Università di Genova
DIBRIS DIPARTIMENTO
DI INFORMATICA, BIOINGEGNERIA,
ROBOTICA E INGEGNERIA DEI SISTEMI

Implementation of a Health Remote Assistance System Based on IoT and Artificial Intelligence

Akshi Sharma

Master Thesis

Università di Genova, DIBRIS Via Opera Pia, 13 16145 Genova, Italy
https://www.dibris.unige.it/

Università
di Genova

MSc Computer Science
Data Science and Engineering Curriculum

Implementation of a Health Remote Assistance System Based on IoT and Artificial Intelligence

Akshi Sharma

Advisor: Prof. Franceso Masulli,
Prof. Stefano Rovetta,
Prof. Alberto Cabri

Examiner: Prof. Giorgio Delzanno

March, 2023

Abstract

With the increase in life expectancy there has been an increase in the old-age related problems. The foundation of the project, undertaken during this thesis, lies in finding innovative methods to ensure the older adult population can maintain their quality of life as they battle with chronic illness.
We discuss the *state of the art* in this field and how with the technological advancement the possibility of improving health care is immense. During the thesis, a remote assistance service was installed in RSA Minoretti and RSA Valpolcevra in collaboration with Hassisto and Vega Research Laboratories. The IOT devices used for this purpose, provided by Hassisto, are explained in the subsections along with the technology they work on. The data recorded by the IOT devices was analysed and after discussions between medical and the technical teams approaches to development were decided. Most of the data processing was a backend activity, but for the healthcare personnel visualizing the data in a user-friendly and comprehensible manner was important, so that aspect also remained a primary factor during the thesis.
Considering the requirements of the project the software technologies too had to be evaluated; Python and Javascript were opted for backend and frontend respectively, reasons for which are discussed in the following the chapters. Finally, analysis of both, machine learning technologies and visualization results, was done to gauge the performances.

Contents

Chapter 1

Introduction

1.1 State of art of Health Remote Assistance Systems

In the recent years, there has been a substantial improvement in the life expectancy of the general population. Average life expectancy has improved due to better health care and invention of new medicines resulting in increase in number of geriatric population. This has led to rise in a whole set of diseases and problems directly related to the old age. The last decade has witnessed an exponential increase in older adult population suffering from chronic life-long diseases and needing healthcare[Esp+18]. The paper [Esp+18] enunciates how with the growing world, there is a need for a better and more effective way to manage the health of old people. With the exposure and access to devices and technologies, that are now omnipresent in our daily life, it has become possible to integrate them into the healthcare mainstream for automated and smart monitoring. There are a lot of new technologies like sensors, smart bands, etc which are helping enhance the capabilities of health monitoring systems (HMS). These HMS play a significant role in not only reducing hospitalization and burden of medical staff, but also bring down the consultation time and waiting lists. Most important impact of this is that it will lower the overall expenditure of healthcare costs borne by a person affected by long-term ailments that demand recurrent or continual examinations.

An advanced version of health monitoring systems (HMS) is called Smart health monitoring systems (SHMS) which deploy smart devices to address health monitoring issues. Smart health monitoring systems (SHMS) can be grouped into different categories. General health monitoring systems (GHMS) pertains to systems which monitor multiple parameters and general disease by recording vital signs. Remote health monitoring systems (RHMS) refer to those systems which can send data to/or from a remote location. They can be used to

perform multi-parameter functions, which cover a variety of symptoms and can be utilized in individual homes as well as hospitals. Mobile health monitoring systems (MHMS) refer to mobile phones, personal digital assistants (PDAs), pocket personal computer (PC) based systems which are used as the main processing station or as the main working module. Wearable health monitoring systems (WHMS) are wearable devices or biosensors that can be worn by patients in combination with RHMS and/or MHMS [BG13].

An example of a Smart health monitoring system is in [Bai+14] who have implemented an integration of Wearable HMS with Mobile HMS. They used Blood pressure monitor and a Pulse Oximeter, both wireless Bluetooth devices that measure blood pressure and oxygen saturation respectively. For Blood glucose meter they used a wireless infrared connected device, while the Ear temperature was measured using an instant ear thermometer. A Set-top-box was responsible for data transmission over secure internet connection to personal PC or laptop. To ensure data accuracy and detect data loss they carried out cross validation on data. Vital sign information was recorded by wireless medical devices and a registered nurse performed manual readings, using standard ward techniques, of the same parameters. The researchers then manually cross checked the records for inaccuracies and time delays. They employed a fuzzy logic model to assess and interpret multiple physiological parameters to detect the possibility of bradycardia (E1). The rules to create the logic were made by combining values of five vital signs: HR, BP, P, SpO2 and T with two levels of priority-P1 and P2, where P1 is higher priority than P2. Fuzzy sets were defined keeping in mind the weightage of any specific parameter for a particular condition. For example for hypotension the following set was defined:

- Low BP (Lbp), High HR (Hhr) and High Pulse rate (Hp) for priority-2 Hypotension,
- Very Low BP (VLbp), Very High HR (VHhr) and Very High pulse rate (VHp) for priority-1 Hypotension

Hypotension will primarily be affected by BP values but for clinical relevance high heart rate as well as high pulse rate will also determine the severity of the condition.

In [BGL15] the authors suggested the use of tablet-based applications. Here the focus was on making the recorded information of the patients easily accessible to the care givers, so that the evaluation and processing of the patient's data can be made uncomplicated. They adopted five screen designs for the purpose of keeping it simple for the user.

- Patient profile - stores patient related information
- Vital signs - contains latest patient physiological information recorded by the Bluetooth devices as well as the medical staff
- History - contains falls history, medications, mental status etc

- Medical Notes - Summary and medical notes of the patient's status
- Contact - allows exchange of information between medical professionals and registered care giver in case of emergency or send clinical information

It also provided the basic functionality required by the clinicians to complete their typical tasks on their routine hospital round. They utilised four components: information and communication technologies (ICT), end-user consideration, security and data analysis; to gauge the success of the monitoring system, which their model was able to fulfill.

Wireless Sensor Networks (WSNs) for healthcare are being used widely for different medical fields. This technology provides healthcare services for patients, especially who suffer from chronic diseases that require catering continuous medical monitoring and get rid of disturbance caused by the instruments. [AAA18] came up with a use of this technology, with the aim to liberate the patients from intrusive instruments that make their check-ups tedious and time-consuming. They proposed a system that works in three stages: *Sensing Circuit Stage* that uses a heart pulse sensor to detect the heart pulse, which is then amplified and sent to the *Filtering and Mapping Stage*. Here the signal is filtered of any noise and mapped to GUI for visualizing using Arduino Uno micro-controller. In the *Network Connecting Stage*, the Arduino board is connected to the internet by using Ethernet shield to send the signals and display the results on computer-based or smartphone-based application.

According to [TSM19] it is important to provide best medical monitoring to the patients, but it is equally important to maintain their health-related quality of life (HRQoL). Frailty or pre-frailty status(PFS) is highly correlated with the decrease in the QoL. The author's idea is to create a monitoring system that would be able to achieve a remote continuous monitoring without any subject collaboration while keeping low cumbersomeness. The emergence of Internet of things and Artificial Intelligence has enabled multi-fold advancement in the remote healthcare systems.

Medical systems, integrated with IoT features, can support healthcare activities and these systems are categorized as Internet of Medical Things (IoMT) by [Kot+19]. They suggest working on a human-centric perspective of designing IoMT systems where human participants can be an integral part of these systems. They believe that human involvement as critical system "components" leads to the emergence of Cyber-Physical Human Systems (CPHSs). In their work [Kot+19] , they analyze the human concerns related to the functioning of these systems' usage, and the criticalities that are extracted from these concerns. Their primary focus was on the Remote Elderly Monitoring System (REMS) and the Smart Ambulance System (SAS) use cases, that belong to the healthcare domain that combines IoMT technologies with classic healthcare practices.
Continuing with the idea of human role as a crucial unit in the current healthcare system,

they evaluate different REMS and SAS systems from the viewpoint of the actors who are a part of this system and affected by its functioning. They use the term "actors" for any human participant (or organization) that interacts with the system-actively and dynamically - in such a way that it affects the system or vice versa. The actors have specific tasks, depending on their area of concerns; these concerns give rise to specific criticality types that puts restrictions on the tasks and the functionality of the system. In REMS patients and caregivers are the actors, each with a specific set of tasks and criticalities arising from it.

Tasks for the patient in REMS: ***Acquire Device, Affording these devices, Wear Device, Activate Device, Measure Vital Signs, View Vital Signs, Manage Personal Data, Communicate on Emergency, Deactivate Device***

REMS caregiver's monitoring process: ***Monitor Patient Vital Signs, Review Patient Record, Act on Emergency***

REMS Criticalities: ***Monitoring, Data preservation, Affordability, Easiness, Comfort***

For SAS, the actors involved are the patient, the ambulance driver, the ambulance caregiver, and the facility caregiver. The SAS process starts with the patient whose task is to ***Contact Emergency Center***. The main task of the ambulance driver is to ***Define Best Route*** and ***Patient Collection***. The tasks of the caregiver are ***Act on Emergency, Activate Device, Measure Vital Signs and Communicate*** with the facility to describe the health status of the patient. Once the patient reaches the facility, their job is to ***Act on Emergency and Communicate*** with caregiver in the ambulance to provide support , ***Monitor the vital signs and Review Patient Record*** transmitted from the ambulance that allow them to make better triage decisions.
The few research-based remote monitoring systems evaluated by them were Smartstone, the senSave, the Textronic, the EMBIoT health monitoring system, and the SW-SHMS smart healthcare monitoring system. While in the commercial system category Medtronic, Biotronik, TruSense and LivelyWearable.

In case of SAS research-based systems Automatic SAS with smart traffic control, SAS with patient monitoring, E- Ambulance and IoT SAS. In the Commercial system category for SAS SatCare, MobiMed, DCX system Pre-hospital Patient Journal (PPJ), HNCEA and EMBIoT's Lifepak 15. They observed that in REMS, the patient is more involved in his own medical treatment by handling health data, whereas in SAS, the caregiver inside the ambulance has a greater responsibility to provide the patient with the best treatment possible, rather than the onus lying on the patient himself/herself.

Similar to [Bai+14], where they incorporated fuzzy logic with the data collected to detect any irregularities, [Esp+18] also took the same path but suggested an upgraded version which utilizes more concepts of Artificial Intelligence. They ensure that the devices that record the data are easy to wear and do not interfere with the patient's regular working life. The proposed architecture for mobile devices with Android platform and employed the ALMA method (Architecture Level Modifiability Analysis), which provides the system the ability to be simply modified and evolve over time.
For this purpose the data acquisition is performed with the help of an application by using the bracelet of the Amiigo2 fitness tracker, which is capable of sensing 3-axis acceleration, is equipped with a temperature sensor and a reflectance-based pulse oximeter, composed of a light source and a detector, with red and infrared (IR) light-emitting diodes (LEDs). Heart rate and SpO2 are estimated from the pulse oximeter waveform by applying Fast Fourier Transform (FFT) on a Hamming window of the signal. Information of the number of steps are extracted from 3-axis accelerometer data and the intensity of physical activity from the number of steps calculated by using the the concepts of fuzzy variables and fuzzy sets.

This data model utilizes an ontological approach - a high-level ontology and more specific low-level ontologies built on the top of it. A common ontology enables knowledge sharing in an adaptive environment and allows addition of new sensing devices. Ontologies have high and formal expressiveness which combined with their well-defined declarative semantics enable automatic reasoning mechanisms on the represented information. Formalized ontologies enable interoperability among devices and architectural components. The architecture described in the paper focuses mostly on mobility along with health monitoring. The architecture is four tier and each tier provides a set of functionality.

- First tier is the sensing tier which comprises of sensors collecting the data of patients.
- Second tier deals with processing the data derived from the sensors.
- The third tier deals with analysing the information using intelligent components.
- The fourth tier deals with sending alerts and other information to doctors and caregivers so that the health is monitored properly.

While a lot of innovation is seen in the field of healthcare in terms of health monitoring and its benefits, the paper [TSM19] focuses on measuring health-related quality of life (HRQoL) of the patients. HRQol is usually assessed based on patient-reported outcomes (PROs) which is imprecise and does not involve subjects' bio-metric parameters sampling. While the authors aim is to obtain objective data from the patient to back or cross check oral reports of patients; monitor and detect changes in habits to evaluate fragility and patient QoL. In order to infer QoL markers a long-term monitoring is required. To implement

this the architecture needed to fulfill their requirements of being convenient, non-intrusive and affordable. The architecture for biometric parameters sampling system has main three components:

1. "At Home" Management system - this comprises of the configuration module which deals with pairing the device with the android box at home and the management module which deals with monitoring the connected devices.
2. Data center management system - this comprises of communication module which is in charge of monitoring devices, and then the data is acquired from these devices, and this data is processed and used to infer some meaning.
3. Sampling devices - these are the monitoring devices which interact directly with humans and the data is passed over the channel to the central database.

The ***At Home*** section consists of an Android TV Box and is further composed of two modules:

Management module works in the background and is responsible for Bluetooth interfacing with the devices for data collection. It is also in charge of hosting a Web server so as to connect remotely to the system for retrieving the information of interest.

Configuration module on the other hand is for user interfacing that permits the pairing of the monitoring devices with the system which makes it possible to see device status and retrieve information by issuing commands. Both the modules are independent of one another, irrespective of whether configuration module has requested data retrieval or not the management module continues to perform its tasks. For the "At Home" management system the authors considered the Raspberry Pi, the Arduino microcontroller and an Intel based mini-PC other than the AndroidTV boxes. But Android box provided the best ability to integrate and interact with both software and hardware aspects of the technology. Another aspect in favor of Android box was that it was the most efficient in terms of cost effectiveness.

The ***Data center management system*** is an application running on Windows that consists of three modules: **Communication module** is in charge of communication with the monitoring devices and stay updated about whether the Android device is online and reachable to get the system status, the battery level and the last time seen of the devices. **Data acquisition module** is responsible for retrieving and processing data. This data can be visualized on interactive charts like pre- and post-processed data comparisons, put different timespans side by side or compare data of monitoring devices. **Data management module** backups the data to a dedicated PostgreSQL database that gives the opportunity to delete or not delete the files resident on the Android device. The application backs up

the results after each data processing request. A *VPN server* - OpenVPN - has also been set up as a subsidiary part of the architecture as a way to assign a valid IP address that authorizes the Android device to register to the system.

Sampling devices were based on the kind of accuracy required : A Bluetooth low-energy (BLE) activity tracker with embedded ECG and a medical-grade portable Bluetooth ECG. The BLE activity tracker has a low cost nRF52832 chip based CPU, including ARM processor and Bluetooth interface. It integrates a HRS3300 heart rate sensor and an ADS129x series ECG chip. The manufacturer was asked to customize the trackers' firmware that allows two operating modalities: live streaming of data and timed acquisition of data. When the activity tracker is out of range of the Android box, the app gathers the data as soon as the device comes back in range. The tracker can acquire PPG and ECG signals but for PPG signals no user interactivity is required. For ECG data, one the the electric poles is on the back which is in contact with the individual's skin and the other one is top of the bracelet body. Therefore, it is necessary for the user to touch the tracker's case with the finger of the opposite hand as a means to acquire the ECG signal -whether in live mode or timed acquisition - before starting an ECG acquisition which is indicated by a vibrating alarm on the device. For a more precise measurement or when the application reports a specific alert, a second device is used - a medical-graded eight-channel portable Bluetooth ECG. Although it is relatively invasive due to the presence of skin electrodes and the cable system.
The whole system is solely intended for monitoring usage only and more importantly, acquiring a large quantity of qualitative data instead of a small quantity of accurate data is what the authors are interested in. Since the system involves the use of low-cost devices, that produce very noisy signals which are then processed with low-performance systems like Android boxes, it was important to find an algorithm that is a trade-off between easiness and processing efficiency. The ECG signal being more prone to noise and was in huge quantities, so the authors focused on creating an algorithm capable of discarding some parts of the ECG track instead of filtering the entire track which can procure a continuous reliable ECG track. Another necessity was to have the capability to automatically discriminate high-signal/noise-ratio segments to reduce the truncate the data. These operations would not affect the accuracy of the data, considering the purpose and the scope of their research, the QoL measurement.
The algorithm, which is used only on ECG signals, consists of two phases: ***Filtering phase*** involves dual filtering of the signal. The raw signal is passed through a high-pass filter so as to expel the baseline oscillations. A second-order filter obtained from a cascade of the combination of two of this filter is implemented. Similarly, a second-order filter obtained from a cascade of a low-pass filter, is applied for removing the movement artefacts present in the signal. This also helps in minimizing the computational impact. ***Template recognition phase*** entails creating a template - based on literature and sampling rate of

the device- of an ECG curve. An ECG curve is composed of four components:

- P wave
- QRS
- T wave
- U wave

The authors decided to decompose the ECG wave into its fragmented waves to create a template. Each wave was approximated to a polynomial function of different orders using polynomial linear regression method which was beneficial in keeping the computational resources to minimum while maintaining a high processing time/quality ratio. They also defined a tolerance range, for avoiding the presence of a track section that contains curved peaks which is present because of some artefacts that might not have been filtered.
The obtained data is then displayed by overlapping two curves composing the post-processed data curve. A green curve shows the template fitting section, and a red one displays the sections where the trend does not fit the template. The visualization allows to jump, zoom and scroll between the red sections showing a customizable number of seconds after and before the red section. It possible for the physician to evaluate the steps data coming from the activity tracker placed side by side to make it convenient to discriminate whether the occurred episode is due to a warning situation or not. Standard techniques to estimate Quality of life variations (QoLV) involves sleeping fidgety, physical movements and resting time. The information about the sleeping timespans coming from the accelerometer, can indicate a fidget sleep or not. Fatigue can be inferred if a habit of walking is producing a high heart rate, it indicates a worsening trend when subject undertakes the basic activity of daily living. It provides the caregivers access to information that oral reports cannot, rather they can obtain objective data.

1.2 Stakeholders

The project involves multiple entities, organisations and people. These components are briefly discussed below to provide the an introduction to their working.

1.2.1 RSA

The RSA stands for Residenza Sanitaria Assistenziale (sanitary residence care). It is a non-hospital home for a variable period for dependent persons, who cannot be cared for

at home and need special medical care. It is different from a hospital in the sense that it is not aimed at people suffering from acute illness or patients in critical condition. It also differs from a nursing home where fully or partially self-sufficient elderly are hosted.

The RSA must provide guests:

- *residential home-like accommodation, that provides laundry, cleaning services*
- *organizing socialization activities among guests*
- *all medical interventions, nursing and rehabilitation necessary to prevent and treat chronic diseases and in case of emergency situations*
- *a personalized assistance, equipped for the protection and improvement of the levels of autonomy*
- *maintenance of personal interests and the promotion of wellness*

The essential staff of the RSA are:

- the medical director - coordinator of the facility
- the general practitioner - provides medical care provider
- the nurse
- the home carer
- physical, occupational therapists

Additional health care professionals like physician, geriatrician, psychologist can be made available by the local ASL. The basic unit of RSA is composed for the reliant elderly and for people with physical, mental and sensory disability. Some RSA have a unit for Alzheimer patients which is a dedicated section to persons with cognitive and behaviour disorders [Mer15].

This project is in collaboration with two RSA based in Genova : Residenza per Anziani Cardinal Minoretti and Residenza per anziani Valpolcevera. Residenza Valpolcevera comprises of two independent sections, one dedicated to Protected Residence and the one active as a Healthcare Assistance Residence, each with their own allocated staff. The rooms can accommodate two or three patients in a room, with indoor and outdoor activity rooms. Residenza Minoretti consists of a Rehabilitation Unit, two Alzheimer's Nuclei in a prosthetic environment and they also offer outpatient activities in their Multi-Specialty Center. There were 10 patients from each RSA that were assigned to the project.

1.2.2 Technical Team

- Hassisto
  Hassisto Srl, is a CNR (National Research Council of Italy) spin-off which develops innovative software platforms for automated systems in e-health monitoring. Technological advancement and human dependence on technology, jointly have lead to rise of multiple internet devices capable of interacting with one another, in every household. Hassisto's idea is to use such devices, that monitor vital and environmental parameters, to provide health assistance in domestic environment. The sensors in the devices collect data while the person goes about with their day to day activities. Hassisto aims to help enhance the current system by allowing continuous routine checkup in an automated manner thereby reducing costs of of indoor healthcare management.
  The platform employs Bluetooth devices: medical devices, professional devices like ECG and other wearable sensor equipped devices useful to get a complete and clear vision of health parameters 24h / 7 `https://www.hassisto.com`. The devices are linked with software that monitors heartbeats, blood pressure, SPO2, glucose, inactivity, quality of sleep, falls which can be integrated with additional bluetooth devices to get more information like weight, enuresis, rehabilitation exercises.
  This project was being coordinated by Egr. P.G. Meo of Hassisto. The details about the platform and details about the equipment were provided by him.

- **Vega Research Laboratories** Vega Research Laboratories is an innovative start-up company, created by the Computational Intelligence Research Group of the University of Genoa. It is founded by Prof. Francesco Masulli, Prof. Stefano Rovetta and Prof. Alberto Cabri. It is involved in the research and development of innovative products and services of high technological value. As displayed in their website `http://www.vegaresearchlabs.com/`, their primary focus is research and development of algorithms and systems for leading edge applications, for a wide variety of industrial sectors and markets. VRLabs was the team in Genoa heading the project and were responsible for supervising the AI and ICT issues.

1.2.3 Medical Team

- Prof. Ernesto Palummeri was the medical supervisor during the project.

- Ms. Ilaria Nolasco was the representative of RSA Minoretti for all interactions.

- Mr. Ubaldo Borchi was responsible for all the undertakings in the course of this project with RSA Valpolcevra

Chapter 2

IOT devices and Technology

The project was highly dependent on the hardware components, which were the medical IOT devices. They were primary sources for getting the data of the users and also to collect and transmit these records to the server. The devices used are introduced in detail in the sections below. There were 3 main equipment used during the project which were provided by Hassisto: smartwatch, sleeping band and android box; discussed in the next sections.

2.1 The Smartwatch

The spovan h03 wristband, unlike other smartwatches is primarily aimed at recording health data. It is an advanced fitness and health tracker that has on-wrist ECG, blood oxygen, blood pressure, heart rate, HRV and sleep monitoring functions. It is touch key controlled and not touch screen which avoids accidental touch actions and is also waterproof. It uses the Nordic 52832 platform, Si1182 ECG sensor and electrode on the watch body, and uses both photoplethysmography and electrocardiography method to detect vitals information like heart rate signals [SPO19]. Although, it is not like another general smartwatch, it does provide the option to get notifications from other apps like messages or set sedentary reminders.

Figure 2.1: Smart watch

The watch is equipped with a 1.14 inch HD screen and 150mAh rechargeable battery, which can work upto 5 days after being fully charged since it uses low power bluetooth. The straps of the watch are detachable and can be removed by pressing the buttons at the base of the dial on the inside of the strap. One of the end, identifiable by the metallic part, acts as the charging pin. It needs to be inserted directly into the adapter where usually a USB cable would go. These detachable straps allows the smartwatch to provide additional feature of taking ECG, using electrodes provided with it, with the body of the same bracelet to make it easier and lesss tedious.

Figure 2.2: Watch sensors and charging

The smartwatch can be turned into ECG monitoring device in multiple ways. The watch itself can also take ECG on the wrist by making sure the sensors underneath the watch are in proper contact with the wrist. Although it is probably less accurate. The watch comes with a pair of electrode sheet and ECG connectors. If the monitoring needs to be more accurate and for a longer duration the electrode sheets can be used which attach to the ECG connectors. Another option is to get a chest strap which would be attached to the ECG connectors and can be worn around the body. The ECG connectors are also capable to take ECG by holding the connectors from the metallic part.

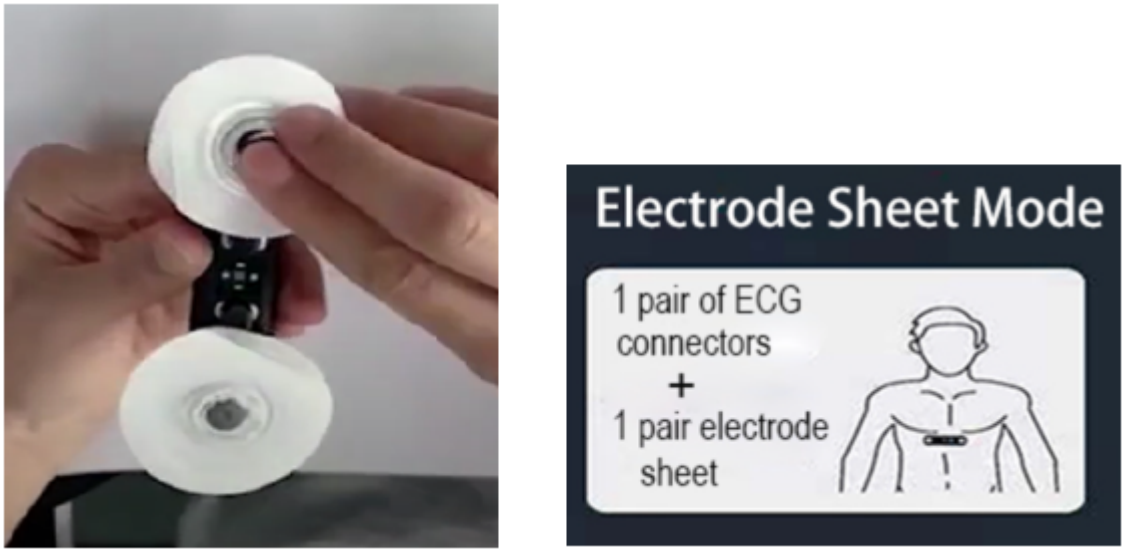
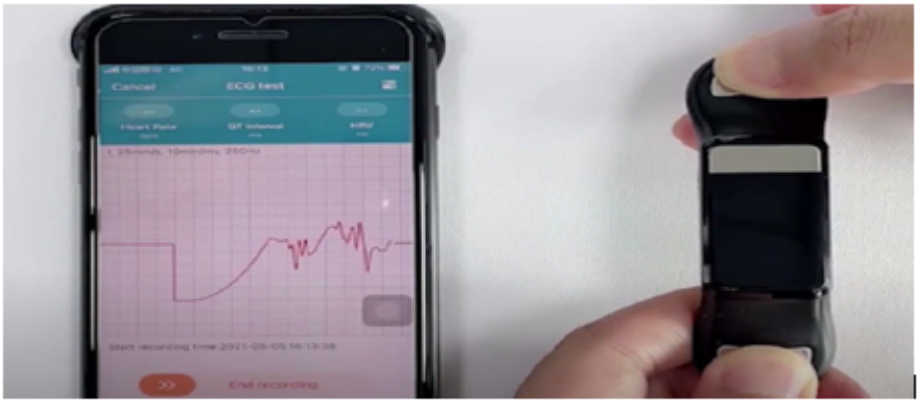


Figure 2.3: Electrode sheet

Any of the options can be used depending on the situation and the requirement. The electrode sheets are for single use while the ECG connectors and the chest strap are reusable. The electrode sheet and the belt require attaching the device to the body, while the ECG connectors are hand-held. After connecting any of the above ECG devices the user needs to connect to the app. Electrode sheet and belt are more precise, however, for routine checkup the ECG connector can be directly used. Long press on the quick ECG monitor option on the watch would lead to an icon appearing on the dashboard in the app, clicking on which would take the user to live monitor.

The user just needs to hold the connectors at the metallic part to see a real time (with a delay of few seconds) ECG.



The data can be uploaded to the server using the Gateway Hassisto app.

Figure 2.4: ECG Connectors

2.2 Sleeping Band

The sleeping band is a non-invasive belt, which needs to be positioned on the mattress and can detect the period of sleep, breath and heart beats. It is made with PC + ABS + UV painting material and works on polyvinylidene fluoride (PVDF) sensors that detect pressure values. It needs to be spread on the bed under the bed-sheet in a way that it lies under the user's rib cage. It monitors the subject's breathing while sleeping by the movement of the chest while inhaling and exhaling that causes change in pressure. It can also identify movements to alert the caregiver in case of sudden chaotic movement, for example in case of a seizure, and sleep apnea the thresholds for which can be customized. It can also be connected to "HappySleep" app to see the real time data. But it needs to be connected to the internet box as a means to send the data. Since it uses unique BLE channel it can be connected to only one of the above mentioned at any single moment.

Figure 2.5: Sleep band

2.3 Android Box

The Android box is responsible for providing Bluetooth interface with the monitoring devices and hosting of an HTTP Web server, which allows to connect remotely to the system for retrieving the collected data[TSM19]. It needs to be connected to the internet either by LAN or Wifi. In order to connect to the sleeping band it should be in the vicinity

of the band, so connecting it through Wifi could be a better option. It comes with an HDMI cable for the purpose of configuring the settings to connect it to the Wifi.

Figure 2.6: TV box

2.4 Apps for Devices

The smartwatch can be controlled by the ***"H Band"*** app. It allows the user to monitor and visualize the data in real time. A detailed, but generalised, explanation is provided for each feature and it's several aspects that were considered while evaluating the data. It also provides definition of some medical terminology used to get a very basic acquaintance with the concept.The user can set alarms, reminders and even private Blood pressure mode depending on the health condition. The dial screen can also be changed using the app. The "H Band" app allows the user to take ECG by using the electrodes provided with the watch.

Figure 2.7: H Band Interface

"Hassisto Gateway" app was developed to sync the TV box, which is composed of two modules:

- **Management module**: Background service that manages Bluetooth interfacing and web services hosting for collecting data
- **Configuration module**: For user interfacing to allow the pairing of the monitoring devices with the system which makes it possible to issue commands and retrieve devices' status

Figure 2.8: Hassisto Gateway Interface

The equipment was provided to the team in Genoa a few weeks prior to installation in the RSA to get acquainted with the working and other features of the equipment. Egr. P.G. Meo of Hassito were always in contact for any queries and problems encountered by the team. The smartwatch and the sleeping band were provided as sets and could not be replaced with another. They were connected so as to make sure the data being collected

is of the same person. To avoid confusion and make identification easier they were colour coded. The android box works on BLE technology and hence is required to be in the proximity of the sleeping band so that it can read the data. The smartwatch on the other hand stores data during the day and it can be monitored on the "H Band" app. For sending this data to the server "Hassisto Gateway" app is used. It can be done only when the android box is in the vicinity of the phone on which the app is installed. It can also be done by using a monitor or android TV and connecting that to the android box.

2.5 Installation of the Technology

The project was in coordination with two RSA based in Genova, RSA Minoretti and RSA Valpolcevra. The team in Genova made visits at the RSA for meetings with the staff and the doctors. Prof. Ernesto Palummeri who were the medical supervisor on this project aslo accompanied the team for the meetings. The RSA staff were introduced to the technology and the features it supported. They were made aware of the general information about the equipment like the charging method, battery life and how to check the status of the battery of the wristband. The basic working of the wristband and the possibility of taking ECG with it was also demonstrated. The team also surveyed the surroundings of the building and activity rooms. It helped in deciding where the TV boxes needed to be installed with a view to cover as much area as possible for monitoring.

The equipment was sent by Hassisto as kits which were color coded. The patient had to wear the same wristband and the sleep band since they were being identified by the data recorded by bands. To ensure the bands do not get exchanged among patients they were color marked. This equipment had to be divided into two groups for installation at each RSA according to the colour coded scheme for each patient. A document was made to keep track of patient IDs and the colour codes of the equipment for each patient at each RSA. To maintain anonymity of the patients, only patient IDs were used for the project. Although the RSA had the information of which patient had what ID.

On the day of the installation, the team in Genova was accompanied by Egr P.G. Meo of Hassisto to the RSA. The team went to both RSA Minoretti and RSA Valpolcevra for installation of the equipment. The teams at the RSA were given a demonstration about the use and working of the equipment. One installation in each of the RSA was exhibited to the technical personnel and the staff there. Due to Covid-19 restrictions the rest of the installations were performed by the technical personnel there. The rooms for patients in the RSA accommodated two patients and had one TV. This allowed us to use only one android box for a single room. The sleep bands required the TV boxes to be in the

vicinity so as to function properly. The android box were connected to the TVs in the room since using apps on the phone for multiple patients was not possible. The TV box has an application Gateway Hassisto to sync the devices to the TV box which can be done through any monitor or as in our case any Android TV. The staff at the RSA were provided access to a dashboard, provided by Hassisto, where they can access and add information about patients. The action plan was discussed with the teams at RSA and their suggestions were also noted.

Chapter 3

Data Analysis and ML Approaches

The data collected by the devices needed to be analyzed to extract essential information from it. This allowed us to understand the data and what algorithms can be applied to it to not only draw out meaningful details but also provide sufficient knowledge to make some inferences.

3.1 Data Analysis

The data recorded by the smartwatch and the band can be seen in the respective apps but for detailed analysis and visualization the collected data can be accessed in two ways. A platform composed of a dashboard where you can manage patients, devices and measurements was provided by Hassisto. It can be accessed by authorized personnel at `https://dibris.hassisto.com`.

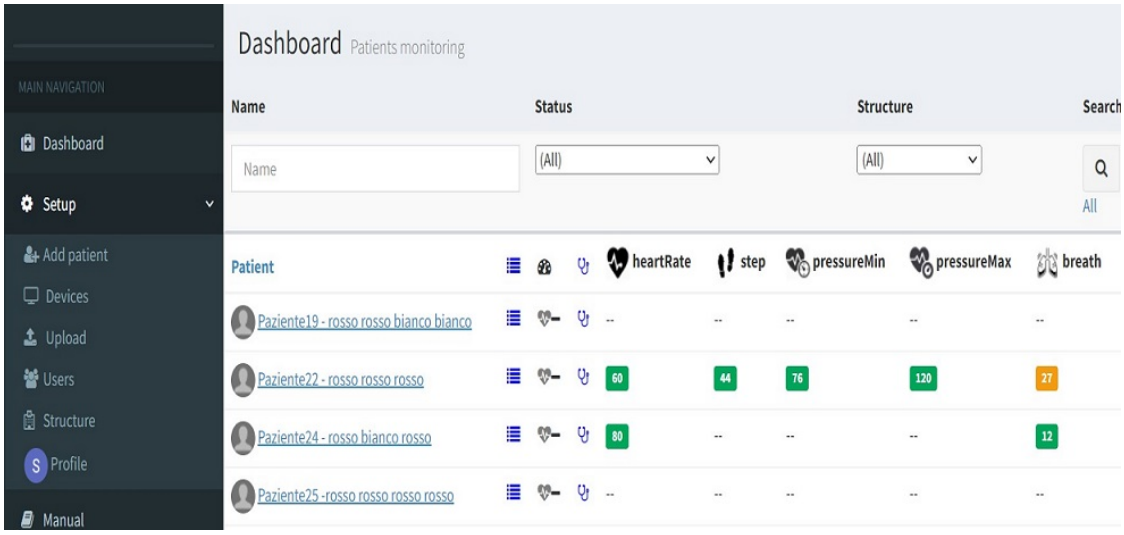


Figure 3.1: TV box

The dashboard allows access to information of each patient's data and device information like battery life. It is primarily aimed for the staff at the RSA and hence has the feature of adding extra medical information about the patient. This allows the possibility to compare measurements calculated by the senors and the medical equipment of the staff. Only authorized personnel can access the dashboard or add/remove information about the patients. A second portal to import the measurements for statistical analysis and exporting results was provided, which is the primary source for data retrieval during the project. The portal can be accessed at `https://api.hassisto.com/swagger`

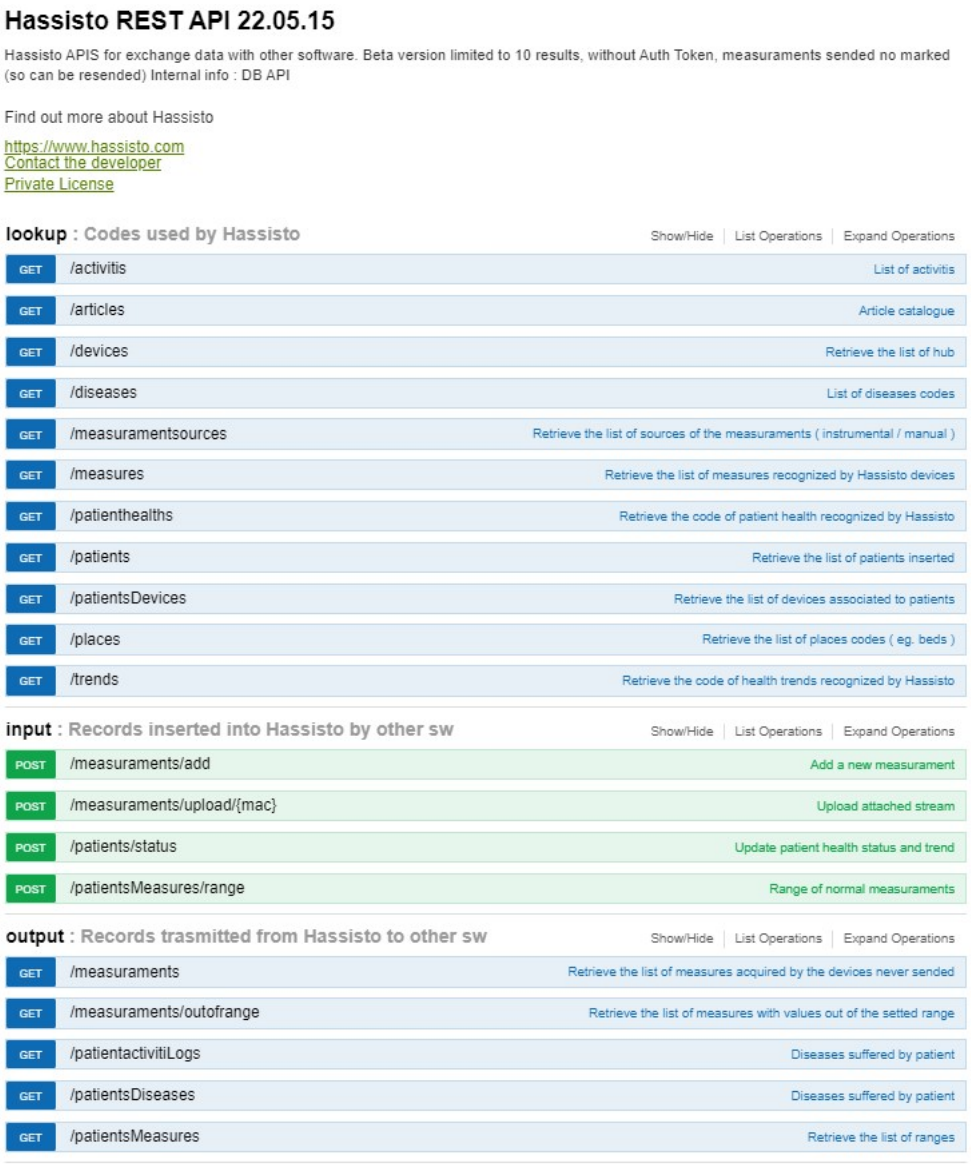


Figure 3.2: Hassisto API

The API has has different sections for get and post requests. Several entities were involved in the project and have been coded in different ways, like device and diseases names, IDs of patients and sources of measurements- devices or staff. *lookup* allows to get access to codes for several categories. *output* section caters to get requests for records of data read by the devices. *input* is the only post requests option like updating patient's health status. During the project, primary work was done on data recorded by the devices. The records can be accessed using specific fields like patient code, time period or name of the measure.

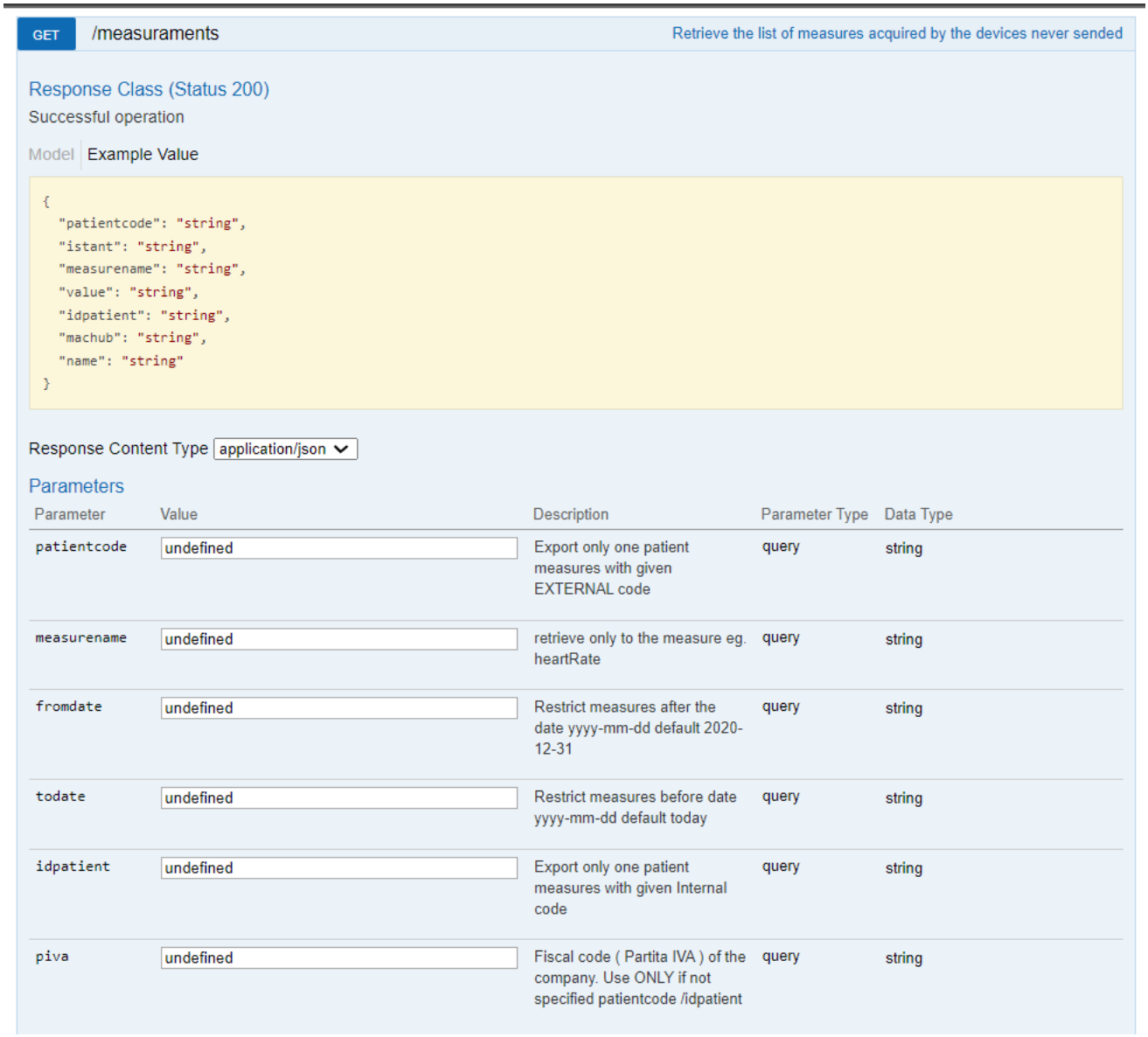


Figure 3.3: Hassisto API for get measure data

The data from the API was accessed using Python's Requests library since it is the simplest way to make HTTP requests. To get an overview of the retrieved data Python's Pandas library was used which is an open source data analysis and manipulation tool. The pandas dataframe made it easier to visualize the structure of the data. The dataframe consisted of the following columns: *"patientcode"* - which gives the patient's code and is also the index; *"istant"* - the timestamp of the instant the value was recorded; *"measurename"* - the name of the vital being recorded; *"value"* - the value of the corresponding

measure; *"idpatient"* and *"measurementsource_id"* give the ID allotted to the patient and the source of the measurement respectively. The vital health records available during the project primarily consisted of heartrate, breath, pressureMin- which is diastolic pressure, pressureMax- which is systolic pressure, saturation(SPO2) and steps. There were other readings of temperature, weight, sleep duration along with some activity data like inBed, motion etc. For the project, data of each measure and from each patient needed to be evaluated.

	patientcode	istant	measurename	value	idpatient	measuramentsource_id
0	Paziente02	2022-10-02T23:40:00+02:00	step	13	PazienteRSA	6
1	Paziente02	2022-10-03T00:00:00+02:00	pressureMax	114	PazienteRSA	6
2	Paziente02	2022-10-03T00:00:00+02:00	pressureMin	85	PazienteRSA	6
3	Paziente02	2022-10-03T00:01:00+02:00	heartRate	79	PazienteRSA	6
4	Paziente02	2022-10-03T00:02:00+02:00	heartRate	75	PazienteRSA	6
...	...	...	...	...	...	...
22513	Paziente24	2022-10-03T09:25:14+02:00	inBed	1	PazienteRSA	6
22514	Paziente24	2022-10-03T09:31:53+02:00	inBed	1	PazienteRSA	6
22515	Paziente24	2022-10-03T09:50:39+02:00	inBed	1	PazienteRSA	6
22516	Paziente24	2022-10-03T09:50:39+02:00	motion	1	PazienteRSA	6
22517	Paziente24	2022-10-03T09:50:41+02:00	inBed	1	PazienteRSA	6

Figure 3.4: Get measures dataframe

The view of the dataframe needed to be changed see the rows of the "measurename" column. Pandas dataframe's "pivot_table" was used, that allowed the ability to change the view of the dataframe and keep only the necessary columns. Another requirement was to get patient's daily or hourly data so the timestamp format of the "istant" column was split to date and time according to requirement during the data pre-processing steps of different tasks. Pandas dataframes also allowed to create separate dataframes which fulfilled specific conditions. It also enabled us to remove the data that was insufficient to make useful observations and so the primary focus was kept on six main measures of vitals.

measurename	date	breath	heartRate	pressureMax	pressureMin	spo2	step
patientcode							
Paziente04	2022-08-22	5.000000	74.000000	122.357487	78.924369	97.184140	76.198892
Paziente04	2022-08-24	5.600000	75.400000	122.357487	78.924369	97.184140	76.198892
Paziente04	2022-08-25	13.333333	72.666667	122.357487	78.924369	97.184140	76.198892
Paziente04	2022-09-14	13.333333	72.915133	120.411168	80.177665	97.184140	114.347826
Paziente04	2022-09-15	13.333333	76.650463	119.965517	80.224138	97.250000	132.136364
Paziente06	2022-09-15	14.000000	79.000000	119.965517	80.224138	97.250000	132.136364
Paziente04	2022-09-16	18.826087	73.022427	120.331081	80.689189	97.250000	70.709677
Paziente06	2022-09-16	12.000000	73.000000	120.331081	80.689189	97.250000	70.709677
Paziente04	2022-09-17	18.777778	74.005727	120.398844	80.849711	97.250000	128.578125
Paziente04	2022-09-18	25.500000	69.703704	121.181818	82.636364	97.000000	29.571429
Paziente02	2022-09-19	25.500000	87.543568	120.469388	80.265306	97.000000	28.700000
Paziente04	2022-09-19	20.064516	64.612903	120.469388	80.265306	97.000000	28.700000
Paziente02	2022-09-20	20.064516	84.560440	119.921053	81.210526	97.000000	41.000000
Paziente02	2022-09-21	20.064516	69.896774	120.098039	80.534653	97.342308	22.909091
Paziente04	2022-09-21	19.444444	65.277778	120.098039	80.534653	97.342308	22.909091
Paziente06	2022-09-21	11.000000	91.000000	120.098039	80.534653	97.342308	22.909091
Paziente04	2022-09-22	18.400000	69.900000	120.098039	80.534653	97.342308	22.909091

Figure 3.5: Manipulated dataframe

The figure above shows the data of each patient for each measure averaged over the date with patient code and date. The index columns were reset using pandas to bring the entire dataframe on the same level. Also for further computations the values of the readings were rounded to two decimal places. A similar dataframe for the hourly was also created for different usecase.

3.2 Patient Record Visualization

One of the requisites of the RSA personnel was to have the ability to view patient profiles on the basis of week, day or hour. The watch and the band read data for a particular measure after a defined interval, so each measure follows a different time series. To find patterns in time series data can be difficult by looking at the data, so visualization methods were considered to make relevant information easily accessible and noticeable for swift detection of abnormalities. The staff at the RSA need information about each patient and their vital

signs, but also an overall picture of the patients, especially patients with similar health conditions. With this in mind Radar plots were chosen as the means of visualization of the records of measures.

Radar plots were used as they are an informative way for visualizing data with multiple variables (three or more) and compare them on a two-dimensional plane.

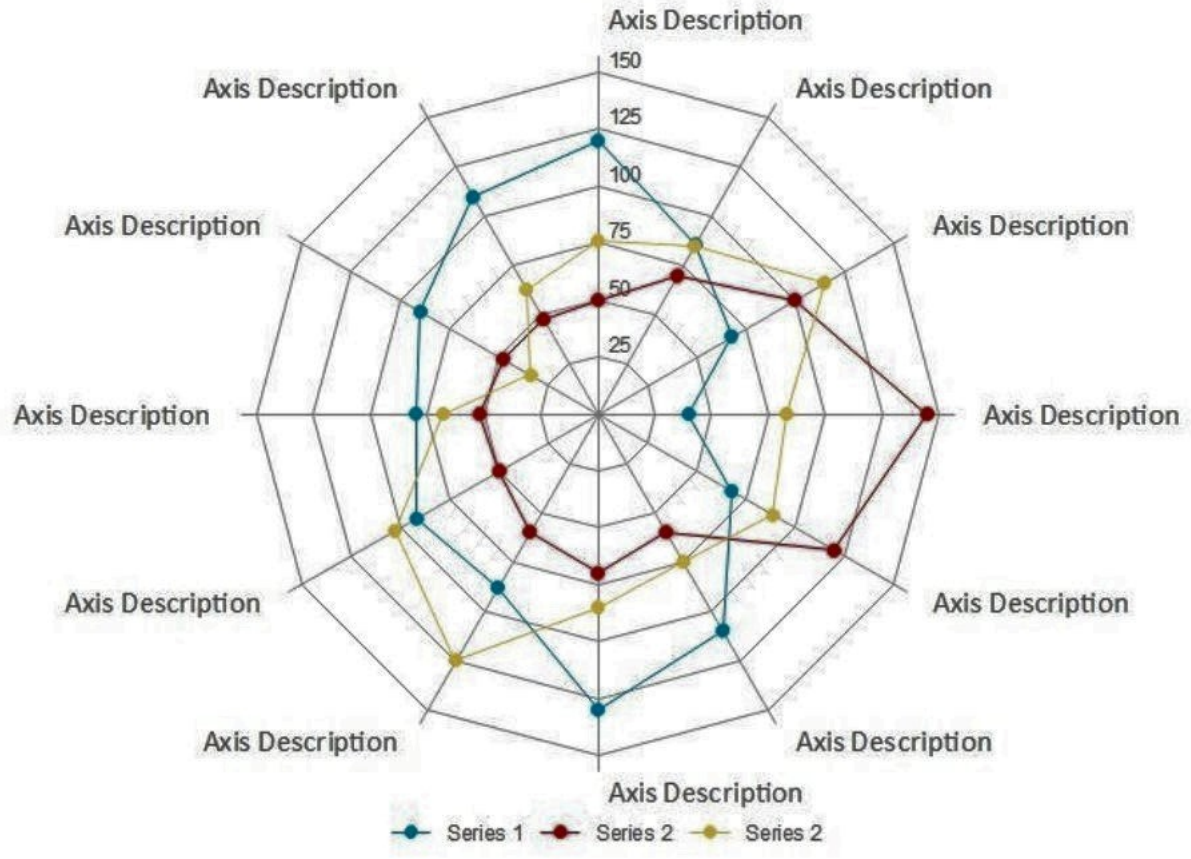


Figure 3.6: Example of Radar Chart for multiple variables

It stacks the plots on the same central point using multiple shades which highlight the contrast. Since the requirement of the RSA includes data for each patient and for a population of patients, multiple versions of radar plots were created. For an individual patient profile, all the measures were required to be visualized at the same time in one plot. But the measures had different ranges of values, steps were mostly in thousands while saturation(SPO2) in percentage primarily above 80. To visualize these measures on the same plot, the axis needed to have its own scale varying according to range of values for that measure.

For visualizing data for an individual patient, during specific activity hours, each measure needed to be analyzed independently. In this case since the data was of the same range of values for each axis, a single scaled radar plot was created. Each radar plot had different requirements, so the technology used to create the radar plots was also required to be flexible and customizable. We first considered python's *Dash* and *Plotly* libraries since the data analysis was already being carried out in python Pandas. But they did not offer enough flexibility to create a radar plot according to our requirements. Python's *Matplotlib* was the next choice, which gave the option to create a radar plot method from scratch. But the visualizations were mostly static in nature and the library did not provide any visually enhancing features. The capability offered by Dash and Plotly to create Web-based visulizations and still make a customized method was the eventual requirement which were fulfilled by the *D3* library of Javascript. The integration with python was also comfortable and efficient. Most importantly it gave us the freedom to have multiple ways of visualization and a huge scope to make alterations in an uncomplicated manner for further future advancement.

3.3 ML Approaches

The requirement of the RSA personnel was to have the means to get a general overview of the health of the patients. Finding clusters among the patients according to their profiles such that it could give a general estimate of what health group a patient falls in, considering the vitals during a certain period, was the route chosen for fulfilling this requirement. The underlying idea was that this might be helpful in categorizing patients according to their health status. For data analysis variants of clustering methods were used to find groups among the patients who show similar symptoms. The number of patients during this project were few, so finding precise clusters was less probable, but would provide us the opportunity to evaluate multiple clustering algorithms to see how they perform.

Another idea was to analyze the trends of the biometric parameters of individual patient for each observation scale of the profiles. To perform trend analysis the strategies used in [Abd+16] were utilized. In their work they have used graded possibilistic clustering approach that can iteratively track outliers and adapt to concept shifts and drifts in non-stationary data. The algorithms used are briefly explained in the next section.

3.3.1 Clustering Algorithms

Clustering is an unsupervised method of machine learning, where "unsupervised" implies that it can understand patterns in the data without target labels. Clustering tries to find categories among a set of objects, or people in our case, that share some key characteristics.

The aim is to to uncover meaningful groups in our data. Clustering algorithms can fall into any of the two categories - Hard Clustering and Soft Clustering. Hard clustering is a stricter version of creating clusters- that each item belongs to one cluster completely or not at all. On the other hand, soft clustering is about grouping the data points in such a manner that any individual point can exist in multiple clusters. The outcome provides a probability of likelihood of a particular data point to be in those clusters. In hard-clustering algorithms, the membership vector is binary, the point is in a cluster or not. For soft clustering algorithms, a fuzziness coefficient- which controls the degree of fuzziness or imprecision - is estimated. Soft clustering is ideal in use cases where an item can be in multiple states or to measure homogeneity of an item to a number of groups.

- Fuzzy Clustering - Fuzzy C-means [Dun73],[BEF84] is a soft clustering algorithm. It is based on fuzzy logic and commonly called FCM algorithm. Fuzzy Logic is a methodology that works on the idea that the exactness of anything can be expressed a range of things which are minutely different from one another. In other words it implies that something is not just true or false but instead partially true or partially false. It works by assigning probabilities to data items, which basically indicate the strength of the association of the item in a particular cluster. This association is expressed in terms of a membership vector, which denotes the probability of the data item to be a member of a cluster. The vector contains values ranging from 0 to 1, indicating the similarity of an item to the mean of the cluster.

- K-means - K-means [Mac67],[Llo82] is a type of partitioning clustering method that uses an iterative algorithm in which the data points are grouped into *'k'* number of distinct clusters, based upon the distance metric used for the clustering. The value of 'k' is specified by the user. The sum of squared distance between the data points and the centroids of the clusters is computed. The data point with minimum value or in other words which is closest to the centroid of the cluster is allocated that cluster. Cluster's centroid is the arithmetic mean of the data points in the cluster and is computed after each iteration until the number of iterations are completed or there is no substantial change in centroids of the clusters.

- K-medoids - It is also called Partitioning Around Medoids algorithm [KR90]. In this algorithm, medoids are chosen as centers of the cluster and it has to be an input data point, unlike K-means where centroid is the average, therefore may not be an actual input data point. While K-Means clustering aims at minimizing the intra-cluster distance, K-Medoid functions by reducing dissimilarities between points of a cluster and the centers of that cluster. Dissimilarity is calculated as the summation of the absolute difference between medoids and data points. K-Means clustering only uses one distance metric- Euclidean distance, but k-medoids can use any distance metric, could be euclidean, manhattan, etc.

- GPCM - Graded Possibilistic Clustering Model is a partially possibilistc version of the fuzzy clustering model introduced in [MR03]. It is based on the idea of possibilistic approach to clustering [KK93] which considers that all membership values are mutually independent. In FCM, the membership of a point in a class is relative to the membership of that point in all other classes. And since memberships are probabilistic, a point has to have some membership in at least one class. This limitation does not allow the algorithm to distinguish between a moderately atypical point and an extremely atypical point. In possibilistic approach they do not put any constraint on the sum of the memberships, i.e. the sum of memberships do not equate to 1, making them independent of the number of classes. The only constraint is that each membership should be in the range [0,1]. Graded possibilistic model assumes that, when one of the membership values is fixed, the other $c-1$ values are constrained into a given interval contained in $[0,1]$[MR03].
  In GPCM, rather than using sum of memberships, total membership mass of an observation is evaluated for partitioning. It also allows to vary values of parameter 'alpha' that permits the algorithm to toggle between probabilistic and possibilistic models. The best value of alpha can be evaluated using *a priori* information according to the dataset to get optimal results by using a mixed version of both probabilistic and possibilistic model and get low error values. The parameter variation also makes the algorithm capable of outlier detection and rejection, by selecting a membership threshold, different for each cluster, which helps in deciding whether a point falls in any of the clusters or not. It helps in not only making the algorithm more robust by removing outliers but can also be used for evaluating outliers and detecting changes in the normal trend of the data.

K value - The ideal number of clusters can be evaluated using the **elbow method**, which involves running the algorithm for a range of clusters creating a plot with the number of clusters on the x-axis and the total within sum of squares on the y-axis. As the number of clusters increases, the differences between clusters gets smaller while the difference between observations in the clusters increases. The idea is to find a trade-off between achieving most homogeneity among the samples and the clusters as diverse as possible. The point on the x-axis where the "elbow" or a bend occurs gives the optimal number of clusters to use.

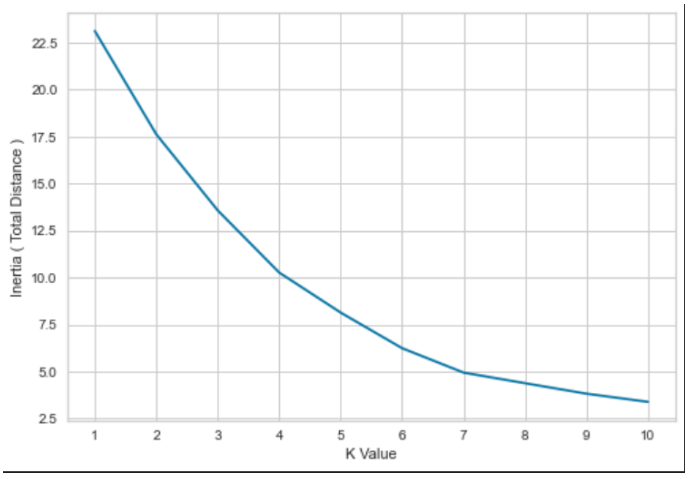


Figure 3.7: Result of using Elbow method with K-means

But it was not able to give a good result when used with a subset of our data. Since the algorithms used during this project varied from hard to soft clustering, choosing a K value with elbow method did not seem beneficial. For most of the part we decided to choose $K = 3$, because our data does not contain sufficient number of observations or categories to get good results for more number of clusters. In most cases we look at only one patient's data and the possibility of existence of many distinct clusters in those samples is very low.

3.3.2 Trend Analysis Parameters

The algorithm in [Abd+16] is customizable to fall in the categories of probabilistic and possibilistic depending on the values of the parameters. For detecting outliers the parameters need to be changed differently as compared to when you want robustness and want to reject outliers. The parameters are discussed below:

- **alpha(α)** - it helps to modulate the behaviour of the algorithm from probabilistic($\alpha = 1$) to possibilistic($\alpha = 0$).
- **beta(β)** - it represents the width of the clusters
- **omega(Ω)** - it is the outlierness index
- **rho(ρ)** - it represents the outlier density
- **theta(θ)** - it controls the learning regime; between stability ($\theta \approx 0$, model stays the same) and plasticity ($\theta \approx 1$, model changes completely)

- **eta(η)** - it is the updating coefficient, to update the centroids, default initial value is 1

These parameters are updated iteratively during the algorithm.

In our analysis, we wanted to detect outliers in the data, which would imply that the patient's biometric variables are exhibiting changes in values that are not in their typical range. It will help in recognizing any irregularities in the patient's health. The outlierness of a data point can be detected using following measures:

- **Outlierness index** - Degree of outlierness is evaluated using the properties of possibilistic memberships. Outlierness was measured by computing the total mass of membership to clusters and measure how short it falls of 1. Extremely low values characterizes regions not in the reasonable vicinity of the centroids, therefore any observation appearing there is an outlier. The index Ω is the complement to one, of outlierness such that it neglects negative values.
- **Outlier Density** - Outlierness index is a point-wise measure; a point is labelled as outlier when Ω exceeds some threshold. But to evaluate data as a whole, the proportion of outliers needs to be considered. Since Ω is a fuzzy concept measuring outlier density(ρ) gives better insight as it accounts for both frequency and intensity of outliers.

Since the data is not stationary and shows varying behaviour, different learning regimes are necessary for the working of the algorithm. In the situation of quasi stationary data, that is, no outliers, the model can work with same incrementally updating regime of parameters and will give good results. This is the *baseline learning regime.*
Isolated changes in data means outliers, so change in learning technique should be minimal or none at all (*no-learning regime*), otherwise the model starts learning from outliers and performs badly.
Abrupt change in the data means the data stream has changed characteristics and the model needs to be updated, called *re-learning regime.*
Theta θ controls learning regime and subsequently beta or width of clusters. In the implementation of the no-learning regime, the value of θ is close to zero. Regulating values of η, which controls the intensity of updates, leads to stochastic iterative updates which gives an averaging effect. After initialization, magnitude of updates depends on the degree of outlierness of the current observation. Increase in outlier density indicates there is shift in the trend of the values. The higher the value of rho (ρ) the bigger the difference in the values of the trend.

Chapter 4

Development

For the development of the project it was decided to use Python. It is the current standard for machine learning tasks. Also the pre-requisite tasks on the data, like cleaning and applying the algorithms; and the techniques for further visual representation of results required an adequate library of functions. Python provided access to abundant and deft modules. During the project multiple technologies were used for performing analysis and algorithms on the data. From retrieving data to cleaning and then visualizing it, several python and JavaScript files were created and integrated for the purpose of making their interaction smooth. Different libraries and techniques used are discussed in their respective sections.

4.1 Work on Data

Since health monitoring is a continuous process, the data recorded also gets constantly added to the server. To retrieve data, a fresh request is sent to server each time to get the current data. More importantly, the data was of real patients so storing it in a database was not opted. The data is called through the API using *get_measurements_request* method for each task so that the data stays consistent and updated for each task. It allows access to current data, however, it makes it necessary to add data pre-processing steps for every function call.

4.1.1 API operations script

To retrieve the data for statistical analysis and other Machine learning algorithms a python class named *apiOperations* was created using python's "requests" module. It consists of two

methods, *get_measurements_request*: which uses the "get" method to retrieve the records. The request url was created using the "base path" to the API, the "path" which defines the category of data requested and the parameters of that category. The parameters are input to the method in the same format as provided in the API. It returns the data as json.The other method, *post_measurements_request*: allows us to post or send data, like health status of the patient or if new measurements category or devices are added to the monitoring system. The data retrieved can be accessed as json data but for analysis purposes *Pandas dataframes* were used which provide more options to get a descriptive look at the data. Dataframes allowed the flexibility to view the data for each specific measure and to manipulate data formats according to requirement.

4.1.2 Data cleaning

The data retrieved through the API contained a lot of information about the vitals collected by the devices. It also consisted of details about the source of the data - wrist band, sleeping band or the doctors. There is also data related to the activities done by the patients at a certain point of time. For the current project our focus was primarily on the data of the measures and to look for patterns that could help provide information to the staff at the RSA. For this reason data columns of such information was dropped. We needed to look at the data of each measure while the raw data had just one column named "measurename" which specified the name of the measure recorded and the corresponding values were in another column named "value". The data table was required to be reshaped according to the "measurename" column in order to have each measure's time series data. The time field - "istant"- was also a time stamp value. To make the process of making these changes less tedious, json data was converted into pandas dataframes. It allowed us to split the timestamp columnn into date and time. Since the data collected followed a time series it was necessary to visualize that in the data table. Pandas "pivot_table" method was used to reshape and reindex the dataframe. The sensor records the data of a health parameter after a certain interval, this means that at one instant only one measure is recorded and hence data of only one measure exists. After reshaping the data table and columns there were a lot of "NA" values in the table. To solve this average values were considered and the average was taken either on Date or Time column, according to task being undertaken.

These steps had to be taken for each request and depending on the task being performed the data manipulation varied. These were later defined as functions for specific tasks according to the requirement of the format of the data to be returned.

4.2 Implementation of radar charts

In the visualization aspect there were three specific tasks that needed to be fulfilled:

- activity based profile - for specific hours during the day
- daily profile of a single patient
- weekly profile of a single patient
- for a certain number of population of patients, for a specific period, visualize clusters on average profile

To achieve these goals it was decided to use Radar plots because our data had multiple features which needed to be compared individually and also as a whole. Radar plots allow us to compare a measure with other instance of that measure on that axis. But as a complete plot it can present the difference in the results of the entire combination of features. To accomplish the above conditions the following solutions were suggested:

- profile of hourly data of the patient for individual measure during a specific period, a single day or week.
- complete patient profile of all vitals of daily average values over a certain period of days
- profiles of selected number of patients, for a selected period of time, for specific number of measures, evaluated by clustering algorithm

This meant that different data manipulation was required for each call and therefore different functions were constructed.

4.2.1 Data Pre-processing

Hourly patient profile for each measure meant that, the timestamp needed to be split to the hour. Additionally, we wanted to focus on only the vital parameters of the patient, so we needed data of each hour for each of the six measures only. The measures are data recorded by the devices and themselves were a row item of the data table, they existed only for certain periods. Values of measures like battery status or motion depended on the time period selected for evaluation. So filtering them out was necessary to be dynamic. The profiles also wanted to show the maximum and minimum values for that hour along with the average value to get an idea of any fluctuations in the data that could be crucial

in determining the patient's health status. After all these steps another important thing was to get the data in the right format to be processed easily for visualization.
Pandas was the chosen library for this task and it had several competent methods for performing these steps. A minor issue encountered during this though, was that after certain operation the object type of the result would change and so would the kind of methods that can be applied on it. But pandas does provide the ease of taking numerous tries out of the umpteen methods in its library. After some trials, a function named *hour_func* was created in *getdata.py* file. The data is retrived by sending the API request using the "get_measurements_request" method. It starts with creating "hour" column by splitting the timestamp column, then the data is grouped according the "measurename" using the dataframes "groupby" method. We want the resultant data file to have just two columns, one containing the hours and the other the corresponding values. Each measure will have three sets of these - maximum values, average values and minimum values. These sets were to be made for each measure, so a "for loop" was designed which also helped in removing the unnecessary measures.

The same "getdata" file consists of another function to get data for other jobs. For most tasks, it was necessary to have measures as column features so that they can be seen as time variant. In this function, after data retrieval using "get_measurements_request" method and creating date and time columns, the dataframe is reindexed using "pivot_table". Since the monitoring intervals for each measure were different, to change the view of the dataframe average values were taken into consideration. The means were calculated on the basis of *Date* or *Time* depending on the plot. The values of each measure are daily averages.

4.2.2 Radar plot

The radar plots required for this project were of two types, one with hourly data of one category in one chart and the other with multiple features on the same chart. For hourly data, all the axis could have same scale since they represented a single variable and the values would stay in that range. For the plot with all features, since our variables had different ranges of data - SPO2 is always around 90s while the value of breath would be 12-15 range - we needed radar plot flexible enough to accommodate different scales. We started with python Matplotlib but it did not provide enough interactivity for the user interface. Plotly and Dash were the next options as they provide a simplistic way for creating interactive web applications in python. But there current version could not provide the necessary customization, specifically to the radar plots. The next option was to use Javascript, which is efficient in making web applications and its D3 -Data Driven Documents - library make data visualization flexible using technology such as HTML, SVG, and CSS in the most efficient way. It has great documentation which was very helpful in

creating the modifications.

For hourly patient profile, the function had to create multiple plots on the same page for each feature. The scales were same for an individual plot, which was created by updating the highest value in the data, which is initially set to 0. The chart was created using SVG container. The radar charts were created using "d3.svg.line.radial()". The background circular grid is used to show the scale of the plot.

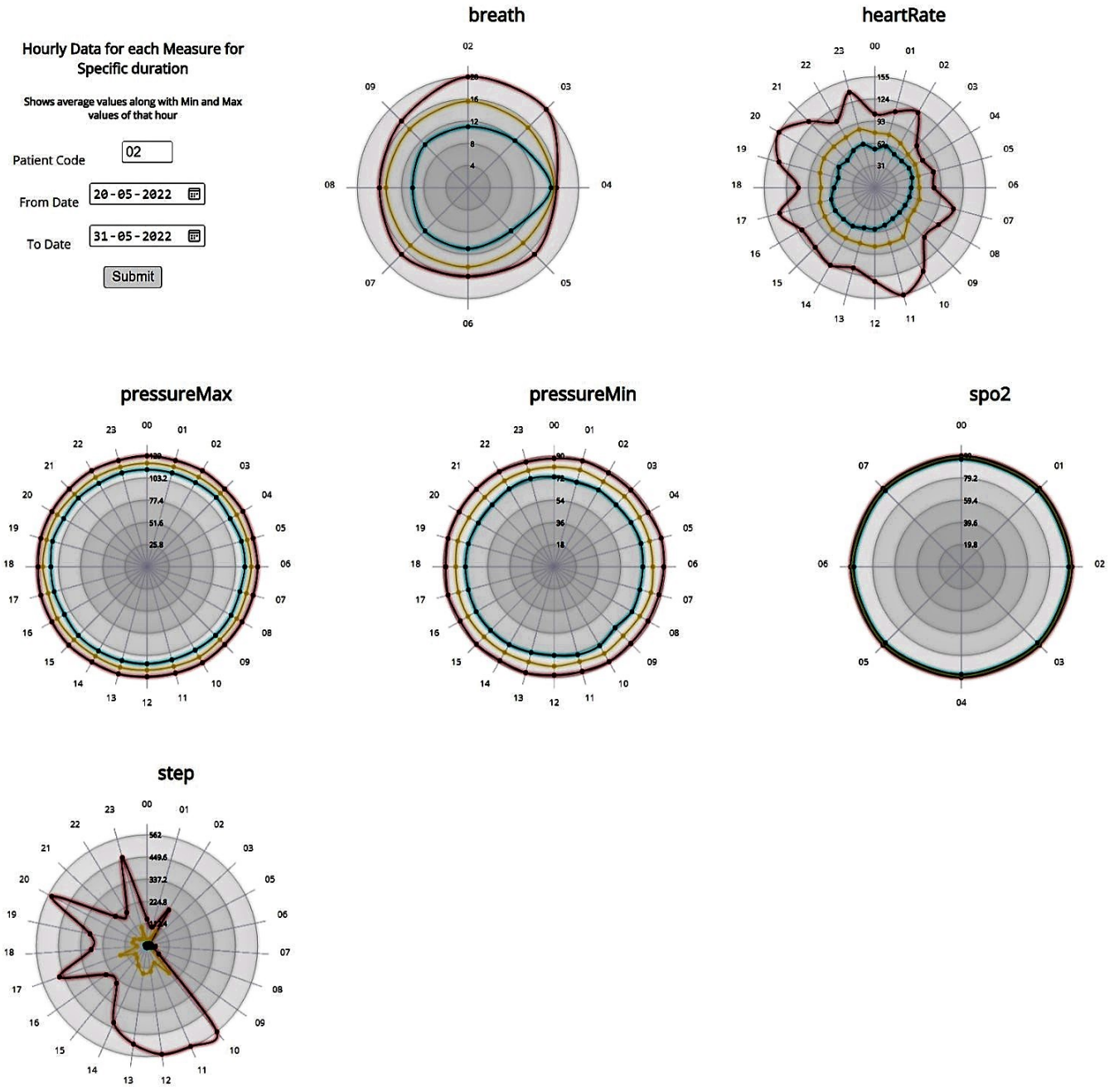


Figure 4.1: Hourly average profiles of all variables

The radar plot with same scale, each axis represented the hour, but for certain periods the data was not recorded, for example when the devices were charging, so the data did not

contain values for all hours. Hence on some occasions the plot would not contain 24 scales for each hour.

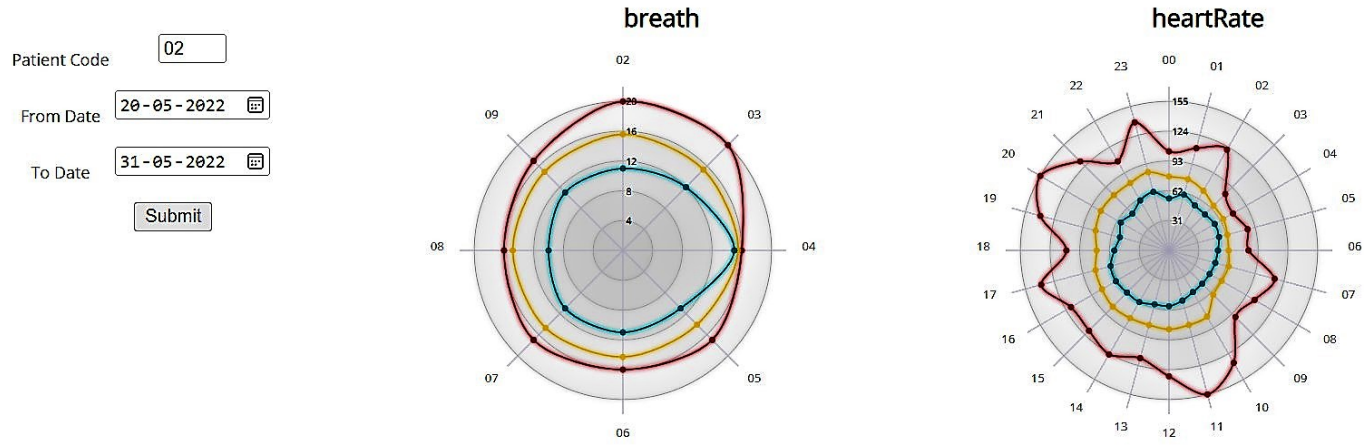


Figure 4.2: Each plot gives hourly average values along with maximum and minimum values for that hour

Radar plot for displaying the patient profile with all features, required different scales on each axis for which a function *"autoScalesAxes"* was created, in which data is sorted and a padding value is added so that the scale doesn't start from the center of the plot. The radar plots were created similar to hourly plot but the circular grid was removed since here the scale was on each axis. The grids would have crowded the space and would hinder proper visualization of the data.

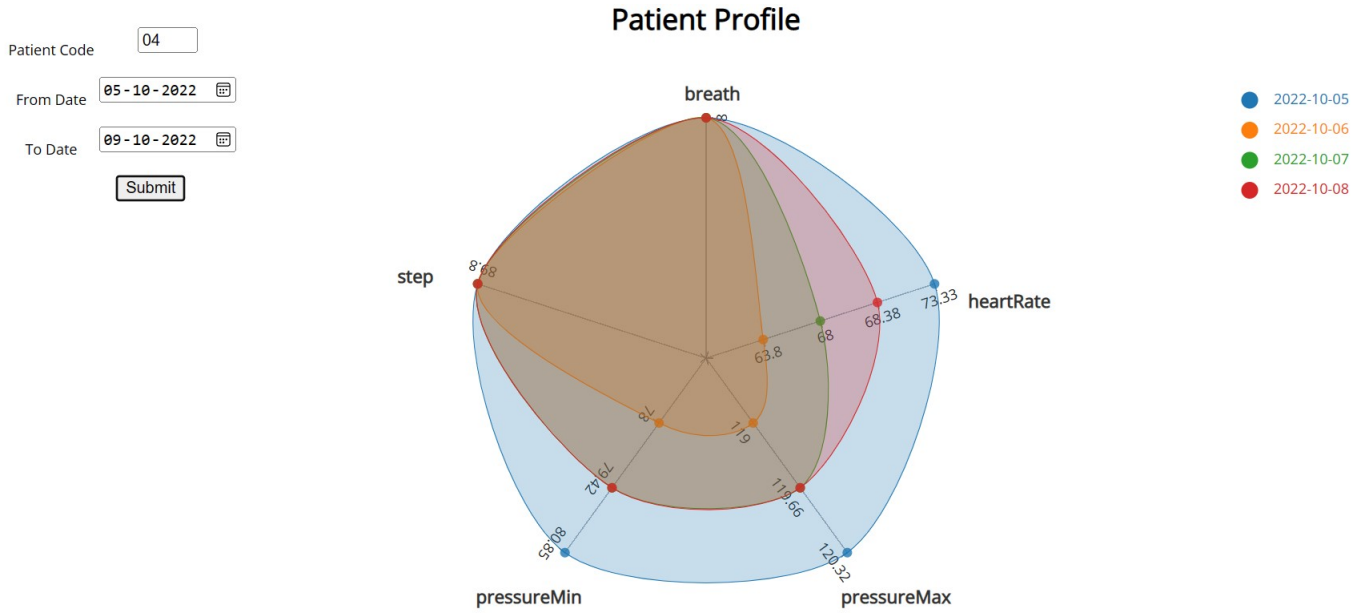


Figure 4.3: Daily average Patient profile

Each plot is filled with color which vary in opacity when hovered or selected. This was done to differentiate it each plot from another since there were overlapping sections. The values of the variables gets highlighted on the axis when hovered on it. Since the data is a daily average, the legend displays the date with the corresponding color code of the plot. The legend is clickable which highlights only the plot for that particular day.

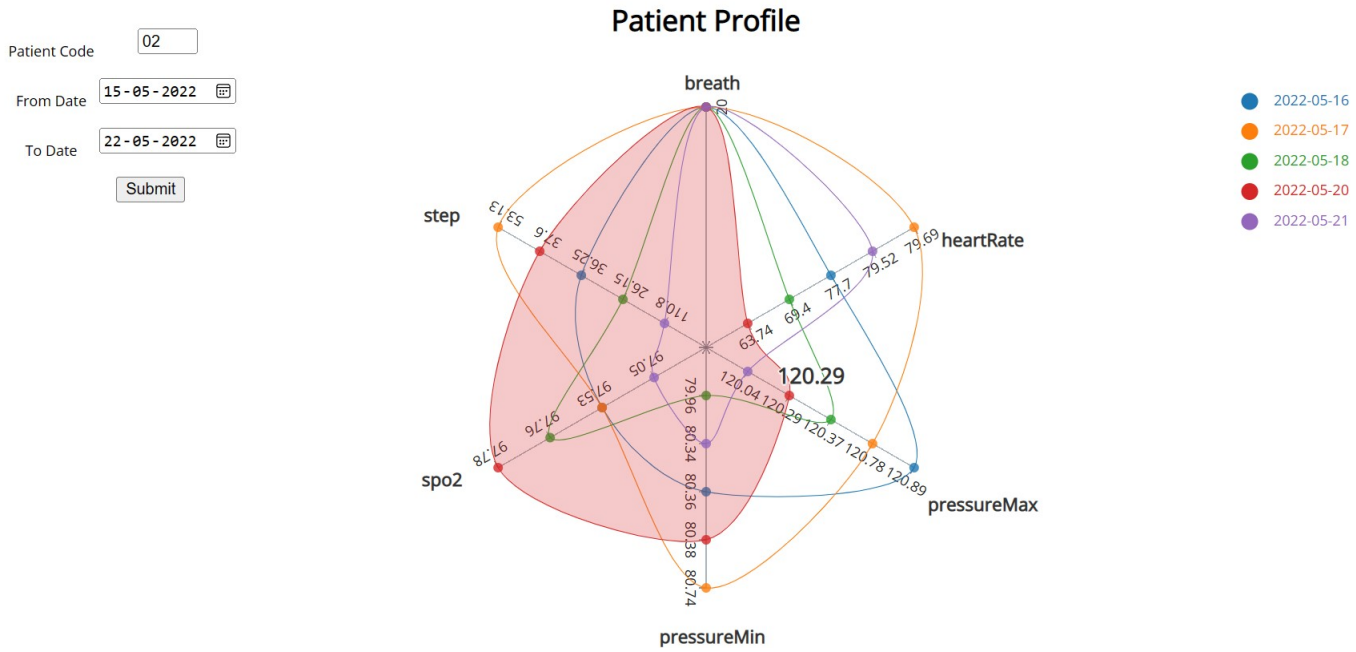


Figure 4.4: Highlighted profile of one day

4.3 Integration of Backend and Frontend

The project was made up of multiple technical aspects; data cleaning and manipulation, clustering algorithms and visualization. For data manipulation Python is the current standard and was also preferred due to its well defined libraries for machine learning algorithms. For visualization aspect, JavaScript and its D3 library brought much more flexibility and future scalability options that it was an obvious choice. Their combination might not be the standard, but its widely used which was evident from the comprehensive options for their integration.

In this project, the primary interaction between python and javascript was exchange of data, where Python was used for backend while javascript was being used for frontend. In python, the library *"Flask"* was used for this which is web application framework. For javascript the Fetch API was used. It provides an interface for accessing and manipulating parts of the protocol, such as requests and responses. It consists of a global fetch() method that provides an easy, logical way to fetch resources asynchronously across the network[Moz23]. On the backend, "app.py" file was created which imports the Flask class and an instance of this class is created whose first argument- *__name__* is the convention - is the name of the application's module or package. This tells Flask where to look for resources such as templates and static files [Pal09]. For each visualization, data was being processed in different files using multiple functions. These functions were called in this

main file, in multiple function calls, to retrieve the final data that needs to be sent to the frontend. The *route()* decorator is used to tell Flask what URL should trigger which function.

On the frontend, each visualization plot had its individual html,css and data.js file. HTML file takes inputs from the user and calls the data.js file. This js file uses the "Document" method *getElementById()* to read the input data. This data is sent to the "route" defined in app.py using the "fetch" method. The response is given input to the "RadarChart.js" file which creates the plot.

4.4 Clustering

Clustering is used to segregate different types of data into a certain number of clusters in such a manner that the data points belonging to a cluster have similar characteristics[Sha22]. During this project, the data collected was of 20 real patients and 5 dummy patients. The aim was to see if there were any similarities in the patients' profiles. Since they were in the same environment, and some of them with similar health problems, it could show some pattern and help in evaluating how the patients are reacting to certain changes. For example, if a change is made in a particular activity, environment or medication; it can help in assessing how it affected the overall patient population. This can help the staff to evaluate the how small changes affect the patients health and quality of life in general.

4.4.1 Data Pre-processing

For different measures the range of values were different, hence it was necessary to normalize the data before we could run any clustering algorithms on the data. Data normalization is a way of transforming numeric columns to a common scale. Initially we wanted to use a customized version of min-max scaling by using the minimum and maximum value of the normal range of the measures. Doctors and other medical forums were sought to decide the range. For example, for heart rate, the range is 60-130/min, resting heart rate could go lower than this but rare, with physical activity for 60-100 age group could go up to 130 [CDC22].
But the medical data can be very volatile, varying according to the patient. For one patient a certain blood pressure could be considered high while for someone suffering from Hypertension it could be considered normal. Also the activity being performed at the time of recording the measure would affect the value and could be either higher or lower than the acceptable range. So it was decided to use the normalization methods available

in the python library. In Python, Pandas library contains multiple built-in methods for calculating common descriptive statistical functions, but we use the Scikit-Learn library to transform the data to a standard scale. Since the data is read every few minutes and is real health data the range of values could vary drastically. Even the outliers can be of much importance in diagnosing any health deterioration or other issues. To make sure the normalized data is not very sensitive towards outliers [Mor21] and can adapt to changing data multiple normalization techniques were used.

The data is called through the API every time before performing clustering according to the period selected by the user, so normalization is also done on the selected data and the range of values in that period. Using pre-defined range could have changed the distribution of the original data which might have affected the results poorly.
A step before clustering is to use "dimensionality reduction" method to reduce the number of dimensions or features. It not only reduces the number of columns of features but also helps find the features which best explain the data. For datasets where the dimensions are more than or almost equal to the number of rows it helps in selecting important features and also reduces the time taken for running the algorithm. Clustering algorithms are mostly used with less features for its easier to visualize the results and clusters in 2D or 3D plane.

Since our dataset had six features the clusters would have been hard to visualize. We used PCA, which is a dimensionality reduction method in the Scikit learn library, to reduce the features to two dimensions only. But since we could use radar plots to visualize the results we decided to use all the features for clustering. Although the UI provides the option to select less than six features. It was done with the intention of providing the staff of the RSA the freedom to choose the measurements they wanted to focus on.
A file specially dedicated to perform pre-processing functions was created, which consists of functions from retrieving data to normalization. The radar plot for clusters calls for a range of population using patient codes. The original API allows for only one patient code or no patient code to retrieve data. Similarly the measures could also be either single or all. So the API get call retrieves all the data with just the duration used as filter. Then the patientcode column is filtered using the patient codes to get data only in the range. After this the measurename column is filtered, but for this it was little more complicated, since we don't know the number of measures that can be chosen by the user. We check the length of the array that contains the names of the measures selected and then filter the data for the measure on that position in the array. The rest of the function follows the timestamp split and reindexing of the dataframe. The radar plot also allows to plot an individual patient's data with the cluster centroids. It provides the option to visualize if that particular patient falls in one of the clusters or is an outlier. For this another method was created following the same steps as above method but for a single patient code.

This file also contains a class called "scale_data"for calling different normalization techniques used before applying clustering algorithms. After clustering, the centroids of the clusters have normalized values. That data does not provide any insight to the staff at the RSA. To get the values of the measures in their initial range, another function was created that uses "inverse_transform" method of the respective normalization method to get the final values which are represented on the plots.

4.4.2 Algorithm calls

For clustering we wanted to use all the features and allow the user to select the number of features or measures on the basis of which they wanted to evaluate the data. Fuzzy clustering and Graded PCM perform well when working with less dimensions but for high dimensional data K-means clustering would perform better. To have the option of utilizing either, all the algorithms were kept. File named "algorithmCall" was created, which contained all the methods for calling a specific algorithm. Since we were working with real health data of a limited number of patients, which algorithm would be ideal was not ensured, so we used multiple methods consisting of both hard and soft clustering. We started with K-means, which is one of the standard machine learning algorithm used for clustering. It was implemented using "sklearn" library method. K-means algorithm, as the name says, uses mean values to estimate the centroids and so the values of the cluster centers are not the actual point but the mean of points present in that cluster.

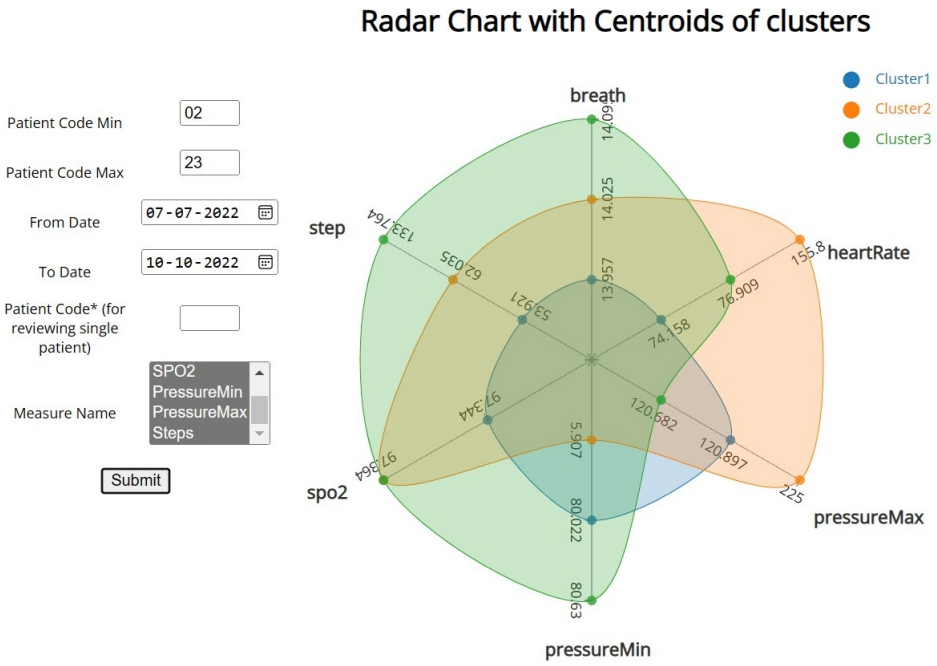


Figure 4.5: Centroid Radar Chart for 3 clusters(using K-means)

In our case since the data is of real patients we were interested to have centroids as actual data point from the dataset which represents a real patient. To make the centroids interpret-able, we opted to use K-medoids algorithm, also called Partitioning Around Medoids (PAM), because it partitions sets of data points around a medoid- a point in the cluster whose sum of distances(also called dissimilarity) to all the objects in the cluster is minimal- and constantly tries to lower the dissimilarity among the points in the same cluster. For the implementation, we installed the package from [22], which implements k-medoids clustering with PAM [SR19] [SR21] [Sch22][LS22].

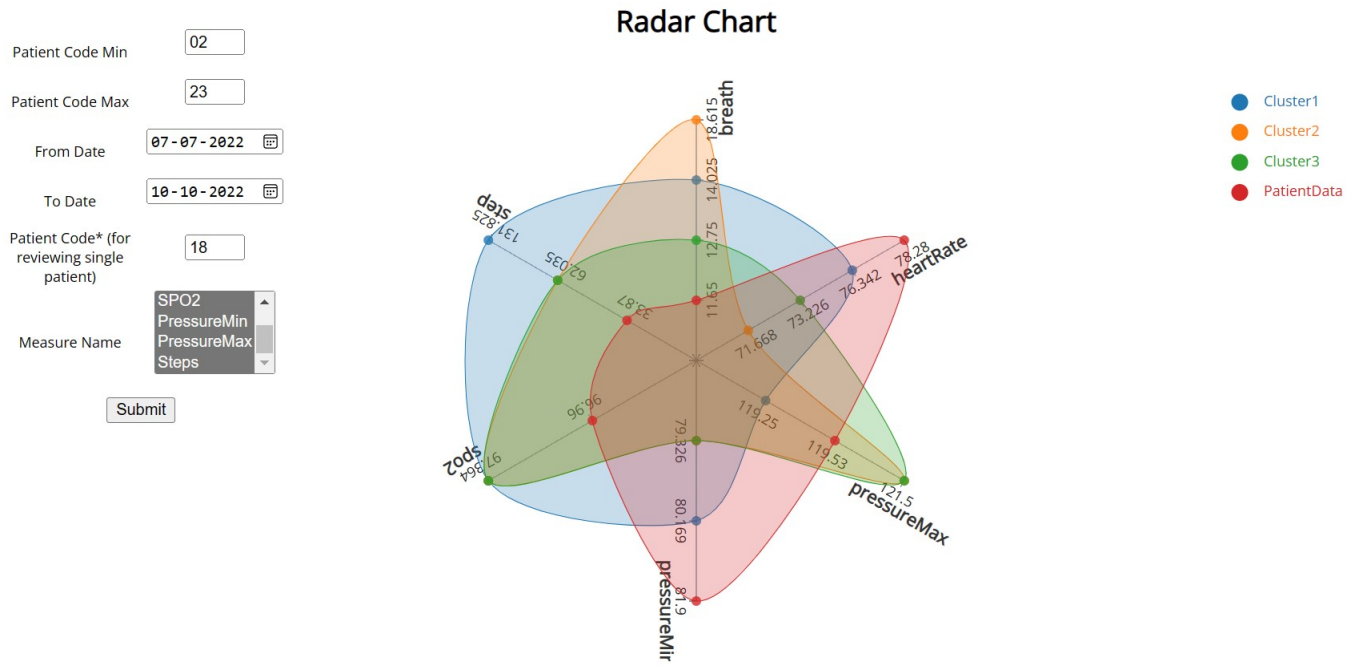


Figure 4.6: Medoid Radar chart; Each plot represents the center of the cluster and a real patient

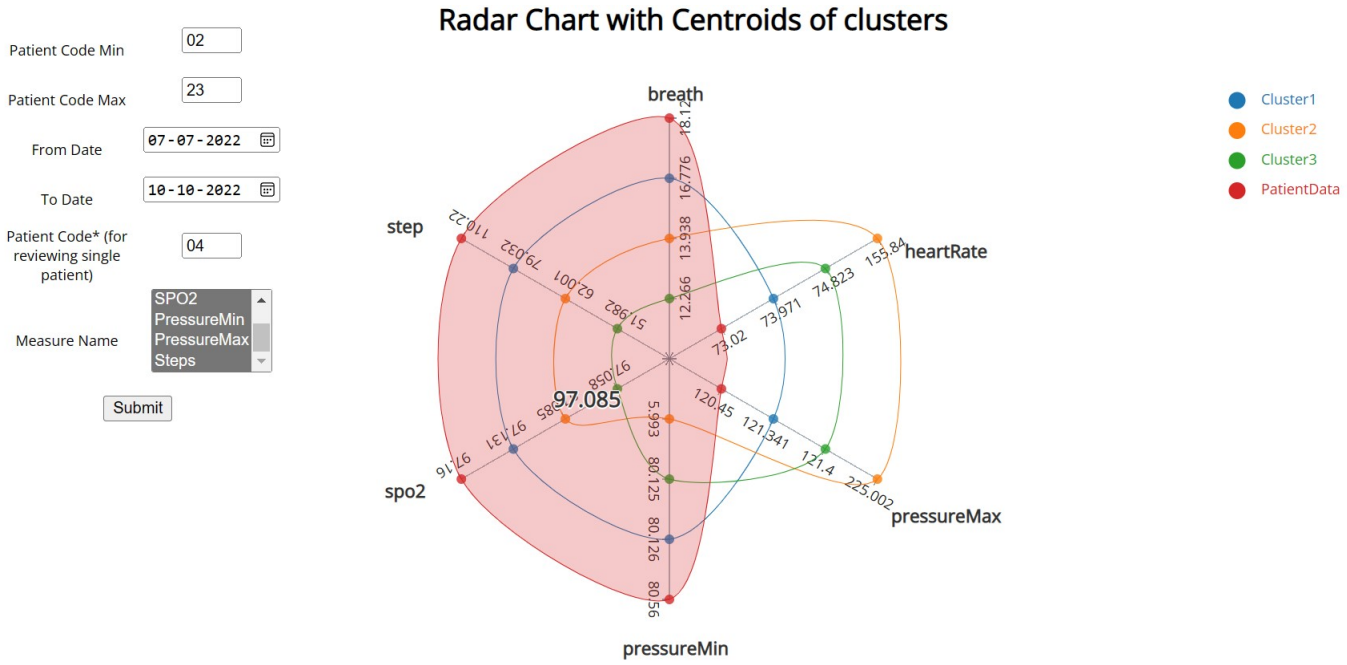


Figure 4.7: Individual patient profile can be compared with the cluster centers

For soft clustering, Fuzzy C means and Graded Possibilistic Clustering Model [MR03] were

used. *Fuzzy-c-means* [Dia19] module of python was installed to implement fuzzy c means algorithm. While GPCM was used from the implementation from the course 'Machine Learning: A Computational Intelligence Approach (MLCI 2022)' [MRD22].

All the functions in this file are called from *app.py* file which is the main flask file that integrates with javascript.

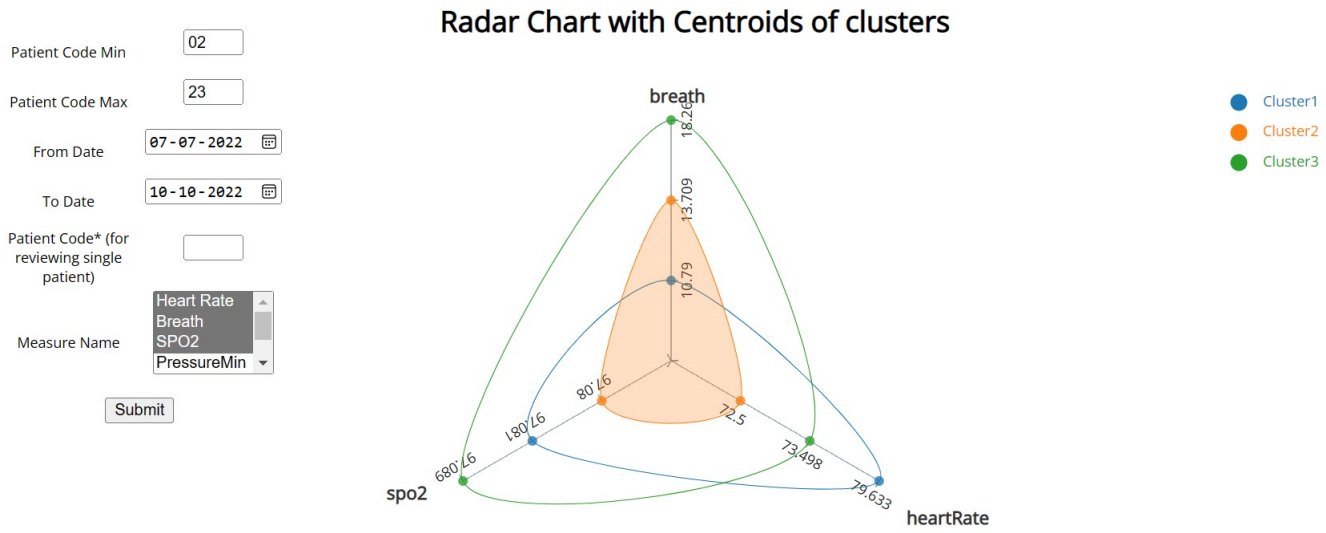


Figure 4.8: Centroid Radar Chart for 3 variables(using Fuzzy C means)

4.5 Trend Analysis

We wanted to explore the scope of trend analysis on health data. The idea was to analyze the trends of bio-metric data of an individual patient's profile and see if we can get any substantial results that can help make some conclusions about the patient's health. Medical data can be difficult to interpret and make any predictions. But considering data as a stream of data, it can be evaluated to see if there are any drastic variations in the pattern of the data. For example if there is sudden continuous change in the heart-rate of the patient, it could indicate the patient is in some form of distress and needs intervention. The method in [Abd+16] is capable of detecting outliers in non-stationary data. It already had an implementation in MATLAB provided by the authors. In our project the data was being accessed through the API, which required a lot of pre-processing for each task. And finally the visualization for the users needed to be dynamic and interactive. Considering all these conditions it was decided to work with python.

4.5.1 Data Pre-processing

Analysis of trends in the data required to see the data as a stream of data, in quasi continuous format. So here for the pre-processing part the data was reshaped according to the time, so that each instant of the measure is kept as its recorded. For disputing values, the aggregation was done using average value of the conflicting observations. Although this did lead to more NA values, since each measure is recorded at a different interval there were certain instants where not all features had a value read by the senors. These NA values were replaced using mean values. The data was normalized before applying the algorithm. The results of the algorithm were stored in a dataframe and sent to the javascript interface in a json format.

4.5.2 Matlab to Python

There were multiple options for translating the Matlab code to python. One of the initial options was a tool named Small Matlab and Octave to Python Compiler (SMOP) capable of understanding basic Matlab code and parsing it to python. It was installed using [16]. For it to work the python version had to be depreciated and even then it had some limitations. Some of the methods used by it for translation were not working properly. The next alternative was to install Matlab engine API for python [Mat22]. It provides Python a means to import the MATLAB module and start the engine. It allows to call certain methods native to Matlab in python. It does make processing more time and resource consuming.
None of the above options were capable of providing an optimal translation. So the final option was to manually convert the code to python. The SMOP translated gave some head start for the translation. Also for some functions, initially using Matlab engine provided insight into what function the method was performing and the ideal expected result. Python's Numpy library was used to replace multiple Matlab functions using the resources in [Num22] and [Qua17]. Since downloading the data was not preferred, initial comparisons of how translated python code is working was done using dummy matrices. The behaviour of Matlab functions and their corresponding python function was not the same and it lead to multiple errors and the functions that were able to compile gave wrong results. Python's Numpy was processing the matrices as *'ndarrays'* and not as matrices like in Matlab [McC19]. It required to add some reshaping methods and 'keepdims' parameters to make the functioning similar to Matlab. In Matlab the Fuzzy C Means toolbox is much better than python's 'fcmeans' implementation. To get better results the implementation of Fuzzy C Means from the course 'Machine Learning: A Computational Intelligence Approach (MLCI 2022)' [MRD22] was used.
To ensure the FCM algorithm was working correctly, we ran a test to visualize the results, since the rest of the algorithm for outlier detection depended on the results of this algo-

rithm. Our dataset had 6 features, so to visualize the results we used sklearn's Principal component analysis (PCA) [Jol86] for dimensionality reduction.

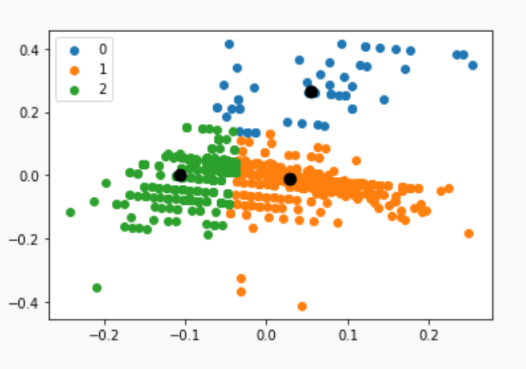


Figure 4.9: Clusters and centroids with FCM

The result of FCM algorithm on a single patient's data can be seen in 4.9. Since the data is of just one patient and is a continuum of observations there are no real clusters or separated groups. But for our aim of detecting outliers the results were sufficient, which are processed in the main algorithm.

```
iter 50 of 869 - rho = 0.27355767744419535, theta = 0.041126172052676546, eta = 0.041126172052676546, alpha = 0.9927355767744419
iter 100 of 869 - rho = 0.4279132600270791 3, theta = 0.046947905930589284, eta = 0.046947905930589284, alpha = 0.9942791326002708
iter 150 of 869 - rho = 0.5214730522887147, theta = 0.0568679682312786, eta = 0.0568679682312786, alpha = 0.9952147305228871
iter 200 of 869 - rho = 0.5781803769008562, theta = 0.06378298184175024, eta = 0.06378298184175024, alpha = 0.9957818037690086
iter 250 of 869 - rho = 0.6125526011982431, theta = 0.06817453803224494, eta = 0.06817453803224494, alpha = 0.9961255260119825
iter 300 of 869 - rho = 0.6333842675498227, theta = 0.07089195615186461, eta = 0.07089195615186461, alpha = 0.9963338426754982
iter 350 of 869 - rho = 0.6460123136467221, theta = 0.07255682772842365, eta = 0.07255682772842365, alpha = 0.9964601231364673
iter 400 of 869 - rho = 0.6536666562863594, theta = 0.07357187967721313, eta = 0.07357187967721313, alpha = 0.9965366665628635
iter 450 of 869 - rho = 0.6583060472282674, theta = 0.07418917488883969, eta = 0.07418917488883969, alpha = 0.9965830604722826
iter 500 of 869 - rho = 0.6611163151459885, theta = 0.07456382847848952, eta = 0.07456382847848952, alpha = 0.9966111631514599
iter 550 of 869 - rho = 0.6628230032178519, theta = 0.0747916220534004, eta = 0.0747916220534004, alpha = 0.9966282300321785
iter 600 of 869 - rho = 0.6638558211864445, theta = 0.07492956952456209, eta = 0.07492956952456209, alpha = 0.9966385582118644
iter 650 of 869 - rho = 0.66448228498486, theta = 0.07501327770358412, eta = 0.07501327770358412, alpha = 0.9966448228498486
...
iter 700 of 869 - rho = 0.6648604212124903, theta = 0.075063817079698, eta = 0.075063817079698, alpha = 0.9966486042121249
iter 750 of 869 - rho = 0.6650893328250774, theta = 0.07509441666270367, eta = 0.07509441666270367, alpha = 0.9966508933282507
iter 800 of 869 - rho = 0.6652291847245769, theta = 0.07511311298291878, eta = 0.07511311298291878, alpha = 0.9966522918472458
iter 850 of 869 - rho = 0.6653133340608659, theta = 0.07512436324884708, eta = 0.07512436324884708, alpha = 0.9966531333406087
```

Figure 4.10: Values of parameters for outlier detection

4.5.3 Line Chart

The parameters get updated in a *for loop*, with each iteration a new data point is added to the dataset, starting from an initial value of the *first batch*. It can be decided according the number of observations, which also eventually decide the number of iterations the

algorithm runs for. The parameters represent the change in the trend of the data in a temporal manner, multi-line plot exhibits the variation in their values individually and compared to each other. The values of the parameters according to the iteration number can be seen in 4.10. It can be seen that the values of θ are very close to zero, because for outlier detection the algorithm is in *no learning regime*. Value of η is exactly same as θ because its initial value is 1 and in subsequent calculations it is calculated by taking product of θ and η_0. We get these results with value of initial $\alpha = 0.99$. In 4.11 the parameters are plotted against the number of iterations. As can be seen θ and η overlap, while values of ρ increase in the first 200 iterations as new data starts coming in. But gets stable once the change in values of observations reduces.

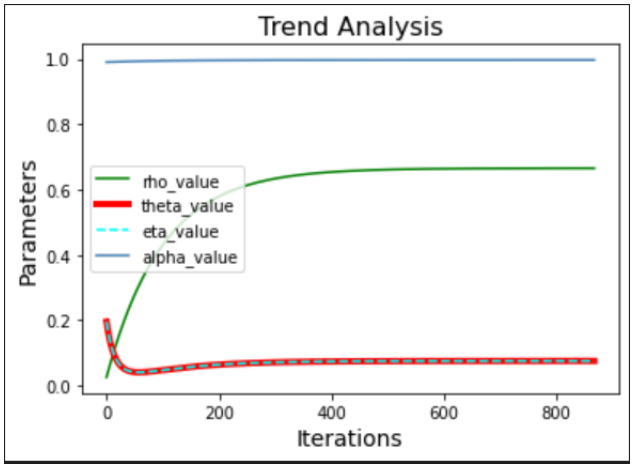


Figure 4.11: Plot of parameters for outlier detection varying with the iterations

As a means to visualize the trend evolve with the increase in the number of iterations, an interactive multi-line chart was created using Javascript D3 library. The integration between python and Javascript was done using Flask and Fetch API of javascript.

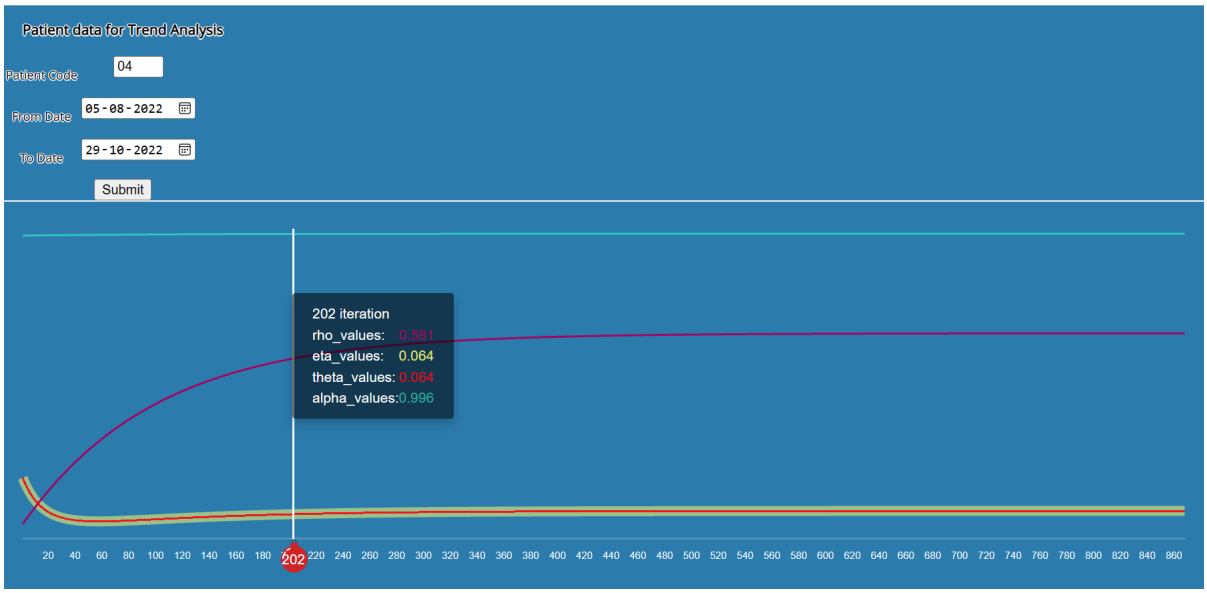


Figure 4.12: Animated multi-line chart of parameters for outlier detection varying with the iterations

As can be seen in the plot in 4.12, the user can select the patient code and time period, but the parameters for analysis algorithm can be only changed in the backend. The plot has a slider to visualize and evaluate the values at each iteration.

Chapter 5

Testing

The project consisted of three main tasks in terms of development - Visualization, Clustering and Trend analysis. These tasks involved other essential operations like data preprocessing and decisions like choosing the clustering algorithms. In this section we discuss all minor as well as major decisions made during the process and how the results of those choices made us manoeuvre throughout the assignment.

5.1 Data Manipulation

The raw data had vital parameters of the patients as row values. For the actions to be performed on the data we needed those measures to be primary features. To convert these measures into columns and their corresponding values as row data pandas "pivot_table" was utilized. This reformatting of the table, sometimes leaves some cells with no values which can be replaced by either any scalar value like 0 or left as Nan using pivot_table. These Nan values can be filled by using pandas dataframes "fillna" method which allows to fill these spaces by preceding observation, next observation or any statistical measure. Using "0" evidently skewed the data, especially for the visualization part. The scale would show a greater difference than actually existed in the values. "ffil" i.e. using the following observation and "bfill" i.e. using the previous observation did not work well because in some cases the neighbour observation was also Nan. Figure 5.1 shows the radar plot with the skewed data. The Nan values finally were replaced using mean values and did not distort the data.

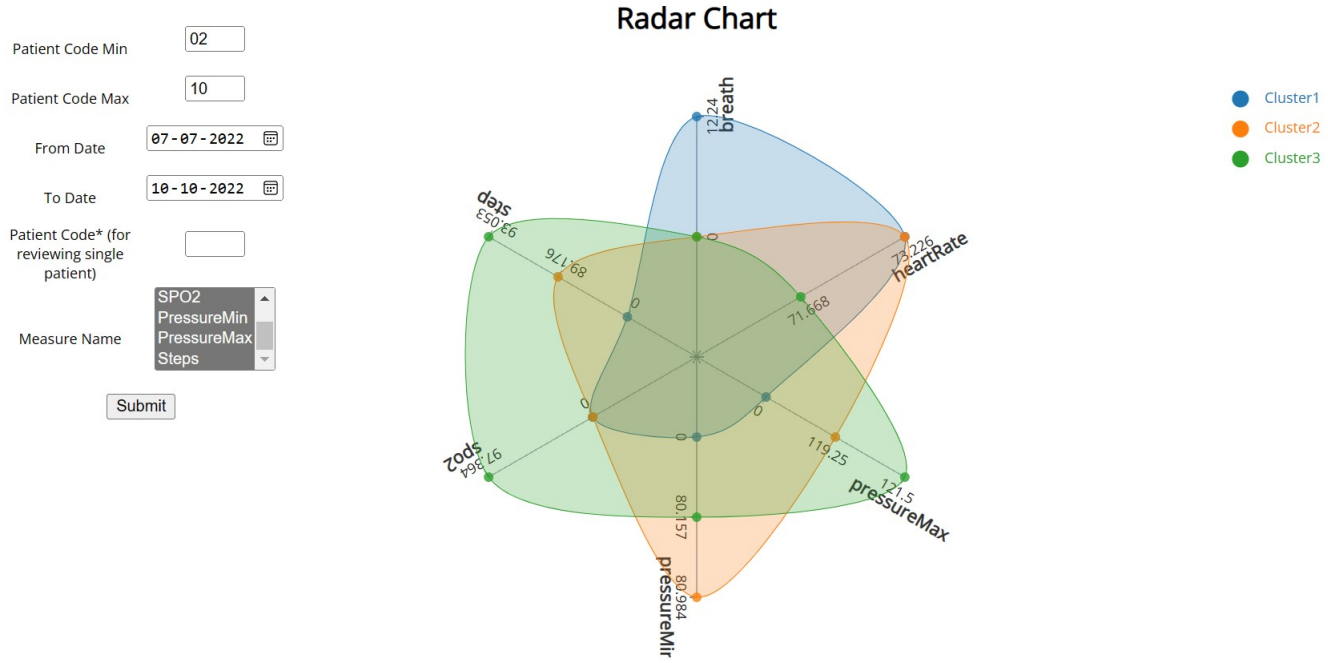


Figure 5.1: Radar Chart with skewed data

The data consisted of different biometric measures which have different ranges of values. For applying clustering algorithm, it was mandatory to have the data of all the features in the same range. For this purpose we used python's library "sklearn" which has several options for normalization of the data.

- Standard Scaler - replaces the data by subtracting the mean of the distribution and then dividing the result by feature variance. It does not bound the values to a specific range. Fits best to Gaussian distributed data.
- Min-Max Normalization - all the numeric values are scaled between 0 and 1, where the minimum value of the feature is transformed to zero and the maximum value to one. It is sensitive to outliers
- MaxAbs-Scaler - it scales each feature by dividing it with the largest maximum value in each feature. Is sensitive to outliers but works well to scale the sparse data.
- Robust-Scaler - works like standard scaler but uses median and interquartile range (IQR) instead, which is the difference between the 75th and 25th percentiles where most data points exist. It can handle outlier data points as well.

MaxAbs gave better results when Nan values were replaced with 0 as compared to other normalization methods.

5.2 Clustering Algorithms

For the purpose of visualization of clusters FCM, K-means, K-medoids and GPCM algorithms were implemented. They all require a normalization step before they can be applied on the data. Each algorithm was used with all the normalization methods to see how they performed. Out of the four algorithm, K-means and K-medoids fall in the hard clustering category while FCM and GPCM are soft clustering methods. GPCM is better at handling outliers [MR03], but it could not work with Robust scaler method. The values after applying robust scaler were very small and were being read as NAN by the algorithm.

Figure 5.2: Data and the legend used for comparisons

For testing and comparing the results of the algorithm, same parameters were used for each request. The legend shows the color coded scheme for the cluster centers.

Clustering using standard scaler normalization, FCM, K-means and GPCM seem to have cluster centers which cover different ranges of values that can represent multiple health categories. While centers using K-medoid have very similar values for most measures. Amongst themselves, FCM exhibits more distinct data distribution for each category. GPCM and K-medoid have very similar values or same values in some cases for different center's variables. K-means on the other hand has some values that don't fall in the normal range and seems affected by outliers.

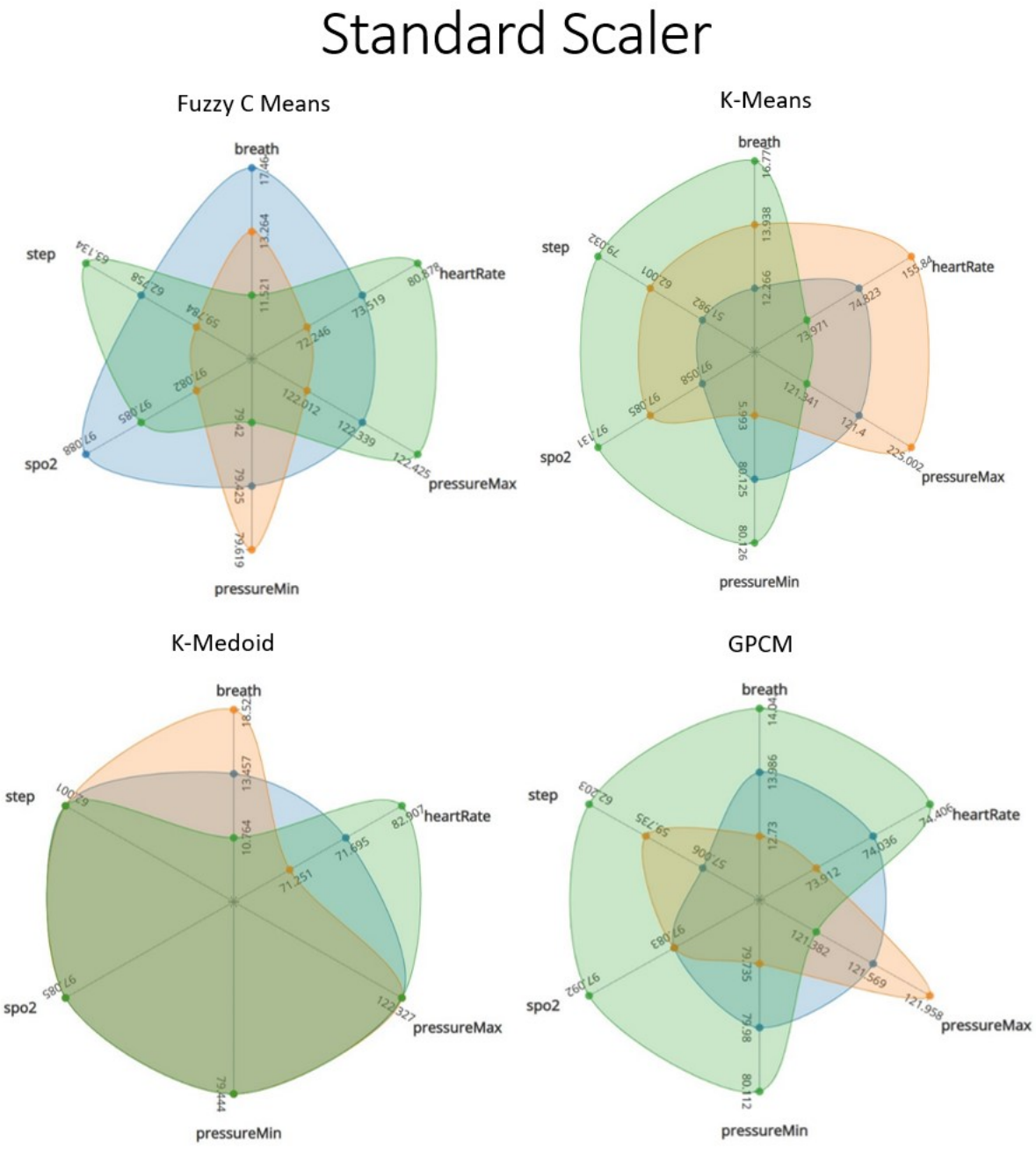


Figure 5.3: Comparison of results using different algorithms with standard scaler normalisation

FCM, K-means and GPCM applied to data normalized using Min-Max scaler provides similar results to those of standard scaler. The centers are very similar and represent varied range.

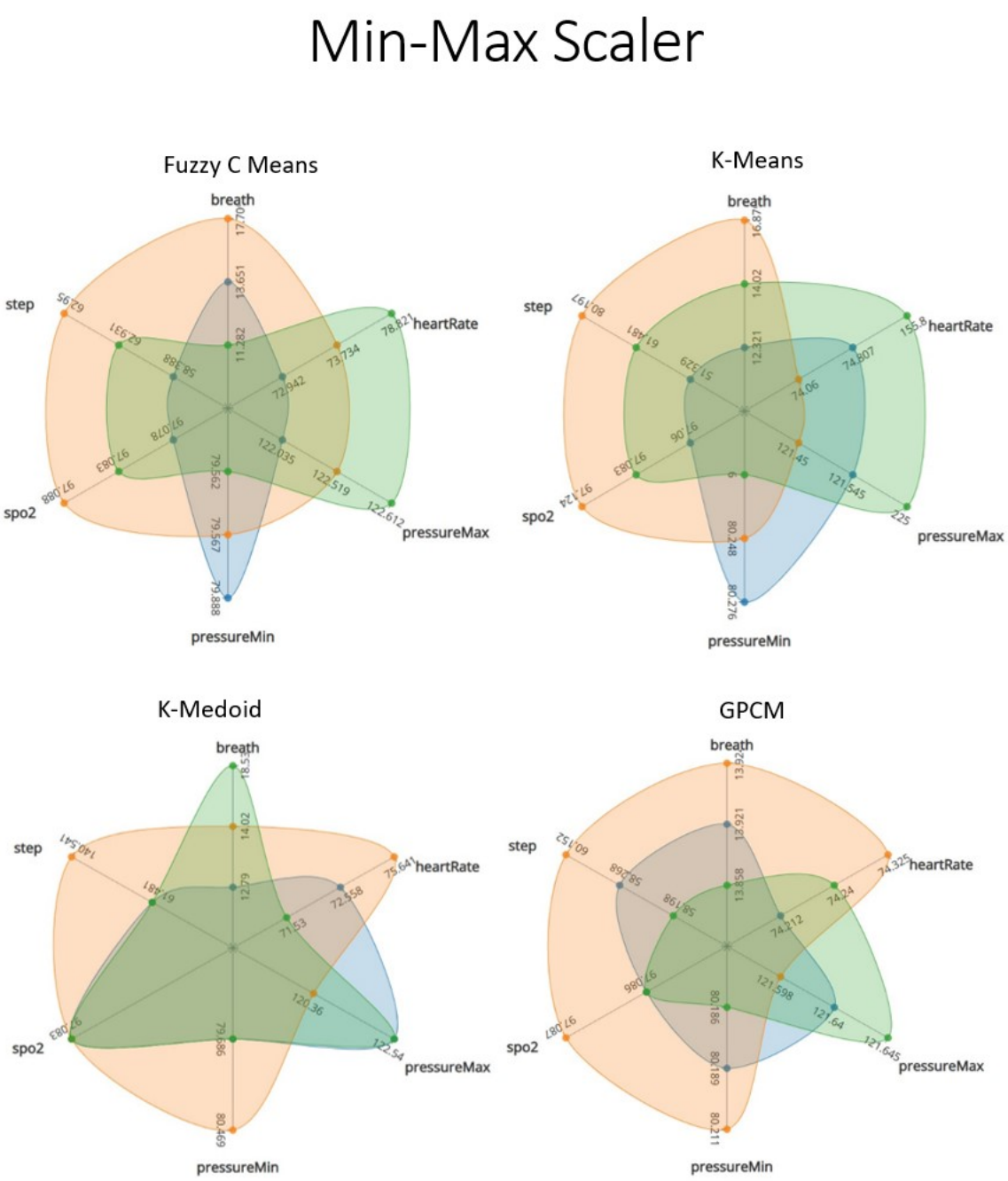


Figure 5.4: Comparison of results using different algorithms with min-max normalisation

K-medoid too seems to perform a little better using min-max normalization and shows more clearer possible categories. Even here, K-means represents some values which can be considered as exceptions. FCM performs as good as with standardization while GPCM

does not improve the range much. However, K-medoid improves the results in this scenario.

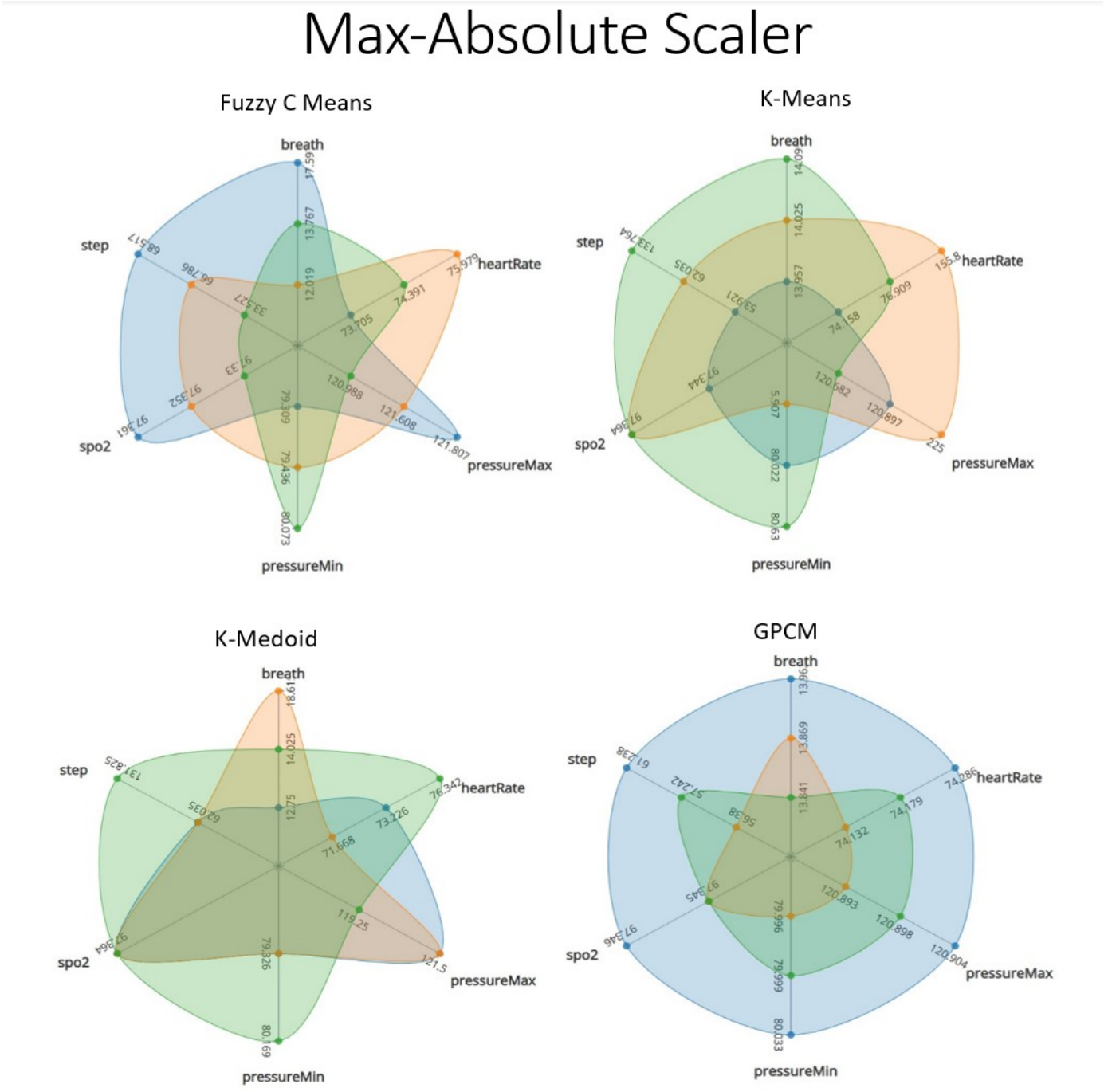


Figure 5.5: Comparison of results using different algorithms with max absolute scaler normalisation

For Max-Absolute normalization, K-medoids does better than others. Even though the shape of the plots look same, it was able to include more range of values. The others remained in the same range of values and did not show any improved results. Between them, K-means appears to be affected by outliers.

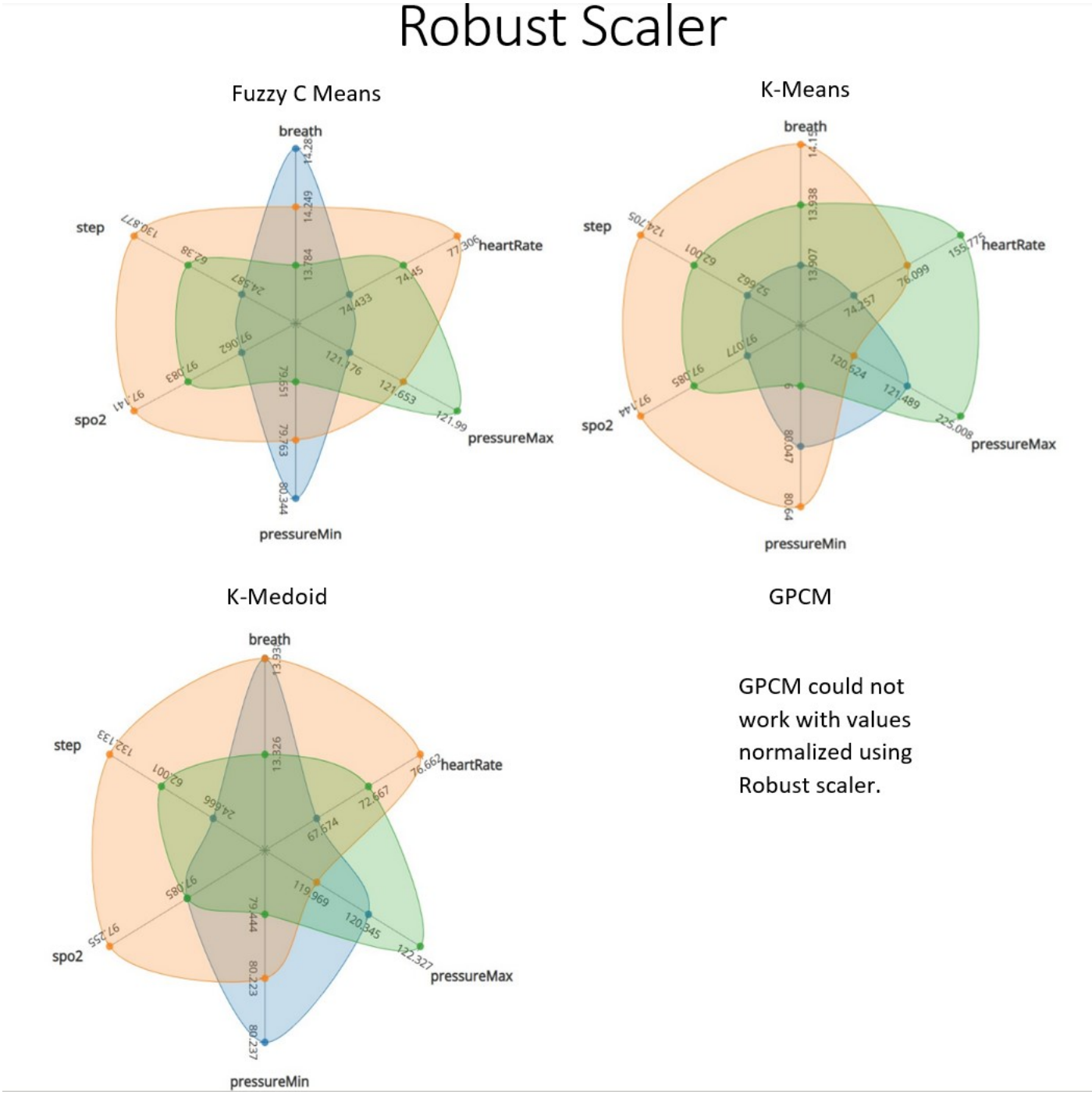


Figure 5.6: Comparison of results using different algorithms with robust scaler normalisation

GPCM algorithm could not process the values normalized using Robust scaler normalization as it was reading them as NAN values. K-medoid showed most improvement in this

case, while FCM and K-means did not show any substantial change.

Clustering algorithms or essentially any other algorithms behaviour primarily depends on the data. The limited number of patients and the subsequent data plays an important role in the results produced by the algorithms. All the algorithms used in the testing phase could not exhibit any distinct results, most likely, due to the sample size being limited.

5.3 Visualization results comparison

The radar plots for visualizing the clusters enable the users to evaluate the groups among the population of patients for specific periods of time. For the purpose of testing we evaluated the data in a four week period, divided in two phases of two week period, to analyse the changes in the centroids.

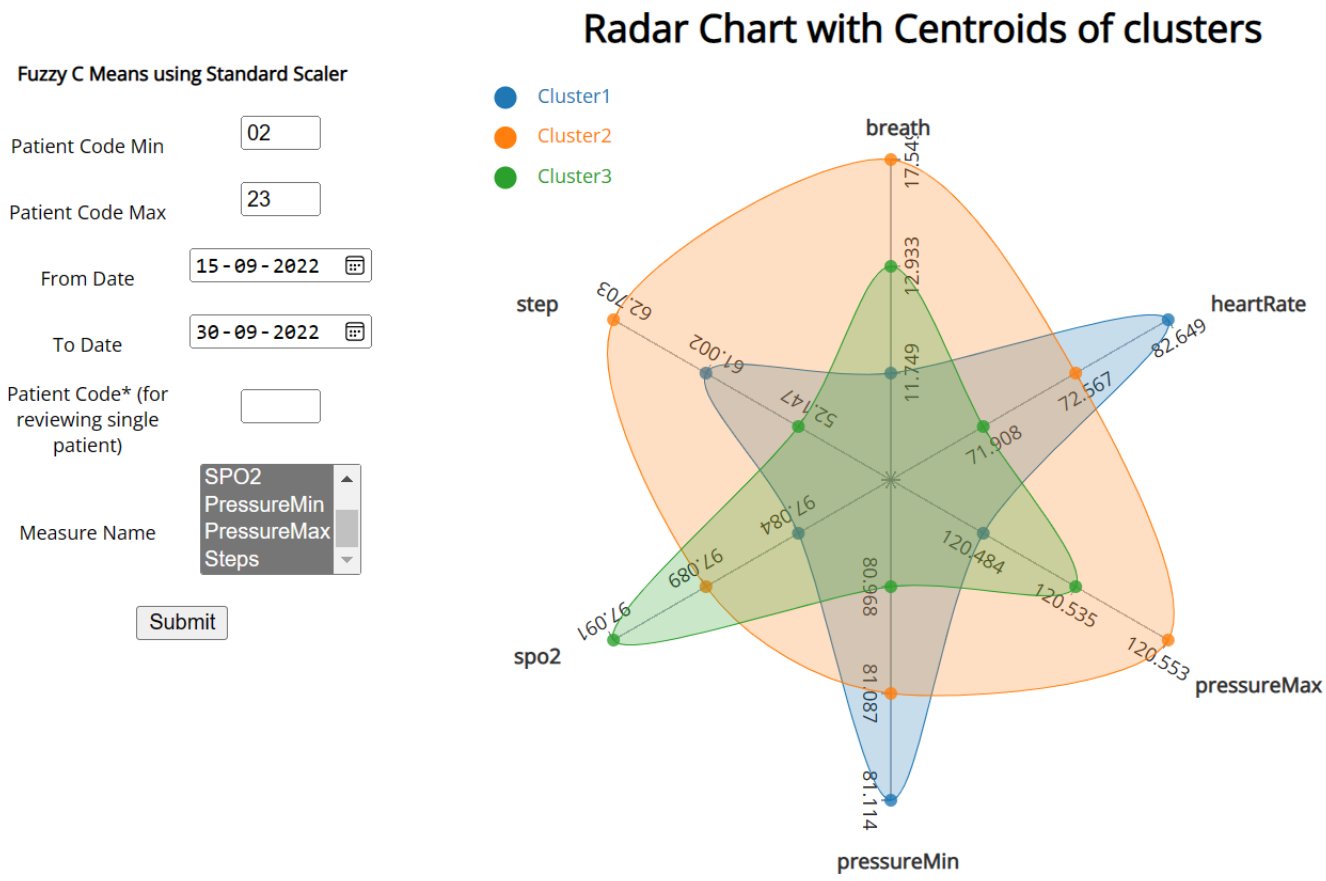


Figure 5.7: Clusters on data for the first period of two weeks using FCM

The test were run using standard scaler normalization. Since we employed both *hard clustering* and *soft clustering* techniques during the project, one method of each was used. Fuzzy C means and K-means, being the more commonly used techniques in machine learning, were selected.

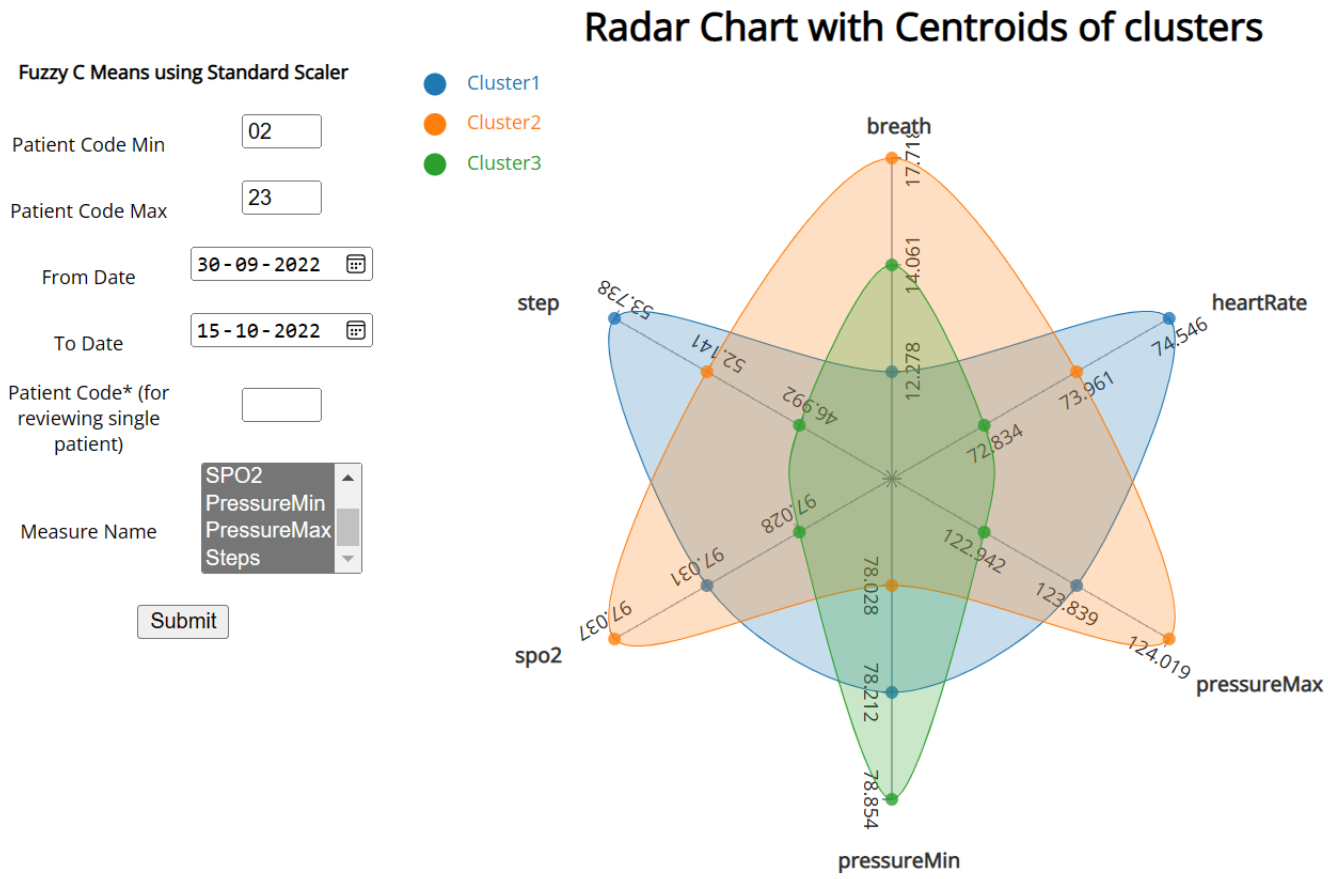


Figure 5.8: Clusters on data for the second period of two weeks using FCM

We can see that both the algorithms give nearly identical results for both first (figures 5.7 and 5.9) and second (figures 5.8 and 5.10) period of two weeks. And in both cases we can see the change in the shapes of plots of centroids.

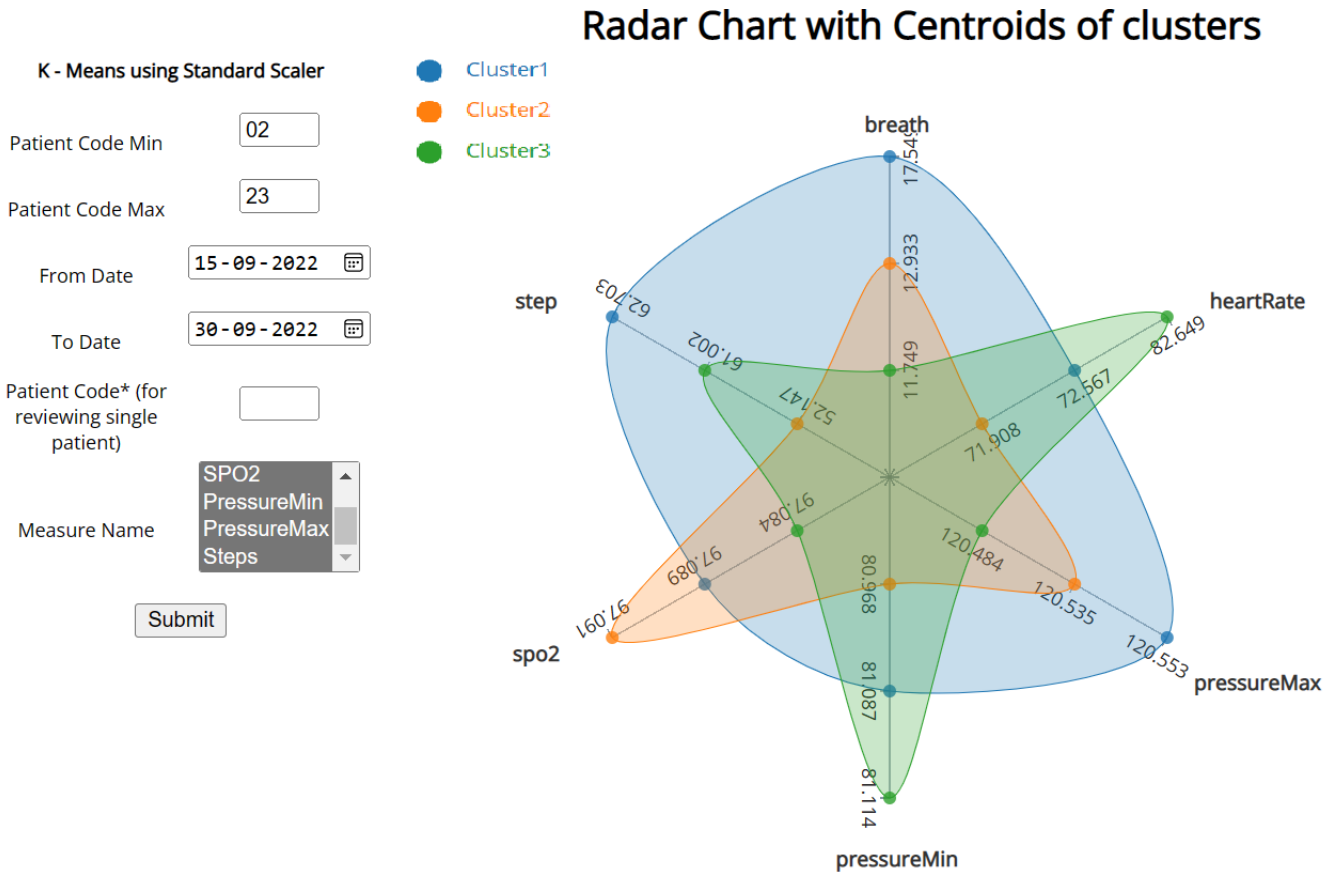


Figure 5.9: Clusters on data for the first period of two weeks using K-means

In the plot of the first period at least one plot can be seen well rounded, which signifies that all the biometric values were in the average range. While in the second phase all the centroids show at least one variable with lower than average value, with others being balanced. The plots can help in analyzing how a certain period of time affected the overall population.

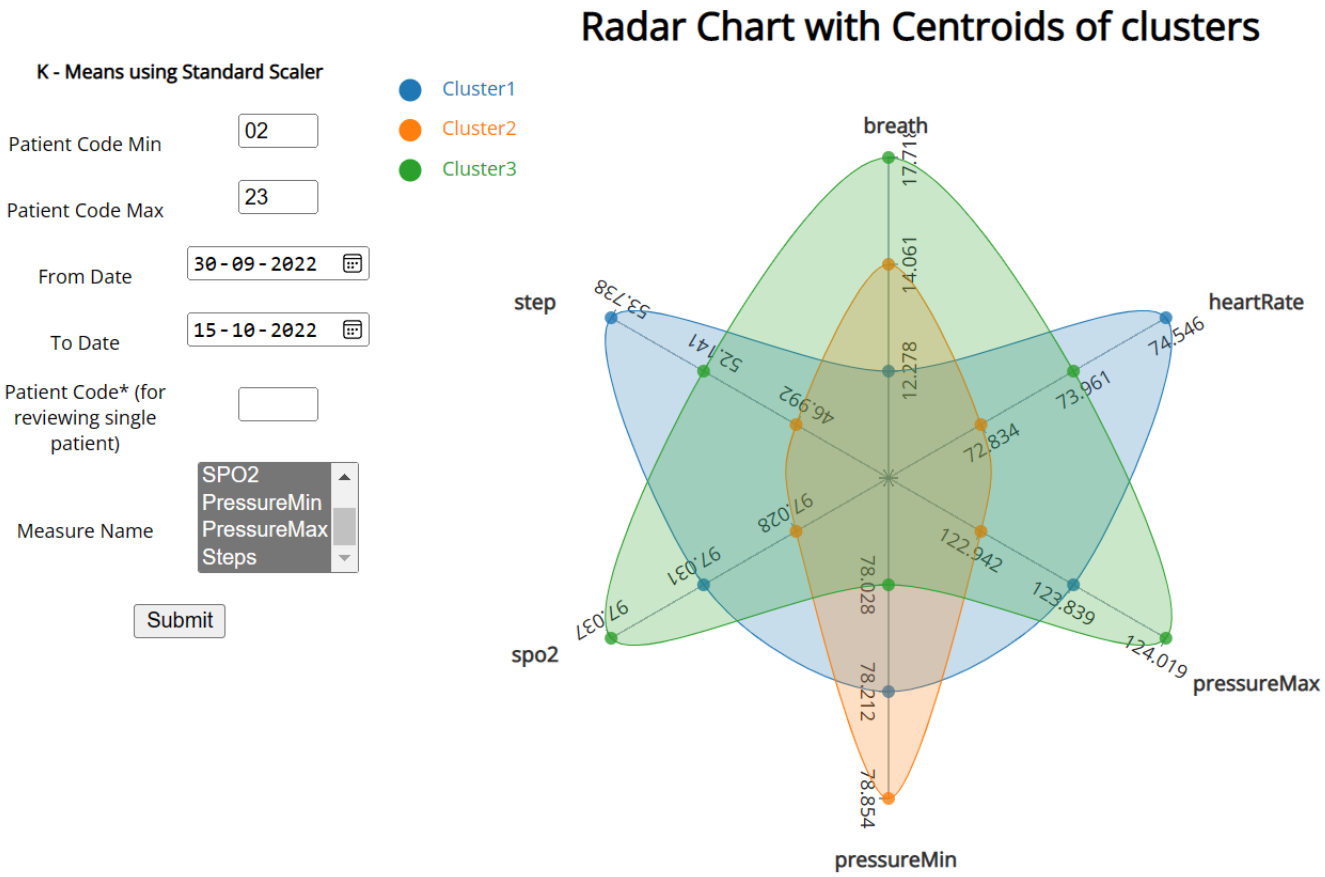


Figure 5.10: Clusters on data for the second period two weeks using K-means

5.4 Trend analysis with varying parameters

For visualizing trend analysis, we provide the user the ability to select the patient and the time period of evaluation. But there are other parameters that affect the behaviour of the results. These parameters can be modulated according to the data, but its usability for the user is not much. So we run a few tests to see how they affect the inference.

In the figures 5.11,5.12,5.13 and 5.14 we can see the variation in the parameters.

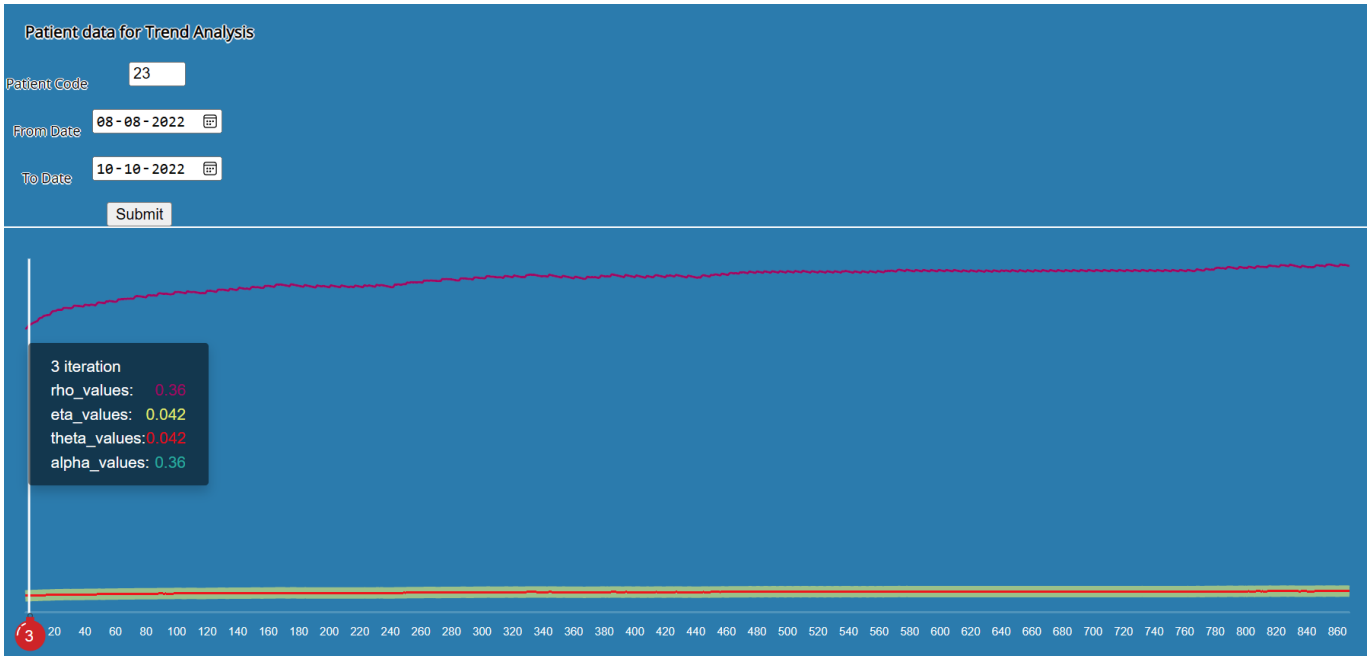


Figure 5.11: Variation in values of rho(ρ) for $\alpha = 0$

Alpha (α) when 0 represents the completely possibilistic method; while its completely probabilistic when $\alpha = 1$. In the iterations, alpha is re-computed with values of rho, to enable the algorithm to adapt according to the data. Due to this for $\alpha = 0$ values of rho and alpha are exactly the same in the plot and the lines overlap.

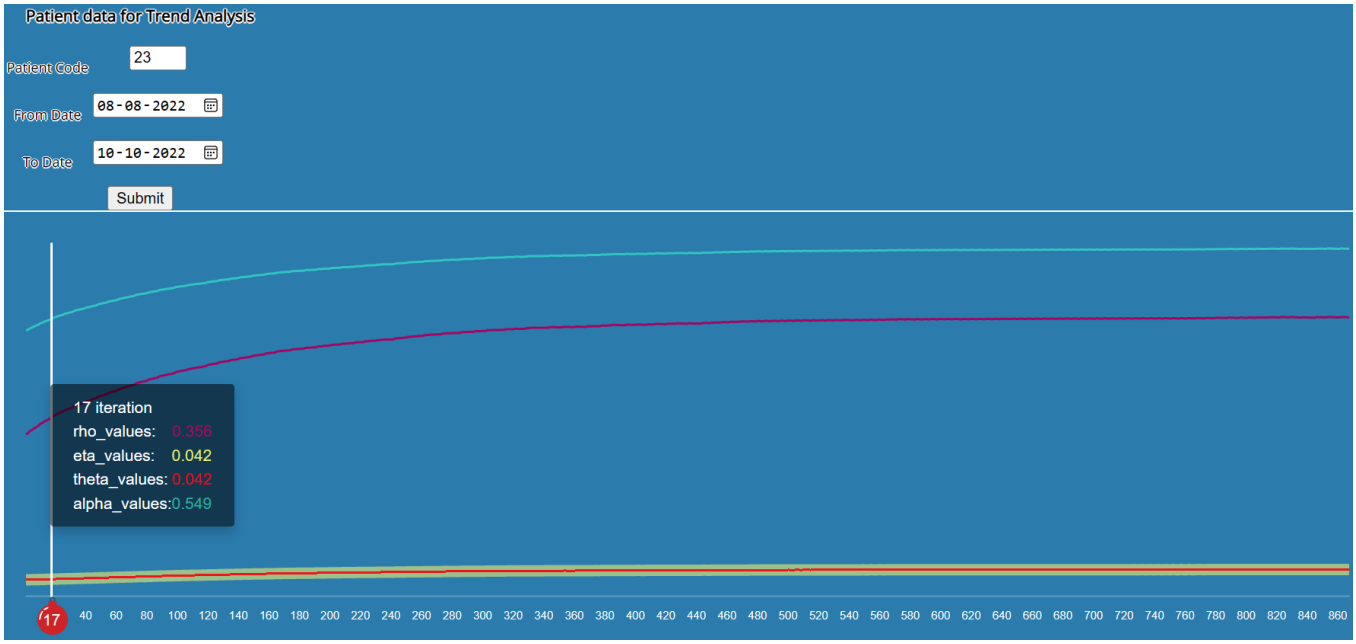


Figure 5.12: Variation in values of rho(ρ) for $\alpha = 0.3$

In the paper [MR03], the authors during their experiments observed that the intermediate values between 0.3 and 0.7 the model behaves in a graded possibilistic way. For the purpose of analysis the pattern those values were also included for testing.

For $\alpha = 0$ the outliers are completely rejected from consideration when defining the clusters and computing the centroids. While for $\alpha = 1$, the model is completely probabilistic and assigns one cluster to each observation even if it is an outlier. The clusters are created by keeping any and all outlier points in consideration. We can see in the plot on the figure5.14, the value of ρ is the least, since it does not interpret any outliers as outliers, rather a data point of one of the clusters.

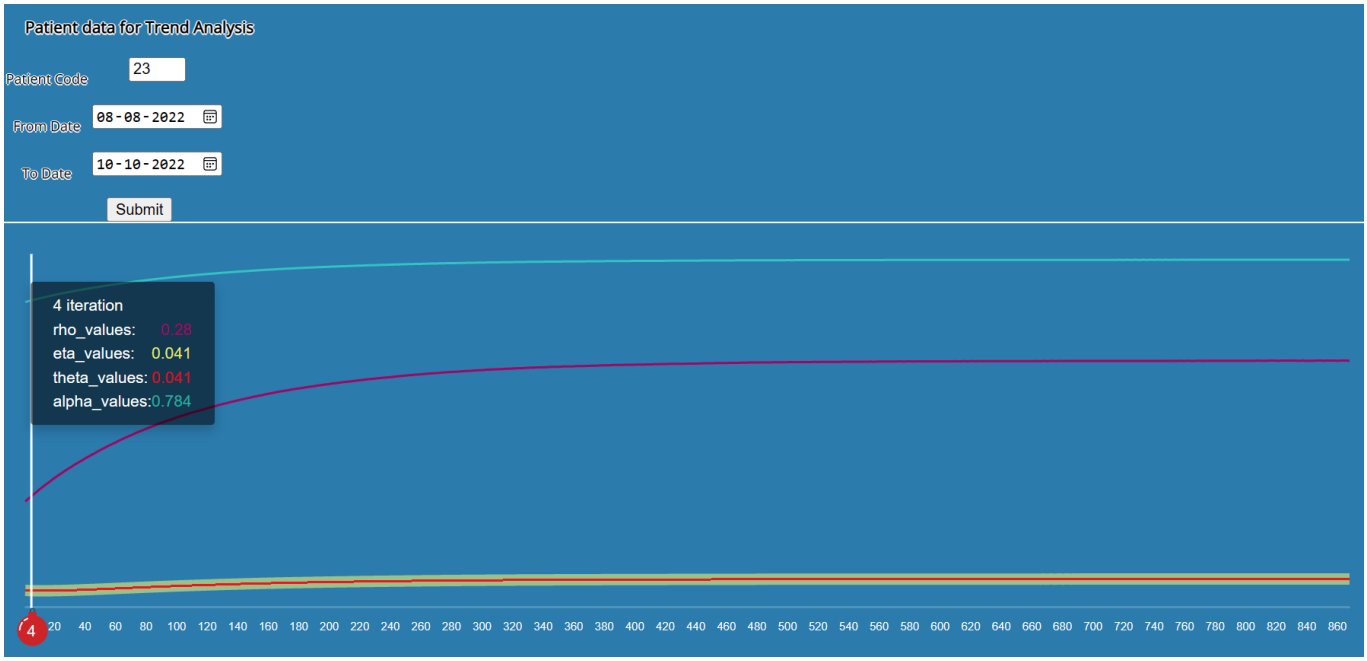


Figure 5.13: Variation in values of rho(ρ) for $\alpha = 0.7$

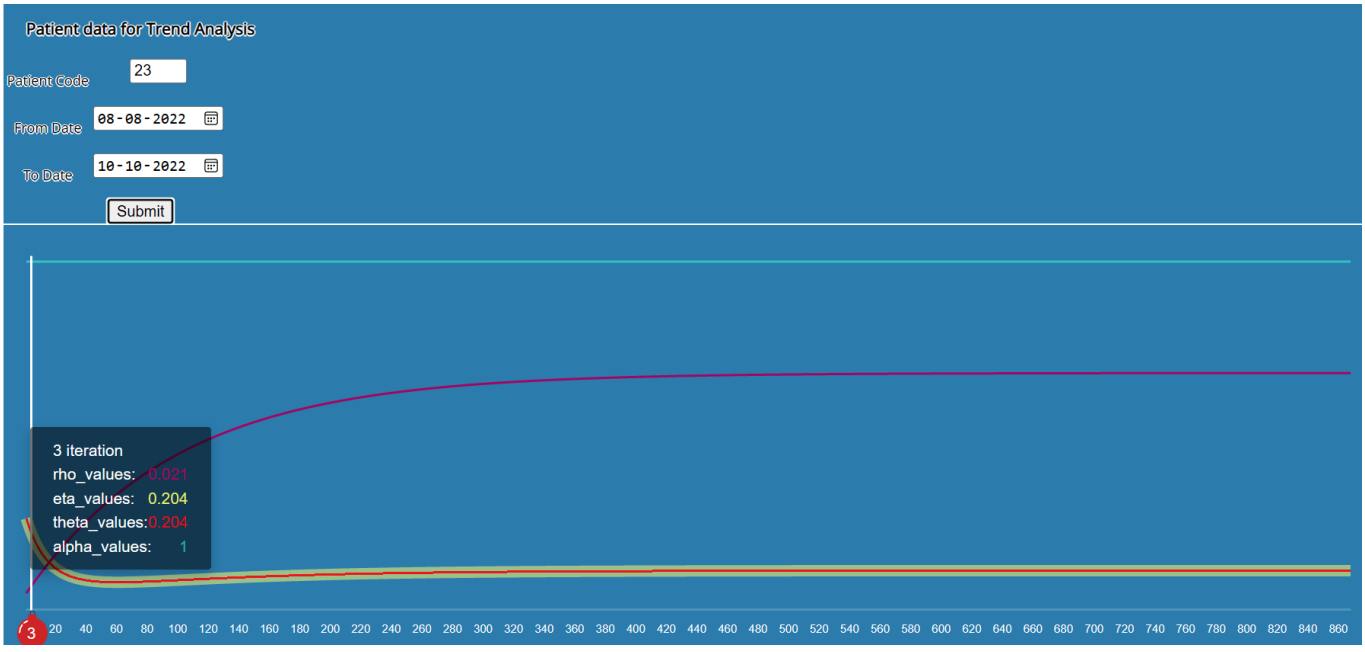


Figure 5.14: Variation in values of rho(ρ) for $\alpha = 1$

Chapter 6

Conclusion

During this thesis we got the opportunity to interact with both the technological and human elements of smart health monitoring systems. The IOT devices' capability of collecting data with minimal obstruction is the primary parameter in maintaining quality of life of the users. It allows us access to continuous and the most recent data. The data available for the project was of real patients due to which the consistency of data was not assured. The devices though with good battery life require re-charging and there can be days where the patient does not realise that and we lose the data. Since the data was of real people, to avoid security issues the data was fetched with API requests for each task, however that makes the processing time longer.
The interactions with the medical team gave an insight on their perspective of having the ability to have the access to see the important data at the right time. A lot of focus was put into the visualization of the data of the patients in a manner suitable for the healthcare providers. To ensure the quality of data, multiple ways for pre-processing the data were used before visualizing them. The user interface was also made user friendly and interactive to see multiple aspects of the data clearly, along with the ability to change the parameters according to the requirement.
The incessant metamorphosis of technology in every sector makes improvement inevitable, so the scope of improvement is present with a huge margin, but the general skeleton of processing and evaluating the data will remain the same. In due course there will be modified version of the IOT devices used which can make data collection more simpler. There is a need for ensuring the data collection process is without major lags as they affect the quality of results.

Bibliography

[Mac67] James MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics* 1 (Jan. 1, 1967), pp. 281–297.

[Dun73] J. C. Dunn. "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". In: *Journal of cybernetics* 3.3 (Jan. 1, 1973), pp. 32–57. DOI: 10.1080/01969727308546046.

[Llo82] S. P. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1, 1982), pp. 129–137. DOI: 10.1109/tit.1982.1056489.

[BEF84] James C. Bezdek, Robert Ehrlich, and William E. Full. "FCM: The fuzzy c-means clustering algorithm". In: *Computers & Geosciences* 10.2-3 (Jan. 1, 1984), pp. 191–203. DOI: 10.1016/0098-3004(84)90020-7.

[Jol86] I. T. Jolliffe. "Principal Component Analysis". In: *Springer series in statistics* (Jan. 1, 1986). DOI: 10.1007/978-1-4757-1904-8.

[KR90] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley-Interscience, Mar. 22, 1990.

[KK93] Raghu Krishnapuram and J. S. Keller. "A possibilistic approach to clustering". In: *IEEE Transactions on Fuzzy Systems* 1.2 (May 1, 1993), pp. 98–110. DOI: 10.1109/91.227387.

[MR03] Francesco Masulli and Stefano Rovetta. "The graded possibilistic clustering model". In: *International Joint Conference on Neural Network* (July 20, 2003). DOI: 10.1109/ijcnn.2003.1223483.

[Pal09] Pallets. *Quickstart — Flask Documentation (2.2.x)*. 2009. URL: https://flask.palletsprojects.com/en/2.2.x/quickstart/.

[BG13] Mirza Mansoor Baig and Hamid Gholamhosseini. "Smart Health Monitoring Systems: An Overview of Design and Modeling". In: *Journal of Medical Systems* 37.2 (Jan. 2013). DOI: 10.1007/s10916-012-9898-z. URL: http://dx.doi.org/10.1007/s10916-012-9898-z.

[Bai+14] Mirza Mansoor Baig et al. "Real-time vital signs monitoring and interpretation system for early detection of multiple physical signs in older adults". In: *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)* (June 2014). DOI: 10.1109/bhi.2014.6864376. URL: http://dx.doi.org/10.1109/bhi.2014.6864376.

[BGL15] Mirza Muhammad Faran Ashraf Baig, Hamid GholamHosseini, and Maria Salete Sandini Linden. "Tablet-based patient monitoring and decision support systems in hospital care". In: *International Conference of the IEEE Engineering in Medicine and Biology Society* (Aug. 1, 2015). DOI: 10.1109/embc.2015.7318585.

[Mer15] Gruppo la Meridiana. *Residenza Sanitaria Assistenziale (RSA): what it is and how it works*. Jan. 20, 2015. URL: http://www.gruppolameridiana.com/en/residenza-sanitaria-assistenziale-rsa-works/.

[Abd+16] Amr Abdullatif et al. "Graded Possibilistic Clustering of Non-stationary Data Streams". In: *Lecture Notes in Computer Science* (Dec. 19, 2016). DOI: 10.1007/978-3-319-52962-2_12.

[16] Victorlei. *GitHub - victorlei/smop: Small Matlab to Python compiler*. 2016. URL: https://github.com/victorlei/smop.

[Qua17] QuantEcon. *MATLAB–Python–Julia cheatsheet — Cheatsheets by QuantEcon documentation*. 2017. URL: https://cheatsheets.quantecon.org/.

[AAA18] Nabeel Salih Ali, Zaid Abdi Alkareem Alyasseri, and Abdulhussein Abdulmohson. "Real-Time Heart Pulse Monitoring Technique Using Wireless Sensor Network and Mobile Application". In: *International Journal of Electrical and Computer Engineering (IJECE)* 8.6 (Dec. 1, 2018), p. 5118. DOI: 10.11591/ijece.v8i6.pp5118-5126. URL: http://dx.doi.org/10.11591/ijece.v8i6.pp5118-5126.

[Esp+18] Massimo Esposito et al. "A smart mobile, self-configuring, context-aware architecture for personal health monitoring". In: *Engineering Applications of Artificial Intelligence* 67 (Jan. 2018), pp. 136–156. DOI: 10.1016/j.engappai.2017.09.019. URL: http://dx.doi.org/10.1016/j.engappai.2017.09.019.

[Dia19] Madson Luiz Dantas Dias. *fuzzy-c-means: An implementation of Fuzzy C-means clustering algorithm*. May 2019. DOI: 10.5281/zenodo.3066222. URL: https://git.io/fuzzy-c-means.

[Kot+19] Christos Kotronis et al. “Evaluating Internet of Medical Things (IoMT)-based systems from a human-centric perspective”. In: *Internet of Things* 8 (2019), p. 100125.

[McC19] McCormick. *Matrix Operations in NumPy vs. Matlab · Chris McCormick*. Oct. 28, 2019. URL: http://mccormickml.com/2019/10/28/matrix-operations-in-numpy-vs-matlab/.

[SR19] Erich Schubert and Peter Rousseeuw. *Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms*. 2019. URL: https://link.springer.com/chapter/10.1007/978-3-030-32047-8_16?error=cookies_not_supported%5C&code=eb0e2e1c-2b78-4ab8-ab69-52a514f16766.

[SPO19] SPOVAN- Shenzhen Tianpengyu Technology Co., Ltd. *High Quality ECG Smart bracelet Smart Band Spovan H03 Wholesale - Shenzhen Tianpengyu Technology Co., Ltd.* 2019. URL: https://www.spovan.com/products-detail-150445 (visited on 02/19/2023).

[TSM19] Adriano Tramontano, Mario Scala, and Mario Magliulo. “Wearable devices for health-related quality of life evaluation”. In: *Soft Computing* 23.19 (July 2019), pp. 9315–9326. DOI: 10.1007/s00500-019-04123-y. URL: http://dx.doi.org/10.1007/s00500-019-04123-y.

[Mor21] Amanda Iglesias Moreno. *Data normalization with Pandas and Scikit-Learn - Towards Data Science*. Dec. 15, 2021. URL: https://towardsdatascience.com/data-normalization-with-pandas-and-scikit-learn-7c1cc6ed6475.

[SR21] Erich Schubert and Peter J. Rousseeuw. “Fast and eager-medoids clustering: Runtime improvement of the PAM, CLARA, and CLARANS algorithms”. In: *Information Systems* 101 (Nov. 1, 2021), p. 101804. DOI: 10.1016/j.is.2021.101804. URL: https://doi.org/10.1016/j.is.2021.101804.

[CDC22] CDC. “Target Heart Rate and Estimated Maximum Heart Rate — Physical Activity — CDC”. In: (June 3, 2022). URL: https://www.cdc.gov/physicalactivity/basics/measuring/heartrate.htm (visited on 10/07/2022).

[22] Kno10. *GitHub - kno10/python-kmedoids: Fast K-Medoids clustering in Python with FasterPAM*. Sept. 24, 2022. URL: https://github.com/kno10/python-kmedoids.

[LS22] Lars Lenssen and Erich Schubert. *Clustering by Direct Optimization of the Medoid Silhouette*. 2022. URL: https://link.springer.com/chapter/10.1007/978-3-031-17849-8_15?error=cookies_not_supported%5C&code=670f5533-e54e-41c2-b4e0-0f273df9b8e8.

[MRD22] Masulli, Rovetta, and DIBRIS. *Machine Learning: A Computational Intelligence Approach*. May 2022. URL: https://person.dibris.unige.it/masulli-francesco/didattica/ML-CI-PhD/MLCI-2022.html.

[Mat22] Mathworks. *Install MATLAB Engine API for Python - MATLAB & Simulink - MathWorks Italia*. 2022. URL: https://it.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html.

[Num22] Numpy Developers. *NumPy for MATLAB users — NumPy v1.24 Manual*. 2022. URL: https://numpy.org/doc/stable/user/numpy-for-matlab-users.html.

[Sch22] Erich Schubert. *Fast k-medoids Clustering in Rust and Python*. July 7, 2022. URL: https://joss.theoj.org/papers/10.21105/joss.04183.

[Sha22] Rohit Sharma. *What is Clustering and Different Types of Clustering Methods*. Nov. 23, 2022. URL: https://www.upgrad.com/blog/clustering-and-types-of-clustering-methods/.

[Moz23] Mozilla. *Using the Fetch API - Web APIs — MDN*. Feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch.