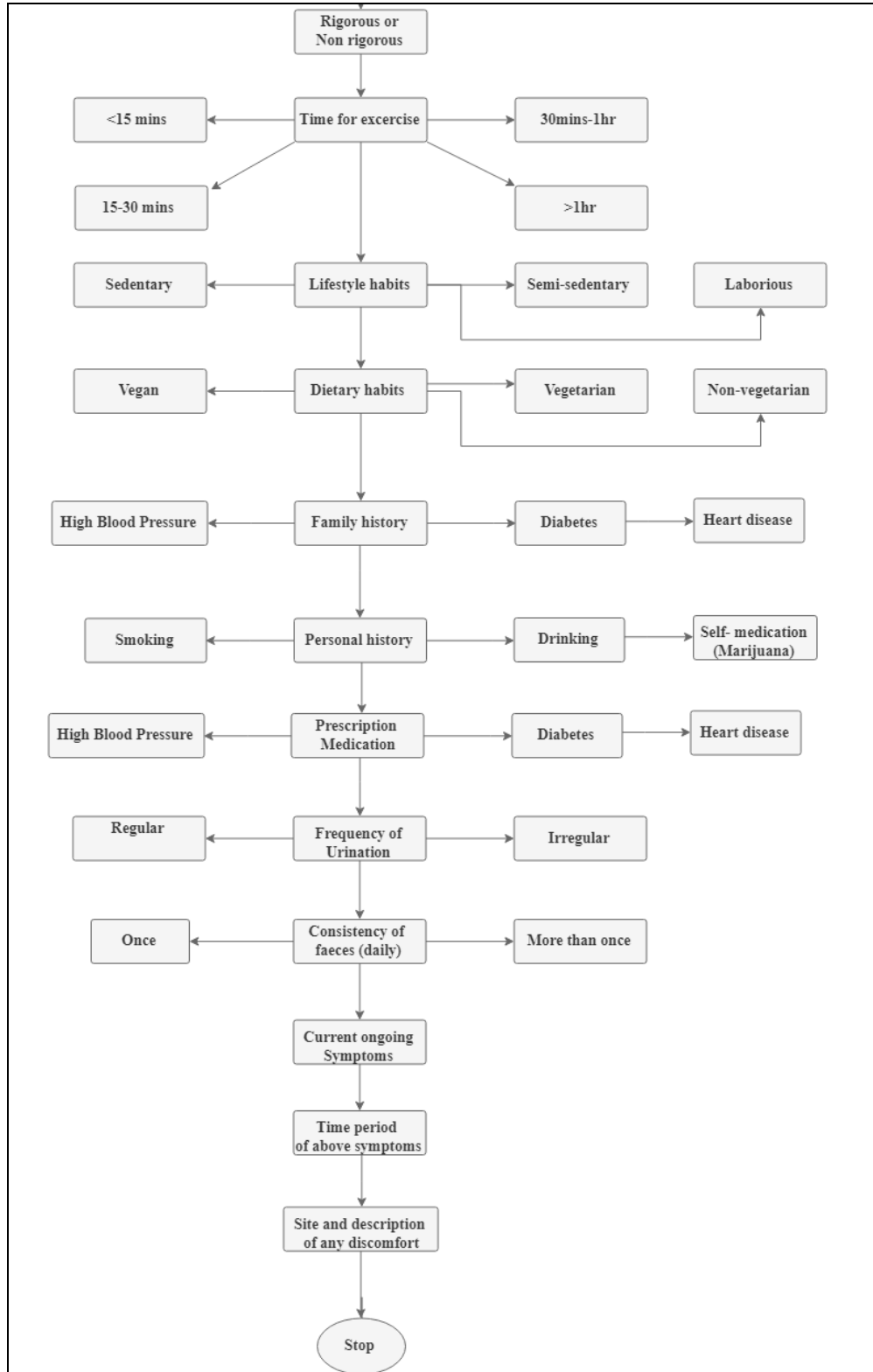


Fig 12. MedXBot Questionnaire

#### **4.1.3. ML-based auto-diagnosis system**

Classifying data is one of the most common tasks in Machine learning. Machine learning provides one of the main features for extracting knowledge from large databases from enterprises operational databases. Machine Learning in Medical Health Care is an emerging field of very high importance for providing prognosis and a deeper understanding of medical data. Most machine learning methods depend on a set of features that define the behaviour of the learning algorithm and directly or indirectly influence the performance as well as the complexity of resulting models.

Heart disease is the leading cause of death all over the world for the past 10 years. There have been several machine learning techniques used for the diagnosis of heart disease in the past. Neural Network and Logistic Regression are some of the few machine learning techniques used in the diagnosis of heart disease showing some amount of success.

We investigate different algorithms such as Neural Network, KNearest Neighbours, Naive Bayes and Logistic Regression along with hybrid techniques involving the above used algorithms for the diagnosis of heart disease.

The system has been implemented in Python platform and trained using benchmark dataset from UCI machine learning repository. The system is possibly expandable for the new dataset.

Nowadays people work on computers for hours and hours, they don't have time to take care of themselves. Due to unhealthy lifestyle, hectic schedules and consumption of junk food, the health of people is being severely affected. With the medical parameters of enough number of patients and the no patients, it is possible to predict how likely it is for a person to be suffering from a heart disease. The challenge is thus to predict which of the tests are likely to turn positive on the heart disease detection test and which ones negative.

Table 4. Dataset Description (UCI Heart Disease Dataset) [16]

Variable (parameter)	Short description	Values
Age	Age of patient	Continuous
Sex	Sex of patient (male/female)	0=Male, 1=Female
Chest pain	Three types (anginal/ non-anginal/ asymptomatic)	4 = Typical type 3 = Atypical type angina 2 = Non-angina pain 1 = Asymptotic
Bpm or Thalach	Max Heart rate achieved by patient	Continuous
(Sys, Dia) or Thes bps	Resting Blood pressure of patient (Sys- Systolic, Dia- Diastolic)	Continuous
Chole	Resting Blood sugar level or Serum Cholestrol	Continuous
Fbg	Fasting Blood sugar larger than resting Blood sugar (Yes/No)	1 $\geq$ 120 mg/dl 0 $\leq$ 120 mg/dl
Rest ECG	Resting ECG result	5 = normal 4 = having ST_T wave abnormal 3 = left ventricular hypertrophy
Exercise	Induced angina (yes/no)	1 = no 2 = yes
Old peak	ST depression induced by exercise relative to rest	Continuous
Exercise_slope or Old peak	Peak time and effort given for laborious work- Slope of the height effect ST section	3 = un sloping 2 = flat 1 = downsloping
Ca	Number of major vessel (colored by fluoroscopy)	1-4 value
Thal	Thallium heart scan	2 = Normal 4 = Fixed 5 = Reversible defect
Num	Diagnosis of heart disease	No disease & Heart disease



Comparative disease profile for heart diseases compared to no heart disease-

- Age is higher in the disease group.
- Sex - the abundance of male is higher in the disease group.
- Chest pain type - the abundance of asymptomatic is higher in the disease group.
- Resting electrocardiographic results - the abundance of normal is lower in the disease group.
- Maximum heart rate achieved is lower in the disease group.
- Exercise induced angina - the abundance of yes is higher in the disease group.
- ST depression induced by exercise relative to rest is higher in the disease group.
- Slope of peak exercise ST segment - the abundance of flat is higher in the disease group.
- Number of major vessels colored by fluoroscopy is higher in the disease group.
- Thallium heart scan-- the abundance of reversible\_defect is higher in the disease group.

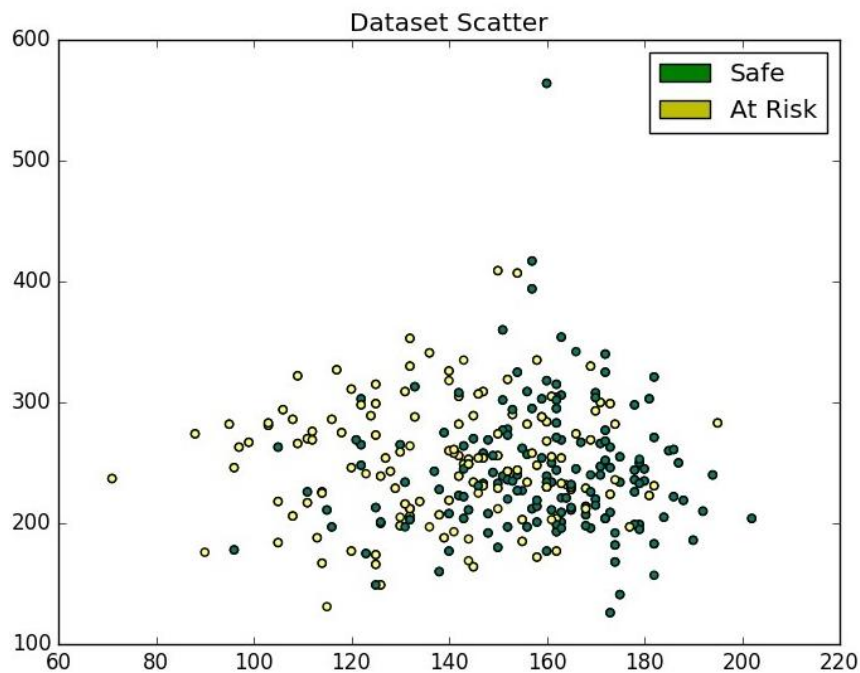


Fig 13. Scatter plot of dataset

#### 4.1.3.1 Logistic Regression

Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum entropy classification (MaxEnt) or the log linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function [17].

The implementation of logistic regression in scikit-learn can be accessed from class Logistic Regression. This implementation can fit a multiclass (one-vs-rest) logistic regression with optional L2 or L1 regularization.

As an optimization problem, binary class L2 penalized logistic regression minimizes the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^n \log(\exp(-y_i(X_i^T w + c)) + 1). \quad (2)$$

Similarly, L1 regularized logistic regression solves the following optimization problem:

$$\min_{w,c} \|w\|_1 + C \sum_{i=1}^n \log(\exp(-y_i(X_i^T w + c)) + 1). \quad (3)$$

An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is useful because it can take an input with any value from negative to positive infinity, whereas the output always takes values between zero and one and hence is interpretable as a probability. The logistic function is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \quad (4)$$

#### **4.1.3.2 Random Forest**

Random Forest is a supervised learning algorithm which builds multiple decision trees and merges them together to get a more accurate and stable prediction. The ‘forest’ it builds, is an ensemble of Decision Trees, most of the time trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. It has mixture of tree classifiers where every single classifier is made using a random vector divided independently from input vector, a tree builds unit vote for most renowned class to identify an input vector [18]. Decision Trees tend to have low bias and high variance, a process known as Bagging Trees (Bootstrap Aggregating). Random Forest aims to reduce this correlation by choosing only a subsample of the feature space at each split. Essentially aiming to make the trees more independent and thus reducing the variance.

Random forests are built by combining the predictions of several trees, each of which is trained in isolation. There are three main choices to be made when constructing a random tree. These are- a) The method for splitting the leaves. b) The type of predictor to use in each leaf. c) The method for injecting randomness into the trees.

#### **4.1.3.3 Boosted Tree**

Boosting is general way used for enhancing the accuracy of a learning algorithm [19]. More importantly, it works on the principle that a “strong” classifier (highly accurate is generated through the linear combination of a lot of “weak” classifiers (less accurate). Typically, each weak hypothesis is a simple rule which can be used to generate a predicted classification for any instance. Boosting is a general method for improving the accuracy of any given learning algorithm. The Boosted Tree algorithm, does training error and generalization error; boosting’s connection to game theory and linear programming; the relationship between boosting and logistic regression. It has extensions for multiclass classification problems; methods of incorporating human knowledge into boosting; and experimental and applied work using boosting.

#### 4.1.3.4 Support Vector Machine

SVM is type of linear classifier [20], a class of supervised learning model, which is used to separate the data into different classes by a hyper plane for N dimensional data. The model has linear classification and can perform nonlinear classification using kernel trick, thereby mapping the input features to higher dimensional feature spaces. In SVM each data item is plotted as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Classification is done by finding the hyper-plane that differentiates between the two classes very well.

Work by constructing a hyperplane that separates points between two classes. The hyperplane is determined using the maximal margin hyperplane, which is the hyperplane that is the maximum distance from the training observations. This distance is called the margin. Points that fall on one side of the hyperplane are classified as -1 and the other +1.

SVM can make some errors to avoid over-fitting. It tries to minimize the number of errors that will be made. Support vector machines classifiers are applied in many applications. They are very popular in recent research. This popularity is due to the good overall empirical performance.

#### 4.1.3.5 K-Means Naïve Bayes

K-Means Algorithm-

The K-Means algorithm clusters data by trying to separate samples in n groups of equal variances, minimizing a criterion known as the inertia or within cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields. The K-Means algorithm divides a set of  $N$  samples  $X$  into  $K$  disjoint clusters  $C$ , each described by the mean  $\mu_j$  of the samples in the cluster. The means are commonly called the cluster “centroids”; Note that they are not, in general,

points from  $X$ , although they live in the same space. The K-Means algorithm aims to choose centroids that minimise the *inertia*, or within cluster sum of squared criterion:

$$\sum_{i=0}^n \min_{\mu_j \in C} (||x_j - \mu_i||^2) \quad (5)$$

Inertia, or the within cluster sum of squares criterion, can be recognized as a measure of how internally coherent clusters are. It suffers from various drawbacks:

- Inertia makes the assumption that clusters are convex and isotropic, which is not always the case. It responds poorly to elongated clusters, or manifolds with irregular shapes.
- Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very high dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called “curse of dimensionality”). Running a dimensionality reduction algorithm such as PCA prior to K-Means clustering can alleviate this problem and speed up the computations.

KMeans is often referred to as Lloyd’s algorithm. In basic terms, the algorithm has three steps. The first step chooses the initial centroids, with the most basic method being to choose  $k$  samples from the dataset  $X$ . After initialization, K-Means consists of looping between the two other steps. The first step assigns each sample to its nearest centroid. The second step creates new centroids by taking the mean value of all of the samples assigned to each previous centroid. The difference between the old and the new centroids are computed and the algorithm repeats these last two steps until this value is less than a threshold. In other words, it repeats until the centroids do not move significantly.

Given enough time, K-Means will always converge, however this may be to a local minimum. This is highly dependent on the initialization of the centroids. As a result, the computation is often done several times, with different initializations of the centroids.

## Naive Bayes-

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes’ theorem with the “naive” assumption of independence [17] between every pair

of features. Given a class variable  $y$  and a dependent feature vector  $x_1$  through  $x_n$ , Bayes' theorem states the following relationship:

$$P(y | x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n | y)}{P(x_1, \dots, x_n)} \quad (6)$$

Using the Naïve assumption independence,

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y), \quad (7)$$

for all  $i$ , this relationship is simplified to

$$P(y | x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i | y)}{P(x_1, \dots, x_n)} \quad (8)$$

Since  $P(x_1, \dots, x_n)$  is constant given the input, we can use the following classification rule:

$$\begin{aligned} P(y | x_1, \dots, x_n) &\propto P(y) \prod_{i=1}^n P(x_i | y) \\ &\Downarrow \\ \hat{y} &= \arg \max_y P(y) \prod_{i=1}^n P(x_i | y), \end{aligned} \quad (9)$$

And we can use Maximum A Posteriori (MAP) estimation to estimate  $P(y)$  and  $P(x_i | y)$ ; The former is then the relative frequency of class  $y$  in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of  $P(x_i | y)$ .

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters.

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional

distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

We initially implemented K Means individually. Then we worked on hybrid by combining K Means with naïve Bayes. K Means is used to group together similar data. Then naïve Bayes was implemented on each cluster and model was made. For each new test case, it was first determined to which cluster it belongs. Then the naïve Bayes model for that particular cluster was used to make prediction for the given test case. K Means was used with the hope that grouping together similar data will help in increasing accuracy of naïve Bayes algorithm. Here we had a trade-off between time of computation and accuracy but additional gained accuracy was preferred. For this algorithm, we first discretized the data as naïve Bayes requires data which is in discrete form. We couldn't implement Gaussian naïve Bayes as data distribution was not Gaussian and still using the algorithm would have resulted in poor accuracy. We had two choices for data discretization, equal width discretization and equal frequency discretization. Equal width discretization resulted in better performance.

#### **4.1.3.6 Fuzzy K-Nearest Neighbour**

K-Nearest Neighbours-

In pattern recognition, the K-Nearest Neighbours algorithm (or K-NN for short) is a nonparametric method used for classification and regression. In both cases, the input consists of the  $k$  closest training examples in the feature space. K-NN is a type of instance based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The K-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, it can be useful to assign weight to the contributions of the neighbours, so that the nearer neighbours contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbour a weight of  $1/d$ , where  $d$  is the distance to the neighbour.

The neighbours are taken from a set of objects for which the class (for K-NN classification) or the object property value (for K-NN regression) is known. This can

be thought of as the training set for the algorithm, though no explicit training step is required.

A shortcoming of the K-NN algorithm is that it is sensitive to the local structure of the data. The algorithm is not to be confused with K-Means, another popular machine learning technique.

The basis of the Fuzzy K-Nearest Neighbour algorithm is to assign membership as a function of the vector's distance from its K-Nearest Neighbours and those neighbours' memberships in the possible classes. The fuzzy algorithm is similar to the crisp version in the sense that it must also search the labelled sample set for the K-Nearest Neighbours. Beyond obtaining these K samples, the procedures differ considerably.

While the fuzzy K-Nearest Neighbour procedure is also a classification algorithm the form of its results differs from the crisp version. The fuzzy K-Nearest Neighbour algorithm assigns class membership to a sample vector rather than assigning the vector to a particular class. The advantage is that no arbitrary assignments are made by the algorithm. In addition, the vector's membership values should provide a level of assurance to accompany the resultant classification.

#### **4.1.3.7 Neural Networks**

Machine Learning implements feed-forward artificial neural networks or, more particularly, multi-layer perceptrons (MLP), the most commonly used type of neural networks. MLP consists of the input layer, output layer, and one or more hidden layers. Each layer of MLP includes one or more neurons directionally linked with the neurons from the previous and the next layer.

All the neurons in MLP are similar. Each of them has several input links and several output links. The values retrieved from the previous layer are summed up with certain weights, individual for each neuron, plus the bias term. The sum is transformed using the activation function  $f$  that may be also different for different neurons.

In other words, given the outputs  $x_j$  of the layer  $n$ , the outputs  $y_i$  of the layer  $n+1$  are computed as:



$$u_i = \sum_j (w_{i,j}^{n+1} * x_j) + w_{i,bias}^{n+1} \quad (10)$$

$$y_i = f(u_i) \quad (11)$$

Different activation functions may be used. Machine Learning implements three standard functions:

- Identity function:  $f(x) = x$
- Symmetrical sigmoid:  $f(x) = \beta * (1 - e^{-\alpha x}) / (1 + e^{-\alpha x})$ , which is the default choice for MLP.
- Gaussian function:  $f(x) = \beta e^{-\alpha x * x}$ , which is not completely supported at the moment.

In Machine Learning, all the neurons have the same activation functions, with the same free parameters ( $\alpha, \beta$ ) that are specified by user and are not altered by the training algorithms.

To compute the network, we need to know all the weights  $w_{i,j}^{n+1}$ . The weights are computed by the training algorithm. The algorithm takes a training set, multiple input vectors with the corresponding output vectors, and iteratively adjusts the weights to enable the network to give the desired response to the provided input vectors.

The larger the network size is, the more the potential network flexibility is. The error on the training set could be made arbitrarily small. But at the same time the learned network also “learns” the noise present in the training set, so the error on the test set usually starts increasing after the network size reaches a limit. Besides, the larger networks are trained much longer than the smaller ones, so it is reasonable to pre-process the data and train a smaller network on only essential features.

The NN uses forward propagation to compute  $(h\theta(x^i))_k$ , the activation (output value) of the  $k$ th output unit and  $\theta$  represents the weights. The code works for any number of input units, hidden units and outputs units. The cost function is formed for neural networks with regularization, which is given by,

$$Cost(\theta) = \frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K [-y_k^{(i)} \ln((h_{\theta}(x^{(i)}))_k) - (1 - y_k^{(i)}) \ln(1 - (h_{\theta}(x^{(i)}))_k)] + \frac{\lambda}{2m} \left[ \sum_{i=1}^m (\theta_i)^2 \right] \quad (12)$$

Then the gradient (also called partial derivative) of the cost function is calculated with respect to all weights in the neural network. The gradient helps in optimizing the weights in order to minimize the value of the cost function.

Another MLP feature is an inability to handle categorical data as is. However, there is a workaround. If a certain feature in the input or output (in case of  $n$ -class classifier for  $n > 2$ ) layer is categorical and can take  $M > 2$  different values, it makes sense to represent it as a binary tuple of  $M$  elements, where the  $i$ -th element is 1 if and only if the feature is equal to the  $i$ -th value out of  $M$  possible. It increases the size of the input/output layer but speeds up the training algorithm convergence and at the same time enables “fuzzy” values of such variables, that is, a tuple of probabilities instead of a fixed value. Machine Learning implements two algorithms for training MLPs. The first algorithm is a classical random sequential back-propagation algorithm. The second (default) one is a batch RPROP algorithm.

### **Overfitting and Regularization**

Overfitting is a major problem in neural networks. This is especially true in modern networks, which often have very large numbers of weights and biases. To train effectively, we need a way of detecting when overfitting is going on, so we don't over train. And we'd like to have techniques for reducing the effects of overfitting.

One method of combating overfitting is called regularization. Regularization modifies the objective function that we minimize by adding additional terms that penalize large weights. In other words, we change the objective function so that it becomes  $\text{Error} + \lambda f(\theta)$ , where  $f(\theta)$  grows larger as the components of  $\theta$  grow larger and  $\lambda$  is the regularization strength (a hyper parameter for the learning algorithm). The value we choose for  $\lambda$  determines how much we want to protect against overfitting. A  $\lambda=0$  implies that we do not take any measures against the possibility of overfitting. If  $\lambda$  is too large, then our model will prioritize keeping  $\theta$  as small as possible over trying to find the parameter values that perform well on our training set. As a result, choosing  $\lambda$  is a very important task and can require some trial and error.

## **Cross Validation**

Cross validation, sometimes called rotation estimation, is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (testing dataset). The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the validation dataset), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem), etc.

In k-fold cross validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining  $k - 1$  subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling (see below) is that all observations are used for both training and validation, and each observation is used for validation exactly once. 10-fold cross validation is commonly used but in general k remains an unfixed parameter. In stratified k-fold cross validation, the folds are selected so that the mean response value is approximately equal in all the folds. In the case of a dichotomous classification, this means that each fold contains roughly the same proportions of the two types of class labels.

## **Result of comparison of ML algorithms**

Finally, after the whole implementation is done, we compare the 4 conventional ML algorithms with 3 of our proposed ML algorithms as described above to find out the best. From a set of 14 variables, the most important to predict heart failure whether or not there is an occurrence of asymptomatic chest pain and many more.

Based on type of heart disease, medication will be provided. Let us tabulate the

comparison of accuracy of each of ML algorithms implemented. From Fig 17, we conclude that Neural Networks (NN) is giving highest accuracy of 98.59 % making it the best possible choice for implementing it in our system.

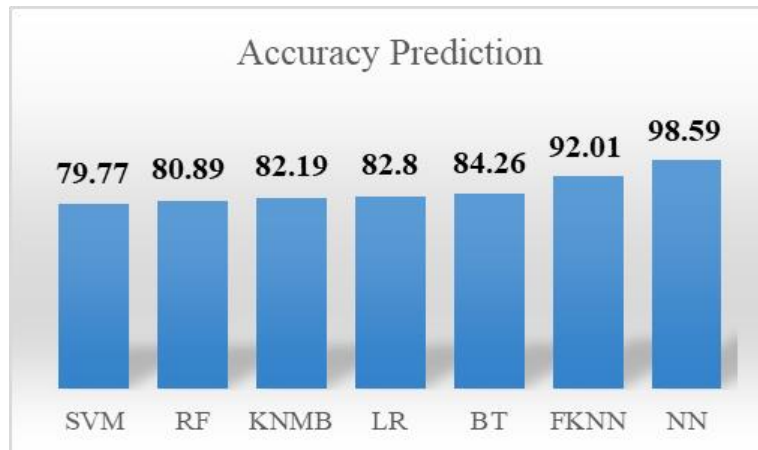


Fig 14. Comparison of ML algorithms

The reason is why Neural Networks is the best is because as we train the network with more and more samples it updates its weights continuously to store them. Thus, next time we feed any test sample; it predicts with highest accuracy possible.

## 4.2 CODES & STANDARDS

Use Wi-Fi standard (IEEE 802.11n) for connecting Raspberry P model 3 to internet. IEEE 802.11n is a wireless-networking standard that uses multiple antennas to increase data rates. Wi-Fi Alliance have also labelled the technology for the standard as Wi-Fi 4.

The temperature sensor (DSB18B20) communicates the data to Raspberry Pi via 1-wire protocol. It is a serial protocol using a single data line plus ground reference for communication. It has low data rates and long range. It is typically used to communicate with small inexpensive devices such as digital thermometers and weather instruments.

Blood pressure sensor has serial module which sends the data to microcontroller by a standard baud rate of 9600 bps (with 8 bit data, No parity and 1 stop bit).

WHATWG wanted to develop HTML as a "Living Standard". A living standard is always updated and improved. New features can be added, but old functionality cannot

be removed. The WHATWG HTML5 Living Standard was published in 2012, and is continuously updated. W3C wanted to develop a definitive HTML5 & XHTML standard. The W3C HTML5 Recommendation was released 28 October 2014. The W3C HTML5.1 2nd Edition Recommendation was released 3 October 2017. The W3C HTML5.2 Recommendation was released 14 December 2017.

W3C standards define an Open Web Platform for application development that has the unprecedented potential to enable developers to build rich interactive experiences, powered by vast data stores that are available on any device. Although the boundaries of the platform continue to evolve, industry leaders speak nearly in unison about how HTML5 will be the cornerstone for this platform. But the full strength of the platform relies on many more technologies that W3C and its partners are creating, including CSS, SVG, WOFF, the Semantic Web stack, XML, and a variety of APIs.

In November 1996, Netscape submitted JavaScript to ECMA International to carve out a standard specification, which other browser vendors could then implement based on the work done at Netscape. This led to the official release of the language specification ECMAScript published in the first edition of the ECMA-262 standard in June 1997, with JavaScript being the most well-known of the implementations. The current version is ECMAScript 2017, released in June 2017. In January 2009, the CommonJS project was founded with the goal of specifying a common standard library mainly for JavaScript development outside the browser.

The PHP Standard Recommendation (PSR) is a PHP specification published by the PHP Framework Interop Group. Similar to Java Specification Request for Java, it serves the standardization of programming concepts in PHP.

PEP8 mentions coding conventions for the Python code comprising the standard library in the main Python distribution. Hence all the standard Python coding definitions are defined and explained in this standard document.

## 4.3 CONSTRAINTS, ALTERNATIVES & TRADE-OFF

### 4.3.1 CONSTRAINTS

- Gather real-time patient data for database creation. The data should be extensive and large in number to come to an accurate conclusion.
- Apply Machine Learning over cloud server. We have to use Flask into our system to apply Machine Learning.
- Implement Chatbot (built in Dialogflow) to our web-site. Deploy the chatbot on customized user interface.

### 4.3.2 ALTERNATIVES

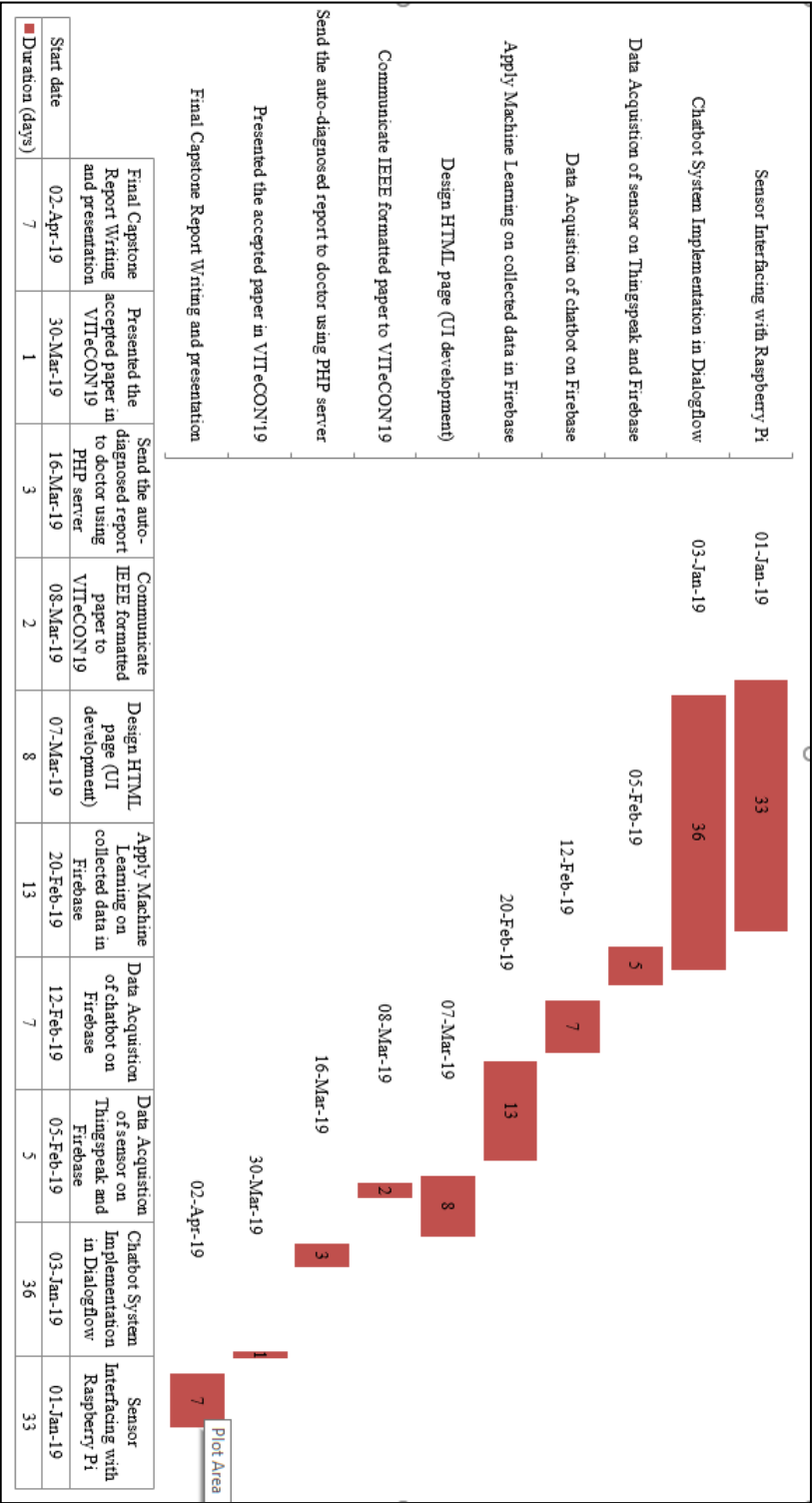
- We could have replaced a better microcontroller (Beaglebone Black, Banana Pi and some more) instead of Raspberry Pi to increase the processing efficiency and handle more data. But we considered Raspberry Pi as best option because it is the cheapest available microcontroller which suits our system application.
- Our chatbot was implemented in Dialogflow, but we have better chatbot building systems like Microsoft Cortana, Amazon Alexa, Apple Siri and some more which provide more sophisticated tools to build a better chatbot. But we considered Dialogflow as best choice because of its simplicity.
- Again, we created our backend server in Google Firebase which stores all our real-time data. But there better IoT web servers like Microsoft Azure Hub, Amazon AWS, IBM Watson Cloud service and some more. But we chose Google Firebase because of its simplicity and full stack development tools under free subscription services.

### 4.3.3 TRADE-OFFS

- Build a remotely accessible healthcare system with user-friendly interface. The interface will be very smooth and clear-cut so that user feels easy to use it.
- Implement highly accurate auto-diagnosis system which will prescribe correct medication to the patient. Getting high accuracy is critical aspect in our system, because nobody would rely on it if it has more errors (less accuracy).

5. SCHEDULE, TASKS AND MILESTONES

The following Gantt chart will show timeline of our whole project-



## 6. PROJECT DEMONSTRATION

The whole system is developed in 3 segments-

- i) IoT based hardware system to generate sensor data of the patient.
- ii) Web based application for great user experience and NLP based chatbot as virtual nurse.
- iii) Machine Learning Algorithm to diagnose about the health and heart condition of the patient.

The whole system working will be demonstrated step by step-

Step 1:

The main system is powered up to get different sensors data of the patient.

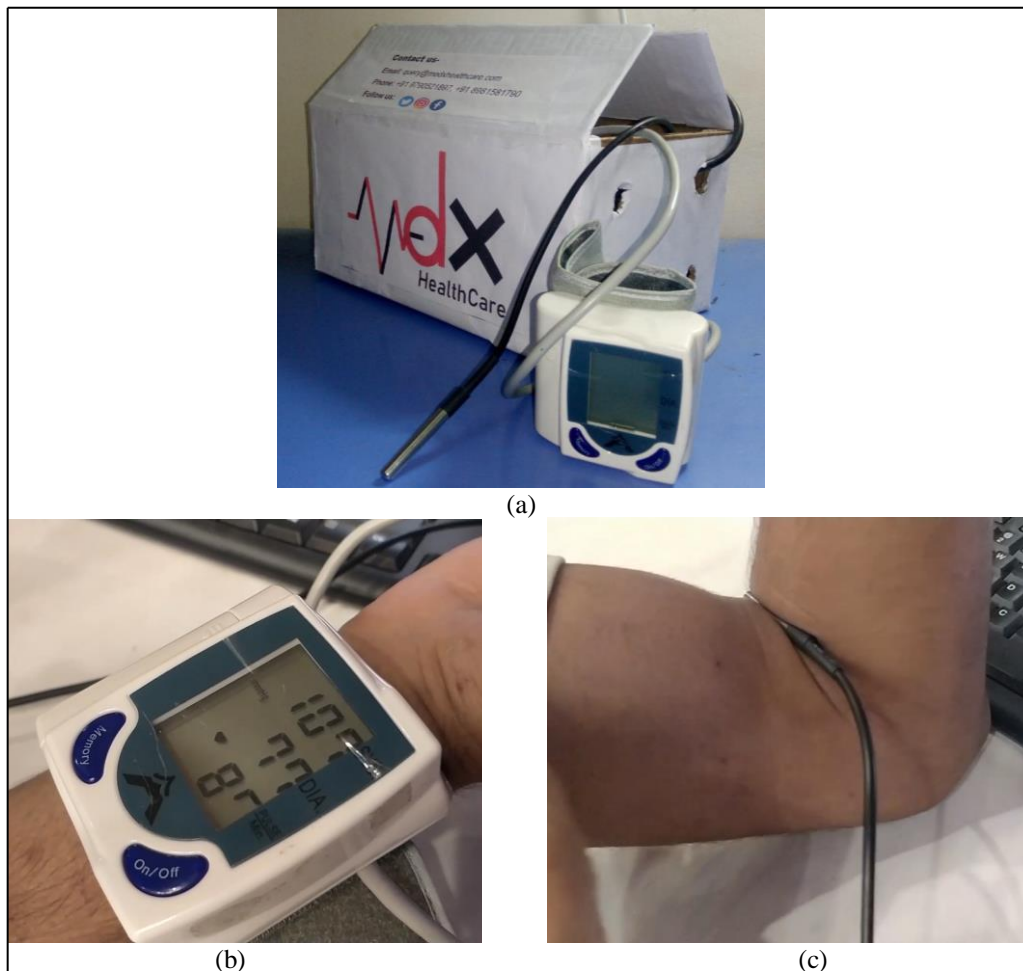
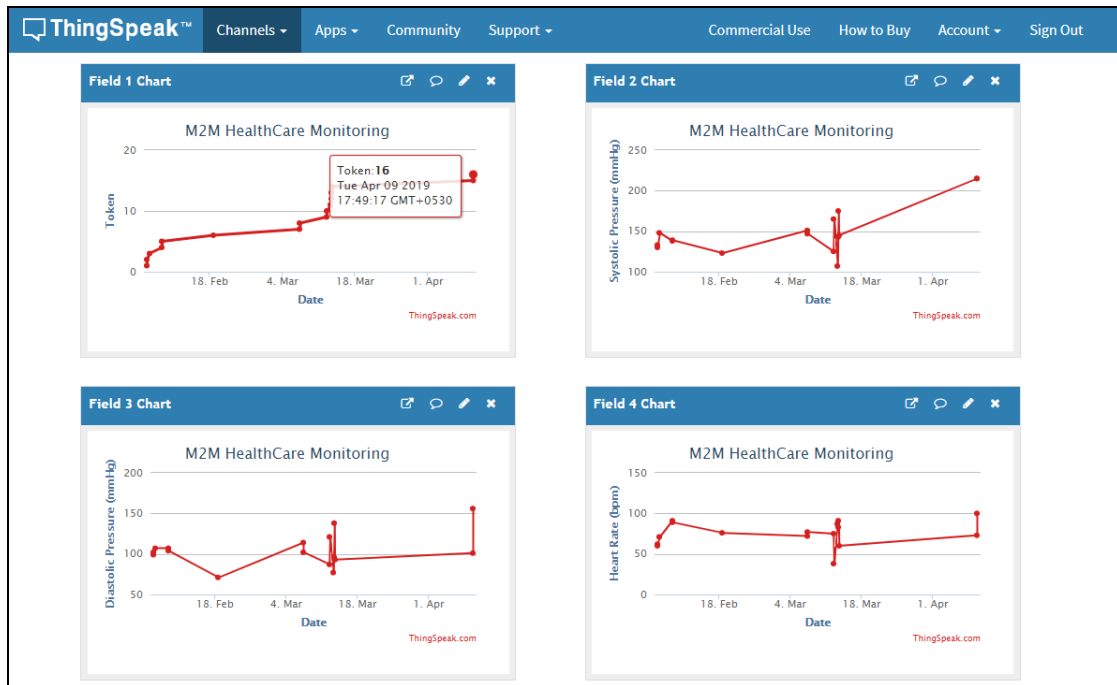


Figure 15. (a) IoT based Hardware System capturing real time data from the patient- (b) Blood pressure and heart rate and (c) Body temperature

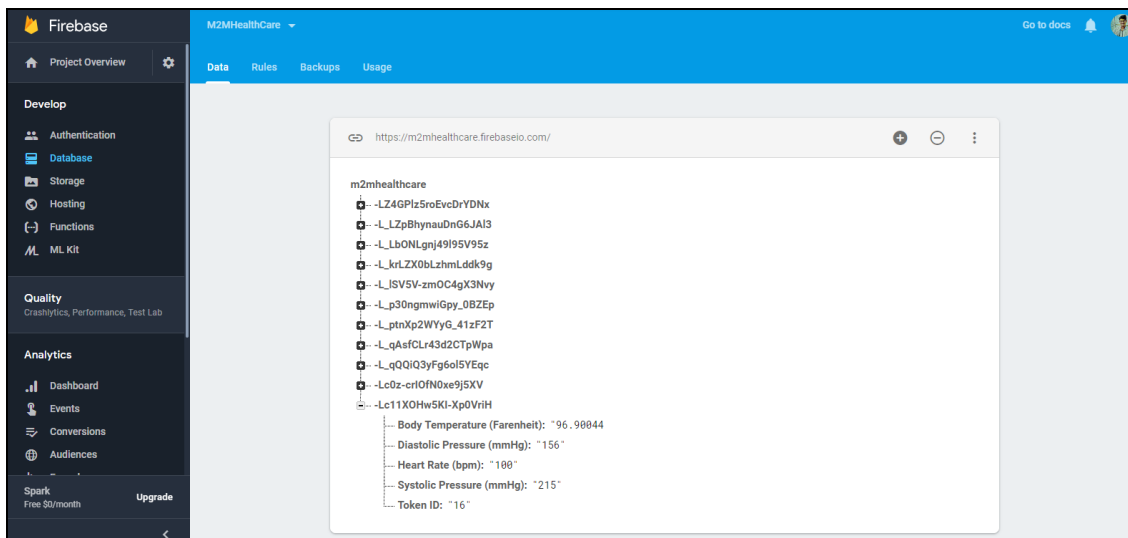


Step 2:

The sensors data is sent to ThingSpeak and then to Firebase.



(a)



(b)

Figure 16. Real time data getting stored in (a) ThingSpeak and then in (b) Firebase

Step 3:

The NLP based chatbot is created in Dialogflow.

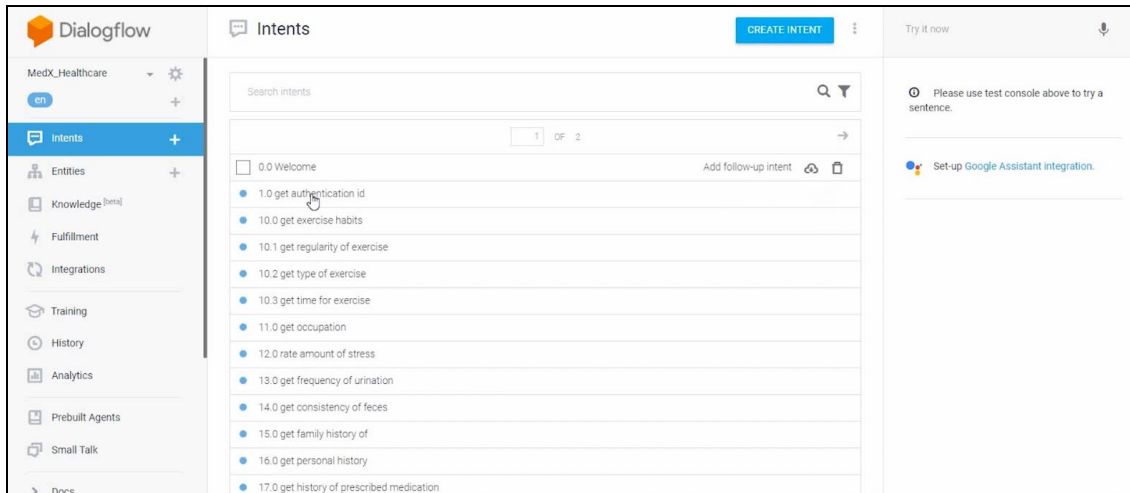
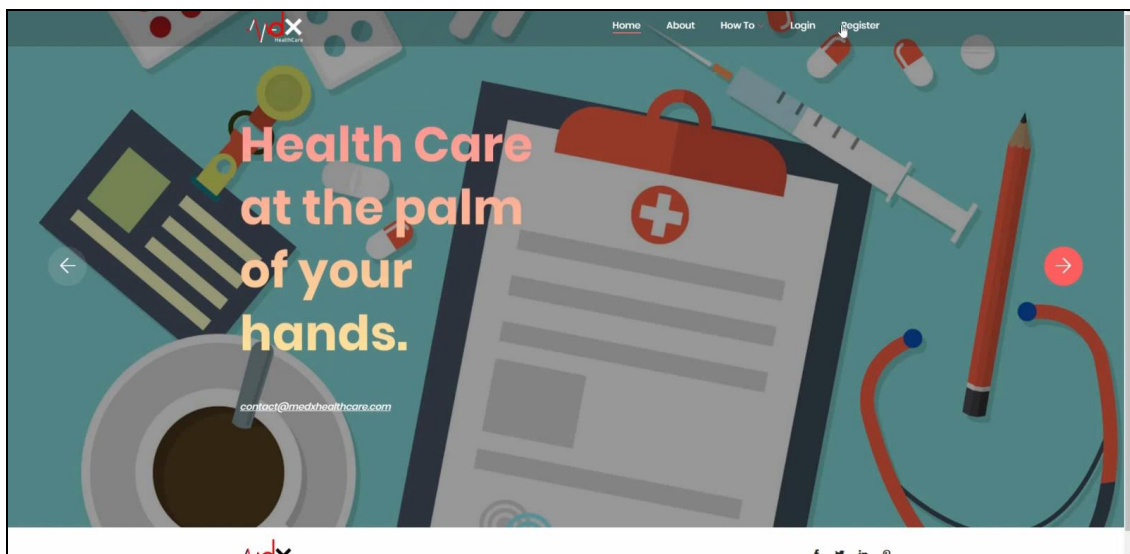


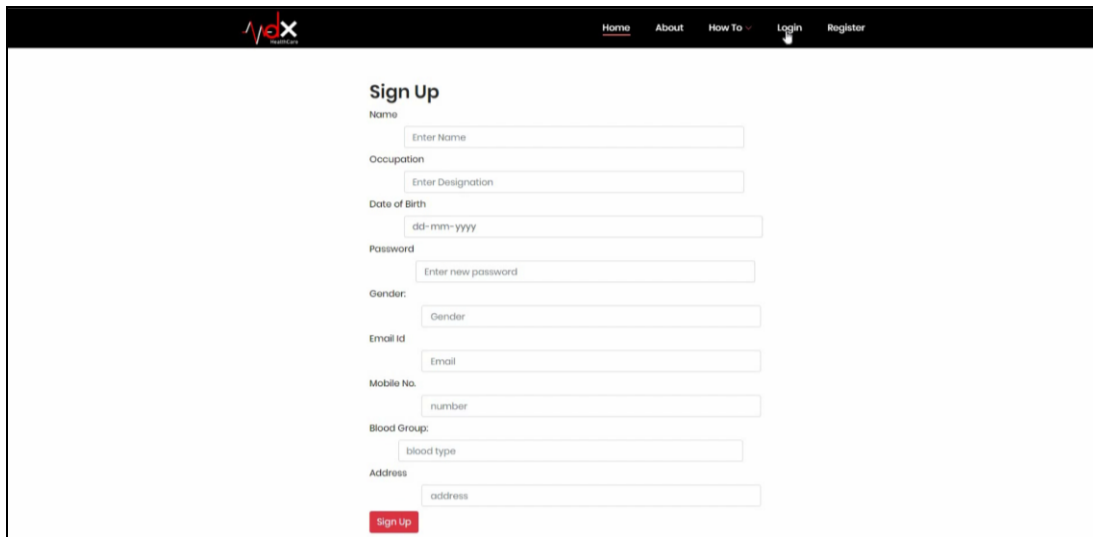
Figure 17. Dialogflow console (for NLP based Chatbot)

Step 4:

One Web based application is created for better user experience. In this Web App, user will input different parameters for the diagnosis. The chatbot is integrated here in this Web App.



(a)



**Sign Up**

Name

Occupation

Date of Birth

Password

Gender:

Email Id

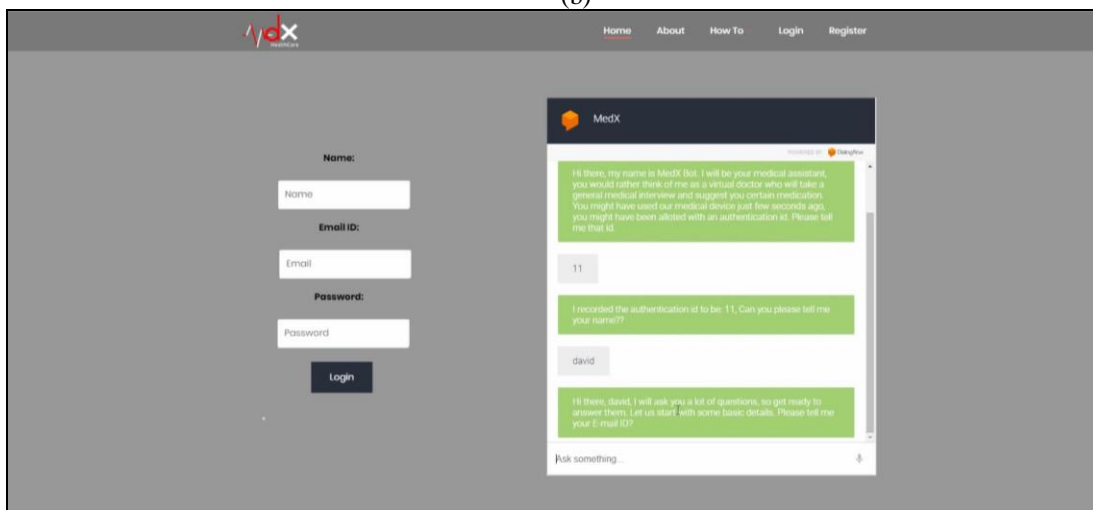
Mobile No.

Blood Group:

Address

**Sign Up**

(b)



**MedX**

Hi there, my name is MedX Bot. I will be your medical assistant, you would rather think of me as a virtual doctor who will take a general medical interview and suggest you certain medication. You might have used our medical device just few seconds ago, you might have been allotted with an authentication id. Please tell me that id.

11

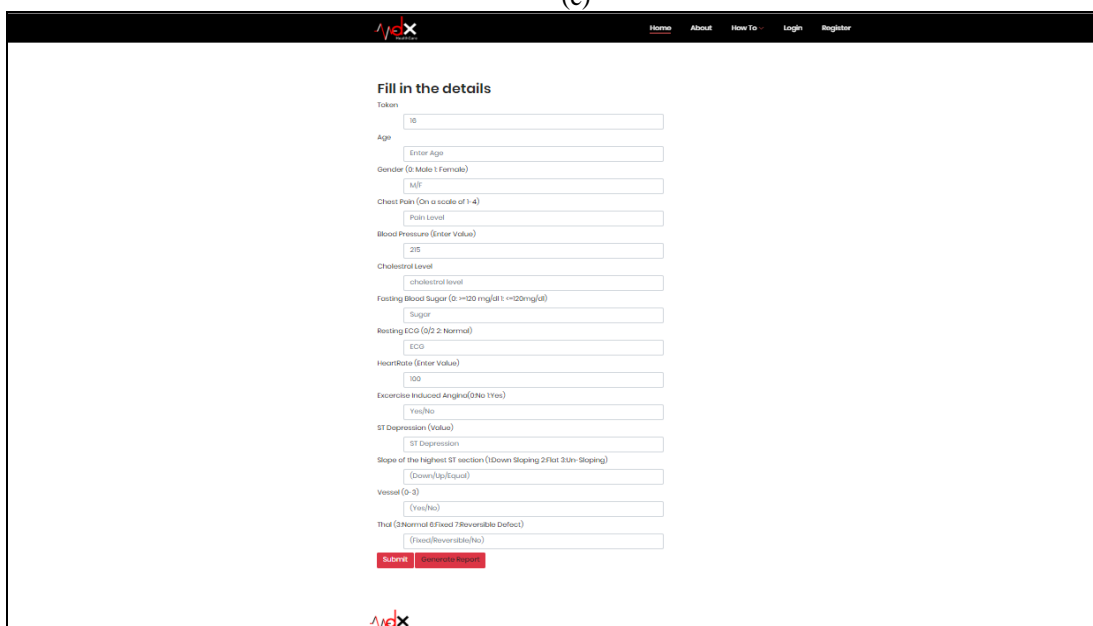
I recorded the authentication id to be: 11. Can you please tell me your name?

david

Hi there, david! I will ask you a lot of questions, so get ready to answer them. Let us start with some basic details. Please tell me your E-mail ID?

Ask something...

(c)



**Fill in the details**

Token

Age

Gender (♂ Male ♀ Female)

Chest Pain (On a scale of 1-4)

Blood Pressure (Enter Value)

Cholesterol Level

Fasting Blood Sugar (0-100 mg/dl 1-100mg/dl)

Resting ECG (0/2 2 Normal)

HeartRate (Enter Value)

Exercise Induced Angina(0/No 1/Yes)

ST Depression (Value)

Slope of the highest ST section (0Down Sloping 2Flat 3Up Sloping)

Vessel (0-3)

Thal (0Normal 6Fixed 7Reversible Defect)

**Submit** **Generate Report**

Figure 18. WebApp (a) Home page, (b) Register page, (c) Login and MedXBot page, (d) Virtual Nurse

Step 5:

The patient first logs into the website for further access.

A screenshot of a web application login page. At the top left is the MedXBot logo, which consists of a red heart with a white 'X' inside, and the text 'MedXBot HealthCare' below it. The page has a light gray background. In the center, there are three input fields: 'Name:' with 'Shivam Narula' entered, 'Email ID:' with 'shivam.narula7@gmail.co' entered, and 'Password:' with '\*\*\*\*\*' entered. Below these fields is a dark blue 'Login' button.

Figure 19. Signing in the WebApp

Step 6:

After signing in the patient talks to the MedXBot as the bot asks about various health condition which will be needed for further analysis.

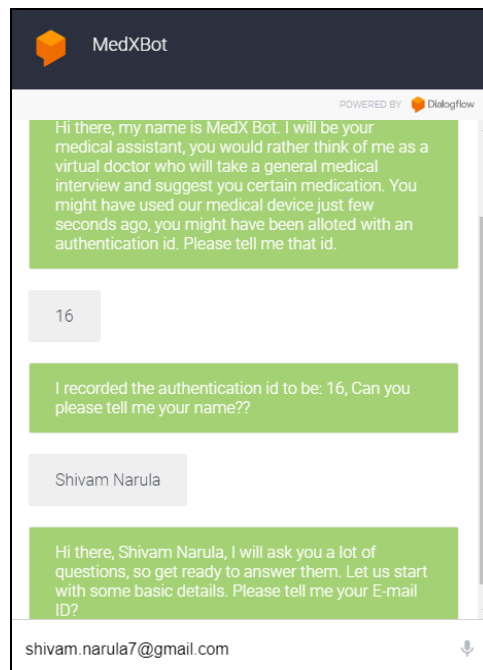
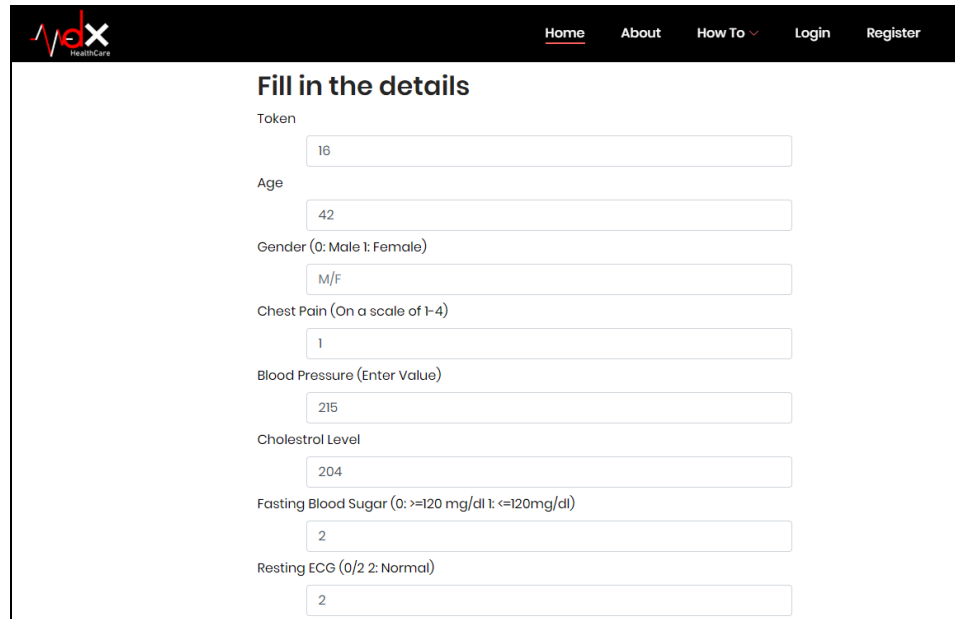
A screenshot of a chat interface for MedXBot. The header is dark blue with the MedXBot logo and 'MedXBot' text. Below the header, it says 'POWERED BY Dialogflow'. The chat area has a light green background. The first message from the bot says: 'Hi there, my name is MedX Bot. I will be your medical assistant, you would rather think of me as a virtual doctor who will take a general medical interview and suggest you certain medication. You might have used our medical device just few seconds ago, you might have been allotted with an authentication id. Please tell me that id.' The user responds with '16' in a gray bubble. The bot responds with 'I recorded the authentication id to be: 16, Can you please tell me your name??'. The user responds with 'Shivam Narula' in a gray bubble. The bot responds with 'Hi there, Shivam Narula, I will ask you a lot of questions, so get ready to answer them. Let us start with some basic details. Please tell me your E-mail ID?'. At the bottom, there is a text input field with 'shivam.narula7@gmail.com' and a microphone icon.

Figure 20. Patient talking to MeXBot

### Step 7:

After answering all the questions, the patient goes to the Virtual Nurse section to input few more parameters related to cardiac health.



**Fill in the details**

Token: 16

Age: 42

Gender (0: Male 1: Female): M/F

Chest Pain (On a scale of 1-4): 1

Blood Pressure (Enter Value): 215

Cholesterol Level: 204

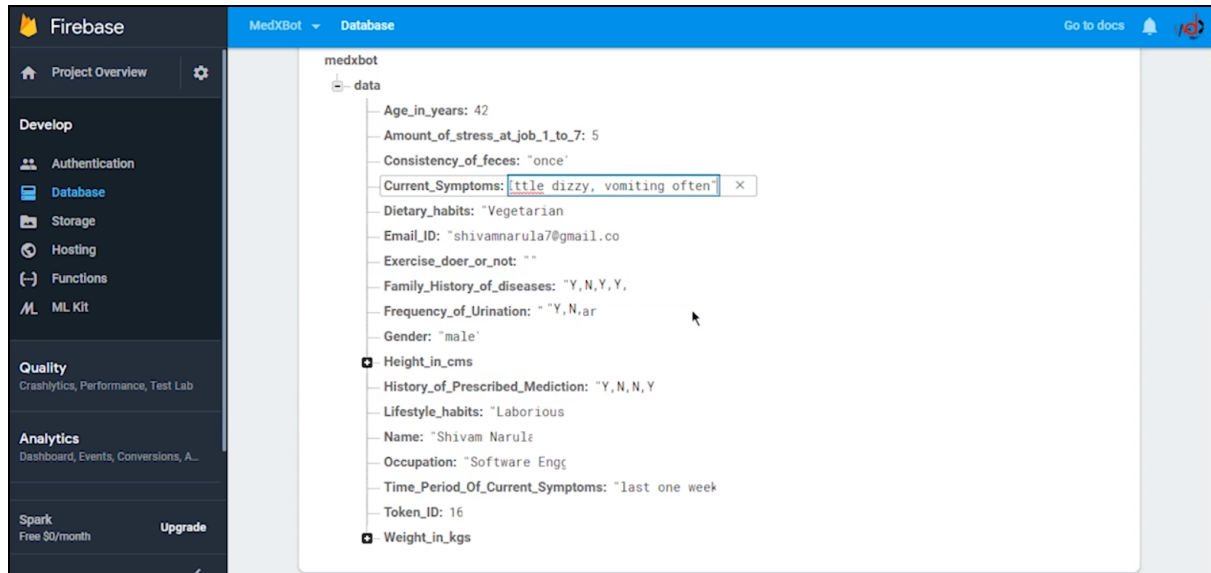
Fasting Blood Sugar (0: >=120 mg/dl 1: <=120mg/dl): 2

Resting ECG (0/2 2: Normal): 2

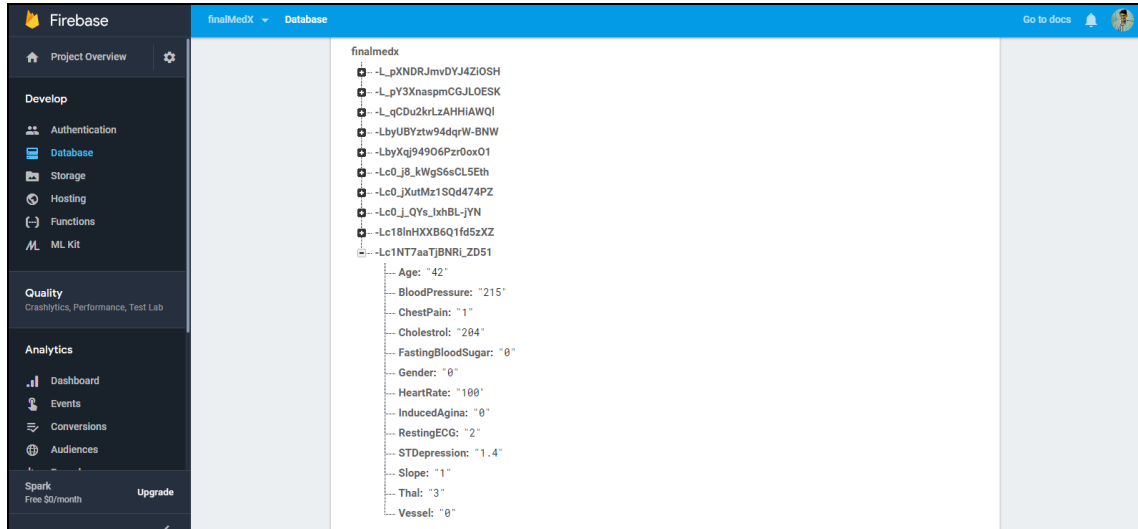
Figure 21. Virtual Nurse collecting all the data related to cardiac health of the patient

### Step 8:

MedXBot data and Virtual Nurse Data are sent to Firebase.



(a)



(b)

Figure 22. Data stored in Firebase of (a) MedXBot and (b) Virtual Nurse

Step 8:

Three Python scripts are run to fetch the data from Firebase and for implementing Machine Learning to the database to predict the heart condition of the patient.

Step 9:

The report is generated based on the data collected and then sent to the doctor for verification.

Age_in_years	Amount_of_stress_at_job_1_to_7	Consistency_of_feces	Current_Symptoms	Dietary_habits	Email_ID	Exercise_door_or_not	Family_History_of_diseases	Frequency_of_Urination	Gender	Height_in_cms	History
42	5	once	Little dizzy vomiting often	Vegetarian	shivammarula7@gmail.com		Y,N,Y,Y	regular	male	(amount: 171, unit: 'cm')	Y,N

Age	BloodPressure	ChestPain	Cholesterol	FastingBloodSugar	Gender	HeartRate	InducedAgiina	RestingECG	STDpression	Slope	Thal	Vessel
42	215	1	204	0	0	100	0	2	1.4	1	3	0

Auto-diagnosed Heart Condition

You are fine! But you can make a visit within a month.

Doctor's email ID

Dr. Saurav

Send Message

Figure 23. Report generated and sent to the doctor

## 7. COST ANALYSIS

The following table shows the budget of our system-

Table 5. Budget of our model

COMPONENT	PRICE (in ₹)
Raspberry Pi Model 3b kit (includes HDMI, RJ-45 connector, 4.5 inch TFT display, 16 GB memory card with Raspbian OS installed)	5500
Sunrom Blood Pressure Sensor with serial module (Manufactured by- Sunrom Electronics)	2550
DS18B20 Water Proof Temperature Probe - Black (1m) Original Chip	150
Pulse Sensor for Heart rate measurement	450
Standard Thingspeak API service (Open Source)	Nil
Standard Firebase API service (Open Source)	Nil
Standard Dialogflow API service (Open Source)	Nil
Web development stack (Open Source)	Nil
<b>TOTAL</b>	<b>8650</b>

This was the initial purchase cost of our system, but if we intend to fabricate our model onto a PCB layout with our own self-made ASIC chip then the budget to produce the whole model will reduce largely.

## **8. SUMMARY**

Cardiovascular disease (coronary heart disease, hypertension, arrhythmias, and other cardiovascular problems) is known as the deadliest disease that causes one-third of deaths in the world. In growing countries, the healthcare service quality is low due to lack of physicians, especially in rural areas. From this condition, a system is needed where patients in rural areas are able to check their health to a doctor who stays in the urban area, which called telemedicine.

The project can be used as an important tool for doctors and healthcare organizations to predict certain critical cases in the practice and used to advise the patient accordingly. The model from the classification will be able to answer more complex queries in the prediction of heart attack diseases.



## 9. REFERENCES

- [1] Gupta, A., Lampropoulos, J. F., Bikkdeli, B., Mody, P., Chen, R., Kulkarni, V. T., & Editor. (2013). Most important outcomes research papers on cardiovascular disease in women. *Circulation: Cardiovascular Quality and Outcomes*, 6(1), e1-e7.
- [2] Chauhan, S., & Aeri, B. T. (2015). The rising incidence of cardiovascular diseases in India: Assessing its economic impact. *J. Preventive Cardiology*, 4.
- [3] Prabhakaran, D., Jeemon, P., & Roy, A. (2016). Cardiovascular diseases in India: current epidemiology and future directions. *Circulation*, 133(16), 1605-1620.
- [4] Marso, S. P., Daniels, G. H., Brown-Frandsen, K., Kristensen, P., Mann, J. F., Nauck, M. A., & Steinberg, W. M. (2016). Liraglutide and cardiovascular outcomes in type 2 diabetes. *New England Journal of Medicine*, 375(4), 311-322.
- [5] Balambigai, S., & Jeevitha, P. (2017, January). A survey on investigation of vital human parameters from photoplethysmography signal to predict the risk of cardiovascular diseases. In *Advanced Computing and Communication Systems (ICACCS), 2017 4th International Conference on* (pp. 1-4). IEEE.
- [6] Reyes, I., Nazeran, H., Franco, M., & Haltiwanger, E. (2012, August). Wireless photoplethysmographic device for heart rate variability signal acquisition and analysis. In *34th Annual International Conference of the IEEE EMBS San Diego, California USA* (Vol. 28).
- [7] Visvanathan, A., Sinha, A., & Pal, A. (2013, November). Estimation of blood pressure levels from reflective Photoplethysmograph using smart phones. In *Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on* (pp. 1-5). IEEE.
- [8] Monte-Moreno, E. (2011). Non-invasive estimate of blood glucose and blood pressure from a photoplethysmograph by means of machine learning techniques. *Artificial intelligence in medicine*, 53(2), 127-138.
- [9] Enriko, I. K. A., Wibisono, G., & Gunawan, D. (2015, May). Designing machine-to-machine (M2M) system in health-care modeling for cardiovascular disease patients: Initial study. In *Information and Communication Technology (ICoICT), 2015 3rd International Conference on* (pp. 528-532). IEEE.
- [10] Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1), 39-59.

- [11] Enriko, I. K. A., Suryanegara, M., & Gunawan, D. My Kardio: A Telemedicine System Based on Machine-to-Machine (M2M) Technology for Cardiovascular Patients in Rural Areas with Auto-Diagnosis Feature Using k-Nearest Neighbor Algorithm.
- [12] Kementrian Kesehatan Republik Indonesia, "Profil Kesehatan Indonesia 2012," Kementrian Kesehatan Republik Indonesia Report, 2013, p. 222.
- [13] Official Raspberry Pi web-site, [https://www.element14.com/community/community/raspberry-pi/content?filterID=contentstatus\[published\]~objecttype~objecttype\[document\]&filterID=contentstatus\[published\]~language~language%5Bcpl%5D](https://www.element14.com/community/community/raspberry-pi/content?filterID=contentstatus[published]~objecttype~objecttype[document]&filterID=contentstatus[published]~language~language%5Bcpl%5D)
- [14] DSB18B20 datasheet, <https://www.instructables.com/id/How-to-use-DS18B20-Temperature-Sensor-Arduino-Tuto/>
- [15] DSB18B20 interfacing with Raspberry Pi, <https://projectiot123.com/2019/01/23/raspberry-pi-gpio-interface-with-temperature-sensor-ds18b20/>
- [16] UCI Heart Disease Databases, David Aha, July 22, 1988, <https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/heart-disease.names>
- [17] Prabhat, Anjuman, and Vikas Khullar. "Sentiment classification on Big Data using Naïve Bayes and Logistic Regression." *International Conference on Computer Communication and Informatics (ICCCI - 2017)*.
- [18] Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1), 217-222.
- [19] Ong, E. J., & Bowden, R. (2004, May). A boosted classifier tree for hand shape detection. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings*. (pp. 889-894). IEEE.
- [20] Waring, Christopher A., and Xiuwen Liu. "Face detection using spectral histograms and SVMs." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 35, no. 3 (2005): 467-476.

## Annexure I- PROGRAM CODES

### Sensor interfacing code [Python 2.7]

```
import serial
from time import sleep
import os
import glob
import time
import urllib2
import webbrowser
import json

myAPI = 'W6WIRRCGLYOMTR06'
baseUrl = 'https://api.thingspeak.com/update?api_key=%s'% myAPI

ser = serial.Serial('/dev/ttyAMA0',baudrate=9600)
print('Blood Pressure & Heart Rate Monitor...')
print('START')
ch = ser.read()
sleep(0.03)
remaining = ser.inWaiting()
ch += ser.read(remaining)
systole = int(ch[1:4])
diastole = int(ch[6:9])
bpm = int(ch[11:14])
print('Blood Pressure & Heart Rate Monitoring DONE')
print('Body Temperature Monitor...')
print('START')
os.system('modprobe wl-gpio')
os.system('modprobe wl-therm')

base_dir = '/sys/bus/wl/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/wl_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f

sum_temp=0
avg_temp=0
i=0
while i<10:
    temperatures = read_temp()
```

```

        sum_temp += temperatures[1]
        i=i+1
        time.sleep(1)
    avg_temp = sum_temp/10
    print('Body Temperature monitoring DONE')
    print('Generating Unique Token...')
    webpage =
    urllib2.urlopen('https://api.thingspeak.com/channels/695679/feeds.json?
    api_key=QORAB2WCPBDLJL4S&results=1')
    data = json.loads(webpage.read())
    for f1 in data['feeds']:
        token = int(f1['field1'])
    token = token + 1
    print('Unique Token GENERATED')
    print('-----')
    print('Showing all sensor data')
    print('Generated Authentication number is ' + str(token))
    print('Systolic Pressure (mmHg): ' + str(systole))
    print('Diastolic Pressure (mmHg): ' + str(diastole))
    print('Heart Rate (bpm): ' + str(bpm))
    print('Average Body Temperature (Celsius): '+ str(avg_temp))
    print('Sending these data to ThingSpeak Server...')
    conn = urllib2.urlopen(baseUrl +
    '&field1=%s&field2=%s&field3=%s&field4=%s&field5=%s' %
    (str(token),str(systole),str(diastole),str(bpm),str(avg_temp)))
    print conn.read()
    conn.close()
    print('DATA SENT')
    print('-----')
    webbrowser.open_new_tab('index.html')
    print('HARDWARE DATA COLLECTED & SAVED')

```

## Machine Learning Algorithms [Python3.7 & R]

### **Graphs**

```

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import matplotlib.patches as mpatches

#importing dataset and converting to dataframe
data = pd.read_csv('heart.csv', header=None)
df = pd.DataFrame(data) #data frame
y = df.iloc[:, 13]
y = y-1

def chol_age():
    x = df.iloc[:, 0:5]

```

```

x = x.drop(x.columns[1:4], axis=1)
chol_avgs = x.groupby(0, sort=True).mean()
ages = (chol_avgs[4].index.values)
avgs = (chol_avgs[4].values)
plt.plot(ages,avgs,'g-')
plt.title('Variation of Cholestrol Levels with Age')
plt.xlabel('Age(years)')
plt.ylabel('Serum Cholestrol in mg/dl')

def heart_atrack_heart_rate_bp():
    x = df.iloc[:, 0:14]
    x[14] = np.round(df[3], -1)

    x_dis = x[x[13] == 2]
    bp_set_dis = x_dis.groupby(14, sort=True)
    nums_dis = (bp_set_dis.count()[0]).index.values
    bps_dis = (bp_set_dis.count()[0]).values
    bar2 = plt.bar(nums_dis+2, bps_dis, color='r', width=2)

    x_nor = x[x[13] == 1]
    bp_set_nor = x_nor.groupby(14, sort=True)
    nums_nor = (bp_set_nor.count()[0]).index.values
    bps_nor = (bp_set_nor.count()[0]).values
    bar1 = plt.bar(nums_nor, bps_nor, color='g', width=2)

    plt.title('Resting blood pressure as heart risk indicator')
    plt.xlabel('Resting Blood Pressure Bucket')
    plt.ylabel('Number of Patients')

    plt.legend((bar1[0], bar2[0]), ('Safe', 'At Risk'))

def pie_chart_chest_pain():
    x = df.iloc[:, 0:3]
    sets = x.groupby(2).count()

```

```

    fin_lab = ['Typical Angina', 'Atypical Angina', 'Non-anginal
Pain', 'Asymptotic']

    values = (sets[0].values)

    plt.pie(values, labels=fin_lab, colors=['yellowgreen', 'gold',
'lightskyblue', 'lightcoral'], explode = [0,0.2,0,0], shadow=True,
autopct='%1.1f%%', startangle=90)

    plt.title('Chest Pain Types')

def scatter_chart():
    x = df.iloc[:, 0:13]

    sc = plt.scatter(x[7],x[4], c=y, cmap='summer')

    plt.title('Dataset Scatter')

    classes = ['Safe', 'At Risk']

    class_colours = ['g','y']

    recs = []

    for i in range(0,len(class_colours)):

        recs.append(mpatches.Rectangle((0,0),1,1,fc=class_colours[i]))

    plt.legend(recs, classes)

plt.show()

```

## Logistic Regression

```

import warnings

warnings.filterwarnings("ignore", category=FutureWarning)

import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics

from sklearn.model_selection import cross_val_score

import numpy as np

import pandas as pd

#importing dataset and converting to dataframe
data = pd.read_csv('heart.csv', header=None)

# data = np.genfromtxt(open("heart.csv","rb"),delimiter=",")

df = pd.DataFrame(data) #data frame

#extracting columns x and y

```

```

x = df.iloc[:, 0:13]
y = df.iloc[:, 13]
y = y-1
# x = data[:, 0:13]
# y = data[:, 13]
# y = y-1

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.4)

#plotting the data
fig = plt.figure()
ax1 = fig.add_subplot(1,2,1)
ax1.scatter(x[3],x[4], c=y)
ax1.set_title("Original Data")


#perform logistic regression
c = 3 #1/lambda    (regularisation contant)
model = LogisticRegression(C = c)

#10-fold cross validation
scores = cross_val_score(model, x, y, scoring='accuracy', cv=10)
print ("10-Fold Accuracy : ", scores.mean()*100)


#creation of the confusion matrix
model = model.fit(x_train,y_train)
print ("Testing Accuracy : ",model.score(x_test, y_test)*100)
predicted = model.predict(x)


ax2 = fig.add_subplot(1,2,2)
ax2.scatter(x[3],x[4], c=predicted)
ax2.set_title("Logistic Regression")
cm = metrics.confusion_matrix(y, predicted)
print (cm/len(y))
print (metrics.classification_report(y, predicted))
plt.show()

```

## Random Forest

```
library(randomForest)
set.seed(10)
RFModel <- randomForest(num ~ .,
                        data=trainData,
                        importance=TRUE,
                        ntree=2000)
#varImpPlot(RFModel)
RFPrediction <- predict(RFModel, testData)
RFPredictionprob = predict(RFModel, testData, type="prob")[, 2]

RFConfMat <- confusionMatrix(RFPrediction, testData[, "num"])

AUC$RF <- roc(as.numeric(testData$num), as.numeric(as.matrix((RFPredictionprob))))
$auc
Accuracy$RF <- RFConfMat$overall['Accuracy']
```

## Boosted Tree

```
library(caret)
set.seed(10)
objControl <- trainControl(method='cv', number=10, repeats = 10)
gbmGrid <- expand.grid(interaction.depth = c(1, 5, 9),
                      n.trees = (1:30)*50,
                      shrinkage = 0.1,
                      n.minobsinnode =10)

# run model
boostModel <- train(num ~ ., data=trainData, method='gbm',
                   trControl=objControl, tuneGrid = gbmGrid,
                   verbose=F)
# See model output in Appendix to get an idea how it selects best model
#trellis.par.set(caretTheme())
#plot(boostModel)
boostPrediction <- predict(boostModel, testData)
boostPredictionprob <- predict(boostModel, testData, type='prob')[2]
boostConfMat <- confusionMatrix(boostPrediction, testData[, "num"])

#ROC Curve
```



```

AUC$boost <-
roc(as.numeric(testData$num), as.numeric(as.matrix((boostPredictionprob)
)))$auc
Accuracy$boost <- boostConfMat$overall['Accuracy']

```

## Support Vector Machine

```

set.seed(10)
svmModel <- train(num ~ ., data = trainData2,
                  method = "svmRadial",
                  trControl = fitControl,
                  preProcess = c("center", "scale"),
                  tuneLength = 8,
                  metric = "ROC")
svmPrediction <- predict(svmModel, testData2)
svmPredictionprob <- predict(svmModel, testData2, type='prob')[2]
svmConfMat <- confusionMatrix(svmPrediction, testData2[, "num"])
#ROC Curve
AUC$svm <-
roc(as.numeric(testData2$num), as.numeric(as.matrix((svmPredictionprob))))$auc
Accuracy$svm <- svmConfMat$overall['Accuracy']

```

## K Means

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize, scale
from sklearn.model_selection import cross_val_score
from sklearn import metrics

def count(predicts):
    c = 0
    for pre in predicts:
        if pre == True:
            c+=1
    return c

#importing dataset and converting to dataframe
data = pd.read_csv('heart.csv', header=None)
df = pd.DataFrame(data) #data frame

```

```

#extracting columns x and y
x = df.iloc[:, 0:5]
x = x.drop(x.columns[1:3], axis=1)
x = pd.DataFrame(scale(x))

y = df.iloc[:, 13]
y = y-1

# x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.4)

#plotting the data
fig = plt.figure()

ax1 = fig.add_subplot(1,2,1)
ax1.scatter(x[1],x[2], c=y)
ax1.set_title("Original Data")

clusters = 2

model = KMeans(init='k-means++', n_clusters=clusters,
n_init=10,random_state=100)

scores = cross_val_score(model, x, y, scoring='accuracy', cv=10)
print ("10-Fold Accuracy : ", scores.mean()*100)

model.fit(x)

predicts = model.predict(x)
print ("Accuracy(Total) = ", count(predicts ==
np.array(y))/(len(y)*1.0) *100)
centroids = model.cluster_centers_

# print centroids

ax1.scatter(centroids[:, 1], centroids[:, 2],
            marker='x', s=169, linewidths=3,
            color='b', zorder=10)

ax2 = fig.add_subplot(1,2,2)
ax2.set_title("KMeans Clustering")
ax2.scatter(x[1],x[2], c=predicts)
ax2.scatter(centroids[:, 1], centroids[:, 2],
            marker='x', s=169, linewidths=3,
            color='b', zorder=10)

#metrics
cm = metrics.confusion_matrix(y, predicts)
print (cm/len(y))
print (metrics.classification_report(y, predicts))

plt.show()

```

## Naïve Bayes

```

import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
import matplotlib.pyplot as plt

```

```

from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import cross_val_score
import numpy as np
import pandas as pd

#importing dataset and converting to dataframe
data = pd.read_csv('heart.csv', header=None)
# data = np.genfromtxt(open("heart.csv", "rb"), delimiter=",")

df = pd.DataFrame(data) #data frame

#extracting columns x and y
x = df.iloc[:, 0:13]
y = df.iloc[:, 13]
y = y-1

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.4)

#plotting the data
fig = plt.figure()
ax1 = fig.add_subplot(1,2,1)
ax1.scatter(x[3],x[4], c=y)
ax1.set_title("Original Data")

model = MultinomialNB()

#10-fold cross validation
scores = cross_val_score(model, x, y, scoring='accuracy', cv=10)
print ("10-Fold Accuracy : ", scores.mean()*100)

#Testing Accuracy
model.fit(x_train,y_train)

predicts = model.predict(x)
ax2 = fig.add_subplot(1,2,2)
ax2.scatter(x[3],x[4], c=predicts)
ax2.set_title("Naive Bayes")

cm = metrics.confusion_matrix(y, predicts)
print (cm/len(x))
print (metrics.classification_report(y, predicts))

plt.show()

```

## **K Means Naïve Bayes**

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import scale
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB

```

```

from sklearn. import KFold

#importing dataset and converting to dataframe
data = pd.read_csv('heart.csv', header=None)
df = pd.DataFrame(data) #data frame

FP = 0
FN = 0
TN = 0
TP = 0

def nbkmh(train_index, test_index):
    #extracting columns x and y separately for kmeans and naive bayes
    classifiers
        x_kmeans = df.iloc[:, 0:5]
        x_kmeans = x_kmeans.drop(x_kmeans.columns[1:3], axis=1)
        x_kmeans = pd.DataFrame(scale(x_kmeans))

        x_naive = df.iloc[:, 0:13]

        y = df.iloc[:, 13]
        y = y-1

        y_train = pd.Series(y.iloc[train_index])
        y_test = pd.Series(y.iloc[test_index])

        x_train_kmeans = x_kmeans.iloc[train_index, :]
        x_test_kmeans = x_kmeans.iloc[test_index, :]

        x_train_naive = x_naive.iloc[train_index, :]
        x_test_naive = x_naive.iloc[test_index, :]

    #Kmeans model for the processed data
    clusters = 5
    model_kmeans = KMeans(init='k-means++', n_clusters=clusters,
n_init=10,random_state=10000)
    model_kmeans.fit(x_train_kmeans)
    kmean_predictions = model_kmeans.predict(x_train_kmeans)

    #building dataset according to clusters
    x = [pd.DataFrame() for ii in range(0,clusters)]
    y = [pd.Series() for ii in range(0,clusters)]

    for kmean_prediction,i in zip(kmean_predictions,
range(len(x_train_kmeans))):
        row_x = x_train_naive.iloc[i, :]
        row_y = pd.Series(y_train.iloc[i])
        index = int(kmean_prediction)
        x[index] = x[index].append(row_x, ignore_index=True)
        y[index] = y[index].append(row_y)

    #applying naive bayes classifier
    clstr_n = [MultinomialNB(alpha=2,fit_prior=True) for ii in
range(0,clusters)]

    for i in range(0,clusters):

```

```

        clstr_n[i].fit(x[i], y[i])

    #calculating predictions for the testing based on the hybrid
    algorithm
    predicts = []
    c=0
    for i in range(len(x_test_kmeans)):
        prediction = model_kmeans.predict(x_test_kmeans.iloc[i,
:]).reshape(1,-1))
        prediction = int(prediction)
        pred_naive =
clstr_n[prediction].predict(x_test_naive.iloc[i, :].reshape(1,-1))
        predicts.append(pred_naive)
        if pred_naive == y_test.iloc[i]:
            c+=1

    print ((c*100.0)/len(x_test_kmeans))
    # print ("Test set accuracy : ", ((c*100.0)/len(x_test_kmeans)))

    #metrics
    predicts = np.array(predicts)
    cm = metrics.confusion_matrix(y_test, predicts)/len(y_test)
    # print (cm)
    global FP
    global FN
    global TN
    global TP

    FP += cm[0][0]
    FN += cm[1][0]
    TN += cm[0][1]
    TP += cm[1][1]

    return ((c*100.0)/len(x_test_kmeans))

def main():
    scores = []
    kf = KFold(n=df.shape[0], n_folds=10)
    for (train_index,test_index),i in zip(kf,range(0,10)):
        print("Iteration " + str(i+1) + " : ")
        scores.append(nbkmh(train_index, test_index))
    print("\n 10 Fold Accuracy",np.array(scores).mean())
    print("FP", FP*10)
    print("FN", FN*10)
    print("TN", TN*10)
    print("TP", TP*10)

if __name__ == '__main__':
    main()

```

## KNN

```

import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

```

```

from sklearn.preprocessing import normalize, scale
from sklearn.model_selection import cross_val_score
import numpy as np
import pandas as pd

# importing dataset and converting to dataframe
data = pd.read_csv('heart.csv', header=None)
# data = np.genfromtxt(open("heart.csv", "rb"), delimiter=",")

df = pd.DataFrame(data) #data frame

#extracting columns x and y
x = df.iloc[:, 0:5]
x = x.drop(x.columns[1:3], axis=1)
x = pd.DataFrame(scale(x))

y = df.iloc[:, 13]
y = y-1

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.4)

#plotting the data
fig = plt.figure()
ax1 = fig.add_subplot(1,2,1)
ax1.scatter(x[1],x[2], c=y)
ax1.set_title("Original Data")

model = KNeighborsClassifier(n_neighbors=5)

#10-fold cross validation
scores = cross_val_score(model, x, y, scoring='accuracy', cv=10)
# print scores
print ("10-Fold Accuracy : ", scores.mean()*100)

#creation of the confusion matrix
model.fit(x_train,y_train)
print ("Testing Accuracy : ",model.score(x_test, y_test)*100)
predicted = model.predict(x)

ax2 = fig.add_subplot(1,2,2)
ax2.scatter(x[1],x[2], c=predicted)
ax2.set_title("KNearestNeighbours")

cm = metrics.confusion_matrix(y, predicted)
print (cm/len(y))
print (metrics.classification_report(y, predicted))

plt.show()

```

## **Fuzzy KNN**

```

import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import normalize, scale
from sklearn.model_selection import cross_val_score

```

```

import numpy as np
import pandas as pd

#importing dataset and converting to dataframe
data = pd.read_csv('heart.csv', header=None)
# data = np.genfromtxt(open("heart.csv", "rb"), delimiter=",")

df = pd.DataFrame(data) #data frame

#extracting columns x and y
x = df.iloc[:, 0:5]
x = x.drop(x.columns[1:3], axis=1)
x = pd.DataFrame(scale(x))

y = df.iloc[:, 13]
y = y-1

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.4)

#plotting the data
fig = plt.figure()
ax1 = fig.add_subplot(1,2,1)
ax1.scatter(x[1],x[2], c=y)
ax1.set_title("Original Data")

model = KNeighborsClassifier(n_neighbors=5, weights='distance')

#10-fold cross validation
scores = cross_val_score(model, x, y, scoring='accuracy', cv=10)
# print scores
print ("10-Fold Accuracy : ", scores.mean()*100)

#creation of the confusion matrix
model.fit(x_train,y_train)
print ("Testing Accuracy : ",model.score(x_test, y_test)*100)
predicted = model.predict(x)

ax2 = fig.add_subplot(1,2,2)
ax2.scatter(x[1],x[2], c=predicted)
ax2.set_title("Fuzzy KNearestNeighbours")

cm = metrics.confusion_matrix(y, predicted)
print (cm/len(y))
print (metrics.classification_report(y, predicted))

plt.show()

```

## **NN**

"""

Variables to be manually initialised:

- lambda
- epsilon
- hidden\_layer\_size
- K

```

"""

import numpy as np
from scipy import optimize
from sklearn.preprocessing import scale
from sklearn import metrics

def featureNormalize(z):
    return scale(z)

def sigmoid(z):
    r = 1.0 / (1.0 + np.exp(-z))
    return r

def sigmoidGrad(z):
    r = sigmoid(z)
    r = r * (1.0 - r)
    return r

def randomizeTheta(l, epsilon):
    return ((np.random.random((l, 1)) * 2 * epsilon) - epsilon)

def KFoldDiv(X, y, m, n, K):
    sz = int(np.ceil(m / K))
    if n == 1:
        X_train = X[sz:, :]
        X_test = X[:sz, :]
        y_train = y[sz:]
        y_test = y[:sz]
    elif n == K:
        X_train = X[((n-1)*sz), :]
        X_test = X[((n-1)*sz):, :]
        y_train = y[((n-1)*sz)]
        y_test = y[((n-1)*sz):]
    else:
        X_train = np.vstack((X[((n-1)*sz), :], X[(n*sz):, :]))
        X_test = X[((n-1)*sz):(n*sz), :]
        y_train = np.vstack((y[((n-1)*sz)], y[(n*sz):]))
        y_test = y[((n-1)*sz):(n*sz)]
    return (X_train, y_train, X_test, y_test)

def nnCostFunc(Theta, input_layer_size, hidden_layer_size, num_labels,
X, y, lambda):
    Theta1, Theta2 = np.split(Theta, [hidden_layer_size *
(input_layer_size+1)])
    Theta1 = np.reshape(Theta1, (hidden_layer_size,
input_layer_size+1))
    Theta2 = np.reshape(Theta2, (num_labels, hidden_layer_size+1))
    m = X.shape[0]
    y = (y == np.array([(i+1) for i in range(num_labels)])) .astype(int)

    a1 = np.hstack((np.ones((m, 1)), X))
    z2 = np.dot(a1, Theta1.T)
    a2 = np.hstack((np.ones((m, 1)), sigmoid(z2)))
    h = sigmoid(np.dot(a2, Theta2.T))

    cost = ((lambda/2) * (np.sum(Theta1[:, 1:] ** 2) + np.sum(Theta2[:,
1:] ** 2)) - np.sum((y * np.log(h)) + ((1-y) * np.log(1-h)))) / m

```



```

        return cost

def nnGrad(Theta, input_layer_size, hidden_layer_size, num_labels, X,
y, lambda):
    Theta1, Theta2 = np.split(Theta, [hidden_layer_size *
(input_layer_size+1)])
    Theta1 = np.reshape(Theta1, (hidden_layer_size,
input_layer_size+1))
    Theta2 = np.reshape(Theta2, (num_labels, hidden_layer_size+1))
    m = X.shape[0]
    y = (y == np.array([(i+1) for i in range(num_labels)])) .astype(int)

    a1 = np.hstack((np.ones((m, 1)), X))
    z2 = np.dot(a1, Theta1.T)
    a2 = np.hstack((np.ones((m, 1)), sigmoid(z2)))
    h = sigmoid(np.dot(a2, Theta2.T))

    delta_3 = h - y
    delta_2 = np.dot(delta_3, Theta2[:, 1:]) * sigmoidGrad(z2)
    Theta2_grad = (np.dot(delta_3.T, a2) + (lambda *
np.hstack((np.zeros((Theta2.shape[0], 1)), Theta2[:, 1:]))) / m
    Theta1_grad = (np.dot(delta_2.T, a1) + (lambda *
np.hstack((np.zeros((Theta1.shape[0], 1)), Theta1[:, 1:]))) / m

    grad = np.hstack((Theta1_grad.flatten(), Theta2_grad.flatten()))
    return grad

K = 10
lambda = 0.03
epsilon = 0.12

input_layer_size = 13
hidden_layer_size = 20
num_labels = 2

X = np.genfromtxt('test.csv', delimiter=',')
m, n = X.shape
n -= 1

y = X[:, n].astype(int).reshape((m, 1))
X = featureNormalize(X[:, :n])
foldAcc = np.ndarray((K, 1))

FP = 0
FN = 0
TN = 0
TP = 0

for i in range(K):
    X_train, y_train, X_test, y_test = KFoldDiv(X, y, m, i+1, K)

    initTheta = randomizeTheta((hidden_layer_size *
(input_layer_size+1)) + (num_labels * (hidden_layer_size+1)), epsilon)
    Theta = optimize.fmin_bfgs(nnCostFunc, initTheta, fprime=nnGrad,
args=(input_layer_size, hidden_layer_size, num_labels, X_train,
y_train, lambda), maxiter=3000)

```

```

    Theta1, Theta2 = np.split(Theta, [hidden_layer_size *
(input_layer_size+1)])
    Theta1 = np.reshape(Theta1, (hidden_layer_size,
input_layer_size+1))
    Theta2 = np.reshape(Theta2, (num_labels, hidden_layer_size+1))

    h1 = sigmoid(np.dot(np.hstack((np.ones((X_test.shape[0], 1)),
X_test)), Theta1.T))
    h2 = sigmoid(np.dot(np.hstack((np.ones((h1.shape[0], 1)), h1)),
Theta2.T))
    predicted = h2.argmax(1) + 1
    predicted = predicted.reshape((predicted.shape[0], 1))
    foldAcc[i] = np.mean((predicted == y_test).astype(float)) * 100

    cm = (metrics.confusion_matrix(y_test, predicted))/len(y_test)

    FP += cm[0][0]
    FN += cm[1][0]
    TN += cm[0][1]
    TP += cm[1][1]

    print('Test Set Accuracy for %dth fold: %f\n' % (i+1, foldAcc[i]))

meanAcc = np.mean(foldAcc)
print('\nAverage Accuracy: ', meanAcc)
print("")

print(FP)
print(FN)
print(TN)
print(TP)

```

## Annexure II- ViTECoN'19 CONFERENCE

We presented our project in an IEEE conference with following details-

**Conference ID-** 45748

**Conference Name-** International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN-2019)

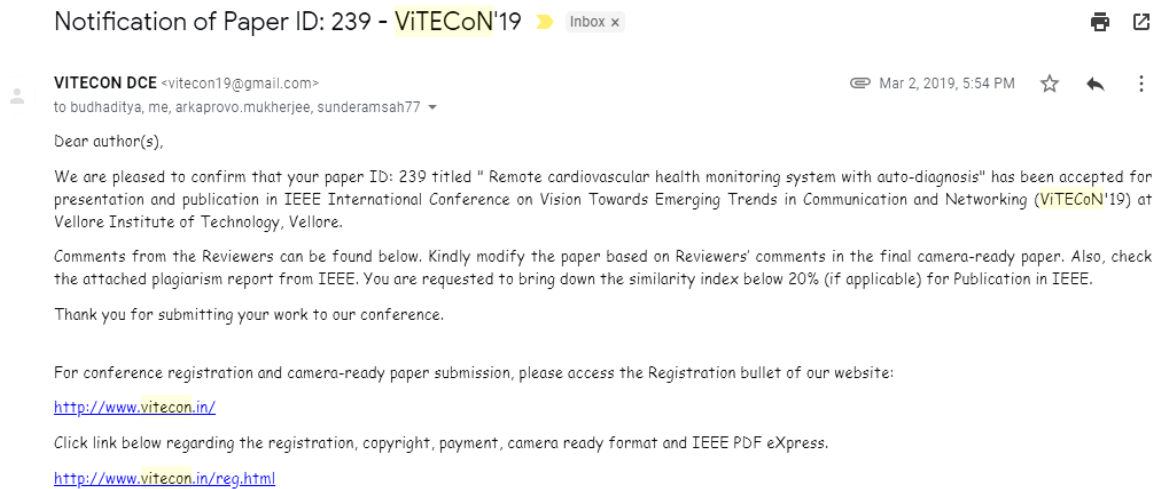
**Publication-** IEEE Xplore

**ISBN-** 978-1-5386-9353-7

**Conducted by-** VIT Vellore

**Conference Date-** 30 & 31<sup>st</sup> March 2019

The following screenshot is the acceptance mail for our project.





**VIT**  
Vellore Institute of Technology  
Chartered in the University under section 3 of UG Act, 1956

**VITECoN'19**

**IEEE**  
Madras Section

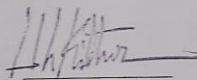
**IEEE ComSoc**  
IEEE Communications Society  
VIT Student Chapter

**IEEE International Conference on Vision  
Towards Emerging Trends in Communication  
and Networking 2019 (ViTECoN'19)**

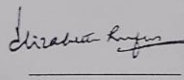
IEEE Conference Record No: 45748

**CERTIFICATE OF PARTICIPATION**

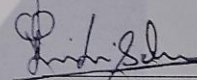
This is to certify that Dr./Mr./Ms. Saurav Mohapatra  
has ~~participated~~/presented the paper titled Remote Cardiovascular  
health monitoring System with Auto-diagnosis  
in ViTECoN'19 in association with IEEE Madras Section, Organized by the  
School of Electronics Engineering (SENSE), Vellore Institute of Technology,  
Vellore - 632014, Tamil Nadu, India on 30<sup>th</sup> & 31<sup>st</sup> March 2019.



Organizing Chair  
Dr. Kittur Harish Mallikarjun  
DEAN - SENSE



Conference Chair  
Dr. Elizabeth Rufus  
Professor - SENSE



Conference Co-chair  
Dr. Thanikaiselvan V  
HOD - Dept of Communication Engineering



**VIT**  
Vellore Institute of Technology  
Chartered in the University under section 3 of UG Act, 1956

**VITECoN'19**

**IEEE**  
Madras Section

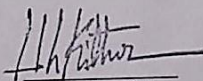
**IEEE ComSoc**  
IEEE Communications Society  
VIT Student Chapter

**IEEE International Conference on Vision  
Towards Emerging Trends in Communication  
and Networking 2019 (ViTECoN'19)**

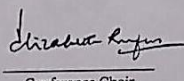
IEEE Conference Record No: 45748

**CERTIFICATE OF PARTICIPATION**

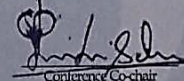
This is to certify that Dr./Mr./Ms. Sunderam Sab  
has ~~participated~~/presented the paper titled Remote Cardiovascular  
health monitoring system with Auto-diagnosis  
in ViTECoN'19 in association with IEEE Madras Section, Organized by the  
School of Electronics Engineering (SENSE), Vellore Institute of Technology,  
Vellore - 632014, Tamil Nadu, India on 30<sup>th</sup> & 31<sup>st</sup> March 2019.



Organizing Chair  
Dr. Kittur Harish Mallikarjun  
DEAN - SENSE



Conference Chair  
Dr. Elizabeth Rufus  
Professor - SENSE



Conference Co-chair  
Dr. Thanikaiselvan V  
HOD - Dept of Communication Engineering



### Annexure III- Reboot'19 HACKATHON

