TOWARDS HEALTHCARE AUTOMATION

Enrollment Numbers – 21303009, 21303012, 21303005, 21303006

Name of Students – Keshan, Kritika Bhatnagar, Himanshu Nigam and Aditi Jain

Dec – 2022

Submitted in partial fulfillment of the Degree of

Master of Technology

in

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

Table of Contents

Declaration

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Signature:

Date: 2nd Dec 2022 Name:

Enrollment No. :

Acknowledgement

Signature of the Students:

Name of the Students:

Enrollment Numbers:

Date: 2nd Dec 2022

Summary

In this project, two approaches for healthcare automation are discussed: (1) the first approach focuses on current healthcare automation system that is practically feasible but it is not user centric and requires human intervention to process the output besides being less accurate and less cost effective. (2) the second approach conceives the idea of user centric healthcare automation that focuses on how automated data collection through quantum sensors will process inputs to a novel automation pipeline through a wearable ring and how such a new automation system will radically transform healthcare to let everyone enjoy almost free healthcare whenever required.

List of Figures

1. Introduction – The Idea of Healthcare Automation

1.1. General Introduction

Emerging digital technologies are transforming healthcare by automating the processes but this leads to hospital centric healthcare for the benefit of few. The user centric healthcare automation will pave way for better online health monitoring, online diagnosis, personalized prescription and online patient care etc.

- 24x7 online health monitoring will help to diagnose any disease at the very beginning besides taking care of the patient remotely.
- Automated medical diagnosis will lead towards a more error-free process in diagnosing the various parameters and making efficient decisions.
- Personalized nutrition will encourage everyone to always keep fit since conception till death and enhance their creative potential to the maximum possible level.

1.2. Problem Statement

- Healthcare facilities are mostly unavailable to the underprivileged people in the whole world. They often depend on traditional ways of healthcare or take medicines on their own.
- Present healthcare systems can’t cater to the individual needs of everyone in a timely manner as it requires huge amount of resources including natural resources, human resources, capital etc. Management of a healthcare facility is itself a very tedious task in many ways.
- No holistic approach among the competing therapies of Allopathy and AYUSH etc.

1.3. Significance / Novelty of the Problem :

- Existing technologies for healthcare automation require data analyst to manually process the outcomes from one stage to the next stage, so automating entire machine learning pipeline is necessary for user centric healthcare.

- All the above mentioned problems can be handled through user centric healthcare as all the focus will shift from infrastructure centric healthcare to personalized healthcare where each individual will be able to monitor, diagnose, get goal oriented recommendations / prescriptions for food & medicines etc. This will gradually empower everyone in the whole world.

1.4. Empirical Study

We read several papers and found out that the current approaches are not synergistic with each other. The current approaches applied to various health problems advocate for isolated solutions without paying any heed to other solutions. This leads to disconnect and only few people can enjoy the ultra-high tech facilities. We are focusing on shaping a user centric healthcare automation system to let anyone enjoy personalized healthcare at almost zero cost whenever required.

1.5. Brief Description of the Solution Approach

There are two solution approaches; (1) Based on current automation technologies whose implementation is discussed at length and (2) A radically different innovative approach for user centric automation of healthcare that needs to be explored.

1.6. Comparison of existing approaches to the problem framed

The current approach for healthcare automation is industry centric for the benefit of few while the idea of user centric automation of healthcare is just conceived.

2. Literature Survey – A brief review of existing technologies for healthcare automation

2.1. Summary of Papers Studied

[1]. Reinforcement Learning is a subfield of machine learning that uses intelligent agents to take actions in an environment to maximize the rewards. A brief introduction of two fundamental concepts in RL is presented in the paper: theoretical foundation & basic solutions and advanced techniques.

These two concepts are further explained further with theoretical foundation & basic solutions constituting of framework, model based planning methods and model free learning methods while advanced techniques constitutes of efficient techniques and representational techniques.

RL focuses on sequential decision making with sampled, evaluative and delayed feedback simultaneously. These features make RL a force to reckon with for developing a variety of useful and powerful solutions in the healthcare domain. The paper discusses a number of broad applications of RL techniques in healthcare domains, namely:

1. Dynamic Treatment Regimes:
    a. Chronic Diseases
    b. Critical Care
2. Automated Medical Diagnosis:
    a. Structured Data
    b. Unstructured Data
3. Other General Domains:
    a. Drug Discovery
    b. Optimal Process Control
    c. Health Management
    d. Resource Scheduling & Allocation

[2]. Quantum computing is a type of computation that harnesses the quantum mechanics phenomena's such as superposition, entanglement and interference. The basic units of operation in a quantum computer (devices that perform quantum computations) are referred to as 'qubits' or quantum bits. The research encompasses several quantum healthcare applications like:

1. Molecular Simulations
2. Precision Medicine

3. Diagnosis Assistance
4. Radiotherapy
5. Drug Research and Discovery
6. Pricing of Diagnosis (Risk Analysis)

Quantum computing applications in healthcare require consideration of multiple factors for effective implementation. These factors are:

1. Computational Power
2. High Speed Connectivity (5G/6G Networks)
3. Higher Dimensional Quantum Communication
4. Scalability of Quantum Computing
5. Fault-Tolerance
6. Quantum Availability of the Healthcare Systems
7. Deployment of Quantum Gates
8. Use of Distributed Topologies
9. Requirements for Physical Implementation
10. Quantum Machine Learning

In addition to this, the concept of security of quantum healthcare computing is also discussed in the said paper. There are six avenues of security that are discussed for healthcare information processing, namely:

1. Quantum Key Distribution (QKD)
2. Defense using D-Level Systems
3. Defense Against General Security Risks
4. Defense using Finite Key Analysis Method
5. Measurement-Device-Independent Quantum Key Distribution
6. Semi-Quantum Key Distribution

[3]. This paper opens with a brief overview of a patient encounter in the age of quantum computing and describes the experience of the said patient suffering from breast cancer. How a quantum imaging software can help in identification of cancer cells is simply demonstrated.

Next we move on to the advantages of quantum computing over our current computer technology. It is already established that quantum computing provides an exponential speed up through quantum algorithms over its counterpart classical algorithms.

And lastly, what changes can we expect to see if quantum computing is made available for daily use. Quantum computing can improve imaging, diagnosis, treatment and population health. The paper closes with the argument that although quantum computing has tremendous applications to healthcare but it when will it be available for daily use in still in question and till then quantum computing will remain the domain of researchers, not physicians.

[4]. To predict the status of hypertension and diabetes using the patient's glucose and blood pressure readings by training the system using machine learning algorithms (Support Vector Machine, k-Nearest Neighbors, Decision Tree, Logistic Regression & Discriminant Analysis) along with sending categorized alerts and real-time notifications to their registered physician or clinic all from home.

In the proposed system, blood pressure diagnosis is made after analysis of the data which is received as input of the blood pressure readings via user interface application. A notification can be sent by the application itself to the health care provider in case of abnormalities detected.

This application also accepts blood glucose level which predicts if the person is diabetic or non-diabetic and a notification can be sent to the physician in case of abnormal blood pressure levels or onset of diabetes.

Supervised machine learning classification approach is used to train the dataset in which 80% of the dataset is used for training and 20% for testing in order to predict if the person is diabetic or not and onset of diabetes based on the reading of blood pressure and blood glucose levels while traditional approach is used to output the blood pressure category based on the blood pressure readings. The results if comes out to be abnormal a notification to the physician via email is sent by the patient. Support Vector Machine classification algorithm had a better accuracy over other algorithms.

[5]. People with intellectual disabilities face a lot of problems and to cater these problems, smart watches can be used to provide early warnings for medical emergencies. Also, these devices can be used to detect the location of the person to detect their wandering behavior.
A notification is sent automatically to the user as well as the caregiver when needed. Use of smart watches that can detect heart rate, oxygen saturation and the location of the person are used. This data is resided on the web server and a notification is sent to the caregiver if any abnormality is detected.

[6]. The proposed system uses sensors to collect health related parameters (heart rate, blood glucose, temperature & uterine contractions) of pregnant ladies in order to determine the pregnancy related risks. Visual intimation to the healthcare providers is done so that appropriate actions can be taken in case of emergency situations. The system allows the registration of the patient and her various health care providers.
The four designated sensors sense the respective parameters that are heart rate, blood glucose, body temperature & uterine contraction. Data analysis is done using SVM classification algorithm to provide the health status of the person. Notification is sent to the health care providers in case of any critical change in the health status of the person.

[7]. A system, HealthCloud, is proposed to monitor the heart patient's health status using machine learning algorithms (SVM, k-NN, NN, Logistic Regression & Gradient Boosting Trees) and cloud computing. Various parameters and cross validation (5-fold) techniques are used to compare the different machine learning algorithms.
Testing of algorithm and mobile application is done with existing user inputs from dataset and unseen data as well. Self-diagnosis and monitoring is done to assist the patients using this system. Use of AI helps in early detection of the disease by finding the patterns in the data of the previous patients.
The performance of these machine learning algorithms is done using the metrics of accuracy, recall, specificity, AUC scores, execution time, latency & memory usage. The algorithm and the mobile application were tested on Google Cloud Firebase. Logistic Regression algorithm has a better accuracy over other algorithms.

[8]. This research paper suggested the applicability of the screening score prediction models for use in clinical settings related to personalized risk assessment and targeted interventions, with the predictors used being suitable for either the clinic or hospital. The simplicity of Gender, Age, Body Mass Index (BMI), Blood Pressure and Waist Circumference as predictors suggested that they can be utilized by the community.

[9]. This research paper stated the importance of BMI in predicting Diabetes .There are different risk factors such as modifiable and non-modifiable associated with diabetes and obesity. Early detectable markers are not well established to detect pre-diabetes and as a result, it becomes diabetes. Recent epidemiological research revealed that lifestyle activities, diet, lack of physical activity and family factors, including genetic factors that play a major role in diabetes mellitus and obesity. Incorporating positive lifestyle changes that include a healthy diet and physical activity can result in preventing diabetes. Although, there is research that does identify that diabetes can occur based on some non-modifiable risk factors such as genetics, understanding ways to prevent or delay diabetes through diet and exercise needs to be placed on the priority list in educational sessions that are provided to individuals with pre-diabetes.

[10]. This research paper showed that blood urea is directly proportional to the diabetes whereas creatinine did not. The correlation between blood sugar and serum urea levels in diabetics has shown a strong dependence as evidenced by “r value”. The “r value” for the fasting blood sugar and serum urea levels was 0.76 and that of post prandial blood sugar and serum urea was 0.83 thus establishing a strong relationship between them. However, the association between hyperglycemia and the serum creatinine levels showed a weaker link.

[11]. In this paper, the highest accuracy obtained in the testing stage corresponds to logistic regression model, which achieves an accuracy of 0.729 with a sensitivity of 0.852 and 0.795 in AUC metric, so there is no big difference between this model and random forest model, due to its performance it indicates an accuracy of 0.704, a sensitivity value of 0.781 and 0.776 in the AUC metric, another model with high performance is neural network. On the other hand, the paper also demonstrated that it is possible to identify subjects who had diabetes treatment from those who didn’t had diabetes treatment, showing the relevance of cholesterol, HDL, LDL and triglycerides features.

The models obtained can be useful for doctors to know if patients are following the established treatments correctly or simply know if a new patient has been taking an anti-diabetic drug, allowing for a better control of the patient's medical history.
Furthermore, according to subjects that are taking sulphonyl urea therapy, have observed effects on their lipids profile, also a group treated with insulin along with metformin had significant improvement in the lipids levels. Because, once the Hemoglobin A1c (HbA1c) is improved through diabetic treatment, the lipids profile can significantly improve.

[12]. In this paper, a system for menu generation PREFer (Prescriptions for REcommending Food) is described. PREFer uses annotations and dataset of recipe for recommending menus as per preferences of user. PREFer provides user with the healthy and personalized menus while it takes into account both medical prescriptions and short or long term based preferences of user. This classifies nutrition behavior ideally for user from view point of health.
An innovative algorithm is the basis of recommendation method to rank and generate menus to ensure high variety in recipes recommended and provide better performance. In present implementation of this system, user is able to register the frequency of recommended menu.

[13]. In this paper, researchers have analyzed the data (involving 46,898 meals and 800 subjects) and devised an algorithm which integrates body composition, physical activity, micro biome, dietary habits and blood testing. This algorithm accurately predicts personalized glycemic response (rise in level of blood sugar after eating a meal) to that of real-life based meals.
The algorithm gives dietary recommendation(s) which notably lower the level of blood sugar after consuming a meal. New diet concluded in continuous changes in microbiome. The findngs state that the personalized diets could prevent spike in blood sugar level which is correlated with outbreak of metabolic problems (obesity, cardiovascular disease, diabetes, inflammation) by recognizing dietary regimens of specific individual.

[14]. This paper discusses "Waan-Noy" chatbot for meeting the need of individuals who are diabetic. Waan-Noy chatbot uses Dialogflow technology which assists the computers in understanding human language from the textual data. The chatbot system structure is divided into two parts: User Interface and Backend. Some of the features of this chatbot are:

1. Food Recommendation
2. Data Collection
3. Nutritional Information
4. Suggestions
5. Report

The chatbot benefits a user in ease of use for choosing menu, food recommendation, gaining awareness about health, also on the basis of nutritional needs profile of individual. Concept of recommendation is based on Basal Metabolic Rate (BMR) which represents needed calories/day.

The chatbot uses the following ratio for recommending food to help in curing the cause and effect of diabetes.

$$\text{BMR} = 66 + (13.7 \times \text{Weight(kg)}) + (5 \times \text{Height(meter)}) - (6.8 \times \text{Age})$$

2.2. Integrated Summary of the literature studied

All the research papers focus on how healthcare is associated with various fields like quantum computing, reinforcement learning, machine learning, internet of things, big data etc. A brief introduction to different medical problems is described in the papers and how those problems can be solved through various technologies.

3. Requirement Analysis and Solution Approach

3.1. Overall Description of the Project

Monitoring in medicine is the process of keeping track of a condition, disease, or one or more medical parameters throughout time. It can be done by continually running medical tests or by continuously monitoring particular parameters using a medical monitor (such as continuously monitoring vital signs with a bedside monitor) (such as blood glucose monitoring with a glucometer in people with diabetes mellitus). The target of interest can be used to categorize monitoring, including:

- Continuous electrocardiography with evaluation of the patient's state in relation to their heart rhythm is known as cardiac monitoring. A Holter monitor is a tiny monitor used for this purpose that is worn by an ambulatory patient. Cardiac monitoring may also entail invasive Swan-Ganz catheter cardiac output monitoring.
- Hemodynamic monitoring, which keeps an eye on the circulatory system's blood flow and blood pressure. An implanted blood pressure transducer assembly or an inflated blood pressure cuff can be used to measure blood pressure invasively or noninvasively.
- Respiratory monitoring includes techniques like pulse oximetry, which measures the blood's saturation level of oxygen, or SpO2, using an infrared finger cuff.
- Using a glucose metre to measure the amount of glucose in the blood is known as blood glucose monitoring (glycemia). A blood glucose test is normally conducted by piercing the skin (often via fingerstick) to extract blood, then putting the blood to a chemically active disposable "test-strip," which is particularly significant in the management of diabetes. The principal alternative is ongoing glucose monitoring (CGM).
- Monitoring of body temperature using an adhesive pad with a thermoelectric transducer. Measuring vital signs can involve all of the aforementioned factors, but is most frequently done with at least blood pressure and heart rate monitoring, as well as pulse oximetry and respiration rate if possible.

Measuring vital signs can involve all of the aforementioned factors, but is most frequently done with at least blood pressure and heart rate monitoring, as well as pulse oximetry and respiration rate if possible. In this project, the target parameters that have been chosen as temperature and pulse.

Diagnosis: The process of identifying the sickness or condition that accounts for a person's symptoms and indicators is known as medical diagnosis. It is most frequently referred to as a diagnostic, with the implied medical context. The history and physical examination of the patient who is seeking medical attention usually yield the data needed for diagnosis. The process frequently includes one or more diagnostic procedures as well, such as medical tests. A diagnosis, in the sense of a diagnostic method, can be thought of as an effort to categorize a person's condition into discrete and distinct groups that enable medical professionals to make

judgments regarding treatment and prognosis. The diagnosis is therefore frequently explained in terms of a sickness or other condition.

3.2. Requirement Analysis

Components Used:

- Arduino Uno
- Temperature sensor
- Pulse sensor
- Jumper wires
- USB Cable

3.3. Solution Approach

Current Approach:

(a) Working principle of Pulse Sensor:

Pulse sensor is also known as a heartbeat sensor or a heart rate sensor. Connecting this sensor from the human ear or fingertip to an Arduino board will enable it to function such that it is simple to compute heart rate. A 24 inch color coding wire, an ear clip, Velcro Dots 2, transparent stickers 3, etc. are all included with the pulse sensor.

Header connectors are linked to a color-coded cable. Therefore, there is no need for soldering to connect this sensor to an Arduino in the project. The heart rate sensor can have an ear clip attached to the rear using hot glue so that it can be worn on the earlobe. On the hook side, two Velcro dots are fully sized toward the sensor. These come in very handy for creating a Velcro strap that covers about a fingertip. This is used to cover the finger's sensor.

Protective coatings called transparent strikers are utilized to shield the sensor from perspirant fingers and earlobes. This sensor has three holes along the external border so that anything can be connected conveniently. The three pins that make up the heartbeat sensor are detailed below:

- Pin 1 (GND) is attached to the system's GND terminal.
- Pin 2 (VCC): This pin is connected to the system's supply voltage (+5V unless otherwise specified, +3.3V).
- The pulsating o/p signal is attached to pin 3 (Signal).

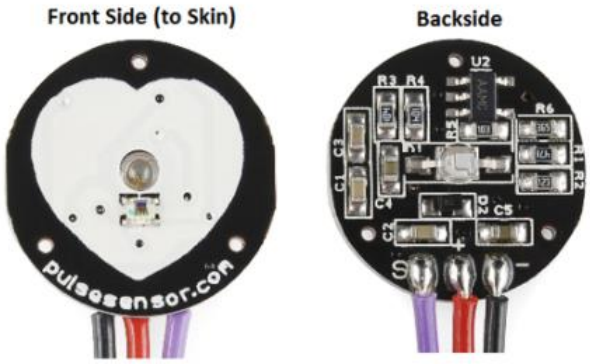


Figure 1: Pulse Sensor

The pulse sensor's operating system is fairly straightforward. The ambient light sensor and light-emitting diode are connected on the first of the sensor's two surfaces. Similar to that, a circuit responsible for noise cancellation and amplification is attached to the second surface. The LED is situated above a vein in the human body, such as an ear tip or fingertip, but it must be situated directly on top of a layer. The LED begins to emit light once it is placed on the vein. When the heart begins pumping, blood will start to flow through the veins. So, if we measure blood flow, we can measure heart rates as well. The ambient light sensor will activate if the blood flow is detected.

(b) Working principle of temperature sensor (MLX90614):

A specific object's temperature can be measured with the MLX90614 Contactless Infrared (IR) Digital Temperature Sensor between -70°C and 382.2°C. The sensor communicates with the microcontroller using the I2C protocol and measures the object's temperature using IR rays without making any physical contact. The contactless IR temperature sensor with great precision that is the MLX90614's distinguishing characteristic.

As a result, it can be utilized in industries to gauge the temperature of spinning motor shafts and other moving objects. Due to its great accuracy and precision, it is also utilized in a variety of commercial, medical, and domestic applications, such as the monitoring of ambient temperature and the measurement of body temperature.

Figure 2 : MLX69014 Sensor

It is already established that the MLX90614 sensor can determine an object's temperature without coming into contact with it. The Stefan-Boltzmann Law states that all objects and living things radiate infrared radiation and that the intensity of this energy is directly inversely related to the temperature of the item or living thing, makes this feasible. In order to determine an object's temperature, the MLX90614 sensor measures the amount of IR energy it emits.
The sensor may be directly interfaced with a microcontroller like Arduino and doesn't need any other components. The power pins (Vin and Gnd), as seen above, can be used to power the sensor directly. Typically, 5V can be used, but there are various variants of this sensor that can function on 3.3V and 7V as well. In order to filter noise and provide the best EMC, the capacitor C1 is an optional component. SCL and SDA, the signal pins used for I2C communication, can be directly linked to a microcontroller running on 5V logic.

Problems with the Current Approach:

- Current approaches require continuous human intervention (data experts) to run any pipeline, so it is not user centric.
- A human expert can apply a fixed number of algorithms only due to cognitive limitations, so the solution might be less accurate.

- Manual processing of algorithms is a time taking process that matters a lot during emergency health conditions.

A Radically Different Innovative Approach:

This approach collects data automatically by using the quantum sensors and processes the inputs in a novel pipeline through a wearable ring. The sequence of processes in this novel automation pipeline are as follows:

(a) Data Collection by the Quantum Sensors: Quantum Sensors work on the principle of quantum entanglement, quantum interference and quantum state squeezing with optimized precision and beat the current limits of sensor technology. Quantum sensing is a technological field which deals with the design and engineering of quantum sources and quantum measurements that can beat any classical strategy in various applications. Quantum Sensors are often built on continuously variable systems (quantum systems characterized by continuous degrees of freedom such as position & momentum). Basic working mechanism of quantum sensor typically relies on optical states of light (involving mechanical properties such as squeezing or two-mode entanglement).

Application areas of quantum sensors in wide variety of fields include:

- Microscopy
- Positioning Systems
- Communication Technology
- Electric and Magnetic field sensors

Quantum Sensors have potential applications in the field of medicine such as personalized medicine, biological process mapping, early cancer detection, nanotechnology, 3-D mapping of individual protein molecules etc. Photomultiplier tubes (PMT) are specifically used in the field of medical imaging.

(b) Inputs to the Pipeline through Wearable Ring:

Wearable ring will be an input-output device that will connect the quantum states using the principles of superposition, entanglement and teleportation etc for communication and data processing. Data collected by the quantum sensors and inputs given to the ring by the user will process through the novel automation pipeline for the most effective outcome.

(c) Processing Inputs in the Pipeline:

- Initialization of automation software engine, followed by
- User defined data collection and labelling or Automated data collection and labelling, followed by
- Automated data cleaning, followed by
- Automated feature engineering, followed by
- Data driven dynamic optimization of hardware resources for automated generation of quantum algorithm. This involves multi-objective optimization of requirement engineering according to the data and availability of the hardware to select / generate the most effective classical / quantum algorithm, followed by
- Automated selection of next system according to time schedule or the process is terminated.

This new pipeline will eliminate the need of processing any task manually as all the modules mentioned above will automatically process the task in a coordinated manner. The output for one module will automatically become the input for the module next in line and there will be no need of any human intervention (to give inputs) to manually run a particular module.

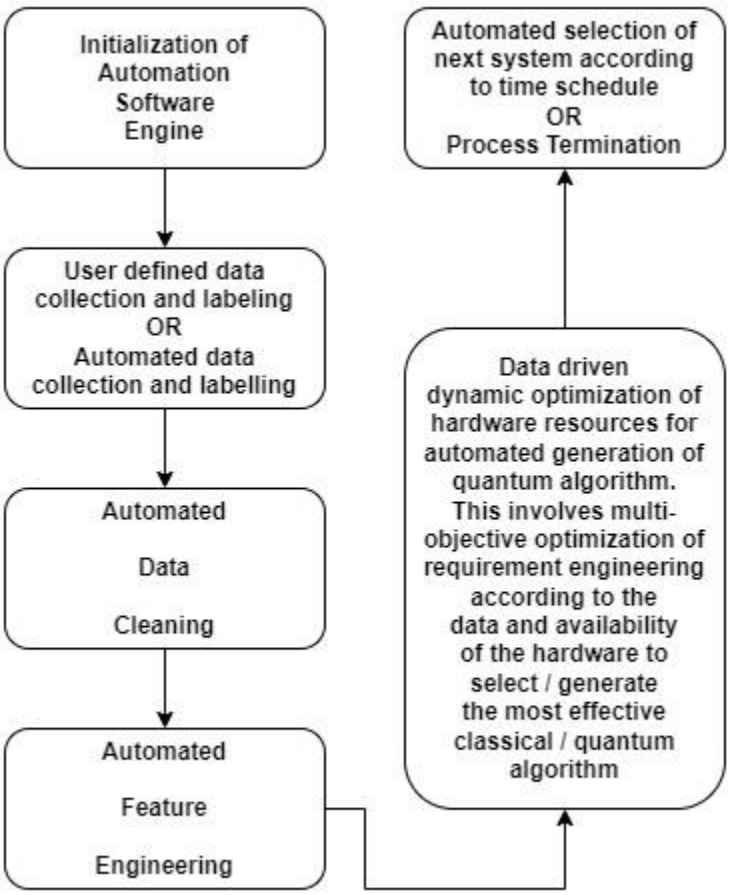


Figure 3 : New Automation Pipeline

How User Centric Automation Will Radically Transform Healthcare:

24 x 7 Online Monitoring: Now quantum sensors are succeeding to take the most accurate data from the remotest areas including the processes inside a human cell. These exploit the quantum principles of superposition and entanglement to collect minute changes in the electric and magnetic field of any quantum particle that has motivated the researchers to observe changes in the cellular processes. The idea is that the quantum sensor will capture even a small change in the cell when it gets infected or inflamed or its cellular process disrupts. This leads to important practical applications to observe changes in any single cell or organ or whole body just by wearing a ring made of quantum sensors that will continuously observe the changes in the electromagnetic field and biochemical changes in the body and will notify as soon as there

is abrupt change in any parameter like Blood Pressure, SpO2, Heart Rate / Pulse, Stroke, Epilepsy, Behavioral Disorders, Infection (CoVid-19, HIV, STDs etc.), Radiation, Poisoning etc. This way any one will be continuously automatically monitored without any worries of succumbing to deadly diseases or any unforeseen circumstances etc. This technology is leading to revolution in healthcare as immediate cure will stop the development of any disease at the very beginning.

Automated First-Aid: When there is an abrupt change in any health parameter or a sudden change in the biochemical composition of any cell, the quantum sensors in the wearable ring will immediately sense the problem at the root level (i.e. changes in the electric field, magnetic impulse, mechanical properties etc.) and will administer the required precautionary medicine from the ingestible chip besides sending notifications to the family members and the healthcare expert(s) e.g. when the first cell gets infected from any deadly virus like Corona or HIV or STD then the quantum sensors in the wearable ring will immediately detect abrupt changes in the cell and will instruct the ingestible chip to release the medicine to create a shield outside the infected cell to stop spread of the virus. Similarly, when any person gets exposed to radiation or poisoning or when (s)he faces abrupt changes in blood pressure, heart rate, sugar level etc. then the precautionary medicine will get released from the ingestible chip that will save her / him for 6-8 hours, so (s)he will find enough time to get proper healthcare. This way neither any disease will develop nor the patient will suddenly die.

On demand Diagnosis: When a person wants a detailed diagnosis of any health parameter or organ or whole body, then (s)he will swallow a capsule made of biomolecules, plants, and herbs etc. This capsule will contain the quantum sensor that will bind to the particular cell / organ to be diagnosed. Then, the patient will select the diagnose button from the ring that will connect the quantum states of the ring with that of the quantum sensor bound with the cell. Now the quantum sensor will send the visual data of that particular cell / organ to be diagnosed to the pipeline through the ring. The new inputs will generate the required quantum algorithm to produce the most accurate diagnosis.

Personalized Prescription and Production: Since continuous monitoring and on demand diagnosis will signal the development of any disease at the very beginning, there will be almost

no need of medicines. Everybody will remain healthy and disease free by enjoying personalized food, lifestyle, nature care etc. Recommendations for detox & cellular regeneration, personalized nutrition, medicine, yoga, aerobics, dance, sound sleep, electromagnetic field energy, hydro therapy etc. is a multi-objective optimization problem which involves various permutations & combinations in the pipeline to suggest according to the personalized needs and goals of the user. The portable health machines will produce only those items and services that will be finalized by the healthcare expert.

Intensive Care through Automated Bed at Home: Today large number of patients die due to delay in healthcare support but since the patient will be continuously monitored online in the new system, the automated bed at home will immediately provide emergency support like CPR, oxygen support, pre-decided medicine etc. to save life. This way the situation will be under control immediately than worsening due to delay in diagnosis and un-availability of the doctors. Since the online monitoring of even a quantum particle inside the cell will be visually analyzed, the impact of holistic medicines can be accessed in time. This will motivate the healthcare experts from various therapies to try and test the impact of combined medicines e.g. a proper combination of allopathic medicine, homeopathic medicine, nutrition, magnetic energy, pranic healing etc. If the administered medicines show any kind of negative effect then the visual data will help the healthcare experts to address those issues with appropriate changes (e.g. giving an anti-dote, improving the combination of various medicines or changing the medicine(s) altogether etc.) immediately.

Intensive Critical Care and Near Death Experience at Automated Health Center: Once the intensive care through automated bed at home fails, the patient will be shifted to the automated healthcare center where the most advanced technologies will try to save her / him. Now, some novel initiatives are being experimented to revive the dead person e.g. Dr Sam Parnia succeeds to bring the dead back to life. He argues that the chances of reviving the dead back to life will be more if we automate the following:

- Cardiopulmonary Resuscitation
- Targeted Temperature Management
- Extracorporeal Membrane Oxygenation

- Brain Oximetry
- Prevention of Reperfusion Injury

The automated healthcare center can also try personalized family therapy, ionized water therapy, vibration therapy, sound therapy, mantra therapy etc. Mrit Sanjeevni Vidya, Maha Mrityunjay Jaap, Tantrik Practices etc. can also be personalized by the pipeline and implemented with the most effective machines.

This new type of monitoring, diagnosis, personalized prescription will empower everyone to automatically keep track of health parameters anytime and reduce their dependence on infrastructure centric healthcare systems.

4. Modeling and Implementation Details

Here we present a brief outline of implementing the current solution approach while the elaboration and implementation of a radically different innovative approach is a matter of future research.

4.1. Design Diagrams

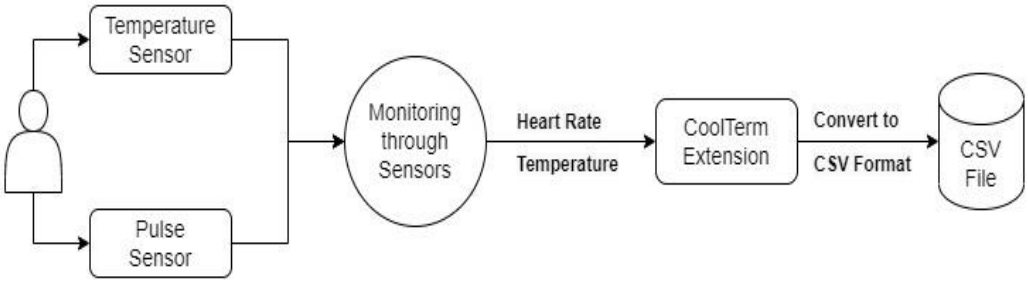


Figure 4: Monitoring Module Design

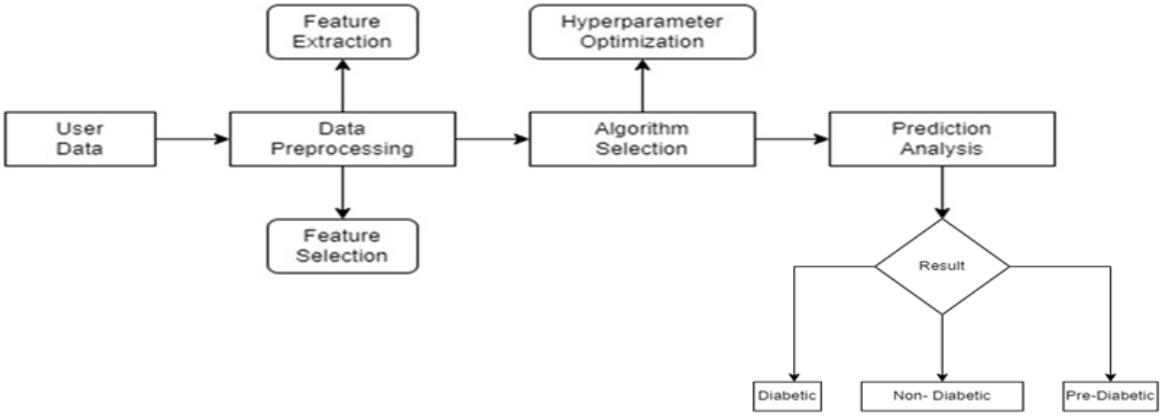


Figure 5: Diagnosis Module Design

4.2. Implementation Details and Issues

Monitoring of Pulse and Temperature Dataset:

- The pulse module can be supplied with either 3.3V or 5V.
- Positive voltage is connected to '+', while ground is connected to '-'. The third 'S' wire is the analog signal output from the sensor, which will be connected to the Arduino's A0 analog input.
- For the temperature sensor, connect the GND to the shared power / data ground. PWR should be connected to the power source, which for the 3V sensor is roughly 3.3V. Use approximately 5VDC for the 5V version.
- Connect Arduino's SDA pin to the I2C data SDA pin. This is also known as A4 on Arduino UNO & '328-based boards, digital 20 on Mega, and digital 2 on Leonardo/Micro boards.
- Connect Arduino's SCL pin to the I2C clock SCL pin. This is also known as A5 on an Arduino UNO & '328-based board, digital 21 on a Mega, and digital 3 on a Leonardo/Micro.

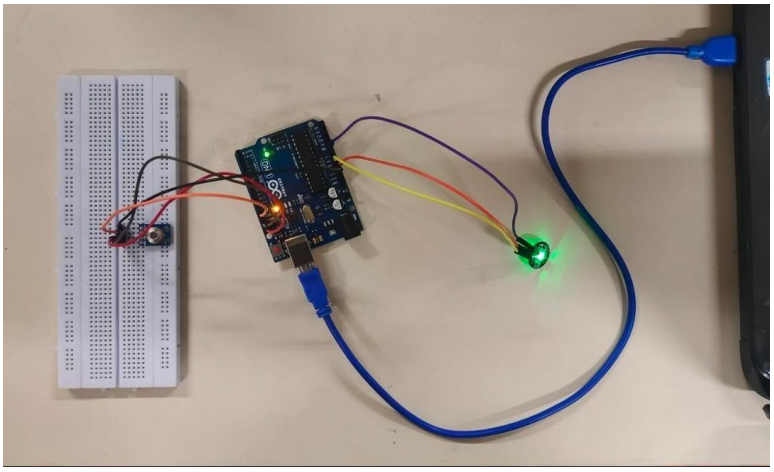


Figure 6: Pulse and Temperature Monitoring

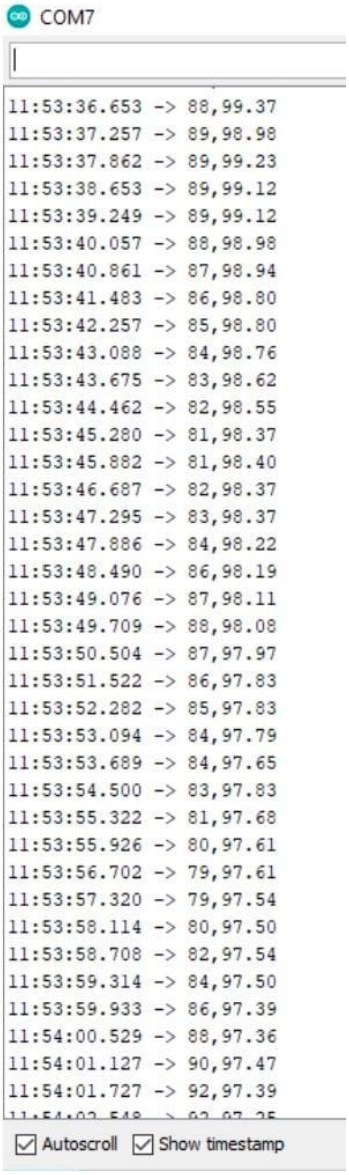


Figure 7: Output of Pulse and Temperature

Diagnosis of Pulse and Temperature Dataset:

- Import the basic machine learning libraries like Pandas & Numpy (for numerical calculations) and Matplotlib & Seaborn (for data visualization).
- Read the dataset (which is in csv format) and display five rows of data.
- Describing the detailed comparison of different measures (like mean, standard deviation, minimum values and maximum values) for the parameters present in the dataset.
- Infer the information whether the dataset parameters are of which data type (integer type, float type or any other type).
- Check for null values and missing values in the dataset as part of data cleaning. We find out that there are no null values as well as no missing values in the dataset.
- Use the Seaborn library to visualize the different parameters of the dataset and perform univariate analysis as well as bivariate analysis on these parameters.
- Plot a histogram plot and box plot for each of the 3 parameters namely Oxygen, Pulse Rate and Temperature. Calculate the mean and median of all the above mentioned parameters. Plot a count plot and a pie plot for the class label namely 'Result'.
- Finding correlations between the above mentioned 3 parameters and the class label. A 4 x 4 matrix is formed showing the correlations between each parameter. All the parameters have a correlation of 1 with themselves. Each parameter (3) and class label has either a positive correlation or a negative correlation with one other.
- Positive correlation between any two parameter indicates that they are highly influenced with each other and they increase & decrease synonymously while negative correlation indicates that they are inversely influenced with each other and act opposite i.e. if one parameter increases then other one decreases and vice versa.
- Plot a correlation matrix with help of Heatmap and annotate all the values. Heatmap helps in visualizing the correlations. Sort the correlation values of all the 11 parameters and the class label. We find out that Pulse Rate and Temperature are positively correlated and Oxygen is negatively correlated with the class label.
- Moving on to the prediction analysis, we drop the parameter Oxygen. Storing the remaining parameters in a separate variable X and storing the class label in another separate variable y. This is necessary for dividing the dataset into training data and testing data.

- Import a very important library – Sklearn (used for data analysis). This library will help in splitting the dataset into training data and testing data. We have split the dataset in the ratio of 75% : 25% i.e. 75% training data and 25% testing data.
- Applying XGBoost, Decision Tree and Support Vector Machine. Checking accuracy of each algorithm and making predictions for the test dataset.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn.metrics import confusion_matrix, accuracy_score
        from sklearn.metrics import classification_report
```

```
In [3]: df = pd.read_csv("HR & T.csv")
        df.head()
```

Out[3]:

	ID	Oxygen	PulseRate	Temperature	Result
0	0	98	65	95	0
1	1	96	92	95	0
2	2	95	92	99	0
3	3	97	56	96	0
4	4	88	94	98	1

```
In [4]: target = df.drop(['ID'], axis=1)
        target
```

Out[4]:

	Oxygen	PulseRate	Temperature	Result
0	98	65	95	0
1	96	92	95	0
2	95	92	99	0
3	97	56	96	0
4	88	94	98	1
...	...	...	...	...
9995	95	124	97	0
9996	88	70	100	1
9997	99	56	105	0
9998	92	49	98	1
9999	85	52	99	1

10000 rows × 4 columns

```
In [5]: target.describe(include='all')
```

Out[5]:

	Oxygen	PulseRate	Temperature	Result

In [5]:
```
target.describe(include='all')
```

Out[5]:

	Oxygen	PulseRate	Temperature	Result
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	92.548900	84.976600	100.000700	0.499000
std	4.611197	26.305841	3.185045	0.500024
min	85.000000	40.000000	95.000000	0.000000
25%	88.000000	63.000000	97.000000	0.000000
50%	93.000000	85.000000	100.000000	0.000000
75%	97.000000	108.000000	103.000000	1.000000
max	100.000000	130.000000	105.000000	1.000000

In [6]:
```
target.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Oxygen       10000 non-null  int64
 1   PulseRate    10000 non-null  int64
 2   Temperature  10000 non-null  int64
 3   Result       10000 non-null  int64
dtypes: int64(4)
memory usage: 312.6 KB
```

In [7]:
```
def count_na(target, col):
    print(f"Null values in {col}: ", target[col].isna().sum())

for feat in target.columns:
    count_na(target, feat)
```

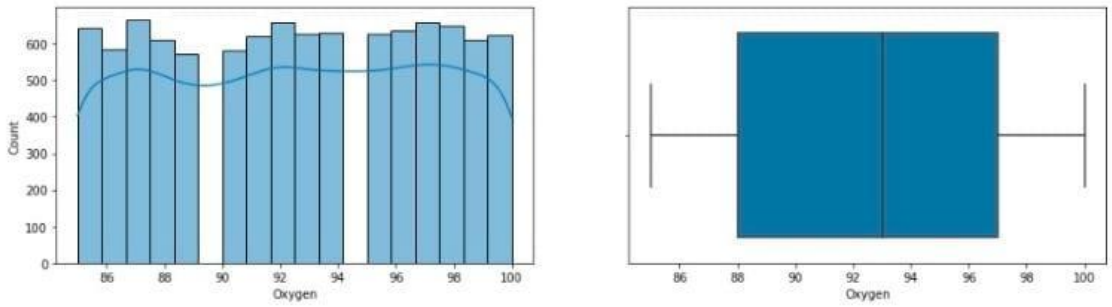
```
Null values in Oxygen:  0
Null values in PulseRate:  0
Null values in Temperature:  0
Null values in Result:  0
```

In [8]:
```
target.isnull()
```

Out[8]:

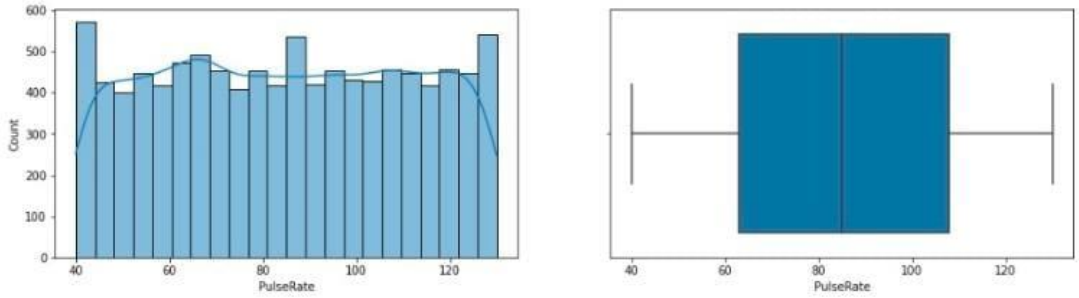
	Oxygen	PulseRate	Temperature	Result
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
...	...	...	...	...
9995	False	False	False	False
9996	False	False	False	False
9997	False	False	False	False

In [9]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="Oxygen", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="Oxygen", ax=ax1[1])
plt.show()

print("Median of Oxygen: ", target["Oxygen"].median())
print("Mean of Oxygen: ", target["Oxygen"].mean())
```



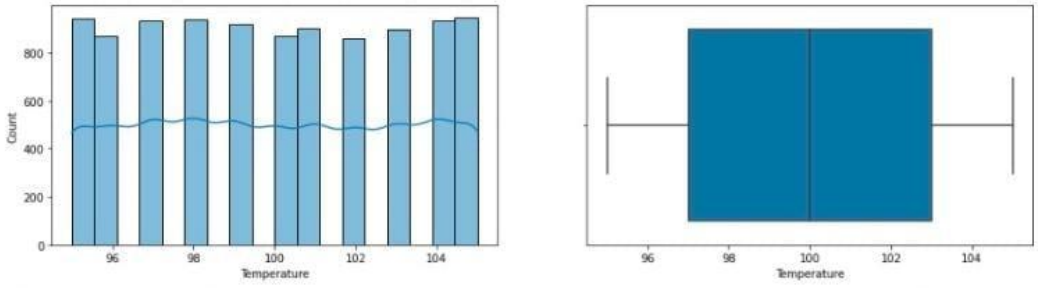

```
Median of Oxygen:  93.0
Mean of Oxygen:  92.5489
```

In [10]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="PulseRate", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="PulseRate", ax=ax1[1])
plt.show()

print("Median of Pulse Rate: ", target["PulseRate"].median())
print("Mean of Pulse Rate: ", target["PulseRate"].mean())
```



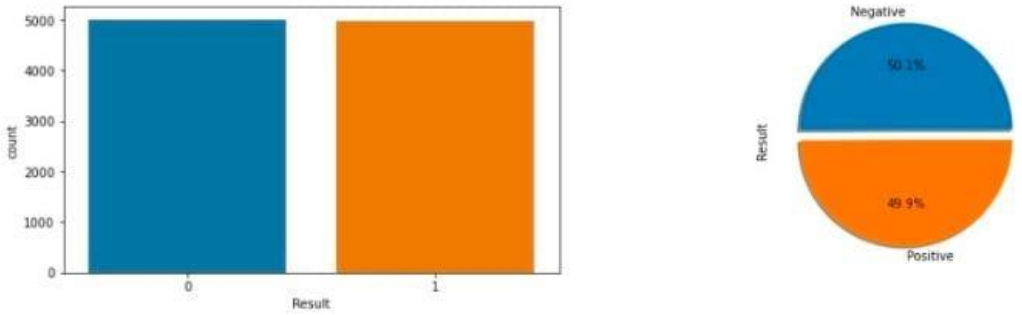

```
Median of Pulse Rate:  85.0
Mean of Pulse Rate:  84.9766
```

In [11]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="Temperature", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="Temperature", ax=ax1[1])
plt.show()

print("Median of Temperature: ", target["Temperature"].median())
print("Mean of Temperature: ", target["Temperature"].mean())
```

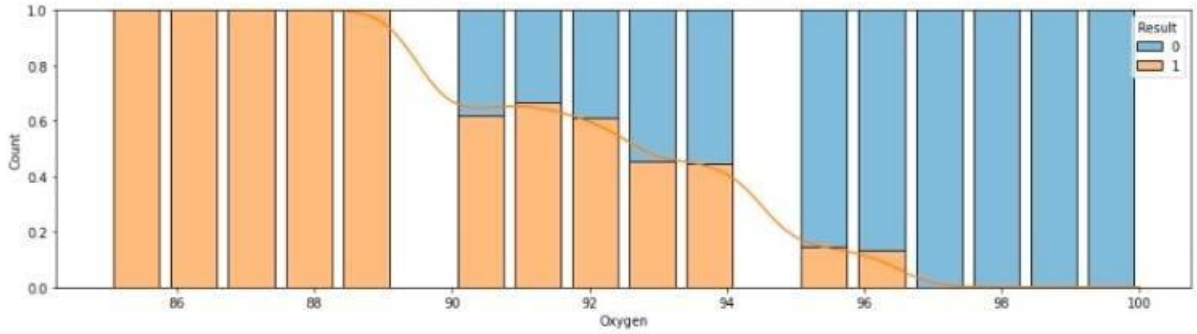



```
Median of Temperature:  100.0
Mean of Temperature:  100.0007
```

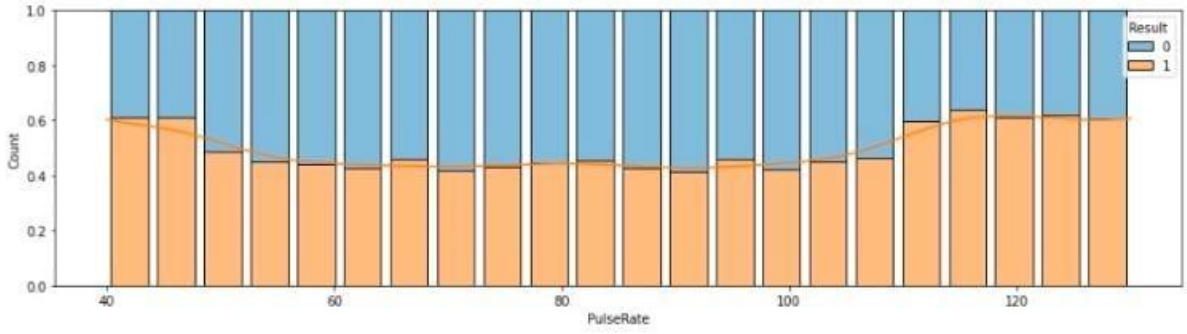
In [12]:
```
fig1, ax = plt.subplots(1, 2, figsize=(16, 4))

sns.countplot(data=target, x="Result", ax=ax[0])
target["Result"].value_counts().plot.pie(explode=[0.1, 0], autopct="%1.1f%%", labels=["Negative", "Positive"], shadow=True, ax=ax
plt.show()
```



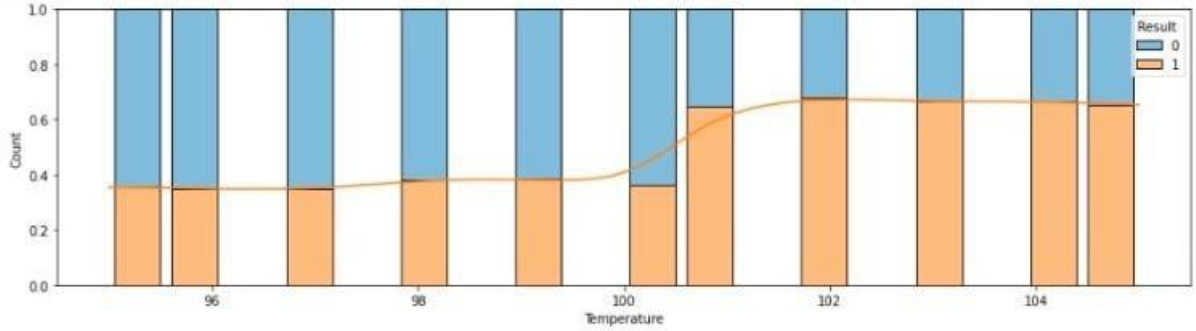
In [13]:
```
fig1, ax1 = plt.subplots(figsize=(16, 4))

sns.histplot(data=target, x="Oxygen", hue="Result", shrink=0.8, multiple="fill", kde=True, ax=ax1)
plt.show()
```



In [14]:
```
fig1, ax1 = plt.subplots(figsize=(16, 4))

sns.histplot(data=target, x="PulseRate", hue="Result", shrink=0.8, multiple="fill", kde=True, ax=ax1)
plt.show()
```



In [15]:
```
fig1, ax1 = plt.subplots(figsize=(16, 4))

sns.histplot(data=target, x="Temperature", hue="Result", shrink=0.8, multiple="fill", kde=True, ax=ax1)
plt.show()
```



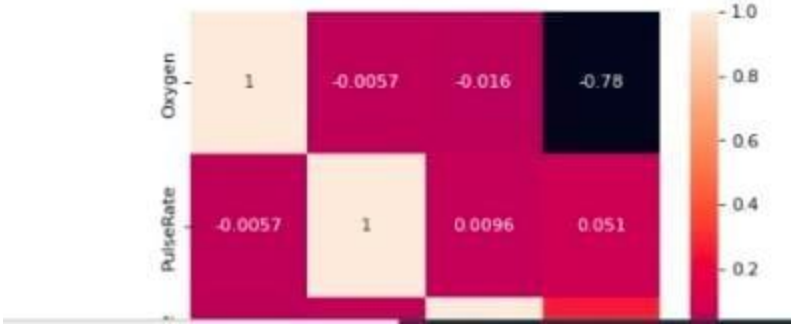
In [16]:
```
correlations = target.corr()
correlations
```

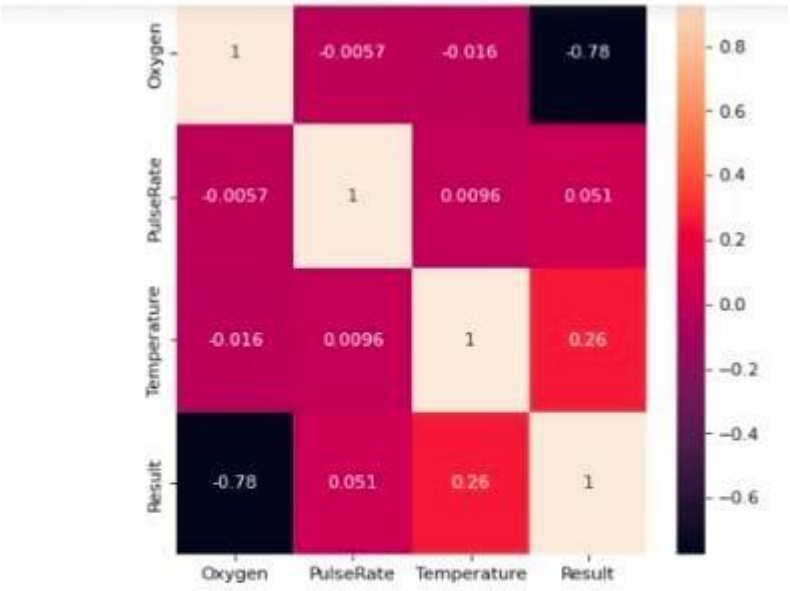
Out[16]:

	Oxygen	PulseRate	Temperature	Result
Oxygen	1.000000	-0.005724	-0.015681	-0.777019
PulseRate	-0.005724	1.000000	0.009602	0.050590
Temperature	-0.015681	0.009602	1.000000	0.262522
Result	-0.777019	0.050590	0.262522	1.000000

In [17]:
```
plt.figure(figsize=(6, 6), dpi=80)
sns.heatmap(correlations, annot=True)
```

Out[17]: <AxesSubplot:>






```
In [18]: correlations["Result"].sort_values(ascending=False)
```

```
Out[18]: Result         1.000000
         Temperature    0.262522
         PulseRate      0.050590
         Oxygen        -0.777019
         Name: Result, dtype: float64
```

```
In [19]: X = target.drop(["Result", "Oxygen"], axis = 1)
         X
```

Out[19]:

	PulseRate	Temperature
0	65	95
1	92	95
2	92	99
3	56	96
4	94	98
...	...	...
9995	124	97
9996	70	100
9997	56	105
9998	49	98
9999	52	99

10000 rows × 2 columns

```
In [20]: y = target["Result"]
         y
```

```
Out[20]: 0       0
```

```
y
```

Out[20]:
```
0       0
1       0
2       0
3       0
4       1
       ..
9995    0
9996    1
9997    0
9998    1
9999    1
Name: Result, Length: 10000, dtype: int64
```

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=69)
y_test
```

Out[21]:
```
5256    0
7272    1
1323    0
4924    1
5845    0
       ..
5827    0
7428    1
1620    1
6529    0
9630    1
Name: Result, Length: 2500, dtype: int64
```

In [22]:
```
import warnings
warnings.filterwarnings('always')  # "error", "ignore", "always", "default", "module" or "once"
```

In [23]:
```
ss = StandardScaler()

X_train = ss.fit_transform(X_train)

X_test= ss.transform(X_test)
```

In [24]:
```
import xgboost as xgb

xgb_clf = xgb.XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
xgb_clf = xgb_clf.fit(X_train, y_train)
y_pred = xgb_clf.predict(X_test)

print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')

cm = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n',cm)

print(classification_report(y_test, xgb_clf.predict(X_test)))
```

```
Accuracy: 62.12 %
Confusion Matrix:
 [[758 455]
 [492 795]]
```

	precision	recall	f1-score	support
0	0.61	0.62	0.62	1213
1	0.64	0.62	0.63	1287
accuracy			0.62	2500
macro avg	0.62	0.62	0.62	2500
weighted avg	0.62	0.62	0.62	2500

```
In [25]: xgb_clf.predict([[89, 99.23], [92, 100.02]])
Out[25]: array([1, 1])
```

```
In [26]: from sklearn import tree

decision_tree = tree.DecisionTreeClassifier(criterion='gini')
decision_tree.fit(X_train, y_train)
y_pred = decision_tree.predict(X_test)

print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')

cm = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n',cm)

print(classification_report(y_test, decision_tree.predict(X_test)))
```

```
Accuracy: 60.12 %
Confusion Matrix:
 [[804 409]
 [588 699]]
```

	precision	recall	f1-score	support
0	0.58	0.66	0.62	1213
1	0.63	0.54	0.58	1287
accuracy			0.60	2500
macro avg	0.60	0.60	0.60	2500
weighted avg	0.61	0.60	0.60	2500

```
In [27]: decision_tree.predict([[89, 99.23], [92, 100.02]])
Out[27]: array([1, 1], dtype=int64)
```

```
In [28]: from sklearn.svm import SVC

svm = SVC(kernel='rbf', random_state=23, gamma=.10, C=1.0)
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)

print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')

cm = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n',cm)

print(classification_report(y_test, svm.predict(X_test)))
```

```
Accuracy: 65.24 %
Confusion Matrix:
 [[801 412]
 [457 830]]
              precision    recall  f1-score   support

           0       0.64      0.66      0.65      1213
           1       0.67      0.64      0.66      1287

    accuracy                           0.65      2500
   macro avg       0.65      0.65      0.65      2500
weighted avg       0.65      0.65      0.65      2500
```

```
In [29]: svm.predict([[81, 98.37], [79, 97.5]])

Out[29]: array([0, 0], dtype=int64)
```

Diagnosis of diabetes dataset:

- Import the basic machine learning libraries like Pandas & Numpy (for numerical calculations) and Matplotlib & Seaborn (for data visualization).
- Read the dataset (which is in csv format) and display five rows of data.
- Drop two variables namely id and patient number from the table as they are of not much use for prediction analysis.
- Describing the detailed comparison of different measures (like mean, standard deviation, minimum values and maximum values) for the parameters present in the dataset.
- Infer the information whether the dataset parameters are of which data type (integer type, float type or any other type).
- Check for null values and missing values in the dataset as part of data cleaning. We find out that there are no null values as well as no missing values in the dataset.
- Use the Seaborn library to visualize the different parameters of the dataset and perform univariate analysis on these parameters.

- Plot a histogram plot and box plot for each of the 11 parameters namely Gender, Age, Urea, Creatinine, HbA1c, Cholesterol, Triglycerides, HDL, LDL, VLDL and BMI. Calculate the mean and median of all the above mentioned parameters. Plot a count plot and a pie plot for the class label namely 'CLASS'.
- Finding correlations between the above mentioned 11 parameters and the class label. A 12 x 12 matrix is formed showing the correlations between each parameter. All the parameters have a correlation of 1 with themselves. Each parameter (11) and class label has either a positive correlation or a negative correlation with one other.
- Positive correlation between any two parameter indicates that they are highly influenced with each other and they increase & decrease synonymously while negative correlation indicates that they are inversely influenced with each other and act opposite i.e. if one parameter increases then other one decreases and vice versa.
- Plot a correlation matrix with help of Heatmap and annotate all the values. Heatmap helps in visualizing the correlations. Sort the correlation values of all the 11 parameters and the class label. We find out that HbA1c, BMI and Age are highly positively correlated with the class label (which indicates that whether a person is non-diabetic, pre-diabetic or diabetic).
- Moving on to the prediction analysis, we drop the parameters Gender, Age, Urea, Creatinine, Cholesterol, Triglycerides, HDL, LDL & VLDL as they have too low correlation with the class label. Storing the remaining parameters in a separate variable X and storing the class label in another separate variable y. This is necessary for dividing the dataset into training data and testing data.
- Import a very important library – Sklearn (used for data analysis). This library will help in splitting the dataset into training data and testing data. We have split the dataset in the ratio of 75% : 25% i.e. 75% training data and 25% testing data.
- Applying XGBoost, Decision Tree and Random Forest. Checking accuracy of each algorithm and making predictions for the test dataset.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report
```

In [3]:
```
df = pd.read_csv("Diabetes.csv")
df.head()
```

Out[3]:

	ID	No_Pation	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS
0	502	17975	1	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	0
1	735	34221	0	26	4.5	62	4.9	3.7	1.4	1.1	2.1	0.6	23.0	0
2	420	47975	1	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	0
3	680	87656	1	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	0
4	504	34223	0	33	7.1	46	4.9	4.9	1.0	0.8	2.0	0.4	21.0	0

In [4]:
```
target = df.drop(['ID', 'No_Pation'], axis=1)
target
```

Out[4]:

In [4]:
```
target = df.drop(['ID', 'No_Pation'], axis=1)
target
```

Out[4]:

	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS
0	1	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	0
1	0	26	4.5	62	4.9	3.7	1.4	1.1	2.1	0.6	23.0	0
2	1	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	0
3	1	50	4.7	46	4.9	4.2	0.9	2.4	1.4	0.5	24.0	0
4	0	33	7.1	46	4.9	4.9	1.0	0.8	2.0	0.4	21.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	30	17.1	344	6.0	4.9	1.6	1.7	2.5	0.7	21.0	2
996	0	54	4.0	88	5.7	4.4	2.9	0.6	2.5	1.3	28.0	2
997	0	30	6.0	97	5.8	4.2	1.7	1.2	2.2	0.8	19.0	2
998	0	39	5.0	106	6.4	3.7	2.0	0.8	2.1	0.9	19.5	2
999	1	48	5.6	79	6.3	4.2	1.2	2.5	2.7	1.4	27.0	2

1000 rows × 12 columns

In [5]:
```
target.describe(include='all')
```

Out[5]:

	Gender	AGE	Urea	Cr	HbA1c	Chol

In [5]:
```
target.describe(include='all')
```

Out[5]:

	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	C
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.0
mean	0.435000	53.528000	5.124743	68.943000	8.281160	4.862820	2.349610	1.204750	2.609790	1.854700	29.578020	0.9
std	0.496005	8.799241	2.935165	59.984747	2.534003	1.301738	1.401176	0.660414	1.115102	3.663599	4.962388	0.3
min	0.000000	20.000000	0.500000	6.000000	0.900000	0.000000	0.300000	0.200000	0.300000	0.100000	19.000000	0.0
25%	0.000000	51.000000	3.700000	48.000000	6.500000	4.000000	1.500000	0.900000	1.800000	0.700000	26.000000	1.0
50%	0.000000	55.000000	4.600000	60.000000	8.000000	4.800000	2.000000	1.100000	2.500000	0.900000	30.000000	1.0
75%	1.000000	59.000000	5.700000	73.000000	10.200000	5.600000	2.900000	1.300000	3.300000	1.500000	33.000000	1.0
max	1.000000	79.000000	38.900000	800.000000	16.000000	10.300000	13.800000	9.900000	9.900000	35.000000	47.750000	2.0

In [6]:
```
target.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Gender  1000 non-null   int64
 1   AGE     1000 non-null   int64
 2   Urea    1000 non-null   float64
 3   Cr      1000 non-null   int64
 4   HbA1c   1000 non-null   float64
 5   Chol    1000 non-null   float64
 6   TG      1000 non-null   float64
 7   HDL     1000 non-null   float64
 8   LDL     1000 non-null   float64
 9   VLDL    1000 non-null   float64
 10  BMI     1000 non-null   float64
 11  CLASS   1000 non-null   int64
dtypes: float64(8), int64(4)
memory usage: 93.9 KB
```

In [7]:
```
def count_na(target, col):
    print(f"Null values in {col}: ", target[col].isna().sum())

for feat in target.columns:
    count_na(target, feat)
```

```
Null values in Gender:  0
Null values in AGE:  0
Null values in Urea:  0
Null values in Cr:  0
Null values in HbA1c:  0
Null values in Chol:  0
Null values in TG:  0
Null values in HDL:  0
Null values in LDL:  0
Null values in VLDL:  0
Null values in BMI:  0
Null values in CLASS:  0
```

In [8]:
```
target.isnull()
```

Out[8]:

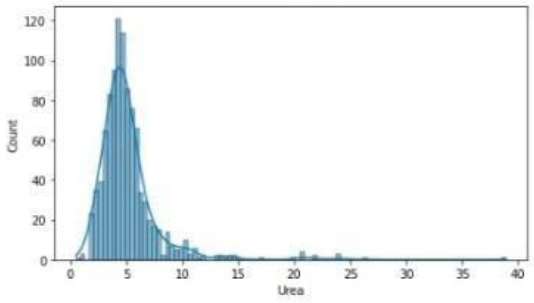
	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS
0	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False	False	False	False	False

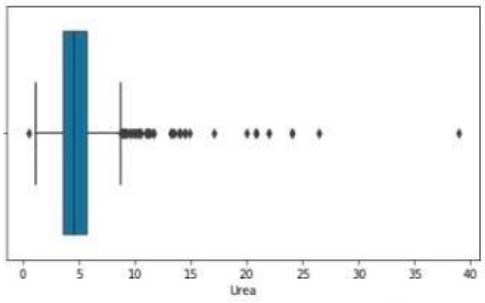
1000 rows × 12 columns

In [9]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="Urea", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="Urea", ax=ax1[1])
plt.show()
```

In [9]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="Urea", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="Urea", ax=ax1[1])
plt.show()
```

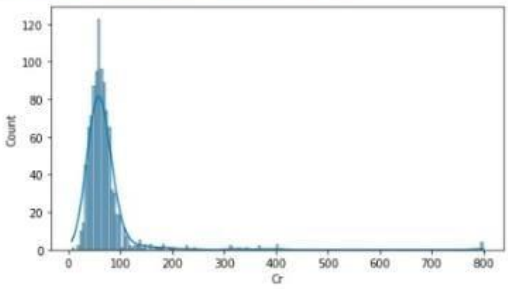
```
print("Median of Urea: ", target["Urea"].median())
print("Mean of Urea: ", target["Urea"].mean())
```

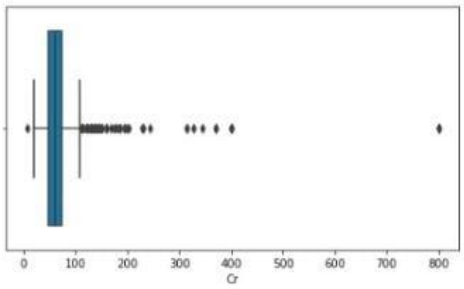





```
Median of Urea:  4.6
Mean of Urea:  5.124743000000003
```

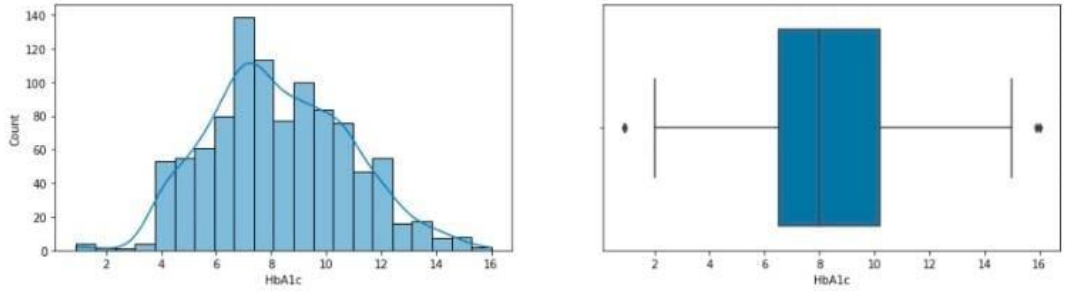
In [10]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="Cr", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="Cr", ax=ax1[1])
plt.show()

print("Median of Creatinine: ", target["Cr"].median())
print("Mean of Creatinine: ", target["Cr"].mean())
```





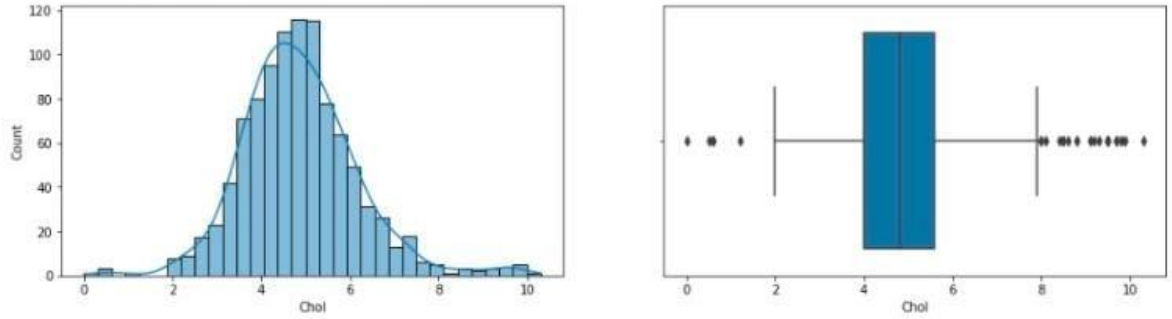

```
Median of Creatinine:  60.0
Mean of Creatinine:  68.943
```

In [11]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="HbA1c", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="HbA1c", ax=ax1[1])
plt.show()

print("Median of HbA1c: ", target["HbA1c"].median())
print("Mean of HbA1c: ", target["HbA1c"].mean())
```



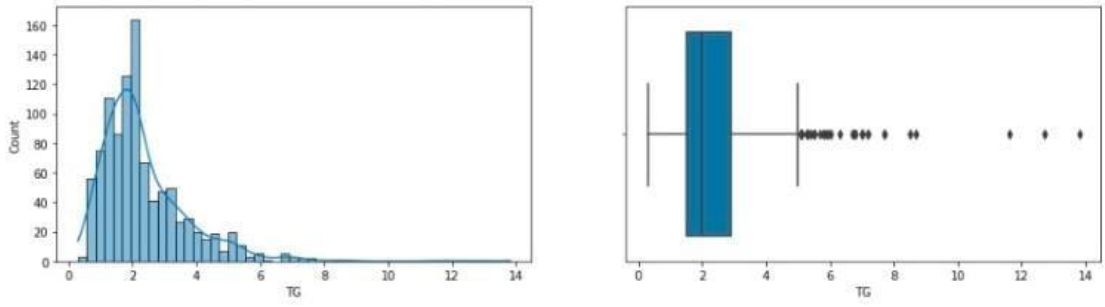

```
Median of HbA1c:  8.0
Mean of HbA1c:  8.281159999999995
```

In [12]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="Chol", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="Chol", ax=ax1[1])
plt.show()

print("Median of Cholesterol: ", target["Chol"].median())
print("Mean of Cholesterol: ", target["Chol"].mean())
```



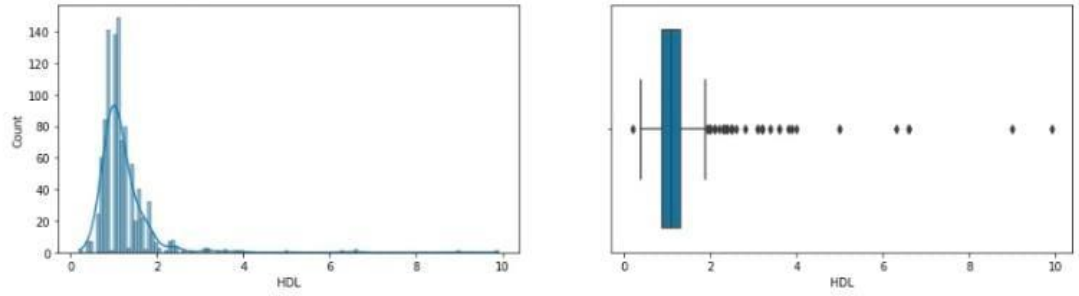

```
Median of Cholesterol:  4.8
Mean of Cholesterol:  4.862819999999998
```

In [13]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="TG", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="TG", ax=ax1[1])
plt.show()

print("Median of Triglycerides: ", target["TG"].median())
print("Mean of Triglycerides: ", target["TG"].mean())
```



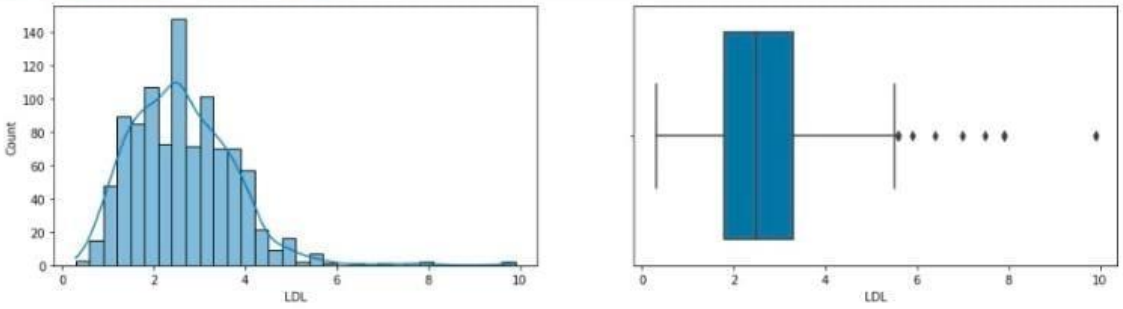

```
Median of Triglycerides:  2.0
Mean of Triglycerides:  2.3496099999999993
```

In [14]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="HDL", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="HDL", ax=ax1[1])
plt.show()

print("Median of HDL: ", target["HDL"].median())
print("Mean of HDL: ", target["HDL"].mean())
```



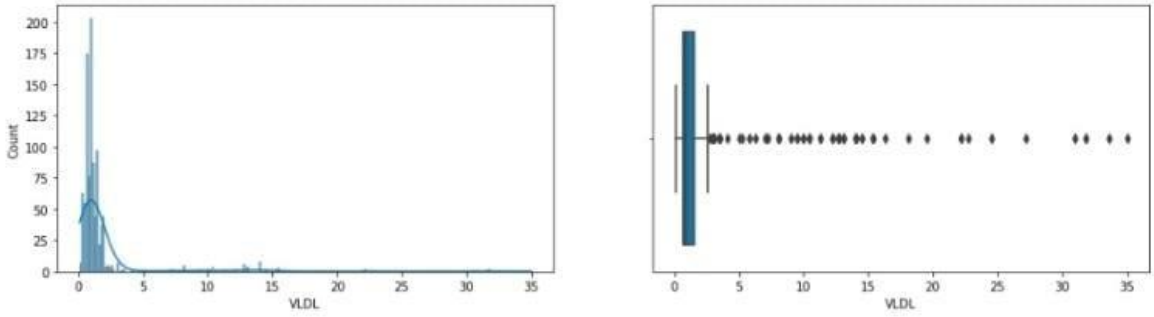

```
Median of HDL:  1.1
Mean of HDL:  1.2047499999999993
```

In [15]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="LDL", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="LDL", ax=ax1[1])
plt.show()

print("Median of LDL: ", target["LDL"].median())
print("Mean of LDL: ", target["LDL"].mean())
```



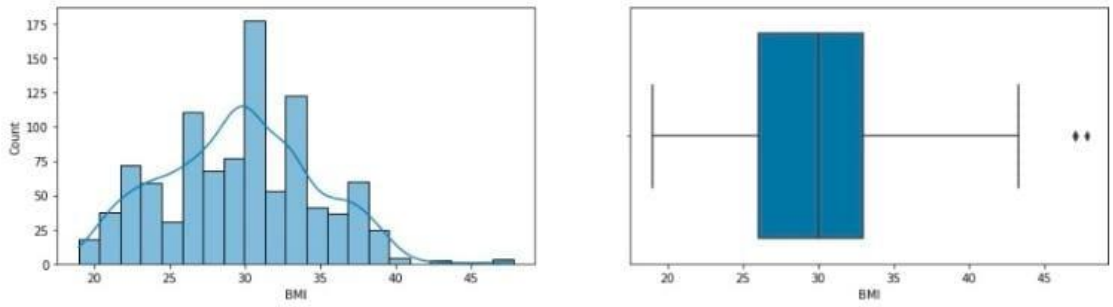

```
Median of LDL:  2.5
Mean of LDL:  2.609789999999997
```

In [16]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="VLDL", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="VLDL", ax=ax1[1])
plt.show()

print("Median of VLDL: ", target["VLDL"].median())
print("Mean of VLDL: ", target["VLDL"].mean())
```



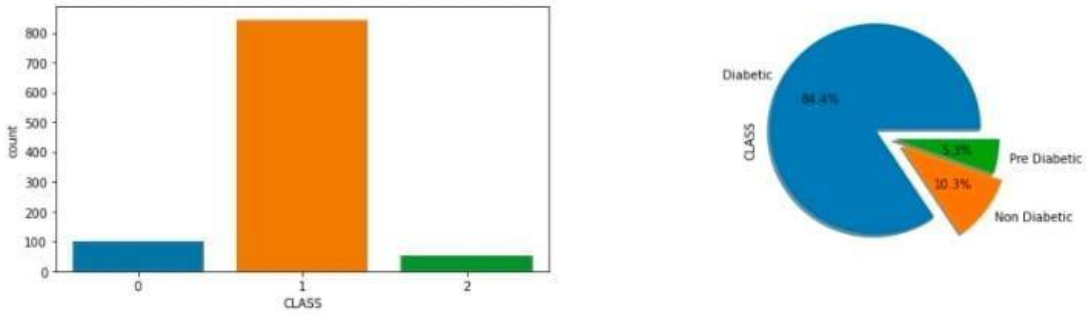

```
Median of VLDL:  0.9
Mean of VLDL:  1.8547000000000018
```

In [17]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="BMI", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="BMI", ax=ax1[1])
plt.show()

print("Median of BMI: ", target["BMI"].median())
print("Mean of BMI: ", target["BMI"].mean())
```

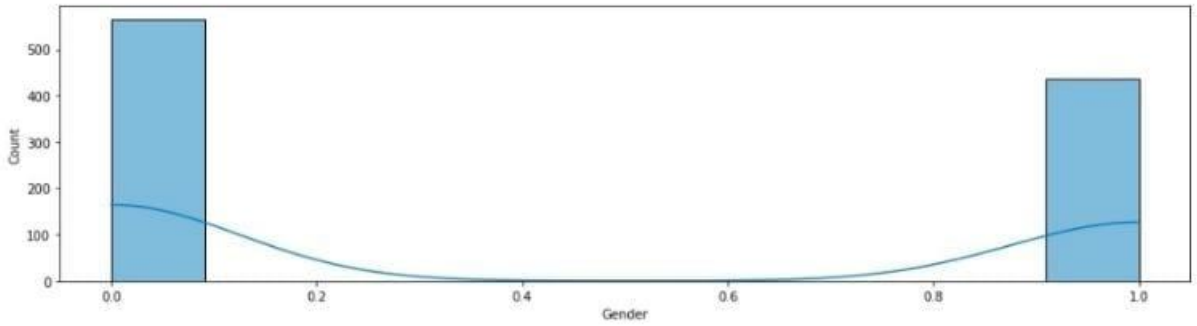



```
Median of BMI:  30.0
Mean of BMI:  29.57801999999999
```

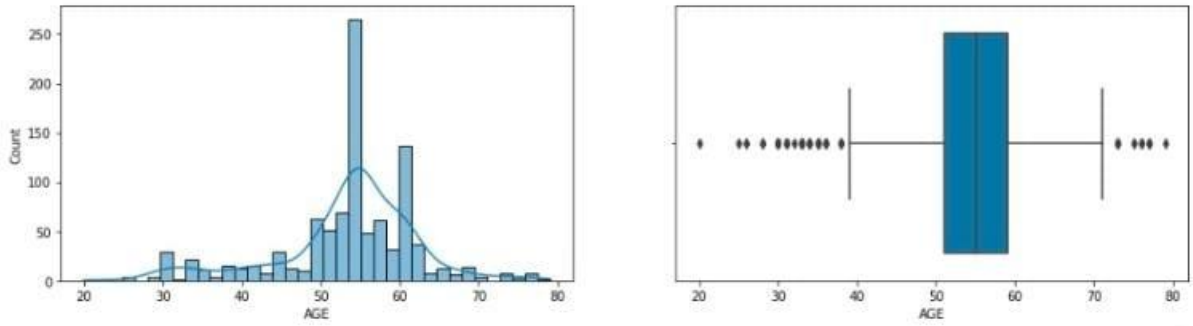
In [18]:
```
fig1, ax = plt.subplots(1, 2, figsize=(16, 4))

sns.countplot(data=target, x="CLASS", ax=ax[0])
target["CLASS"].value_counts().plot.pie(explode=[0.2, 0.1, 0], autopct="%1.1f%%", labels=["Diabetic", "Non Diabetic", "Pre Diabet
plt.show()
```



In [20]:
```
fig1, ax1 = plt.subplots(figsize=(16, 4))

sns.histplot(data=target, x="Gender", kde=True)
plt.show()
```



In [19]:
```
fig1, ax1 = plt.subplots(1, 2, figsize=(16, 4))

sns.histplot(data=target, x="AGE", kde=True, ax=ax1[0])
sns.boxplot(data=target, x="AGE", ax=ax1[1])
plt.show()

print("Median of Age: ", target["AGE"].median())
print("Mean of Age: ", target["AGE"].mean())
```




```
Median of Age:  55.0
Mean of Age:  53.528
```

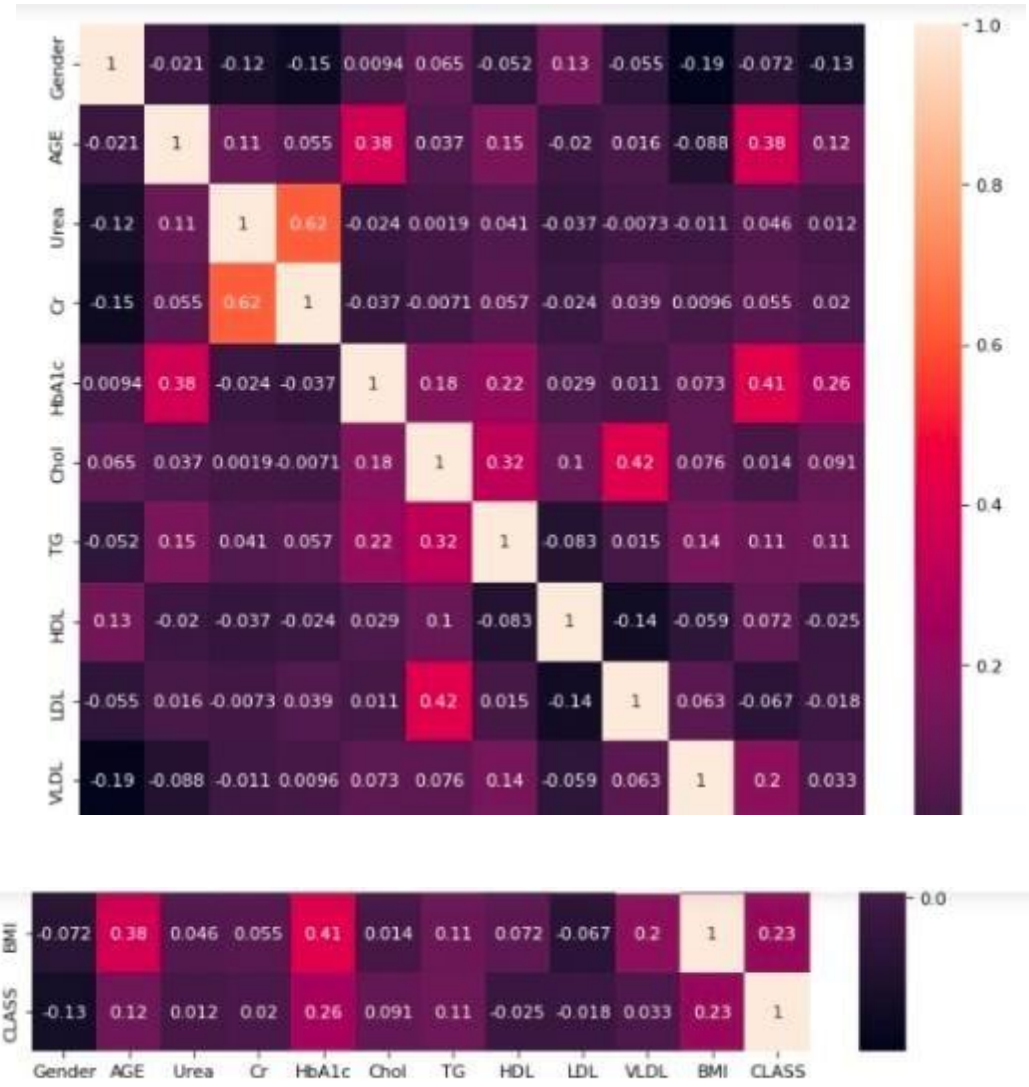
In [21]:
```
correlations = target.corr()
correlations
```

Out[21]:

	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS
Gender	1.000000	-0.021486	-0.116311	-0.154870	0.009362	0.064763	-0.052111	0.130130	-0.054563	-0.194120	-0.072097	-0.129998
AGE	-0.021486	1.000000	0.105092	0.054941	0.379136	0.036649	0.148204	-0.020038	0.016105	-0.087903	0.375956	0.120265
Urea	-0.116311	0.105092	1.000000	0.624134	-0.023603	0.001852	0.040980	-0.036994	-0.007301	-0.011191	0.045618	0.011517
Cr	-0.154870	0.054941	0.624134	1.000000	-0.037412	-0.007097	0.056579	-0.023804	0.039479	0.009615	0.054746	0.020483
HbA1c	0.009362	0.379136	-0.023603	-0.037412	1.000000	0.177489	0.218556	0.028933	0.011057	0.073462	0.413350	0.264595
Chol	0.064763	0.036649	0.001852	-0.007097	0.177489	1.000000	0.321789	0.103814	0.416665	0.076294	0.013678	0.090516
TG	-0.052111	0.148204	0.040980	0.056579	0.218556	0.321789	1.000000	-0.083001	0.015378	0.144570	0.110757	0.114418
HDL	0.130130	-0.020038	-0.036994	-0.023804	0.028933	0.103814	-0.083001	1.000000	-0.142079	-0.059275	0.072409	-0.024796
LDL	-0.054563	0.016105	-0.007301	0.039479	0.011057	0.416665	0.015378	-0.142079	1.000000	0.062795	-0.067322	-0.017772
VLDL	-0.194120	-0.087903	-0.011191	0.009615	0.073462	0.076294	0.144570	-0.059275	0.062795	1.000000	0.198133	0.033203
BMI	-0.072097	0.375956	0.045618	0.054746	0.413350	0.013678	0.110757	0.072409	-0.067322	0.198133	1.000000	0.227867
CLASS	-0.129998	0.120265	0.011517	0.020483	0.264595	0.090516	0.114418	-0.024796	-0.017772	0.033203	0.227867	1.000000

In [22]:
```
plt.figure(figsize=(10, 10), dpi=80)
sns.heatmap(correlations, annot=True)
```

Out[22]: <AxesSubplot:>



In [23]:
```
correlations["CLASS"].sort_values(ascending=False)
```

Out[23]:
```
CLASS      1.000000
HbA1c      0.264595
BMI        0.227867
AGE        0.120265
TG         0.114418
Chol       0.090516
VLDL       0.033203
Cr         0.020483
Urea       0.011517
LDL       -0.017772
HDL       -0.024796
Gender    -0.129998
Name: CLASS, dtype: float64
```

In [24]:
```
X = target.drop(["CLASS", "Gender", "Urea", "Cr", "Chol", "TG", "HDL", "LDL", "VLDL"], axis = 1)
X
```

Out[24]:

	AGE	HbA1c	BMI

	AGE	HbA1c	BMI
0	50	4.9	24.0
1	26	4.9	23.0
2	50	4.9	24.0
3	50	4.9	24.0
4	33	4.9	21.0
...	...	...	...
995	30	6.0	21.0
996	54	5.7	28.0
997	30	5.8	19.0
998	39	6.4	19.5
999	48	6.3	27.0

1000 rows × 3 columns

In [25]:
```
y = target["CLASS"]
y
```

Out[25]:
```
0      0
1      0
2      0
3      0
4      0
      ..
995    2
996    2
997    2
998    2
999    2
Name: CLASS, Length: 1000, dtype: int64
```

In [26]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=69)
y_test
```

Out[26]:
```
823    1
999    2
891    1
781    1
967    2
      ..
847    1
741    1
734    1
134    1
441    1
Name: CLASS, Length: 250, dtype: int64
```

In [27]:
```
ss = StandardScaler()

X_train = ss.fit_transform(X_train)

X_test= ss.transform(X_test)
```

In [28]:
```
import xgboost as xgb

xgb_clf = xgb.XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
xgb_clf = xgb_clf.fit(X_train, y_train)
y_pred = xgb_clf.predict(X_test)

print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')

cm = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n',cm)

print(classification_report(y_test, xgb_clf.predict(X_test)))
```

```
C:\Users\Kishan\anaconda3\lib\site-packages\xgboost\compat.py:36: FutureWarnin
moved from pandas in a future version. Use pandas.Index with the appropriate d
  from pandas import MultiIndex, Int64Index
```

```
Accuracy: 97.2 %
Confusion Matrix:
 [[ 25   2   0]
 [  5 203   0]
 [  0   0  15]]
              precision    recall  f1-score   support

           0       0.83      0.93      0.88        27
           1       0.99      0.98      0.98       208
           2       1.00      1.00      1.00        15

    accuracy                           0.97       250
```

```
 [  5 203   0]
 [  0   0  15]]
              precision    recall  f1-score   support

           0       0.83      0.93      0.88        27
           1       0.99      0.98      0.98       208
           2       1.00      1.00      1.00        15

    accuracy                           0.97       250
   macro avg       0.94      0.97      0.95       250
weighted avg       0.97      0.97      0.97       250
```

In [29]:
```
xgb_clf.predict([[45, 5.8, 24], [51, 6.7, 22]])
```

Out[29]:
```
array([1, 1], dtype=int64)
```

In [30]:
```
from sklearn import tree

decision_tree = tree.DecisionTreeClassifier(criterion='gini')
decision_tree.fit(X_train, y_train)
y_pred = decision_tree.predict(X_test)

print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')

cm = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n',cm)

print(classification_report(y_test, decision_tree.predict(X_test)))
```

```
Accuracy: 96.0 %
Confusion Matrix:
 [[ 23   4   0]
 [  5 203   0]
 [  0   1  14]]
              precision    recall  f1-score   support

           0       0.82      0.85      0.84        27
           1       0.98      0.98      0.98       208
           2       1.00      0.93      0.97        15

    accuracy                           0.96       250
   macro avg       0.93      0.92      0.93       250
weighted avg       0.96      0.96      0.96       250
```

In [31]:
```
decision_tree.predict([[45, 5.8, 24], [51, 6.7, 22]])
```

Out[31]:
```
array([1, 1], dtype=int64)
```

In [32]:
```
from sklearn.ensemble import RandomForestClassifier

random_forest = RandomForestClassifier()
random_forest.fit(X_train, y_train)
y_pred = random_forest.predict(X_test)

print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')
```

*

```
random_forest = RandomForestClassifier()
random_forest.fit(X_train, y_train)
y_pred = random_forest.predict(X_test)

print(f'Accuracy:',accuracy_score(y_test, y_pred)* 100 ,'%')

cm = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n',cm)

print(classification_report(y_test, random_forest.predict(X_test)))
```

```
Accuracy: 96.39999999999999 %
Confusion Matrix:
 [[ 24   3   0]
 [  5 203   0]
 [  0   1  14]]
              precision    recall  f1-score   support

           0       0.83      0.89      0.86        27
           1       0.98      0.98      0.98       208
           2       1.00      0.93      0.97        15

    accuracy                           0.96       250
   macro avg       0.94      0.93      0.93       250
weighted avg       0.97      0.96      0.96       250
```

In [33]:
```
random_forest.predict([[45, 5.8, 24], [51, 6.7, 22]])
```

Out[33]:
```
array([1, 1], dtype=int64)
```

4.3. Risk Analysis and Mitigation

A firm may use risk mitigation as a technique to prepare for and decrease the impact of potentia l hazards. Risk mitigation, like risk reduction, involves taking measures to lessen the detrimenta l consequences of risks and disasters on company continuity (BC). Cyberattacks, weather-related disasters, and other potential sources of real or imagined harm are examples of threats th at could endanger a business. One aspect of risk management is risk mitigation, and each organ isation will apply it differently

5. Testing

5.1. Testing Plan

An automated tool is not used during the manual testing process, which involves manually running test cases. According to the viewpoint of the end user, the tester manually completed all the test cases. It confirms whether or not the application functions as specified in the requirement document. Planning and execution of test cases allows for nearly complete software application completion. Additionally, test case reports are manually created.

One of the most essential testing techniques is manual testing since it can identify both obvious and subtle software flaws. A fault is defined as the discrepancy between the output that was anticipated and the output that the software produced. After the developer repaired the flaws, the tester received the product to retest.

Before any freshly developed software is subjected to automated testing, manual testing is required. Although it takes a lot of time and work, this testing ensures that the code is bug-free. There are various methods used for manual testing. Each technique is used according to its testing criteria. Types of manual testing:

- White Box Testing
- Black Box Testing
- Gray Box Testing

White Box Testing: In contrast to evaluating an application's functionality, it examines an application's internal mechanisms. It is also referred to as clear box testing, glass box testing, transparent box testing. In white-box testing, test cases are created from the system's internal viewpoint. The tester selects inputs to test several code routes and establish the anticipated results. Similar to evaluating the nodes in a circuit, such as by in-circuit testing (ICT). The unit, integration, and system levels of the software testing process can all use white-box testing.

White-box testing is now more widely utilized for integration and system testing, despite the fact that previous testers tended to think of it as being performed at the unit level. It may test the connections between subsystems during a system-level test, as well as the connections between units during integration. Although this approach to test design can find a lot of faults or issues, it runs the risk of missing requirements or sections of the specification that haven't been implemented. White-box testing is design-driven, which means it may analyze if requirements have been implemented or are missing.

Black Box Testing: Software testing that assesses an application's functioning without looking at its internal components. Practically every level of software testing, including unit, integration, system, and acceptance testing, may be conducted using this test methodology.

It is not necessary to have detailed understanding of the application's code, internal structure, or general programming knowledge. The tester is aware of what the software is meant to do, but not how it actually accomplishes that. For example, the tester may be aware that a specific input leads to a specific, deterministic outcome but may not be aware of how the software generates the output in the first place.

Specifications and requirements, or what the application is supposed to do, are the foundation upon which test cases are created. Typically, test cases are produced from external descriptions of the software, such as requirements, specifications, and design criteria. Non-functional tests may also be utilized, despite the fact that functional tests represent the majority of the tests used.

Without being aware of the internal structure of the test object, the test designer chooses both valid and invalid inputs and decides the correct output, frequently with the aid of a test oracle or a prior result that was known to be good.

Gray Box Testing: White-box testing and black-box testing are combined in gray-box testing. The purpose of this testing is to look for any flaws caused by inappropriate application usage or structure.

5.2. Component Decomposition and Type of Testing Require

Our project consists of testing of different modules through manual testing consisting of White Box Testing, Black Box Testing and Gray Box Testing. Component decomposition and the type of testing applied is stated as follows:

For the monitoring component, we are using the White box technique. The testing plan includes the verification of the values inherited from the monitoring systems. The values taken from the monitoring systems are cross checked from the prescribed ranges in the medical science through manual cross checking and still if some values are left then they are filtered out by the use of complex if/else statements.

For the diagnosis component we are using black-box testing. The predefined datasets are taken from the kaggle and then the values of the attributes in datasets are cross checked manually from the prescribed ranges in the medical science. When there were any outliers or irrelevant values present in the dataset then those values were filled up or corrected or deleted in data pre-processing and replenished through basic statistical techniques such as mean, mode, and median.

The Implementation component consists of the gray-box testing, where the code to execute are checked line by line consisting of multiple functions where the desired output is known and cross-checked.

5.3. List All Test Cases in Prescribed Format

Pulse and Temperature:

```
Accuracy: 62.12 %
Confusion Matrix:
 [[758 455]
 [492 795]]
              precision    recall  f1-score   support

           0       0.61      0.62      0.62      1213
           1       0.64      0.62      0.63      1287

    accuracy                           0.62      2500
   macro avg       0.62      0.62      0.62      2500
weighted avg       0.62      0.62      0.62      2500
```

```
xgb_clf.predict([[89, 99.23], [92, 100.02]])
```

```
array([1, 1])
```

Figure 8: XGBoost

```
Accuracy: 60.12 %
Confusion Matrix:
 [[804 409]
 [588 699]]
              precision    recall  f1-score   support

           0       0.58      0.66      0.62      1213
           1       0.63      0.54      0.58      1287

    accuracy                           0.60      2500
   macro avg       0.60      0.60      0.60      2500
weighted avg       0.61      0.60      0.60      2500
```

```
decision_tree.predict([[89, 99.23], [92, 100.02]])
```

```
array([1, 1], dtype=int64)
```

Figure 6: Decision Tree

```
Accuracy: 65.24 %
Confusion Matrix:
 [[801 412]
 [457 830]]
              precision    recall  f1-score   support

           0       0.64      0.66      0.65      1213
           1       0.67      0.64      0.66      1287

    accuracy                           0.65      2500
   macro avg       0.65      0.65      0.65      2500
weighted avg       0.65      0.65      0.65      2500
```

```
svm.predict([[81, 98.37], [79, 97.5]])
```

```
array([0, 0], dtype=int64)
```

Figure 10: Support Vector Machine

Diabetes:

```
Accuracy: 97.2 %
Confusion Matrix:
 [[ 25   2   0]
 [  5 203   0]
 [  0   0  15]]
              precision    recall  f1-score   support

           0       0.83      0.93      0.88        27
           1       0.99      0.98      0.98       208
           2       1.00      1.00      1.00        15

    accuracy                           0.97       250
   macro avg       0.94      0.97      0.95       250
weighted avg       0.97      0.97      0.97       250
```

```
xgb_clf.predict([[45, 5.8, 24], [51, 6.7, 22]])
```

```
array([1, 1], dtype=int64)
```

Figure 11: XGBoost for Diabetes

```
Accuracy: 96.0 %
Confusion Matrix:
 [[ 23   4   0]
 [  5 203   0]
 [  0   1  14]]
              precision    recall  f1-score   support

           0       0.82      0.85      0.84        27
           1       0.98      0.98      0.98       208
           2       1.00      0.93      0.97        15

    accuracy                           0.96       250
   macro avg       0.93      0.92      0.93       250
weighted avg       0.96      0.96      0.96       250
```

```
decision_tree.predict([[45, 5.8, 24], [51, 6.7, 22]])
```

```
array([1, 1], dtype=int64)
```

Figure 7: Decision Tree for Diabetes

```
Accuracy: 96.39999999999999 %
Confusion Matrix:
 [[ 24   3   0]
 [  5 203   0]
 [  0   1  14]]
              precision    recall  f1-score   support

           0       0.83      0.89      0.86        27
           1       0.98      0.98      0.98       208
           2       1.00      0.93      0.97        15

    accuracy                           0.96       250
   macro avg       0.94      0.93      0.93       250
weighted avg       0.97      0.96      0.96       250
```

```
random_forest.predict([[45, 5.8, 24], [51, 6.7, 22]])
```

```
array([1, 1], dtype=int64)
```

Figure 13: Random Forest for Diabetes

5.4. Error and Exception Handling

Errors typically happen at runtime and are of an unchecked type. Exceptions are issues that can arise during compile and runtime. It primarily appears in developer-written code. Checked exceptions and unchecked exceptions are two categories into which exceptions can be classified.

Since there was no automated handling of the data, there were no arising of the errors or the need to handle the errors and exceptions in our project.

5.5. Limitations of the Solution

The hardware limitations include the time taken by the sensors to give the accurate values as the sensor takes few seconds to stabilize the values. Also, the heart rate and spo_2 sensor (MAX30100) has incorrect internal connections.

6. Findings, Conclusion and Future Work

6.1. Findings

Pulse and Temperature Dataset – SVM performed best with 65.24% accuracy while XGBoost had an accuracy of 62.12% and Decision Tree had an accuracy of 60.12%.

Diabetes Dataset – XGBoost performed best with 97.2% accuracy while Random Forest had an accuracy of 96.39% and Decision Tree had an accuracy of 96%.

6.2. Conclusion

(a) The technology of automated data collection through quantum sensors and processing that data through a novel automation pipeline to produce the most effective outcomes will radically transform healthcare to let everyone enjoy personalized healthcare services in time.

(b) Since user centric automation of the healthcare system will facilitate 24x7 online monitoring, automated first aid, on demand diagnosis, personalized prescription and production; the cost of healthcare will reduce drastically. The patients will need to use some money only when they need intensive care at home or intensive critical care at the health center. This will force the hospitals, laboratories and pharmaceutical enterprises to reinvent themselves.

(c) Since the development of disease will be notified immediately, nobody will need heavy medicines. Even if some mild disease develops then personalized nutrition, lifestyle, nature care etc. will overcome it and will keep the person healthy. The focus will shift from curing diseases to enhancing meta human capabilities and designing superior human structures.

(d) The visual observation of cellular processes will encourage the experts from various therapies to come up with the most synergistic combination of medicines to improve the quality of healthcare and save the dying person.

(e) New initiatives to bring the dead back to life will open new understanding of the most fundamental questions like origin of life, existence of soul, life after death etc.

6.3. Future Work

(a) Understanding Consciousness for Holistic Healthcare

Competing health care systems like Ayurveda, Homeopathy, Allopathy etc are grounded in a biased understanding of mind and consciousness. We need a holistic understanding of mind and consciousness to synthesize the conceptual foundations of all the health care systems and produce computable model for personalized healthcare.

(b) Data Driven Dynamic Optimization of Hardware Resources for Automated Generation of Quantum Algorithm

User centric automation is possible only when there is no need of human intervention from data collection to the end of process. This requires that the machine itself collect data, clean data, identify features, select optimized hardware, generate quantum algorithm, and produce the most accurate results. Automated machine learning techniques are succeeding to achieve much of it but research is needed to understand 'data driven dynamic optimization of hardware resources for automated generation of quantum algorithm.' This means that the machine itself will generate the most accurate quantum code according to the requirements of the input data to be processed.

(c) Wearable Ring As Input-Output Device for Every Online Facility As A Service

Since quantum sensors are needed to collect data inside the cell, a new type of wearable ring as input-output device will do the work of transmitting the data collected by the quantum sensors to the automation pipeline for the most optimum outcomes. This means that all the hardware will be at one place to process data from billions of rings while the role of the ring will be just for connecting the quantum states for communication and for controlling the instructions by the user.

All this requires deep research for next 5-10 years.

References

[1]. Chao Yu, Jiming Liu, Fellow, IEEE, and Shamim Nemati, "**Reinforcement Learning in Healthcare: A Survey**".

[2]. Raihan Ur Rasool, Hafiz Farooq Ahmad, Wajid Rafiq and Adnan Qayyum, "**Quantum Computing for Healthcare: A Review**".

[3]. Dmitry Solenov, Jay Brieler and Jeffrey F. Scherrer, "**The Potential of Quantum Computing and Machine Learning to Advance Clinical Research and Change the Practice of Medicine**".

[4]. Saiteja Prasad Chatrati, Gahangir Hossain, Ayush Goyal, Anupama Bhan, Sayantan Bhattacharya, Devottam Gaurav and Sanju Mishra Tiwari, "**Smart home health monitoring system for predicting type 2 diabetes and hypertension**".

[5]. Athina Grammatikopoulou, Nikos Grammalidis, Maria Papadogiorgaki and Michalis Zervakis**,** "**A platform for health emergency warning and wandering behaviour detection supporting people with Intellectual Disability**".

[6]. S. Veena and D. John Aravindhar**,** "**Remote Monitoring System for the Detection of Prenatal Risk in a Pregnant Woman**".

[7]. Forum Desai, Deepraj Chowdhury, Rupinder Kaur, Marloes Peeters, Rajesh Chand Arya, Gurpreet Singh Wander, Sukhpal Singh Gill and Rajkumar Buyya, "**HealthCloud: A system for monitoring health status of heart patients using machine learning and cloud computing**".

[8]. Azmawati Mohammed Nawi, Puteri Sofia Nadira Megat Kamaruddin, Nor Rumaizah Mohd Nordin, Sharifah Saffinas Syed Soffian and Mazni Baharom, **"Machine Learning Models in Prediabetes Screening: A Systematic Review"**.

[9]. Annette Boles, Ramesh Kandimalla and P. Hemachandra Reddy, "**Dynamics of diabetes and obesity: Epidemiological perspective"**.

[10]. SA Bamanikar, AA Bamanikar and A Arora, **"Study of Serum urea and Creatinine in Diabetic and non-diabetic patients in in a tertiary teaching hospital"**.

[11]. Vanessa Alcalá-Rmz, Carlos E. Galván-Tejada et al., **"Identification of People with Diabetes Treatment through Lipids Profile Using Machine Learning Algorithms"**.

[12]. Bianchini Devis, Antonellis De Valeria, Franceschi De Nicola and Melchiori Michele, "**PREFer: a Prescription-based Food recommender system**".

[13]. MD Spector Paul, "**The Personalized Diet Prescription: New science ends era of universal dietary guidelines**"**.**

[14]. Thongyoo Phupat, Anantapanya Phuttipong, Jamsri Pornsuree and Chotipant Supannada, "**A Personalized Food Recommendation Chatbot System for Diabetes Patients**".